facebook Artificial Intelligence

Learning Invariant Representations for Reinforcement Learning without Reconstruction

Amy Zhang*[1,2], Rowan McAllister*[3], Roberto Calandra[1], Yarin Gal[4], Sergey Levine[3]

* Equal contribution. [1]Facebook AI Research, [2]McGill University, [3]UC Berkeley, [4]Oxford University

Introduction

Objective

We study how representation learning can accelerate reinforcement learning from rich observations, such as images, without relying either on domain knowledge or pixel-reconstruction. Our goal is to learn representations that both provide for effective downstream control and invariance to task-irrelevant details.

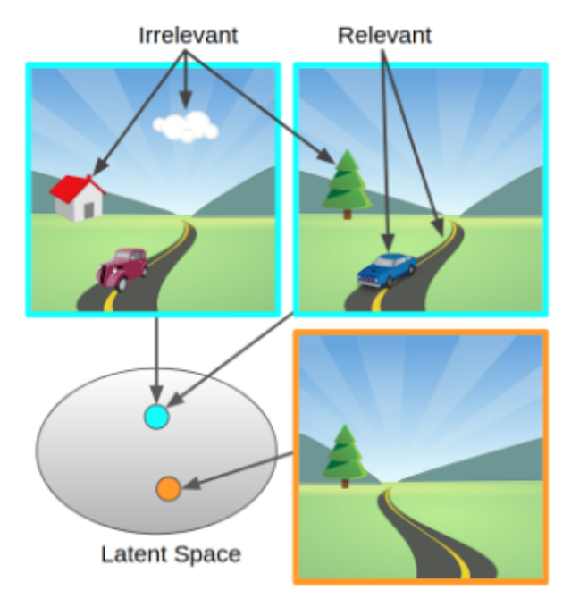

Figure 1: Robust representations of the visual scene should be insensitive to irrelevant objects (e.g., clouds) or details (e.g., car types), and encode two observations equivalently if their relevant details are equal (e.g., road direction and locations of other cars).

Contribution

We propose learning such an invariant representation using the bisimulation metric, where the distance between two observation encodings correspond to how "behaviorally different" both observations are. Our main contribution is a practical representation learning method based on the bisimulation metric suitable for downstream control, which we call Deep Bisimulation for Control (DBC). We additionally provide theoretical analysis that proves value bounds between the optimal value function of the true MDP and the optimal value function of the MDP constructed by the learned representation.

Background

We assume the underlying environment is a Markov decision process (MDP). An MDP is defined as the tuple $\langle S, A, P, R\rangle$ where S and A are the set of state and actions. $P(s'|s,a)$ is the transition function of the environment. $R(s,a,s')$ is the reward. An agent is usually represented by the policy distribution $\pi(a|s)$.

Definition 1 (Bisimulation Relations (Givan et al., 2003)). *Given an MDP $\mathcal{M}$, an equivalence relation B between states is a bisimulation relation if, for all states $\mathbf{s}_i, \mathbf{s}_j \in \mathcal{S}$ that are equivalent under B (denoted $\mathbf{s}_i \equiv_B \mathbf{s}_j$) the following conditions hold:*

$$\mathcal{R}(\mathbf{s}_i, \mathbf{a}) = \mathcal{R}(\mathbf{s}_j, \mathbf{a}) \qquad \forall \mathbf{a} \in \mathcal{A}, \tag{1}$$

$$\mathcal{P}(G|\mathbf{s}_i, \mathbf{a}) = \mathcal{P}(G|\mathbf{s}_j, \mathbf{a}) \quad \forall \mathbf{a} \in \mathcal{A}, \quad \forall G \in \mathcal{S}_B, \tag{2}$$

where $\mathcal{S}_B$ is the partition of $\mathcal{S}$ under the relation B (the set of all groups G of equivalent states), and $\mathcal{P}(G|\mathbf{s},\mathbf{a}) = \sum_{\mathbf{s}'\in G} \mathcal{P}(\mathbf{s}'|\mathbf{s},\mathbf{a})$.

Definition 2 (Bisimulation Metric). *From Theorem 2.6 in Ferns et al. (2011) with $c \in [0,1)$:*

$$d(\mathbf{s}_i, \mathbf{s}_j) = \max_{\mathbf{a}\in\mathcal{A}} (1-c)\cdot|\mathcal{R}^{\mathbf{a}}_{\mathbf{s}_i} - \mathcal{R}^{\mathbf{a}}_{\mathbf{s}_j}| + c\cdot W_1(\mathcal{P}^{\mathbf{a}}_{\mathbf{s}_i}, \mathcal{P}^{\mathbf{a}}_{\mathbf{s}_j}; d). \tag{3}$$

Method

Learning a Bisimulation Metric

$$J(\phi) = \Big(||\mathbf{z}_i - \mathbf{z}_j||_1 - |\hat{\mathcal{R}}(\bar{\mathbf{z}}_i) - \hat{\mathcal{R}}(\bar{\mathbf{z}}_j)| - \gamma\cdot W_2\big(\hat{\mathcal{P}}(\cdot|\bar{\mathbf{z}}_i, \bar{\pi}(\bar{\mathbf{z}}_i)),\, \hat{\mathcal{P}}(\cdot|\bar{\mathbf{z}}_j, \bar{\pi}(\bar{\mathbf{z}}_j))\big)\Big)^2$$

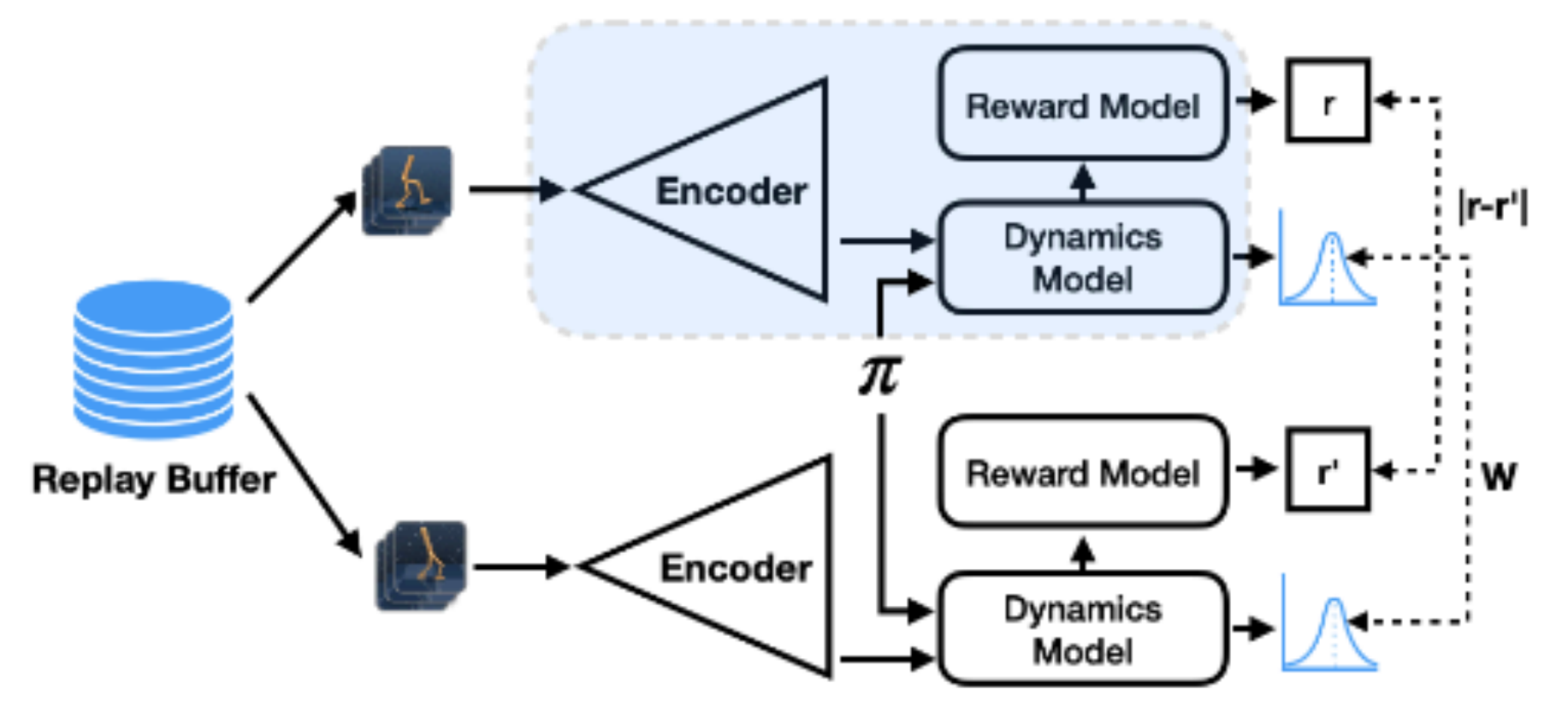

Figure 2: Deep bisimulation for control: for learning a bisimulation metric representation. Shaded in blue is the main model architecture, it is reused for both states, like a Siamese network. The loss is computed as a weighted sum of the reward and transition distribution distances (using the Wasserstein metric W). There is a separate optimization step to train the reward and dynamics models separately.

Method Cont.

Algorithm 1 Deep Bisimulation for Control (DBC)

```
1:  for Time $t = 0$ to $\infty$ do
2:      Encode observation $\mathbf{z}_t = \phi(\mathbf{s}_t)$
3:      Execute action $\mathbf{a}_t \sim \pi(\mathbf{z}_t)$
4:      Record data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_{t+1}\}$
5:      Sample batch $B_i \sim \mathcal{D}$
6:      Permute batch randomly: $B_j = \text{permute}(B_i)$
7:      Train policy: $\mathbb{E}_{B_i}[J(\pi)]$                ▷ Algorithm 2
8:      Train encoder: $\mathbb{E}_{B_i,B_j}[J(\phi)]$          ▷ Equation (4)
9:      Train dynamics: $J(\hat{\mathcal{P}},\phi) = (\hat{\mathcal{P}}(\phi(\mathbf{s}_t), \mathbf{a}_t) - \bar{\mathbf{z}}_{t+1})^2$
10:     Train reward: $J(\hat{\mathcal{R}},\hat{\mathcal{P}},\phi) = (\hat{\mathcal{R}}(\hat{\mathcal{P}}(\phi(\mathbf{s}_t),\mathbf{a}_t)) - r_{t+1})^2$
```

Algorithm 2 Train Policy (changes to SAC in blue)

```
1: Get value: $V = \min_{i=1,2} \hat{Q}_i(\hat{\phi}(\mathbf{s})) - \alpha \log \pi(\mathbf{a}|\mathbf{s})$
2: Train critics: $J(Q_i, \phi) = (Q_i(\phi(\mathbf{s})) - r - \gamma V)^2$
3: Train actor: $J(\pi) = \alpha \log p(\mathbf{a}|\mathbf{s}) - \min_{i=1,2} Q_i(\mathbf{s})$
4: Train alpha: $J(\alpha) = -\alpha \log p(\mathbf{a}|\mathbf{s})$
5: Update target critics: $\hat{Q}_i \leftarrow \tau_Q Q_i + (1-\tau_Q)\hat{Q}_i$
6: Update target encoder: $\hat{\phi} \leftarrow \tau_\phi \phi + (1-\tau_\phi)\hat{\phi}$
```

Rich Observation Results

Default Setting. Here, the pixel observations have simple backgrounds as shown in Figure 4 (top row) with training curves for our DBC and baselines. We see SLAC, a recent state-of-the-art model-based representation learning method that uses reconstruction, generally performs best.

Simple Distractors Setting. In the next setting, we incorporate simple, easy to predict distractors into the background -- different colored balls that obey the dynamics of an ideal gas, no attraction or repulsion between each pair of objects. Figure 4 (middle row) shows stills of observations in this new setting.

Natural Video Setting. Next, we incorporate natural video from the Kinetics dataset as background, shown in Figure 4 (bottom row). The results confirm our hypothesis: although a number of prior methods can learn effectively in the absence of complex distractors, when distractors are introduced, our non-reconstructive bisimulation-based method attains substantially better results.

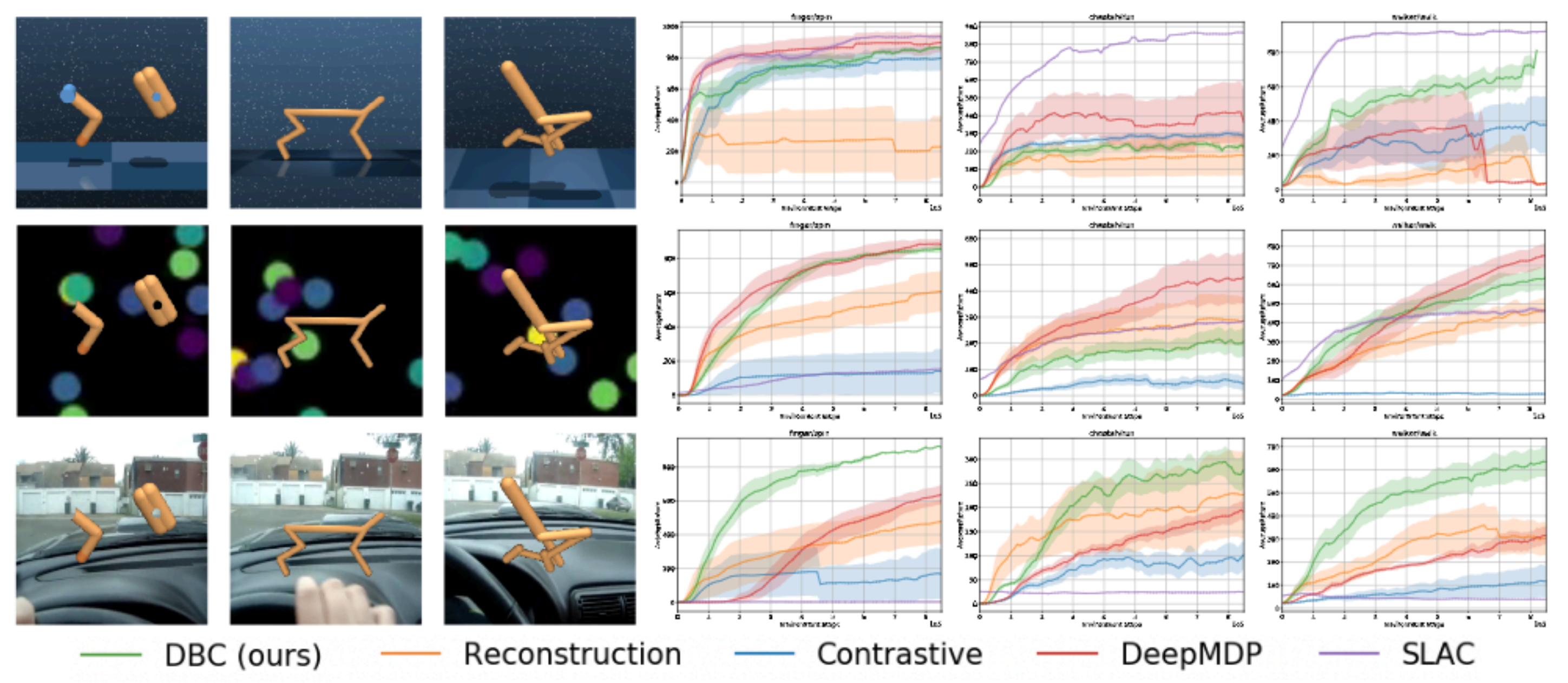

Figure 4: **Left observations**: Pixel observations in DMC in the default setting (top row) of the finger spin (left column), cheetah (middle column), and walker (right column), and natural video distractors (bottom row). **Right training curves**: Results comparing out DBC method to baselines on 10 seeds with 1 standard error shaded in the default setting. The grid-location of each graph corresponds to the grid-location of each observation.

Generalization Results

Theorem 4 (Task Generalization). *Given an encoder $\phi : O \mapsto S$ that maps observations to a latent bisimulation metric representation where $||\phi(\mathbf{s}_i) - \phi(\mathbf{s}_j)||_2 := \tilde{d}(\mathbf{s}_i, \mathbf{s}_j)$, S encodes information about all the causal ancestors of the reward $AN(R)$.*

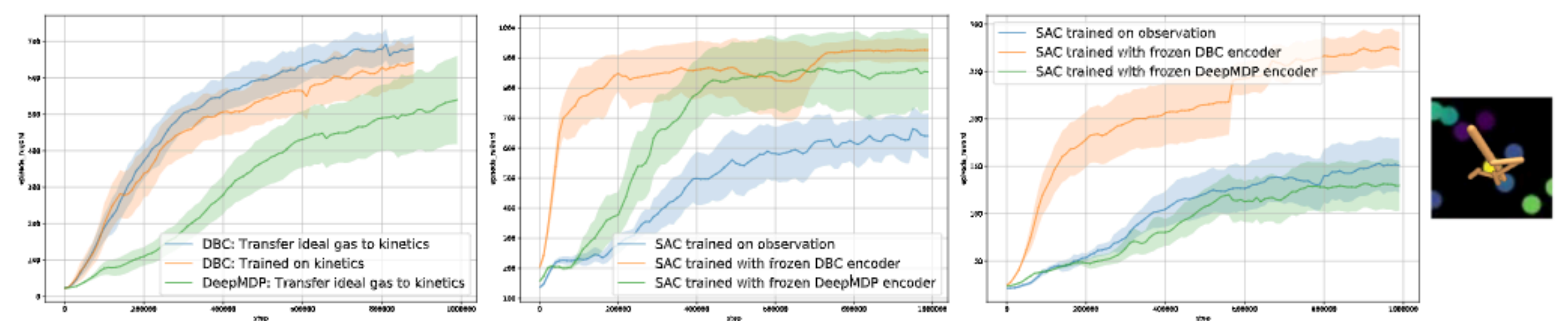

Figure 6: Generalization of a model trained on `simple distractors` environment and evaluated on `kinetics` (left). Generalization of an encoder trained on `walker_walk` environment and evaluated on `walker_stand` (center) and `walker_run` (right), all in the `simple distractors` setting. 10 seeds, 1 standard error shaded.

Qualitative Analysis of the Representation

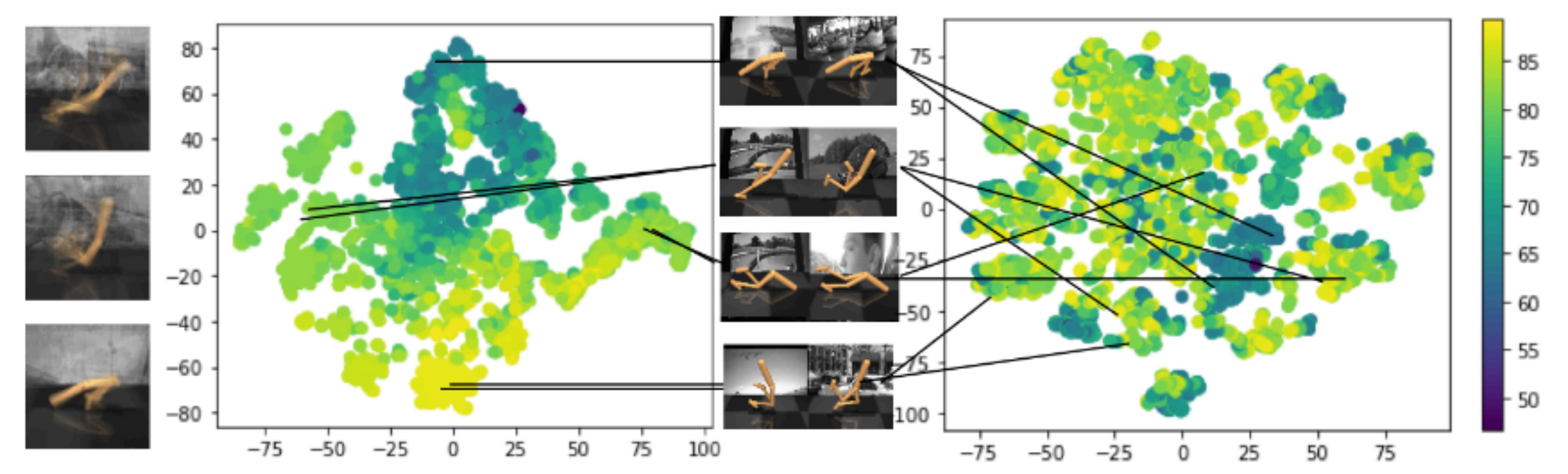

Figure 5: t-SNE of latent spaces learned with a bisimulation metric (left t-SNE) and VAE (right t-SNE) after training has completed, color-coded with predicted state values (higher value yellow, lower value purple). Neighboring points in the embedding space learned with a bisimulation metric have similar states and correspond to observations with the same task-related information (depicted as pairs of images with their corresponding embeddings), whereas no such structure is seen in the embedding space learned by VAE, where the same image pairs are mapped far away from each other. On the left are 3 examples of 10 neighboring points, averaged.

CARLA Results

Figure 8: Highway loop, third-person view of ego car (red), traffic during episode.

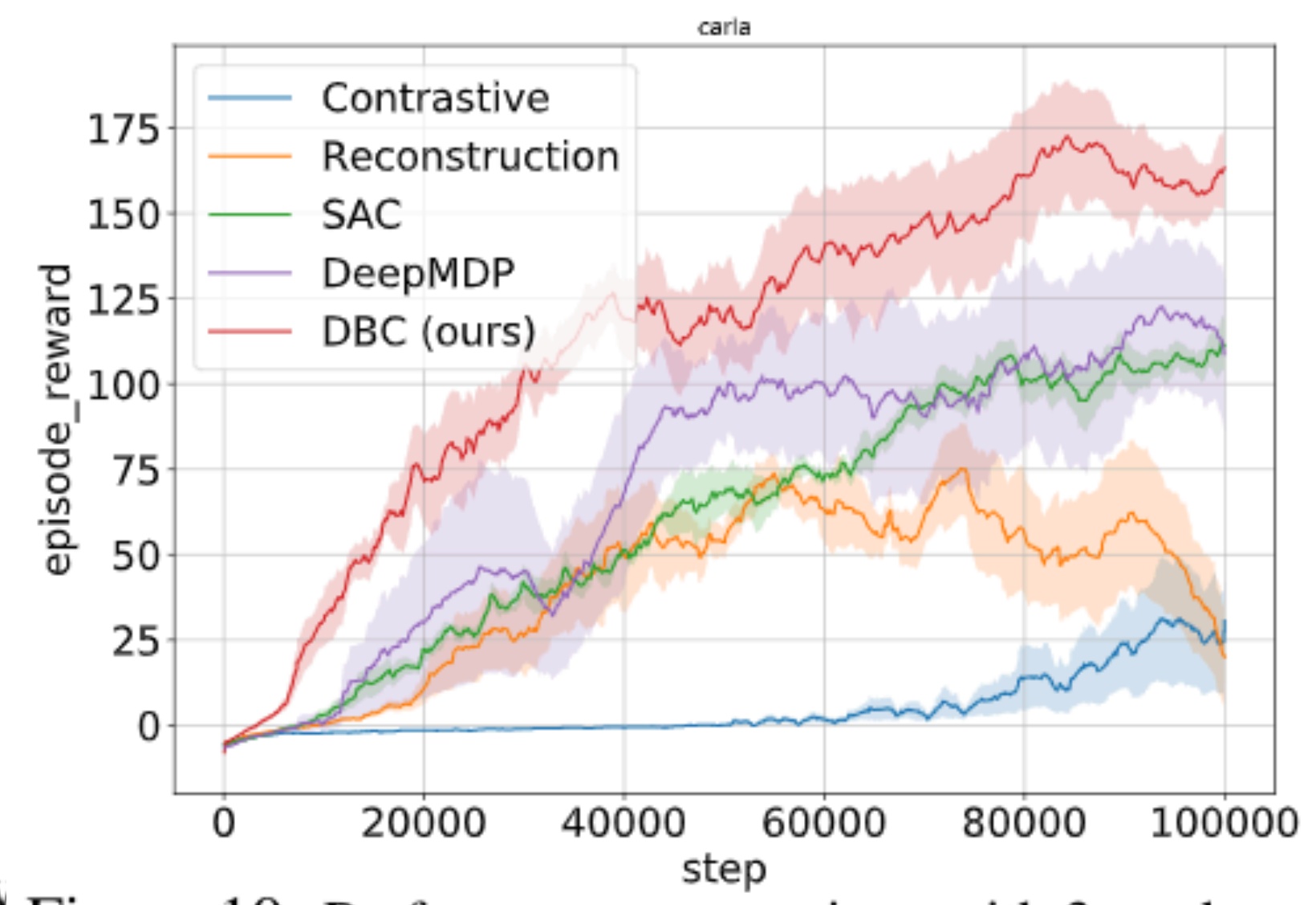

Figure 10: Performance comparison with 3 seeds on the driving tasks. Our DBC method (red) performs better than DeepMDP (purple) or learning straight from pixels without a representation (SAC, green), and much better than using contrastive losses (blue). The final performance of our method is 46.8% better than the next best baseline (SAC).

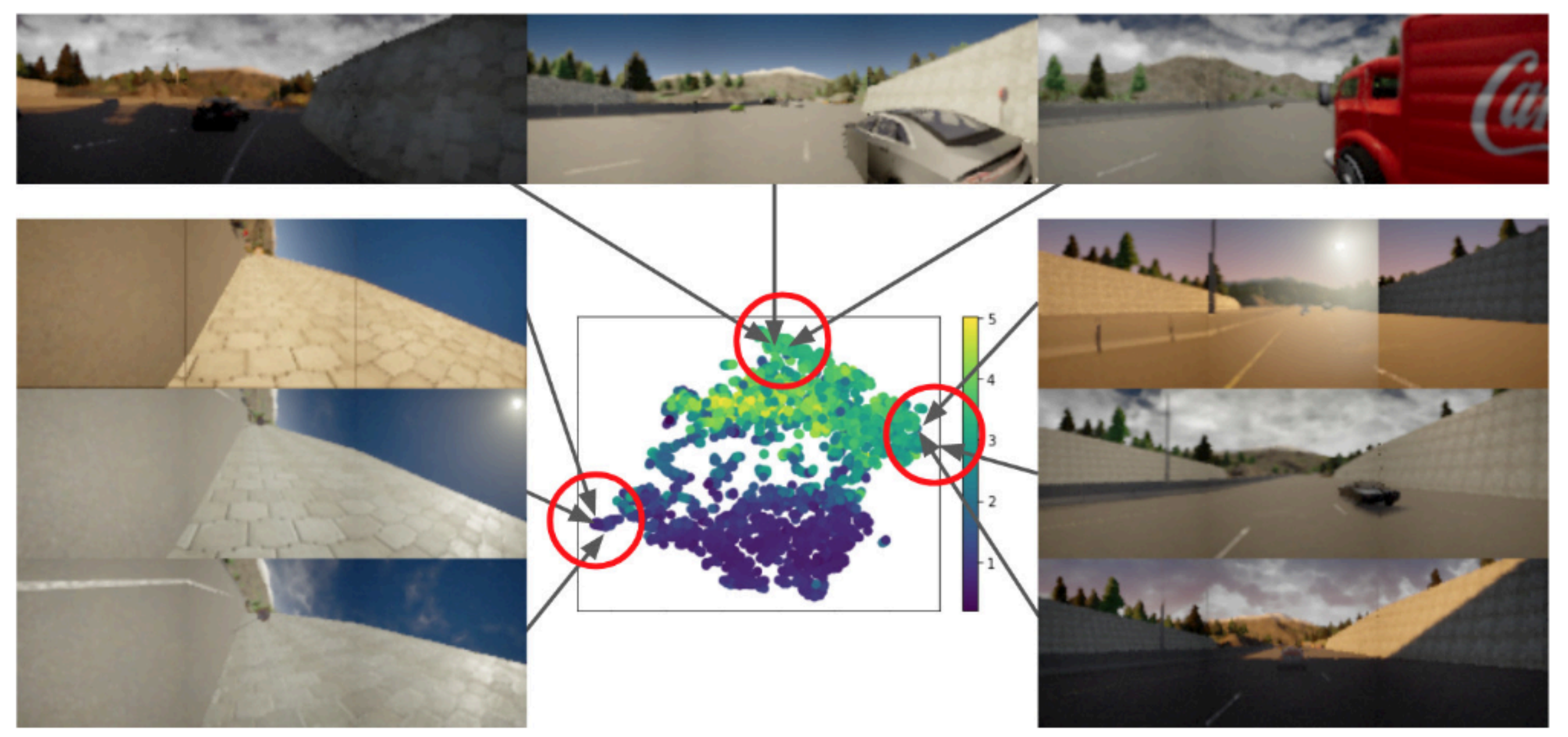

Figure 16: A t-SNE diagram of encoded first-person driving observations after 10k training steps of Algorithm 1, color coded by value (V in Algorithm 2). **Top**: the learned representation identifies an obstacle on the right side. Whether that obstacle is a dark wall, bright car, or truck is task-irrelevant: these states are behaviourally equivalent. **Left**: the ego vehicle has flipped onto its left side. The different wall colors, due to a setting sun, is irrelevant: all states are equally stuck and low-value (purple t-SNE color). **Right**: clear highway driving. Clouds and sun position are irrelevant.