Click to Move: Controlling Video Generation with Sparse Motion

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

Abstract

This paper introduces Click to Move (C2M), a novel framework for video generation where the user can control the motion of the synthesized video through mouse clicks specifying simple object trajectories of the key objects in the scene. Our model receives as input an initial frame, its corresponding segmentation map and the sparse motion vectors encoding the input provided by the user. It outputs a plausible video sequence starting from the given frame and with a motion that is consistent with user input. Notably, our proposed deep architecture incorporates a Graph Convolution Network (GCN) modelling the movements of all the objects in the scene in a holistic manner and effectively combining the sparse user motion information and image features. Experimental results show that C2M outperforms existing methods on two publicly available datasets, thus demonstrating the effectiveness of our GCN framework at modelling object interactions. The source code is publicly available at https://github.com/PierfrancescoArdino/C2M.

1. Introduction

Recent years have witnessed several breakthroughs in the generation of high dimensional data such as images [6, 8, 24] or videos [36, 40]. However, most practical and commercial applications require to control generated visual data on inputs provided by the user. For instance, in image manipulation, photo editing software [1] applies deep learning models to allow users to change portions of an image [27, 31, 48].

Regarding videos, several possible ways to control the generated sequences have been considered. For instance, the generation of frames can be conditioned on simple categorical attributes [13], short sentences [22] or sound [35]. An interesting recent research direction comprises works that attempt to condition the video generation process providing motion information as input [33, 34, 36, 43]. These approaches allow to generate videos of moving faces [43], human silhouettes and, in general, of arbitrary objects [33, 34, 36]. However, these works mainly deal with videos depicting a single object. It is indeed extremely more challenging to animate images and generate videos when multiple objects are present in the scene, as there is no simple way to disentangle the information associated with each object and easily model and control its movement.

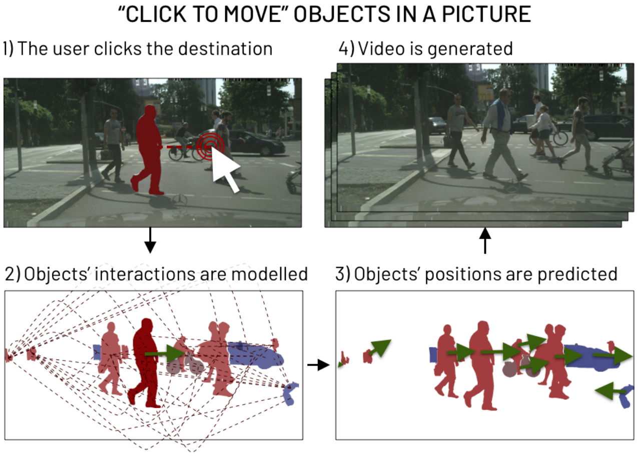
"CLICK TO MOVE" OBJECTS IN A PICTURE
1) The user clicks the destination
4) Video is generated
2) Objects' interactions are modelled
3) Objects' positions are predicted

Figure 1. Illustration of the video generation process of Click to Move (C2M): 1) the user selects the objects in a scene and specify their movements. 2) Our network models the interactions between *all* objects through the GCN and 3) predicts their displacement. 4) The network produces a realistic and temporally consistent video.

This paper introduces Click to Move (C2M), the first approach that allows users to generate videos in complex scenes by conditioning the movements of specific objects through mouse clicks. Fig. 1 illustrates the video generation process of C2M. The user only needs to select few objects in the scene and to specify the 2D location where each object should move. Our proposed framework receives as inputs an initial frame with its segmentation map and synthesizes a video sequence depicting objects for which movements are coherent with the user inputs. The proposed deep architecture comprises three main modules: (i) an appearance encoder that extracts the feature representation from the first frame and the associated segmentation map, (ii) a motion module that predicts motion information from user inputs and image features, and (iii) a generation module that outputs the synthesised frame sequence. In complex scenes with multiple objects, modelling interactions is essential to generate coherent videos. To this aim, we propose to adopt a Graph Neural Network (GCN), which models object interactions and infers the plausible displacements for all the objects in the video, while respecting the user's constraints. Experimental results show that our approach outperforms previous video generation methods on two publicly available datasets and demonstrate the effectiveness of the proposed GCN framework in modelling object interactions in complex scenes.

Our work is inspired by previous literature that generates videos from an initial frame and the associated segmentation maps [28, 32]. From these works, we inherit a two-stage procedure where we first estimate the optical flows between an initial frame and all the generated frames, and subsequently refine the image obtained by warping the initial frame according to the estimated optical flows. However, our framework improves over these previous works as it allows the user the possibility to directly control the video generation process with simple mouse clicks. Similarly to the work of Hao *et al.* [12], we propose to control object movements via sparse motion inputs. However, thanks to the GCN, our approach can deal with scenes with multiple objects, while [12] cannot. Furthermore, the method in [12] does not explicitly consider the notion of *object*, as it does not use any instance segmentation information, and does not model the temporal relation between multiple frames. We instead work on multiple frames and in the semantic space, so the user can intuitively select the object of interest and move it in a temporal consistent way. The use of semantic information is motivated by recent findings in the area of image manipulation where it has been shown that semantic maps are beneficial in complex scenes [2, 20].

Contributions. Overall, the main contributions of our work are as follows:

- We propose Click to Move (C2M), a novel approach for video generation of complex scenes that permits user interaction by selecting objects in the scene and specifying their final location through mouse clicks.
- We introduce a novel deep architecture that leverages the initial video frame and its associated segmentation map to compute the motion representations that enable the generation of frame sequence. Our deep network incorporates a novel GCN that models the interaction between objects to infer the motion of all the objects in the scene.
- Through an extensive experimental evaluation, we demonstrate that the proposed approach outperforms its competitors [28,32] in term of video quality metrics and can synthesize videos where object movements follow the user inputs.

2. Related Works

Video generation with user control. With the recent progress in deep video synthesis, researchers have focused in designing new approaches that include user input in the generation process. Video generation can be controlled by different means. For example, MoCoGAN [36] disentangles videos into motion and content latent spaces. Therefore, it is possible to control videos by "copying" the action from another video or by changing the identity of the person. Chan *et al.* [3] propose to generate dance videos following a "do as I do" motion transfer strategy: body poses are estimated for every frame of another video and transferred to control the pose of the person in the generated video. Wiles *et al.* [43] control human face motion through a driving vector that can be extracted from videos or pose information. Siarohin *et al.* [33,34] propose an approach suitable to arbitrary objects and learn motion representations without requiring specific prior knowledge. This approach can be employed with various types of videos, ranging from human bodies to robotics. Regarding audio-visual methods, talking heads video can be generated from an initial image and an input audio clip [4, 43, 50]. In this paper, we propose a novel framework that involves the user in the generation process. However, while previous works mostly focus on generating videos depicting a single object (e.g. a face or human body), we address the more challenging task of video synthesis of complex scenes where multiple objects have to move consistently while accounting for user input.

Future frame prediction. The problem we address in this work is closely related to future frame prediction, which aims to generate a video sequence given its initial frames. Early works formulate the problem as a deterministic prediction task [9, 26, 39]. However, this formulation cannot work on most real world videos due to the inherent motion uncertainty. Thus, recent approaches adopt adversarial [18] or variational [11, 19, 36] formulations that can model stochasticity. Several works focus on the architectural design and propose to estimate optical flow [10, 18, 21, 23] to generate the future frames by warping the previous one. Others works study solutions for long term predictions [14, 29, 38, 47]. Similarly, Li *et al.* [21] propose a multi-step network that first generates an optical flow, then converts it back to the RGB space to generate novel videos. Instead, Zhang *et al.* [49] propose to employ an optical flow encoder that maps motion information to a latent space. At test time, different random motion vectors can be sampled to generate video with different motion.

When it comes to complex environment involving multiple objects, additional supervision is highly beneficial. For example, Wu *et al.* [44] use video frames, optical flows, instance maps and semantic information together to decouple the background from the dynamic objects and thus predict their trajectory. Similarly, Hao *et al.* [12] show that providing sparse motion trajectories to their model helps generating videos with higher quality. However, contrary to our approach, their method does not take advantage of instance segmentation and does not model object interactions.

Recently, Pan *et al.* [28] and Sheng *et al.* [32] have proposed to get a benefit from segmentation information to improve video generation. Videos are generated from a single frame and the corresponding segmentation map. Both approaches are based on a two-stage procedure. The first stage aims at estimating the optical flow between the initial frame and every generated frame. In the the second stage, the initial frame is warped according to the optical flow and refined by an encoder-decoder network. Inspired from these works, our approach adopts a similar variational auto-encoder framework boosted with optical flow and occlusion supervision. However, we include a novel Graph Convolutional Network (GCN) that models object interactions and takes into account the sparse motion vectors provided by the user.

3. Click to Move framework

We aim at generating a video from its initial frame $\mathbf{X}_0 \in \mathbb{R}^{H\times W\times 3}$ and a set of user-provided 2D vectors that specify the motion of the key objects in the scene. At test time, we assume that we also have at our disposal the instance segmentation maps of the initial frame. Our system is trained on a dataset of videos composed of T frames with the corresponding instance segmentation maps at every frame. As we will see later, in practice, instance segmentation is obtained using a pre-trained model.

Considering a set of C classes, we assume that N objects are detected at time t in the frame $\mathbf{X}_t \in \mathbb{R}^{H\times W\times 3}$. The instance segmentation is represented via a segmentation map $\mathbf{S}_t \in \{0,1\}^{H\times W\times C}$, a class label map $\mathbf{C}_t \in \{1,...,C\}^{H\times W}$ and an instance map $\mathbf{I}_t \in \{1,...,N\}^{H\times W}$ that specifies the instance index for every pixel. At test time, the user provides the motion of the M objects in the scene by drawing 2D arrows corresponding to the displacement between the barycenter of the object in $\mathbf{X}_0$ and the object's desired position at time T (See Fig. 1). Notably, the user is free to provide motion vectors for as many objects as desired. Therefore the motion vectors are represented by a list $\mathcal{M} = \{(\boldsymbol{\delta}_m, i_m), 1 \leq m \leq M\}$, where $\boldsymbol{\delta}_m \in \mathbb{R}^2$ contains the barycenter displacement of the object with instance index i_m. At training time, the list $\mathcal{M}$ is obtained by randomly sampling objects in every video and estimating their corresponding $\boldsymbol{\delta}_m$, which is defined as the displacement of the instance segmentation's barycenters between the first and last frame.

The proposed framework is articulated in three main modules, as illustrated in Fig. 2. First, the *Appearance encoding* is in charge of encoding the initial frame. This module receives as input the concatenation of the initial frame $\mathbf{X}_0$, the segmentation $\mathbf{S}_0$ and the instance map $\mathbf{I}_0$, while it outputs a feature map $\boldsymbol{z}_a$ via the use of an Encoder E_A. Second, the *Motion encoding*, predicts the video motion from the motion vectors provided by the user and the image features $\boldsymbol{z}_a$. This module includes a novel Graph Convolutional Network (GCN) that infers the motion of all the objects in the scene by combining the object motion vectors in $\mathcal{M}$ and the image features $\boldsymbol{z}_a$. This motion module is described in Sec. 3.2 while the details specific to our GCN are given in Sec. 3.1. Finally, the *Generation module* is in charge of combining the encoded appearance and the predicted motion to generate every frame of the output video.

3.1. Object motion estimation with GCNs

Our GCN aims at inferring the motion of all the objects in the scene by combining the motion vectors provided by the user and the image features $\boldsymbol{z}_a$. This section first describes the specific message-passing algorithm that we introduce to model the motion vectors. Then we show how our GCN is embedded into a Variational Auto-Encoder (VAE) framework to allow sampling the possible object motions that respect the user's constraints.

Handling user control with GCNs. We propose to use a graph to model the interactions between the objects in the scene. Each node corresponds to one of the N objects detected in $\mathbf{X}_0$. The graph is obtained fully connecting all the objects with each other. Let us introduce the following notations: $\boldsymbol{f}_n$ is the feature vector for the n^{th} object and is extracted from $\boldsymbol{z}_a$ via region-wise average pooling. $\boldsymbol{d}_n \in \mathbb{R}^2$ is the estimated barycenter displacement for the n^{th} object. Finally, $u_n \in \{0,1\}$ is a binary value that specifies whether the object motion has been provided by the user ($u_n = 1$) or if it should be inferred ($u_n = 0$).

In a standard GCN [45], the layer-wise propagation rule specifies how the features $\boldsymbol{f}_n^{(k)}$ at iteration k of the node n are computed from the features of it neighbouring nodes at the previous iteration $\boldsymbol{f}_j^{(k-1)}$:

$$\boldsymbol{f}_n^{(k)} = \sum_{j\in\mathbf{N}(n)} \frac{1}{\sqrt{\mathcal{D}_{nj}}} \boldsymbol{\theta}^\top \boldsymbol{f}_n^{(k-1)} \qquad (1)$$

where $\mathbf{N}(n)$ denotes the neighbours of the node n, $\boldsymbol{\theta}$ are the trainable parameters and $\mathcal{D}_{nj}$ is a normalization factor equal to the sum of the degree of the nodes n and j. In our context, we need to modify this update rule to take into account that

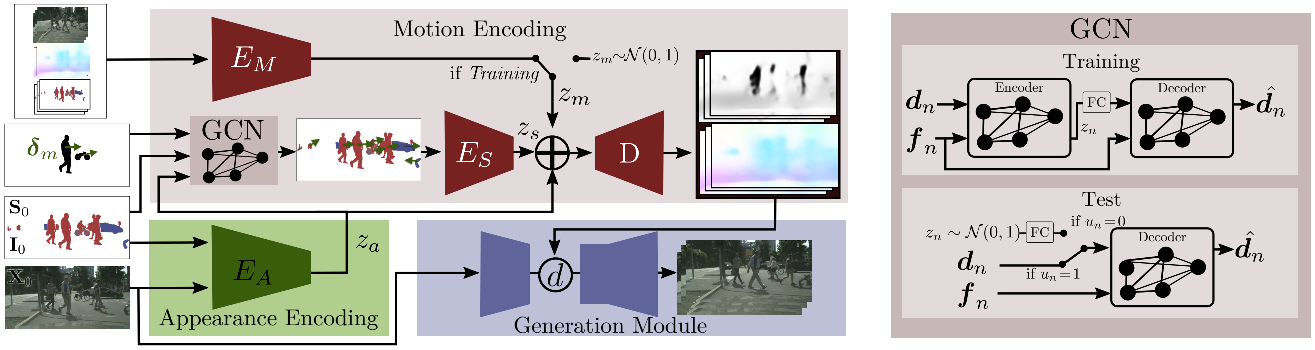

Figure 2. Our network is composed of three modules, namely (i) Appearance encoding, (ii) Motion encoding, and (iii) Generation module. The Appearance Encoding focuses on learning the visual appearance from $\mathbf{X}_0$. The Motion Encoding models the interactions between the objects, predicts their displacement, encodes the motion, and generates the optical flow and occlusion mask for the Generation Module, which focuses on generating temporal consistent and realistic videos. On the right, we show our GCN module to model objects' interactions.

the object motion of each node is either known or unknown. Besides, we propose two different propagation rules for the node features $\boldsymbol{f}_n$ and the motion vectors $\boldsymbol{d}_n$. We propose to make these rules depending on u_n. If $u_n = 1$, the node corresponds to an object with a motion controlled by the user and we update only the features:

$$\boldsymbol{f}_n^{(k)} = \boldsymbol{f}_n^{(k-1)} + \sum_{j \in \mathbf{N}(n)} \frac{1}{\sqrt{\mathcal{D}_{nj}}} \boldsymbol{\theta}_f^\top (\boldsymbol{f}_n^{(k-1)} \oplus \boldsymbol{d}_n^{(k-1)}) \quad (2)$$

$$\boldsymbol{d}_n^{(k)} = \boldsymbol{d}_n^{(k-1)}. \quad (3)$$

Here, $\boldsymbol{\theta}_f$ denotes the trainable parameters and $\oplus$ is the concatenation operation. This formulation allows propagating feature information through the node while keeping the object motion constant for the nodes with known motion. Note that, in (2), we opt for a residual update since the messages from the neighbouring nodes are added to the current value $\boldsymbol{f}_n^{(k-1)}$. Our preliminary results showed that (1) update rule ended up with all the nodes having the exact same features. On the contrary, the 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.

If $u_n = 0$, the node corresponds to an object with unknown motion and we update both the features and the motion vector. The feature update remains identical to (2) and the motion vector is updated as follows:

$$\boldsymbol{d}_n^{(k)} = \boldsymbol{d}_n^{(k-1)} + \sum_{j \in \mathbf{N}(n)} \frac{1}{\sqrt{\mathcal{D}_{nj}}} \boldsymbol{\theta}_d^\top (\boldsymbol{f}_n^{(k-1)} \oplus \boldsymbol{d}_n^{(k-1)}) \quad (4)$$

where $\boldsymbol{\theta}_d$ denotes the trainable parameters for the motion estimation. This novel propagation rule allows to aggregate the information contained in the neighbouring nodes to refine the motion estimation of nodes with unknown motion. In the next section, we detail how this GCN is embedded into a VAE framework in order to sample possible object motions.

Overall architecture for motion sampling. Our GCN is embedded into a VAE framework composed of an encoder and a decoder network. At training time, we employ an encoder and a decoder while only the decoder is used at test time, as illustrated in Fig. 2-Right. Note that the features $\boldsymbol{f}_n$ condition both the encoder and the decoder. The goal of the encoder network is not map the input value $\boldsymbol{d}_n$ of every node to a latent space $\boldsymbol{z}_n$. This encoder is implemented using a GCN that employs the propagation rule described in Sec 3.1 and receives as input $\boldsymbol{f}_n \oplus \boldsymbol{d}_n$ for every node. For every node, the latent variable $\boldsymbol{z}_n$ is given by $\boldsymbol{f}_n^{(k)}$ after the last message propagation update. We assume $\boldsymbol{z}_n$ follows a unit Gaussian distribution ($\boldsymbol{z}_n \sim \mathcal{N}(0,1)$). The decoder network receives as input the randomly sampled latent variable $\boldsymbol{z}_n$ for the nodes with unknown motion (*i.e.* $u_n = 0$) and is trained to reconstruct the input motion $\boldsymbol{d}_n$. The decoder is implemented with another GCN with the same propagation rules and with inputs $\boldsymbol{f}_n^{(0)} \oplus \boldsymbol{d}_n^{(0)}$ where $\boldsymbol{f}_n^{(0)} = \boldsymbol{f}_n$ and:

$$\boldsymbol{d}_n^{(0)} = \begin{cases} \text{FC}(\boldsymbol{z}_n) & \text{if } u_n = 0 \\ \sum_{m=1}^{M} \mathbb{1}(i_m = n)\boldsymbol{\delta}_m & \text{if } u_n = 1. \end{cases} \quad (5)$$

where $\mathbb{1}$ denotes the indicator function and FC(.) denotes a fully-connected layer that projects the sampled latent variable $\boldsymbol{z}_n$ to the space of $\boldsymbol{d}_n$ (*i.e.* $\mathbb{R}^2$). Intuitively, the sum in (5) iterates over all the objects in $\mathcal{M}$ to select the corresponding motion vector provided by the user.

At test time, the GCN encoder is not used. The latent variable $\boldsymbol{z}_n$ is sampled according to our unit Gaussian prior distribution for every object with unknown motion and forwarded to the decoder. The decoder outputs the 2D motion of every object in the scene.

3.2. Motion encoding

This module is in charge of predicting the optical flows and the occlusion maps between the initial frame $\mathbf{X}_0$ and every frame that has to be generated. To this aim, for every time step t, we compute a binary tensor $\mathbf{B}_t \in \{0,1\}^{H\times W}$ that specifies the locations of the objects in the scene. At time $t=0$, the object-location map $\mathbf{B}_0$ is computed from the instance segmentation map $\mathbf{I}_0$:

$$\forall (i,j) \in H \times W, \mathbf{B}_0[i,j] = \sum_{n}^{N} \mathbb{1}(\mathbf{I}_0[i,j] = n). \qquad (6)$$

For $t > 0$, $\mathbf{B}_t$ cannot be estimated with the previous equation since $\mathbf{I}_t$ is not known at test time. Instead, we consider a simple rigid model for every object and obtain $\mathbf{B}_t$ by warping $\mathbf{B}_0$ according to the the object motion $\boldsymbol{d}_t$. At training time, $\boldsymbol{d}_t$ is estimated from the segmentation maps while, at test time, we employ $\hat{\boldsymbol{d}}_t$, which is the displacement predicted by our GCN. Finally, this object-location tensor is mapped to a latent tensor $\boldsymbol{z}_s$ via an encoder E_S.

Note that, the output video cannot be fully encoded via the initial frame and the motion of each object since there exist other sources of variability such as the appearance of new objects or change in object sizes. Therefore, we introduce a latent motion variable $\boldsymbol{z}_m$ that encodes all the motion information that cannot be described by $\boldsymbol{z}_s$ and $\boldsymbol{z}_a$. We employ an auto-encoder strategy at training time, estimating $\boldsymbol{z}_m$ from the complete video sequence with an encoder E_M. More precisely, E_M receives as input the concatenation of all the video frames, the instance segmentation maps S_0 and I_0, and the optical flow for every frame. At test time, the latent motion code $\boldsymbol{z}_m$ is sampled according to the prior distribution (*i.e.* $\boldsymbol{z}_m \sim \mathcal{N}(0,I)$).

Finally, we provide the latent variables $\boldsymbol{z}_a$, $\boldsymbol{z}_s$ and $\boldsymbol{z}_m$ to the same decoder, which outputs the bi-directional optical flows and occlusion maps. More precisely, the decoder outputs the forward and the backward optical flow at every time steps denoted by $\mathbf{F}_t^f$ and $\mathbf{F}_t^b$ respectively and the corresponding occlusion maps $\mathbf{O}_t^f$ and $\mathbf{O}_t^b$. Note that the backward optical flows and occlusion maps are then provided to the generation modules, while the forward optical flow and occlusion maps are used only for loss computation.

3.3. Generation module and training objectives

We employ a generation module inspired by [34]. After two down-sampling convolutional blocks applied on the initial frame $\mathbf{X}_0$, we obtain a feature map. We proceed independently for every frame to generate and warp the feature map according to the optical flow predicted by the motion module. Then we multiply the warped feature map by the occlusion map predicted by the occlusion estimator to diminish the impact of the features corresponding to the occluded parts. Finally, the masked feature maps are fed to a subsequent network to output the generated video. This network is composed of several residual blocks, followed by two up-sampling convolutional blocks.

Objective functions. Our GCN framework employs the evidence lower bound of the VAE framework. It is composed of a reconstruction term on the predicted motion vector and the Kullback-Leibler divergence (KL) between the conditional distribution of $\boldsymbol{z}_n$ and its unit Gaussian prior:

$$\mathcal{L}_{VAE} = \frac{1}{N}\sum_{n=0}^{N} \|\boldsymbol{d}_n - \hat{\boldsymbol{d}}_n\|_1 - \mathcal{D}_{KL}(\boldsymbol{z}_n \| \mathcal{N}(0,I)), \quad (7)$$

where $\hat{\boldsymbol{d}}_n$ is the displacement predicted by the GCN.

Forward-backward Consistency. Similarly to [31], we ensure the cycle consistency between forward and backward optical flows. More precisely, for every non-occluded pixel location $\boldsymbol{p}$, we minimize the L_1 distance between the corresponding optical flows:

$$\begin{aligned}\mathcal{L}_{Fc}(F^f, F^b) = \frac{1}{T}\sum_{i=1}^{T}\sum_{\boldsymbol{p}} \mathbf{O}_t^f(\boldsymbol{p})|\mathbf{F}_t^f(\boldsymbol{p}) - \mathbf{F}_t^b(\boldsymbol{p} + \mathbf{F}_t^f(\boldsymbol{p}))|_1 \\ + \mathbf{O}_t^b(\boldsymbol{p})|\mathbf{F}_t^b(\boldsymbol{p}) - \mathbf{F}_t^f(\boldsymbol{p} + \mathbf{F}_t^b(\boldsymbol{p}))|_1\end{aligned} \quad (8)$$

Smoothness. Following [32], we employ a smoothness loss that penalizes high gradient values in the optical-flow map that do not correspond to high-gradient values in the image $\mathbf{X}_0$ (for more details refer to [32]).

Supervised flow. To improve the quality of the generated videos in our multi-objects setting, we take advantage of a pre-trained FlowNet2 [15] network for optical flow and occlusion estimation. FlowNet2 provides high quality optical flow maps that we use as supervision for our motion decoder network using a standard L1 loss.

Motion Encoding uncertainty. To allow the sampling of $\boldsymbol{z}_m$ at test time, the output of the motion encoder E_M is mapped to a unit Gaussian distribution via the KL-divergence:

$$\mathcal{L}_m = -\mathcal{D}_{KL}(z_m \| \mathcal{N}(0,I)) \qquad (9)$$

Generation module. The generation module is trained using state-of-the-art losses for video generation. Following [16, 25, 41] we adopt a PatchGAN discriminator trained with a Least Square loss. For the generator, we apply the structural similarity loss [42], the perceptual loss [17], feature matching loss [41], and a standard pixel-level reconstruction L1 loss.

4. Experiments

Datasets. We evaluate our model with two publicly available datasets, namely Cityscapes and KITTI 360.

- *Cityscapes* [7] provides videos at 17 Frames Per Second (FPS) of European urban scenes. We resize all images to 256 × 128 resolution for performance reasons. The dataset contains 2975 video sequences for training and 500 video sequences for testing. Since Cityscapes does not provides instance and semantic segmentations for the video sequences, we used [5] to generate them.
- *KITTI 360* [46] provides a richly annotated videos at 11 FPS in German suburban areas. We resize all images to 192×64 resolution. The dataset for our evaluation contains 6941 training videos and 423 test sequences. We aggregate the segmentation categories to match the 19 classes of Cityscapes.

Baselines. We compare with the state-of-the-art model for video generation in complex scenarios, *i.e.* Sheng *et al.* [32], which can generate high-quality videos from a staring frame and its associated semantic segmentation map. Since Sheng *et al.* [32] is not able to generate videos controlling object positions, we modify it by including the object location tensor $\mathbf{B}_t$ into the appearance encoder of the original model. We call this model *Sheng**. For a fair comparison, we also test our approach with a variant of the method of Sheng *et al.*, referred to as *S. Sheng**, where we add our *Supervised flow* loss that uses the supervision of a pretrained network in order to improve optical flow prediction. We note that Sheng *et al.* [32] is an extension of Pan *et al.* [28] and that these two works correspond to the same method. Thus, Pan *et al.* [28] is not included in our comparison. It is also worth that Hao *et al.* [12] is not included in the baselines, as it focuses on image generation and does not *explicitly* model the semantic space. Thus, it would be unfair to compare Hao *et al.* with our method on the temporal consistency and object displacements in videos.

Settings. We design three test settings to evaluate our proposal extensively.

- *Oracle (O).* For each video, we select a random object that has to be moved, we feed the networks with the ground truth displacements between the first and last frames, and let the models generate the video. This setting evaluates the network capacity to benefit from the given sparse motion information.
- *Custom.* For each input video, we select a random object that has to be moved, we feed the networks with displacement shifted by $\lambda = 1.5$ (i.e. $\boldsymbol{d}'_n = \lambda \boldsymbol{d}_n$) and let the models generate the video. This setting evaluates the network capacity to condition the video on sparse motion inputs, which are different from the ground truth.

Then, we also experiment a drastic scenario where *all* the objects are moved following the *Custom*. In this experiments, all future positions are provided as input. In this experiment, the GCN can be by-passed since $u_n = 1$ for every object. This experiments differ from *Ground truth* and *Custom* where our GCN has to infer the plausible future positions of all the objects that are not provided by the user. In all our experiments, we generate 5 future frames starting from the provided initial frame.

Evaluation metrics.

- *FVD.* We adopt the Fréchet video distance (FVD) metric [37] to evaluate both the video quality and temporal consistency of generated frames. We compute the FVD between the ground truth test videos and the generated ones. The lower the FVD, the better.
- *NDE.* We measure the adherence of generated videos with the user-provided motions by computing the Normalised Displacement Error (NDE) as the Euclidean distance between the coordinate specified by the user and the coordinate where the object ends-up in the generated video, which is then normalised Euclidean distance of the ground truth starting coordinate and the ending one. All object's positions are detected through YOLOv3 [30]. We discard the objects that cannot be detected in the ground truth videos due to the resolution of videos, or because objects are too small to be correctly detected by YOLOv3. The lower the NDE, the better.
- *Acc.* The object's positions in generated videos can be difficult to track due to the presence of artifacts, occlusions and low-quality images. Thus, we report here the Accuracy (Acc) of the YOLOv3 detector in generated videos. The higher the Accuracy, the better.

Model	**FVD↓**	**NDE↓**	**Acc↑**
A: Our proposal	288	1.01	0.84
B: (A) w/o GCN	369	1.42	0.70
C: (A) w/o Obj. Interactions	375	1.38	0.76
D: (A) w/o Sup.	301	1.13	0.84

Table 1. Ablation study results on Cityscapes.

4.1. Ablation Study

We conduct an ablation study on Cityscapes to evaluate the impact of the individual components of the model. We begin by testing the contribution of our GCN by removing the motion estimation module and directly use the object location tensor of the user-controlled object $\mathbf{B}_t$ in the appearance encoder. Table 1-B shows that removing the motion estimator leads to a drop in all three metrics. Without the GCN, the network cannot infer the positions of the objects in the scene and fails at moving the object. The quality of the video decreases as well (FVD 369 vs FVD 289).

Setting (N)	**Model**	**Cityscapes**			**KITTI 360**		
		FVD↓	NDE↓	Acc↑	FVD↓	NDE↓	Acc↑
Oracle (1)	Sheng [32]	373	2.11	0.68	**443**	3.92	0.68
	Sheng*	498	2.12	0.58	507	3.66	0.66
	S. Sheng*	493	1.78	0.57	527	3.79	0.33
	Ours	**288**	**1.01**	**0.84**	463	**1.83**	**0.75**
Custom (1)	Sheng [32]	373	1.53	0.66	**443**	3.98	0.62
	Sheng*	498	1.61	0.57	506	3.27	0.60
	S. Sheng*	493	1.41	0.59	527	3.34	0.30
	Ours	**303**	**0.66**	**0.88**	470	**2.06**	**0.81**
Custom (all)	Sheng [32]	373	1.48	0.73	**443**	2.93	0.48
	Sheng*	498	1.47	0.67	506	3.19	0.49
	S. Sheng*	493	1.38	0.60	527	2.71	0.24
	Ours	**321**	**0.96**	**0.86**	464	**1.58**	**0.72**

Table 2. Quantitative comparison in the *Oracle* and *Custom* setting. N is the number of user-controlled objects. $N = 1$ selects one object at random

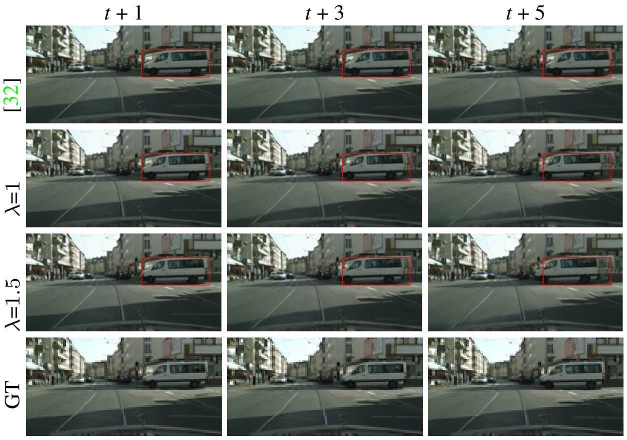

Figure 3. Qualitative comparison in the *Custom* setting on the Cityscapes dataset with ground truth reference. The position of the moved object at $t = 0$ is highlighted in red. Zoom for details.

Then, we test a version of the GCNs that does not model the interactions between objects. To do so, we remove all the edges between the nodes of the GCN, thus considering each object as independent. Tab. 1-C shows that, while the object is correctly moved (NDE and Acc are similar to A), the video quality is considerably worse. In the Supplementary Material, we qualitatively show that the network cannot move the other objects consistently.

Finally, we also test the network without flow supervision (*i.e*. Tab. 1-D). As expected, the performance decreases in NDE and FVD. Nevertheless, the quality of the image quality measured with FVD remains higher than when we do not model object interactions (*i.e*. Tab. 1-C).

4.2. Comparison with State-of-the-art

Quantitative comparison. We compare our method with the method of Sheng *et al.* [32], and its modifications, namely *Sheng** and *S. Sheng**. To the best of our knowledge, the method of Sheng *et al.* [32] model is the most similar work that generates videos in complex environments also leveraging the semantic space of frames.

Tab. 2 shows the quantitative evaluation of all the models. We first compare our proposal in the *Oracle* setting, where the displacement $\boldsymbol{d}_n$ of one random object n is computed from ground truth frames. From NDE and Acc, we observe that our approach consistently outperforms state-of-the-art methods by enabling the user to move objects more precisely in both the datasets (see Tab. 2-Oracle results). In particular, NDE decreases from Sheng *et al.* [32] results by 47% and 53% in Cityscapes and KITTI 360, respectively. Regarding the video quality, which evaluates both the temporal consistency and image quality, we significantly improve the state-of-the-art performance in Cityscapes (FVD decreases by 22.79% from [32]), while in KITTI 360 we are slightly worse than Sheng *et al.* [32]. We hypothesize this result is caused by the low frame rate of KITTI 360, which rewards Sheng *et al.* [32] that is dominated by modelling only the ego-motion, while ignoring other objects' movements. Through *Sheng** results, we note that adding the information to move the objects in the scene helps the baseline through NDE, but the video quality decreases significantly. Only through additional supervision (i.e. *S. Sheng**), FVD partially decreases. However, our model is far better at moving the objects in the scene, while having better video quality than *Sheng** and *S. Sheng**.

Tab. 2-*Custom* also shows the *Custom* experiment, where $\boldsymbol{d}_i$ is multiplied by $\lambda = 1.5$ from the ground truth displacement. Again, we observe that our proposal offers better control of the object's movements compared to state-of-the-art approaches and improves video quality. Our approach moves objects to positions that differ from the ground truth, showing that the *Motion Encoding* module correctly follows the user inputs, infers the missing objects and composes them in a temporally consistent manner.

We also perform experiments where we ask the models to move *all* objects (i.e. N objects) with the *Custom* setting. The last rows of Tab. 2 show that our proposal achieves the best results even at this "drastic" task.

Finally, we note that, our approach without supervised optical flow (Tab.1-D) outperforms the existing approaches compared in Tab. 2 both in terms of video quality and object control. This result confirms that the performance of our approach are not due to our use of supervision for optical flow but rather to our architecture.

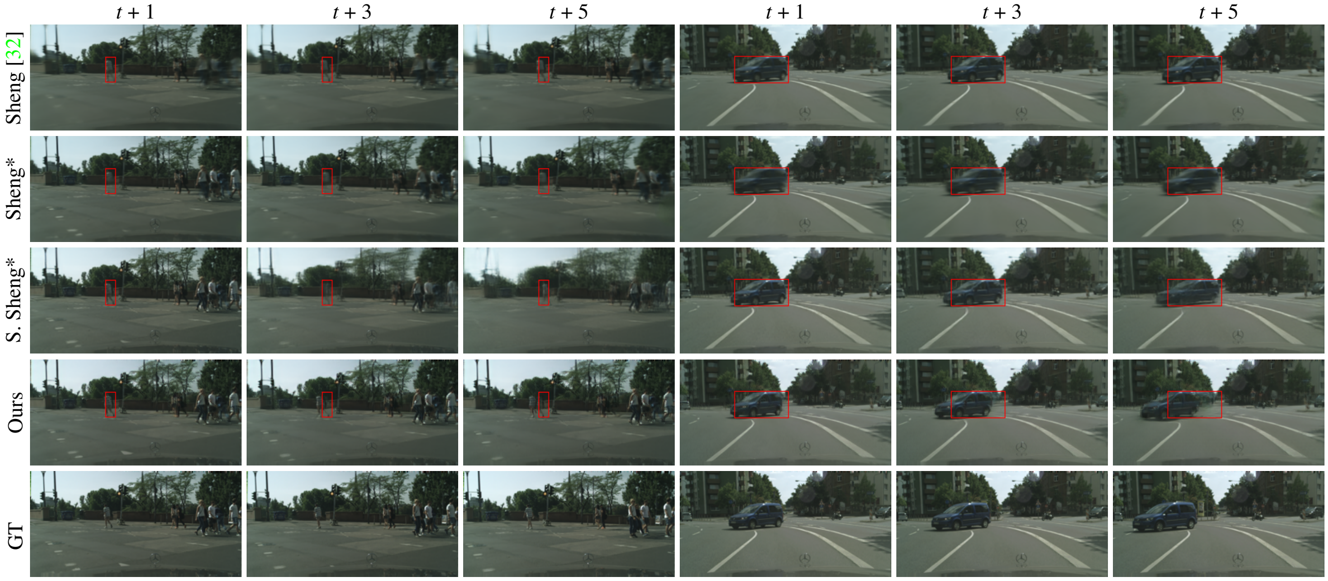

Figure 4. Results of predicting the frames t + 1, t + 3 , and t + 5 on the Cityscapes dataset [7] with ground truth reference. On first three columns, we move the pedestrian near the semaphore to left. On the last three columns we move car crossing the street. The position of the moved object at $t = 0$ is highlighted in red. Zoom for details.

Qualitative comparison. We now report the qualitative comparison for the tested models. Fig.4 shows the results of two groups of experiments, where we feed the network with two different initial frames. In the first group of images, we want to move to the left the pedestrian that in the ground truth is in the position highlighted with the red bounding box. All three baselines fail to move the pedestrian. Sheng *et al.* [32] only moves the ego vehicle slightly to the right, leaving the pedestrian in the same position, while moving the entire scene. *Sheng** and *S. Sheng** moves the ego vehicle forward but fail at moving the pedestrian, which stays exactly in the same position in all the frames. C2M, instead, correctly and gradually moves to the left of the pedestrian, which goes out the red bounding box.

In the second group of images in the last three columns of Fig.4, we aim to move a car that in the ground truth was in the position highlighted in red. Sheng *et al.* [32] can only move the ego-motion forward while the car remains in the same starting position. The other two baselines slightly move the car but not to the desired position specified by the user. However, our proposal significantly moves the car to the left, which goes partially out from the bounding box, while changing very little in the ego-motion of the video.

Finally, Fig.3 shows a qualitative example of how our model can modify the van's position with different displacement. Moving it to the ground truth position ($\lambda = 1$) and to custom coordinates ($\lambda = 1.5$). As seen in the previous experiment, the baseline fails at moving the white van to the left. Instead, it stretches the back of the van. In contrast, with $\lambda = 1$ and $\lambda = 1.5$ the van goes from the bounding box with different horizontal shifts, depicting that our network can correctly change the position of the van to the user-specified positions.

5. Conclusions

In this work, we introduce *Click to Move*, a framework for video generation that allows the user to select key objects in the scene and control their motion by specifying their position in the last video frame. At test time, our approach receives the initial frame and the corresponding instance segmentation maps to generate a video that starts from the provided frame and respects the object motion constraints specified by the user. Objects in a scene are often not independent one from another. Thus, we introduce a novel GCN framework that employs specific message-passing rules to model object interaction while accounting for the user inputs. Experimentally, we demonstrate that our method outperforms state-of-the-art approaches and that the proposed GCN architecture allows better motion control. As future works, we plan to extend our approach to allow the generation of videos with variable length.

6. Acknowledgments

E. R. acknowledges financial support from the EU project PROTECTOR: Protecting Places of Worship. This work was carried out under the *Vision and Learning joint Laboratory* between FBK and University of Trento.

References

[1] Photoshop: Now the world's most advanced AI application for creatives. https://tinyurl.com/yzg97uaq. Accessed: 2021-02-21. 1

[2] Pierfrancesco Ardino, Yahui Liu, Elisa Ricci, Bruno Lepri, and Marco De Nadai. Semantic-guided inpainting network for complex urban scenes manipulation. In *ICPR*, 2021. 2

[3] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *ICCV*, pages 5933–5942, 2019. 2

[4] Lele Chen, Ross K Maddox, Zhiyao Duan, and Chenliang Xu. Hierarchical cross-modal talking face generation with dynamic pixel-wise loss. In *CVPR*, pages 7832–7841, 2019. 2

[5] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, pages 12475–12485, 2020. 6

[6] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *CVPR*, 2020. 1

[7] 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. 6, 8

[8] Haoye Dong, Xiaodan Liang, Yixuan Zhang, Xujie Zhang, Xiaohui Shen, Zhenyu Xie, Bowen Wu, and Jian Yin. Fashion editing with adversarial parsing learning. In *CVPR*, pages 8120–8128, 2020. 1

[9] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2016. 2

[10] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *arXiv preprint arXiv:1605.07157*, 2016. 2

[11] Jean-Yves Franceschi, Edouard Delasalles, Mickael Chen, Sylvain Lamprier, and P. Gallinari. Stochastic latent residual video prediction. *ArXiv*, abs/2002.09219, 2020. 2

[12] Zekun Hao, Xun Huang, and Serge Belongie. Controllable video generation with sparse trajectories. In *CVPR*, pages 7854–7863, 2018. 2, 3, 6

[13] Jiawei He, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal. Probabilistic video generation using holistic attribute control. In *ECCV*, pages 452–467, 2018. 1

[14] Yung-Han Ho, Chuan-Yuan Cho, Wen-Hsiao Peng, and Guo-Lun Jin. Sme-net: Sparse motion estimation for parametric video prediction through reinforcement learning. In *ICCV*, pages 10462–10470, 2019. 2

[15] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, pages 2462–2470, 2017. 5

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 1125–1134, 2017. 5

[17] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711. Springer, 2016. 5

[18] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A conditional flow-based model for stochastic video generation. In *ICLR*, 2020. 2

[19] Alex X. Lee, Richard Zhang, Frederik Ebert, P. Abbeel, Chelsea Finn, and S. Levine. Stochastic adversarial video prediction. *ArXiv*, abs/1804.01523, 2018. 2

[20] Donghoon Lee, Sifei Liu, Jinwei Gu, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Context-aware synthesis and placement of object instances. *arXiv preprint arXiv:1812.02350*, 2018. 2

[21] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Flow-grounded spatial-temporal video prediction from still images. In *ECCV*, pages 600–615, 2018. 2

[22] Yitong Li, Martin Min, Dinghan Shen, David Carlson, and Lawrence Carin. Video generation from text. In *AAAI*, volume 32, 2018. 1

[23] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P Xing. Dual motion gan for future-flow embedded video prediction. In *ICCV*, pages 1744–1752, 2017. 2

[24] Yahui Liu, Enver Sangineto, Yajing Chen, Linchao Bao, Haoxian Zhang, Nicu Sebe, Bruno Lepri, Wei Wang, and Marco De Nadai. Smoothing the disentangled latent style space for unsupervised image-to-image translation. In *CVPR*, pages 10785–10794, June 2021. 1

[25] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *ICCV*, pages 2794–2802, 2017. 5

[26] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015. 2

[27] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Z Qureshi, and Mehran Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning. *arXiv preprint arXiv:1901.00212*, 2019. 1

[28] Junting Pan, Chengyu Wang, Xu Jia, Jing Shao, Lu Sheng, Junjie Yan, and Xiaogang Wang. Video generation from single semantic label map. In *CVPR*, pages 3733–3742, 2019. 2, 3, 6

[29] Fitsum A Reda, Guilin Liu, Kevin J Shih, Robert Kirby, Jon Barker, David Tarjan, Andrew Tao, and Bryan Catanzaro. Sdc-net: Video prediction using spatially-displaced convolution. In *ECCV*, pages 718–733, 2018. 2

[30] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 6

[31] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *CVPR*, pages 9243–9252, 2020. 1, 5

[32] 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. 2, 3, 5, 6, 7, 8

[33] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In *CVPR*, pages 2377–2386, 2019. 1, 2

[34] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Neurips*, 2019. 1, 2, 5

[35] Yukitaka Tsuchiya, Takahiro Itazuri, Ryota Natsume, Shintaro Yamamoto, Takuya Kato, and Shigeo Morishima. Generating video from single image and sound. In *CVPR Workshops*, pages 17–20, 2019. 1

[36] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *CVPR*, pages 1526–1535, 2018. 1, 2

[37] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. Fvd: A new metric for video generation. 2019. 6

[38] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. In *ICML*, pages 3560–3569. PMLR, 2017. 2

[39] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating the future by watching unlabeled video. *arXiv preprint arXiv:1504.08023*, 2, 2015. 2

[40] Carl Vondrick and Antonio Torralba. Generating the future with adversarial transformers. In *CVPR*, pages 1020–1028, 2017. 1

[41] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, pages 8798–8807, 2018. 5

[42] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 5

[43] Olivia Wiles, A Koepke, and Andrew Zisserman. X2face: A network for controlling face generation using images, audio, and pose codes. In *ECCV*, pages 670–686, 2018. 1, 2

[44] Yue Wu, Rongrong Gao, Jaesik Park, and Qifeng Chen. Future video synthesis with object motion prediction. In *CVPR*, pages 5539–5548, 2020. 3

[45] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020. 3

[46] 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

[47] Yufei Ye, Maneesh Singh, Abhinav Gupta, and Shubham Tulsiani. Compositional video prediction. In *ICCV*, pages 10353–10362, 2019. 2

[48] Tao Yu, Zongyu Guo, Xin Jin, Shilin Wu, Zhibo Chen, Weiping Li, Zhizheng Zhang, and Sen Liu. Region normalization for image inpainting. In *AAAI*, 2020. 1

[49] Jiangning Zhang, Chao Xu, Liang Liu, Mengmeng Wang, Xia Wu, Yong Liu, and Yunliang Jiang. Dtvnet: Dynamic time-lapse video generation via single still image. In *ECCV*, pages 300–315. Springer, 2020. 2

[50] Yang Zhou, Xintong Han, Eli Shechtman, Jose Echevarria, Evangelos Kalogerakis, and Dingzeyu Li. Makelttalk: speaker-aware talking-head animation. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020. 2