Thin-Plate Spline Motion Model for Image Animation

Jian Zhao Hui Zhang
School of Software, BNRist, Tsinghua University, Beijing, China
zhaojian20@mails.tsinghua.edu.cn huizhang@tsinghua.edu.cn

Abstract

Image animation brings life to the static object in the source image according to the driving video. Recent works attempt to perform motion transfer on arbitrary objects through unsupervised methods without using a priori knowledge. However, it remains a significant challenge for current unsupervised methods when there is a large pose gap between the objects in the source and driving images. In this paper, a new end-to-end unsupervised motion transfer framework is proposed to overcome such issues. Firstly, we propose thin-plate spline motion estimation to produce a more flexible optical flow, which warps the feature maps of the source image to the feature domain of the driving image. Secondly, in order to restore the missing regions more realistically, we leverage multi-resolution occlusion masks to achieve more effective feature fusion. Finally, additional auxiliary loss functions are designed to ensure that there is a clear division of labor in the network modules, encouraging the network to generate high-quality images. Our method[1] can animate a variety of objects, including talking faces, human bodies, and pixel animations. Experiments demonstrate that our method performs better on most benchmarks than the state of the art with visible improvements in motion-related metrics.

1. Introduction

Image animation (Fig. 1) transfers the motion of the object in the driving video to the static object in the source image, which is widely used for video conferencing [31], movie effects [21] and entertainment videos. It can stimulate people's creativity to create more interesting works.

Researches have been done on motion transfer by using a priori knowledge of objects, such as 3D models, landmarks, domain labels [6, 9, 11, 18, 23, 27, 30, 35, 37]. However, these approaches, which rely on labeled data, only work for specific objects, such as faces [9, 11, 30, 35] and human bodies [6, 18, 23, 27, 37]. It is costly to obtain such labeled data or pre-trained keypoint extractors. Therefore, these approaches cannot be applied to objects without labeled data.

Figure 1. Example animations generated by our method trained on different datasets.

Recently, some unsupervised motion transfer methods have been proposed that do not require a priori knowledge of objects [25, 26, 28, 32]. These methods use two frames sampled from a video for training, where one frame is used as the source image to reconstruct the other frame as the driving image. And the methods are optimized using reconstruction losses to learn the motion representations. Some unsupervised methods [25, 26, 28] divide motion transfer into two steps. First, an optical flow is estimated using the motion representation that warps the feature maps of the source image to the feature domain of the driving image. Second, an occlusion mask is predicted to indicate the missing regions of the warped feature maps, which are then inpainted in the network. Experiments have shown that unsupervised methods can perform motion transfer on various objects [25, 26, 28].

However, there are still some challenges with the unsupervised methods. First, the motion representation is not flexible enough, making it difficult for the network to learn the large pose gap between the objects in the source and driving images during training. This deficiency results in large discrepancies between the warped feature maps and the feature domain of the driving image. Moreover, the area of the occlusion mask will increase, making motion transfer

[1]Our source code is publicly available: https://github.com/yoyo-nb/Thin-Plate-Spline-Motion-Model.

too dependent on the inpainting capability of the network, which leads to the second problem: inadequate inpainting capability of the network. Previous works [25, 26, 28] did not take full advantage of features at different scales to inpaint the missing regions, so it is difficult to generate more realistic images.

Some unsupervised methods [26, 28] improve the quality of the animation by combining local affine transformations to estimate the motion. However, the affine transformation is linear, which makes it difficult to represent complex motions. In fact, the motions of objects are often not linear locally (for example, when people open their mouths, their lips are curved). To overcome this, we introduce a more flexible nonlinear transformation, thin-plate spline (TPS) transformation, to approximate the motion and propose a new end-to-end unsupervised motion transfer framework. First, we predict several sets of keypoints to generate TPS transformations and combine them with the affine background transformation [28] to estimate the optical flow. Furthermore, we perform dropout for multiple TPS transformations during the early stage of training so that each TPS transformation contributes to the estimated optical flow. TPS motion estimation makes the estimated optical flow more flexible, stable and robust than previously estimated [26, 28]. Second, we predict occlusion masks for each layer of warped feature maps, making the feature maps have a different focus for more efficient feature fusion. Finally, we design the auxiliary loss functions to make each module have a clearer division of labor, encouraging the network to generate high-quality images. The proposed framework approximates the motion more accurately and has a stronger inpainting capability. To summarize, the main contributions are as follows:

- We present TPS motion estimation to approximate the motion from the source image to the driving image. In addition, we perform dropout on multiple TPS transformations before combining them during the early stage of training.
- We propose a new end-to-end unsupervised motion transfer framework. It warps the feature maps of the source image using the estimated optical flow and then leverages multi-resolution occlusion masks to indicate the missing regions for inpainting.
- Experiments demonstrate that our method outperforms previous unsupervised motion transfer methods on various datasets, including talking faces, taichi videos, TED-talk videos and pixel animations. In particular, there is a visible improvement in motion-related metrics.

2. Related Work

Motion transfer. There are many supervised motion transfer methods that require a priori knowledge of moving objects, such as landmarks [6, 11, 23, 27, 35, 37], 3D models [9, 18, 30] or domain labels [7]. Specially, GANimation [22] uses the Facial Action Coding System (FACS) [10] to describe facial expressions. However, these methods cannot be applied to new objects without labeled data, such as pixel animations.

As a comparison, unsupervised methods do not need to introduce a priori knowledge of the animated object during training [25, 26, 28, 32]. X2Face [32] learns the identity representation of the source image by the embedding network, and then generates an optical flow to warp the embedded image. Some unsupervised methods attempt to model the motion representation and disentangle identity and pose from the image. Monkey-Net [25] estimates optical flow for animating by predicting several pairs of unsupervised keypoints. Based on this, first order motion model (FOMM) [26] performs first-order Taylor expansions near each keypoint and approximates the motion in the neighborhood of each keypoint using local affine transformations, which significantly improves the quality of motion transfer. Siarohin *et al*. proposed motion representations for articulated animation (MRAA) [28], which improves the shortcomings of FOMM [26] and achieves state-of-the-art performance of unsupervised methods. MRAA [28] uses PCA-based motion estimation, which has better quality in representing articulated motions (e.g., human body). In addition, it adds background motion estimation to eliminate the negative effects of camera motion. These unsupervised methods can perform motion transfer on arbitrary objects. Comparing these methods, our approach uses TPS motion estimation, which estimates optical flow more flexibly and works better for large-scale motions.

Multi-scale feature fusion. Multi-scale feature fusion has proven to be effective in several tasks in computer vision, including keypoint prediction [4, 29, 33], segmentation [19, 24, 36] and image generation [8, 15, 16]. Extensive experiments have demonstrated that different layers of the network focus on different levels of features [8, 16, 17]. Lower scale feature maps focus on the overall patterns of the image, while larger scale feature maps emphasize detailed textures. The unsupervised motion transfer methods [26, 28] estimate an occlusion mask for the missing regions of the warped feature maps and inpaint them through feature fusion ways. FOMM [26] uses an hourglass network to upsample the warped feature maps gradually to reconstruct the driving image. While Monkey-Net [25] and MRAA [28] warp multi-scale feature maps and add them to the decoder part of the hourglass network via skip connections [27]. However, they use a single occlusion mask for feature maps at different scales, which is not conducive to the network that learns features with different focuses at multiple scales. Our approach also uses a skip-connected hourglass network for inpainting. The difference is that we

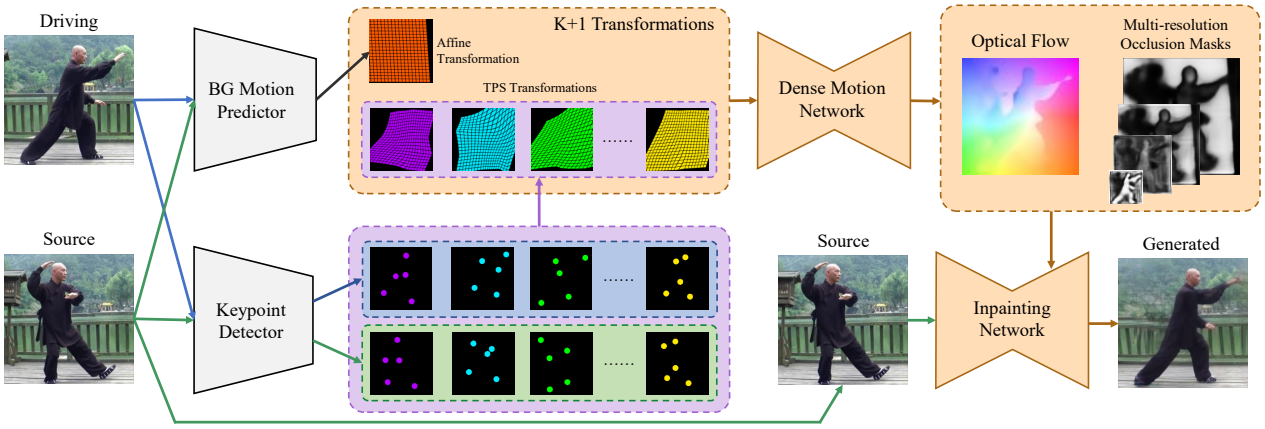

Figure 2. Overview of our model. The BG Motion Predictor predicts the affine transformation, representing the background motion from the source image to the driving image. At the same time, we estimate K sets of keypoints using the Keypoint Detector, each of which generates a TPS transformation. The Dense Motion Network will then combine the $K+1$ transformations (K TPS transformations and one affine transformation) for estimating optical flow and multi-resolution occlusion masks. Finally, we feed the source image into the Inpainting Network, warp the feature maps extracted by the encoder using optical flow, and mask them with the corresponding resolution occlusion masks. The generated image will be output at the last layer of the Inpainting Network.

estimate multi-resolution occlusion masks for feature maps at different scales, allowing features to be more fully fused for more realistic inpainting.

3. Method

Fig. 2 shows the overview of our proposed model. It generates the reconstructed driving image $\hat{\mathbf{D}}$ given a source image $\mathbf{S}$ and a driving image $\mathbf{D}$. The model consists of following modules:

- **Keypoint Detector**. The Keypoint Detector E_{kp} receives $\mathbf{S}$ and $\mathbf{D}$ to predict $K \times N$ pairs of keypoints to generate K TPS transformations.
- **BG Motion Predictor**. The concatenation of $\mathbf{S}$ and $\mathbf{D}$ is fed into the BG Motion Predictor E_{bg} to estimate the parameters of the affine background transformation.
- **Dense Motion Network**. The module is an hourglass network. It receives K TPS transformations from E_{kp} and one affine transformation from E_{bg}. The optical flow will be estimated by combining these $K+1$ transformations. At the same time, the multi-resolution occlusion masks are predicted by different layers of the decoder part to indicate missing regions of the warped feature maps.
- **Inpainting Network**. It is also an hourglass network. It warps the feature maps of the source image using the estimated optical flow and inpaints the missing regions of the feature maps for each scale. The generated image is output at the last layer.

3.1. TPS Motion Estimation

Motion estimation aims to approximate the mapping $\mathcal{T}$ such that $\mathcal{T}(\mathbf{S}) = \mathbf{D}$. In contrast to the combination of local affine transformations in FOMM [26] and MRAA [28], we propose TPS motion estimation to approximate $\mathcal{T}$.

TPS transformation [2] is a flexible, nonlinear transformation that allows representing more complex motions. Given corresponding keypoints in two images, we can warp one to the other with minimum distortion by using TPS transformation $\mathcal{T}_{tps}$:

$$\begin{aligned} \min \iint_{R^2} \left(\left(\frac{\partial^2 \mathcal{T}_{tps}}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 \mathcal{T}_{tps}}{\partial x \partial y} \right)^2 \right. \\ \left. + \left(\frac{\partial^2 \mathcal{T}_{tps}}{\partial y^2} \right)^2 \right) dx dy, \\ \text{s.t.} \quad \mathcal{T}_{tps}(P_i^{\mathbf{S}}) = P_i^{\mathbf{D}}, \quad i = 1, 2, \ldots, N, \end{aligned} \tag{1}$$

where $P_i^{\mathbf{X}}$ is the keypoints of image $\mathbf{X}$. We use the Keypoint Detector to predict $K \times N$ keypoints for $\mathbf{S}$ and $\mathbf{D}$, where K is the number of TPS transformations. Every N pairs ($N = 5$ for this paper) of keypoints generate one TPS transformation from $\mathbf{S}$ to $\mathbf{D}$. According to the derivation in [2] , the k^{th} TPS transformation is obtained as follows:

$$\mathcal{T}_k(p) = A_k \begin{bmatrix} p \\ 1 \end{bmatrix} + \sum_{i=1}^{N} w_{ki} U \left(\left\| P_{ki}^{\mathbf{D}} - p \right\|_2 \right), \tag{2}$$

where $p = (x, y)^\top$ is pixel coordinates, $A_k \in \mathcal{R}^{2\times 3}$ and $w_{ki} \in \mathcal{R}^{2\times 1}$ are the TPS coefficients obtained by solving

Eq. (1), $U(r)$ is a radial basis function, which represents the influence of each keypoint on the pixel at p:

$$U(r) = r^2 \log r^2. \tag{3}$$

Besides, the camera motion in videos will cause the predicted keypoints to appear in the background area, leading to deviations in the motion estimation. To address this problem, we additionally predict an affine background transformation like MRAA [28] to model the background motion:

$$\mathcal{T}_{bg}(p) = A_{bg} \begin{bmatrix} p \\ 1 \end{bmatrix}, \tag{4}$$

where $A_{bg} \in \mathcal{R}^{2\times 3}$ is an affine transformation matrix predicted by the BG Motion Predictor.

Now, we will combine the $K+1$ transformations (K TPS transformations and one affine transformation) to approximate the mapping $\mathcal{T}$. We use the $K+1$ transformations to warp $\mathbf{S}$, cascade the warped images and feed them into the Dense Motion Network. The module predicts $K+1$ contribution maps $\widetilde{\mathbf{M}}_k \in \mathcal{R}^{H\times W}, k = 0, \ldots, K$, where H and W are the height and width of the image, $\widetilde{\mathbf{M}}_0$ corresponds to $\mathcal{T}_{bg}$. The contribution maps are activated by softmax to make them sum to 1 at any pixel location:

$$\mathbf{M}_k(p) = \frac{\exp(\widetilde{\mathbf{M}}_k(p))}{\sum_{i=0}^{K} \exp(\widetilde{\mathbf{M}}_i(p))}, k = 0, \ldots, K, \tag{5}$$

where $\mathbf{M}_k(p)$ is the value of $\mathbf{M}_k$ at coordinate p. We use $\mathbf{M}_k, k = 0, \ldots, K$ to combine the $K+1$ transformations to compute the optical flow:

$$\widetilde{\mathcal{T}}(p) = \mathbf{M}_0(p)\mathcal{T}_{bg}(p) + \sum_{k=1}^{K} \mathbf{M}_k(p)\mathcal{T}_k(p), \tag{6}$$

which is the result of our approximate mapping $\mathcal{T}$. We use $\widetilde{\mathcal{T}}$ to warp the feature maps of $\mathbf{S}$ extracted by the encoder of the Inpainting Network and reconstruct $\mathbf{D}$ in the decoder.

Dropout for TPS transformations. We use K TPS transformations to approximate the motion, but only a few of them may work for the estimated optical flow at the early stage of training. Their contribution maps have zero values at any pixel location after softmax, and will have no contribution during the entire training stage. Therefore, the network can easily fall into local optimums, resulting in poor quality of the generated images.

We use dropout [13] for TPS transformations to avoid this, which is a technique for regularization. Specifically, $\exp(\widetilde{\mathbf{M}}_k(p)), k = 1, \ldots, K$ are set to zero respectively with probability P in softmax, such that some of K TPS transformations do not work for estimating optical flow in this mini-batch training. And the terms not set to zero are divided by $1-P$ to ensure that the expectation of $\sum_{i=1}^{K} \exp(\widetilde{\mathbf{M}}_i(p))$ remains constant. Let $b_i, i = 1, \ldots, K$ obeys Bernoulli distribution with parameter $1-P$, we change Eq. (5) by:

$$\mathbf{M}_k(p) = \begin{cases} \exp(\widetilde{\mathbf{M}}_0(p))/\mathbf{M}_T(p), & k = 0 \\ b_k \exp(\widetilde{\mathbf{M}}_k(p))/(1-P)\mathbf{M}_T(p), & else \end{cases}, \tag{7}$$

where

$$\mathbf{M}_T(p) = \exp(\widetilde{\mathbf{M}}_0(p)) + \sum_{i=1}^{K} b_i \exp(\widetilde{\mathbf{M}}_i(p))/(1-P). \tag{8}$$

Dropout keeps the network from excessive reliance on a few TPS transformations in the early stage of training, and increases the robustness of the network. We remove the dropout operation when each TPS transformation contributes to the estimated optical flow after training several epochs.

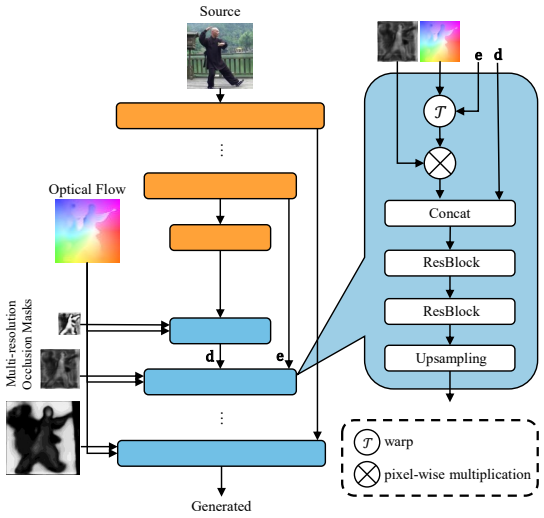

Figure 3. Implementation details of the Inpainting Network.

3.2. Multi-resolution Occlusion Masks

For the Dense Motion Network and the Inpainting Network, we employ the hourglass architecture network to fuse features of different scales, which has been proven effective in various works [4, 17, 24]. In [26, 28], Dense Motion Network estimates a single occlusion mask for the warped feature maps to inpaint the missing regions. However, many experiments have shown that the focus of the feature map changes with its scale [8, 16, 17]. Low-scale feature maps focus on abstract patterns, while high-scale feature maps are more concerned with detailed textures. If a single occlusion mask is used to mask out the feature maps of different scales, what it learns during training will be the trade-off among the focus of different scale feature maps. Hence, we predict occlusion masks of different resolutions for each

layer of the feature map. The Dense Motion Network will not only estimate the optical flow but also predict the multi-resolution occlusion masks by using an additional convolution layer at each layer of the decoder. The estimated optical flow and the multi-resolution occlusion masks are fed together into the Inpainting Network.

In the Inpainting Network, we fuse multi-scale features to generate high-quality images, the details are shown in Fig. 3. We feed $\mathbf{S}$ into the encoder and use optical flow $\widetilde{\mathcal{T}}$ to warp the feature map of each layer. The warped feature map is then masked using the occlusion mask of corresponding resolution and is concatenated to the decoder part via a skip connection. The feature map is then upsampled after passing through two residual blocks. The generated image is output at the final layer.

3.3. Training Losses

Following FOMM [26] and MRAA [28], we use a pre-trained VGG-19 [14] network to calculate the reconstruction loss between $\mathbf{D}$ and the generated image $\hat{\mathbf{D}}$ at multi-resolutions:

$$\mathcal{L}_{rec} = \sum_{j}\sum_{i}\left|V_i(\mathbf{D}_j) - V_i(\hat{\mathbf{D}}_j)\right|, \tag{9}$$

where V_i is the i^{th} layer of the pre-trained VGG-19 network, and j represents that the image is downsampled j times. Equivariance loss is also used to constrain the Keypoint Detector in [26, 28]:

$$\mathcal{L}_{eq} = \left|E_{kp}(\mathcal{T}_{ran}(\mathbf{S})) - \mathcal{T}_{ran}(E_{kp}(\mathbf{S}))\right|, \tag{10}$$

where $\mathcal{T}_{ran}$ is the random nonlinear transformation. We use the random TPS transformation like FOMM [26] and MRAA [28]. In addition, we designed auxiliary loss functions for the modules, namely bg loss and warp loss.

We used an affine transformation to model the background motion, and we made additional constraints on the BG Motion Predictor to make the predicted parameters more accurate and stable. We cascade in the order of $\mathbf{S}$ and $\mathbf{D}$ and feed them into the BG Motion Predictor to obtain the affine transformation matrix A_{bg}, representing the background's motion from $\mathbf{S}$ to $\mathbf{D}$. We then re-cascade them in reverse order to obtain the affine transformation matrix A'_{bg}. We expect the two affine transformation matrices to remain consistent:

$$\begin{bmatrix} A'_{bg} \\ 0\ \ 0\ \ 1 \end{bmatrix} = \begin{bmatrix} A_{bg} \\ 0\ \ 0\ \ 1 \end{bmatrix}^{-1}. \tag{11}$$

However, we cannot use Eq. (11) as a loss function because it is easy to make the BG Motion Predictor output a zero matrix that minimizes the difference between the two sides of the equation. We reformulate Eq. (11) in the following way:

$$\mathcal{L}_{bg} = \left|\begin{bmatrix} A'_{bg} \\ 0\ \ 0\ \ 1 \end{bmatrix}\begin{bmatrix} A_{bg} \\ 0\ \ 0\ \ 1 \end{bmatrix} - I\right|, \tag{12}$$

where I is 3×3 identity matrix.

An additional constraint is also designed for the Inpainting Network. We feed $\mathbf{D}$ into the encoder of the Inpainting Network. The warped encoder feature maps of $\mathbf{S}$ are used to compute the loss with the encoder feature maps of $\mathbf{D}$ at each layer:

$$\mathcal{L}_{warp} = \sum_{i}\left|\widetilde{\mathcal{T}}(E_i(\mathbf{S})) - E_i(\mathbf{D})\right|, \tag{13}$$

where E_i is the i^{th} layer of the Inpainting Network encoder. $\mathcal{L}_{warp}$ can encourage the network to estimate the optical flow more reasonably, making the warped feature maps closer to the feature domain of $\mathbf{D}$.

The final loss is the sum of terms:

$$\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{eq} + \mathcal{L}_{bg} + \mathcal{L}_{warp}. \tag{14}$$

3.4. Testing Stage

At the testing stage, we use a source image $\mathbf{S}$ and a driving video $\{\mathbf{D}_t\}, t = 1, 2, \ldots, T$ for image animation. FOMM [26] has two modes for image animation: *standard* and *relative*. The *standard* mode uses each frame $\mathbf{D}_t$ and $\mathbf{S}$ directly to estimate the motion using Eq. (6), while the *relative* mode estimates the motion between $\mathbf{D}_t$ and the first frame $\mathbf{D}_1$ and then applies it to $\mathbf{S}$. However, the *standard* mode does not perform well when there is a large mismatch between identities (for example, animate a thin person according to the motion of a fat person). The *relative* mode requires that the pose of $\mathbf{D}_1$ be similar to the pose of $\mathbf{S}$. MRAA [28] proposes a new mode, animation via disentanglement (*avd*), that uses an additional trained network to predict the motion that should be applied to $\mathbf{S}$, and we use the same mode for image animation.

We train a shape and a pose encoder as in MRAA [28]. The shape encoder learns the shape of the keypoints of $\mathbf{S}$, and the pose encoder learns the pose of the keypoints of $\mathbf{D}_t$. Then a decoder reconstructs the keypoints, preserving the shape of $\mathbf{S}$ and the pose of $\mathbf{D}_t$. Both the encoders and the decoder are implemented by fully connected layers. We use keypoints of two frames from a video to train the networks, where the keypoints of one frame are randomly deformed to simulate the pose of a different identity. $\mathcal{L}_1$ loss is used to encourage the networks to reconstruct the keypoints before deformation. For image animation, we feed the keypoints of $\mathbf{S}$ and $\mathbf{D}_t$ into the shape and pose encoders to get the reconstructed keypoints, and then use Eq. (6) to estimate the motion.

4. Experiments

4.1. Benchmarks

We trained on multiple types of datasets, including talking faces, human bodies and pixel animation. The pre-processing and training-test splitting strategies for each dataset are the same as in [28]. The datasets are as follows:

	TaiChiHD			TED-talks			VoxCeleb			MGif
	$\mathcal{L}_1$	(AKD, MKR)	AED	$\mathcal{L}_1$	(AKD, MKR)	AED	$\mathcal{L}_1$	AKD	AED	$\mathcal{L}_1$
X2Face [32]	0.080	(17.65, 0.109)	0.27	-	-	-	0.078	7.69	0.405	-
Monkey-Net [25]	0.077	(10.80, 0.059)	0.288	-	-	-	0.049	1.89	0.199	-
FOMM [26]	0.055	(6.62, 0.031)	0.164	0.033	(7.07, 0.014)	0.163	0.041	1.29	0.135	0.0225
MRAA [28]	0.048	(5.41, 0.025)	**0.149**	**0.026**	(4.01, 0.012)	**0.116**	0.040	1.29	0.136	0.0274
Ours	**0.045**	(**4.57**, **0.018**)	0.151	0.027	(**3.39**, **0.007**)	0.124	**0.039**	**1.22**	**0.125**	**0.0212**

Table 1. Video reconstruction: comparison with the state of the art on four different datasets. $K = 10$ for all methods. (Lower is better, best result in bold)

- *VoxCeleb* [20]: Videos of different celebrities talking downloaded from youtube cropped to 256*256 resolution according to the bounding boxes of the faces. The length of the videos ranges from 64 to 1024 frames.
- *TaiChiHD* [26]: Videos of full bodies TaiChi performance downloaded from youtube cropped to 256*256 resolution according to the bounding boxes of the bodies.
- *TED-talks* [28]: Videos of TED talk downloaded from youtube, which is a new dataset proposed in MRAA. The videos were downscaled to 384*384 resolution according to the upper part of the human bodies. The length of the videos ranges from 64 to 1024 frames.
- *MGif* [25]: A dataset of *.gif* files of pixel animations about animals running, which was collected using google searches.

In previous work, video reconstruction was used to evaluate the quality of motion transfer by taking the first frame $\mathbf{D}_1$ of a video as the source image to reconstruct $\{\mathbf{D}_t\}, t = 1, 2, \ldots, T$. We used the same quantitative metrics:

- $\mathcal{L}_1$: Average $\mathcal{L}_1$ distance between the generated and driving image pixel values.
- *Average keypoint distance* (AKD): AKD evaluates the poses of the generated images. We use the same pre-trained detectors for bodies [5] and faces [4] as MRAA [28] to extract keypoints from the generated and driving images. Then calculate the average distance of the corresponding keypoints.
- *Missing keypoint rate* (MKR): The proportion of keypoints extracted from the pre-trained model [4, 5] that are present in the driving image but missing in the generated image.
- *Average Euclidean distance* (AED): AED evaluates the identity of the generated images. We use the same pre-trained re-identification networks for bodies [12] and faces [1] as MRAA [28] to extract identities from the generated and driving images. Then calculate the average $\mathcal{L}_2$ distance of the extracted identity pairs.

4.2. Comparison

We used two GeForce RTX 3090 GPUs to train 100 epochs on each dataset. VoxCeleb, TaiChiHD, and MGif for three days, while TED-talks was trained for eight days due to higher resolution. We compared our method with the current state-of-the-art unsupervised motion transfer method, MRAA [28], on video reconstruction and image animation tasks. We also compared with other baseline methods FOMM [26], Monkey-Net [25] and X2Face [32] on video reconstruction. FOMM [26], MRAA [28] and our methods are trained for the same number of epochs with K = 10.

Figure 4. Some bad cases of MRAA [28], while our method shows high quality on video reconstruction task.

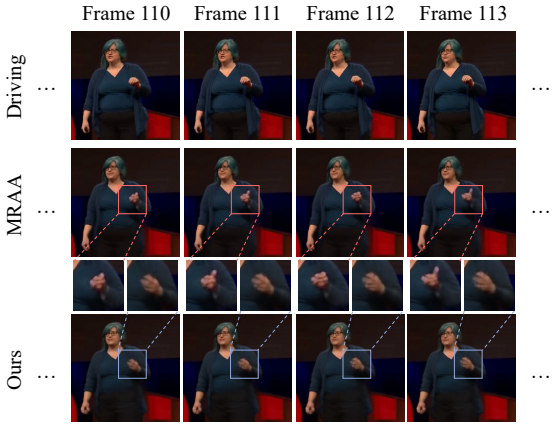

Figure 5. Our method has better temporal continuity than MRAA [28] on the video reconstruction task.

Video reconstruction. Quantitative results of video recon-

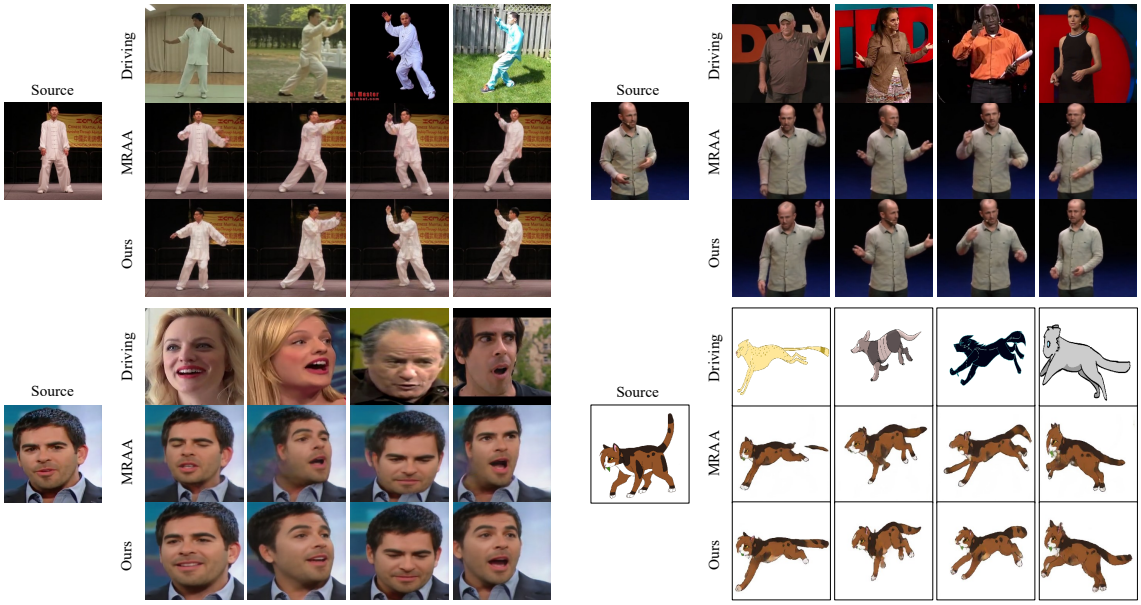

Figure 6. Qualitative comparison with MRAA [28] on image animation task: TaiChiHD (top left), TED-talks (top right), VoxCeleb (bottom left), MGif (bottom right).

struction are shown in Tab. 1. Our method reaches state-of-the-art results on VoxCeleb, TaiChiHD and MGif datasets, with significant improvements in motion-related metrics on TaiChiHD dataset (15.5% for AKD and 28.0% for MKR). This suggests that our method estimates motions more accurately than others. Our method also outperform MRAA [28] on TED-talks dataset with motion-related metrics (AKD), but was slightly worse in other metrics (L1 and AED). The latter metrics are related to identity. Fig. 4 shows several bad cases in MRAA [28] on TED-talks dataset while our method has better reconstruction quality in hand, arm, and head areas.

Another advantage of our method is that the reconstructed video has a better temporal continuity. MRAA [28] uses local affine transformations near the keypoints to estimate the motion. Therefore, the temporal continuity of the reconstructed video depends on the smoothness of the keypoints change. If the location of the keypoints in two adjacent frames change greatly, it will cause pixel jitter (as shown in Fig. 5, the video reconstructed by MRAA [28] has a redundant finger at frames 111 and 113, but not at frames 110 and 112). Instead, we use TPS motion estimation and each transformation is generated by multiple keypoints, which increases the robustness of motion estimation.

Image Animation. Fig. 6 shows selected image animation results of our method compared with MRAA [28] on the four datasets. Both MRAA [28] and our method use the *avd* mode to generate image animation. The results show that our method has better motion transfer performance for human bodies (TaiChiHD and TED-talks), but the ability to maintain image details such as clothes and faces is slightly poor. For human faces (VoxCeleb), ghosts will appear in the images generated by MRAA [28], while our method has better generation quality. Our method also performs better than MRAA [28] on pixel animations (MGif). However, when the size of the identity differs greatly, neither MRAA [28] nor our method works well (for example, animate a puppy according to the motion of a giraffe).

	Continuity	Authenticity
TaiChiHD	71.30%	86.14%
TED-talks	63.93%	58.44%
VoxCeleb	80.95%	61.54%

Table 2. User study on image animation, numbers respect the proportion (%) of users that prefer our method over MRAA [28].

In order to quantitatively evaluate the quality of image animation, we designed a questionnaire containing randomly selected 20 pairs of videos generated by our method and MRAA [28] on TaiChiHD, TED-talks and VoxCeleb for user preference study. People were invited to judge which of the two videos was better for continuity and authenticity. Results are reported in Tab. 2. Our method performs much better on temporal continuity than MRAA [28], and most users prefer the videos generated by our method for authenticity.

4.3. Ablations

We perform ablation experiments on TaiChiHD dataset to demonstrate the improvement from each of our proposed

	K=5, (K=2 for ours)			K=10, (K=4 for ours)			K=20, (K=8 for ours)		
	$\mathcal{L}_1$	(AKD, MKR)	AED	$\mathcal{L}_1$	(AKD, MKR)	AED	$\mathcal{L}_1$	(AKD, MKR)	AED
FOMM [26]	0.062	(7.34, 0.036)	0.181	0.056	(6.53, 0.033)	0.172	0.062	(8.29, 0.049)	0.196
MRAA [28]	0.049	(6.04, 0.029)	**0.162**	0.048	(5.41, 0.025)	**0.149**	0.046	(5.17, 0.026)	**0.141**
Ours	**0.048**	(**5.24**, **0.022**)	0.166	**0.046**	(**4.84**, **0.020**)	0.156	**0.045**	(**4.67**, **0.019**)	0.150

Table 3. Additional experiments on TaiChiHD for different K and similar bottleneck sizes for MRAA and ours. (Best result in bold.)

	$\mathcal{L}_1$	(AKD, MKR)	AED
MRAA [28]	0.048	(5.41, 0.025)	**0.149**
TPS	0.048	(4.96, 0.020)	0.153
+ Dropout	0.048	(4.66, 0.018)	0.156
+ Multi-Masks	0.046	(4.73, 0.018)	0.150
$+\mathcal{L}_{bg}$	0.046	(4.64, 0.020)	0.151
$+\mathcal{L}_{warp}$	**0.045**	(**4.57**, **0.018**)	0.151

Table 4. Ablation study for video reconstruction on TaiChiHD. (Lower is better, best result in bold)

components. We add our components in turn and compare the video reconstruction metrics with MRAA [28]. Firstly, we use TPS transformations to estimate the optical flow instead of local affine transformations in MRAA [28]. Then we added the dropout operation and the multi-resolution occlusion masks. Finally, we add $\mathcal{L}_{bg}$ and $\mathcal{L}_{warp}$ during training. Quantitative results are shown in Tab. 4.

The second row of Tab. 4 demonstrates that TPS motion estimation improves AKD and MKR, resulting in more accurate motion estimation. Comparing the second and third rows of Tab. 4, dropout can bring lower AKD and MKR, which indicates that dropout makes each TPS transformation contributes to the optical flow to distort the object in **S** into a more accurate pose. However, dropout also affects AED because the more complex optical flow means the larger area of missing regions in warped feature maps, resulting in insufficient information for the Inpainting Network to revise the image. The fourth row of Tab. 4 shows that the multi-resolution occlusion masks bring an improvement to $\mathcal{L}_1$ and AED, which can help the Inpainting Network to generate higher quality images. Fig. 7 shows the multi-resolution occlusion masks we learned by our full method, compared with the single occlusion mask learned by the method in the third row of Tab. 4. But at the same time, it brings a higher AKD, which is not what we expected. When we add $\mathcal{L}_{bg}$ and $\mathcal{L}_{warp}$, AKD gradually decreases, the full method can achieve a better balance on the four metrics.

As with MRAA and FOMM, one of the most important hyper-parameters in our model is the number of TPS transformations, K, which corresponds to the dimension of the motion representation. The dimensions of FOMM and MRAA are $K*(2+4)$ and $(K+1)*(2+4)$, while ours is $K*(6+5*2)+6$. When K = 5, 10, and 20, the dimensions of MRAA are 36, 66, and 126, respectively. We set K of our method to be 2, 4, and 8 to compare with them, and the dimensions are 38, 70, and 118, respectively, which is similar to MRAA. Tab. 3 shows the experiment results, which demonstrates that our method can achieve better motion-related metrics than MRAA when using similar motion description dimensions.

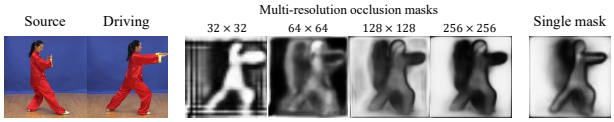

Figure 7. Comparison of learned multi-resolution occlusion mask and single occlusion mask.

5. Discussion and Conclusion

In this paper, we first discuss the drawbacks of using local affine transformations to approximate motions in previous works and propose TPS motion estimation to estimate an optical flow that warps the feature maps of the source image to the feature domain of the driving image. In addition, we use dropout for TPS transformations before combining them in the early stage of training, which keeps the network from excessive reliance on a few TPS transformations and avoids the network falling into local optimums. Secondly, the multi-resolution occlusion masks are used to achieve more effective feature fusion instead of a single occlusion mask. Finally, we design additional auxiliary loss functions and proved experimentally effective.

Our method achieves state-of-the-art performance on most benchmarks with visible improvements in motion-related metrics. However, our approach does not perform well when an extreme identity mismatch occurs. Unsupervised motion transfer remains a worthwhile challenge.

Potential negative societal impact. While the proposed method may be used to make fake videos for spoofing, some detection software will easily determine the authenticity of videos by analyzing the color textures [3] or using the depth information obtained through the proximity sensors, which cannot be simulated by 2D image generating methods. And our approach can create a new benchmark for face anti-spoofing researches [34].

References

[1] Brandon Amos, Bartosz Ludwiczuk, Mahadev Satyanarayanan, et al. Openface: A general-purpose face recognition library with mobile applications. *CMU School of Computer Science*, 2016. 6

[2] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 1989. 3

[3] Zinelabidine Boulkenafet, Jukka Komulainen, and Abdenour Hadid. Face anti-spoofing based on color texture analysis. In *IEEE International Conference on Image Processing*, 2015. 8

[4] 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 *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 2, 4, 6

[5] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017. 6

[6] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 1, 2

[7] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018. 2

[8] Emily L Denton, Soumith Chintala, arthur szlam, and Rob Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NeurIPS*, 2015. 2, 4

[9] Michail Christos Doukas, Stefanos Zafeiriou, and Viktoriia Sharmanska. Headgan: One-shot neural head synthesis and editing. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 1, 2

[10] Paul Ekman and Erika L Rosenberg. *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, USA, 1997. 2

[11] Sungjoo Ha, Martin Kersner, Beomsu Kim, Seokjun Seo, and Dongyoung Kim. Marionette: Few-shot face reenactment preserving identity of unseen targets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 1, 2

[12] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv:1703.07737*, 2017. 6

[13] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012. 4

[14] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European Conference on Computer Vision*, 2016. 5

[15] Animesh Karnewar and Oliver Wang. Msg-gan: Multi-scale gradients for generative adversarial networks. In *CVPR*, 2020. 2

[16] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, 2017. 2, 4

[17] Jie Liang, Hui Zeng, and Lei Zhang. High-resolution photo-realistic image translation in real-time: A laplacian pyramid translation network. In *CVPR*, 2021. 2, 4

[18] Wen Liu, Zhixin Piao, Jie Min, Wenhan Luo, Lin Ma, and Shenghua Gao. Liquid warping gan: A unified framework for human motion imitation, appearance transfer and novel view synthesis. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 1, 2

[19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2

[20] A. Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: a large-scale speaker identification dataset. In *INTERSPEECH*, 2017. 6

[21] Jacek Naruniec, Leonhard Helminger, Christopher Schroers, and Romann M Weber. High-resolution neural face swapping for visual effects. In *Computer Graphics Forum*, 2020. 1

[22] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. Ganimation: Anatomically-aware facial animation from a single image. In *Proceedings of the European Conference on Computer Vision*, 2018. 2

[23] Yurui Ren, Xiaoming Yu, Junming Chen, Thomas H Li, and Ge Li. Deep image spatial transformation for person image generation. In *CVPR*, 2020. 1, 2

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015. 2, 4

[25] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In *CVPR*, 2019. 1, 2, 6

[26] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In *NeurIPS*, 2019. 1, 2, 3, 4, 5, 6, 8

[27] Aliaksandr Siarohin, Enver Sangineto, Stéphane Lathuiliere, and Nicu Sebe. Deformable gans for pose-based human image generation. In *CVPR*, 2018. 1, 2

[28] Aliaksandr Siarohin, Oliver Woodford, Jian Ren, Menglei Chai, and Sergey Tulyakov. Motion representations for articulated animation. In *CVPR*, 2021. 1, 2, 3, 4, 5, 6, 7, 8

[29] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019. 2

[30] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *CVPR*, 2016. 1, 2

[31] Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-shot free-view neural talking-head synthesis for video conferencing. In *CVPR*, 2021. 1

[32] Olivia Wiles, A Koepke, and Andrew Zisserman. X2face: A network for controlling face generation using images, audio,

and pose codes. In *Proceedings of the European Conference on Computer Vision*, 2018. 1, 2, 6

[33] Xi Yin and Xiaoming Liu. Multi-task convolutional neural network for pose-invariant face recognition. *IEEE Transactions on Image Processing*, 2017. 2

[34] Zitong Yu, Yunxiao Qin, Xiaobai Li, Chenxu Zhao, Zhen Lei, and Guoying Zhao. Deep learning for face anti-spoofing: A survey. *arXiv:2106.14948*, 2021. 8

[35] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 1, 2

[36] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 2

[37] Zhen Zhu, Tengteng Huang, Baoguang Shi, Miao Yu, Bofei Wang, and Xiang Bai. Progressive pose attention transfer for person image generation. In *CVPR*, 2019. 1, 2