Regenerating Arbitrary Video Sequences with Distillation Path-Finding

Thi-Ngoc-Hanh Le, Sheng-Yi Yao, Chun-Te Wu, and Tong-Yee Lee, *Senior Member, IEEE*

Abstract—If the video has long been mentioned as a widespread visualization form, the animation sequence in the video is mentioned as storytelling for people. Producing an animation requires intensive human labor from skilled professional artists to obtain plausible animation in both content and motion direction, incredibly for animations with complex content, multiple moving objects, and dense movement. This paper presents an interactive framework to generate new sequences according to the users' preference on the starting frame. The critical contrast of our approach versus prior work and existing commercial applications is that novel sequences with arbitrary starting frame are produced by our system with a consistent degree in both content and motion direction. To achieve this effectively, we first learn the feature correlation on the frameset of the given video through a proposed network called RSFNet. Then, we develop a novel path-finding algorithm, SDPF, which formulates the knowledge of motion directions of the source video to estimate the smooth and plausible sequences. The extensive experiments show that our framework can produce new animations on the cartoon and natural scenes and advance prior works and commercial applications to enable users to obtain more predictable results.

Index Terms—animation, sequencing, RSFNet, distillation, SDPF

1 Introduction

Video has long been a widespread media form in our daily life. In addition to visualizing, the sequence of animation in a video is mentioned as storytelling for people. Animating production is usually a specialized and time-consuming job, requiring intensive human labor from skilled professional artists. In traditional cartoon animation (*i.e.,* cel-based and path-based animation) the procedure is complicated and needs much repeated manual labor, and a large amount of cartoon materials have been produced during this procedure. If all these material can be effectively managed and reused, we not only can speed up the time of producing an art but also easily create variations of the existing material. Although the recent computer-aid techniques have removed the burden of artists from tedious work in producing new animations, understanding the content (*i.e.*, character's gesture, background scene, etc.) and finding smooth transitions, are still challenging. The existing commercial applications, *e.g.*, Toon Boom, Adobe Animate, mostly serve the capability on cartoon characters with basic animations. They lack diversity in animation styles and cartoon scenes. Therefore, it's necessary to establish and develop a cartoon images management and retrieval system supporting interactive fast animation making, so that the artists can pay more attention to the creative work, rather than those repeated work like colorizing, repainting, etc.

This problem has been explored. Previous work on this domain can be divided into feature-based and sequence-estimation methods. In feature-based methods, research attempts have been made to get knowledge on image content [5, 28, 26, 3]. Fried et al. [5] train a convolutional neural network to map images into lower dimensional space and define their similarity by a distance calculation. Yu et al. [29] propose an algorithm to construct the feature space according to the shape context of the character in the image and the user's label. However, their dataset is labeled by human judgment, which is difficult or time-consuming to collect. Then, after projecting images into the feature space, the distance metric between two images can be considered as the similarity distance. Nevertheless, the user still needs to manually label the relation between the data. Yang et al. [26] extract three different features of the character in the image's shape context, color histogram, and motion direction. These features are then fused to result in the feature vectors of the character images. But, the segmentation of the character images required by their algorithm is not easy to generate correctly without professional skill.

In contrast, the sequence-estimation methods investigate different approaches to generate a plausible animation. Schödl et al. [21] train a binary classifier and apply the Q-learning algorithm [13] on the images library to produce arbitrary length video sequences. Yu et al. [27] use a semisupervised algorithm to select the next frame of the initial frame according to the similarity distance. Then, they will treat the next frame as the initial frame and repeat this iterative process to generate the results. Recently, Morace et al. [17] construct a graph by the similarity distance of images and compute the shortest Hamiltonian path for reconstructing the sequence from a set of un-ordered images.

However, there are three major drawbacks in the above research. First, they solely focus on cartoon characters. Second, feature extraction and the distance metric used to measure such features are developed independently. And third, with these two mentioned issues, such a prior system is not sufficient to challenge the input clip that consists of dense motion and content. Therefore, we address the demanding problem by combining knowledge learned from

• *All authors are with the Department of Computer Science and Information Engineering, National Cheng-Kung University, Taiwan, ROC.*

Manuscript received May 2022

a self-trained network and modeling them in a path-finding strategy to produce plausible and smooth videos efficiently.

In this paper, we propose a framework to address the above challenges. We aim to create new smooth sequences according to users' preferences of the starting frame. We do not know the sequence we are to generate except the starting frame. Our designed framework attempts to minimize the artifacts caused by *cold transition* and the *flip-flop* phenomenon. The proposed framework pays attention to the pairwise relationship on both content and motion direction of an image and others in the image gallery. Our essential contribution to reducing user effort is automatically propagating user preference to predict a future sequence in a meaningful manner. To achieve this, we present a novel path-finding algorithm that absorbs the knowledge of features in our self-defined network and motion properties in the ground truth, which remedies the drawbacks of prior work.

Our framework consists of an online knowledge learning and an offline sequence generation stage. The online stage learns the feature correlations of pairs of images in a given image set. These feature correlations serve as the initial guidance for new paths explored in the offline stage. The content of frames in real-world videos is complex in both background and foreground. Meanwhile, to model the user's selection to a plausible and novel animation, we need to calculate a meaningful degree of interchangeability between any two frames. We achieve this by proposing a neural network model, *Recursive-based Semantic Feature Network* (RSFNet), to learn the high-level representation of images. It is because the neighborhoods tend to be selected as correlation, which may prevent us from exploring new animations.

In the offline stage, the correlation of images learned in the online stage is performed in a graph. Users can specify their preferences for any node on the graph as the starting frame of their desired animation. Besides the meaningful degree, we need to preserve the temporal coherency in transitions. We tackle this by proposing an algorithm, *Single-source Distillation Path-Finding* (SDPF), in which we embed constraints to interpret potential candidates for plausible animations. In summary, our main contributions are as follows:

- A framework for resequencing videos, which exploits the feature correlation and the motion direction between frames to efficiently produce plausible and smooth video results.
- A framework to extract the representative feature vectors of the images in general style without requiring a large amount of dataset. And, the distance of the vectors can properly match the similarity of the images.
- A novel path-finding algorithm that can synthesize the resultant videos with smooth transitions from the image collection. Moreover, the random selection of our algorithm can increase the diversity of the results, and thus make each resultant sequence distinct from the others.
- Our overall system significantly reduces interaction time required to produce desired results. Besides, the proposed method works well in both cartoon scenes and natural videos, and therefore this enables users to obtain more predictable results.

2 Related work

2.1 Feature Extraction and Dimension Reduction

Researchers seek different approaches in analyzing images to learn the correlation between their representation. Osadchy et al. [19] propose an energy-based model to detect faces with different views. Yang et al. [26] use multiple features of cartoon characters to project images into lower dimensional space. Zhang et al. [31] provide a flexible way for the extraction and completion steps to reflect the unique characteristics of cartoon animation. The transductive algorithm [6] can fuse these different features together and construct a model which projects the character images into lower dimensional space. Combining multiple types of features [29] has achieved great success in many areas. After extracting the feature vectors from the character's shape in the image, users can provide image pairs' positive and negative relationships to restrict the distance between feature vectors.

With the revolution of deep learning technologies, researchers develop the alternative promising approach. Fried et al. [5] analyze patches by embedding them to a vector space in which the texture of image patches are considered to define the similarity of them. Holden et al. [10] use an autoencoder for human manifold. Zhang et al. [30] propose an autoencoder architecture for image clustering. They first train the local stacked contractive autoencoder for the neighborhoods of training dataset based on Euclidean distance metric. Zhang et al. [33] use a convolutional autoencoder network to project the images into lower dimensional space, and the L2 distance between the latent vectors are considered as the similarity of the images. Morace et al. [17] utilize an off-the-shelf network LPIPS [32] to compute the similarity distance between images. Most recently, Xu et al. [25] introduce a dual-task deep learning scheme for separating the structure content in a cartoon animation, i.e., content video and effect video.

Contrasting the above approaches, we handle arbitrary animation objects, including cartoon and natural scenes, rather than only focusing on cartoon characters. We get the knowledge of image representation by a self-defined network which is sufficient to capture comprehensive features. The network requires much less training dataset but has better performance than those in prior work. Plus, it enables us to be independent from such an intermediate network.

2.2 Images sequence ordering

Ordering a collection of images is usually considered as path-finding problem in a weighted graph, in which images are represented by vertices and the weights of the edges are the similarity of two end points, and other constraints such as temporal ordering, path smoothness, or user-control.

A variety of methods have been early developed to create sequences [3, 21, 20]. Given a starting and ending frame, the system proposed in de Juan and Bodenheimer [3] traverses on the manifold to re-sequence an existing cartoon library to a novel animation. Video textures [21] uses L2

distance of raw pixels of images as similarity and applies Q-learning algorithm [13] to generate an arbitrary length video sequence whose motion is similar to input video. Their method can produce convincing results in which input video has repetitive motion or unstructured stochastic motion. However, as the way to calculate similarity cannot adequately describe the high-level features of images, the case of complex structured motion such as human body motion will fail. To overcome this problem, Schödl and Essa [20] extract six specified features from the key-frames to train a binary classifier. This classifier will judge whether the transition between the key-frames will be accepted or not, and the cost of the transition depends on their original video sequences. They use the beam search to find the smoothest sequence. For the better result of the beam search, a hill climbing algorithm is used to interactively minimize the total cost of the sequence from initial random.

The seminal work has motivated researchers to investigate deeper recently [27, 28, 26]. Yang et al. [26] present a cartoon gesture space to cartoon retrieval and synthesis. They use color, shape, and motion information in dissimilarity estimation. Yu et al. [27] propose a semi-supervised algorithm to create new cartoon animation from the image library. They extract the shape context of the characters in the images, and calculate the similarity distance based on the shape correspondence. Inspired by these methods, Yu et al. [28] use a semisupervised multiview subspace learning algorithm to encode different features in a unified space. To model the diverse dynamics, Khan and Storkey [14] introduce a deep generative model for image sequences, in which they split the motion space into subspaces and perform a unique Hamiltonian operator for each subspace.

Some different approaches are recently introduced [33, 17]. To create a sequence, Zhang et al. [33] embed image collection into a convolutional autoencoder network. They then build the proximity graph based on the complete graph of the latent vectors and apply Monte Carlo algorithm to find the smoothest animation sequence. Meanwhile, Morace et al. [17] remove the last 10 percent outliers according to the generalized gamma probability distribution to fine-tune the smoothness of sequence. Then, they find the shortest Hamiltonian path to generate the resequencing results.

The sharp contrast between our framework and theirs [3, 28, 26, 33, 17] is that we develop a novel path-finding algorithm SDPF to generate new sequences with arbitrary starting frame. Our SDPF is faster than a greedy path-finding, effective to explore novel sequence and control the motion consistency.

3 System Overview

The framework of our video resequencing is illustrated in Fig.1, which consists of two primary models: a *semantic relation graph* (SRG) model for representing the relation of images in the given set of images, and a *Single-source Distillation path-finding* (SDPF) algorithm for exploring a path on SRG to resequence the video. Our system takes as input a video, we aim to generate new smooth sequences with arbitrary starting frame while maintaining the consistency in both content relations and temporal coherency.

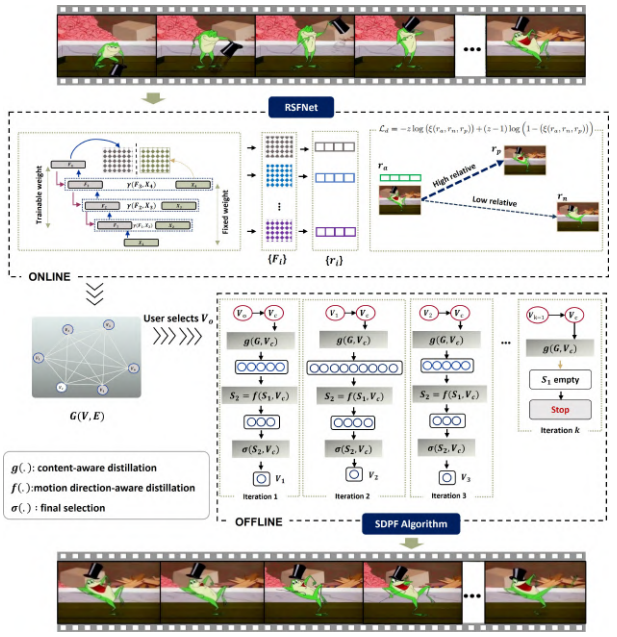

Fig. 1: Our framework for regenerating video sequence.

The SRG models the set of frames in the given video to a completed graph. RSFNet explores this, i.e., the network we propose in this paper. RSFNet shoulders the task of converting images ($\{x_i\}$) into feature representation ($\{v_i\}$) in which every single v_i represents a node in SRG. To describe the semantic relation of v_i, we merge the triplet of recursive-based encoders (called R-Encoder) as a single one, *i.e.*, RSFNet, and train it with a distance loss function. As a result, the connected edges in SRG are assigned by the pairwise distances between feature representations.

Instead of naively traversing the graph and finding the shortest path, which potentially prevents us from exploring a new sequence, we find paths by the proposed SDPF algorithm. Conceptually, SDPF firstly estimates the candidates, which are potential to construct a new sequence, and then distills them through constraints to define the final node at each path-finding-iteration. Finally, the sequence of nodes in the path is mapped to the corresponding frames to produce smooth video results. We subsequently elaborate on each module.

4 Graph Generation with RSFNet

Given a set of frames, we now aim to build a complete graph of this set prior to the resequencing manner. As mentioned in the related work, we propose a network RSFNet to get knowledge on their feature representation and embed the samples to a specific metric space where the similarity or distance between any two samples is clearly represented. Once the distance metric is learned, feature representations and distance are capable of reflecting the relation of input images. RSFNet is a reusable structure that reduces the computational cost and efficiently represents image information. RSFNet shoulders two significant roles in the graph generation: firstly, RSFNet calculates latent vectors corresponding to the given frames. Each vector is treated as a node in the graph. Secondly, RSFNet is trained

with a proposed distance loss to infer the similarity of latent vectors. This manner facilitates the distance of latent vectors more accurate. The details of the proposed framework are presented as follows.

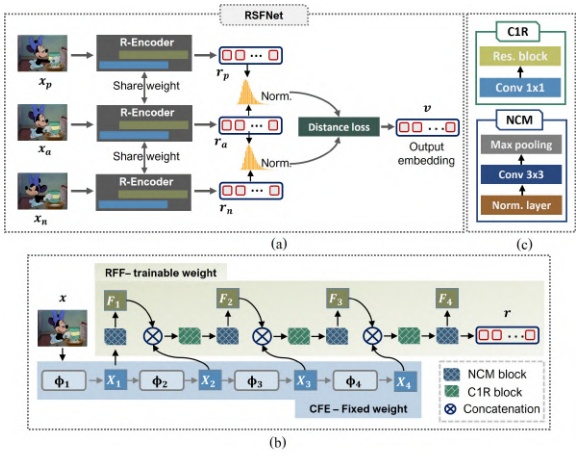

Fig. 2: (a) Architecture of RSFNet; (b) Zoom-in of an R-Encoder; (c) Structure of NCM and C1R block.

4.1 RSFNet Structure

In a common Convolutional Neural Network (CNN) framework, an encoder converts the input image x into a representation vector r ($r = \Phi(x)$). The architecture of an encoder $\Phi(.)$ depends on the input in a specific application. For instance, in the application of image classification, the CNN is a good choice. When applied to video resequencing, such an off-the-shelf CNN might not be suitable since contextual information in a specific frame is necessary for generating new sequences. Besides, human often relies on a high-level semantic understanding of the video contents, usually after viewing the whole sequence, she/he can decide which frame should be selected the next frame in the sequence. Therefore, it is necessary to differentiate the target sequence scene to make the resultant sequence semantic, reasonable, and smooth. At this point, an encoder with a pure CNN structure may lack sufficient information for such an appealing sequence.

Motivated by the above reason, we design our RSFNet by the triplet of **R-Encoders**, which share the parameters, i.e., weights and biases. Each R-Encoder consists of two modules, *Coarse Feature Extractor* (CFE) and *Recursive Feature Fining* (RFF). The design is visualized in Fig.2. For the CFE module, we treat it as an extractor to obtain the initial feature maps. The backbone network of CFE is based on the VGG-19 network [23]. This pre-trained network is widely used in several applications in a feature extraction manner. Hence, it is reliable to be considered a good feature extractor. Furthermore, VGG-19 has been trained on a large-scale dataset. With this strategy, we can reduce the burden in training for this process. We solely use the first four blocks and remove the fifth block from the original VGG-19 framework since it lacks pixel-wise content information [7]. An input image $\mathcal{I}$ is firstly fed to CFE. Let matrix $\mathcal{X}_i \in \mathrm{R}^{H_i \times W_i \times k}$ denote corresponding feature maps produced by four layers of CFE. Here, k is the number of channels of each feature response. H_i and W_i are respectively the height and width of the feature maps in layer i ($i = 1 \ldots 4$). As shown in Fig.2-(b), the feature maps $\mathcal{X}_i$ are enhanced along the channel and space dimensions to obtain the feature maps $\mathcal{F}_i$ by RFF module. In other words, instead of directly utilizing feature maps from CFE, we propose an RFF module to integrate with CFE to produce features that can depict the variety content in frames. The effectiveness of this design is visualized by the analysis in the later session **A.1**.

The ***Recursive Feature Fining*** (RFF) module is the core of an R-Encoder, which shoulders the task of preserving contextual information of images during encoding into latent space. RFF is formulated by recursively integrating feature maps of CFE. A straightforward technique could be used instead of RFF is that re-scaling the feature maps obtained from CFE and combining them together. However, the feature extraction from a backbone, *e.g.* either VGG or ResNet, is performed by a repeated process of convolutional and max-pooling operations. These extracted features by themselves loss the low-level information that is likely to aid in discriminating object regions from the background regions. Thus, such a simple technique, *i.e.*, re-scaling, might neglect smaller objects or information in the background regions and eventually decrease the capability of the encoder. In the structure of our RFF, we embed two blocks, NCM and C1R. NCM is to normalize the input feature maps before the concatenation. Meanwhile, the C1R block's task is to compress the size of feature maps without losing information.

The network architecture of the proposed RFF is shown in Fig.2-(b). Four feature maps with different resolutions obtained from CFE ($\mathcal{X}_i$) are taken as the inputs of the RFF. Mathematically, the above process can be recursively expressed as:

$$\begin{cases} \mathcal{F}_i = \Psi(\varphi(\mathcal{F}_{i-1} \otimes \mathcal{X}_i)), (i = 2 \ldots 4) \\ \mathcal{F}_1 = \Psi(\mathcal{X}_1) \end{cases}, \quad (1)$$

where $\Psi(.)$ and $\varphi(.)$ denote the functions from the NCM block and C1R block, respectively; $\otimes$ is the concatenation operation. By concatenating two different feature maps, resultant feature maps $\mathcal{F}_i$ ($i > 1$) simultaneously captures two different receptive fields.

To be more specific, NCM is designed to enhance the spatial representation for the input feature maps from VGG-19. This block performs the *Normalization*→*Conv3*×*3*→*MaxPooling* structure. Output of input feature maps $\mathcal{F}^{in}$ passed through NCM is performed as

$$\Psi(\mathcal{F}^{in}) = P(C^3(Norm(\mathcal{F}^{in}))), \quad (2)$$

where $P(.)$ represents the Max-Pooling operator; $C^3(.)$ indicates the standard convolution with the kernel size of 3×3; and $Norm$ is a normalization operator.

C1R employs a 1×1 point-wise convolution and a residual block. Our residual block consists of two batch normalization (BN) layers and two 3×3 convolutional layers. Note that, compared with the basic residual block [9], our residual block removes the RELU layer after the first convolutional layer to preserve more spatial details. See Fig.2-(b), immediately after the concatenation which is used to transmit the information of these two distinct layers, this

block is embedded to learn the correlation of feature maps from different layers. This process is expressed as:

$$\varphi(\mathcal{F}_c) = C^1(\mathcal{F}_c) + BN(C^3(BN(C^1(\mathcal{F}_c)))), \quad (3)$$

where $\mathcal{F}_c$ is the resultant feature maps after the concatenation phase. C^1 represents the 1×1 point-wise convolution.

With our above design, some benefits can be gained. First, using a pre-trained network as a backbone significantly reduces the training cost. Second, RFF can be easily embedded into an existing neural network. In the design of C1R block, 1×1 convolution is to increase channels corresponding to the previous layer. Meanwhile, residual connection sufficiently mitigates the gradient vanishing problem, which usually occurs when training the deep network.

We need to build the embeddings of frames such that they have the following properties: (1) two similar frames produce two embeddings so that the mathematical distance between them is small, and (2) two very different frames produce two embeddings so that the mathematical distance between them is large. To do that, we model RSFNet that contains the triplet of R-Encoders, which use the same weights while working in tandem triplet of different input vectors to compute comparable output vector. In our training, the distance loss $\mathcal{L}_d$ is used as the objective function to reinforce the distance between two latent vectors to match the similarity of the images well reflect pixel-level image similarity. We train RSFNet using a set of triplet images - an anchor x_a, its positive x_p, and negative x_n. Detail of preparing such triplet data is discussed in our supplementary material. For three embeddings r_a, r_p, r_n of the images x_a, x_p, x_n, respectively, the formula of the distance loss is as follows.

$$\mathcal{L}_d = -z \log\big(\xi\big) + (z-1)\log\big(1-\xi\big), \quad (4)$$

where

$$\xi = \gamma(\parallel r_a - r_n \parallel_2 - \parallel r_a - r_p \parallel_2), \quad (5)$$

and

$$z = \begin{cases} 1, & \text{if } d_p(x_a, x_n) > d_p(x_a, x_p) \\ 0, & \text{if } d_p(x_a, x_n) < d_p(x_a, x_p) \end{cases}. \quad (6)$$

Here, $\gamma(.)$ is the sigmoid function [18], used here to ease the severe gradient problem. $d_p(.)$ is the PSNR measurement [11]; Eq.(6) is used to get the initial knowledge about the similarity of image pairs. It indicates which term in Eq.(4) will be visible during the training. Therefore, z can be treated as an indicator to determine whether x_a is similar to x_n or x_p. We note here that using any pixel-level distance metric in Eq.(6) could yield an equivalent effect. It is clear that Eq.(4) encourages the embedding of x_a to be closer to x_p than to x_n. Optimizing these terms boosts the margin between distances of negative pairs and distances of positive pairs. The effectiveness of this formulation is discussed by ablated results in later session **A.2**.

4.2 Learning-Based Euclidean metric

To define the weight of each edge in the complete graph, we calculate the distance of all pairs of latent vectors. The distance metric used in this manner should satisfy two criteria: (1) it can well reflect the distance of images, i.e., a distance in low dimensional space should be consistent with the content correlation of images, and (2) not too expensive to reduce the burden when the number of given images is significant. Our early experiments considered five different distance metrics: the Hausdorff distance, Earth Movement Distance (EMD), the LPIPS distance, SSIM, and the Euclidean distance. However, Hausdorff and EMD have a good performance on specific data, i.e., cartoon characters [3, 26]. LPIPS is an expensive computation metric, it takes approximately five seconds on an image pair. SSIM and Euclidean distance metrics are potential. However, Euclidean metric is the most common use of distance measure and known as simple distance. When data is dense or continuous, this is the best proximity measure. Thus, we consider Euclidean as the baseline in learning the relation of images in our current application. It's worth noting that directly using pixel-level distance metrics, such as SSIM or Euclidean, without Eq.(4) is not sufficient in our current application. The reason is that we target to explore new transitions on the diverse content frames. Simply employing a plain distance metric without the objective function $\mathcal{L}_d$ prevents us to reach this goal. This could be seen in the ablated visualization **A.2**

Instead of directly using Euclidean distance to measure the metric value between two features, in RSFNet, we apply deep learning technique to further learn their similarity. When the self-defined metric space is an Euclidean space, the metric value between two samples is a distance metric, which is defined as:

$$d_{ij}(v_i, v_j) = \parallel \mathcal{R}(x_i) - \mathcal{R}(x_j) \parallel, \quad (7)$$

where $\mathcal{R}$ is our trained RSFNet; x_i and x_j are the corresponding frame of embedding v_i and v_j, respectively.

5 Single-source Distillation path-finding

In this section, we present our approach of finding the path on the complete graph to construct new sequences. Let Ω be the set of latent vectors v_i obtained from our RSFNet and d_{ij} be the distance between two latent vectors $v_i, v_j \in \Omega$ defined by Eq. (7). We construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in which each node $V_i \in \mathcal{V}$ represents a latent vector $v_i \in \Omega$ and the weight of each direct edge $e_{ij} \in \mathcal{E}$ (from V_i to V_j) is assigned by the corresponding distance $d_{ij}(v_i, v_j)$.

Once graph $\mathcal{G}$ is constructed, our system lets the user choose a node randomly. An expected sequence can be constructed by traversing the graph starting from this node. A possible and straightforward way is finding the shortest path on the graph with the selected node because the edge of a pairwise node reflects their similarity, i.e., if the weight of an edge is smaller, the connected nodes are more similar and vice versa. Hence, this naive strategy is tolerant of plausible sequences if the input clips do not have dense motion and content.

The question here is - *How do we construct the sequences that are different from those in the input video?* Resequencing videos without pre-processing (e.g., extracting objects from the background), we may face a range of challenges in image content (e.g., the video has multiple moving objects, dense motion directions, or with complicated background). Generating new sequences while avoiding flicking artifacts, such a classic shortest path-finding technique by itself is not

tailored. The reason is that the resultant path found by this technique, such as [17], tightly reflects the similarity of the sequence in the given clip. It may yield a similar sequence to the given sequence. Otherwise, it may fall into chaotic motion if the clip has dense movements. Yang et al. [26] tackle this issue by extracting the cartoon character from the image content and using the motion direction feature (MDF) to evaluate the gesture dissimilarity. However, they focus on the frames that have a single cartoon character. If the frames demonstrate the motion of multiple objects, using MDF might be insufficient. Recent work by Morace et al. [17] also suffers from this issue if there exist dense motion directions (Chinese ink in Fig.5-(K)). To overcome these challenges and produce new sequences, we propose an algorithm called *Single-source Distillation Path-Finding* (SDPF) to find the path when traversing on the graph.

When designing SDPF, we base on the fact that the adjacent frames have to be consistent in content information and temporal coherency in a particular clip. Thus, in finding a new path that satisfies these two aspects, we consider them whenever choosing a node at every step. We call the phenomenon, which is caused by missing one of the two aspects, as a *cold transition*. More specifically, we model our SDPF to work under the control of two-layer distillation. Given a graph and a starting node, the first layer is to distill the set of candidates, which are the potential to be consistent with the content. Taking this set as input, the second layer estimates the plausible motion direction that could be generated and distill the candidates on the set that are potentially temporally coherent. In the following, we call the current node V_c; our SDPF aims to find the adjacent node of V_c, denoted as V_{c+1}. We visualize the difference of Single-source shortest PathFinding (SSPF) versus our proposed SDPF in Fig.3(a), (b). SSPF chooses only one node, which has the shortest cost, to add to the path. In contrast, our SDPF considers a number of nodes, *e.g.*, three nodes in this example, which have the cost lower than a designated threshold, as the potential candidates in equivalent probability to be added to the path.

There are several benefits of using our SDPF algorithm. First, we can explore new paths since we do not strictly follow the theory of the shortest path. Second, we can control the motion direction to be locally consistent in clip segments and globally realistic in the generated clip. Third, it is faster than such a greedy strategy. We subsequently describe how we model the constraints in our SDPF algorithm. The pseudo-code of SDPF is presented in Algorithm 1.

Algorithm 1 SDPF Algorithm

Input: Set of latent vectors $\{v_i\}$, distance metric $\{d_{ij}\}$
$\mathcal{V} \leftarrow \{v_i\}$, $\mathcal{E} \leftarrow \{d_{ij}\}$;
Construct graph $\mathbf{G} = (\mathcal{V}, \mathcal{E})$;
$V_o \leftarrow$ user's selection;
Initialize a list $\mathcal{P}$ to subsequently push the selected node to the path;
Add V_o to $\mathcal{P}$
$V_c \leftarrow V_o$; */* V_c is the node at current state*/*
/ Distillation in the first layer*/*
for each node $V_j \in \mathbf{G}(\mathcal{V} - \mathcal{P})$ **do**
  if $e_{cj} < \eta$ **then** */* η is defined in Eq.(8)*/*
    Add V_j to $\mathcal{S}_1$
  end if
end for
/ Distillation in the second layer */*
for each node $V_k \in \mathcal{S}_1$ **do**
  if $V_c \in LMS$ **then**
    $\mathcal{S}_2 = C_d(V_c, V_k) + C_t(V_c, V_k)$
  else
    $\mathcal{S}_2 = C_t(V_c, V_k)$
  end if
end for
for each $V_i \in \mathcal{S}_2$ **do**
  Compute possibility Ω for each V_i by Eq.(20);
end for
Choose V_i by randomly selecting Ω;
Add V_i to path $\mathcal{P}$;
Update $V_c \leftarrow V_i$
Output: Sequence of path $\mathcal{P}$

5.1 Content-aware distillation

In this layer, we find the set of candidates that have relevant content to the current node V_c rather than finding the node that have smallest distance to V_c. Obviously, if V_{c+1} is the node that has the smallest weight to V_c among the directed nodes of V_c, this may yield the resultant sequences that are similar to the source sequences. Thus, we find the candidates that are potential to explore new transitions. This saves the generated video from flicking artifacts due to the "*cold transition*" between them. We construct a set $\mathcal{S}_1$ of candidates that are relevant to V_c as:

$$\mathcal{S}_1 = \{V_i \in \mathcal{G} : e_{ci} < \eta; \text{s.t. } \eta = \frac{\sum e_{ij}}{N}\}, \tag{8}$$

where N is the total number of nodes in the graph $\mathcal{G}$. In this equation, η is the threshold that represents for the mean of the weights in the graph $\mathcal{G}$. By this configuration, an edge has the weight that smaller than η could be considered as a "*potential candidate*". It is hypothesized that we set another variable such as top $k\%$ of the candidates that have the closest weight to the minimum weight of the graph, the size of $\mathcal{S}_1$ is increasing with the total number of frames in the given video. If the clip is short, the size of $\mathcal{S}_1$ is small, and thus it might be not sufficient to explore a new path. If the clip is long, the size of $\mathcal{S}_1$ accordingly increases, and thus it might include the wrong candidates (i.e., the candidates are not correlated). Therefore, the threshold $\eta < mean(.)$ is tolerant with different amounts of frames and able to avoid these phenomena.

With Eq.(8), we can eliminate the nodes that have low reliability and drive our focus on the nodes that are highly potential to be content correlation. Each element of $\mathcal{S}_1$ represents for a possible way that we can explore from V_c without suffering from *cold transition*. Note that the size of $\mathcal{S}_1$ varies along with V_c at each iteration. And in $\mathcal{S}_1$, the nodes are treated equivalently, i.e., the probability of choosing a node is independent with the edge weight.

5.2 Motion direction-aware distillation

Having computed the set $\mathcal{S}_1$, the next question is - *which candidates in this set can yield a plausible motion direction?* Note that here "*plausible*" refers to both backward motions,

forward motions, or any movement but avoiding the results from the flip-flop and jumping phenomena. To do this, we propose to our SDPF algorithm two constraints: *Directional distillation* and *Coherent distillation*. Detail of each constraint is described as follows.

5.2.1 Directional distillation

This constraint, denoted as C_d, is proposed to control the consistency of motion direction. To achieve this, linear motion segment (LMS) is the factor we consider here. LMS is ubiquitous in real-world videos. Readers can see the visual example of LMS here [1]. Resequencing such videos may fall into two kinds of motion-noise: (1) flip-flop phenomenon due to the LMS-frame is not recognized, causing inconsistent direction, and (2) abnormal motion since both backward and forward motions yield smooth transitions. Therefore, there is a need to recognize the major motion direction in frames as well as detect the LMS to avoid these motion-noises.

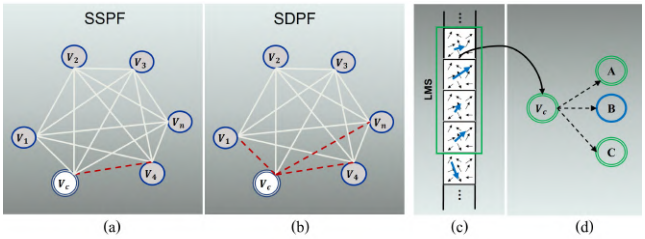

Fig. 3: Visualization of SSPF (a) and SDPF (b). (c) is visualization of motion tendency of each frame (*i.e.*, blue arrow). The frames in green rectangle belong to an LMS, *i.e.*, the adjacent frames satisfy Eq.(11). (d) is visual constraint C_d in Eq.(12), here the node outlined by double circle means it is an LMS-frame.

Let $\mathcal{X} = \{x_a\}_{a=0}^{n}$ be the sequence of frames x_a in the given video, n is the number of frames, we first calculate the optical flow [24] of $\mathcal{X}$ and denote this set as $\mathbf{Y} = \{F_{a\to a+1}\}_{a=0}^{n-1}$, here $F_{a\to a+1}$ is the optical flow of frame x_a to x_{a+1}. To focus on drastic changes in the optical flow, we normalize each element in $\mathbf{Y}$ as follows:

$$\widehat{\mathbf{N}_{ij}} = \frac{\parallel F_{ij} \parallel_2 - \min_{ij} \parallel F_{ij} \parallel_2}{\max_{ij} \parallel F_{ij} \parallel_2 - \min_{ij} \parallel F_{ij} \parallel_2}. \tag{9}$$

We denote this set as $\widehat{\mathbf{Y}} = \{\widehat{\mathbf{N}^a}\}_{a=0}^{n-1}$. Video frames may have various motions, *e.g.*, motions of main object(s) or light motions of background objects. To recognize the major motion direction in frames, we mask on each frame a value called *"motion tendency"* (T), as shown in Fig.3(c). This value represents for the motion direction that dominates in a frame, which is formulated by average normalized vectors of partial optical flow:

$$T = \angle\left(\frac{\sum \widehat{\mathbf{N}_{ij}}}{n \times m}\right), \text{ s.t., } \widehat{\mathbf{N}_{ij}} > \sigma \frac{F_{ij}}{\parallel F_{ij} \parallel_2}, \tag{10}$$

where m, n is the width and height of frames, respectively. Threshold σ is set to 0.5 in our experiments to ensure only huge changes to be concentrated.

1. http://graphics.csie.ncku.edu.tw/SDPF/LMS.mp4

Thereafter, we base on motion tendency in frames to detect LMS-frame, the frame that belongs to an LMS. The definition of an LMS-frame is expressed as:

$$\exists j, k \in \mathrm{N} : j \le i \le j+k, k > 2, \text{ s.t. } |T_j - T_l| \le \delta, \tag{11}$$

for all $l \in [j, j+k]$. In our experiments, we compute motion tendency of frames and mask them with motion tendency value if the frames belong to LMS prior of path-finding manner, and threshold δ is set to $\frac{\pi}{4}$.

Finally, we configure the constraint for directional distillation as:

$$C_d = \left|T_c - T_k\right| \le \xi, \textit{ if } x_c \in \text{LMS} \,\&\, \exists V_k \in \mathcal{S}_1 \textit{ s.t., } x_k \in \text{LMS}, \tag{12}$$

where T_c, T_k is the motion tendency of the corresponding frame x_c, x_k of the node V_c, V_k, respectively, $k \in [1 \dots n_1]$, and ξ is set to $\frac{\pi}{3}$. Here, n_1 is the size of set $\mathcal{S}_1$. The condition in Eq.(12) reveals that constraint C_d only works if the corresponding frame of V_c is an LMS-frame and there exist an LMS-frame in $\mathcal{S}_1$. Otherwise, we skip this constraint. The visual sample can be found in Fig.3(d). We can see that, V_c and two of its three candidates are LMS-frames. In this case, Eq.(12) is used to avoid flip-flop phenomenon. We analyze the effectiveness of this constraint with ablated results in session **A.3.1**.

5.2.2 Coherent distillation

The distillation in this layer is proposed to maintain the temporal coherency in generated sequences. Yang et al. [26] extract cartoon characters from frames and compute the angle of two motion direction features of the characters to define the differences of motions. In the cases that video frames consists of multiple moving objects, this technique is not practical. We instead propose a *Pixel-wised Motion Similarity Measurement* (PMSM) to shoulder the smoothness of generated sequences.

As named, PMSM measures the pixel-wise motion similarity between two frames. To get knowledge of motion in frames, inspired by [12], we use optical flow as the motion feature. Thus, a possible and straightforward way we can measure the motion differences is using optical flow directly. Nonetheless, as aforementioned, various motions of multiple objects in frames cause challenging to define the consistency between them. We therefore learn the motion feature by mapping optical flow domain to image domain. In other words, given two frames, we use the corresponding the optical flow of these frames to construct the instance in image domain, dubbed *pseudo-image*. A pseudo-image is made by the major motions in the corresponding frame and the correlated motion of frame-pair. We finally calculate the distance of pseudo-images to measure how smooth the motion changes in a transition. A smaller PMSM reveals a smooth transition. Consequently, we use PMSM to configure the constraint in this distillation layer, so-called $\mathbf{C}_t$, to control the motion in adjacent frames not to change frequently or drastically.

Fig.4 outlines the flowchart of PMSM. For each node V_k in the set $\mathcal{S}_1$, we treat it as a hypothesized adjacent node of V_c. And x_c, x_k respectively are the corresponding frames of node V_c, V_k. The smoothness of transition from frame x_c to x_k is now defined by the motion distance of two optical

flows FC and FK, where FC is the optical flow of frame x_c to its backward adjacent frame in the input video. The reason for the order of this calculation is explained in detail in our supplementary. Similarly, FK is the one of x_k.

With two optical flows FC and FK, we first normalize their magnitude by Eq.(9), denoted as $\widehat{\text{FC}}$, $\widehat{\text{FK}}$, respectively. We then define a map of significant motions with:

$$\mathcal{M}_{ij} = \max(\widehat{\text{FC}}_{ij}, \widehat{\text{FK}}_{ij}), \tag{13}$$

with $i = 0 \dots W$, $j = 0 \dots H$; W, H is the width and height of the frame, respectively. The map $\mathcal{M}$ represents for the correlation of major motions in frame-pair. We get these information to learn how to control the pixel-wise consistency in the pseudo-images.

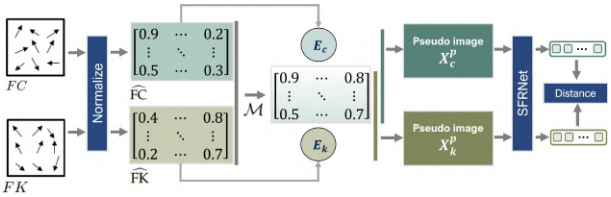

Fig. 4: Flowchart of our proposed PMSM.

In each normalized optical flow $\widehat{\text{FC}}$, $\widehat{\text{FK}}$, we count the number of elements that are larger than a threshold μ. We denote as $E = \{e_1, \dots, e_{N_e}\}$. Note that the size of set E varies along $\widehat{\text{FC}}$ or $\widehat{\text{FK}}$. Since we only use N_e optical flows in $\widehat{F}$ to model C_t, value of N_e should be large enough. A small N_e will decrease the difference between optical flows. This yields to that we may fail to define the difference correctly. Therefore, if N_e is smaller than a threshold, i.e., 224 is the height and width of frames, we will cut μ in a half and compute μ again to ensure N_e is sufficient. Initially, we set μ to $\frac{1}{2}$.

Thereafter, we rely on $\mathcal{M}$, E to map back to the input optical flow to construct pseudo-image. More specifically, $\forall i, j$ in optical flow $F_{ij} = (x_{ij}, y_{ij})$, pseudo-image $\mathbf{X}^p \in \mathbb{R}^{H \times W \times 3}$ is expressed as:

$$\mathbf{X}^p_{ij} = \begin{cases} \left(\frac{x_{ij}}{2\|F_{ij}\|_2} + \frac{1}{2}, \frac{y_{ij}}{2\|F_{ij}\|_2} + \frac{1}{2}, 1\right), \text{ if } \mathcal{M} \geq e_k \\ \left(\frac{1}{2}, \frac{1}{2}, 0\right), \text{ if } \mathcal{M} < e_k \end{cases}, \tag{14}$$

where e_k is the largest element of E in $\mathcal{M}$. In Eq.(14), if the parameters are $\mathcal{M}$, E_c, and $\widehat{\text{FC}}$, we can construct pseudo-image of node V_c, denoted as $\mathbf{X}^p_c$. Similarly, we can get the pseudo-image $\mathbf{X}^p_k$ from those of node V_k. The first two channels in $\mathbf{X}^p$ are the unit vector of F with constant translation, in which unit vector provides only direction information. The constant translation makes the value to be in range of $[0, 1]$ without any computation error. The third channel is used to enlarge the difference between the feature point and other pixels.

At the end, motion distance of two optical flows is formulated as the similarity of the corresponding pseudo-images:

$$\delta(FC, FK) = - \parallel \mathcal{R}(\mathbf{X}^p_c) - \mathcal{R}(\mathbf{X}^p_k) \parallel_2, \tag{15}$$

where $\mathcal{R}(.)$ indicates our trained RSFNet. In essence, pseudo-images have different appearance compared to video frames, *i.e.*, pixel value represents for the motion intensity of objects in the corresponding frame. Encoding such pseudo-image serves the knowledge of the regions that have considerable motions. It's worth noting that motion distance of two frames in Eq.(15) also could be expressed by the similarity of pseudo-images. However, to make $\delta(.)$ stable when working on diverse motions, we feed them to RSFNet. Although RSFNet is trained on video frames data, RSFNet on the other hand learn a similarity function to see if two images are the same. This enables to discriminate new classes of data without training the network again. We give out discussion and visualization on these effects in the supplementary file.

Equation (15) represents the relation adjacent frames in term of motion change degree. For each node V_k in the set $\mathcal{S}_1$, $k = 0, \dots, n_1$, we define constraint C_t as:

$$\delta(FC, FK) \leq \omega, \tag{16}$$

where ω is set by:

$$\omega = \begin{cases} \frac{1}{n_1} \sum_{k \in \mathcal{S}_1} \delta(FC, FK), \text{ if } n_1 \geq 2 \\ \min\left(\delta(FC, A_1), \delta(FC, A_2)\right), \text{ if } n_1 < 2 \end{cases}, \tag{17}$$

here A_1 and A_2 are the augment form of FC, i.e., A_1 is the rotation of FC with angle $\frac{1}{2}\pi$ and A_2 is the rotation of FC with angle $-\frac{1}{2}\pi$. We set ω as the average difference of $\mathcal{S}_1$ is intuitive. However, the average will loss its function if the number elements in $\mathcal{S}_1$ is less than 2. Therefore, we calculate the difference between FC and the rotation of itself to ensure the direction of the motion is sufficiently smooth. We analyze the effectiveness of this constraint by the ablated results in later session **A.3.2**.

In summary, the constraint model of distillation in this layer can be factorized as:

$$\mathbf{C}_d(V_c, V_k) + \mathbf{C}_t(V_c, V_k), \tag{18}$$

where $V_k \in \mathcal{S}_1$. In the cases that V_c does not belong to LMS or there does not exist a candidate in $\mathcal{S}_1$ that belongs to LMS, the first factor in this equation is omitted. In other words, we define the candidates that we can add to the path as:

$$\mathcal{S}_2 = \begin{cases} \mathbf{C}_d(V_c, V_k) + \mathbf{C}_t(V_c, V_k), \text{ if } V_c \in LMS \\ \mathbf{C}_t(V_c, V_k), \text{otherwise} \end{cases} \tag{19}$$

5.2.3 *Final selection*

Thus far, the candidates in $\mathcal{S}_2$ are the possible nodes we can choose to explore. In the cases that $\mathcal{S}_2$ is empty, the algorithm will early stop to maintain the quality of resultant clips. If $\mathcal{S}_2 > 1$, we adopt Softmax parameterization protocol [8] to converge the selection in each iteration. Let δ_j be the motion distance from a node $V_j \in \mathcal{S}_2$ to V_c, we parameterize the possibility of choosing V_j as:

$$\Omega(V_j | \mathcal{S}_2) = \frac{\exp(\delta_j)}{\sum_{i=1}^{n_2} \exp(\delta_i)}, \tag{20}$$

where n_2 denotes the number of candidates in $\mathcal{S}_2$, δ_i is the motion distance of node V_i and V_c. This equation is used to compute the possibility of a vertex to be chosen. Then, we select the adjacent node of V_c according to randomly choose the possibility Ω. It's worth pointing that choosing any candidates in $\mathcal{S}_2$ is sufficient to guarantee smooth and plausible sequence. However, we aim to explore the novel

path, we thus utilize Eq.(20) to increase the possibility of sequencing novelty rather than choosing the smallest edge-weight node. Furthermore, this strategy enables users to have more predictable results. The efficiency of this design is visualized by video results in the supplementary video.

6 Experimental Results

6.1 Implementation Details

We implemented our proposed resequencing system in Tensorflow [1]. All experiments were performed on a PC equipped with Intel Core i7-770 CPU, 16GB RAM and an NVIDIA GTX 1070 GPU. The User Interface (UI) is developed by QT toolkit [4]. We train our model with patch size of 8. Adam optimizer [15] is used. Early-stopping with 10 epochs patience is used to prevent over-fitting. To reach the minimum of loss, we cut the learning rate in half when the validation loss does not improve in 3 epochs.

6.2 Our results and discussion

Fig.5 exhibits the frames of some typical videos in our experiments. Readers are encouraged to explore our project website[2] to access more visual results. The aspects that make our results and system advance prior works could be summarized as follows.

We are capable of resequencing both cartoons (Fig.5-(A) to (G)) and natural videos (Fig.5-(H) to (L)). Cartoon images often consist of sharp lines, flat backgrounds, and smooth color blocks, while natural images contain more complex and local textures [2]. This ability is adopted by benefiting of the proposed RSFNet and the distance loss. RSFNet boosts the performance of our system in understanding high-level features of natural images; meanwhile, the distance loss facilitates the accuracy of image feature-pairs similarity.

We are capable of resequencing the clips, which consist of complex motions, i.e., the motion of multiple objects or dense motion directions. This aspect is adopted by the *Motion Direction-Aware Distillation* in our SDPF algorithm. As examples, let us take Fig.5-(F) and Fig.5-(G). The challenge here is that both cases consists of multiple simultaneous motions. In Fig.5-(G), we have to control the consistency of movements of two objects: 1) the direction when the bear raises his hand to hold the flower and rotates it, and 2) the other flower waves with the wind. Meanwhile, in Fig.5-(F), such a resequenced clip should maintain the consistency of the movements of the lady, baby, the car, and the windshield wipers. Nevertheless, we can generate appealing results, *i.e.*, we re-sequence the new clips without damaging the coherency and flicking artifacts. The challenge also falls in the natural scenes here. These samples encompass linear motion segments, which cause resultant sequences to be a flip-flop phenomenon. Thanks to the constraints embedded in our SDPF, we revolve this challenge and produce smooth transitions.

Another interesting aspect of our system is the ability to produce the sequences which are different to those in the given clip. This aspect is adopted by the *Content-Aware Distillation* in our SDPF Algorithm. More specifically, we visualize the filmstrips of two paths which are from the original video and our result in Fig.6. We can see with the same image gallery, but our sequence is quite different from those in the input video. By observing this resequencing result, we can see the transition of each single frame pair is plausible. The full clips can be seen in our supplementary videos.

In addition, we can generate different sequences according to the starting frame, which the user selects. This aspect enables users to obtain various predictable results. Fig.6 is a sample. More results can be seen on our project website. By observing the filmstrips in the figure, the sequence generated by our method is not only relatively different from the source sequence but also smooth in transitions.

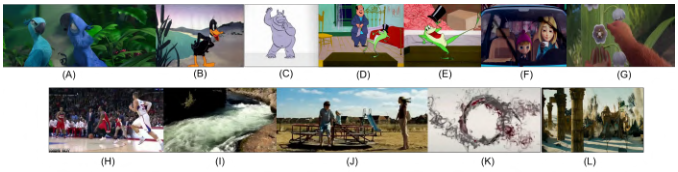

Fig. 5: Frames in some of the videos we use to evaluate our method.

Fig. 6: Demonstrates the differences in the sequence generated by our method versus those in the original video. Shown in this figure are the filmstrips from original video (first row) and our rendered video (second row).

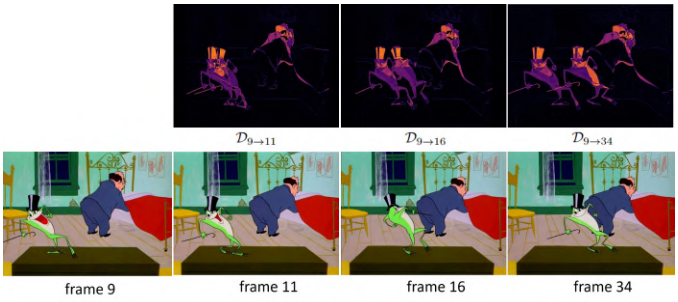

Fig. 7: Visualizes the heatmap of differences of frames.

6.3 Evaluation Metrics

To evaluate the performance of the proposed method, we measure the generated sequences with three aspects: (1) the stability of videos, (2) the difference degree of generated sequences, and (3) human perception on our results. In this evaluation manner, we totally use 12 videos (shown in Fig.5), which are rendered from our system. Then, we synthesize them for the below evaluation metrics.

6.3.1 ***E.1.*** *Stability measurement*

As the generated videos are explored according to the user's selection of the starting frame, they may not have the

2. http://graphics.csie.ncku.edu.tw/SDPF

ground truth. To measure the stability of rendered videos, we synthesize 12 videos by our method and the corresponding source video; and measure the differences between adjacent frames. The reason is that the source videos by themselves are temporally coherent; our results are rendered from the same image set with them but probably in different orders. Thus, we treat them as the standard to judge the stability degree of the results.

Given two adjacent frames F_{t-1} and F_t, difference of them is factorized as:

$$\mathcal{D}_{t\to t-1} = \| F_t - F_{t-1} \|, \tag{21}$$

here $t \in [0 \dots N_s]$, N_s is the total frames in the video. After that, we calculate the mean ($M_{\mathcal{D}}$) of $\mathcal{D}_{t\to t-1}$, and we compare them against those in the source video. On each single pair, $D_{t\to t-1}$ in our result might be higher than those in the ground truth, but it should be at an acceptable rate to guarantee there does not exist notable flicking artifact. This eventually affects the quality of the entire rendered clip. Therefore, we base on M_D to judge the stable quality of the results, i.e., the more tightly asymptotic to those in the source is better. We visualize an example of this manner in Fig.7. In this visualization, we choose a mutual frame between our rendered clip and the corresponding source video (i.e., frame 9). We can observe that the adjacent frames of frame 9 in the source (i.e., frame 11) and the adjacent candidates (i.e., frames 16 and 34) are in the same motion direction, but the heatmaps show that $D_{9\to 16}$ is closer to $D_{9\to 11}$ than $D_{9\to 34}$. This result reveals that the transition from frame 9 to 16 is better among two potential candidates than to frame 34, i.e., there could be a noticeable jumping artifact in the transition from frame 9 to 34 in this context. The average of $M_{\mathcal{D}}$ in this experiment is reported in Fig.8-(a). The analysis shows that the stable rates of our rendered clips are relatively close to those in the source video. There are three cases (e.g., clip A, C, and F) in which the stable rates are relatively higher than the source. However, they are still at acceptable rates.

6.3.2 ***E.2.*** *Degree of differences*

It is difficult to find a standard objective metric to measure the differences of the generated sequences compared to the input ones. Therefore, in this regard, we elaborate as follows.

We evaluate how different the rendered clips compared to the ground-truth by calculating the overlapping rate between them. To do this, we follow the well-known **F**-measure [16] as the evaluation metric. Previous works use this metric to measure the coherency of the rendered videos. The higher **F**-measure is, the higher the coherent rate will be. Reversely, our purpose is to measure how different they are. To avoid confusion, we denote this value as Δ_o. As a result, the smaller Δ_o represents the more difference. Note that the clips generated by our system may be of different lengths and also less than those of the source clip. Let G be the generated clip and T be the corresponding source clip, the precision P and recall R is defined based on the amount of temporal overlap between G and T, which are expressed as:

$$P = \frac{\delta}{d_G} \text{and } R = \frac{\delta}{d_T}, \tag{22}$$

where δ is the duration of overlap between G and T; d_G and d_T denotes the duration of clip G and T, respectively. Finally, Δ_o is formulated as:

$$\Delta_o = \frac{2 \times P \times R}{P + R} \times 100\%, \tag{23}$$

Quantitative results on this aspect are shown in Fig.8-(b). We can see that Δ_o of the testing data are relatively different from the ground-truth, especially on the clip D-Frog dance. It is worth pointing that the significant difference in this manner does not mean the clip is not stable. Inferring this clip in Fig.8-(a), the results reveal that the sequence this clip is still stable. There are three cases (e.g., clip H, I, and J) where the different rate is low. This implies that these results are not significantly different with the ground-truth. The reason is that these cases consist of linear motion in the entire video. Therefore, our method can only generate the smooth sequence as the ground-truth in such cases. In addition to these metrics, we conduct a user study to further learn about the human preferences on the visual quality of our results. Detail of the user study is described in the supplementary file.

In summary, if we denote the total number of linear motion segment in a certain source video is L, the quality on these two aspects of the rendered clips is defined as follows. The stability (M_D) is covariate with L and the differences of sequence (Δ_o) is inverse with L.

6.3.3 ***E.3.*** *Human perception-based evaluation*

In addition to the above measurements, we further use human visual perception on the sequences generated by our method. Seven testing clips with small Δ_o are used in this evaluation. We first collect two summarization per sequence. Then we recruit a group of 11 users rank (in five levels) the summarization based on how well they describe the clip according to two questions. The detail of this study is described in the supplementary file.

For each question, let s be the score if the i^{th} user rates for the corresponding level of s and N_s be the number of rating of s. We use the following equation to compute the rate of each summarization to each question, which reflects the users' opinions:

$$RA = \left(\sum_{s=1}^{5} s \times N_s \right) / (5 \times 11) \tag{24}$$

We then average RA of two summarizations for each sequence to define the users' opinion. Fig.8-(c) shows the statistics of users' preference. We can see that the scores of two questions are not extremely high but all of them are over the average degree (*i.e.,* in range of 0.62 - 0.79, and 64% is greater than 0.7). The results reveal that most users think the sequences generated by our work can tell meaningful stories.

6.4 Comparisons to prior works

We compare our system with some seminal works in this domain, including de Juan and Bodenheimer [3], Yu et al. [28], Yang et al. [26], and Morace et al. [17]. The different

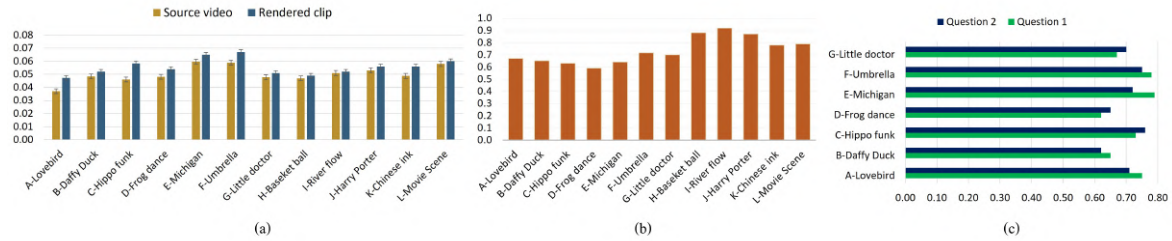

Fig. 8: Analysis results on (a) stability, (b) difference degree, and (c) human perception on our resequencing results.

aspects in comparisons are summarized in Table 1. In general, the early works [3, 28, 26] share the same two shortcomings: first, their mutual focus is the cartoon characters, and second, they need to do a pre-processing to extract the cartoon characters from the frames. Manifold method [17] is more general, i.e., it does not need such a pre-processing and thus, it is adaptable to cartoon scenes. However, they do not consider the motion direction as the other competitors [3, 28, 26] do, clips with dense motion directions are the major limitation in their system. In shape contrast, our approach has three major advantages. First, our system performs well on arbitrary input video scenes. Second, our system does not need any pre-processing. And third, our system is able to produce novel animations compared with those in the given video. The remainder of this subsection describes detailed comparison on each single competitor.

Fig.9 shows a qualitative comparison between our results and those in de Juan and Bodenheimer [3]. The pair of frames in (b) is mentioned as a bad transition in [3]. As a result, they have to insert inbetweens to obtain good transition. In contrast, our method automatically defines the adjacent frame with a smooth transition without refinement. It is observed that our transition in (a) is more plausible compared to (b).

Similar to our approach, Yang et al. [26] consider motion direction in transitions. The significant difference here is that they focus on cartoon characters. Gesture of characters needs to be extracted to define the similarity between frames (see Fig.7 in the supplementary file). Moreover, the motion direction feature (MDF) cannot accurately describe the gesture of a cartoon character. Thus, their approach is not effective to explore the challenging input. Reversely, our system gets knowledge from self-defined network to learn the similarity of images in terms of content correlation and embeds optical flow to maintain consistency in motion directions. Therefore, we advance not only in arbitrary input but also in accuracy.

Fig.10 shows the comparison with Morace et al. [17]. The source clip of this example consists of dense motions of fish and chinese ink, in which there exist several linear motion segments. As in our early discussion, since Morace et al. [17] do not consider the motion direction, there is significant abrupt motion in the regions masked in red rectangles. Fortunately, thanks to the constraints in our scheme, we resolve this phenomenon and obtain smooth transitions in the generated sequence. Another aspect makes [17]'s system suffer some limitations (*i.e.*, image content is complex) is that they use LPIPS metric to define the similarity of image pairs. This metric is learned by training a "small network" which is designed to predict perceptual judgement from distance pair and not originally designed for resequencing application. Besides, it takes approximately 5 seconds to compute on a pair. Therefore, the performance of [17] heavily relies on those in this model.

TABLE 1: Comparisons between our method and prior works

Methods	Pre-processing	New sequence?	Type of data
GCCS [3]	Yes	No	Cartoon characters
RCCS [26]	Yes	No	Cartoon characters
semi-MSL [28]	Yes	No	Cartoon characters
Manifold [17]	No	No	Cartoon scenes
Our method	No	Yes	Arbitrary scenes

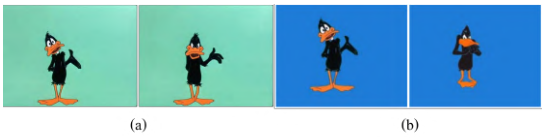

Fig. 9: Visualizes the differences in transition between our method (a) and [3] (b). Photos in (b) are obtained from [3].

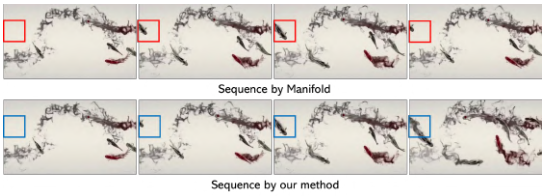

Fig. 10: Comparisons with Manifold sequence [17].

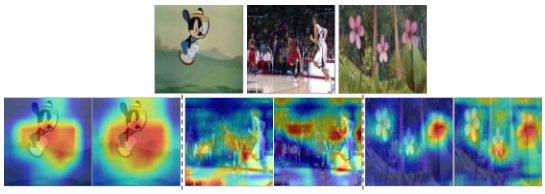

Fig. 11: Second row is the Grad-CAM visualization of the backbone VGG-19 (left) and our RSFNet (right).

Apart from the above visual comparisons, we quantitatively compare the quality of our results against those of prior work by two metrics M_D and Δ_o. We also use the data in Fig.5 in this comparison. And our competitor is Manifold [17] since the other three methods [3, 26, 28] focus on cartoon characters, and their results are not available

for a fair comparison. Meanwhile, Manifold [17]'s focus is comparable to ours, and the source code is provided by the authors. Table 2 presents the statistic results in this comparison. We can see that our method outperforms on the average of stability score. In terms of Δ_o, Manifold and ours have the comparable scores. However, we can see that their values of Δ_o are relatively equal, and the score in cartoon data (A-G) are higher than natural scenes (H-K). When we inspect M_D of data A-G, they are not at good stability degree. This reveals that Morace et al. [17] fail to either generate new sequences for cartoon data or produce smooth sequences with linear motion in natural scene data. Conversely, in our method, smaller Δ_o on cartoon data implies that it can explore new sequences. For the natural scene data with linear motion, higher Δ_o side by side with smaller M_D reveal that it can tolerate to avoid flip-flop phenomenon in such data.

6.5 Ablation Study

6.5.1 *A.1. Verify the effectiveness of RSFNet*

Our RSFNet is structured in the integration of a backbone and the proposed RFF module. Without RFF module, generated sequences include inconsistencies due to the lack of information on the features that are extracted from the backbone. We demonstrate the effectiveness of RFF module by removing it from our training. We show these ablation analyses in Fig.11. Here, we visualize the Grad-CAMs [22] of those obtained from our RSFNet with and without RFF module. The results show that with FRR module, our RSFNet has much larger attended regions. This enables our system to have more predictable results.

6.5.2 *A.2. Study on the impact of distance metric*

Performance of our resequencing system is affected by the feature correlation calculation. To analyze the influence of feature correlation on the quality of rendered sequences, we change the model to calculate the distance metric by a pure Euclidean distance calculation. That is, we remove the distance loss (*e.g.*, Eq. (4)) and use the Euclidean distance to measure the correlation in pairs of latent vectors. Fig.12 shows the contrast results. It is observed that Euclidean distance metric performs the correlation of the neighbors well. For example, we inspect on frame 5, which is highlighted in green rectangle. We can see that most similar frames are adjacent frames of this frame (*e.g.*, frame 4, 5, 7). Meanwhile, our distance is able to capture more (*e.g.*, frame 1, 2, 3, 4, 5, 12, 20). Therefore, if we directly use Euclidean as the distance metric, it prevents us from exploring new paths.

6.5.3 *A.3. Study on constraints in SDPF*

A.3.1. Directional distillation. This constraint is configured to detect the motion's property of a certain frame. As we mentioned in previous session, the "property" here is the linear motion. To verify the impact of this constraint (C_d) in the results, we remove it from the full procedure. That is, C_d is omitted from Eq.(18). Fig.13 shows the results of ablation analysis. In this example, we deliberately choose a frame (*i.e.*, frame 187) that belongs to such a linear motion segment to clearly reveal the influence of this constraint. After the first distillation, we define five candidates that have feature correlation to frame 187. Among them, frame 152 does not belong to LMS, meanwhile, the remainders are. In the remained candidates, frame 121 and 144 are in reverse direction motion with frame 187, and frame 193 is the same direction with frame 187. Without constraint C_d, frame 152 and 135 are selected as being adjacent with frame 187. Obviously, the flip-flop phenomenon will occur. Reversely, with constraint C_d, frame 193 is chosen. This result yields a reasonable transition.

TABLE 2: Comparisons on the quality of results

Methods	Ground-truth	Morace et al. [17]		Our method	
Testing data	M_D	M_D	Δ_o	M_D	Δ_o
A- Lovebird	0.037	0.056	0.78	0.047	0.67
B- Daffy Duck	0.048	0.067	0.84	0.052	0.64
C- Hippo funk	0.046	0.062	0.81	0.058	0.63
D- Frog dance	0.048	0.079	0.77	0.054	0.59
E- Michigan	0.060	0.062	0.79	0.065	0.64
F- Umbrella	0.059	0.083	0.69	0.067	0.72
G- Little doctor	0.048	0.073	0.73	0.051	0.70
H- Basket ball	0.047	0.081	0.58	0.049	0.88
I- River flow	0.051	0.075	0.75	0.052	0.92
J- Harry Porter	0.053	0.078	0.68	0.056	0.87
K- Chinese ink	0.049	0.052	0.73	0.056	0.78
L- Movie Scene	0.058	0.065	0.74	0.060	0.79
Average	0.050	0.069	0.74	0.055	0.73

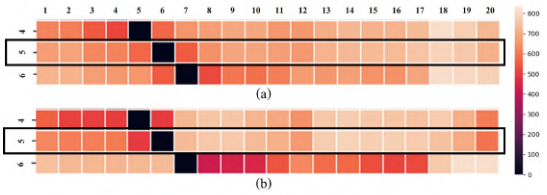

Fig. 12: Zoom-in the heat map of distance metric calculated by Euclidean distance (a) and our learning-based Euclidean distance (b). The experiment is conducted on segment with 20 frames of Daffy Duck clips. Entire heat maps could be seen in Fig.6 in the supplementary file.

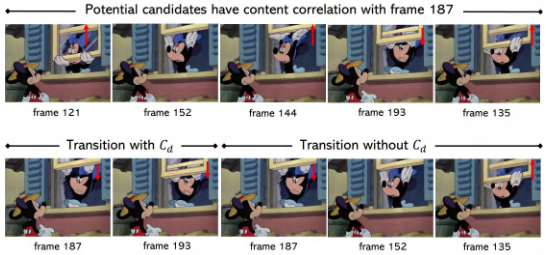

Fig. 13: Ablated results of constraint C_d. Five candidates in $\mathcal{S}_1$ are correlated with frame 187. Because frame 187 belongs to LMS, if $\mathcal{S}_1$ has LMS-frames, they will be considered to guarantee the coherency with frame 187. With $\mathbf{C}_t$, frame 193 is chosen. This is an LMS-frame, we can see the transition is visual smooth. In the contrast case, both frame 152 and 135 do not belong to LMS, exploring to these frames may cause artifacts.

A.3.2. Coherent distillation. Without this constraint, the motion in generated sequences can be realistic but may fail in temporal coherence. We measure this effect quantitatively by removing this constraint (i.e., C_t) from our proposed procedure. Fig.14 visualizes the ablated results in this aspect. It is observed that without C_t, the adjacent frame of frame 143 is frame 172. In the contrast case, it is frame 149. Although both frame 149 and 172 are the same direction motion with frame 143, the heat maps reveal that the differences from frame 143 to 172 is significant. This is the reason that causes the jumping transition in the rendered clips without C_t.

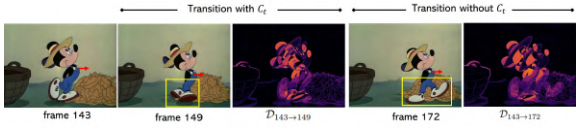

Fig. 14: Ablated results of constraint C_t

TABLE 3: Analysis of the quality on ablated results

Method	M_D	Δ_o
Ground-truth	0.0535	1
w/o C_t	0.118	0.783
w/o C_d	0.079	0.827
w/o RSFNet	0.082	0.731
w/o $\mathcal{L}_d$	0.281	0.816
Full configure	0.0648	0.706

In summary, we verify the effectiveness of RSFNet, distance loss, and two constraints (C_d, C_t) by testing on 12 videos in Fig.5. The analysis is shown in Table 3. From these results, we can conclude that the each constraint plays an important role for the stability of the rendered clips; the distance metric and RSFNet affect to the ability in exploring new sequences. Full configuration guarantees better quality results.

6.6 Limitations

In the cases that the input videos consist of subtle motion of landscape scenes (see the visualized sample here[3]), our method may not perform well. The failure phenomenon in such data is that the resultant sequence is quite short, i.e., approximately 20% of the total number of frames in the source video. We note here that these results are still smooth. The reason is our SDPF utilizes the temporal coherency or the velocity of motion in the source video to estimate the adjacent frame in each single pair of frames. In such subtle motion, the differences of the adjacent frames are small and the motion is looped. Therefore, our SDPF will early stop if the changes are relatively large to avoid cold transitions.

7 Conclusion

We propose a framework to create new animations from cartoon and natural videos with plausible smooth motion. We demonstrate that our novel path finding algorithm SDPF is especially useful to create the novel animations and control the consistency in generated clips. This gives our system the capability of resequencing various video contents with flexible sequences. We perceive that our system, on the one hand, will be useful as an aid to charge with generating new art in animation video, on the other hand, allows ordinary users with minimal expertise to explore compelling animations by reusing the existing video frames. Our results and evaluations show that the proposed scheme substantially advances prior works. For the drawback we mentioned in our limitation session, we plan to investigate such techniques to detect the bad transitions in the initial sequence and make it smooth by a novel algorithm rather than early stopping. Besides, this could be a possibility to develop a GAN-based network architecture to produce new images from existing image collection, and increase the diversity of the results in the near future.

3. http://graphics.csie.ncku.edu.tw/SDPF/Failure.mp4

Acknowledgements

We thank the reviewers for their insightful comments and suggestions. This work was supported in part by the National Science and Technology Council (under nos. 111-2221-E-006 -112 -MY3 and 110-2221-E-006-135-MY3), Republic of China (ROC), Taiwan.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning.

[2] Y. Chen, Y. Zhao, S. Li, W. Zuo, W. Jia, and X. Liu. Blind quality assessment for cartoon images. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(9):3282–3288, 2019.

[3] C. de Juan and B. Bodenheimer. Cartoon textures. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 267–276, 2004.

[4] E. Eng. Qt gui toolkit: Porting graphics to multiple platforms using a gui toolkit. *Linux Journal*, 1996(31es): 2–es, 1996.

[5] O. Fried, S. Avidan, and D. Cohen-Or. Patch2vec: Globally consistent image patch representation. In *Computer Graphics Forum*, volume 36, pages 183–194. Wiley Online Library, 2017.

[6] A. Gammerman, V. Vovk, and V. Vapnik. Learning by transduction. *arXiv preprint arXiv:1301.7375*, 2013.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[8] I. Goodfellow, Y. Bengio, and A. Courville. Deep feedforward networks. *Deep learning*, (1), 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[10] D. Holden, J. Saito, T. Komura, and T. Joyce. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*, pages 1–4. 2015.

[11] A. Hore and D. Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE, 2010.

[12] Y. Jiao, G. Shi, and T. D. Tran. Optical flow estimation via motion feature recovery. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2558–2562. IEEE, 2021.

[13] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[14] A. Khan and A. Storkey. Hamiltonian operator disentanglement of content and motion in image sequences. *arXiv preprint arXiv:2112.01641*, 2021.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] B. Mahasseni, M. Lam, and S. Todorovic. Unsupervised video summarization with adversarial lstm networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 202–211, 2017.

[17] C. C. Morace, T.-N.-H. Le, S.-Y. Yao, S.-W. Zhang, and T.-Y. Lee. Learning a perceptual manifold with deep features for animation video resequencing. *Multimedia Tools and Applications*, pages 1–21, 2022.

[18] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[19] M. Osadchy, Y. Le Cun, and M. L. Miller. Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research*, 8(5), 2007.

[20] A. Schödl and I. A. Essa. Controlled animation of video sprites. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 121–127, 2002.

[21] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498, 2000.

[22] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[24] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018.

[25] C. Xu, W. Qu, X. Xu, and X. Liu. Multi-scale flow-based occluding effect and content separation for cartoon animations. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[26] Y. Yang, Y. Zhuang, D. Tao, D. Xu, J. Yu, and J. Luo. Recognizing cartoon image gestures for retrieval and interactive cartoon clip synthesis. *IEEE transactions on circuits and systems for video technology*, 20(12):1745–1756, 2010.

[27] J. Yu, D. Tao, M. Wang, and J. Cheng. Semi-automatic cartoon generation by motion planning. *Multimedia systems*, 17(5):409–419, 2011.

[28] J. Yu, D. Liu, D. Tao, and H. S. Seah. On combining multiple features for cartoon character retrieval and clip synthesis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(5):1413–1427, 2012.

[29] J. Yu, D. Tao, J. Li, and J. Cheng. Semantic preserving distance metric learning and applications. *Information Sciences*, 281:674–686, 2014.

[30] J. Zhang, J. Yu, and D. Tao. Local deep-feature alignment for unsupervised dimension reduction. *IEEE transactions on image processing*, 27(5):2420–2432, 2018.

[31] L. Zhang, H. Huang, and H. Fu. Excol: An extract-and-complete layering approach to cartoon animation reusing. *IEEE transactions on visualization and computer graphics*, 18(7):1156–1169, 2011.

[32] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[33] S.-W. Zhang, C. C. Morace, T. Ngoc Hanh Le, C.-K. Yeh, S.-Y. Yao, S.-S. Lin, and T.-Y. Lee. Animation video resequencing with a convolutional autoencoder. In *SIGGRAPH Asia 2019 Posters*, pages 1–2. 2019.