PoseTween: Pose-driven Tween Animation

Jingyuan Liu
Hong Kong University of Science and Technology

Hongbo Fu*
City University of Hong Kong
hongbofu@cityu.edu.hk

Chiew-Lan Tai
Hong Kong University of Science and Technology

Figure 1. We present *PoseTween*, a system for allowing novice users to easily create pose-driven tween animation of virtual objects.

ABSTRACT

Augmenting human action videos with visual effects often requires professional tools and skills. To make this more accessible by novice users, existing attempts have focused on automatically adding visual effects to faces and hands, or let virtual objects strictly track certain body parts, resulting in rigid-looking effects. We present PoseTween, an interactive system that allows novice users to easily add vivid virtual objects with their movement interacting with a moving subject in an input video. Our key idea is to leverage the motion of the subject to create pose-driven tween animations of virtual objects. With our tool, a user only needs to edit the properties of a virtual object with respect to the subject's movement at keyframes, and the object is associated with certain body parts automatically. The properties of the object at intermediate frames are then determined by both the body movement and the interpolated object keyframe properties, producing natural object movements and interactions with the subject. We design a user interface to facilitate editing of keyframes and previewing animation results. Our user study shows that PoseTween significantly requires less editing time and fewer keyframes than using the traditional tween animation in making pose-driven tween animations for novice users.

Author Keywords

Tween animation; human pose; interpolation; interface.

*Corresponding author.

UIST '20, October 20–23, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7514-6/20/10 ...$15.00.
https://doi.org/10.1145/3379337.3415822

CCS Concepts

•**Human-centered computing** → **Human computer interaction (HCI); Graphics input devices;** User studies;

INTRODUCTION

Adding visual effects to enhance videos has become popular among ordinary mobile users nowadays. Professional tools for adding visual effects, such as Adobe Premiere and Foundry Nuke, typically require trained skills. The advancement of real-time face detection and tracking technologies has recently enabled various camera mobile apps that support live face stickers (e.g., Sweet Snap app supports 2800 live stickers and are being used by over 100 million users). In such recreational apps, the pre-defined visual effects are aligned to track facial parts, which have a fixed structure. The recent efforts toward real-time human pose estimation (such as OpenPose [7] and DensePose [1]) bring similar opportunities to adding visual effects (pose stickers) to augment human body actions. However, due to the high degrees of freedom of a human pose, designing pose stickers would require a specialized tool.

There only exist few systems, such as *Octi* and Microsoft's Story Remix, which allow adding visual effects (texts and virtual objects) on or around a human body. However, such systems support only effects by tracking-based methods, which make virtual objects strictly follow the movement of the entire body or certain body parts, thus failing to capture the actual transformations of different objects. For example, to augment a subject's dribbling a basketball, simply letting the basketball track the subject's hand would fail to produce the bounce-and-back effect (see Figure 3(a)). Even though the movements of virtual objects are driven by human actions, they may have their own motion paths and speed variations. To achieve vivid effects in the interaction with human actions, the movements of virtual objects need to be precisely controlled. In this paper, we design a system that takes a step towards this emerging content creation scenario.

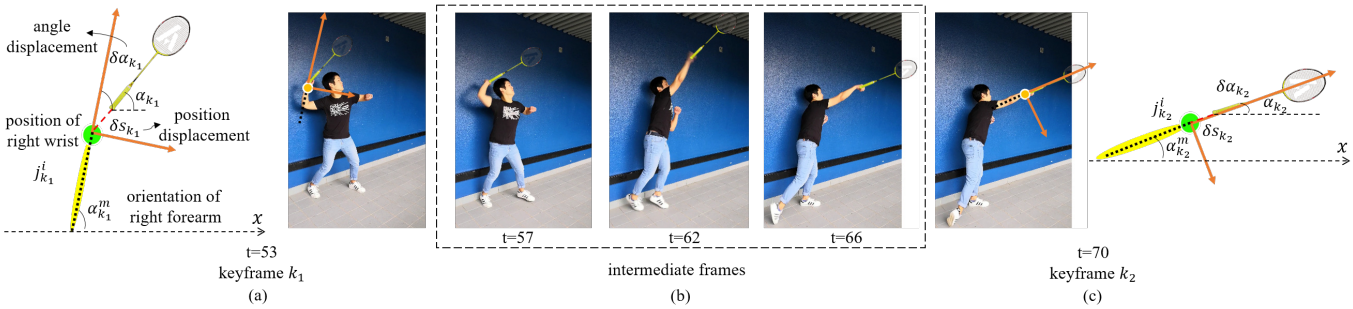


Figure 2. Motion tween of a virtual badminton racket driven by the right wrist and right forearm, which (more specifically, the position of the right wrist and the orientation of the right forearm) define a local frame. (a) and (c) are two keyframes. (b) are intermediate interpolated frames, where the positions and orientations of the racket are jointly determined by the transformed local frame, and the interpolated position displacement and angle displacement within the local frame.

Other frameworks have incorporated human movements to guide object transformations. For example, a subject's performance for manipulating physical puppets can be captured to transform corresponding 3D virtual puppets for making 3D animations [13]. Recent gesture-based frameworks [3, 26] first map a number of hand gestures or body gestures to the parameters of virtual objects (e.g. opacity, motion speed, etc.), which are then changed by gestures in real-time during live performance. Such methods would require gesture classifiers and thus are usually confined to limited types of gestures. In addition, these frameworks all require special hardware setups (e.g., Kinect) that are not necessarily often accessible to novice users.

We present *PoseTween*, an interactive system that allows novice users to easily add visual effects driven by the motion of a subject in an input video (Figure 1). The challenge of developing such a system is how to simplify the authoring process while keeping the expressiveness of our system. We propose to model the visual effects as pose-driven tween animations of virtual objects. In this way, we can utilize the flexibility of human poses to drive the movements of virtual objects, thus reducing the authoring efforts. It also allows us to use keyframing animation of virtual objects to preserve motion properties native to the objects, thus precisely controlling the movements of visual effects to achieve vivid results. While tween animation includes both motion tween and shape tween, in *PoseTween* human poses determine only the motion tween of virtual objects. Since the shape of objects often has non-trivial correlations with the motion of human poses, we let users author the desired shape tween effects when necessary. *PoseTween* can be used to add both 2D and 3D effects. However, since the majority of human actions is captured on videos by novice users, and 2D design assets are available in large volume, our current implementation focuses on 2D visual effects for simplicity.

During editing, a user only needs to specify the properties of a virtual object, such as positions and scales, with respect to the subject's movement at a few keyframes. The underlying body parts of the subject driving the object are automatically inferred according to the similarity between the spatial-temporal changes of the object's position and orientation and the changes in the subject's pose. In this way, one or more body parts (joints and limbs) determine a local frame for the virtual object, as illustrated in Figure 2. The tween animation of the object is then jointly determined by the intermediate positions and orientations of the local frame formed by the driving body parts as a coarse control, and the interpolation of the object's keyframe properties as a fine control. The coarse control via the local frame enables the motion path and speed of the object to roughly follow those of the body parts, while the fine control via displacements within the local frame facilitates the relative motion between the object and human pose, such that the interaction between the object and the human pose looks natural. For example, in Figure 2, the right wrist and the right forearm determine the origin and the orientation of the virtual racket's local frame respectively, while the displacements of the racket within the local frame enable the racket to be not strictly aligned with the right forearm and thus avoid a rigid-looking effect. We provide a user interface for editing the keyframe properties while associating objects with human actions.

Another challenge is to obtain accurate human poses from input videos. There exist deep learning-based human pose detection methods [7, 1] that support estimation of human poses from video frames and achieve state-of-the-art accuracy. However, even well pre-trained models may fail to accurately compute human poses in certain situations, e.g., when there exist occlusions and motion blurs. Even though there are automatic pose refinement methods that correct the initial human pose estimation results [10, 21], the refined human poses still contain errors that require further corrections. To address this problem, we provide a simple interface for modifying raw human pose estimation results. Upon correction of incorrect joint coordinates at certain video frames, the changes of joint positions are propagated to the neighboring frames accordingly. The experiment shows that this method is able to semi-automatically correct pose estimation errors in videos with only a small amount of user intervention.

PoseTween differs from existing tracking-based systems (e.g., *Octi* and [26]) in that it allows a more flexible and controllable movement of virtual objects. This enables more complex interactions between an object and a subject, and increases the expressiveness of body-driven visual effects (see a comparison in Figure 8). Besides adding visual effects to human action

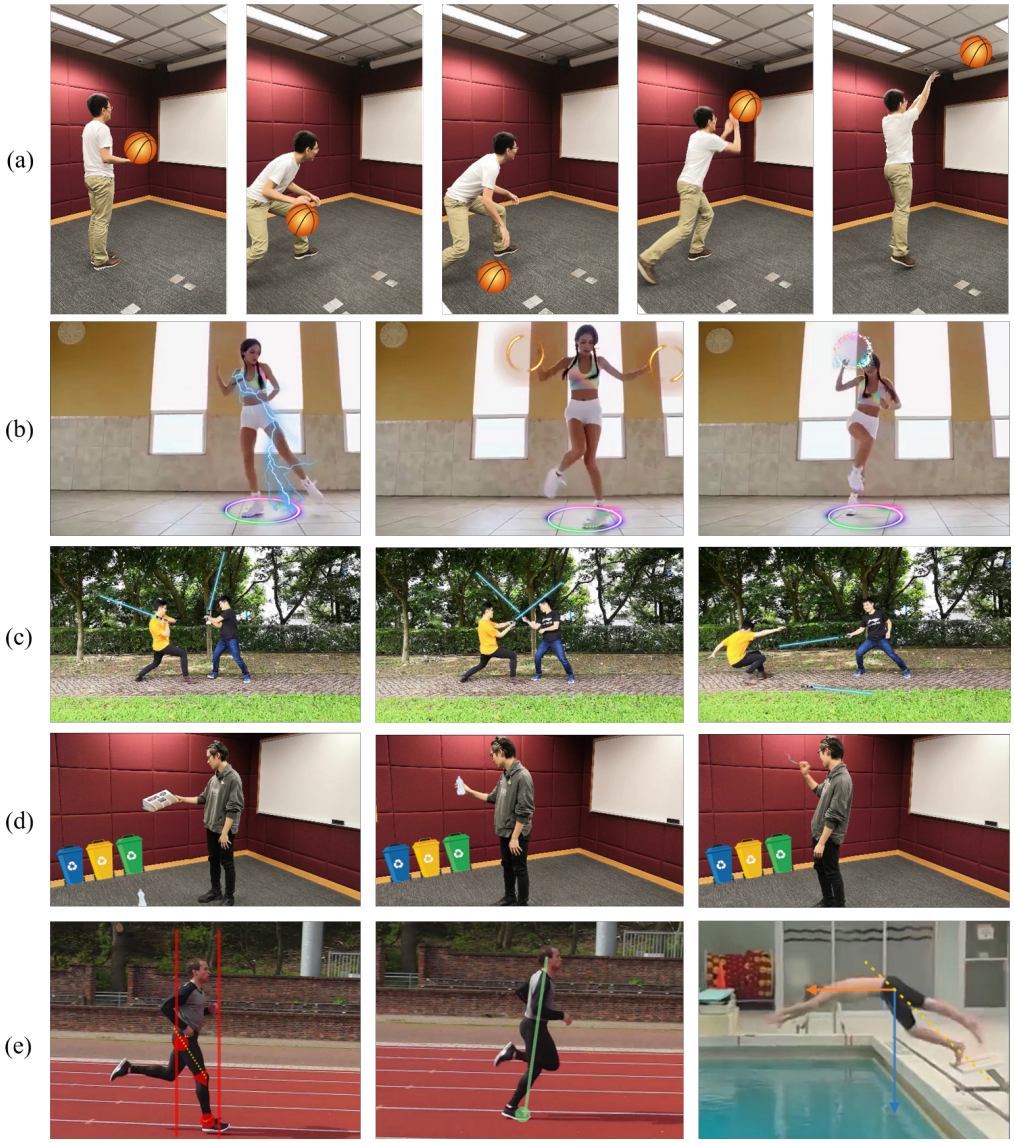


Figure 3. Examples of PoseTween applications: (a) "basketball": augmenting a subject's dribbling action by making the animation of a virtual basketball driven by the right wrist; (b) "dancing": adding visual effects to a dancing video; (c) "starwars": adding visual effects to subjects imitating a movie clip. PoseTween generalizes to videos containing multiple subjects by inferring associations among the body parts of all the subjects; (d) "recycle": making educational videos; (e) "sports": augmenting sport tutorial videos with auxiliary markers to visualize athletes' performance.

videos, *PoseTween* can be creatively used for other purposes, e.g., making education animations and tutorial videos (see Figure 3). Our contributions are summarized as follows: (a) a new concept of pose-driven tween animations; (b) the first interactive system that allows novice users to easily author the movements of virtual objects interacting with human actions by making pose-driven tween animations; (c) an interface for modifying incorrect results produced by automatic human pose estimation methods.

RELATED WORK

Automatic Tweening. Automatic tweening is a basic function in many animation software, such as Adobe Flash and Adobe Animate, Synfig, Toon Boom, CACANi, etc. Manually editing the motion tween for complex motion in these existing tools would require skillful users. Even though such software provides preset tweening methods, they often only cover common interpolation algorithms, such as linear, ease in/out, clamped, constant, etc. For novice users, it is tedious to create tween animations whose trajectories are well aligned with a subject's movement with such preset tweening methods.

The interpolation algorithms for tween animations have also been extensively studied. For example, the intermediate positions of an object or a vertex can be determined by a high-order polynomial curve [6] or a spiral curve [35] to achieve a more complex motion path. Another group of algorithms has been proposed to interpolate stroke or object parameters instead of interpolating positions directly to reduce distortions [27, 11]. However, applying these interpolation algorithms might still

result in unnatural animations when the desired motion path and morphing of objects are non-analytical, or when their parts move differently. Even though some works provide interactive tools that allow users to modify tween results by their proposed interpolation algorithms [37, 35], the editing processes might be laborious to novice users.

Human-body Guided Animation. Several existing works introduce human factors in making animations. One group of methods allows users to map a few hand gestures [18] or body gestures [3, 26, 28] to the parameters of target animated objects in an authoring process, and the parameters can then be changed in real time by the authored gestures in a performance process. However, such gesture-based methods require a gesture recognition model, which prevents them from incorporating new user-customized gestures easily. Since a user can only perform a limited number of gestures at one time, the parameters that can be changed simultaneously are thus also limited. In addition, even if the gestures are well-categorized in terms of expressiveness [26], they do not necessarily have semantic correlations with the parameters of objects to be animated and thus users need to memorize the customized mappings for live performance.

Another group of methods builds alignments between a human skeleton and a virtual object, and transfer the body movement to the object directly. For example, [15] uses hand-drawn figures to drive 3D proxies, and [8, 16] use the movement of a human body to deform a 3D mesh. Such methods mainly deal with morphing of virtual elements, i.e., shape tween, while our work focuses on *motion* tween animations of virtual objects, which do not need to have similar structures with a human body for alignment. By contrast, our system is especially suitable for the situations when virtual objects have their own characteristic motions, while being related to human actions, which can thus be used as movement references for virtual objects (e.g., dribbling a basketball, as shown in Figure 3(a)).

Animation Using Videos. There exist methods that generate animations of a static object using motion cues extracted from videos containing a similar object. For example, Bregler et al. [4] retarget the motions from a traditionally animated cartoon to 3D models, 2D drawings and photographs. In *LiveSketch* [31], the motion at a group of control points is first extracted from a video and then transferred to a static sketch via the corresponding control points. In *RealitySketch* [32], the motions of objects are tracked to interactively animate augmented sketches in videos. Park et al. [24] and Willett et al. [36] use reference motions in human action videos to guide the deformation of virtual characters. These works make use of the similarity in structure between a source object in a reference video and target objects to be animated. In contrast, since our method uses the human motion in a video to drive the animation of an object rather than directly transferring the animation to the object, it only requires the object's movement to be related to part of the human motion.

Human Pose Estimation. Human pose estimation has been studied extensively for applications such as gaming, human-computer interaction, health care, etc. Depth sensors like Microsoft Kinect enable easy detection of human poses in nearly real-time, but the detection is limited by physical limits like the angle of vision [19]. Motion capture (MoCap) [25] often uses multiple cameras or multiple types of sensors such as accelerometers and gyroscopes to obtain more accurate pose data. Their practical usage, however, is limited by the availability of the hardware devices in daily life. Pose estimation from images or videos acquired by monocular cameras has also been extensively studied. State-of-the-art human pose estimation methods can be generally classified into two categories: top-down approaches, such as G-MRI [23] and RMPE [9], and bottom-up approaches, such as OpenPose [7], DeeperCut [14] and other pictorial structure methods [2]. Automatic human pose correction methods [29, 33, 10, 21] have been proposed to refine initial human pose estimation results as a post-processing step. These methods use neural networks to capture the joint distribution of images and errors in human poses, and directly output refined poses. However, the refinement results are hardly perfect and thus need further corrections. Besides a few human pose annotation tools for building datasets [34], to the best of our knowledge, there is no existing tool that allows users to interactively modify human poses computed by pose estimation algorithms.

USER INTERFACE

Figure 4 shows our main user interface. The input to our system is a human action video and design assets of the virtual objects to be animated. Our system has three modes: keyframe editing (Figure 4), pose editing (Figure 5), and preview (Figure 6).

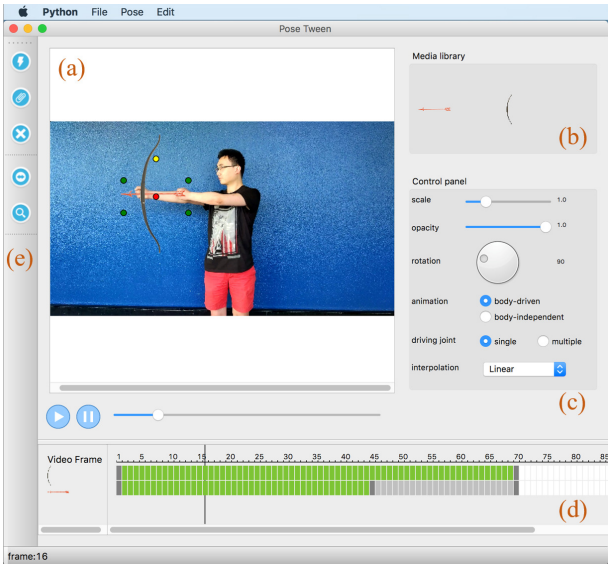


Figure 4. The user interface in the keyframe editing mode. (a) main canvas with a human action video as background; (b) pre-loaded virtual objects in media library; (c) a control panel for editing keyframe and animation properties; (d) timelines for virtual objects, where dark gray blocks represent keyframes. Green, blue (not shown here), light gray blocks represent single-joint driven, multiple-joint driven, and body independent frames, respectively; (e) toolbox buttons, mainly used for switching between different modes.

Pose Editing Mode

We adopt lightweight OpenPose [22] in our implementation for computing human poses from videos. There are two types of error, namely, incorrect and missing, for joint coordinates by the pre-trained OpenPose model. We fill the missing joints caused by occlusions automatically using action coherence (see Section "Human Pose Processing"). Our system provides a tool for users to semi-automatically modify incorrect human pose estimation results, as well as fine-tune filled missing joints. This modification is typically done as a pre-processing step prior to editing keyframes for motion tweening, but a user may switch to this mode from the other two modes whenever necessary. In this mode, human joint coordinates are visualized by draggable circle joint handles. For joints with incorrect locations (e.g., left wrist and left elbow in Figure 5(a)), or missing joints that cannot be filled automatically (e.g., left eye and left ear at top-left corner of Figure 5(a)), the user may explicitly specify the correct joint locations by dragging the joint handles. Due to the coherence of human actions, incorrect joint coordinates often appear in consecutive frames. Instead of letting users manually modify the human pose in each frame, the changes made in one frame are propagated to neighboring frames using both the user input and the joint coordinate coherence constraints (see Section "Human Pose Processing" for more details).

(a) (b) (c) (d)

Figure 5. The user interface in the pose editing mode. (a) failed human pose detection with missing (left eye and left ear, shown at the top-left corner) or incorrect (left wrist and left elbow) joint locations; (b) a state-of-the-art automatic refinement method PoseFix [21] failed to correct the pose and even introduced new errors that need further corrections; (c) a user may specify the locations of missing joints (the handles for left eye and left ear are edited in this case) or modify incorrect joint locations by directly dragging joint handles; (d) the modified human pose using our UI.

Keyframe Editing Mode

In this mode, users manually edit the properties deciding the appearances and motions of the virtual objects at keyframes, with their timings dependent on the subject's action in the video. The editing process follows the conventional timeline-based operations (as in most animation software), which are familiar to experienced animators and can be easily learned by novices. Please refer to the supplementary video for a sample creation process. Below we introduce three key elements in this mode: objects, keyframes, and associations.

Objects. A virtual object is editable once it is dragged from the media library (Figure 4(b)) onto the canvas (Figure 4(a)). The user-editable object properties include transformation center, position, orientation, scale and opacity. Table 1 lists how our interface allows users to edit each property. Other non-editable object properties include position displacement and angle displacement, which are computed automatically after association inference (see "*Associations*" later in this section). Among all the object properties, the position and the orientation are driven by human poses because they are physically related, while others are interpolated by user-specified interpolation algorithms (Table 2).

Property	Description
transformation center	The default transformation center of an object is its geometric center. The user can change it by dragging the center handle (the red circle in Figure 4(a)).
position	The position of an object is defined as the global position of its transformation center in a video frame. It is changed by directly dragging the object to a target position.
orientation and scale	The orientation of an object is defined as the orientation of its principle axis. The scale is the ratio of the user-specified size to the original size. To resize and rotate an object, the user may either intuitively manipulate the control points on the object, or precisely change the exact values in the control panel.
opacity	The opacity that moderates an object's visibility ranges from 0 to 1, where 0 represents invisible.

Table 1. Editable properties of virtual objects supported by our interface.

Property	Description
body dependency	The user may select "body independent" to disable the association of an object with body parts between the current keyframe and the previous keyframe.
single or double driving joints	By default each object is driven by a joint and a limb. When an object is to be aligned with two joints that do not belong to a common limb the user may change this option to enable the association of an object with two joints.
if-disabled	The user may disable the animation of objects between the current keyframe and the previous keyframe.
interpolation method	The default interpolation method is linear. Other options provided include ease in, ease out, logarithm, and auto-Bezier.

Table 2. Properties of animation controls supported by our system.

Keyframes. Each object instance has its corresponding track on the timeline, as shown in Figure 4(d). For temporal alignment, a user adds keyframes for each object at proper timings in the subject's action. A keyframe is added to the timeline

either automatically when one of the object properties at a particular frame is changed, or manually by pressing a toolbox button. A shortcut menu on the timeline enables deleting or changing the timings of keyframes. Table 2 shows the animation properties to be specified at keyframes, namely, body dependency, single/multiple driving joints, if-disabled, and interpolation method.

Associations. The association inference that finds the driving body parts of an object is conducted automatically when users edit the keyframes of an object (details in Section "Inference of Associations"). An alternative approach is to let users directly indicate object-to-body associations, as in [26]. Such direct graphics-to-body associations might work well when an object is associated to a fixed body part during animation. However, when creating animations driven by different body parts during different periods (e.g., frequent changing from one hand to the other hand in juggling balls in Figure 1, changing from hands to feet in the soccer example in Figure 10(d)), automatic inference of associations can reduce the amount of user intervention. It also has more potential than the manual specification of object-to-body associations in generalizing to mobile app creation scenarios, where the technical details (e.g., the human poses, keyframe animations) might be hidden from end users. For example, when authoring the movement of the basketball in Figure 3(a) in an entertainment mobile app, an end user may directly specify desired positions of the basketball at certain action keyframes without having to understand the principle that it is associated to and driven by the wrist. Thus we propose to automatically infer associations as initial guesses, and let users explicitly specify body associations only when the inference has errors. After association, the object's non-editable keyframe properties (position displacement and angle displacement) are computed with respect to the local frame formed by the associated body parts. Specifically, the position displacement is the difference between the object's position and the local frame origin, while the angle displacement is the orientation difference between the object and the local frame (see Figure 2).

Preview Mode

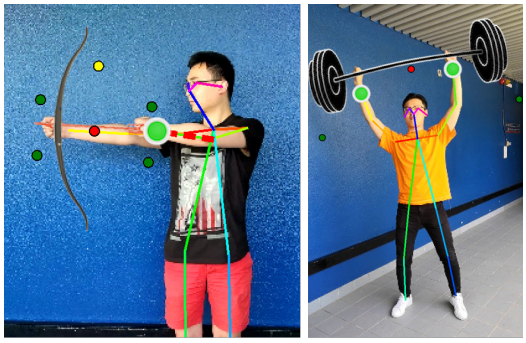
In the preview mode, a user may preview the details of an object's joint and limb association. We provide two types of handles to visualize the association: position handle (a circle in light green in Figure 6) representing a driving joint, and orientation handle (a red dotted line in Figure 6(a)) representing a driving limb. Users may change the driving joint or limb by directly dragging the corresponding handles, which are snapped to the nearest newly specified joint or limb.

METHODOLOGY

In this section, we introduce the core algorithms behind our UI, including the propagation of joint coordinate changes to neighboring frames, inference of driving body parts of an object given the video and user-specified object properties, and finally the pose-driven tweening method.

Human Pose Processing

We first describe our method for correcting missing and incorrect joint coordinates. One possible solution for filling missing joints is to fit a spline curve to the non-missing joints [30]. However the actual joint trajectories might not follow the pre-defined type of spline curve. Enlightened by the recent success of autoregressive (AR) model in long-term 3D human prediction [38], we model the joint coordinate sequences by autoregressive models to make full use of the prior knowledge of the joint sequences. The main idea is to first use raw joint coordinates to build initial AR models for automatic filling of missing joints [20], and then fix possible errors by propagating manual corrections at one frame and the updated AR model parameters to neighboring frames.



(a) (b)

Figure 6. The user interface in the preview mode. (a) the arrow is driven by a joint and a limb, visualized by a position handle in light green and an orientation handle as a dotted red line segment, respectively; (b) the barbell is driven by two joints, namely, the left and right wrists, visualized by two position handles.

OpenPose returns the (x,y) coordinates of 18 joints. For simplicity, we denote a joint's coordinate sequence as $X_1, X_2, ..., X_N$ for an N-frame video. The forward and backward AR models of a joint position sequence are represented by $X_t = \sum_{w=1}^{W} f_w X_{t-w},\ t \in [W+1, N]$ and $X_t = \sum_{w=1}^{W} b_w X_{t+w},\ t \in [1, N-W]$, respectively, where f_w and b_w are the respective parameters of forward and backward autoregressive models of order W ($W = 5$ in our experiment). Given the raw human poses by OpenPose, we initialize a pair of forward and backward AR models for each joint, where the parameters f_w and b_w are estimated using non-missing joint coordinates by a least-squares method. The missing joint coordinates are then automatically filled using these initial models [20], as shown in Figure 7(a). When the non-missing joints are scarce to fit the initial models, users can still manually specify the missing joint positions by joint handles.

To reduce the amount of user intervention and to ensure the coherence of joint position sequences, we propagate user-specified corrections at a certain frame in a temporal window of size $2 \times d + 1$ ($d = 2$ in our experiment). See Figure 7(b) for an illustrated example. Specifically, we model the propagation process as a quadratic programming problem, where the user-specified input is used as a hard constraint in updating the neighboring joint locations, while satisfying the autoregression model constraints to ensure coherence. Suppose the user modifies the joint position X_m into $\hat{X}_m$ at time $t = m$. The forward AR model for the modified joint is first updated using the sequence $X_{m-d-W}, X_{m-d-W+1}, ..., X_{m-d-1}$. Let f'_w denote

the updated parameters. The objective is to minimize the difference between the updated joint positions in the neighboring frames of X_m and the joint positions predicted by the updated forward AR model:

$$\begin{aligned} \min \quad & \sum_{u=-d}^{d} \|X_{m+u} - \sum_{v=1}^{W} f_v' X_{m+u-v}\|^2, \\ \text{s.t.} \quad & \hat{X}_m = X_m, \ \hat{X}_m = \sum_{v=1}^{W} f_v' X_{m-v}. \end{aligned} \tag{1}$$

For the first W joints of the sequences, since the forward model is not defined, the backward AR model will be used as the constraint in Equation (1).

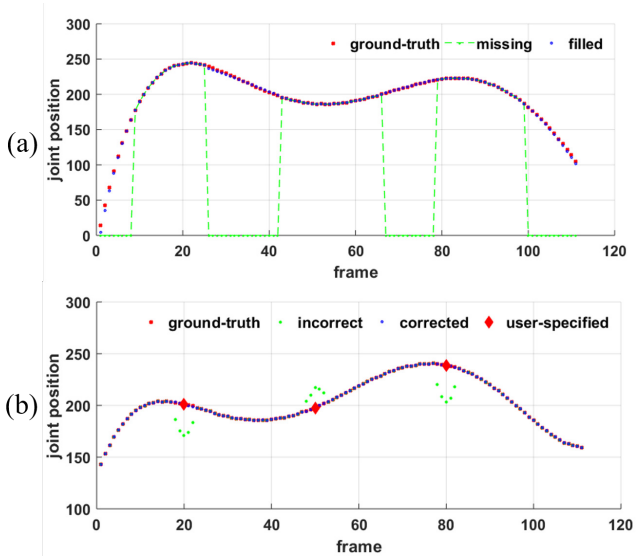


Figure 7. A demonstration of our missing joint filling and pose correction method on simulated random joint trajectories. (a) Filling missing values using the autoregressive model estimated from non-missing values. For missing joints in the midst of the sequence, the predictions by the forward and backward AR models are averaged to fill the missing joints. Only the backward (forward) model prediction is used for predicting missing joints at the beginning (end) where the forward (backward) model is not defined; (b) when the user manually specifies a joint position at one frame (as shown by the red dots) to fix incorrect joint positions (the green dots), the neighboring frames are updated accordingly by a quadratic programming.

Inference of Associations

In this subsection, we present a method to infer the association of an object to body parts given their positions and orientations in two consecutive keyframes. The output is indexes of an optimal joint and limb, or of a joint pair that drive the movement of the object.

By analyzing videos of common human actions that drive movements of objects (e.g., daily living activities, sports, etc.), we make the following observations for designing the inference method:

- O_1: The translation and rotation of an object might be driven independently. For example, in the juggling example in Figure 1, the balls constantly change their positions but their orientations hardly change. Therefore, if an object is driven by a joint and a limb, we separately infer the joint that decides the origin of the object's local frame, and the limb that determines the orientation of the local frame;

- O_2: The deciding orientation of an object to be aligned with body parts (limb and/or joint(s)) is often along the principle axis of the object. For example, the principle axis of a badminton racket is to be aligned with the forearm (see Figure 2). Thus the orientation of the principal axis of an object is used for limb inference;

- O_3: Objects are typically aligned with a joint and a limb. But some objects are to be aligned with two joints that do not belong to a common limb. For example, the orientation of a golf pole is largely determined by a wrist and a neck joint (see Figure 1). In such cases, since an object rotates with respect to the orientation angle of the vector formed by the two joints, we infer the two joints from such orientation angles.

Denote the 18 2D joint coordinates for each frame computed by OpenPose as j_t^i, $t = 1,2,...,N$, $i = 1,2,...18$. At each frame, the 18 joints form 19 semantic limbs with predefined positive directions, denoted as l_t^m, $t = 1,2,...,N$, $m = 1,2,...19$, and their correspond orientations α_t^m, which are the angular displacements of limb positive directions from the positive x-axis direction in the pixel coordinates of video frames. Suppose two consecutive keyframes are at time k_1 and k_2 with $k_1 \leq k_2$. Further denote the positions of the object at the two keyframes as s_{k_1} and s_{k_2}, and the orientations of the object α_{k_1} and α_{k_2}, respectively. Please see the notations in Figure 2 for an illustration. Since the numbers of joints and limbs are small and fixed, the optimal joint or limb can be inferred by matching the user-specified object positions and orientations with those of the body parts.

We first describe association inference for the case of an object being driven by a joint and a limb. The joint deciding the origin of the object's local frame is inferred from both the spatial distances between the object and all the joints at k_1 and k_2, and the motions represented by their position changes between k_1 and k_2. Specifically, the driving joint is inferred by:

$$\begin{aligned} i^* = \arg\min_i \ & \|j_{k_1}^i - s_{k_1}\|_2 + \|j_{k_2}^i - s_{k_2}\|_2 \\ & + \|(j_{k_2}^i - j_{k_1}^i) - (s_{k_2} - s_{k_1})\|_2. \end{aligned} \tag{2}$$

Similarly, we infer the limb deciding the orientation of the object's local frame from the similarity in both the orientation of the object and all the limbs at k_1 and k_2, and their orientation changes between k_1 and k_2:

$$\begin{aligned} m^* = \arg\min_m \ & \left|\alpha_{k_1}^m - \alpha_{k_1}\right| + \left|\alpha_{k_2}^m - \alpha_{k_2}\right| \\ & + \left|(\alpha_{k_2}^m - \alpha_{k_1}^m) - (\alpha_{k_2} - \alpha_{k_1})\right|. \end{aligned} \tag{3}$$

After inference of association, we compute the object's position displacement and angle displacement in the local frames

at the two keyframes. The displacements are given by

$$\begin{aligned} \delta s_{k_p} &= s_{k_p} - j_{k_p}^{i^*}, \\ \delta \alpha_{k_p} &= \alpha_{k_p} - \alpha_{k_p}^{m^*}, \quad p = 1,2. \end{aligned} \tag{4}$$

The joint pair (i_1^*, i_2^*) for virtual objects driven by two joints are inferred similarly by matching the object orientation with the orientation angles of vectors of all joint pairs. Then we set the origin of the object's local frame as the midpoint of the two joints.

The above inference method might fail when the object has a large displacement with the target driving joint, or when multiple pairs of joints have similar orientations (see the discussions on failure cases in Section "Evaluations"). The user may then interactively use the position handle and orientation handle to overwrite the inference results.

Interpolation

The position and orientation of an object in an intermediate frame are computed as a combination of the intermediate position and orientation of the local frame, and the interpolated position and angle displacements in the local frame. Specifically, at time $t = \tau$ between keyframes k_1 and k_2, the position and orientation of the object driven by a joint and a limb are given by

$$\begin{aligned} s_\tau &= j_\tau^{i^*} + F(\delta s_{k_1}, \delta s_{k_2}), \\ \alpha_\tau &= \alpha_\tau^{m^*} + F(\delta \alpha_{k_1}, \delta \alpha_{k_2}). \end{aligned} \tag{5}$$

Similarly, the position and orientation of the object driven by two joints are computed as follows:

$$\begin{aligned} s_\tau &= (j_\tau^{i_1^*} + j_\tau^{i_2^*})/2 + F(\delta s_{k_1}, \delta s_{k_2}), \\ \alpha_\tau &= \alpha_\tau^{i_1^*, i_2^*} + F(\delta \alpha_{k_1}, \delta \alpha_{k_2}), \end{aligned} \tag{6}$$

where $F(\bullet)$ is one of the user-specified preset interpolation algorithms in Table 2.

EVALUATIONS

In this section, we show the results of a user study evaluating the effectiveness of *PoseTween*, and the results of quantitative experiments evaluating the accuracies of the human pose processing and the association inference methods.

User Evaluations

We conducted a user study to evaluate the improvement of our system in using human poses over the traditional tween animation without using human poses, with respect to simplifying the authoring process and improving the quality of adding visual effects to human action videos for novice users.

Apparatus. Our user interface was developed with Python 3.6 on a laptop (Intel i5 @2.7GHz, 8GB RAM) running macOS. The lightweight OpenPose ran at 9∼15 fps on this PC. We captured action videos by a mobile phone camera with 1080p resolution at 30 fps.

Participants. 12 university students (a1∼a12, aged 24∼30, 3 female) were invited to participate in the user study. Among them, 9 had no animation background while 3 had intermediate experience in making animations (on a 1-10 point rating, a6 rated 7 with Blender and 3ds MAX, a8 rated 6 with Toon Boom, a10 rated 4 with Adobe Flash). Each participant was asked to perform exploitation and exploration tasks, and to complete a questionnaire at the end of the study.

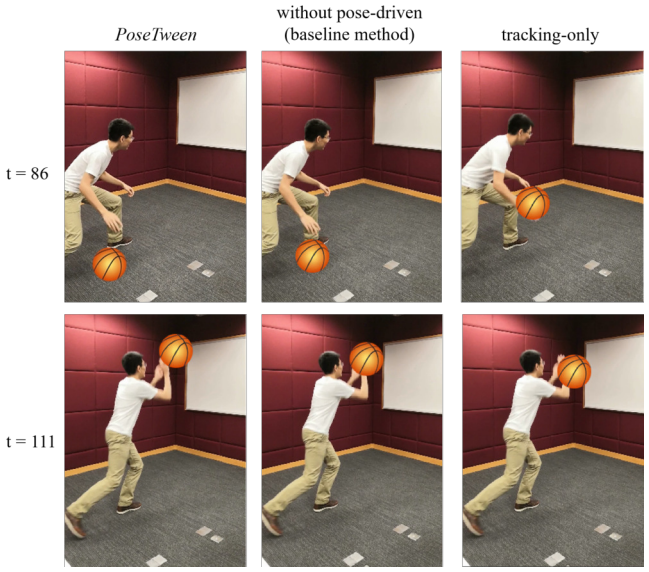


Figure 8. A comparison of PoseTween, baseline method (using traditional tween animation), and tracking-only method (i.e., simply letting the basketball track the wrist) on selected intermediate frames.

Exploitation Task

The goal of the exploitation task was to evaluate the ease of use and the satisfactory of outcome of our system by a set of fixed tasks. For comparison, we prepared a baseline system by removing the pose-driven functions from our user interface such that the baseline system supports making tween animation of objects only by interpolation algorithms, similar to tweening tools in existing animation software. We proposed the following hypotheses:

- ***H1.*** Using *PoseTween* to make tween animations of objects with respect to human actions simplifies the authoring process. Specifically, *PoseTween* requires less creation time (*H1a*) and fewer keyframes (*H1b*) than the baseline system;

- ***H2.*** *PoseTween* yields animations of objects that appear more naturally interacting with human actions than the baseline system.

During training, we first gave a detailed tutorial on the operations of our system as well as the baseline system. The participants then tried freely to get familiar with the two systems. After that, each participant was asked to reproduce three target augmented videos given human action videos and object design assets to animate using both our system and the baseline system (i.e., 2 methods × 3 animations). The order of the methods was counterbalanced, and the order of the tasks was randomized to reduce the learning effects.

The three videos chosen were: (1) "badminton", animating a racket driven by a wrist and a forearm (a joint and a limb), which is a medium complex motion involving translation and rotation; (2)"weight lifting", animating a barbell driven by two wrists (two joints) to test animating using the multi-joint functionality; (3)"arrow", animating an arrow and a bow (two objects) driven by two wrists to test animating multiple objects.

The creation time and the number of keyframes needed for making each animation are shown in Figure 9. Paired t-tests between *PoseTween* and the baseline system on the creation time and the number of keyframes for each animation show that, except for the creation time for "weight lifting" ($p = 0.5501$, *H1a* rejected), *PoseTween* required significantly less creation time and fewer keyframes than the baseline system ($p < 0.05$, *H1a* and *H1b* supported). The movement of the barbell in "weight lifting" was relatively simple though it involved two joints. Since there was no complex object transformation, in the baseline system the participants only needed to add keyframes to adapt the barbell's speed to that of the two wrists. The most significant differences in the creation time and the number of keyframes occurred in the "badminton" action ($p = 0.0105$ and $p = 1.87 \times 10^{-5}$, respectively), in which the complex motion path and speed variation of the racket required the participants to add more keyframes in the baseline system to achieve such a movement. In contrast, with *PoseTween*, the body movements of the subject provided more motion guidance to the racket. This result indicates that our system has clear advantages especially when the motion of an object is complex. This is confirmed by a8, who had used Toon Boom, commenting on our system: "*It can easily handle some complex movements (e.g., rotating and translating simultaneously), which are difficult with common interpolation algorithms.*"

For the creation time we also record the respective times needed for keyframe editing, previewing videos and animation results, and modifying object animations and association errors. There is no significant difference in previewing and modifying times between *PoseTween* and the the baseline system, while *PoseTween* requires significantly less keyframe editing time in all of the three actions ($p = 0.0005$ for "badminton", $p = 0.026$ for "weight-lifting" and $p = 0.0003$ for "arrow"). The keyframe editing time in *PoseTween* accounts for 35.2% of the total creation time on average, compared with 60.5% in the baseline system. Even though the users needed to spend a certain amount of time previewing the videos and modifying results during creation, *PoseTween* effectively reduced the keyframe editing time.

To verify the improvement in the appearance of visual effects using *PoseTween* compared with the baseline system, we let the participants rate the level of satisfaction towards the videos augmented with visual effects made by *PoseTween* and the baseline system separately (1 is the least satisfied and 10 is the most satisfied). The satisfaction level of augmented videos made by *PoseTween* (*Mean=9.3, SD=0.78*) was significantly higher than those made with the baseline system (*Mean=6.0, SD=1.94*) (p<0.001), supporting *H2*.

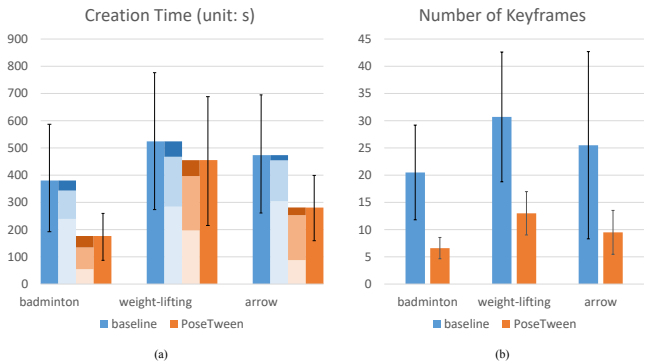


Figure 9. Comparisons of average creation time and average number of keyframes required for the three actions using *PoseTween* and the baseline method. The stacked bars in (a) show from bottom to top the average times for keyframe editing, previewing and modifying, respectively.

The participants were asked to rate *PoseTween* and the baseline system in terms of the ease of use in the questionnaire, including a standard System Usability Scale (SUS) [5] for each of the two systems. The SUS score of *PoseTween* was 80.2 compared with 69.5 of the baseline system. Besides the SUS, the participants were also explicitly asked in the questionnaire whether they found *PoseTween* has simplified the editing process and achieved more natural-looking video results by a 10-point rating (1 is strongly disagree and 10 is strongly agree), resulting in the average scores of 8.5 (*SD=0.81*) and 8.7 (*SD=1.1*), respectively. Participant a6, who had experience using Blender and 3ds MAX, verbally commented that he preferred to use *PoseTween* to edit object animations related to human actions, and that he was able to easily adapt his previous skills to using *PoseTween*.

Exploration Task

The goal of this task was to test the expressiveness of our system. In this task, each participant was asked to think of a favourite scenario that involves one or more objects driven by a human action and to produce it using *PoseTween*. The participants then performed the actions, which were video-recorded, and prepared the design assets of objects to be animated. Even though our system does not support live preview, the participants' performances were not influenced by the absence of actual objects, since they were aware that they could adapt the objects to their actions in the editing process. Figure 10 shows several representative animations created by the participants. Most participants came up with a sport action, such as playing soccer and golf, or an action of a subject manipulating a tool, such as a fork and an axe. In the creation process, 8 participants made use of both the body-independent function and the body-driven function, making the objects alternatively being driven by a body part and move independently. In the action "playing shuttlecock" (Figure 10(e)), the shuttlecock moves independently of body parts in most of the frames, in which the participant edited the movement of the shuttlecock with respect to the timings in the subject's movement. In this case, the object is hardly driven by the subject's movement, and thus the creation process with *PoseTween* would approach that with the baseline system. Even though *PoseTween* mainly

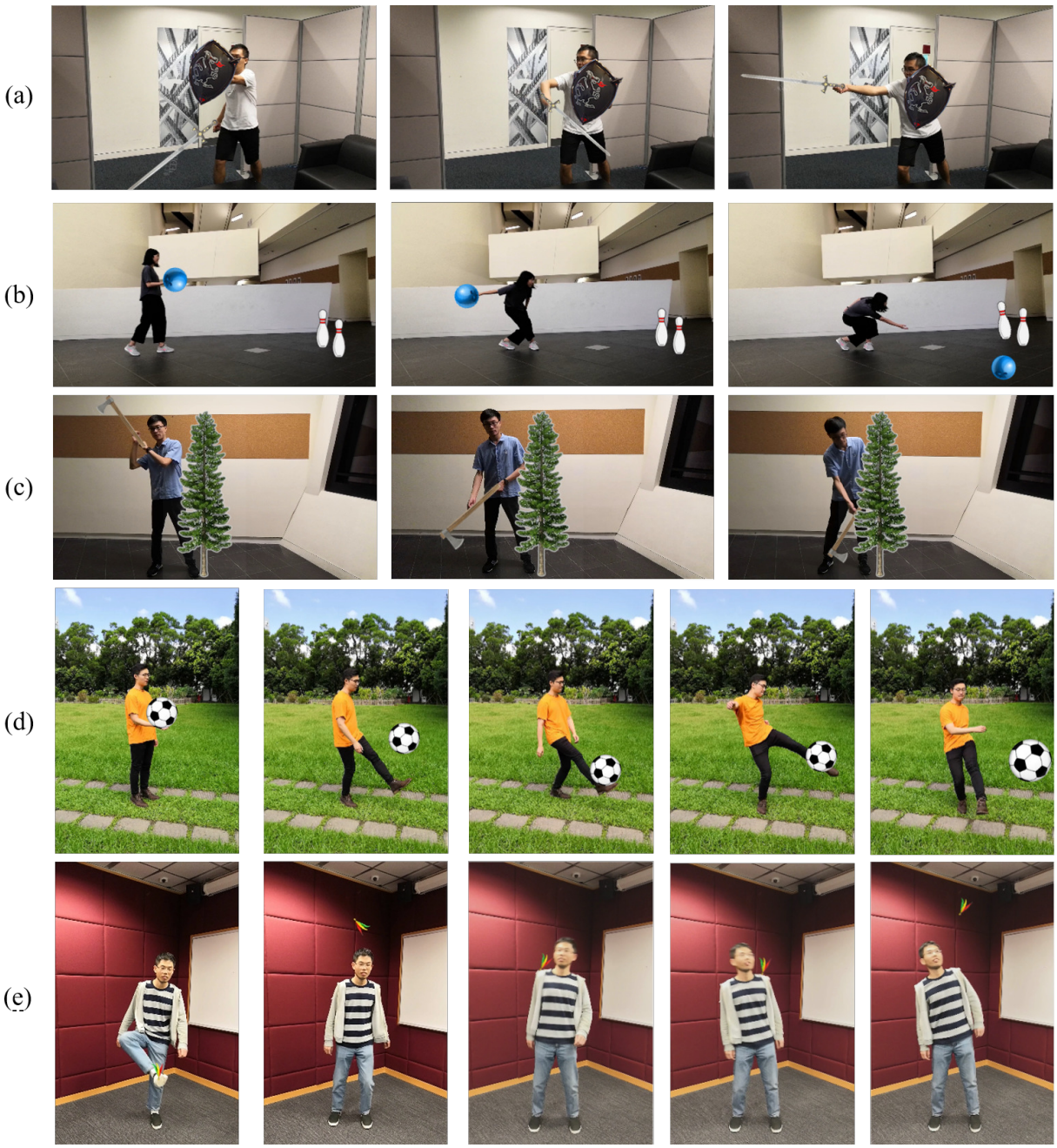


Figure 10. Video augmentation results from the exploration task. (a) "knight": 25 keyframes, 9min 31s; (b) "bowling": 13 keyframes, 10min 51s; (c) "cutting tree": 14 keyframes, 14min 29s; (d) "soccer": 13 keyframes, 2min 07s; (e) "playing shuttlecock": 23 keyframes, 5min 49s.

targets at animating objects interacting with human actions, five of the creation scenarios involve interactions between virtual objects (e.g. bowling ball-versus-pins and axe-versus-tree in Figure 10(b)(c)).

User Feedback

In the study, all the participants were able to learn to use our system quickly. During training, all of them took less than 8 minutes to get familiar with the operations in both systems. The participants commented favorably about our system. Even the participants with little animation experience found *PoseTween* to be a useful tool for adding visually appealing effects without having to learn a lot of background knowledge on making animations.

a3: "*The tool is easy and intuitive to use. I do not need to fine-tune keyframes frequently.*"

a5: "*... It reduces manual operations, and makes the results more realistic especially for more complex motions and longer videos.*"

a2: "*A very nice tool, easy to operate the objects to make them interact with the human body, to make funny and meaningful videos. I would prefer to use it in an instant message chat in the future.*"

a11: "*The body-driven function provides very natural effects when an object has a close interaction with the human, especially when the motion is complex or the interaction between the human and object is not just mounted to a static point.*"

a1: "*... I'm looking forward to using the system in the future.*"

A participant also proposed suggestions and possible improvements:

a3: "*In daily life scenarios, objects are often driven by hands, thus providing templates of pre-set hand associations and build-in visual effects for common object categories may be more convenient for most common users.*"

Human Pose Processing

To quantitatively evaluate the effectiveness of our pose editing tool in its accuracy and ability to reduce the amount of user intervention, we collected the statistics of correcting errors in human poses estimated by lightweight OpenPose. Ideally this evaluation would have been conducted by the invited participants. But since some joints remotely related to the virtual objects (e.g., ears, nose) might not require corrections for the animation creation processes, and both the joint correction process and the measurements in the experiment are objective, we, the designer of the experiment, modified the human poses instead of user study participants for a thorough evaluation of our tool. The videos selected for the experiment included both simple ("arrow", "weight lifting" and "golf") and complex ("badminton" and "boxing") actions, covering a large variety of joint trajectories. For each video we counted the number of errors by OpenPose while manually correcting each error as ground-truth joint positions for the later experiments.

In the first experiment we evaluate the ability of our tool in reducing the amount of user intervention. We corrected the human poses in the same selected videos using our tool, and recorded the numbers of corrections by joint handles, and the time needed, as shown in Table 3. Since the manual correction process also included fine-tuning the automatically filled results, there might be more manual corrections than incorrect joints by OpenPose, as in the action "boxing". In "boxing", since the wrist joints have large speed and position variations, the initialized AR models for the wrists failed in predicting the precise occluded wrist positions. For all of the five videos, the numbers of manual corrections using our tool were far less than the total numbers of missing and incorrect joints, and the correction times were far less than manual corrections. These indicate that our tool is able to semi-automatically correct the joint coordinates produced by pose estimation methods with a small amount of user intervention.

In the second experiment we evaluate the accuracy of our human pose correction method. We compared the joint position errors with respect to the ground-truth positions among the raw pose positions by lightweight OpenPose, the Catmull-Rom interpolation method [30], a state-of-the-art automatic pose refinement method PoseFix [21], and our semi-automatic method without and with manual collection, as shown in Table 4. In implementing the Catmull-Rom interpolation method for comparison, we filled the missing joints by interpolating all the non-missing joints. This interpolation method generally performed well but failed to recover the complex wrist trajectories in "boxing". Since PoseFix does not involve an error detection process, its refinement results sometimes mistakenly overwrote correct joint positions, and contained errors that needed further corrections (see Figure 5(b)). In contrast, our tool achieved a reasonable accuracy by correcting only a portion of incorrect joints.

Association Inference

To evaluate the effectiveness of our association inference method, in the exploitation task we recorded the numbers of false joint and limb associations, as shown in Table 5. These false associations appeared in a few common situations. In the "weight-lifting" action, the keyframe with the subject having a pose in Figure 6(b) caused the association errors, where the barbell was associated with two shoulders instead of two wrists. This is largely because the shoulders have similar orientation angles to those of the two wrists. However, since these two pairs of joints had similar orientation angles, the false associations did not noticeably influence the animation results even when the participants did not correct them. The "arrow" action had the least association errors, mostly because this action did not involve large rotations, and between every two keyframes of the arrow only the driving body parts had obvious movements while others remained stationary. In the failure cases, a participant let the subject hold the front end of the arrow, where the transformation center of the arrow was closer to the left elbow than the left wrist, resulting in the arrow driven by the left elbow and rotated with the left shoulder. Since the position displacement between the arrow and the left elbow was nearly constant in the translation of the arrow, false associations also did not cause noticeable artifacts in this action. For the "badminton" action, the inference of association failed when the user-specified orientation of the racket differed much from the orientation of the right forearm. Since the orientation of the racket was largely determined by the orientation of the forearm, false associations would cause obvious artifacts in the animation results. In the user study, all the participants were able to identify the false joint or limb associations and correct the associations accordingly.

LIMITATIONS

As discussed previously, our current implementation supports 2D visual effects only. To achieve 3D rotation effects with our current implementation, users can only use 2D scaling of objects to simulate the effects (Figure 11). This limitation might be addressed by introducing view perspectives in the animation of 3D assets. By jointly estimating the camera pose and human pose from videos [17], the rendered positions and orientations of the objects are determined by both the human pose and camera pose.

Our current system does not handle occlusions between human bodies and objects. Users have to manually handle occlusions by setting the objects to be invisible when they should be

	Ground Truth				Pose Correction	
video name	*# frames*	*# missing*	*# incorrect*	*time (s)*	*# correction*	*time (s)*
arrow	201	172	45	1196.27	15	125.22
badminton	119	107	21	941.21	11	154.74
boxing	83	103	8	821.16	13	139.25
weight lifting	386	24	27	1528.09	10	296.90
golf	71	69	11	677.61	6	85.89

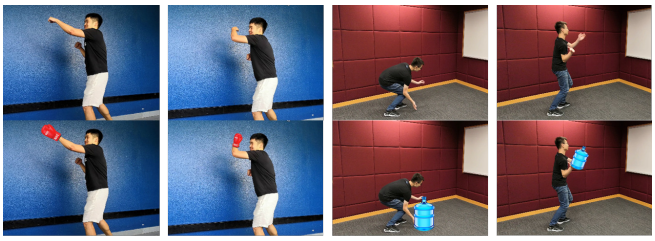
Table 3. Statistics of correcting human poses using our tool. Ground truth includes the total number of frames in each video, number of missing and incorrect joint positions in the raw OpenPose results, and time for manually correcting each error. During correction we record the number of manual corrections and correction times.

	Raw	*Catmull*	*PoseFix*	***Ours(w/o)***	***Ours***
arrow	27.25	9.45	44.75	2.87	**2.14**
badminton	12.93	6.90	30.87	3.10	**1.26**
boxing	53.71	34.87	59.65	12.89	**4.57**
lifting	15.43	7.02	18.37	0.54	**0.42**
golf	25.62	8.73	23.18	1.73	**0.69**

Table 4. A comparison of average joint location errors per frame (Euclidean distance with ground-truth joint locations) between raw human poses [22], the Catmull-Rom spline interpolation, the automatic pose refinement method PoseFix [21], and our semi-automatic pose correction method (*Ours(w/o)*: without manual correction, errors of our missing joint filling method; *Ours*: errors after manual correction). (unit: pixel.)

	badminton		**arrow**		**weight-lifting**
	joint	*limb*	*joint*	*limb*	*joint pair*
# inference	79	79	74	74	116
# failure	6	10	3	4	20

Table 5. The total numbers of association inferences in the exploitation task for each action, and the numbers of failures by automatic inference.



(a) (b) (c) (d)

Figure 11. Limitation: (a)(c) tweening results involving only object rotation in the image plane; (b)(d) results using 2D scaling to simulate object rotations perpendicular to the image plane.

occluded. This might be addressed by segmenting the subjects in video frames [12] to support layered animations.

CONCLUSION AND FUTURE WORK

In this paper we presented the first system that allows novice users to easily add visual effects to human action videos by creating pose-driven tween animations of virtual objects. We designed a user interface that allows users to edit keyframe properties to animate the objects with respect to human actions, and designed an association inference method to associate the objects to driving body parts automatically. Our pose-driven method utilizes the flexibility of human body to both reduce users' efforts in the visual effect authoring process and achieve visually appealing results. We also provided a tool for users to semi-automatically modify human poses computed by pose estimation algorithms.

We have developed only a prototype of *PoseTween*. In the future we would like to explore more functions to make *PoseTween* more powerful. As discussed in Section "Limitations", by incorporating 3D human poses and 3D objects, *PoseTween* can be generalized to editing keyframes in 3D context with depth and perspectives to generate more realistic tween animations. In addition, currently users need to manually find the keyframes of human actions for timings for editing the objects' keyframe properties. There are existing works that detect keyframes in human action videos automatically [39]. Our system has the potential to further simplify the editing process by suggesting timings according to the keyframes of human actions. In *PoseTween*, the keyframes of objects are associated with the keyframes of human actions, it would thus also be interesting to make human action-based visual effect templates, which would be useful for building libraries for entertainment applications. After a user makes a tween animation, the authored keyframe properties are saved as an animation preset. When another user performs a similar action, the preset animation can be adaptively applied to generate an augmented new video automatically.

Acknowledgement

We thank the anonymous reviewers for the constructive comments and the user study participants for their great help. This research was supported by grants from the Research Grants Council of HKSAR (Project No. HKUST16210718), and the Centre for Applied Computing and Interactive Media (ACIM) of School of Creative Media, City University of Hong Kong.

REFERENCES

[1] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. 2018. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7297–7306.

[2] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. 2009. Pictorial structures revisited: People detection and articulated pose estimation. In *IEEE conference on*

computer vision and pattern recognition. IEEE, 1014–1021.

[3] Rahul Arora, Rubaiat Habib Kazi, Danny Kaufman, Wilmot Li, and Karan Singh. 2019. MagicalHands: Mid-Air Hand Gestures for Animating in VR. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. ACM, New York, NY, USA, 12.

[4] Christoph Bregler, Lorie Loeb, Erika Chuang, and Hrishi Deshpande. 2002. Turning to the masters: motion capturing cartoons. In *ACM Transactions on Graphics (TOG)*, Vol. 21. ACM, 399–407.

[5] John Brooke and others. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[6] N. Burtnyk and M. Wein. 1975. Computer Animation of Free Form Images. *SIGGRAPH Comput. Graph.* 9, 1 (April 1975), 78–80.

[7] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7291–7299.

[8] Jiawen Chen, Shahram Izadi, and Andrew Fitzgibbon. 2012. KinÊtre: animating the world with the human body. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 435–444.

[9] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. 2017. Rmpe: Regional multi-person pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*. 2334–2343.

[10] Mihai Fieraru, Anna Khoreva, Leonid Pishchulin, and Bernt Schiele. 2018. Learning to refine human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 205–214.

[11] Hongbo Fu, Chiew-Lan Tai, and Oscar Kin-Chung Au. 2005. Morphing with laplacian coordinates and spatial-temporal texture. In *Proceedings of Pacific Graphics*. 100–102.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 2961–2969.

[13] Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala. 2012. 3D puppetry: a kinect-based interface for 3D animation.. In *UIST*. Citeseer, 423–434.

[14] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. 2016. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*. Springer, 34–50.

[15] Eakta Jain, Yaser Sheikh, Moshe Mahler, and Jessica Hodgins. 2012. Three-dimensional proxies for hand-drawn characters. *ACM Transactions on Graphics (ToG)* 31, 1 (2012), 1–16.

[16] Ming Jin, Dan Gopstein, Yotam Gingold, and Andrew Nealen. 2015. AniMesh: interleaved animation, modeling, and editing. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 207.

[17] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. 2018. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7122–7131.

[18] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, and George Fitzmaurice. 2014. Kitty: sketching dynamic and interactive illustrations. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 395–405.

[19] UI KINECT. 2013. Human Interface Guidelines. *Microsoft Corporation* (2013).

[20] Abdel-Mohsen Onsy Mohamed. 2006. *Principles and applications of time domain electrometry in geoenvironmental engineering*. Vol. 5. CRC Press.

[21] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. 2019. Posefix: Model-agnostic general human pose refinement network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7773–7781.

[22] Daniil Osokin. 2018. Real-time 2D Multi-Person Pose Estimation on CPU: Lightweight OpenPose. *arXiv preprint arXiv:1811.12004* (2018).

[23] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. 2017. Towards accurate multi-person pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4903–4911.

[24] Min Je Park, Min Gyu Choi, Yoshihisa Shinagawa, and Sung Yong Shin. 2006. Video-guided motion synthesis using example motions. *ACM Transactions on Graphics (TOG)* 25, 4 (2006), 1327–1359.

[25] Daniel Roetenberg, Henk Luinge, and Per Slycke. 2009. Xsens MVN: full 6DOF human motion tracking using miniature inertial sensors. *Xsens Motion Technologies BV, Tech. Rep* 1 (2009).

[26] Nazmus Saquib, Rubaiat Habib Kazi, Li-Yi Wei, and Wilmot Li. 2019. Interactive Body-Driven Graphics for Augmented Video Performance. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 622.

[27] Thomas W Sederberg, Peisheng Gao, Guojin Wang, and Hong Mu. 1993. 2-D shape blending: an intrinsic solution to the vertex path problem. In *siggraph*, Vol. 93. 15–18.

[28] Yeongho Seol, Carol O'Sullivan, and Jehee Lee. 2013. Creature features: online motion puppetry for non-human characters. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 213–221.

[29] Wei Shen, Ke Deng, Xiang Bai, Tommer Leyvand, Baining Guo, and Zhuowen Tu. 2012. Exemplar-based human action pose correction and tagging. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1784–1791.

[30] Jakub Smołka and Maria Skublewska-Paszkowska. 2014. Comparison of interpolation methods based on real human motion data. *Przegląd Elektrotechniczny* 90, 10 (2014), 226–229.

[31] Qingkun Su, Xue Bai, Hongbo Fu, Chiew-Lan Tai, and Jue Wang. 2018. Live sketch: Video-driven dynamic deformation of static drawings. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 662.

[32] Ryo Suzuki, Rubaiat Habib Kazi, Li-Yi Wei, Stephen DiVerdi, Wilmot Li, and Daniel Leithinger. 2020. RealitySketch: Embedding Responsive Graphics and Visualizations in AR through Dynamic Sketching. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*.

[33] Sirnam Swetha, Vineeth N Balasubramanian, and CV Jawahar. 2017. Sequence-to-Sequence Learning for Human Pose Correction in Videos. In *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*. IEEE, 298–303.

[34] Carl Vondrick, Donald Patterson, and Deva Ramanan. 2013. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision* 101, 1 (2013), 184–204.

[35] Brian Whited, Gioacchino Noris, Maryann Simmons, Robert W Sumner, Markus Gross, and Jarek Rossignac. 2010. BetweenIT: An interactive tool for tight inbetweening. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 605–614.

[36] Nora S Willett, Hijung Valentina Shin, Zeyu Jin, Wilmot Li, and Adam Finkelstein. 2020. Pose2Pose: pose selection and transfer for 2D character animation. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*. 88–99.

[37] Wenwu Yang. 2017. Context-aware computer aided inbetweening. *IEEE transactions on visualization and computer graphics* 24, 2 (2017), 1049–1062.

[38] Jason Y Zhang, Panna Felsen, Angjoo Kanazawa, and Jitendra Malik. 2019. Predicting 3d human dynamics from video. In *Proceedings of the IEEE International Conference on Computer Vision*. 7114–7123.

[39] Zhipeng Zhao and Ahmed M Elgammal. 2008. Information Theoretic Key Frame Selection for Action Recognition.. In *BMVC*. 1–10.