Pacific Graphics (2018)
H. Fu, A. Ghosh, and J. Kopf (Editors)

TAVE: Template-based Augmentation of Visual Effects to Human Actions in Videos

J. Liu[1] and X. Zhou[1] and H. Fu[2] and C. L. Tai[1]

[1]Department of Computer Science and Engineering, HKUST
[2]School of Creative Media, City University of Hong Kong

Abstract

We present TAVE*, a framework that allows novice users to add interesting visual effects by mimicking human actions in a given template video, in which pre-defined visual effects have already been associated with specific human actions. Our framework is mainly based on high-level features of human pose extracted from video frames, and uses low-level image features as the auxiliary information. We encode an action into a set of code sequences representing joint motion directions and use a finite state machine to recognize the action state of interest. The visual effects, possibly with occlusion masks, can be automatically transferred from the template video to a target video containing similar human actions.*

1. Introduction

Adding visual effects to enhance videos typically requires professional tools with trained skills, such as using Adobe Premiere and Foundry Nuke. New types of entertainment applications, such as FxGuru, allow novice users to add predefined block busters to real-life scenes by aligning the appearing times and positions manually. Existing automatic augmentation methods, such as FaceU and Instagram, are mostly constrained to adding visual effects to hands and faces because they can be detected and aligned easily.

We present *TAVE*, a template-based framework that adds visual effects to videos with respect to the actions performed by a subject in the video. An action involved in this task can be viewed as a series of sub-actions, which follow a semantic order and can be distinguished by the motion states of one or several joints. For example, the sub-actions of the action in Figure 1 are distinguished by hands, and are in the following semantic order: the hands first push out, then stop, and finally pull back. The key insight of our system is to utilize the similarity in the semantic order between the sub-actions in a template video and the sub-actions in a target video. A template is defined for an action, containing both the semantic order of the constituent sub-actions and the appearing times and positions of visual effects. We choose to use a finite state machine to record the semantic order of sub-actions because, besides constraining the semantic order, a finite state machine gives much freedom on the speed a subject can perform an action while ignoring trivial sub-actions by individuals. By letting users imitate the actions, visual effects in the templates are transferred to newly recorded videos automatically without any user intervention.

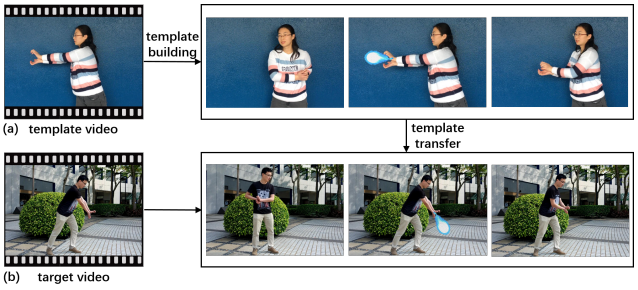

Figure 1: TAVE*: a framework for visual effect augmentation with respect to human actions in videos by (a) building a template from a video with human-action-associated visual effects and then (b) transferring the visual effects onto new videos with similar human actions.*

2. Methodology

2.1. Template Building

Given a template action video, our system takes as input a sequence of human poses represented by joint coordinates, which is extracted by OpenPose [CSWS17] from the video on a frame-by-frame basis. When building a template, the template designer first specifies a subset of joints as dominant joints for distinguishing the sub-actions (e.g., the left and right wrists in Figure 1). The set of spatial location sequences of the dominant joints are obtained directly from the human pose sequence. To make the spatial location sequences recognizable by a finite state machine, we use an encoding diagram [WYH*16] to map the motion direction of each dominant joint between two consecutive frames into a discrete code, such that

DOI: 10.2312/pg.20181289

an action is represented by a set of code sequences of dominant joints.

A crucial step of template building is the construction of the finite state machine. Assuming that the motion directions of all the dominant joints are unchanged within a sub-action, a sub-action is identified whenever the code changes in the code sequence of one or more dominant joints. A finite state machine is built by setting a state for each sub-action in the semantic order, such that the state transition order of the finite state machine records the semantic order of the sub-actions. Each state is represented by a code vector comprising the codes of dominant joints in the corresponding sub-action. Finally, the template designer associates a visual effect to a state corresponding to the sub-action at which the desired visual effect is to appear. For example, the visual effect in Figure 1 is associated with the state corresponding to hands stop. The corresponding constructed finite state machine is shown in Figure 2.

The template also contains other visual effect information, such as height and width of the visual effect, which is recorded in a tuple that can be easily retrieved.

2.2. Template Transfer

In this stage, a user first chooses a template and mimics the action in the template video while having the action recorded. This target video is then processed by OpenPose, and the coordinate sequences of template-designated dominant joints are encoded into a set of code sequences using the same method as in the template building stage. The finite state machine of the chosen template runs on the code sequences of dominant joints to recognize the sub-actions in the target video. Whenever there is a visual effect output associated with a certain state, the corresponding visual effect is properly scaled, rotated and occluded using the information recorded in the template tuple and then added to the target video.

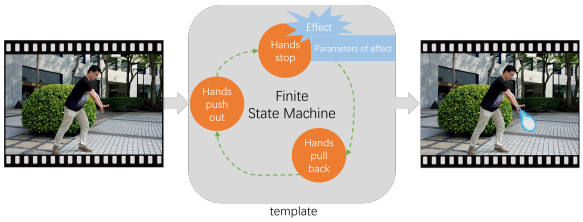

Figure 2: *For a given target video (Left), the finite state machine runs on the code sequences of template-designated dominant joints to recognize the sub-actions and add visual effects to the video at the associated states automatically (Right).*

2.3. Other Implementation Details

For some actions, we want to also leverage the movement of body parts other than the joints, such as hands and mouth, to distinguish sub-actions so that a more accurate trigger time for a visual effect can be detected. We adopt a non-parametric temporal clustering [Lia05] subroutine to divide the image sequence between two states into two clusters in terms of the local appearance of the body part.

When there exists occlusions between human body parts and visual effects, we extract occlusion masks of the occluded parts from template videos using color-based segmentation [IW08] and transfer occlusion masks to target videos. The occlusion masks are fine-tuned using snake contour [KWT88] after transfered to the target video.

2.4. Experiments

In the experiments, we captured action videos by a mobile phone camera with 1080p resolution at 30 fps. For pose estimation, we ran OpenPose C++ library on a desktop (Intel i7-7700 @3.60GHz, 32GB RAM, GTX 1080 Ti). Template building and template transfer were implemented in Matlab on another PC (Intel i7-3612QM @2.10GHz, 16GB RAM) running Windows 8. We tested the framework on a wide variety of actions and visual effects. Some representative experiment results are shown in Figure 3.

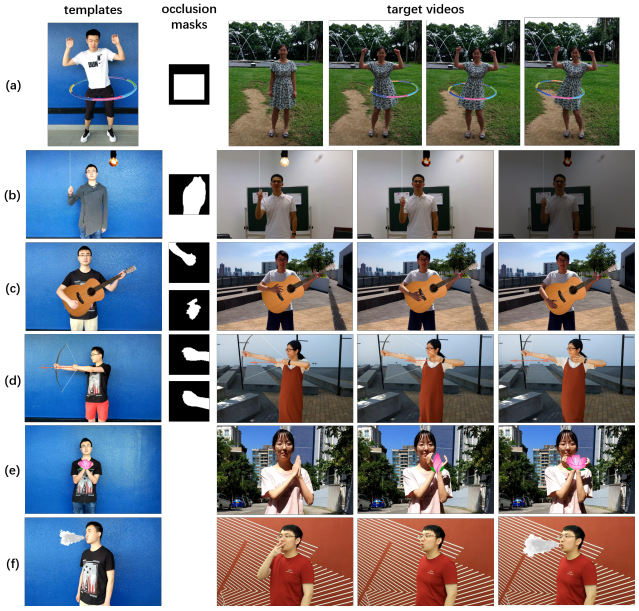

Figure 3: *Representative results of visual effects augmentation.*

References

[CSWS17] CAO Z., SIMON T., WEI S.-E., SHEIKH Y.: Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR* (2017), vol. 1, p. 7. 1

[IW08] ILEA D. E., WHELAN P. F.: Color image segmentation using a spatial k-means clustering algorithm. *Dublin City University*, 2006 (2008). 2

[KWT88] KASS M., WITKIN A., TERZOPOULOS D.: Snakes: Active contour models. *International journal of computer vision 1*, 4 (1988), 321–331. 2

[Lia05] LIAO T. W.: Clustering of time series data-a survey. *Pattern recognition 38*, 11 (2005), 1857–1874. 2

[WYH*16] WANG P., YUAN C., HU W., LI B., ZHANG Y.: Graph based skeleton motion representation and similarity measurement for action recognition. In *European Conference on Computer Vision* (2016), Springer, pp. 370–385. 1