Animating Arbitrary Objects via Deep Motion Transfer

Aliaksandr Siarohin[1], Stéphane Lathuilière[1], Sergey Tulyakov[2], Elisa Ricci[1,3] and Nicu Sebe[1,4]
[1]DISI, University of Trento, Italy, [2] Snap Inc., Santa Monica, CA,
[3] Fondazione Bruno Kessler (FBK), Trento, Italy,
[4]Huawei Technologies Ireland, Dublin, Ireland
{aliaksandr.siarohin,stephane.lathuilire,e.ricci,niculae.sebe}@unitn.it, stulyakov@snap.com

Abstract

This paper introduces a novel deep learning framework for image animation. Given an input image with a target object and a driving video sequence depicting a moving object, our framework generates a video in which the target object is animated according to the driving sequence. This is achieved through a deep architecture that decouples appearance and motion information. Our framework consists of three main modules: (i) a Keypoint Detector unsupervisely trained to extract object keypoints, (ii) a Dense Motion prediction network for generating dense heatmaps from sparse keypoints, in order to better encode motion information and (iii) a Motion Transfer Network, which uses the motion heatmaps and appearance information extracted from the input image to synthesize the output frames. We demonstrate the effectiveness of our method on several benchmark datasets, spanning a wide variety of object appearances, and show that our approach outperforms state-of-the-art image animation and video generation methods. Our source code is publicly available [1].

1. Introduction

This paper introduces a framework for motion-driven image animation to automatically generate videos by combining the appearance information derived from a *source image* (*e.g.* depicting the face or the body silhouette of a certain person) with motion patterns extracted from a *driving video* (*e.g.* encoding the facial expressions or the body movements of another person). Several examples are given in Fig. 1. Generating high-quality videos from static images is challenging, as it requires learning an appropriate representation of an object, such as a 3D model of a face or a human body. This task also requires accurately extracting the motion patterns from the driving video and mapping them on the object representation. Most approaches are object-specific, using techniques from computer graphics [7, 38]. These methods also use an explicit object representation, such as a 3D morphable model [5], to facilitate animation, and therefore only consider faces.

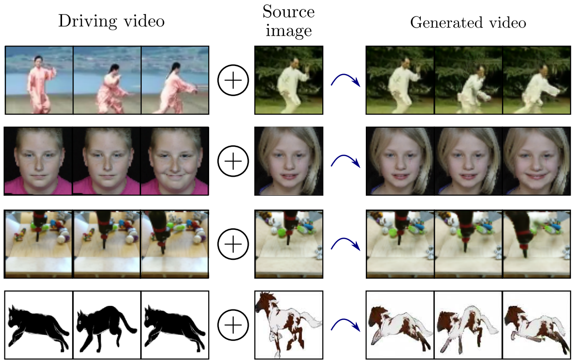


Figure 1: Our deep motion transfer approach can animate arbitrary objects following the motion of the driving video.

Over the past few years, researchers have developed approaches for automatic synthesis and enhancement of visual data. Several methods derived from Generative Adversarial Networks (GAN) [16] and Variational Autoencoders (VAE) [24] have been proposed to generate images and videos [19, 32, 30, 39, 29, 37, 36, 33]. These approaches use additional information such as conditioning labels (*e.g.* indicating a facial expression, a body pose) [45, 31, 15, 35]. More specifically, they are purely data-driven, leveraging a large collection of training data to learn a latent representation of the visual inputs for synthesis. Noting the significant progress of these techniques, recent research studies have started exploring the use of deep generative models for image animation and video retargeting [46, 9, 4, 43, 3]. These works demonstrate that deep models can effectively transfer motion patterns between human subjects in videos [4], or transfer a facial expression from one person to another [46]. However, these approaches have limitations: for example, they rely on pre-trained models for extracting object representations that require costly ground-truth data annotations [9, 43, 3]. Furthermore, these works do not address the problem of animating arbitrary objects: instead, consid-

[1] https://github.com/AliaksandrSiarohin/monkey-net

ering a single object category [46] or learning to translate videos from one specific domain to another [4, 22].

This paper addresses some of these limitations by introducing a novel deep learning framework for animating a static image using a driving video. Inspired by [46], we propose learning a latent representation of an object category in a self-supervised way, leveraging a large collection of video sequences. There are two key distinctions between our work and [46]. Firstly, our approach is not designed for specific object category, but rather is effective in animating arbitrary objects. Secondly, we introduce a novel strategy to model and transfer motion information, using a set of sparse motion-specific keypoints that were learned in an unsupervised way to describe relative pixel movements. Our intuition is that only relevant motion patterns (derived from the driving video) must be transferred for object animation, while other information should not be used. We call the proposed deep framework Monkey-Net, as it enables motion transfer by considering MOviNg KEYpoints.

We demonstrate the effectiveness of our framework by conducting an extensive experimental evaluation on three publicly available datasets, previously used for video generation: the Tai-Chi [39], the BAIR robot pushing [11] and the UvA-NEMO Smile [10] datasets. As shown in our experiments, our image animation method produces high quality videos for a wide range of objects. Furthermore, our quantitative results clearly show that our approach outperforms state-of-the-art methods for image-to-video translation tasks.

2. Related work

Deep Video Generation. Early deep learning-based approaches for video generation proposed synthesizing videos by using spatio-temporal networks. Vondrick *et al.* [42] introduced VGAN, a 3D convolutional GAN which simultaneously generates all the frames of the target video. Similarly, Saito *et al.* [30] proposed TGAN, a GAN-based model which is able to generate multiple frames at the same time. However, the visual quality of these methods outputs is typically poor.

More recent video generation approaches used recurrent neural networks within an adversarial training framework. For instance, Wang *et al.* [45] introduced a Conditional MultiMode Network (CMM-Net), a deep architecture which adopts a conditional Long-Short Term Memory (LSTM) network and a VAE to generate face videos. Tulyakov *et al.* [39] proposed MoCoGAN, a deep architecture based on a recurrent neural network trained with an adversarial learning scheme. These approaches can take conditional information as input that comprises categorical labels or static images and, as a result, produces high quality video frames of desired actions.

Video generation is closely related to the future frame prediction problem addressed in [34, 26, 13, 40, 48]. Given a video sequence, these methods aim to synthesize a sequence of images which represents a coherent continuation of the given video. Earlier methods [34, 26, 23] attempted to directly predict the raw pixel values in future frames. Other approaches [13, 40, 2] proposed learning the transformations which map the pixels in the given frames to the future frames. Recently, Villegas *et al.* [41] introduced a hierarchical video prediction model consisting of two stages: it first predicts the motion of a set of landmarks using an LSTM, then generates images from the landmarks.

Our approach is closely related to these previous works since we also aim to generate video sequences by using a deep learning architecture. However, we tackle a more challenging task: image animation requires decoupling and modeling motion and content information, as well as a recombining them.

Object Animation. Over the years, the problems of image animation and video re-targeting have attracted attention from many researchers in the fields of computer vision, computer graphics and multimedia. Traditional approaches [7, 38] are designed for specific domains, as they operate only on faces, human silhouettes, *etc*. In this case, an explicit representation of the object of interest is required to generate an animated face corresponding to a certain person's appearance, but with the facial expressions of another. For instance, 3D morphable models [5] have been traditionally used for face animation [49]. While especially accurate, these methods are highly domain-specific and their performance drastically degrades in challenging situations, such as in the presence of occlusions.

Image animation from a driving video can be interpreted as the problem of transferring motion information from one domain to another. Bansal *et al.* [4] proposed Recycle-GAN, an approach which extends conditional GAN by incorporating spatio-temporal cues in order to generate a video in one domain given a video in another domain. However, their approach only learns the association between two specific domains, while we want to animate an image depicting one object without knowing at training time which object will be used in the driving video. Similarly, Chan *et al.* [9] addressed the problem of motion transfer, casting it within a per-frame image-to-image translation framework. They also proposed incorporating spatio-temporal constraints. The importance of considering temporal dynamics for video synthesis was also demonstrated in [43]. Wiles *et al.* [46] introduced X2Face, a deep architecture which, given an input image of a face, modifies it according to the motion patterns derived from another face or another modality, such as audio. They demonstrated that a purely data-driven deep learning-based approach is effective in animating still images of faces without demanding explicit 3D representation. In this work, we design a self-supervised deep network for animating static images, which is effective for generating arbitrary objects.

3. Monkey-Net

The architecture of the Monkey-Net is given in Fig. 2. We now describe it in detail.

3.1. Overview and Motivation

The objective of this work is to animate an object based on the motion of a similar object in a driving video. Our framework is articulated into three main modules (Fig. 2). The first network, named Keypoint Detector, takes as input the source image and a frame from the driving video and automatically extracts sparse keypoints. The output of this module is then fed to a Dense Motion prediction network, which translates the sparse keypoints into motion heatmaps. The third module, the Motion Transfer network, receives as input the source image and the dense motion heatmap and recombines them producing a target frame.

The output video is generated frame-by-frame as illustrated in Fig. 2.a. At time t, the Monkey-Net uses the source image and the t^{th} frame from the driving video. In order to train a Monkey-Net one just needs a dataset consisting of videos of objects of interest. No specific labels, such as keypoint annotations, are required. The learning process is fully self-supervised. Therefore, at test time, in order to generate a video sequence, the generator requires only a static input image and a motion descriptor from the driving sequence. Inspired by recent studies on unsupervised landmark discovery for learning image representations [20, 47], we formulate the problem of learning a motion representation as an unsupervised motion-specific keypoint detection task. Indeed, the keypoints locations differences between two frames can be seen as a compact motion representation. In this way, our model generates a video by modifying the input image according to the landmarks extracted from the driving frames. Using a Monkey-Net at inference time is detailed in Sec. 3.6.

The Monkey-Net architecture is illustrated in Fig. 2.b. Let $\boldsymbol{x}$ and $\boldsymbol{x}' \in \mathcal{X}$ be two frames of size $H \times W$ extracted from the same video. The $H \times W$ lattice is denoted by $\mathcal{U}$. Inspired by [20], we jointly learn a keypoint detector Δ together with a generator network G according to the following objective: G should be able to reconstruct $\boldsymbol{x}'$ from the keypoint locations $\Delta(\boldsymbol{x}) \in \mathcal{U}$, $\Delta(\boldsymbol{x}') \in \mathcal{U}$, and $\boldsymbol{x}$. In this formulation, the motion between $\boldsymbol{x}$ and $\boldsymbol{x}'$ is implicitly modeled. To deal with large motions, we aim to learn keypoints that describe motion as well as the object geometry. To this end, we add a third network M that estimates the optical flow $\mathcal{F} \in \mathbb{R}^{H\times W\times 2}$ between $\boldsymbol{x}'$ and $\boldsymbol{x}$ from $\Delta(\boldsymbol{x})$, $\Delta(\boldsymbol{x}')$ and $\boldsymbol{x}$. The motivation for this is twofold. First, this forces the keypoint detector Δ to predict keypoint locations that capture not only the object structure but also its motion. To do so, the learned keypoints must be located especially on the object parts with high probability of motion. For instance, considering the human body, it is important to obtain keypoints on the extremities (as in feet or hands) in order to describe the body movements correctly, since these body-parts tend to move the most. Second, following common practises in conditional image generation, the generator G is implemented as an encoder-decoder composed of convolutional blocks [19]. However, standard convolutional encoder-decoders are not designed to handle large pixel-to-pixel misalignment between the input and output images [31, 3, 14]. To this aim, we introduce a deformation module within the generator G that employs the estimated optical flow $\mathcal{F}$ in order to align the encoder features with $\boldsymbol{x}'$.

3.2. Unsupervised Keypoint Detection

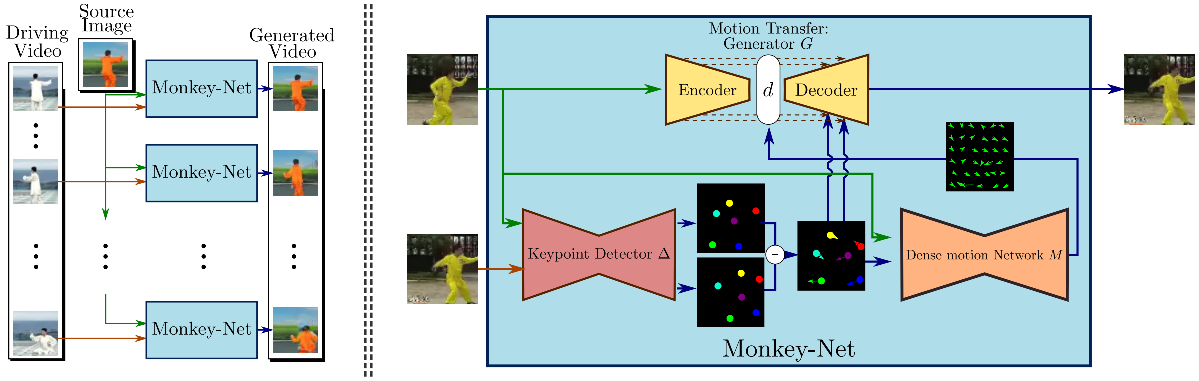
In this section, we detail the structure employed for unsupervised keypoint detection. First, we employ a standard U-Net architecture that, from the input image, estimates K heatmaps $H_k \in [0,1]^{H\times W}$, one for each keypoint. We employ softmax activations for the last layer of the decoder in order to obtain heatmaps that can be interpreted as detection confidence map for each keypoint. An encoder-decoder architecture is used here since it has shown good performance for keypoints localization [6, 27].

To model the keypoint location confidence, we fit a Gaussian on each detection confidence map. Modeling the landmark location by a Gaussian instead of using directly the complete heatmap H_k acts as a bottle-neck layer, and therefore allows the model to learn landmarks in an indirect way. The expected keypoint coordinates $\boldsymbol{h}_k \in \mathbb{R}$ and its covariance Σ_k are estimated according to:

$$\boldsymbol{h}_k = \sum_{p\in\mathcal{U}} H_k[p]p; \; \Sigma_k = \sum_{p\in\mathcal{U}} H_k[p](p-\boldsymbol{h}_k)(p-\boldsymbol{h}_k)^\top \quad (1)$$

The intuition behind the use of keypoint covariances is that they can capture not only the location of a keypoint but also its orientation. Again considering the example of the human body: in the case of the legs, the covariance may capture their orientation. Finally, we encode the keypoint distributions as heatmaps $H_k^i \in [0,1]^{H\times W}$, such that they can be used as inputs to the generator and to the motion networks. Indeed, the advantage of using a heatmap representation, rather than considering directly the 2D coordinates $\boldsymbol{h}_k$, is that heatmaps are compatible with the use of convolutional neural networks. Formally, we employ the following Gaussian-like function:

$$\forall p \in \mathcal{U}, H_k(\mathbf{p}) = \frac{1}{\alpha} exp\left(-(\mathbf{p}-\boldsymbol{h}_k)\Sigma_k^{-1}(\mathbf{p}-\boldsymbol{h}_k)\right) \quad (2)$$



(a) Image Animation

(b) Monkey-Net Architecture for Self-Learned Animation

Figure 2: A schematic representation of the proposed motion transfer framework for image animation. At testing time (Fig. (a)), the model generates a video with the object appearance of the source image but with the motion from the driving video. Monkey-Net (Fig. (b)) is composed of three networks: a motion-specific keypoint detector Δ, a motion prediction network M and an image generator G. G reconstructs the image $\boldsymbol{x}'$ from the keypoint positions $\Delta(\boldsymbol{x})$ and $\Delta(\boldsymbol{x}')$. The optical flow computed by M is used by G to handle misalignments between $\boldsymbol{x}$ and $\boldsymbol{x}'$. The model is learned with a self-supervised learning scheme.

where α is normalization constant. This process is applied independently on $\boldsymbol{x}$ and $\boldsymbol{x}'$ leading to two sets of K keypoints heatmaps $H = \{H_k\}_{k=1..K}$ and $H' = \{H'_k\}_{k=1..K}$.

3.3. Generator Network with Deformation Module

In this section, we detail how we reconstruct the target frame $\boldsymbol{x}'$ from $\boldsymbol{x}$, $\Delta(\boldsymbol{x}) = H$ and $\Delta(\boldsymbol{x}') = H'$. First we employ a standard convolutional encoder composed of a sequence of convolutions and average pooling layers in order to encode the object appearance in $\boldsymbol{x}$. Let $\boldsymbol{\xi}_r \in \mathbb{R}^{H_r \times W_r \times C_r}$ denote the output of the r^{th} block of the encoder network ($1 \leq r \leq R$). The architecture of this generator network is also based on the U-Net architecture [28] in order to obtain better details in the generated image. Motivated by [31], where it was shown that a standard U-net cannot handle large pixel-to-pixel misalignment between the input and the output images, we propose using a deformation module to align the features of the encoder with the output images. Contrary to [31] that defines an affine transformation for each human body part in order to compute the feature deformation, we propose a deformation module that can be used on any object. In particular, we propose employing the optical flow $\mathcal{F}$ to align the features $\boldsymbol{\xi}_r$ with $\boldsymbol{x}'$. The deformation employs a warping function $f_w(\cdot, \cdot)$ that warps the feature maps according to $\mathcal{F}$:

$$\boldsymbol{\xi}'_r = f_w(\boldsymbol{\xi}_r, \mathcal{F}) \tag{3}$$

This warping operation is implemented using a bilinear sampler, resulting in a fully differentiable model. Note that $\mathcal{F}$ is down-sampled to $H_r \times W_r$ via nearest neighbour interpolation when computing Eq. (3). Nevertheless, because of the small receptive field of the bilinear sampling layer, encoding the motion only via the deformation module leads to optimization problems. In order to facilitate network training, we propose inputing the decoder the difference of the keypoint locations encoded as heatmaps $\dot{H} = H' - H$. Indeed, by providing $\dot{H}$ to the decoder, the reconstruction loss applied on the G outputs (see Sec. 3.5) is directly propagated to the keypoint detector Δ without going through M. In addition, the advantage of the heatmap difference representation is that it encodes both the locations and the motions of the keypoints. Similarly to $\mathcal{F}$, we compute R tensors $\dot{H}_r$ by down-sampling $\dot{H}$ to $H_r \times W_r$. The two tensors $\dot{H}_r$ and $\boldsymbol{\xi}'_r$ are concatenated along the channel axis and are then treated as skip-connection tensors by the decoder.

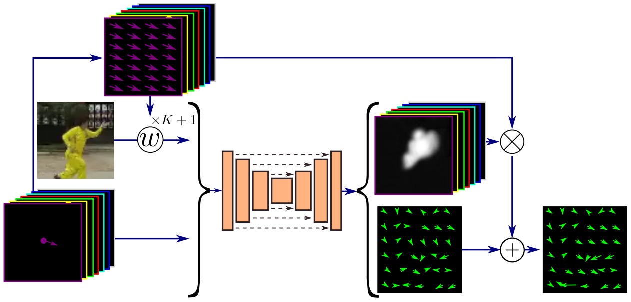


Figure 3: A schematic representation of the adopted part-based model for optical flow estimation from sparse representation. From the appearance of the first frame and the keypoints motion, the network M predicts a mask for each keypoint and the residual motion (see text for details).

3.4. From Sparse Keypoints to Dense Optical Flow

In this section, we detail how we estimate the optical flow $\mathcal{F}$. The task of predicting a dense optical flow only from the displacement of a few keypoints and the appearance of the first frame is challenging. In order to facilitate the task of the network, we adopt a part base formulation. We make the assumption that each keypoint is located on an object part that is locally rigid. Thus, the task of computing the optical flow becomes simpler since, now, the problem consists in estimating masks $M_k \in \mathcal{R}^{H\times W}$ that segment the object in rigid parts corresponding to each keypoint. A first coarse estimation of the optical flow can be given by:

$$\mathcal{F}_{\text{coarse}} = \sum_{k=1}^{K+1} M_k \otimes \rho(h_k) \tag{4}$$

where $\otimes$ denotes the element-wise product and $\rho(\cdot) \in \mathcal{R}^{H\times W\times 2}$ is the operator that returns a tensor by repeating the input vector $H \times W$ times. Additionally, we employ one specific mask M_{K+1} without deformation (which corresponds to $\rho([0,0])$) to capture the static background. In addition to the masks M_k, the motion network M also predicts the residual motion $\mathcal{F}_{\text{residual}}$. The purpose of this residual motion field is to refine the coarse estimation by predicting non-rigid motion that cannot be modeled by the part-based approach. The final estimated optical flow is: $\mathcal{F} = \mathcal{F}_{\text{coarse}} + \mathcal{F}_{\text{residual}}$.

Concerning the inputs of the motion network, M takes two tensors, $\dot{H}$ and $\boldsymbol{x}$ corresponding respectively to the sparse motion and the appearance. However, we can observe that, similarly to the generator network, M may suffer from the misalignment between the input $\boldsymbol{x}$ and the output $\mathcal{F}$. Indeed, $\mathcal{F}$ is aligned with $\boldsymbol{x}'$. To handle this problem, we use the warping operator f_w according to the motion field of each keypoint $\rho(h_k)$, *e.g.* $\boldsymbol{x}_k = f_w(\boldsymbol{x}, \rho(h_k))$. This solution provides images $\boldsymbol{x}_k$ that are locally aligned with $\mathcal{F}$ in the neighborhood of h'_k. Finally, we concatenate $H' - H$, $\{\boldsymbol{x}_k\}_{k=1..K}$ and $\boldsymbol{x}$ along the channel axis and feed them into a standard U-Net network. Similarly to the keypoint and the generator network, the use of U-Net architecture is motivated by the need of fine-grained details.

3.5. Network Training

We propose training the whole network in an end-to-end fashion. As formulated in Sec. 3.1, our loss ensures that $\boldsymbol{x}'$ is correctly reconstructed from $\Delta(\boldsymbol{x}) \in \mathcal{U}$, $\Delta(\boldsymbol{x}') \in \mathcal{U}$ and $\boldsymbol{x}$. Following the recent advances in image generation, we combine an adversarial and the feature matching loss proposed in [44] in order to learn to reconstruct $\boldsymbol{x}'$. More precisely, we use a discriminator network D that takes as input H' concatenated with either the real image $\boldsymbol{x}'$ or the generated image $\hat{\boldsymbol{x}}'$. We employ the least-square GAN formulation [25] leading to the two following losses used to train the discriminator and the generator:

$$\begin{aligned}\mathcal{L}^D_{\text{gan}}(D) =& \mathbb{E}_{\boldsymbol{x}'\in\mathcal{X}}[(D(\boldsymbol{x}' \oplus H') - 1)^2] \\ &+ \mathbb{E}_{(\boldsymbol{x},\boldsymbol{x}')\in\mathcal{X}^2}[D(\hat{\boldsymbol{x}}' \oplus H'))^2]\end{aligned}$$

$$\mathcal{L}^G_{\text{gan}}(G) =\mathbb{E}_{(\boldsymbol{x},\boldsymbol{x}')\in\mathcal{X}^2}[(D(\hat{\boldsymbol{x}}' \oplus H') - 1)^2] \tag{5}$$

where $\oplus$ denotes the concatenation along the channel axis. Note that in Eq (5), the dependence on the trained parameters of G, M, and Δ appears implicitly via $\hat{\boldsymbol{x}}'$. Note that we provide the keypoint locations H' to the discriminator to help it to focus on moving parts and not on the background. However, when updating the generator, we do not propagate the discriminator loss gradient through H' to avoid that the generator tends to fool the discriminator by generating meaningless keypoints.

The GAN loss is combined with a feature matching loss that encourages the output image $\hat{\boldsymbol{x}}'$ and $\boldsymbol{x}'$ to have similar feature representations. The feature representations employed to compute this loss are the intermediate layers of the discriminator D. The feature matching loss is given by:

$$\mathcal{L}_{\text{rec}} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{x}')}\left[\|D_i(\hat{\boldsymbol{x}}' \oplus H') - D_i(\boldsymbol{x}' \oplus H'))\|_1\right] \tag{6}$$

where D_i denotes the i^{th}-layer feature extractor of the discriminator D. D_0 denotes the discriminator input. The main advantage of the feature matching loss is that, differently from other perceptual losses, [31, 21], it does not require the use of an external pre-trained network. Finally the overall loss is obtained by combining Eqs. (6) and (5), $\mathcal{L}_{\text{tot}} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \mathcal{L}^G_{\text{gan}}$. In all our experiments, we chose $\lambda_{rec} = 10$ following [44]. Additional details of our implementation are shown in the Supplementary Material A.

3.6. Generation Procedure

At test time, our network receives a driving video and a source image. In order to generate the t^{th} frame, Δ estimates the keypoint locations h^s_k in the source image. Similarly, we estimate the keypoint locations h^1_k and h^t_k from first and the t^{th} frames of the driving video. Rather than generating a video from the absolute positions of the keypoints, the source image keypoints are transferred according to the relative difference between keypoints in the video. The keypoints in the generated frame are given by:

$$h^{s\prime}_k = h^s_k + (h^t_k - h^1_k) \tag{7}$$

The keypoints $h^{s\prime}_k$ and h^s_k are then encoded as heatmaps using the covariance matrices estimated from the driving video, as described in Sec. 3.2. Finally, the heatmaps are given to the dense motion and the generator networks together with the source image (see Secs. 3.3 and 3.4). Importantly, one limitation of transferring relative motion is that it cannot be applied to arbitrary source images. Indeed, if the driving video object is not roughly aligned with the source image object, Eq. (7) may lead to absolute keypoint positions that are physically impossible for the considered object as illustrated in Supplementary Material C.1.

	Tai-Chi			Nemo			Bair
	$\mathcal{L}_1$	(AKD, MKR)	AED	$\mathcal{L}_1$	AKD	AED	$\mathcal{L}_1$
X2Face	0.068	(4.50, 35.7%)	0.27	0.022	0.47	0.140	0.069
Ours	**0.050**	**(2.53, 17.4%)**	**0.21**	**0.017**	**0.37**	**0.072**	**0.025**

Table 1: Video reconstruction comparisons

4. Experiments

In this section, we present a in-depth evaluation on three problems, tested on three very different datasets and employing a large variety of metrics.

Datasets. The UvA-*Nemo* dataset [10] is a facial dynamics analysis dataset composed of 1240 videos We follow the same pre-processing as in [45]. Specifically, faces are aligned using the OpenFace library [1] before re-sizing each frame to 64×64 pixels. Each video starts from a neutral expression and lasts 32 frames. As in [45], we use 1110 videos for training and 124 for evaluation.

The *Tai-Chi* dataset [39] is composed of 4500 tai-chi video clips downloaded from YouTube. We use the data as pre-processed in [39]. In particular, the frames are resized to 64×64 pixels. The videos are split into 3288 and 822 videos for training and testing respectively. The video length varies from 32 to 100 frames.

The *BAIR* robot pushing dataset [11] contains videos collected by a Sawyer robotic arm pushing a variety of objects over a table. It contains 40960 training and 256 test videos. Each video is 64×64 pixels and has 30 frames.

Evaluation Protocol. Evaluating the results of image animation methods is a difficult task, since ground truth animations are not available. In addition, to the best of our knowledge, X2Face [46] is the only previous approach for data-driven model-free image animation. For these two reasons, we evaluate our method also on two closely related tasks. As proposed in [46], we first evaluate Monkey-Net on the task of video reconstruction. This consists in reconstructing the input video from a representation in which motion and content are decoupled. This task is a "proxy" task to image animation and it is only introduced for the purpose of quantitative comparison. In our case, we combine the extracted keypoints $\Delta(x)$ of each frame and the first frame of the video to re-generate the input video. Second, we evaluate our approach on the problem of Image-to-Video translation. Introduced in [42], this problem consists of generating a video from its first frame. Since our model is not directly designed for this task, we train a small recurrent neural network that predicts, from the keypoint coordinates in the first frame, the sequence of keypoint coordinates for the other 32 frames. Additional details can be found in the Supplementary Material A. Finally, we evaluate our model on image animation. In all experiments we use K=10.

Metrics. In our experiments, we adopt several metrics in order to provide an in-depth comparison with other methods. We employ the following metrics.

- $\mathcal{L}_1$. In the case of the video reconstruction task where the ground truth video is available, we compare the average $\mathcal{L}_1$ distance between pixel values of the ground truth and the generated video frames.
- AKD. For the *Tai-Chi* and *Nemo* datasets, we employ external keypoint detectors in order to evaluate whether the motion of the generated video matches the ground truth video motion. For the *Tai-Chi* dataset, we employ the human-pose estimator in [8]. For the *Nemo* dataset we use the facial landmark detector of [6]. We compute these keypoints for each frame of the ground truth and the generated videos. From these externally computed keypoints, we deduce the *Average Keypoint Distance* (AKD), *i.e.* the average distance between the detected keypoints of the ground truth and the generated video.
- MKR. In the case of the *Tai-Chi* dataset, the human-pose estimator returns also a binary label for each keypoint indicating whether the keypoints were successfully detected. Therefore, we also report the *Missing Keypoint Rate* (MKR) that is the percentage of keypoints that are detected in the ground truth frame but not in the generated one. This metric evaluates the appearance quality of each video frame.
- AED. We compute the feature-based metric employed in [12] that consists in computing the *Average Euclidean Distance (AED)* between a feature representation of the ground truth and the generated video frames. The feature embedding is chosen such that the metric evaluates how well the identity is preserved. More precisely, we use a network trained for facial identification [1] for *Nemo* and a network trained for person re-id [17] for *Tai-Chi*.
- FID. When dealing with Image-to-video translation, we complete our evaluation with the *Frechet Inception Distance* [18] (FID) in order to evaluate the quality of individual frames.

Furthermore, we conduct a user study for both the Image-to-Video translation and the image animation tasks (see Sec. 4.3).

4.1. Ablation Study

In this section, we present an ablation study to empirically measure the impact of each part of our proposal on the performance. First, we describe the methods obtained by "amputating" key parts of the model described in Sec. 3.1: (i) *No* $\mathcal{F}$ - the dense optical flow network M is not used; (ii) *No* $\mathcal{F}_{\text{coarse}}$ - in the optical flow network M, we do not use the part based-approach; (iii) *No* $\mathcal{F}_{residual}$ - in the Optical Flow network M, we do not use $\mathcal{F}_{\text{residual}}$; (iv) *No* Σ_k - we do not estimate the covariance matrices Σ_K in the keypoint detector Δ and the variance is set to $\Sigma_k = 0.01$ as in [20]; (v) the source image is not given to the motion network M,

	Tai-Chi		
	$\mathcal{L}_1$	(AKD, MKR)	AED
No $\mathcal{F}$	0.057	(3.11, 23.8%)	0.24
No $\mathcal{F}_{\text{residual}}$	0.051	(2.81, 18.0%)	0.22
No $\mathcal{F}_{coarse}$	0.052	(2.75, 19.7%)	0.22
No Σ_k	0.054	(2.86, 20.6%)	0.23
No $\boldsymbol{x}$	0.051	(2.71, 19.3%)	**0.21**
Full	**0.050**	(**2.53, 17.4%**)	**0.21**

Table 2: Video reconstruction ablation study *TaiChi*.

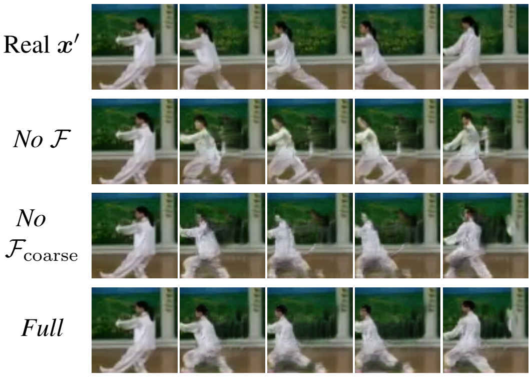


Figure 4: Qualitative ablation evaluation of video reconstruction on *Tai-Chi*.

M estimates the dense optical flow only from the keypoint location differences; (vi) *Full* denotes the full model as described in Sec. 3.

In Tab. 2, we report the quantitative evaluation. We first observe that our full model outperforms the baseline method without deformation. This trend is observed according to all the metrics. This illustrates the benefit of deforming the features maps according to the estimated motion. Moreover, we note that *No* $\mathcal{F}_{coarse}$ and *No* $\mathcal{F}_{\text{residual}}$ both perform worse than when using the full optical flow network. This illustrates that $\mathcal{F}_{coarse}$ and $\mathcal{F}_{\text{residual}}$ alone are not able to estimate dense motion accurately. A possible explanation is that $\mathcal{F}_{coarse}$ cannot estimate non rigid motions and that $\mathcal{F}_{\text{residual}}$, on the other hand, fails in predicting the optical flow in the presence of large motion. The qualitative results shown in Fig. 4 confirm this analysis. Furthermore, we observe a drop in performance when covariance matrices are replaced with static diagonal matrices. This shows the benefit of encoding more information when dealing with videos with complex and large motion, as in the case of the *TaiChi* dataset. Finally, we observe that if the appearance is not provided to the deformation network M, the video reconstruction performance is slightly lower.

4.2. Comparison with Previous Works

Video Reconstruction. First, we compare our results with the X2Face model [46] that is closely related to our proposal. Note that this comparison can be done since we employ image and motion representation of similar dimension. In our case, each video frame is reconstructed from the source image and 10 landmarks, each one represented by 5 numbers (two for the location and three for the symmetric covariance matrix), leading to a motion representation of dimension 50. For X2face, motion is encoded into a driving vector of dimension 128. The quantitative comparison is reported in Tab. 1. Our approach outperforms X2face, according the all the metrics and on all the evaluated datasets. This confirms that encoding motion via motion-specific keypoints leads to a compact but rich representation.

	Tai-Chi		
	FID	AED	MKR
MoCoGAN [39]	54.83	0.27	46.2%
Ours	**19.75**	**0.17**	**30.3%**

	Nemo	
	FID	AED
MoCoGAN [39]	51.50	0.33
CMM-Net [45]	27.27	0.13
Ours	**11.97**	**0.12**

	Bair
	FID
MoCoGAN [39]	244.00
SV2P [2]	57.90
Ours	**23.20**

Table 3: Image-to-video translation comparisons.

Image-to-Video Translation: In Tab. 3 we compare with the state of the art Image-to-Video translation methods: two unsupervised methods MoCoGAN [39] and SV2P [2], and CMM-Net which is based on keypoints [45]. CMM-Net is evaluated only on *Nemo* since it requires facial landmarks. We report results SV2P on the *Bair* dataset as in [2]. We can observe that our method clearly outperforms the three methods for all the metrics. This quantitative evaluation is confirmed by the qualitative evaluation presented in the Supplementary material C.3. In the case of MoCoGAN, we observe that the AED score is much higher than the two other methods. Since AED measures how well the identity is preserved, these results confirm that, despite the realism of the video generated by MoCoGAN, the identity and the person-specific details are not well preserved. A possible explanation is that MoCoGAN is based on a feature embedding in a vector, which does not capture spatial information as well as the keypoints. The method in [45] initially produces a realistic video and preserves the identity, but the lower performance can be explained by the apparition of visual artifacts in the presence of large motion (see the Supplementary material C.3 for visual examples). Conversely, our method both preserves the person identity and performs well even under large spatial deformations.

Image Animation. In Fig. 5, we compare our method with X2Face [46] on the *Nemo* dataset. We note that our method

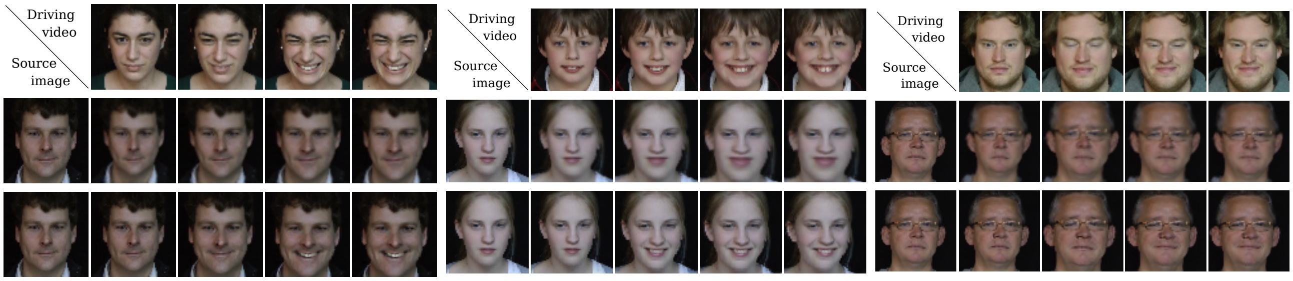


Figure 5: Qualitative results for image animation on the Nemo dataset: X2face (2-nd row) against our method (3-rd row).

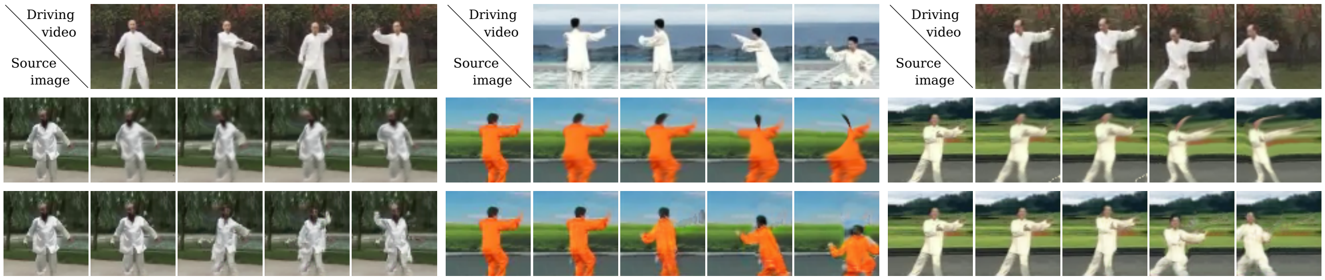


Figure 6: Qualitative results for image animation on the *Tai-Chi* dataset: X2face (2-nd row) against our method (3-rd row)

Tai-Chi	*Nemo*	*Bair*
85.0%	79.2%	90.8%

Table 4: User study results on image animation. Proportion of times our approach is preferred over X2face [46].

generates more realistic smiles on the three randomly selected samples despite the fact that the XFace model is specifically designed for faces. Moreover, the benefit of transferring the relative motion over absolute locations can be clearly observed in Fig. 5 (column 2). When absolute locations are transferred, the source image inherits the face proportion from the driving video, resulting in a face with larger cheeks. In Fig. 6, we compare our method with X2Face on the *Tai-Chi* dataset. X2Face [46] fails to consider each body-part independently and, consequently, warps the body in such a way that its center of mass matches the center of mass in the driving video. Conversely, our method successfully generates plausible motion sequences that match the driving videos. Concerning the *Bair* dataset, exemplar videos are shown in the Supplementary material C.3. The results are well in line with those obtained on the two other datasets.

4.3. User Evaluation

In order to further consolidate the quantitative and qualitative evaluations, we performed user studies for both the Image-to-Video translation (see the Supplementary Material C.3) and the image animation problems using Amazon Mechanical Turk.

For the image animation problem, our model is again compared with X2face [46] according to the following protocol: we randomly select 50 pairs of videos where objects in the first frame have a similar pose. Three videos are shown to the user: one is the driving video (reference) and 2 videos from our method and X2Face. The users are given the following instructions: *Select the video that better corresponds to the animation in the reference video*. We collected annotations for each video from 10 different users The results are presented in Tab. 4. Our generated videos are preferred over X2Face videos in almost more than 80% of the times for all the datasets. Again, we observe that the preference toward our approach is higher on the two datasets which correspond to large motion patterns.

5. Conclusion

We introduced a novel deep learning approach for image animation. Via the use of motion-specific keypoints, previously learned following a self-supervised approach, our model can animate images of arbitrary objects according to the motion given by a driving video. Our experiments, considering both automatically computed metrics and human judgments, demonstrate that the proposed method outperforms previous work on unsupervised image animation. Moreover, we show that with little adaptation our method can perform Image-to-Video translation. In future work, we plan to extend our framework to handle multiple objects and investigate other strategies for motion embedding.

Acknowledgments

This work was carried out under the "Vision and Learning joint Laboratory" between FBK and UNITN.

References

[1] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. Openface: A general-purpose face recognition. 2016.

[2] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. In *ICLR*, 2017.

[3] Guha Balakrishnan, Amy Zhao, Adrian V Dalca, Fredo Durand, and John Guttag. Synthesizing images of humans in unseen poses. In *CVPR*, 2018.

[4] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. In *ECCV*, 2018.

[5] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH*, 1999.

[6] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In *ICCV*, 2017.

[7] Chen Cao, Qiming Hou, and Kun Zhou. Displaced dynamic expression regression for real-time facial tracking and animation. *TOG*, 2014.

[8] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[9] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *ECCV*, 2018.

[10] Hamdi Dibeklioğlu, Albert Ali Salah, and Theo Gevers. Are you really smiling at me? spontaneous versus posed enjoyment smiles. In *ECCV*, 2012.

[11] Frederik Ebert, Chelsea Finn, Alex X Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. In *CoRL*, 2017.

[12] Patrick Esser, Ekaterina Sutter, and Björn Ommer. A variational u-net for conditional appearance and shape generation. In *CVPR*, 2018.

[13] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *NIPS*, 2016.

[14] Yaroslav Ganin, Daniil Kononenko, Diana Sungatullina, and Victor Lempitsky. Deepwarp: Photorealistic image resynthesis for gaze manipulation. In *ECCV*, 2016.

[15] Zhenglin Geng, Chen Cao, and Sergey Tulyakov. 3d guided fine-grained face manipulation. In *CVPR*, 2019.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[17] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv:1703.07737*, 2017.

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*.

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

[20] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In *NIPS*, 2018.

[21] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[22] Donggyu Joo, Doyeon Kim, and Junmo Kim. Generating a fusion image: One's identity and another's shape. In *CVPR*, 2018.

[23] Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *ICML*, 2016.

[24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

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

[26] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *NIPS*, 2015.

[27] Joseph P Robinson, Yuncheng Li, Ning Zhang, Yun Fu, et al. Laplace landmark localization. *arXiv:1903.11633*, 2019.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*. Springer, 2015.

[29] Subhankar Roy, Enver Sangineto, Nicu Sebe, and Begüm Demir. Semantic-fusion gans for semi-supervised satellite image classification. In *ICIP*, 2018.

[30] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping. In *ICCV*, 2017.

[31] Aliaksandr Siarohin, Enver Sangineto, Stéphane Lathuilière, and Nicu Sebe. Deformable gans for pose-based human image generation. In *CVPR*, 2018.

[32] Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening and coloring transform for GANs. In *ICLR*, 2019.

[33] Aliaksandr Siarohin, Gloria Zen, Nicu Sebe, and Elisa Ricci. Enhancing perceptual attributes with bayesian style generation. In *ACCV*, 2018.

[34] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015.

[35] Hao Tang, Wei Wang, Dan Xu, Yan Yan, and Nicu Sebe. Gesturegan for hand gesture-to-gesture translation in the wild. In *ACM MM*, 2018.

[36] Hao Tang, Dan Xu, Nicu Sebe, Yanzhi Wang, Jason J. Corso, and Yan Yan. Multi-channel attention selection gan with cascaded semantic guidance for cross-view image translation. In *CVPR*, 2019.

[37] Hao Tang, Dan Xu, Wei Wang, Yan Yan, and Nicu Sebe. Dual generator generative adversarial networks for multi-domain image-to-image translation. In *ACCV*, 2019.

[38] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *CVPR*, 2016.

[39] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *CVPR*, 2018.

[40] Joost Van Amersfoort, Anitha Kannan, Marc'Aurelio Ranzato, Arthur Szlam, Du Tran, and Soumith Chintala. Transformation-based models of video sequences. *arXiv preprint arXiv:1701.08435*, 2017.

[41] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. In *ICML*, 2017.

[42] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *NIPS*, 2016.

[43] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *NIPS*, 2018.

[44] 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*, 2017.

[45] Wei Wang, Xavier Alameda-Pineda, Dan Xu, Pascal Fua, Elisa Ricci, and Nicu Sebe. Every smile is unique: Landmark-guided diverse smile generation. In *CVPR*, 2018.

[46] Olivia Wiles, A Sophia Koepke, and Andrew Zisserman. X2face: A network for controlling face generation using images, audio, and pose codes. In *ECCV*, 2018.

[47] Yuting Zhang, Yijie Guo, Yixin Jin, Yijun Luo, Zhiyuan He, and Honglak Lee. Unsupervised discovery of object landmarks as structural representations. In *CVPR*, 2018.

[48] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris Metaxas. Learning to forecast and refine residual motion for image-to-video generation. In *ECCV*.

[49] Michael Zollhöfer, Justus Thies, Pablo Garrido, Derek Bradley, Thabo Beeler, Patrick Pérez, Marc Stamminger, Matthias Nießner, and Christian Theobalt. State of the art on monocular 3d face reconstruction, tracking, and applications. In *Computer Graphics Forum*.