Noname manuscript No.
(will be inserted by the editor)

Learning a Perceptual Manifold with Deep Features for Animation Video Resequencing

Charles C.Morace · Thi-Ngoc-Hanh Le · Sheng-Yi Yao · Shang-Wei Zhang · Tong-Yee Lee *

Received: DD Month YEAR / Accepted: DD Month YEAR

Abstract We propose a novel deep learning framework for animation video resequencing. Our system produces new video sequences by minimizing a perceptual distance of images from an existing animation video clip. To measure perceptual distance, we utilize the activations of convolutional neural networks and learn a perceptual distance by training these features on a small network with data comprised of human perceptual judgments. We show that with this perceptual metric and graph-based manifold learning techniques, our framework can produce new smooth and visually appealing animation video results for a variety of animation video styles. In contrast to previous work on animation video resequencing, the proposed framework applies to wide range of image styles and does not require hand-crafted feature extraction, background subtraction, or feature correspondence. In addition, we also show that our framework has applications to appealingly arrange unordered collections of images.

Keywords deep learning · deep features · manifold learning · animation video resequencing

1 Introduction

From its beginnings, animation has brought to life the creative potential of the human mind. It has developed into a dominant visual storytelling tool, and today there exists a wealth of archived animation video sequences created with both traditional and modern computer animation techniques. The visual style of animation video sequences is diverse, from stop-motion and three-dimensional photo-realistic renderings to cartoon illustrations and line-sketches. Although many techniques have been developed to ease the computer

All authors are with Department of Computer Science and Information Engineering, National Cheng-Kung University, Taiwan
*Corresponding author and his email: tonylee@mail.ncku.edu.tw

animation pipeline, production is still an arduous process, in large part to the complexity of digital characters, environments, and motion. The goal of this paper is to regenerate new video resequence for general and diverse animation video source. Animators can efficiently and interactively regenerate new animations according to their desire. Therefore, it reduces the complexity and timing/expense cost in creating animations.

Fig. 1: An example of the manifold topology and two animation video resequences generated from a collection of unordered animation images with the proposed method.

Previous works on animation resequencing [5, 31, 32] have focused on hand-craft feature extraction techniques to measure texture and shape similarity for a cartoon style animation and require background subtraction, segmentation, and other image processing techniques. Motivated by this, we propose a framework for animation video resequencing which can be applied to a broader range of image styles and does not require hand-craft feature extraction, background subtraction, or feature correspondence. Given the abundance of animation data which is currently available, our proposed resequencing framework generates new animations from an existing animation video clips.

The proposed framework learns a topological manifold of images where paths on the manifold represent smooth and visually plausible animation video frame sequences. To demonstrate our proposed framework to be general enough to handle a variety of animation video styles as source data, including photorealistic, non-photorealistic, styzied, and line-sketches we experimentally evaluate our method using different kinds of image and animation data. Also, given selected key-frames, our animation video resequencing method can create smooth and appealing user-controlled animation video.

In this paper, we utilize the activations of deep convolutional neural networks for smooth image sequencing and animation video resequencing. Given a pre-trained CNN and selected activation layers, we learn a perceptual similarity metric which reflects the perceptual judgments of humans. Then, from

a collection of images and their pair-wise perceptual distances, we generate smooth new animation video sequences by traversing paths and cycles in an estimated perceptual manifold. To the best of our knowledge, we are the first to apply deep features to the problem of animation video resequencing. Our technical contributions can be summarized in the following issues:

- We combine deep feature extraction and a perceptual similarity metric with a graph-based manifold learning technique to generate new and smooth animation video sequences.
- We implement our method with two well-known deep learning architectures, VGG [26], and AlexNet [14], and perform an experimental evaluation of the deep features learned by these architectures.
- We give a quantitative comparison of our animation video resequencing results with other image similarity metrics including L_2 distance in image space, L_2 distance of the bottleneck layer activations of a denoising autoencoder [29], and results obtained by traditional manifold learning techniques, locally Linear Embedding [20] and Isomap [28].
- We demonstrate that our method can facilitate several applications such as creating image layouts and video synthesis.
- To the best of our knowledge, this is the first work that shows our framework can generally handle different kinds of animation styles without extra efforts on different data.

We organize the remainder of this paper as follows. In section 2, most previous research in this sequencing domain are reviewed. In section 3, we introduce the overview of our proposed system. In section 4, the methods used in our system are described. In section 5, our experimental results and evaluations are presented and our additional applications follow. The conclusions and our future work are presented in the last section.

2 Related work

Smooth image sequencing is the key to producing a visually pleasing animation video. Most previous research divides this sequencing problem into two distinct steps. The first step is to establish a suitable distance measure for the similarity between the input images. The second step is to determine an optimal sequence according to the similarity measure defined in the first step. While the earliest work [24] used simple L_2 distance in the original image space as a similarity measure, more recent works have focused on feature extraction and dimension reduction techniques to measure higher-level features of shape [17], appearance [6], pose [19], and motion [8].

In the proposed method, we use the activations of deep convolutional neural networks for feature extraction and a metric inspired by the Learned Perceptual Image Patch Similarity (LPIPS) metric proposed by [33] to measure the perceptual distance of images. Although PSNR and SSIM [9] are mentioned as a well-known perceptual metrics, they are widely used to measure

the similarity between two images. The goal of our work is to learn a "perceptual manifold" with deep features, LPIPS metric is suitable for perceptual similarity across deep visual representations [33].

2.1 Feature Extraction and Non-linear Dimension Reduction

Patch2Vec [6] proposed a novel learning framework for image patch embedding, where an embedding is learned so that L_2 distance in the embedding space provides a useful measure for high-level features of texture dissimilarity. They trained a convolutional neural network (CNN), using a segmentation dataset and triplet loss function to map image patches having the same texture to points which are nearby in the embedding space while mapping image patches having other textures as far away as possible. The proposed method takes a similar approach by using deep convolutional networks to extract an optimal set of image features. However, instead of learning a perceptual metric with a triplet loss, we use the LPIPS metric trained on perceptual judgments of humans.

[19] proposed a view-independent Energy-Based Model that simultaneously detects faces and estimates head pose. They train a CNN to map images containing faces to points on a lower-dimensional face-manifold. After training, if the CNN maps an image to a point close to the manifold then its pitch, roll, and yaw are estimated by the position of the point projection onto the manifold. However, for general animation sequencing, there does not have a related energy-based model since the motion of a variety of characters and scenes must be estimated.

[8] used a convolutional autoencoder for learning a manifold of human motion. They trained their proposed CNN with a motion capture data consisting of time-series of human-poses, and each convolution layer of the CNN performs one-dimensional convolution over the temporal domain. Since their CNN is trained using motion capture data, it is not suitable for feature extraction of images.

In term of image synthesis, [3] developed a model for photographic image synthesis from pixelwise semantic layout. With their proposed method, they showed that photographic image can be synthesized from semantic layouts by a single feedforward network.[18] proposed new type of residual blocks an improved U-net to synthesize the photographic images. However, residual encoder-decoder networks may suffer from checkerboard artifacts.

[31, 30] described methods for cartoon retrieval and clip synthesis using a multi-feature distance function and a partially user-labeled database of cartoon characters to construct a lower dimensional feature space with a sparse transfer learning technique. Cartoon Textures [5] proposed a feature-distance based on the shape [10], appearance, and the temporal ordering of an input cartoon sequence, and then utilize the manifold learning technique Spatio-Temporal Isomap (ST-Isomap) to recover a lower-dimensional embedding of the input. Unlike the proposed method, ST-Isomap requires an initial ordering

for the input images and thus does not apply to unordered collections of images. Moreover, ST-Isomap, and traditional manifold learning techniques like Locally Linear Embedding (LLE) [20] and Laplacian Eigen Maps (LEM) [2] required predetermined parameters including the dimension of the manifold and the number of neighbors of each input image. These parameters require fine-tuning for each collection of input images. Cartoon Textures [5] and the methods proposed by [31, 30] all use hand-crafted feature-extraction methods specific to cartoon images and require much preprocessing including segmentation of a character and pairwise computation of the Hausdorff distance. Furthermore, the Cartoon Textures [5] method relied on knowledge of the ordering of the original input sequence and the methods of [31, 30] required user labeled data to construct the embedding to measure image similarity. In contrast, the method proposed in this paper, requires no user-labeling, no segmentation, and works with a variety of image styles.

2.2 Sequential Ordering of Images

Determining a sequential ordering of images is usually posed as a path-finding problem in a weighted graph, where nodes correspond to images and transition costs are based on the image dissimilarity measure and possibly other criteria such as path smoothness, temporal ordering of the input sequence, or user-control. Previous work considers transition costs in complete graphs and nearest neighbor graphs.

Video textures [24] is a video-based rendering technique. They apply Q-learning [12] to generate a video sequence of arbitrary length with similar dynamics to the input video. The method produces convincing results when the input video has repetitive motion or unstructured stochastic motion but will fail for complex structured motion like full body human motion. This limitation is a consequence of using L_2 distance on the raw pixels of the images which cannot sufficiently measure similarity in high-level features of motion. To ameliorate this issue, Shödl and Essa [22, 23] trained a linear binary-classifier from manually labeled training data based on six hand-crafted features. Images deemed unacceptable by the linear classifier are not considered for transitions, and for the other images, the similarity measure is the linear classifying function. Shödl and Essa [22] applied a beam-search technique to obtain the optimal sequence. To improve the sequencing results of their beam search, Shödl and Essa [23] considered the temporal information of the original input sequence in the transition cost and adopt a greedy hill-climbing optimization, which starts with a random image sequence and iteratively changes subsequences which lower the total path cost. Contrast to these previous methods, our method applies to unordered and unlabeled collections of images.

Unlike the previously discussed techniques which sequence photo-realistic images, Cartoon textures [5] and the works by [31, 30] synthesize cartoon animations. [30] used a greedy method, choosing a random cartoon image as the first frame, and then choosing the most similar image, measured in the

low dimensional subspace, for each subsequent frame. Cartoon Textures and [31] synthesized new animations by finding the shortest paths in a graph constructed by the ST-Isomap and Isomap manifold learning algorithms, respectively. It is important to note that traditional manifold learning algorithms such as ST-Isomap and Isomap do not construct the graph automatically and require defined neighbor relations for the input data and the dimension of the embeddings beforehand, which is difficult to estimate. In our framework, we do not compute an explicit embedding and can adapt to different CNN architectures. Also, we automatically determined neighbor relations by minimizing the perceptual distance of the input data. Therefore, our method does not require fine-tuning for each input collection like traditional manifold learning techniques.

3 System overview

We outline the system overview of our proposed framework in Figure 2. The input to our system is a collection of unordered images and a pre-trained CNN network $\mathcal{F}$. The input collection could be from an unstructured collection of images or an existing video. In the cases that they are from an unstructured collection, we resize them to the same size before feeding them to the system. The CNN serves as an image feature extractor, and we learn a "perceptual distance" by training another neural network $\mathcal{G}$ on a dataset comprised of perceptual judgments of humans. In our implementation, we test the activations of AlexNet and VGG trained for image classification. However, features extracted by other CNNs trained for tasks other than classification are also useful for measuring perceptual distance [33] and could be incorporated into our system. So, after extracting the deep features from each image, we compute the pairwise perceptual distance of each image in the input collection using the LPIPS metric proposed by [33]. Once the perceptual distance is learned, the proposed system can create:

- a path sequence which uses all images in the collection given start and terminal frames;
- a cycle animation which uses all or some subset of images in the collection;
- or a path animation with smooth in-between images given a set of keyframes.

For some collections of images, it may not be possible to obtain a smooth animation sequence using every image in the collection. For input collections obtained from densely sampled videos, we can assume there exists at least one smooth sequence which uses all images in the input collection, i.e., the original animation. However, if the images come from sparsely sampling an existing animation video or from an unordered collection of images, it may not be possible to resequence all of the images into a smooth sequence. Therefore, we also detect and prune outliner images from the input collection by fitting the perceptual distance of nearest neighbors to an optimal probability distribution using maximum likelihood estimation.

Then, from the pairwise perceptual distances, we construct a complete graph where each image corresponds to a node, and the weight of each edge is equal to the perceptual distance of adjacent images. To generate an optimal animation sequence through all of the input images, we compute the shortest Hamiltonian path from starting, and terminal frames assigned by the user or generate a looping animation by computing shortest tours. In key-frame path finding, we compute a minimal spanning tree (MST) of the complete graph and generate animation sequences by traversing paths in the MST. Figure 2 shows an overview of the system.

Section 4.1 describes the feature extraction details, section 4.2 describes the procedure for computing perceptual distance, section 4.3 describes the method for automatic outliner detection and removal, and section 4.4 describes animation resequencing.

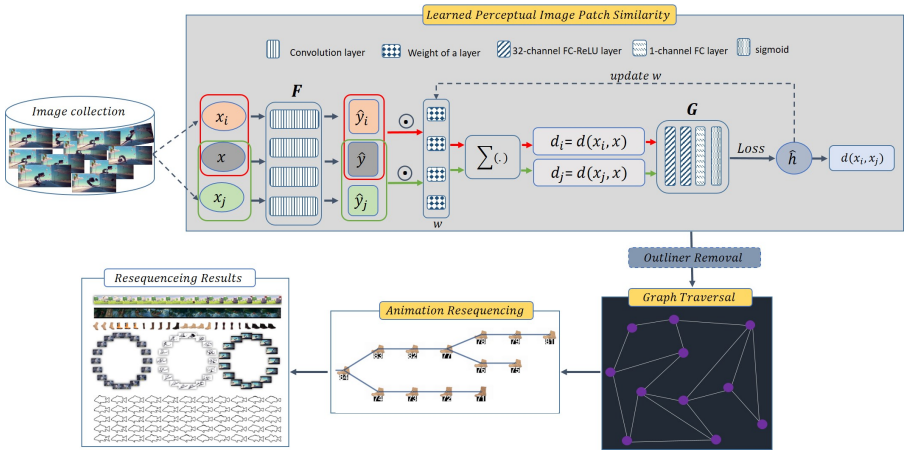


Fig. 2: System overview of the proposed method.

4 Method

4.1 Deep feature extraction

To compute the perceptual distance, we first extract features with a trained convolutional neural network (CNN) $\mathcal{F}$. Image features for an image $x_i \in X$ are a set of activations, $\{\hat{y}_i^l \in \mathbb{R}^{w_l h_l c_l}\}_{l=1}^{L}$, obtained from the activations of selected layers after applying the image x_i to the CNN $\mathcal{F}$, where L is the total number of selected activation layers, and w_l, h_l, and c_l are the dimensions of a selected activation layer l's output. Thus, the size of features and the speed at which that are extracted depends on the architecture of the network architecture of $\mathcal{F}$.

In our implementation, we test two off-the-shelf networks, namely, VGG [26] and AlexNet [14]. Both two networks are trained on ImageNet dataset [21] and have excellent performance in image classification, object detection, etc. Besides, there are several deep learning-based studies in this problem domain adopting them to extract features in their system. Thus, these two pre-trained networks are suitable to adequately extract features from images. In each backbone, we remove the later layers, fully connected, and utilize the first five activation layers as feature extractors. While VGG and AlexNet are trained for classification of natural images, our experimental results show that their activations are also useful for re-sequencing non-photorealistic image styles used in animation.

4.2 Perceptual Distance

In this section, we briefly summarize the perceptual distance used for generating animation sequences. Once images are encoded into feature space by the network $\mathcal{F}$, we adopt LPIPS metric proposed by [33] to learn the perceptual distance of images from their features. The procedure of this algorithm is presented in pseudocode shown in algorithm 1. Given a trained convolutional network $\mathcal{F}$ and two images x_i and x_j, we extract the activations from L selected layers and unit-normalize in the channel dimension to obtain features $\hat{y}_i^l$ and $\hat{y}_j^l$ for each selected layer $l \in \{1, \ldots, L\}$. To compute the perceptual distance $d(x_i, x_j)$, we scale the difference of activations element-wise by learned "perceptual calibration" weight tensors w_l, compute the L_2 norm, average spatially, and sum over all layers. This distance is expressed as follows.

$$d(x_i, x_j) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \parallel w_l \odot (\hat{y}_{ihw}^l - \hat{y}_{jhw}^l \parallel^2, \tag{1}$$

where $\odot$ denotes scale operator; H and W is the height and width of images in the given image collection, respectively; h and w is the channel dimension of features at a certain layer l; w_l denotes the calibration weights at layer l which are learned by a "*small network*".

Small network, denoted by $\mathcal{G}$, is designed to predict perceptual judgement from distance pairs. Authors in [33] showed that perceptual similarity is an emergent property shared across deep visual representations. In essence, using $w_l = 1$ for all layers in network $\mathcal{F}$ is equivalent to compute cosine distance. To obtain perceptual similarity, the weights w_l are learned by training network $\mathcal{G}$ from distance pairs. Given a reference image x and two distorted images x_0, x_1 of x. A distance pair (d_0, d_1) is obtained by using equation 1, where $d_0 = (x, x_0)$, $d_1 = (x, x_1)$. The judgement $h \in (0, 1)$ is determined based on the proportion of humans that perceived the image x to be more similar to x_1 than x_0 and the weights w_l are obtained by minimizing a cross entropy loss function which is formulated as:

$$\begin{aligned}\mathcal{L}(x, x_0, x_1, h) = &-h \log \mathcal{G}\big(d(x, x_0), d(x, x_1)\big) \\ &- (1-h) \log \big(1 - \mathcal{G}\big(d(x, x_0), d(x, x_1)\big)\big)\end{aligned} \tag{2}$$

The architecture of network $\mathcal{G}$ uses 32-channel FC-ReLU layers, followed by a 1-channel FC layer and a sigmoid [33], as we depict in Figure 2. For each pair of images x_i and x_j in the input collection, we use the LPIPS metric in equation 1 to compute a "perceptual distance".

In section 5.2 we show a comparative analysis of animation results obtained with the LPIPS metric and other image similarity metrics, including L_2 in image space, L_2 in a denoising autoencoder's bottleneck activation feature space, two traditional manifold learning methods' embeddings, and the cosine distance of the same deep features used with LPIPS.

Algorithm 1 Algorithm of LPIPS

Input: pair of images (x_i, x_j), network $\mathcal{F}$ with L layers, network $\mathcal{G}$

Taking a reference image x corresponding to x_i and x_j.
for each layer $l = 1, \ldots, L$ **do**
    Calculate $\hat{y}_i^l \leftarrow \mathcal{F}^l(x_i)$, $\hat{y}_j^l \leftarrow \mathcal{F}^l(x_j)$, $\hat{y}^l \leftarrow \mathcal{F}^l(x)$.
    Scale activation channel-wise by initial weight $w_l = 1$.
end for
Compute $d_i \leftarrow d(\hat{y}_i, \hat{y}, w_l)$, $d_j \leftarrow d(\hat{y}_j, \hat{y}, w_l)$ by eq.(1)
for each iteration **do**
    Feed d_i, d_j to $\mathcal{G}$
    Scale activation channel-wise by weight w_l.
    Compute $d_i \leftarrow d(\hat{y}_i, \hat{y}, w_l)$, $d_j \leftarrow d(\hat{y}_j, \hat{y}, w_l)$ by eq.(1)
    Update w_l by loss $\mathcal{L}(.)$ in eq.(2).
end for

Output: $d(x_i, x_j)$.

4.3 Outliner Detection and Removal

Outliner detection and removal is an optional part of our proposed system which serves two goals. One goal is to give a user more control over the length of the re-sequenced animation and the other is to remove frames which would result in a non-smooth re-sequenced animation. Outliner removal is particularly important in the images are from an unstructured collection of images, rather than from an existing video.

Images which have a large perceptual distance from all other images in the input collection may negatively affect the smoothness of the resequenced animation result. Thus to maintain the smoothness of the generated sequence, we remove outliner images which have a large perceptual distance from their nearest neighbors. A naive approach would be to simply threshold the perceptual distance of nearest neighbors in the complete graph. However, a constant

threshold value cannot adapt to disparate input data. Therefore, we fit the perceptual distance of nearest neighbors to a probability distribution to detect and remove outliner images.

Let X_i be a random variable equal to the average perceptual distance of image x_i and its nearest neighbors $x_{i(j)}$ for $j \in 1, \ldots, K$.

$$X_i = \frac{1}{K} \sum_{j-1}^{K} d(x_i, x_{i(j)}), \tag{3}$$

In our implement, we choose the number of nearest neighbors to be $K = 5$.

To find the most likely distribution, we estimate the parameters of the generalized gamma probability distribution function [27] given the samples X_i for $i \in 1, \ldots, N$ where N is the total number of images in the input collection. The generalized gamma probability density is defined as,

$$f(x; \alpha, \beta, \gamma, \mu) \propto= \begin{cases} (x-\mu)^{\alpha\gamma-1} exp\big(-\big(\frac{x-\mu}{\beta}\big)^{\gamma}\big), & \text{if } x > \mu \\ 0, & \text{if } x < \mu, \end{cases} \tag{4}$$

The parameters α, β, γ, and μ are obtained with maximum likelihood estimation, i.e., by maximizing the log-likelihood function of f given the random samples X_i. Once the parameters are found, we calculate the 0.9 quantile value T as a threshold and remove the $i-th$ column and $i-th$ row from the original distance matrix for any $X_i > T$ and update the complete graph. In our implementation we choose the number of nearest neighbors to be $K = 5$.

We choose the generalized gamma function as a distribution because of its flexibility. We tested the Normal and Beta distributions but found that the generalized gamma distribution produced better fits to the sample histograms than other distribution models.

4.4 Animation Resequencing

Once perceptual distances between images in the image collection are computed, we construct a complete graph $\mathbf{G}(V, E)$ where a node $v_i \in V$ corresponds to image x_i and the weight of an edge $e_{ij} \in E$ is $d(x_i, x_j)$ computed by Equation 1. Given the index of starting frame (s) and terminal frame (t) from the user, the new video sequencing is generated by finding the minimum path from s to t on $\mathbf{G}$. This algorithm returns a set of indices of images which are used to reconstruct a new sequence. The pseudo code of this procedure is presented in Algorithm 2.

Initially, we view the complete graph $\mathbf{G}$ as a crude approximation of a perceptual manifold. Traversing the complete graph would allow for large "perceptual jumps" since each pair of images are adjacent and a large perceptual distance of adjacent frames would result in an unsmooth animation sequence. To improve our estimation of the perception manifold, we find subgraphs which prune large edges from the complete graph and generate animations by traversing a modified graph structure.

To find the minimum path, our current re-sequencing system considers three different types of algorithms, the shortest Hamiltonian path [4], the shortest Hamiltonian cycle [4], and the minimum spanning tree [4]. In the following subsections, we formalize these problems and give additional details and justification for these methods.

Algorithm 2 Procedure of generating new sequence

Input: A collection of unordered images,
index of starting frame (s),
index of terminal frame (t).

1: Initialize a distance matrix $\mathcal{A}$
2: **for** each pair of images (x_i, x_j) **do**
3: Compute perceptual distance $d = d(x_i, x_j)$ by Eq.(1).
4: Index d.
5: Add d to matrix $\mathcal{A}$.
6: **end for**
7: Detect and Remove outliner images by Eq.(3) and Eq.(4).
8: Update matrix $\mathcal{A}$ after removing outliner images.
9: Construct a complete graph $G(V, E)$ from $\mathcal{A}$.
10: Find the minimum path from s to t on G.

Output: Set of indices of images in the minimum path.

4.4.1 Shortest Hamiltonian Path Sequence

To generate an optimal animation sequence through all the input images, we compute the shortest path from starting and terminal frames which are assigned by the user. To achieve this, we adopt the shortest Hamiltonian path [4].

For an input collection of images with m frames, the shortest Hamiltonian path in the complete graph is the permutation of image ψ in the set of images Ψ which minimizes the total perceptual distance between adjacent frames:

$$min_{\psi \in \Psi} \sum_{i=1}^{m-1} d\big(x_{\psi(i)}, x_{\psi(i+1)}\big), \tag{5}$$

Optionally, a user can add constraints to the set of permutations so that the first frame in the Hamiltonian path has index $\psi(1) = s$ and the terminal frame has index $\psi(m) = t \neq s$. For input animations which do not contain cyclic motion, this method can be used to reconstruct the original animation sequence given $s = 1$ and $t = m$.

4.4.2 Shortest Hamiltonian Cycle Sequence

To compute a cyclic animation sequence, we compute the shortest Hamiltonian cycle [4] of the complete graph. Finding the shortest Hamiltonian cycle is equivalent to the well-known traveling salesman problem, and corresponds to

a cyclic permutation of images ψ which minimizes the total perceptual distance between adjacent frames in set Ψ:

$$min_{\psi \in \Psi} d\big(x_{\psi(1)}, x_{\psi(m)}\big) + \sum_{i=1}^{m-1} d\big(x_{\psi(i)}, x_{\psi(i+1)}\big), \tag{6}$$

The Hamiltonian cycle can generate looping sequences with continuously smooth motion which can be chained together to create a looping animations of arbitrary length. We show the result from uniformly sampling the shortest Hamiltonian Cycle Sequence in Figure 3.

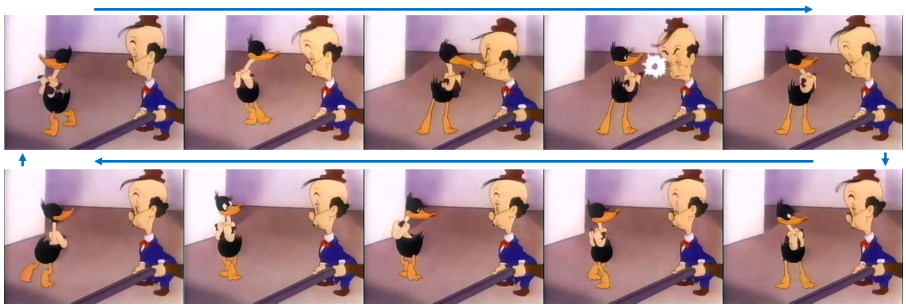


Fig. 3: Results from uniformly sampling the shortest Hamiltonian cycle sequence. In this example, the sequence is generated from six images. The blue arrows navigate the starting frame (top-left frame)to the ending frame (bottom-left frame) in a cycle.

4.4.3 Key-frame Path-finding

In key-frame path-finding, we use paths in the minimum spanning tree (MST) to return temporally-coherent in-between frames given a set of key-frames by the user. Animators typically choose key-frames as the beginning and end points of a temporally coherent transition. Thus, for in-between sequencing, we would like a high level of confidence that in-between images remain close to the perceptual manifold and we hope to return a sequence of many temporally coherent images to the user.

Since all of the perceptual distances are positive, the MST is the minimum-distance subgraph which connects all of the images. Therefore our proposed method produces in-between frames by traversing the path from one key-frame node to another along an MST. The paths connecting key-frame nodes in an MST are well suited for finding in-between images since the distance between nodes is relatively small which gives us a higher level of confidence that the in-between images are temporally coherent. The MST also has the advantage of having the minimal set of edges for a path-connected graph containing each image in the input collection, thus reducing both the time and space complexity of path-finding.

With this method, users may create animations from any number of key-frames by computing paths between consecutive key-frames and combining the results. The user can also view a 2D linear embedding of the MST to see an overview of the entire dataset and help drive their decisions in key-frame selection. Figure 1 shows how a user can use the MST's 2D linear embedding to choose key-frames and view the sequence of in-between images.

5 Experimental Results

5.1 Implementation Details

All our experiments are conducted on a PC with Intel Core i7 2.5GHz, 16GB RAM. The perceptual judgments used to train in network $\mathcal{G}$ are obtained from publicly available *Two Alternative Forced Choice* (2AFC) dataset collected by [33]. The AFC evaluators were given 3 image patches (1 reference + 2 distorted) and asked to select which of the distorted was "closer" to the reference. Additional details about training and a thorough evaluation of the effectiveness of the LPIPS metric can be found in [33].

In general, verifying if a sequence is a shortest Hamiltonian path or shortest Hamiltonian cycle is an NP-complete problem [7]. For a given input collection with m-frames, verification requires an exhaustive search of all $m!$ permutations of the set $\{1, \ldots, m\}$. For larger image collections, finding an exact solution quickly becomes infeasible. However, for a complete graph, the existence of a Hamiltonian path and Hamiltonian cycle is guaranteed, and many polynomial approximation algorithms with bounded error have been proposed [16]. In our implementation we use commercial software Mathematica [11] to solve the shortest Hamiltonian path and Hamiltonian cycle problems. The MST, on the other hand, can be computed very efficiently using a greedy method such as Kruskal's algorithm [15].

5.2 Animation Resequencing Results

In this section, we show some representative results generated with our framework. To generate new animation sequences we collected test data by sampling frames from animation videos, extracting deep features from the first five activation layers of VGG, applying the LPIPS metric to each pair of images in the input collection, and resequencing the animations with the proposed outliner and graph traversal methods.

Figure 4 shows uniformly sampled frames of sequences generated by computing the Hamiltonian path. We show the full sequences in our supplementary video as well as a comparison with animation results using features extracted from AlexNet and other feature extraction methods described in section 5.3.

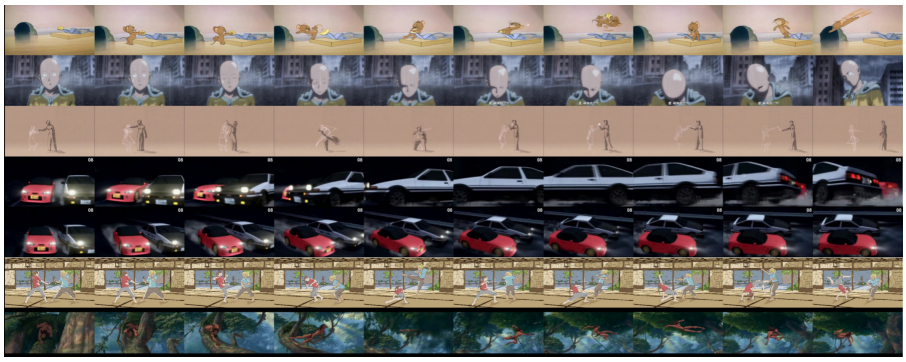


Fig. 4: Results from uniformly sampling the shortest Hamiltonian path sequence.

The Hamiltonian path often reconstructs the original animation if the user gives the initial and terminal frames as constraints. However, by using the outliner removal and other frame constraints it is possible to create new motion sequences which do not resemble the original input.

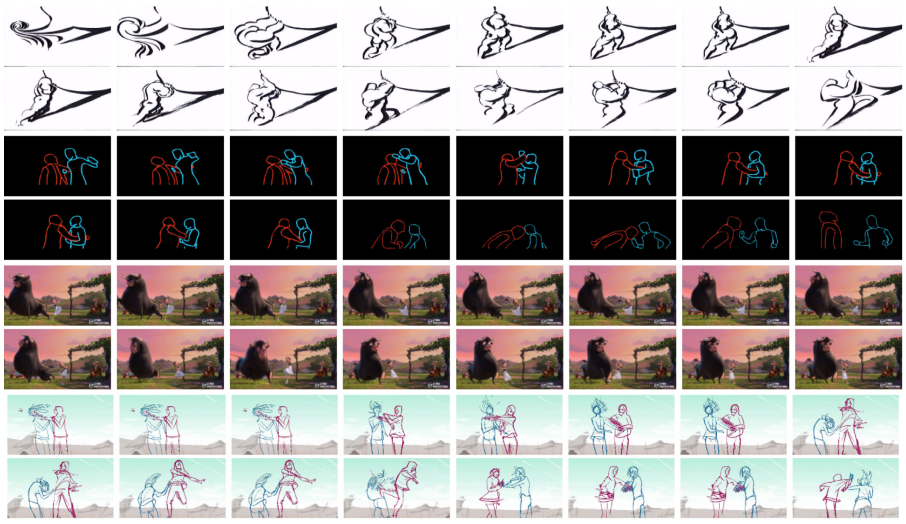


Fig. 5: Results of our proposed key-frame method. The first and last frames are selected by a user and the in-between frames are generated by traversing the minimum spanning tree. Please explore the supplementary video in our project website for better visualization.

Our results show that Hamiltonian cycles can generate novel looping motion. A Hamiltonian cycle can create a pleasing looping effect, even when the input images come from an animation that is not originally a loop. Figure 3 shows a uniform sampling of the Hamiltonian cycle results. See our supplementary material and video for additional results. Although we obtained many

pleasing looking results, there are input data where the Hamiltonian cycle cannot immediately produce a smooth looping sequence. The proposed outliner removal method can improve outcomes in some of these cases. Another option is manually removing outliner images. One advantage of our system is that image layouts of the Hamiltonian cycle make it much easier to visually detect outliners and smooth subsequences.

Figure 5 shows key-frame results generated with the proposed method. To create these results, we examined the MSTs to guide key-frame selection and return six in-between images. In general, the user cannot directly control the number of in-between images returned for arbitrary key-frame selection. However, using the linear embeddings of the MST for visualization provides a useful way to select key-frames that produce the desired number of in-betweens. Figure 1 shows a portion of an MST's 2D linear embedding and our supplementary material shows the full versions for all results shown in the paper.

The in-between frames generated by the proposed method are typically temporally-coherent for key-frames which have relatively short path distance in the MST, but as the path distance between key-frame nodes increase, so does the probability of unreasonable in-betweens. In practice, we do not consider this a significant draw-back since choosing additional intermediate key-frames can avoid this issue.

In addition to the results presented here, please see our supplementary material and video for additional results and comparisons which are available on our project website:
http://graphics.csie.ncku.edu.tw/ManifoldAnimationSequence.

5.3 Quantitative Results

In order to quantitatively evaluate the effectiveness of the proposed method, we compare resequencing results using LPIPS metric against other image similarity metrics previously applied to video and cartoon animation resequencing. We compare the LPIPS metrics against the following metrics:

- L_2 distance in image space;
- L_2 on the deep features of the bottleneck layer of a custom denoising autoencoder;
- L_2 distance on the embeddings learned by traditional manifold learning LLE and Isomap;
- cosine distance of deep features of VGG and AlexNet.

The first three metrics have been applied in previous research in video and cartoon animation resequencing [5, 24, 34]. To the best of our knowledge the activations of VGG and AlexNet have not been previously applied to animation resequencing. However, we also compare these features to measure the effect of the learned perceptual weights used by the LPIPS metric. Each compared image metric is briefly described in subsection 5.3.1. In subsection 5.3.2, we present how we setup animation reconstruction for this quantitative evaluation.

5.3.1 Compared image Metrics

L_2 Distance in Image Space. To compute L_2 distance in image space, we represent each RGB image as a flat vector $\hat{x} \in \mathbb{R}^{w \times h \times c}$, where the our test images have $c = 3$ color channels and a width $w = 320$ and height $h = 180$ pixels:

$$d(x_i, x_j) = \sqrt{\sum_{c,w,h} \left(\hat{x}_{ic,w,h} - \hat{x}_{jc,w,h}\right)^2}, \tag{7}$$

L_2 Distance in DAE Bottleneck Activation Space. We compare our results with a custom denoising autoencoder (DAE) [29]. An autoencoder is a kind of neural network divided into two parts, an encoder and a decoder. We consider the output of the bottleneck layer as the features that the encoder retrieves and encodes from the input. The encoding network of our DAE reduces the dimension of each image x to a lower dimensional latent vector $\hat{y} \in \mathbb{R}^{w \times h \times c}$, where the latent space has $c = 500$ channel dimensions and a width $w = 16$ and height $h = 9$ spatial dimensions. To measure image similarity we use L_2 distance on the activations of the bottleneck layer as below:

$$d(x_i, x_j) = \sqrt{\sum_{c,w,h} \left(\hat{y}_{ic,w,h} - \hat{y}_{jc,w,h}\right)^2} \tag{8}$$

The architecture and training procedure of the denoising autoencoder is described in the appendix.

L_2 Distance in LLE and Isomap Embeddings. The traditional manifold learning techniques LLE and Isomap map a set of images $X = \{x_i\}_{i=1}^m$ to a set of low dimensional vectors $Y = \{y_i\}_{i=1}^m$ where $y \in \mathbb{R}^d$ and $d \in \{1, \ldots, m-1\}$ is the dimension of the embedding which must be specified by the user. In addition to the dimension of the embedding, the neighbors of each image must be specified.

In our comparison with traditional manifold learning, we test both LLE and Isomap with all parameters for the number of nearest neighbors, $k \in \{2, \ldots, 10\}$ and embedding dimensions, $d \in \{2, \ldots, 20\}$, with L_2 distance on the learned embedding vectors $Y^{k,d} = \{y_i^{k,d}\}_{i=1}^m$.

$$d(x_i, x_j; k, d) = \sqrt{\sum_{c,w,h} \left(y_i^{k,d} - y_j^{k,d}\right)^2}, \tag{9}$$

Cosine Distance in VGG and AlexNet Activation Space. Lastly we compare the cosine distance in the channel dimension of the same deep features used with the LPIPS metric described in Section 4.1.

$$d(x_i, x_j) = \sum_l (1 - \frac{1}{H_l W_l} \sum_{h,w} \hat{y}_{ihw}^l . \hat{y}_{jhw}^l), \tag{10}$$

5.3.2 Setup Animation Reconstruction Experiment

We collected 39 animations, between 24 and 230 frames in length and with various images styles and content. Figure 6 shows example frames of the animation video used in this experiment. For each animation, the image order is randomly shuffled, and we attempt to reconstruct the original sequence by computing a new sequence which starts and ends with the same frames as the original animations' initial and terminal frames. We then minimize the total image dissimilarities between adjacent frames.

For each image metric, we repeat the following procedure for each animation in our test set:

1. compute the complete weighted graph of images with the appropriate distance function as edge weights;
2. compute a Hamiltonian path from the node corresponding to the first frame to the node corresponding to the last frame;
3. calculate the normalized Kendall tau distance [13] of the original sequence and the sequence generated by the Hamiltonian path.

For an animation with m frames, let $A = \langle X_i \rangle_{i=1}^{m}$ denote the original sequence of frames X_i and let $H(A) = \langle X_{\psi(i)} \rangle_{i=1}^{m}$ denote the shortest Hamiltonian path from frame-1 to frame-m. Then the normalized Kendall tau distance between the original sequence and $H(A)$ is defined as:

$$\kappa_\tau\big(H(A)\big) = \frac{2}{m(m-1)} \sum_{i=1}^{m} \sum_{j=i+1}^{m} \bar{\kappa}_\tau\big(\psi(i), \psi(j)\big), \tag{11}$$

and

$$\bar{\kappa}_\tau(i,j) = \begin{cases} 0, & \text{if } i < j \\ 1, & \text{if } i > j, \end{cases} \tag{12}$$

The Kendall tau distance measures the number of discordant pairs in the Hamiltonian path sequence. It is normalized so that the distance $\bar{\kappa}_\tau(H(A)) \in [0,1]$ for any number of frames in the animation clip. A Hamiltonian path sequence with the same order as the test animation has zero distance and a sequence with the reverse order has a distance of one, thus the Normalized Kendall tau distance also gives a measure of rank correlation.

To consider an input animation as ground truth for a Hamiltonian path, it must not contain cyclic motion. Thus we visually inspect each animation and remove examples with cyclic motion. Additionally, we removed trivial cases where all test methods perfectly reconstruct the animation.

We use 4 aforementioned compared image metrics to test on 39 reconstruct animations. The comparison of everage Kendal tau distance is shown in Figure 7. In the case of traditional manifold learning algorithms LLE and Isomap, we test all parameters $k \in \{2, \ldots, 10\}$ and $d \in \{2, \ldots, 20\}$ and select the lowest reconstruction error for each test animation. Besides, we compare the normalized Kendall tau distance for the reconstruction of 39 animations with different image similarity metrics in Figure 8 and a box and whisker chart

Fig. 6: Examples frames from the animation reconstruction test set.

for all test animations and test methods in Figure 9. The results show that, on average, using features from VGG or AlexNet with the LPIPS metric produce Hamiltonian Path sequences which are closer to the original sequence than all other test metrics. While all similarity metrics have relatively small reconstruction errors, the bottleneck activations of the denoising autoencoder has the worst results. This may be due to the fact that the DAE is trained solely on japanese manga style images. More diverse training data could possibly improve the results of the DAE's bottleneck features.

The traditional manifold learning technique LLE outperforms Isomap and slightly outperforms the cosine distance of the extracted deep features of VGG and Alexnet. However the experiment was slightly biased towards traditional manifold learning since each animation was tested with 171 different parameter settings and only the single best result was counted towards the average reconstruction error. Despite this bias, LPIPS with VGG features and AlexNet features performed better than LLE. This comparison is shown in Figure 10. We can see that LLE solely has lower error rate in few cases. In most of the tested animations, LPIPS with VGG features and AlexNet features have lower error rate in reconstruction sequences. In addition, finding good parameter settings for the k nearest neighbors and embedding dimension d can be difficult with traditional manifold learning algorithms and require different settings for different input images. Our experimental results show that animations reconstructed with features extracted from VGG or AlexNet, and the LPIPS metric have lower average reconstruction error than the animations reconstructed

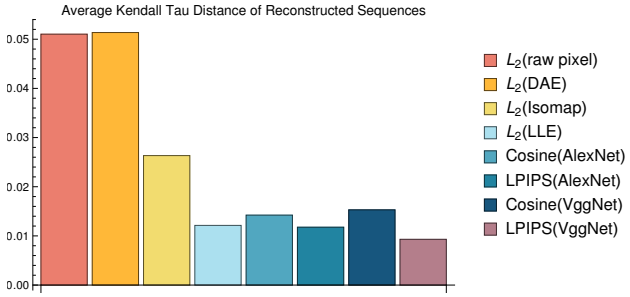


Fig. 7: Comparison of the average Kendall tau distance (equation 12) for 39 reconstructed animations. We test 8 different distance measures, pairwise L_2 distance of the raw image pixels, the bottleneck layer of a denoising autoencoder, and the low dimensional embeddings learned by Isomap and LLE; cosine distance and LPIPS of the activations of selected layers of VGG and AlexNet.

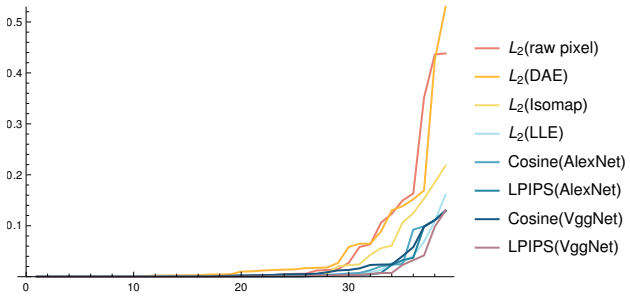


Fig. 8: A comparison of the normalized Kendall tau distance for the reconstruction of 39 animations with different image similarity metrics (independently sorted). Obviously, LPIPS(VggNet) and LPIPS(AlexNet) are the lowest in this comparison.

with traditional manifold learning embeddings without the need for any parameter tuning. We give additional results and details of the reconstruction experiment in our supplementary material.

5.4 Additional Applications

5.4.1 Image Layouts

The proposed framework can also be used to create image layouts used for quickly browsing large collections of unordered images. Placing perceptually similar images next to each other can improve human image retrieval tasks by reducing the perceptual load and thus accelerating visual processing [25].

We tested our framework on input data for the data driven morphing technique proposed by [1] comprised of four image sets where each image set contains between 148 and 722 images of different instances of the same object. Our framework was capable of producing many smooth and visually appealing

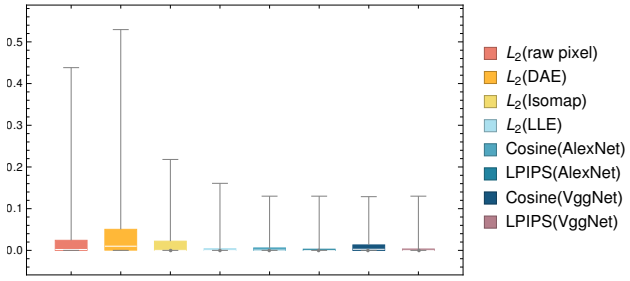


Fig. 9: A box and whisker chart for all test animations and test methods. The whisker endpoints show the maximum and minimum distance values. the solid box shows the 25 percent and 75 percent quantiles, and the white notch shows the median value. In this chart, L_2(LLE), LPIPS(AlexNet) and LPIPS(VggNet) are comparable in the minimum values.

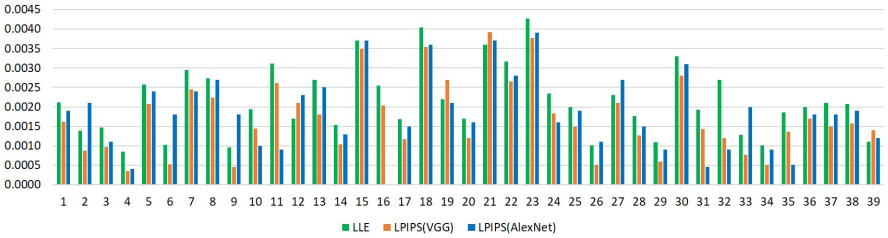


Fig. 10: Comparison of reconstruction error rate between LLE and LPIPS(VGG, AlexNet) on 39 single test cases. Obviously, LPIPS(VGG) and LPIPS (AlexNet) have lower error rate in most of the test animations.

image layouts from this data. Because of a large number of images in the datasets, our framework can be useful for visualizing smooth sub-sequences. For example, by identifying continuous subsequences of a given length with a minimum perceptual distance between adjacent frames or sampling longer sequences at even intervals. Figure 11 shows a radial image layout for images sequences generated with our proposed method, an example of how our system could be used to visualize large datasets of images of similar objects. Figure 12 shows an example of a smooth linear image layout generated by the proposed Hamiltonian path sequencing method applied to a collection of textured boot images. In the supplementary materials, we also present more results on this kind of application.

5.4.2 Video Synthesis

While this work focuses on animation video resequencing, our framework also applies to natural image video resequencing. If the input images depict stochastic motion, such as grass swaying in the win or ripples of water in a pool as the examples shown in Figure 13, smooth video resequences and cycle animations

Fig. 11: A radial image layout with sequences automatically generated by our system.

Fig. 12: Linear image layout example generated by the proposed method. Readers are suggested to see our supplementary video for a better visualization.

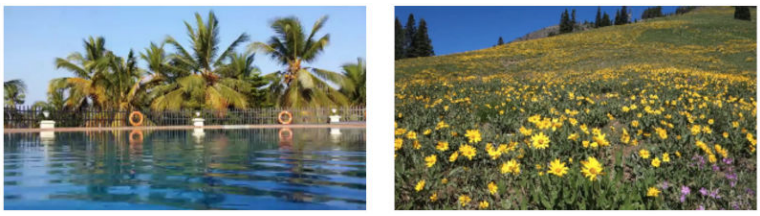
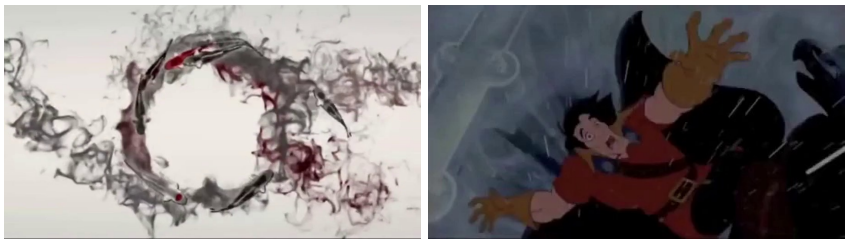


Fig. 13: Natural image video examples.



(a) (b)

Fig. 14: The school of fish example (a) and the falling sequence (b) are the examples where the proposed framework cannot produce smooth sequences other than the original video sequence.

can be generated using the LPIPS distance and the graph traversal algorithm described in our framework.

5.5 Limitations

Our framework's main limitation is its dependence on the input data. If the collection of input images is taken by densely sampling a video sequence with a strong distinction between backward and forward motion, such as the school of fish and falling sequence shown in Figure 14, then the MST may be path-like, and the proposed sequencing methods will likely select frames which are very similar to the dynamics of the original video sequence. In general, it may not be possible to generate new dynamics from input collections that do not contain a sufficient variety in motion and appearance. One possible way to overcome this limitation would be to develop an image-synthesis technique to generate new images that interpolate or extrapolate new motion by considering motion directions of objects.

6 Conclusion and Future work

We proposed a novel deep-learning framework for a new application for animation video resequencing which can generate smooth sequences and subsequences for many image styles. Our framework can serve as an efficient tool to automatically create new animation sequences from a collection of images. We also believe our framework could assist users in creating a comprehensive animation dataset by extracting many smooth subsequences from existing animation data. To our knowledge, a well-labeled dataset for general animation data does not yet exist.

Our results suggest that the activations of convolutional neural networks are useful features for smooth sequencing of photorealistic, non-photorealistic, ordered, and unordered image collections. Our quantitative analysis shows that

deep-features and the LPIPS metric can reconstruct animation sequences with greater accuracy than cosine distance of the same features, L_2 distance of the activations of a denoising autoencoder, L_2 distance in image space, and L_2 distance in the embedding space obtained by traditional manifold learning. Our qualitative results also show that the LPIPS metric produces a visible improvement over these other methods.

Despite the various styles, animators utilize a standard set of principles, including natural movement, to create more realistic looking animations. Thus, in the future, we would like to develop a self-supervised learning technique to extract motion features from existing animation video and combine metric learning and sequencing in a single deep learning optimization framework to solve problems in Figure 14.

Acknowledgements The authors would like to thank the reviewers for the many constructive comments that help improve the paper. This work was supported in part by the Ministry of Science and Technology (contracts 107-2221-E-006-196-MY3 and 108-2221-E-006-038-MY3), Taiwan.

References

1. H. Averbuch-Elor, D. Cohen-Or, and J. Kopf. Smooth image sequences for data-driven morphing. In *Computer Graphics Forum*, volume 35, pages 203–213. Wiley Online Library, 2016.
2. M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.
3. Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1511–1520, 2017.
4. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2009.
5. C. de Juan and B. Bodenheimer. Cartoon textures. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 267–276, 2004.
6. 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.
7. M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.
8. 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.
9. A. Hore and D. Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE, 2010.

10. D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993.
11. W. R. Inc. Mathematica, Version 11.3. Champaign, IL, 2018.
12. L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.
13. M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.
14. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.
15. J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.
16. G. Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, 1992.
17. H. Ling and D. W. Jacobs. Shape classification using the inner-distance. *IEEE transactions on pattern analysis and machine intelligence*, 29(2):286–299, 2007.
18. G. Liu, J. Si, Y. Hu, and S. Li. Photographic image synthesis with improved u-net. In *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*, pages 402–407. IEEE, 2018.
19. M. Osadchy, Y. L. Cun, and M. L. Miller. Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research*, 8(May):1197–1215, 2007.
20. S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.
21. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.
22. A. Schödl and I. A. Essa. Machine learning for video-based rendering. In *Advances in neural information processing systems*, pages 1002–1008, 2001.
23. 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.
24. 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.
25. K. Schoeffmann and D. Ahlstrom. Similarity-based visualization for image browsing revisited. In *2011 IEEE International Symposium on Multimedia*, pages 422–427. IEEE, 2011.
26. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

718 27. E. W. Stacy et al. A generalization of the gamma distribution. *The Annals*
719 *of mathematical statistics*, 33(3):1187–1192, 1962.
720 28. J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global ge-
721 ometric framework for nonlinear dimensionality reduction. *science*,
722 290(5500):2319–2323, 2000.
723 29. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and
724 L. Bottou. Stacked denoising autoencoders: Learning useful representa-
725 tions in a deep network with a local denoising criterion. *Journal of machine*
726 *learning research*, 11(12), 2010.
727 30. J. Yu, J. Cheng, and D. Tao. Interactive cartoon reusing by transfer
728 learning. *Signal processing*, 92(9):2147–2158, 2012.
729 31. J. Yu, D. Liu, D. Tao, and H. S. Seah. On combining multiple features
730 for cartoon character retrieval and clip synthesis. *IEEE Transactions on*
731 *Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(5):1413–1427,
732 2012.
733 32. J. Yu, M. Wang, and D. Tao. Semisupervised multiview distance metric
734 learning for cartoon synthesis. *IEEE Transactions on Image Processing*,
735 21(11):4636–4648, 2012.
736 33. R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unrea-
737 sonable effectiveness of deep features as a perceptual metric. In *Proceedings*
738 *of the IEEE conference on computer vision and pattern recognition*, pages
739 586–595, 2018.
740 34. S.-W. Zhang, C. C. Morace, T. Ngoc Hanh Le, C.-K. Yeh, S.-Y. Yao, S.-S.
741 Lin, and T.-Y. Lee. Animation video resequencing with a convolutional
742 autoencoder. In *SIGGRAPH Asia 2019 Posters*, pages 1–2. 2019.