Click to Move: Controlling Video Generation with Sparse Motion (Supplementary Material)

Pierfrancesco Ardino[1,2], Marco De Nadai[2], Bruno Lepri[2], Elisa Ricci[1,2], Stéphane Lathuilière[3]

[1]University of Trento [2]Fondazione Bruno Kessler
[3]LTCI, Télécom Paris, Institut Polytechnique de Paris

In this supplementary material, we provide additional information regarding network architectures (Section 1) and implementation details (Section 2). Then, we provide additional high-resolution qualitative results in Section 3 on both Cityscapes and KITTI 360. We shared an HTML file named *video.html*, which shows qualitative comparisons in videos.

1. Network architectures

We provide the details of the trajectory encoder E_s and flow predictor D in Table 1. The flow decoder is composed of two heads (referred to as $Flow_D$ and Occ_D) and shared layers (referred to as $Feat_D$). For more details regarding the architecture of the other networks, please refer to the code attached with this supplementary material.

1.1. Convergence issue

During preliminary experiments, we observed that using eq.(1) update rule (from the manuscript) ended up with all the nodes having the exact same features. For this reason, we added a residual update that helped objects converging to better features. Indeed, this residual update can be seen as skip connections, similar to those of resnet architectures, that allow gradient information to pass through the GCN updates and mitigate vanishing gradient problems.

2. Implementation.

Our architecture is implemented with Pytorch 1.7.0, while the graph neural network has been implemented using Pytorch Geometric. We use the ADAM optimizer [2] with a learning rate of 2e-4 for the Generation module and 1e-4 for the Motion estimation. The modules are trained upon convergence. Training takes about one day for the Cityscapes dataset and two days for the KITTI 360 dataset. The experiments are done using two Nvidia RTX 2080Ti.

3. Additional qualitative results

For the Cityscapes dataset, we train an additional model at higher resolution (i.e. 256×128 pixels) without changing any hyper-parameter. The results obtained with this model on two initial frames are shown in Figure 1 and Figure 2. These results are well in-line the the qualitative results reported in the main paper. We observe that the other methods are not able to move the object (see the red bounding boxes that indicate the initial position of the object). Indeed, the cars are either static, in Sheng *et al.* [3] and Sheng*, or blurry, in S. Sheng*. On the contrary, our approach is able to move the object and generates frames of good quality.

Figure 4 and Figure 5 instead show some additional visual results on KITTI 360.

Finally, Figure 6 shows a qualitative example of the ablation study we performed in the main paper. The first row of this Figure shows a version of our network that does not model object interactions. By comparing the first and second rows of Figure 6, we clearly see that, while the highlighted object correctly moves, all the other objects have unrealistic motions. This confirms the quantitative results about the importance of modelling the object interactions to have temporal consistent movements of different objects in the scene.

Part	Input → Output Shape	Layer Information
E_s	(BS,5, 64,128, 1) → (BS,5, 32,64,32)	3DCONV-(N32, K{3,4,4}, S{1,2,2}, P{1,1,1}), BN, LeakyReLU
	(BS,5, 32,64,32) → (BS,5, 16,32,64)	3DCONV-(N64, K{3,4,4}, S{1,2,2}, P{1,1,1}), BN, LeakyReLU
	(BS,5, 16,32,64) → (BS,5, 8,16,128)	3DCONV-(N128, K{3,4,4}, S{1,2,2}, P{1,1,1}), BN, LeakyReLU
$Feat_D$	(BS * 5, 2, 4, 272) → (BS * 5, 2, 4, 512)	CONV-(N512, K3, S1, P1), BN, LeakyReLU
	(BS * 5, 2, 4, 512) → (BS * 5, 4, 8, 256)	UPCONV, CONV-(N256, K3, S2, P1), BN, LeakyReLU
	(BS * 5, 4, 8, 512) → (BS * 5, 8, 16, 128)	SKIP, UPCONV, CONV-(N128, K3, S1, P1), BN, LeakyReLU
	(BS * 5, 4, 8, 512) → (BS, 5, 8, 16, 128)	RESHAPE
	(BS, 5, 8, 16, 256) → (BS, 5, 8, 16, 128)	SKIP, 3DCONV-(N128, K{3,3,3}, S{1,1,1}, P{1,1,1}), BN, LeakyReLU
	(BS, 5, 8, 16, 128) → (BS * 5, 8, 16, 128)	RESHAPE
	(BS * 5, 8, 16, 256) → (BS * 5, 16, 32, 64)	SKIP, UPCONV, CONV-(N64, K3, S1, P1), BN, LeakyReLU
	(BS * 5, 16, 32, 64) → (BS, 5, 16, 32, 64)	RESHAPE
	(BS, 5, 16, 32, 128) → (BS, 5, 16, 32, 64)	SKIP, 3DCONV-(N64, K{3,3,3}, S{1,1,1}, P{1,1,1}), BN, LeakyReLU
	(BS, 5, 16, 32, 64) → (BS * 5, 16, 32, 64)	RESHAPE
	(BS * 5, 16, 32, 128) → (BS * 5, 32, 64, 32)	SKIP, UPCONV, CONV-(N32, K3, S1, P1), BN, LeakyReLU
	(BS * 5, 32, 64, 32) → (BS, 5, 32, 64, 32)	RESHAPE
	(BS, 5, 32, 64, 64) → (BS, 5, 32, 64, 32)	SKIP, 3DCONV-(N32, K{3,3,3}, S{1,1,1}, P{1,1,1}), BN, LeakyReLU
$Flow_D$	(BS * 5, 32, 64, 64) → (BS * 5, 64, 128, 32)	SKIP, UPCONV, CONV-(N32, K3, S1, P1), BN, LeakyReLU
	(BS * 5, 64, 128, 32) → (BS * 5, 64, 128, 2)	CONV-(N2, K5, S1, P2), IN, Tanh
Occ_D	(BS * 5, 32, 64, 64) → (BS * 5, 64, 128, 32)	SKIP, UPCONV, CONV-(N32, K3, S1, P1), BN, LeakyReLU
	(BS * 5, 64, 128, 32) → (BS * 5, 64, 128, 2)	CONV-(N1, K5, S1, P2), Sigmoid

Table 1: Network architecture. We use the following notation: Z: the dimension of motion vector, K: kernel size, S: stride size, P: padding size, CONV: a convolutional layer, UPCONV: upsample, 3D-CONV: 3D convolutional layer, BN: Batch Normalization, SKIP: skip connection

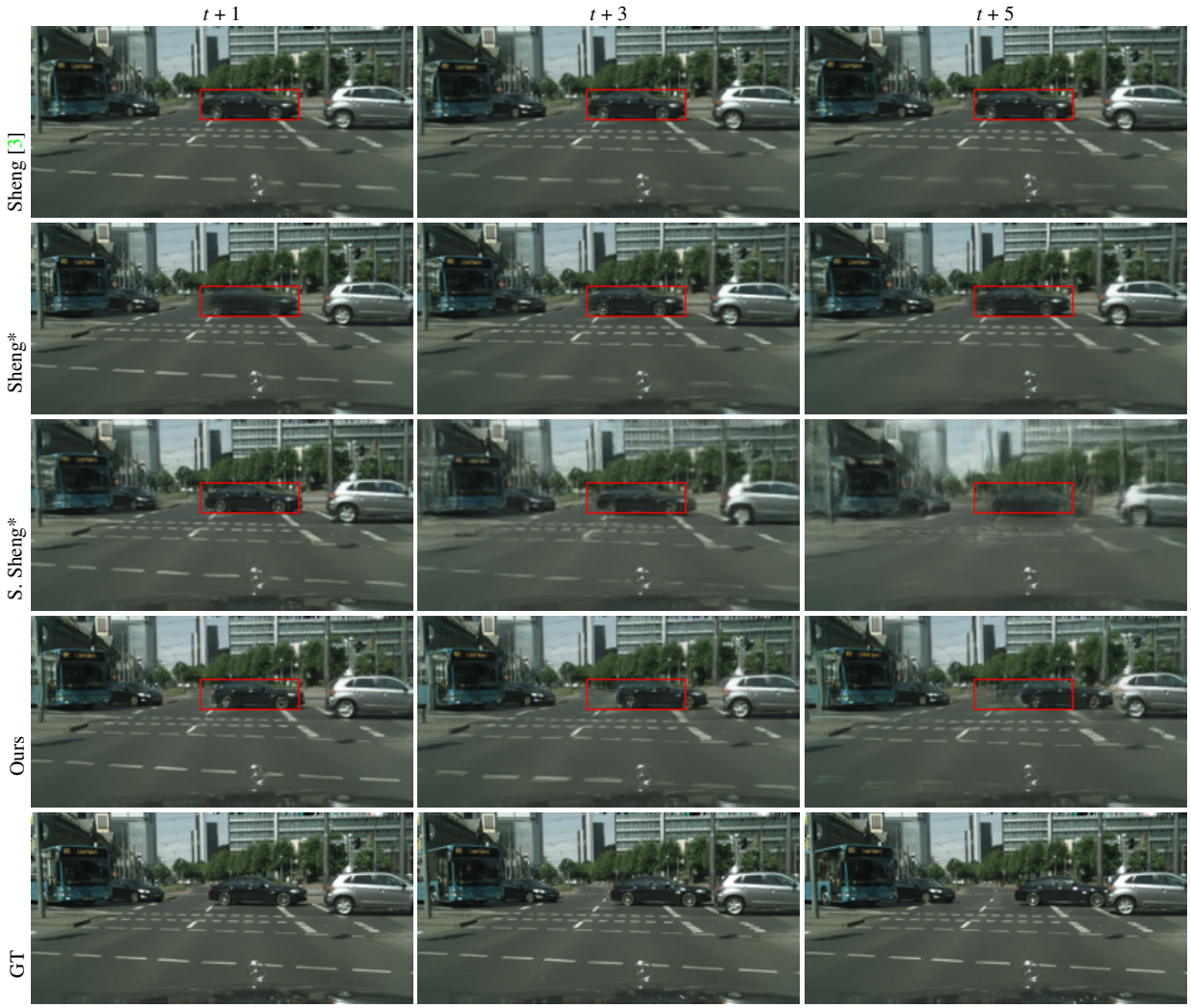

Figure 1: Results of predicting the frames $t + 1$, $t + 3$, and $t + 5$ on the Cityscapes dataset [1] with ground truth reference. On first three columns, we move the pedestrian near the semaphore to left. On the last three columns we move car crossing the street. The position of the moved object at $t = 0$ is highlighted in red. Zoom for details.

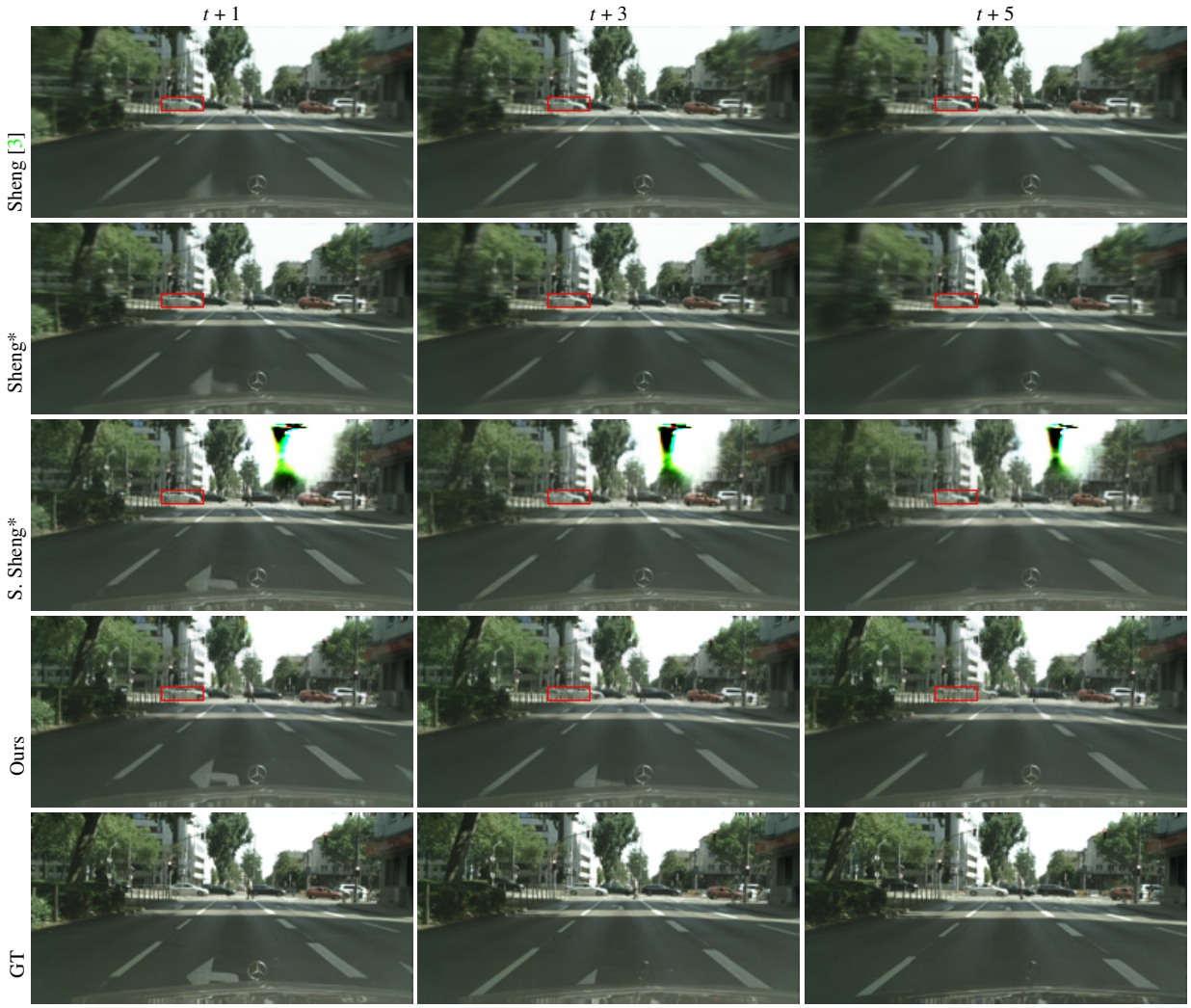

Figure 2: Results of predicting the frames $t + 1$, $t + 3$, and $t + 5$ on the Cityscapes dataset [1] with ground truth reference. On first three columns, we move the pedestrian near the semaphore to left. On the last three columns we move car crossing the street. The position of the moved object at $t = 0$ is highlighted in red. Zoom for details.

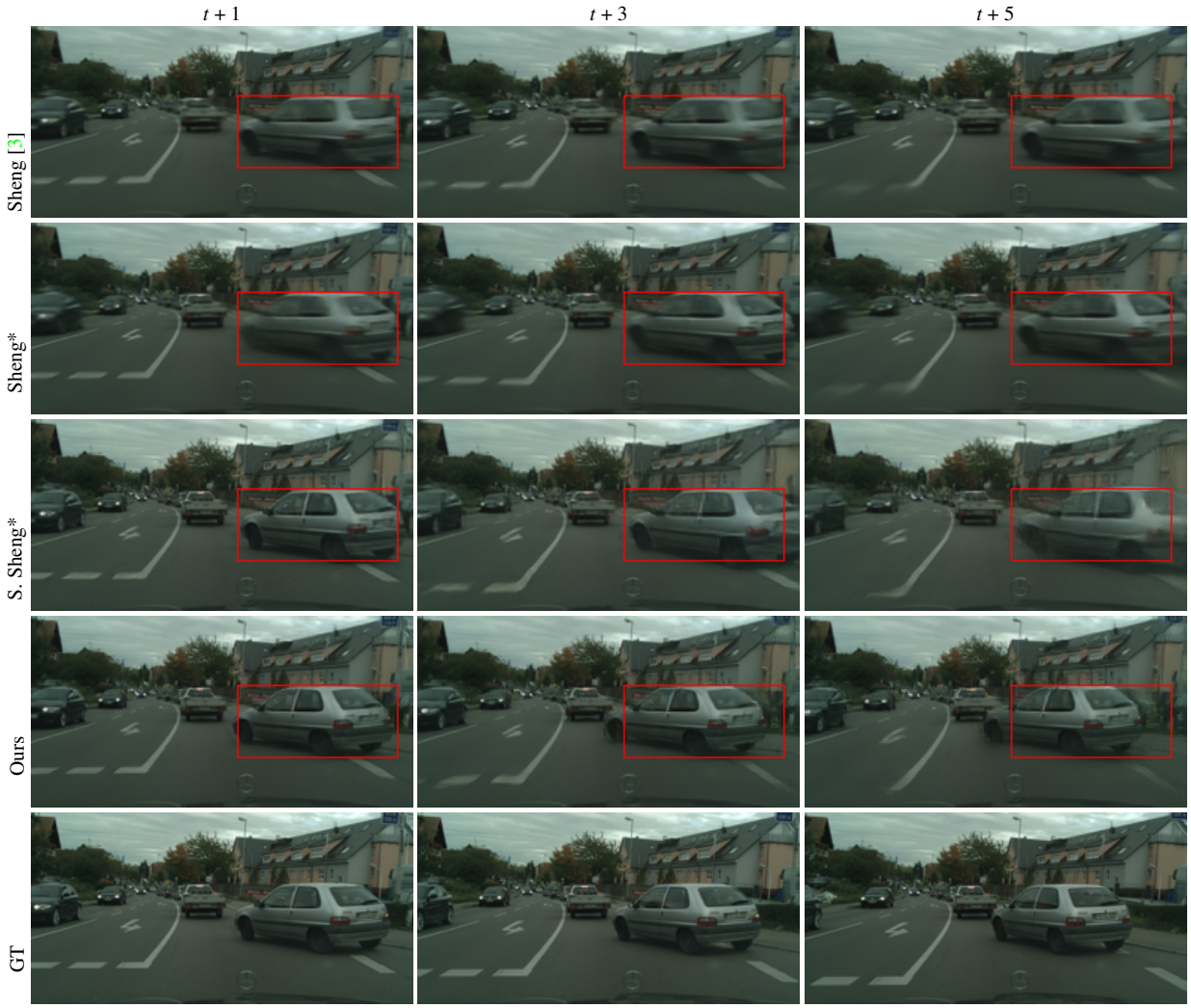

Figure 3: Results of predicting the frames $t + 1$, $t + 3$, and $t + 5$ on the Cityscapes dataset [1] with ground truth reference. On first three columns, we move the pedestrian near the semaphore to left. On the last three columns we move car crossing the street. The position of the moved object at $t = 0$ is highlighted in red. Zoom for details.

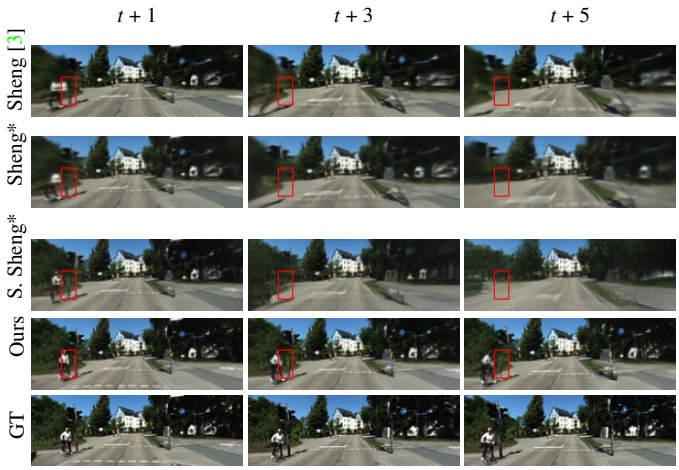

Figure 4: Results of predicting the frames $t + 1$, $t + 3$, and $t + 5$ on the KITTI360 dataset [4]. The position of the moved object at $t = 0$ is highlighted in red. Zoom for details.

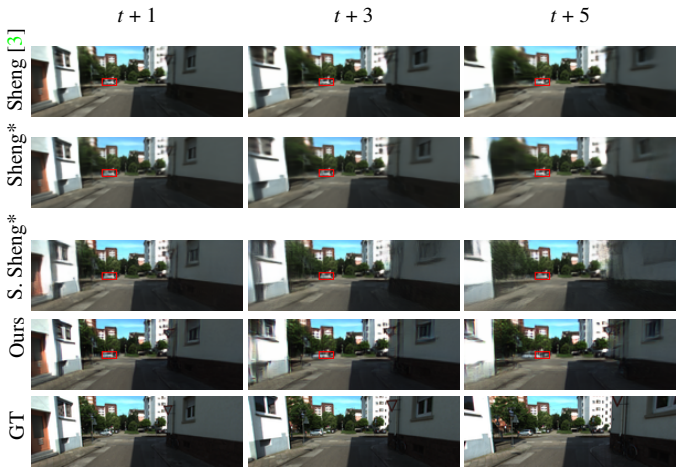

Figure 5: Results of predicting the frames $t + 1$, $t + 3$, and $t + 5$ on the KITTI360 dataset [4]. The position of the moved object at $t = 0$ is highlighted in red. Zoom for details.

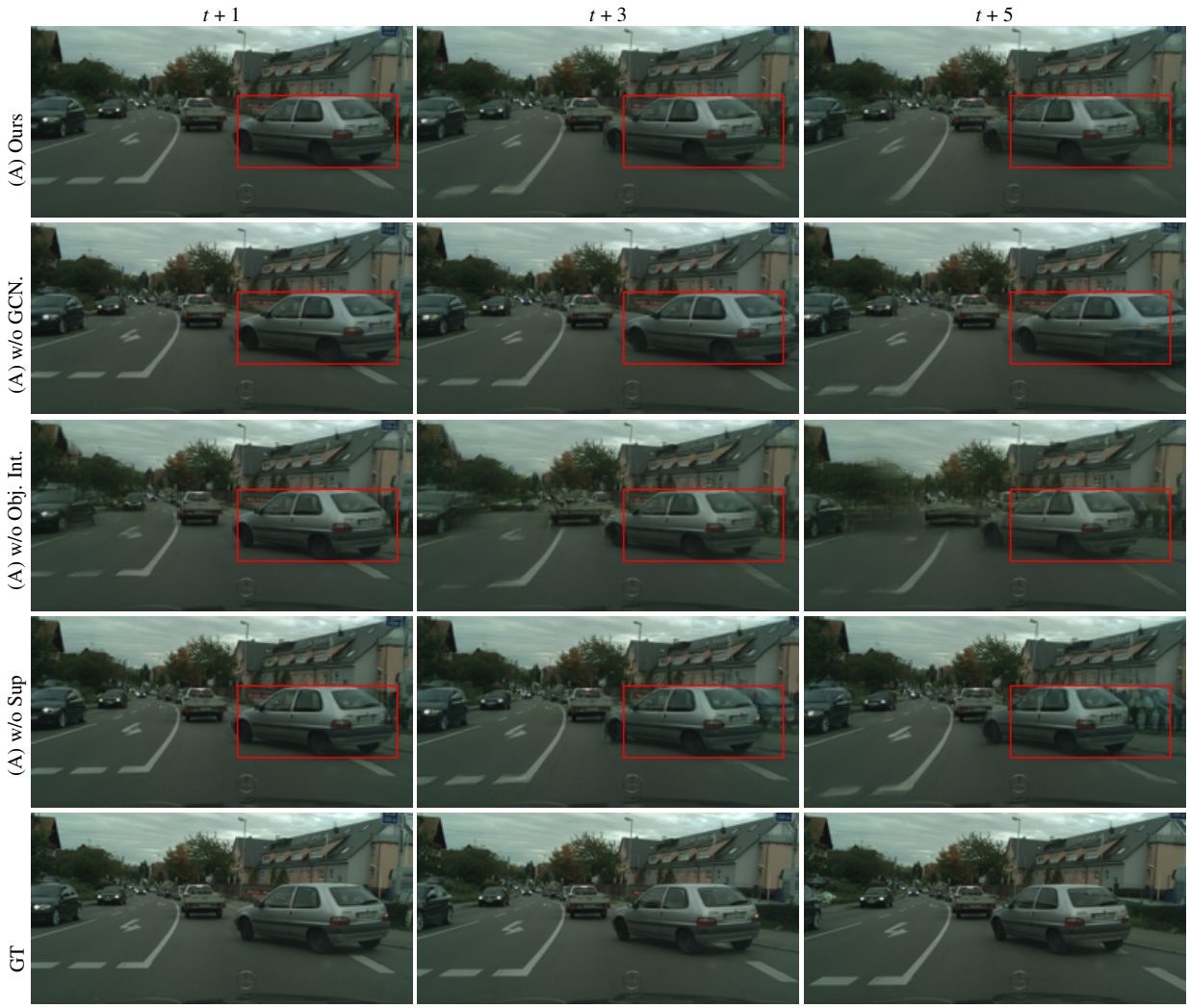

Figure 6: Results of the ablation test on the Cityscapes dataset [1].

References

[1] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223, 2016. 3, 4, 5, 7

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[3] Lu Sheng, Junting Pan, Jiaming Guo, Jing Shao, and Chen Change Loy. High-quality video generation from static structural annotations. *International Journal of Computer Vision*, 128:2552–2569, 2020. 1, 3, 4, 5, 6

[4] Jun Xie, Martin Kiefel, Ming-Ting Sun, and Andreas Geiger. Semantic instance annotation of street scenes by 3d to 2d label transfer. In *CVPR*, 2016. 6