Make Static Person Walk Again via Separating Pose Action from Shape

Yongwei Nie[a], Meihua Zhao[a], Qing Zhang[b], Ping Li[c], Jian Zhu[d,*] and Hongmin Cai[a,*]

[a] *South China University of Technology, Guangzhou, 510006, Guangdong, China*
[b] *Sun Yat-sen University, Guangzhou, 510006, Guangdong, China*
[c] *The Hong Kong Polytechnic University, Hong Kong, China*
[d] *Guangdong University of Technology, Guangzhou, , Guangdong, China*

ARTICLE INFO

Keywords:
Human motion
Character animation
Action synthesis
Prediction of action and shape

ABSTRACT

This paper addresses the problem of animating a person in a static image. The core task is to infer future poses for the person. Existing approaches predict future poses in 2D space, suffering from entanglements of pose action and pose shape. We propose a method that generates pose actions in 3D space while inheriting pose shape in 2D space. By this separated generation of pose action and shape, our method is free of the above entanglements. Specifically, our method first lifts the 2D pose of person into the 3D space. Then we propose a 3D action synthesis network that generates a sequence of 3D skeletons beginning with the lifted pose. Finally, we transfer the 3D skeletons to a set of 2D poses that inherit actions from the 3D skeletons while holding the same shape as the given 2D pose. Experiments on several datasets validate the effectiveness of our method.

1. Introduction

The Computer Graphics community is the one among others that has devoted lots of effort to animate single images. Representative works include motion textures [19], Cinemagraphs [69], Cliplets[22]. The object been animated varies from passive elements responding to natural forces [11] to a group of animals [56], from human faces [14; 61] to cartoon characters [53], etc. In this paper, we attempt to make a walk person (e.g., a pedestrian) in a static image move again, which is useful in many image/video processing applications [55; 38; 32; 64; 25].

To make a static person walk, the major challenge comes from the extremely large possible future motion and appearance combinations of the pedestrian. Nowadays, with deep neural networks and leveraging large datasets, many works in vision [50; 47; 26; 66; 52] train a model that can directly maps an image to video. In order to reduce complexity, the most successful approaches [50; 47; 66] decompose motion from appearance and predict motion at first, then synthesize future frames guided by the corresponding motion vectors. This process is usually deterministic [47; 66], yet can become stochastic [49; 50; 13] by utilizing generative model such as GAN or VAE. In this paper, we follow the overall pipeline of these approaches, i.e., predicting human motion at first, and then use an existing pose-guided image synthesis approach to generate future frames. However, the existing approaches predict poses in the 2D space, which are subject to the entanglement of pose action and pose shape. By "action", we mean the movement type that a pose take, such as jump, running, etc., while by "shape" we mean the length of arms, the height of pose, etc. Due to the entanglement of action and shape, and also the camera perspectives and occlusions inherently baked in the 2D frames, existing approaches are usually required to be trained on large video datasets. The trained model are also not general enough to handle persons not seen during training.

In this paper, we propose a pose prediction method that separates the prediction of action and shape. Our model is only required to be trained on 3D MoCap datasets, and is general enough to process images from different domains. The core contribution of our method is that we synthesize actions in the 3D space, and transfer the actions to 2D poses while making the shape of the generated 2D poses similar to the shape of the 2D pose of the input person. Our key idea is to utilize the abundant of 3D MoCap data that fully captures the moving characteristics of human, with which we can synthesize high-fidelity human actions. By contrast, learning actions on 2D video data is relatively more challenging and ambiguous.

To fulfill the above objectives, we propose a method that is consisted of three steps. Firstly, we lift the 2D pose of the person to be re-animated to a 3D skeleton. Then in the 3D space, we propose a method to synthesize a sequence of 3D skeletons beginning with the lifted skeleton. Finally, we propose a network to generate the target future 2D poses which inherit the actions of the 3D skeletons and at the same time hold the shape of the pose of the input person. We stress that for the first step we just need to guarantee that the lifted 3D skeleton has similar action as the 2D pose. Therefore, instead of the cumbersome deep learning-based lifting approaches such as [3; 70; 37], we propose a simple yet effective virtual 3D skeleton surrounding projection method together with several pose similarity metrics, by which we directly retrieve the most similar 3D skeleton in the MoCap dataset to the given 2D pose. For the second step, we introduce user intervention into the action synthesis process, allowing the user to control the path of the synthesized actions. As for the third step, we propose a LSTM-based transfer network, trained with action transfer loss and temporal smoothness loss, with no need of ground-truth 2D poses.

*Corresponding author
zj@gdut.edu.cn (J. Zhu); hmcai@scut.edu.cn (H. Cai)
ORCID(s): 0000-0002-8922-3205 (Y. Nie)

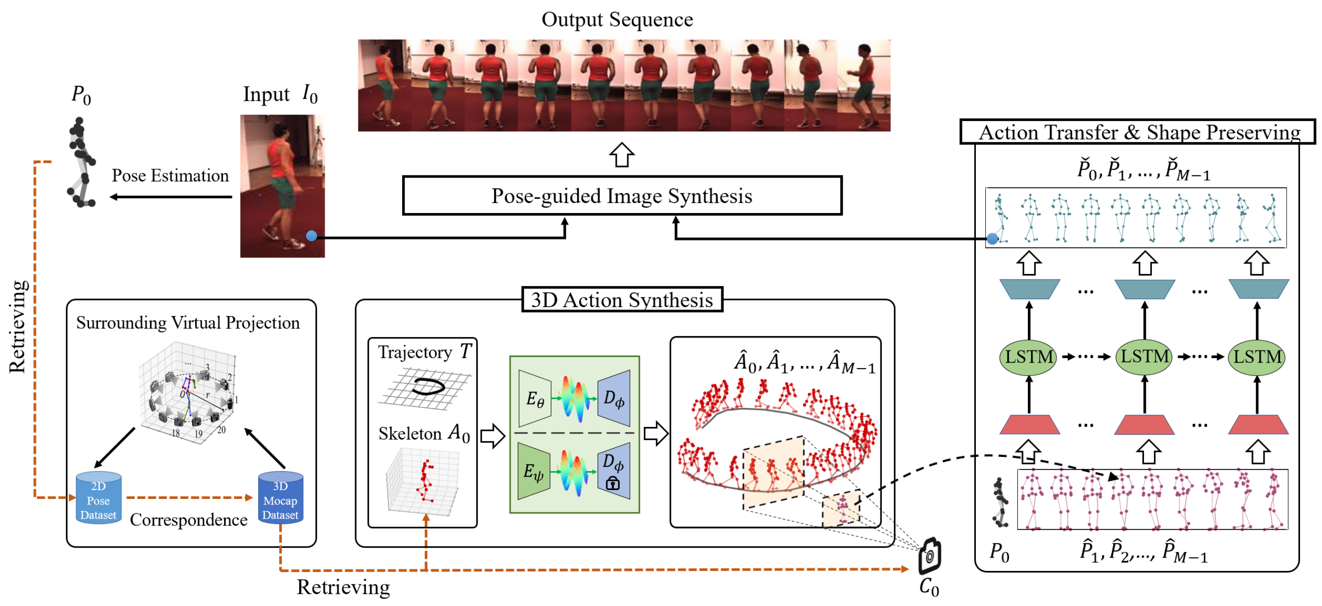

Output Sequence
P_0
Input I_0
Pose Estimation
Pose-guided Image Synthesis
Action Transfer & Shape Preserving
$\check{P}_0, \check{P}_1, \dots, \check{P}_{M-1}$
LSTM
Retrieving
Surrounding Virtual Projection
2D Pose Dataset
3D Mocap Dataset
Correspondence
3D Action Synthesis
Trajectory T
Skeleton A_0
E_θ
D_ϕ
E_ψ
$\hat{A}_0, \hat{A}_1, \dots, \hat{A}_{M-1}$
P_0
$\hat{P}_1, \hat{P}_2, \dots, \hat{P}_{M-1}$
Retrieving
C_0

Figure 1: Overview of the proposed framework. Given an input image I_0, the pose P_0 of it is firstly extracted. We retrieve the most similar pose P_0' of P_0 from the projected 2D pose dataset from 3D MoCap dataset by virtual surrounding projection. P_0' corresponds to skeleton A_0 in the 3D MoCap dataset and the camera projection matrix C_0 from A_0 to P_0'. With A_0 and a user-specified trajectory T, a skeleton sequence $\hat{A}_0, \hat{A}_1, \cdots, \hat{A}_{M-1}$ is synthesized by the proposed 3D action synthesis network. The generated 3D skeletons are then projected to a sequence of 2D poses $\hat{P}_0, \hat{P}_1, \cdots, \hat{P}_{M-1}$ by the camera C_0. Then, the input pose P_0 and poses $\hat{P}_1, \cdots, \hat{P}_{M-1}$ are refined to obtain another sequence of poses $\check{P}_0, \check{P}_1, \cdots, \check{P}_{M-1}$ by the action transfer and shape preserving network. Finally, with $\check{P}_1, \cdots, \check{P}_{M-1}$ and the input image I_0, we synthesize the future frames by pose-guided image synthesis network [71].

In total, our method is simple to implement, easy to train, and effective to predict future poses. The contributions of this paper include:

- We propose a pose prediction approach, which separates the prediction of action from the shape. The prediction of the action is achieved in the 3D space, while the shape is inherited from the given 2D pose.
- We propose a 3D action synthesis network incorporating user intervention and a LSTM-based action transfer network trained on the synthesized 3D actions and the 2D pose of the given person.
- Since only trained on 3D MoCap datasets, our method is general to process different domains of persons, as the actions of the person is generated in the 3D space (thus only requiring 3D MoCap data as training data) and the shape of the person is already given in the input image. We test our method on the datasets of Market-1501 [67], DeepFashion [29], and Human3.6M [21].

2. Related Work

Animating objects in a still image is a long-standing problem in Computer Graphics. Lots of attempts have been made to handle this task. Some works animate fluid elements in an image such as water, cloud, etc. For example, Chuang et al. [7] divide image into layers and synthesize stochastic motion textures to drive layers containing passive elements responding to natural forces to animate. Holynski et al. [19] propose an image-to-image translation network that encodes motion priors of natural scene videos at training stage while at test time generating Eulerian motion fields that animate image recursively. Cinemagraph [2; 44] and cliplets [22] create a still image with some portion of it showing dynamic motion provided by an additional video. Recently, Zhou et al. [69] propose a method that can generate cinemagraph sequences from single still images by utilizing deep reinforcement learning. In [1], a method is proposed to animate a still portrait driven by an extra human face video. Techniques of image registration, warping and hallucination are used to achieve realistic animation, relying on some user interactions. Geng et al. [14] also register and warp the target face image to the face in a video, but use conditional Generative Neural Networks (GNNs) [15] to synthesize appearance details including creases and wrinkles for the warped face. In [61], the object to be warped is the pose of face, while the face that conforms to a new pose is totally synthesized by GNNs. More related to ours are character animation from 2D pictures [20; 53]. Hornung et al. [20] animate 2D characters with the help of 3D MoCap data. They find the most similar 3D skeleton for the 2D character and use the projections of the skeleton sequence into the image place to warp the 2D mesh of the character. Weng et al. [53] construct a SMPL model [30] that tightly fitted with the silhouette of the target

2D character, bringing the character to stand up from image. There are many other single-image animation applications, such as generating time-lapse video [68], animating groups of animals [56] and manga images [4], etc.

Future frame prediction has become possible with the advent of deep neural networks since they can learn complex mapping between input and output space by training on massive datasets. Works of [35; 42; 48] directly predict pixels, learning to extrapolate multiple past frames into the future. While they work well for simple images such as those from MINIST, they tend to produce blurry futures when applied to more complex datasets. Castrejon et al. [6] ascribe this to underfitting and propose more sophisticated hierarchical VRNN models with higher expressiveness capacity. Kwon et al. [23] predict both future and past frames and use retrospective cycle GAN to enforce the consistency between the bi-directional predictions. Apart from these efforts, many other researchers have realized that the blurriness is caused by the great uncertainty about how each pixel will evolve in the future. Most of recent works thus employ high-level structure to guide the prediction process [60; 57; 46; 45; 26; 41; 47; 58; 50; 66]. For example, although Ye et al. [60] still predict pixels, they assume a scene is composed of independent entities and implicitly predict future positions for them. Instead of pixels, works of [46; 57] model the difference between adjacent images. In MoCoGAN [45], motion is explicitly modeled in a separate branch other than appearance as a cue for future frame synthesis. Li et al. [26] formulate multi-frame prediction task as a flow prediction stage followed by flow-to-frame synthesis. Siarohin et al. [41] extract keypoints from images and compute affine transformations between them as guidance to transfer the motion of a driven video to an image. For human video prediction, lots of approaches [47; 58; 50; 66] predict future human poses at first, then synthesize frames according to these poses. We follow this line of research, but unlike they predict future poses by training models on sequential data, we synthesize actions in the 3D space with the help of a global trajectory and then transfer these actions to a human pose.

Human Pose Prediction outputs future poses after observing a sequence of past poses. Existing approaches can be roughly classified into three categories: CNN-based methods, RNN-based methods, and GCN-based methods [59; 9; 31; 43; 51; 10]. CNN-based methods [59; 28] treat input poses data as a two-dimensional matrix, then spatiotemporal convolutional filters can be applied to the pose data like what has done for an image. RNN-based methods [34] have advantages in dealing with time-related tasks, but they usually suffer from discontinuity and error accumulation problems. One of the above mentioned pose-guided video prediction method [58] uses CNN for future pose prediction, while others [50; 66] are based on RNN [12]. Recently, works of [33; 9; 31] demonstrate that Graph Convolution Network (GCN) is very suitable for pose prediction, as it can flexibly learn relations between any pair of human joints. In this paper, we compare our method with the pose prediction method of [31].

3. Methodology

Fig. 1 shows the overview of the proposed method. Given an image containing a walk person, we extract 2D pose P_0 of it by pose estimation approach [5]. Other pose estimation methods [65; 62; 40] can also be used here. Given the 2D pose, we use a pose retrieval scheme (i.e., virtual surrounding projection) to obtain the most similar 3D skeleton in MoCap dataset under the projection of a camera. We then propose a 3D action synthesis network to generate a sequence of 3D skeletons, given the retrieved skeleton and a user-provided trajectory. Next, we propose an action transfer and shape preserving network that transfers the 3D actions to a set of 2D poses while making the 2D poses have the same shape as the input pose. Finally, we adopt the pose-guided image synthesis method by [71] for the animated frame generation.

In the following, we describe each component of our framework in detail.

3.1. 2D Pose Lifting to 3D Skeleton

We need to find the most similar 3D skeleton A_0 in the MoCap dataset for P_0 and the camera projection matrix C_0 that best projects A_0 to P_0. The work of [20] has proposed a camera and skeleton determination method between 2D poses and 3D skeletons. However, their method is based on model fitting and optimization which is time-consuming for large-scale dataset. Training a network that maps a 2D pose to 3D skeleton is also cumbersome [3; 70; 37]. Since we just want to find the skeleton that has similar actions as the 2D pose, we solve this problem by proposing the following virtual surrounding projection scheme together with several pose similarity metrics.

Virtual Surrounding Projection. We collect walk skeleton sequences of CMU MoCap dataset [8]. In order to find A_0 in the MoCap data that is most similar to P_0, our basic idea is to project all the 3D skeletons to 2D space from sufficiently different viewpoints, and then find the most similar pose $P_0^{'}$ in the 2D dataset obtained by projection through pose similarity metrics. With $P_0^{'}$, we can obtain A_0 correspondingly and the projection matrix C_0 from A_0 to $P_0^{'}$.

We propose a virtual surrounding projection scheme described as follows. For each skeleton in the CMU MoCap dataset [8], we move its root joint to the origin $(0, 0, 0)$. Then, as shown in Fig. 2, we evenly place a set of cameras (20 in our setting) around the skeleton on the X-Z plane. The cameras are used to project the skeleton to a set of 2d poses. This group of cameras forms a circle, and every one is oriented towards the 3D skeleton. Let r be the radius of the circle, α be the rotation angle of a camera, f be the focal length of these cameras, w and h be the width and height of captured image, and d_x and d_y be the scaling factors with respect to the image resolution. The rotation matrix R, translation matrix T and the intrinsic calibration matrix I of a camera are represented

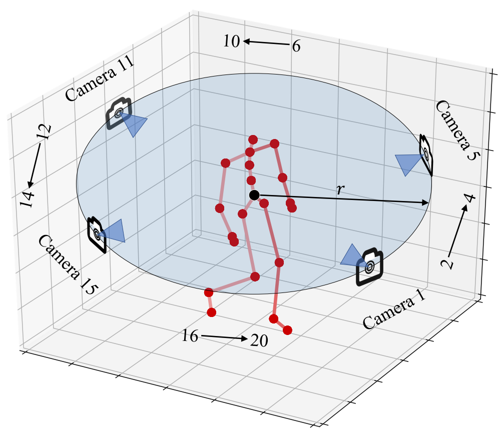


Figure 2: Virtual surrounding projection a 3D skeleton to 2D poses.

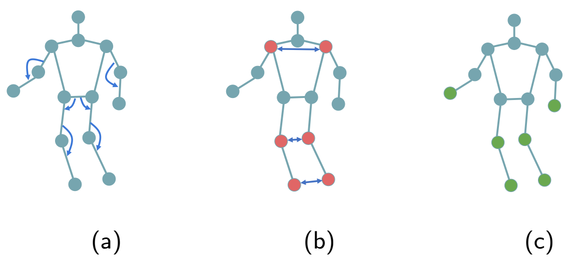


Figure 3: (a) Representative angles in a pose. (b) Pairs of key joints between which relative distances are computed. (c) Key joints whose absolute positions are used.

by:

$$R = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}, T = \begin{bmatrix} 0 \\ 0 \\ r \end{bmatrix}, \tag{1}$$

and

$$I = \begin{bmatrix} \frac{1}{d_x} & 0 & u_0 \\ 0 & \frac{1}{d_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2}$$

where (u_0, v_0) is the principal point at the center of image, which means $u_0 = w/2$ and $v_0 = h/2$. The projection matrix is defined as:

$$C = I \begin{bmatrix} R & T \\ \vec{0} & 1 \end{bmatrix} \tag{3}$$

In this way, we obtain a set of projections of a 3D skeleton by a discrete group of virtual cameras. We have tested moving the camera along the Y axis or shifting it towards or away from the skeleton, but found the current projection scheme is the most efficient while at the same time sufficiently simulating the full projection space.

2D Pose Similarity Metrics. After obtaining all the 2D poses projected from 3D skeletons, we need to find the one among them that is the most similar to P_0. Note that P_0 detected by OpenPose [5] has 14 key points (see Fig. 3), while the 3D MoCap data involves 21 key points (see Fig. 2). To retrieve the most similar pose, at the first glance, we need to convert the 3D MoCap data to the format of OpenPose, making the 3D pose have 14 key points too. But in practice, we do not retarget the 3D MoCap data to make a 3D pose have the same configuration as the OpenPose 2D skeleton. Instead, we directly compare a 14-joint 2D pose with a 21-joint projected pose between the corresponding points of the two kinds of poses. For example, we identify which two points in the 14 key points indicate an upper arm and which two points in the 21 key points represent the same arm. Then, we can calculate the similarity between the corresponding joints using the following method.

We design similarity metrics between 2D poses from three aspects: difference in representative angles of pose, difference in relative positions between key joints of pose, and difference in absolute positions of key joints of pose, as shown in Fig. 3. The difference in angles between two 2D poses consists of 3 kinds of angles: bending angles of legs, bending angles of arms, and angles between waist and legs as shown in Fig. 3 (a). Bending angles of legs and arms can indicate the state of walking, but the pedestrian may step on different legs given the same set of angles of limbs. Therefore, we use the angles between waist and legs for disambiguation. The difference in angles between two 2D poses is:

$$D_{Angle}(P, \hat{P}) = \sum_{k=1}^{K} \left\| \theta_k - \hat{\theta}_k \right\|^2 \tag{4}$$

where P and $\hat{P}$ are two 2D poses, θ_k denotes the angle defined above which is calculated by the vector inner product, and $K = 6$.

The difference in relative positions between 2D poses is defined over distances between joints of shoulders, knees, and feet of pose, which can represent macroscopic features such as the orientation of 2D poses. The three joint pairs for computing relative positions are indicated by the arrows in Fig. 3 (b). We compute the following loss over them:

$$D_{Relative}(P, \hat{P}) = \sum_{m=1}^{M} \left\| R_m - \hat{R}_m \right\|^2 \tag{5}$$

where R_m denotes the distance between a pair of joints, $\|R_m - \hat{R}_m\|$ computes the difference of relative distances, and $M = 3$.

The difference in absolute locations of key joints between two 2D poses is used as a complement. The key joints used are shown in Fig. 3 (c), including hands, knees and ankles. This difference is defined as:

$$D_{Position}(P, \hat{P}) = \sum_{j=1}^{J} \left\| p_j - \hat{p}_j \right\|^2 \tag{6}$$

where p_j denotes a key joint, and J denotes the total number of key joints used which is 6.

With the above three differences defined, the similarity between two 2D poses is:

$$S(P,\hat{P}) = D_{Angle}(P,\hat{P}) + D_{Relative}(P,\hat{P}) + D_{Position}(P,\hat{P}) \quad (7)$$

With the 2D pose similarity metric, we search the whole 2D projected pose dataset for the pose P_0' which minimizes the similarity defined in Eq. 7. The searching process is accelerated by KD-Tree technique. Accordingly, we can obtain the 3D skeleton A_0 that is projected to P_0' in the virtual surrounding projection stage with projection matrix C_0.

3.2. 3D Action Synthesis Network

We synthesize motion in 3D space since MoCap datasets are relatively abundant and we would like focus on the actions themselves while ignoring factors such as human bone lengths and different camera viewpoints that should be considered in 2D space.

We collect freely available MoCap datasets [8; 36; 39; 54; 18; 21; 17; 63; 27; 24] and unify them to obtain a single MoCap dataset by retargeting different characters in different datasets to the same basic character. The MoCap dataset contains many other kinds of human actions, while we just retain the actions of walking.

Let $S = (A_0, A_1, \cdots, A_{M-1})$, where $S \in \mathbb{R}^{M\times 63}$, be a skeleton sequence with $M = 240$. From S, we extract the trajectory $T \in \mathbb{R}^{M\times 3}$ which is computed as the root position differences along the X and Z axes and the forward direction differences a round t he Y -axis b etween adjacent skeletons A_i and A_{i+1} in S. In addition, we also compute foot contact labels $G \in \mathbb{R}^{M\times 4}$ from S, which indicate whether the toe and heel of the left and right foots of each skeleton are below a certain height and velocity or not. The footstep contact label G is computed automatically in our system. Specifically, we train a regressor that outputs G given a trajectory T following the method described in [18]. The user only needs to provide a trajectory T by such as Bézier interactive tool at test time.

With the above preparations, our aim is to synthesize S given A_0, T and G, and we achieve this by training a neural network. Following [18], our motion synthesis network is shown in Fig. 4 which contains two steps. In the first step, we train an autoencoder comprising encoder E_θ and decoder D_ϕ, in which E_θ accepts a skeleton sequence $S = (A_0, A_1, \cdots, A_{M-1})$, and D_ϕ tries to reconstruct the input sequence and outputs $\hat{S} = (\hat{A}_0, \hat{A}_1, \cdots, \hat{A}_{M-1})$. The hidden representation in the bottleneck learned by the autoencoder is in the space $\mathbb{R}^{\frac{M}{2}\times d}$ with $d = 256$. We train the autoencoder by the following loss:

$$L_{AE} = \sum_{dataset} \sum_{i=0}^{M-1} \|A_i - \hat{A}_i\|_2^2. \quad (8)$$

After training the autoencoder, the encoder E_θ is discarded and the parameters of the decoder D_ϕ are fixed. We design another encoder E_ψ and link it with D_ϕ to form the second encoder-decoder network. Instead of motion sequence, the

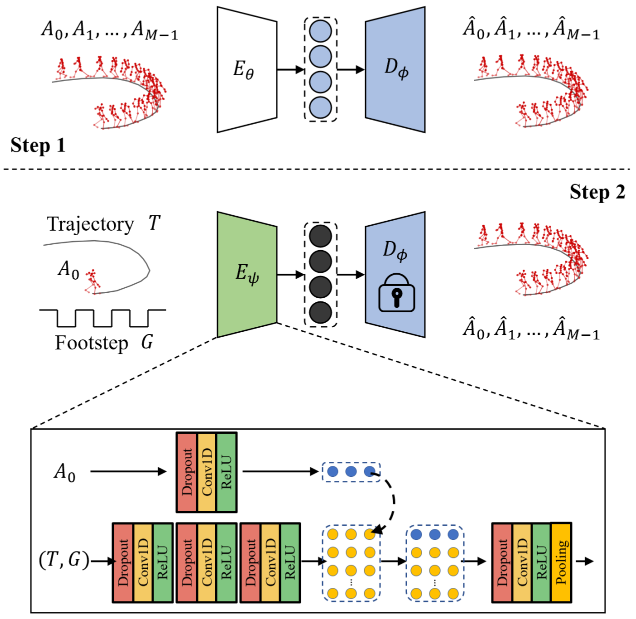

$A_0, A_1, \dots, A_{M-1}$
E_θ
D_ϕ
$\hat{A}_0, \hat{A}_1, \dots, \hat{A}_{M-1}$
Step 1
Step 2
Trajectory T
A_0
Footstep G
E_ψ
D_ϕ
$\hat{A}_0, \hat{A}_1, \dots, \hat{A}_{M-1}$
A_0
Dropout
Conv1D
ReLU
(T,G)
Pooling

Figure 4: 3D action synthesis network is composed of two encoder-decoder networks trained asynchronously. At step 1, an autoencoder is trained to reconstruct 3D skeleton sequences. At step 2, the decoder in the autoencoder is frozen and linked with another encoder. The decoder still outputs skeleton sequences but the inputs to the new encoder are the first skeleton A_0, the trajectory T, and the foot contact labels G.

data input into E_ψ are high-level control signals A_0, T and G, while the output of D_ϕ is still $\hat{S}$. The parameters of E_ψ are trained by the loss in Eq. 8 too.

For E_θ and D_ϕ, we borrow the network architectures in [18], while for E_ψ we design a different network, since besides T and G we introduce a new input A_0. As shown in Fig. 4, E_ψ is composed of two branches, one processing the concatenation of A_0, T_0 and G_0 where T_0 and G_0 are the first element of T and G respectively, the other one processing the concatenation of T and G. The first branch just contains a single convolutional block comprising a dropout layer, 1D convolutional layer, and a ReLU activation layer. The second branch consists of three such blocks. The first branch outputs features in the space $\mathbb{R}^{1\times d}$, while the second branch outputs features of size $\mathbb{R}^{M\times d}$. We replace the first row of the second feature map with the feature vector of the first branch. The combined features are then fed into a convolutional block and a pooling layer, obtaining features in the space $\mathbb{R}^{\frac{M}{2}\times d}$ that can be input into the decoder D_ϕ.

At test time, A_0 is obtained by the pose retrieving method in Section 3.1, T is specified by user, and G is computed from T by a simple feedforward network also trained on the MoCap dataset [18].

3.3. Action Transfer and Shape Preserving Network

After action synthesis, we obtain a skeleton sequence $\hat{S} = (\hat{A}_0, \hat{A}_1, \cdots, \hat{A}_{M-1})$. In this section, we show how to

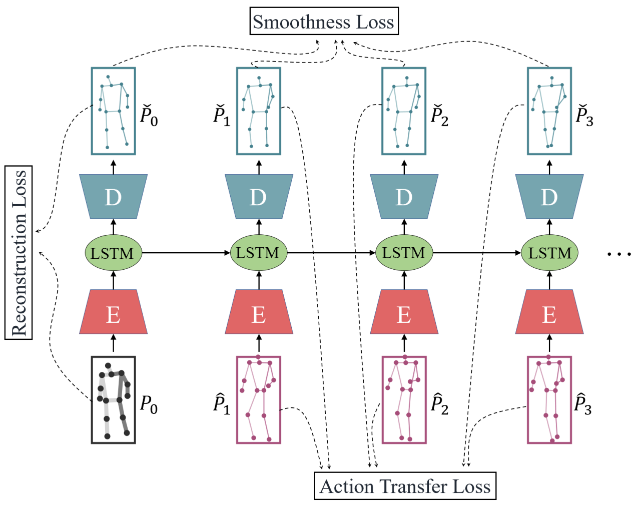


Figure 5: The action transfer and shape preserving network. The architecture of the network is an encoder-LSTM-decoder structure. The losses used to train it are carefully designed, including a reconstruction loss for preserving shape of input 2D pose, an action transfer losses and a temporal smoothness loss.

transfer the actions of these skeletons to a set of 2D poses while at the same time making these poses have the same shape as the pose P_0 of the input person.

Firstly, we project $\hat{S}$ into the 2D space by C_0, obtaining a set of poses $\hat{P}_0, \hat{P}_1, \cdots, \hat{P}_{M-1}$. Then we design a Long Short-Term Memory (LSTM) network [16] that takes P_0 and $\hat{P}_1, \cdots, \hat{P}_{M-1}$ as input, and outputs the target pose sequence $\check{P}_0, \check{P}_1, \cdots, \check{P}_{M-1}$. As shown in Fig. 5, the action transfer and shape preserving network is very simple. Each input pose is firstly processed by an encoder, changing pose into feature space. Then the features are processed by LSTM cell, the output of which is decoded by a decoder to obtain the corresponding output pose. The encoder comprises two linear layers and a ReLU activation layer, mapping the input pose into space of $\mathbb{R}^d$ ($d = 256$). The decoder has the same architecture as the encoder, but mapping feature in $\mathbb{R}^d$ into the pose space.

3.3.1. Training Losses

While the network architecture is simple, the loss function used to train the model should be carefully designed in order to achieve the action transfer and shape preserving goals. Our loss function contains absolute reconstruction term, relative orientation and position terms, and smoothness term.

Firstly, we define a reconstruction loss between P_0 and $\check{P}_0$, forcing the network to inherit shape of P_0, which is defined as the mean square error between the corresponding joints of P_0 and $\check{P}_0$:

$$L_{recon}(P_0, \check{P}_0) = \sum_{dataset} \sum_{j=1}^{J} \left\| \check{P}_{0,j} - P_{0,j} \right\|_2^2, \tag{9}$$

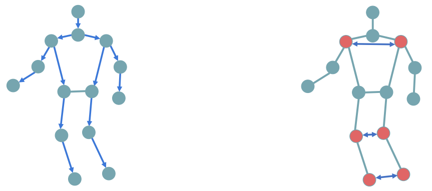


Figure 6: (a) Limbs whose orientations are computed. (b) Pairs of key joints between which relative distances are computed.

where $\check{P}_{0,j}$ denotes the absolute position of the j^{th} joint and J is the total number of the joints.

Secondly, for poses from timestep 1 to $M-1$, we do not have corresponding ground-truth poses to compute the absolute distance as in Eq. 9. Instead, we compute relative distances between them and the pose of the input person, including limb orientation loss and relative position loss of key joints, to make the generated poses have the same action as the input poses. The limb orientation loss function is defined as:

$$L_{orient} = \sum_{dataset} \sum_{i=1}^{M-1} \sum_{k=1}^{K} \left\| \hat{D}_{i,k} - \check{D}_{i,k} \right\|_2^2 \tag{10}$$

where $\hat{D}_{i,k}$ denotes the orientation of the k^{th} limb of pose $\hat{P}_i$, computed as the unit vector of the limb in the 2D coordinate system, $\check{D}_{i,k}$ denotes the orientation of the corresponding limb of pose $\check{P}_i$, and $K = 14$ is the number of the limbs which are shown in Fig. 6 (a). The relative position loss function is:

$$L_{relative} = \sum_{dataset} \sum_{i=1}^{M-1} \sum_{n=1}^{N} \left\| \hat{R}_{i,n} - \check{R}_{i,n} \right\|_1 \tag{11}$$

where $\hat{R}_{i,n}$ denotes the distance between the n^{th} pair of key joints of pose $\hat{P}_i$, $\check{R}_{i,n}$ is the corresponding distance in pose $\check{P}_i$, $N = 3$ denotes the number of the pairs of key joints, including shoulders, knees and ankles as shown in Fig.6 (b).

Finally, in order to make the generated pose sequence look more smooth, we introduce a temporal smoothness loss:

$$L_{smooth} = \sum_{dataset} \sum_{i=0}^{M-2} \sum_{j=1}^{J} \left\| \check{Y}_{i+1,j} - \check{Y}_{i,j} \right\|_1 \tag{12}$$

where $\hat{Y}_{i,j}$ denotes the height of the j^{th} joint of pose $\hat{P}_i$, $\check{Y}_{i,j}$ denotes the height of the corresponding joint in pose $\check{P}_i$, J is the total number of joints in the pose. We consider only "height" in L_{smooth} because we find in the experiments that the up and down shaking is the most perceivable undesired visual artifacts in our results. Therefore, we try to smooth the Y coordinates in a sequence. As for the X coordinate, it is usually entangled with actual movements of a person. Since we would like to preserve high-fidelity object movements, we do not smooth the X coordinate.

The full loss function is defined as:

$$L = \lambda_1 L_{recon} + \lambda_2 L_{orient} + \lambda_3 L_{relative} + \lambda_4 L_{smooth} \quad (13)$$

During training and testing, we set $\lambda_1 = 1$, $\lambda_2 = 5$, $\lambda_3 = 1$, $\lambda_4 = 10$.

The above provides self-supervised training losses for the action transfer and shape preserving network. That is, we do not provide ground truths for the output poses. Instead, we design reconstruction loss, limb orientation loss, relative position loss, and temporal smoothness loss, which are either defined between the outputs themselves or between the output and the input. The reconstruction loss and smoothness loss are used to inherit shape from P_0, while the limb orientation loss and relative position loss are used to inherit actions from other input poses.

Since the losses used to train the action transfer and shape preserving network are not very complex, one can directly optimize the input poses by quadratic programming to obtain the target poses. The optimization-based approach is also able to generate results. However, the optimization is prone to local minima and sometimes time-consuming. The proposed LSTM-based regression approach, though requiring training, is fast and robust at test time.

3.4. Pose-guided Image Synthesis for Pedestrian Animation

Now, we have predicted future poses $\check{P}_1, \cdots, \check{P}_{M-1}$ for the input pedestrian image I_0. We finally employ the pose-guided image synthesis model [71] to generate future images for the pedestrian.

4. Experiments

In the following, we conduct experiments to evaluate the effectiveness of our method. Please refer to the supplemental materials for more information.

4.1. Training and Evaluating Datasets

The 3D action synthesis network is trained on the 3D MoCap dataset as described in Section 3.2. The action transfer network is trained on the synthesized 3D skeletons and 2D pose of images. We test our method on three datasets including Market-1501 [67], DeepFashion [29], and Human3.6M [21].

4.2. Implementation Details

There are two networks to train in our method: the 3D action synthesis network, action transfer and shape preserving network. We train the 3D action synthesis network by Adam optimizer with learning rate of $1e-5$, β_1 of 0.9, β_2 of 0.999, and batch size of 1. We first train the encoder E_θ and the decoder D_ϕ for 100 epochs, then train the encoder E_ψ for another 100 epochs. The kernel and stride size of the 1D convolutions and the rate of dropout follow the settings of [18]. For action transfer and pose shape preserving network,

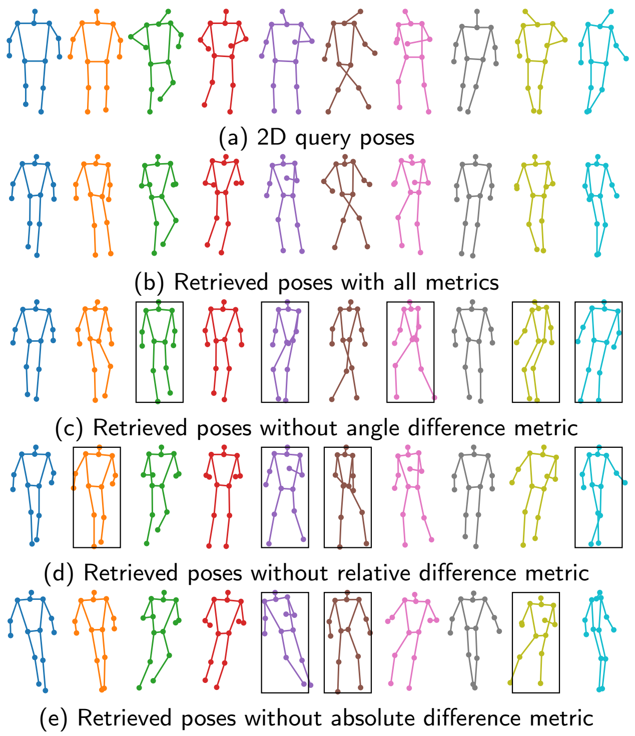

(a) 2D query poses
(b) Retrieved poses with all metrics
(c) Retrieved poses without angle difference metric
(d) Retrieved poses without relative difference metric
(e) Retrieved poses without absolute difference metric

Figure 7: Validation of the 2D pose similarity metrics. The retrieved poses in (b) are very similar to those in (a). In each of (c)-(e), we exclude one metric. As can be seen, there exist bad poses retrieved, shown with black boundingbox.

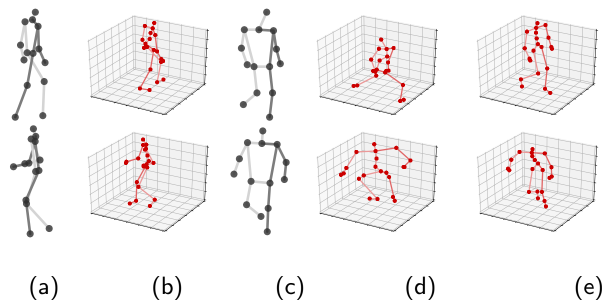


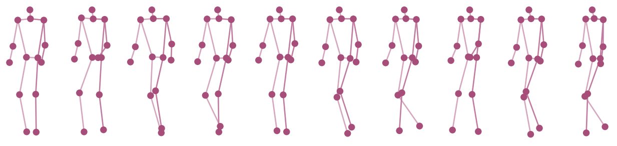
Figure 8: (a) 2D poses projected from MoCap skeletons. (b) 3D skeletons of (a) computed by neural network. (c) 2D poses from Market-1501. (d) 3D skeletons of (c) computed by neural network. (e) 3D skeletons of (c) computed by retrieving.

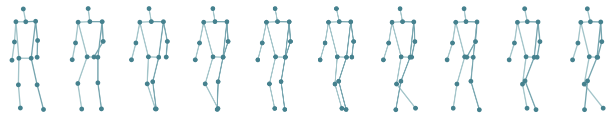
we train it by RMSprop optimizer with learning rate of $5e-5$ and batch size of 64 for 2000 epochs.

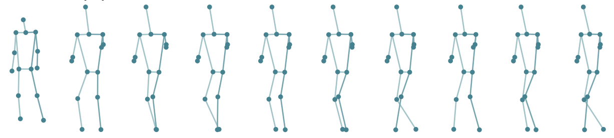
4.3. Ablation Study

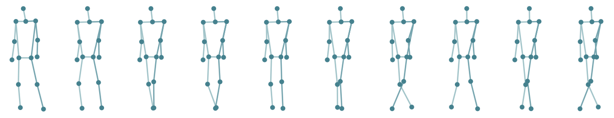
We conduct various ablation studies to validate the design choices of the proposed method.

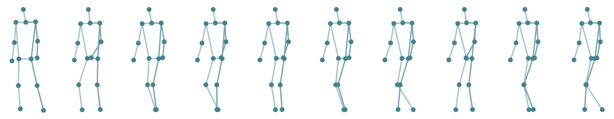
Firstly, we show examples of queried poses and the corresponding retrieved poses using the proposed pose similarity metrics defined in Section 3.1. The similarity metrics include difference in angles, difference in relative distances between shoulders, knees and feet, and difference in absolute positions of key joints. In order to validate the importance of the three kinds of metrics, we show the poses retrieved


(a) Projected poses from 3D skeletons


(b) Poses after action transfer with all losses


(c) Poses after action transfer without reconstruction loss


(d) Poses after action transfer without orientation loss


(e) Poses after action transfer without relative loss

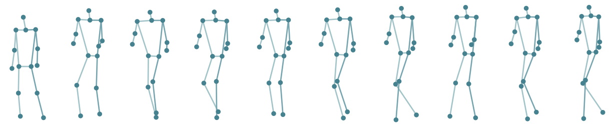
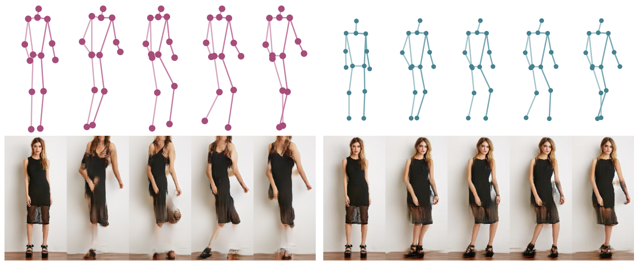

(f) Poses after action transfer without smoothness loss

Figure 9: Ablation results of losses used to train the action transfer and shape preserving network. (a) shows the actions to be transferred. (b) shows results with all losses. From (c) to (f), one kind of loss is removed at each time to show the effect of that loss.

without one of them. The pose retrieving results are shown in Figure 7 in which the first row gives the poses for query, the second row shows retrieved results with all comparison metrics, and the last three rows show poses retrieved without angle difference, relative difference, and absolute difference metric, respectively. As can be seen, the poses in the second row are all very similar to their corresponding query poses. While in other rows, there exist some poses marked with bounding boxes that are visually different from the query ones.

An alternative way to find the most similar skeleton for a 2D pose is to directly train a neural network between the 2D poses and 3D skeletons. We have designed and trained such a network that maps the 2D pose dataset obtained by virtual surrounding projection to the corresponding 3D MoCap dataset. During testing, the network works very well for the poses in the 2D pose dataset as shown in Figure 8 (a) and (b) where (a) shows two poses from the 2D pose dataset and (b) shows the output skeletons by the network. However, when tested on poses extracted from real pedestrian images (see Figure 8 (c) from Market-1501), the network performs badly whose results are shown in (d). This is due to the reason that the poses extracted from specific pedestrian datasets are different from ones in the 2D pose dataset. For example, they may have different bone lengths and the ratio between width and height is also very different. The network cannot generalize well to the poses that it has never seen. Figure 8 (e) gives the results by our pose retrieving scheme, which are much better than those in (d).


(a) (b)

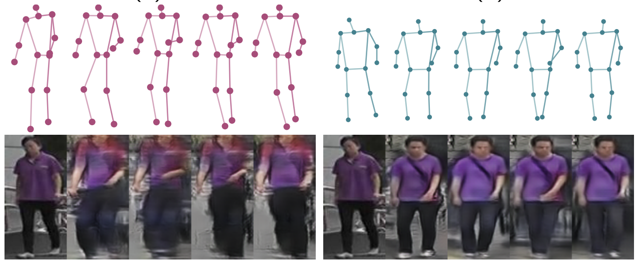
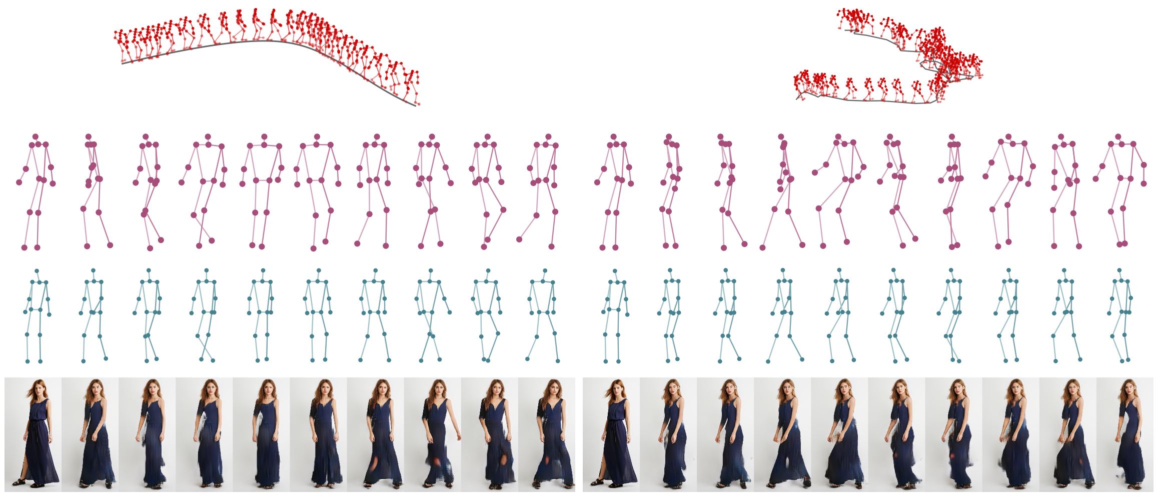

(c) (d)

Figure 10: (a)(c) Poses projected from 3D skeletons and the corresponding synthesized frames guided by the poses. (b)(d) Poses after action transfer and shape preserving network and the corresponding synthesized frames guided by the poses.

In Figure 9, we evaluate the necessity of the four losses used to train the action transfer and shape preserving network described in Section 3.3. The four losses include: reconstruction loss for pose shape characteristic preserving, orientation and relative losses for action transfer, and smoothness loss. In Figure 9, the first row gives 2D poses projected from the 3D space, providing actions to be transferred. The second row shows the output of the action transfer and shape preserving network trained with all the losses. The third to sixth rows show results of the network trained without the reconstruction, orientation, relative, and smoothness loss respectively. Note that the first poses in (b)-(f) are the same, which are the pose extracted from the input pedestrian image. As can be seen, the poses in (b) look very natural, everyone taking the same action as the corresponding pose in (a), all of them having consistent shape as the input 2D pose. Since there is no reconstruction loss, the poses in (c) do not hold the same shape as the input pose. The most obvious difference is in the length of neck: those of the generated poses are much longer than that of the input pose. Some poses in (d) look very strange due to missing of the orientation loss, such as the last one in this



(a) 3D skeletons along a Bézier curve (b) 3D skeletons along a randomly generated curve

Figure 11: For the same person, two different animations are generated given different trajectories.

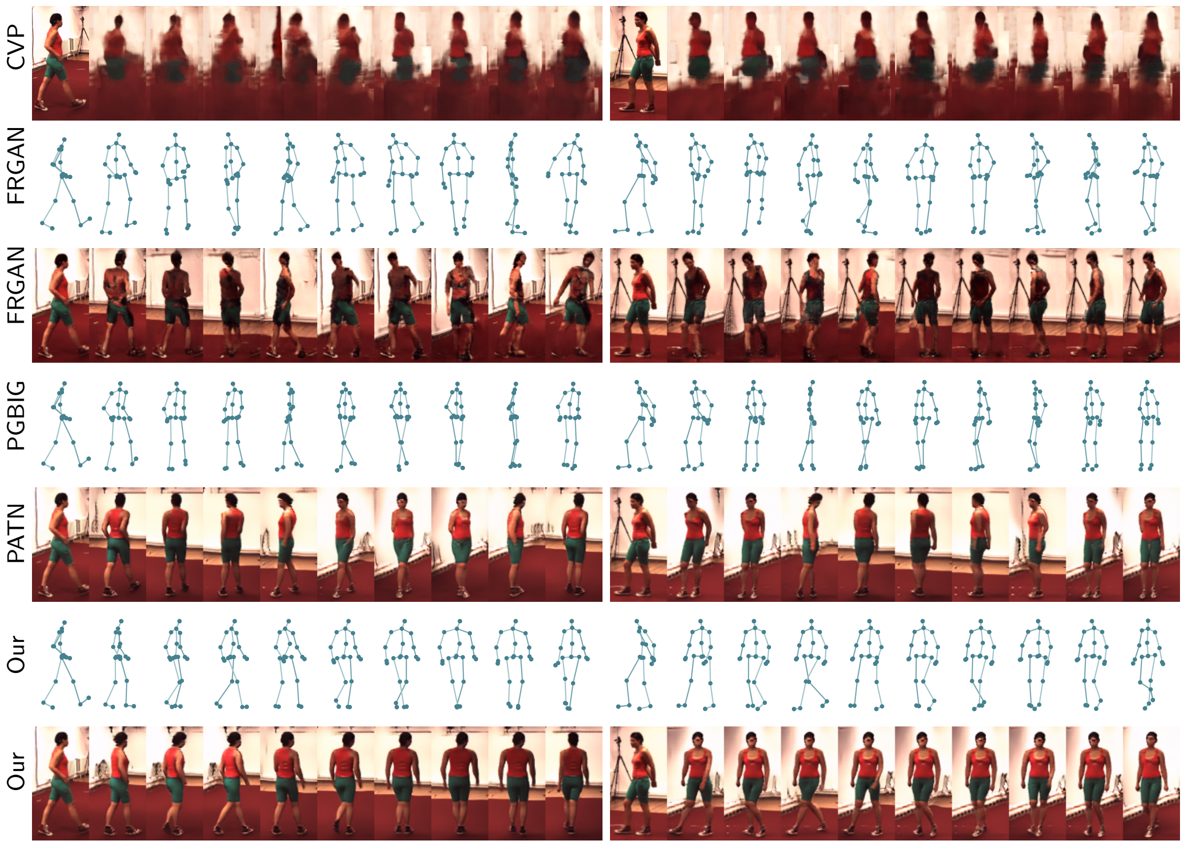


Figure 12: Comparisons between our method and PGBIG+PATN, FRGAN and CVP on examples of Human3.6M. The results of CVP are with blurry artifacts. Our results are better than those of FRGAN in both future poses and frames. Our results are better than those of PGBIG+PATN in poses.

row that has crossed legs. The poses in (e) are very similar to those in (b), except that the areas of the torsos are slightly smaller. Without the smoothness loss, the poses in (f) show obvious discontinuities.

In Figure 10, we show the importance of the action transfer and shape preserving network itself. In this figure, we show two examples, one from the DeepFashion dataset ((a) and (b)), and the other one from Market-1501 ((c) and (d)). The left half ((a) and (c)) shows the poses projected from

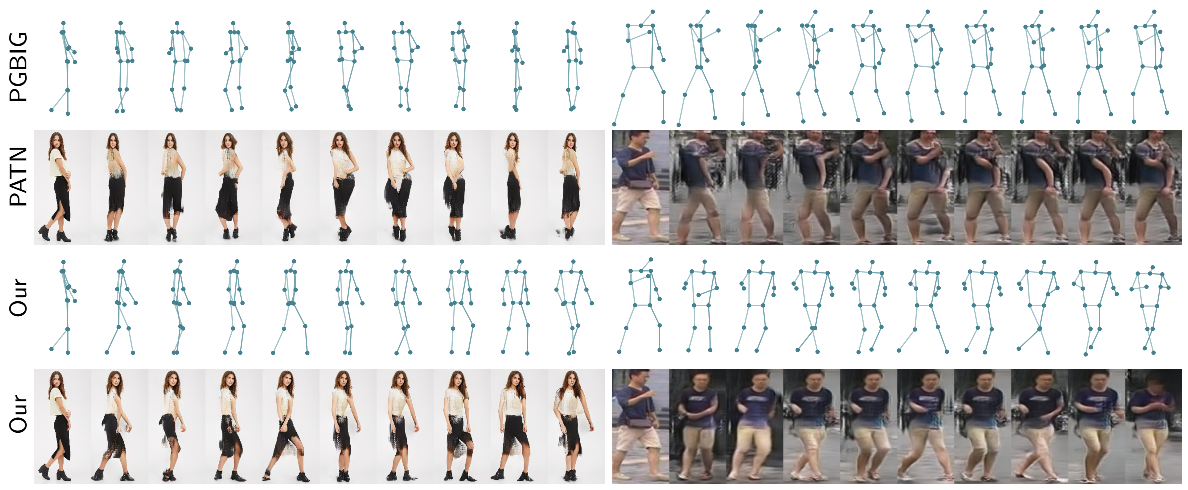


Figure 13: Comparisons between our method and PGBIG+PATN on examples of DeepFashion and Market-1501. Here, PGBIG is trained on Human3.6M. When tested on DeepFashion and Market-1501, it shows many artifacts. By contrast, our method shows higher ability of generalibility, which outputs better results in both poses and frames. Note that our method is also not trained on the DeepFashion and Market-1501 datasets.

the synthesized 3D skeletons and the images synthesized according to these poses. As can be seen, the synthesized images contain lots of artifacts. This is because the pose-guided image synthesis model is trained on each pedestrian dataset using the poses extracted from the dataset, while the projected poses are not similar to extracted poses from the specific pedestrian dataset. The action transfer and shape preserving network is used to transform the poses on the left into the pose shape space of the corresponding pedestrian dataset, obtaining poses on the right ((b) and (d)) for which the pose-guided image synthesis model works well and synthesizes better images.

Figure 11 shows examples about user interaction. For an example from DeepFashion, we allow a user to specify two different trajectories for the 3D action synthesis network to plan different future actions. In the first example of this figure (Figure 11 (a)), we use Bézier as the interactive tool. For the second example (Figure 11 (b)), the trajectory is randomly generated. For each example, the first row shows the generated 3D skeletons, the second row shows 2D poses projected from 3D, the third row gives poses after action transfer and shape preserving network, and finally the last row shows the synthesized images. As can be seen, our method successfully generates plausible future frames for the pedestrian, with different actions.

4.4. Comparison with Previous Approaches

We compare our method with three approaches, including PGBIG [31]+PATN [71], FRGAN [66], and CVP [60]. FRGAN is a two-stage generation network that transforms an image to video. The method comprises a future motion generation network, a motion-guided image synthesis network, and a refining network that considers temporal information. Their future motion generation network predicts future poses based on a single pose, which is an extremely difficult task due to the large ambiguities in the future. We alleviate this problem by performing action synthesis in the 3D space with trajectory and foot contact information for disambiguation. Since FRGAN uses a relatively simple pose prediction model [12], we thus compare with a combination of the latest pose prediction model PGBIG [31] and the pose-guided image synthesis method PATN [71] (note that we also use PATN for pose-guided frame generation). Finally, CVP [60] is a future video frame prediction method without explicit structure guidance.

These compared approaches all make their codes public, and we retrain them on Human3.6M. We select the 6 subjects having 2D poses as training dataset. Sequences of 240 frames at 60fps are extracted to train the compared models. PGBIG [31] was originally designed for 3D skeleton. We modify it to take 2D pose as input. CVP [60] needs a set of points as input, representing the entities for which the future positions are predicted. In our cases, the joints of the first pose are treated as these entities.

Figure 12 shows the comparison results between these approaches and our method on Human3.6M. We show two examples on the left and right of the figure. For each example and each method, we show the future poses on the top, and the corresponding future frames right under these poses. The first row shows image results of CVP. Since CVP does not predict poses, there is no pose result for CVP. The second to third rows show results of FRGAN. The fourth to fifth rows show results of PGBIG+PATN. The last two rows show our results. Apparently, CVP outputs worst results with lots of blurry artifacts. Compared with the other two competitors, our method is better in the plausibility of the future poses. As can be seen, the poses generated by our method takes many steps forward, while the poses by FRGAN and PGBIG are almost static. From the aspect of future frames, the results of FRGAN contain lots of artifacts, while the results by PGBIG+PATN look plausible except the actions in these frames.

Figure 13 shows the comparisons between our method and PGBIG+PATN on Market-1501 and DeepFashion. The other two approaches cannot be applied to the two datasets, as they cannot be trained when there is no sequential data. PGBIG is trained on Human3.6M, but we test it on Market-1501 and DeepFashion. We choose one example from each dataset, and show the predicted poses and frames for each example. As can be seen, the poses generated by PGBIG are much worse than ours, demonstrating that PGBIG trained on the poses of Human3.6M fails to process poses from Market-1501 and DeepFashion. The action transfer and shape preserving network in our method helps our method smoothly adapts to different datasets. Accordingly, the future frames generated by PATN based on the poses of PGBIG are blurry and distorted, while our results are much better.

4.5. User Study

There is no standard metric to quantitatively evaluate the quality of the generated future poses and images, as there is no ground truth. Therefore we carry out a user study for this aim. We randomly choose 10 pedestrian images from Human3.6M, and use FRGAN, CVP, PGBIG+PATN, and the proposed method to generate future poses and frames for these images (CVP can only produce future frames). We then convert the pose and frame sequences into GIF images. For each pedestrian, we place the GIFs of poses side by side in a row and the GIFs of frames in the other row right below the poses. The results of the same method are in the same column, while the orders of different methods along the row are randomly disturbed. In this way, we obtain 10 groups of results. We then recruit 10 participants without any training and show each one the 10 results. They are asked to answer two questions after watching each group of results. The first question is: which pose do you think is the most natural, realistic and smooth? The second question is: which image do you think is the most natural, clear and has the least artifacts?

The results of the user study are shown in Table 1. For both questions, most of the participants (nearly eighty percent) hold the view that our results are the best. The second best is PGBIG+PATN. This is because compared with our method, though PGBIG can not predict detailed actions for the pedestrians, it has indeed learned global rotations of them. Therefore, some participants may be cheated by the overall smooth rotation impression of the results of PGBIG+PATN, without noticing the nearly static body shape. The results of FRGAN and CVP are apparently worse, therefore much fewer participants recommend them as the best.

4.6. Running Time

Our method comprises several steps. In Table 2, we list the time used in each stage at test time for our approach. Firstly, the pose estimation stage, i.e., extracting pose from a given pedestrian image, is very fast, just taking 0.036s on average. Since we need to traverse a very large 2D pose dataset, we use KD-Tree to accelerate the pose retrieving stage. On average, it takes 1.3s. The three networks are very fast, taking 1.102s, 0.004s, and 0.112s respectively. The whole time needed to animate a pedestrian costs about 40s.

Table 1
A user study used to compare our method with FRGAN, CVP, PGBIG+PATN quantitatively. 77.5% participants on average view our results as the best.

	FRGAN	CVP	PGBIG + PATN	Ours + PATN
Q1	6%	0%	15%	79%
Q2	6%	1%	17%	76%
Avg.	6%	0.5%	16%	77.5%

Table 2
Time used in each stage of the proposed pedestrian animation framework.

Step	Running time
Pose Estimation	0.036s
Retrieving	1.3s
3D Action Synthesis	1.102s
Action Transfer and Shape Preserving	0.004s
Image Synthesis	0.112s
Total	2.554s

4.7. Limitations and Future Work

In this paper, we mainly focus on future pose generation. For future frame generation, we employ the state-of-the-art pose-guided image synthesis method [71]. Since the frames are generated one by one while ignoring the consistency between them, there may exist shaking and blinking artifacts in the generated frame sequence. Besides, it is difficult to generate information about occluded body parts in the given static image. For example, the left side of Figure 14 shows the same person with different poses. Since the hair in the input image is occluded, the hair in the output image is not correctly synthesized. As shown on the right side of Figure 14, the resulting pose may exhibit a degree of bow-legging (see the second image of the sequence of output), and pants might be mistakenly generated as skirts. Based on the above analysis, one task in the future is to investigate a more effective pose sequence guided future frame generation model, considering temporal consistency and handling occlusions.

5. Conclusion

In this paper, we present a novel framework for transforming a pedestrian image to a sequence of images containing dynamically walking states of the pedestrian with plausible future poses generated for the pedestrian. Our method takes a single pose of the given pedestrian as input. Resorting to 3D MoCap dataset, we plan future actions for the pedestrian in 3D space, and propose a network to transfer the 3D actions to the given 2D pose while preserving the shape of the input 2D pose. This idea makes our method able to process pedestrian datasets even they just contain images,

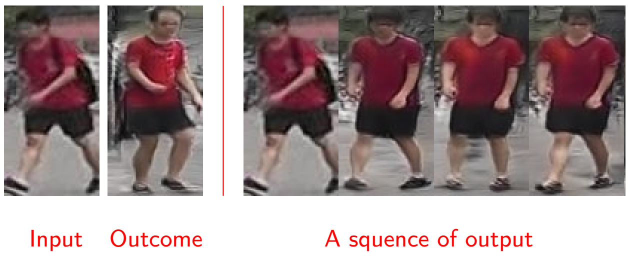


Figure 14: Failure cases. Left: occluded region (hair) is not correctly synthesized. Right: bowlegs (second image) and pants becoming skirts (second to fourth images).

thus enhancing the generality of our method. We allow users to control the action synthesis in 3D space by providing a global trajectory. We have shown that this additional information can help user to generate different animation results for the same pedestrian image. After obtaining future poses, we use the existing state-of-the-art pose-guided image synthesis method to compute the corresponding frame for each pose. In the future, we will investigate an unified pose prediction and pose-guided frame generation framework for more temporally consistent animation videos.

Acknowledgements

This work was supported in part the National Natural Science Foundation of China (NSFC) for Key Program [grant number 62237001], in part by the NSFC for General Program [grant number 62072191].

References

[1] H. Averbuch-Elor, D. Cohen-Or, J. Kopf, and M. F. Cohen. Bringing portraits to life. *ACM Transactions on Graphics (Proceeding of SIGGRAPH Asia 2017)*, 36(6):196, 2017.

[2] J. Bai, A. Agarwala, M. Agrawala, and R. Ramamoorthi. Automatic cinemagraph portraits. In *Computer Graphics Forum*, volume 32, pages 17–25. Wiley Online Library, 2013.

[3] S. Biswas, S. Sinha, K. Gupta, and B. Bhowmick. Lifting 2d human pose to 3d: A weakly supervised approach. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2019.

[4] Y. Cao, X. Pang, A. B. Chan, and R. W. Lau. Dynamic manga: Animating still manga via camera movement. *IEEE Transactions on Multimedia*, 19(1):160–172, 2016.

[5] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[6] L. Castrejon, N. Ballas, and A. Courville. Improved conditional vrnns for video prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7608–7617, 2019.

[7] Y.-Y. Chuang, D. B. Goldman, K. C. Zheng, B. Curless, D. H. Salesin, and R. Szeliski. Animating pictures with stochastic motion textures. In *ACM SIGGRAPH 2005 Papers*, pages 853–860. 2005.

[8] CMU. Carnegie-mellon mocap database. `http://mocap.cs.cmu.edu/`.

[9] L. Dang, Y. Nie, C. Long, Q. Zhang, and G. Li. Msr-gcn: Multi-scale residual graph convolution networks for human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11467–11476, 2021.

[10] L. Dang, Y. Nie, C. Long, Q. Zhang, and G. Li. Diverse human motion prediction via gumbel-softmax sampling from an auxiliary space. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 5162–5171, 2022.

[11] Y. Endo, Y. Kanamori, and S. Kuriyama. Animating landscape: Self-supervised learning of decoupled motion and appearance for single-image video synthesis. *ACM Trans. Graph.*, 38(6), Nov. 2019.

[12] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015.

[13] J.-Y. Franceschi, E. Delasalles, M. Chen, S. Lamprier, and P. Gallinari. Stochastic latent residual video prediction. In *International Conference on Machine Learning*, pages 3233–3246. PMLR, 2020.

[14] J. Geng, T. Shao, Y. Zheng, Y. Weng, and K. Zhou. Warp-guided gans for single-photo facial animation. *ACM Transactions on Graphics (TOG)*, 37(6):1–12, 2018.

[15] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] D. Holden, T. Komura, and J. Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics*, page 1–13, Aug 2017.

[18] D. Holden, J. Saito, and T. Komura. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)*, 35(4):1–11, 2016.

[19] A. Holynski, B. Curless, S. M. Seitz, and R. Szeliski. Animating pictures with eulerian motion fields. *arXiv preprint arXiv:2011.15128*, 2020.

[20] A. Hornung, E. Dekkers, and L. Kobbelt. Character animation from 2d pictures and 3d motion data. *ACM Transactions on Graphics (ToG)*, 26(1):1–es, 2007.

[21] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, jul 2014.

[22] N. Joshi, S. Mehta, S. Drucker, E. Stollnitz, H. Hoppe, M. Uyttendaele, and M. Cohen. Cliplets: juxtaposing still and dynamic imagery. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 251–260, 2012.

[23] Y.-H. Kwon and M.-G. Park. Predicting future frames using retrospective cycle gan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1811–1820, 2019.

[24] J. Lee, T. Kwon, and Y. Lee. Interactive character path-following using long-horizon motion matching with revised future queries. *IEEE Access*, page 9942–9956, Jan 2023.

[25] P. Li, H. Sun, C. Huang, J. Shen, and Y. Nie. Interactive image/video retexturing using gpu parallelism. *Computers & Graphics*, 36(8):1048–1059, 2012.

[26] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Flow-grounded spatial-temporal video prediction from still images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 600–615, 2018.

[27] H. Y. Ling, F. Zinno, G. Cheng, and M. Van De Panne. Character controllers using motion vaes. *ACM Transactions on Graphics*, Aug 2020.

[28] X. Liu, J. Yin, J. Liu, P. Ding, J. Liu, and H. Liub. Trajectorycnn: a new spatio-temporal feature learning network for human motion prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020.

[29] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[30] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015.

[31] T. Ma, Y. Nie, C. Long, Q. Zhang, and G. Li. Progressively generating better initial guesses towards next stages for high-quality human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6437–6446, 2022.

[32] T. Ma, Y. Nie, Q. Zhang, Z. Zhang, H. Sun, and G. Li. Effective video stabilization via joint trajectory smoothing and frame warping. *IEEE*

Transactions on Visualization and Computer Graphics, 26(11):3163–3176, 2019.

[33] W. Mao, M. Liu, M. Salzmann, and H. Li. Learning trajectory dependencies for human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9489–9497, 2019.

[34] J. Martinez, M. J. Black, and J. Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2891–2900, 2017.

[35] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[36] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber. Documentation mocap database hdm05. 2007.

[37] Q. Nie, Z. Liu, and Y. Liu. Lifting 2d human pose to 3d with domain adapted 3d body concept. *International Journal of Computer Vision*, 131(5):1250–1268, 2023.

[38] Y. Nie, Q. Zhang, R. Wang, and C. Xiao. Video retargeting combining warping and summarizing optimization. *The Visual Computer*, 29:785–794, 2013.

[39] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy. Berkeley mhad: A comprehensive multimodal human action database. In *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, pages 53–60. IEEE, 2013.

[40] M. Shi, W. Feng, L. Gao, and D. Zhu. Generating diverse clothed 3d human animations via a generative model. *Computational Visual Media*, 10(2):261–277, 2024.

[41] A. Siarohin, S. Lathuilière, S. Tulyakov, E. Ricci, and N. Sebe. First order motion model for image animation. In *Conference on Neural Information Processing Systems (NeurIPS)*, December 2019.

[42] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852. PMLR, 2015.

[43] J. Tang, J. Wang, and J.-F. Hu. Predicting human poses via recurrent attention network. *Visual Intelligence*, 1(1):18, 2023.

[44] J. Tompkin, F. Pece, K. Subr, and J. Kautz. Towards moment imagery: Automatic cinemagraphs. In *2011 Conference for Visual Media Production*, pages 87–93. IEEE, 2011.

[45] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018.

[46] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee. Decomposing motion and content for natural video sequence prediction. *ICLR*, 2017.

[47] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee. Learning to generate long-term future via hierarchical prediction. In *international conference on machine learning*, pages 3560–3569. PMLR, 2017.

[48] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. *arXiv preprint arXiv:1609.02612*, 2016.

[49] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision*, pages 835–851. Springer, 2016.

[50] J. Walker, K. Marino, A. Gupta, and M. Hebert. The pose knows: Video forecasting by generating pose futures. In *Proceedings of the IEEE international conference on computer vision*, pages 3332–3341, 2017.

[51] Q. Wang, J. Zhou, Z. Li, X. Sun, and Q. Yu. Robust monocular object pose tracking for large pose shift using 2d tracking. *Visual Intelligence*, 1(1):22, 2023.

[52] Y. Wang, P. Bilinski, F. Bremond, and A. Dantcheva. Imaginator: Conditional spatio-temporal gan for video generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1160–1169, 2020.

[53] C.-Y. Weng, B. Curless, and I. Kemelmacher-Shlizerman. Photo wake-up: 3d character animation from a single photo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5908–5917, 2019.

[54] S. Xia, C. Wang, J. Chai, and J. Hodgins. Realtime style transfer for unlabeled heterogeneous human motion. *ACM Transactions on Graphics (TOG)*, 34(4):1–10, 2015.

[55] C. Xiao, Y. Nie, W. Hua, and W. Zheng. Fast multi-scale joint bilateral texture upsampling. *The Visual Computer*, 26:263–275, 2010.

[56] X. Xu, L. Wan, X. Liu, T.-T. Wong, L. Wang, and C.-S. Leung. Animating animal motion from still. In *ACM SIGGRAPH Asia 2008 papers*, pages 1–8. 2008.

[57] T. Xue, J. Wu, K. L. Bouman, and W. T. Freeman. Visual dynamics: Stochastic future generation via layered cross convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2236–2250, 2018.

[58] C. Yang, Z. Wang, X. Zhu, C. Huang, J. Shi, and D. Lin. Pose guided human video generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 201–216, 2018.

[59] H. Yang, C. Yuan, L. Zhang, Y. Sun, W. Hu, and S. J. Maybank. Sta-cnn: Convolutional spatial-temporal attention learning for action recognition. *IEEE Transactions on Image Processing*, 29:5783–5793, 2020.

[60] Y. Ye, M. Singh, A. Gupta, and S. Tulsiani. Compositional video prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10353–10362, 2019.

[61] E. Zakharov, A. Shysheya, E. Burkov, and V. Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9459–9468, 2019.

[62] F. Zhang, L. Zhao, S. Li, W. Su, L. Liu, and W. Tao. 3d hand pose and shape estimation from monocular rgb via efficient 2d cues. *Computational Visual Media*, 10(1):79–96, 2023.

[63] H. Zhang, S. Starke, T. Komura, and J. Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics*, page 1–11, Aug 2018.

[64] Q. Zhang, Y. Nie, L. Zhu, C. Xiao, and W.-S. Zheng. A blind color separation model for faithful palette-based image recoloring. *IEEE Transactions on Multimedia*, 24:1545–1557, 2021.

[65] W. Zhang, J. Fang, X. Wang, and W. Liu. Efficientpose: Efficient human pose estimation with neural architecture search. *Computational Visual Media*, 7:335–347, 2021.

[66] L. Zhao, X. Peng, Y. Tian, M. Kapadia, and D. Metaxas. Learning to forecast and refine residual motion for image-to-video generation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 387–403, 2018.

[67] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pages 1116–1124, 2015.

[68] Y. Zhou and T. L. Berg. Learning temporal transformations from time-lapse videos. In *European conference on computer vision*, pages 262–277. Springer, 2016.

[69] Y. Zhou, Y. Song, and T. L. Berg. Image2gif: Generating cinemagraphs using recurrent deep q-networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 170–178. IEEE, 2018.

[70] Y. Zhu and D. Picard. Decanus to legatus: Synthetic training for 2d-3d human pose lifting. In *Proceedings of the Asian Conference on Computer Vision*, pages 2848–2865, 2022.

[71] Z. Zhu, T. Huang, B. Shi, M. Yu, B. Wang, and X. Bai. Progressive pose attention transfer for person image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2347–2356, 2019.