Deep Sketch-guided Cartoon Video Inbetweening

Xiaoyu Li, Bo Zhang, Jing Liao, and Pedro V. Sander

Abstract—We propose a novel framework to produce cartoon videos by fetching the color information from two input keyframes while following the animated motion guided by a user sketch. The key idea of the proposed approach is to estimate the dense cross-domain correspondence between the sketch and cartoon video frames, and employ a blending module with occlusion estimation to synthesize the middle frame guided by the sketch. After that, the input frames and the synthetic frame equipped with established correspondence are fed into an arbitrary-time frame interpolation pipeline to generate and refine additional inbetween frames. Finally, a module to preserve temporal consistency is employed. Compared to common frame interpolation methods, our approach can address frames with relatively large motion and also has the flexibility to enable users to control the generated video sequences by editing the sketch guidance. By explicitly considering the correspondence between frames and the sketch, we can achieve higher quality results than other image synthesis methods. Our results show that our system generalizes well to different movie frames, achieving better results than existing solutions.

Index Terms—2D cartoon animation, sketch-guided synthesis, frame interpolation

✦

1 INTRODUCTION

CREATING cartoon animations can be partitioned into three main steps: drawing keyframes, inbetweening, and painting. First, an experienced animator draws the keyframes that capture the primary motion. Once completed, inbetweeners draw the inbetween frames for completing the motion, followed by a painter to fill the color in these sketches. Drawing and painting this large amount of inbetween frames is usually a specialized and time-consuming job, requiring intensive human labor from skilled professional artists, thus increasing the production cost. Therefore, we propose a system to alleviate this situation by automatically completing the inbetween frames including both the motion and color by only requiring some sketches for guidance, while maintaining the current animation workflow.

Research attempts have been made in helping users produce cartoon animations more easily. Sykora *et al.* [1] propose an interactive tool which simplifies the sketch colorization process by filling the color within a region, but it still requires significant manual labor. Whited *et al.* [2] present the BetweenIT system for the user-guided automation of tight inbetweening. Their methods mainly reduce the workload of drawing inbetween frames but not the painting process. We focus on completing the whole video considering both the motion and color. Some methods also utilize a hand-drawn sketch [3] or a color-coded skeleton [4] to guide synthetic animations, but Dvorožňák *et al.* [4] focus on one specific category of object and Zhu *et al.* [3] produce more free drawing animation but cannot handle motions with occlusions.

Furthermore, some related techniques can potentially be applied to assist the cartoon animation production, but many challenges restrict their direct use. One straightforward solution is to apply the state-of-the-art frame interpolation methods to two keyframes directly. However, these methods mainly focus on live-action (photorealistic) videos which makes it challenging to get satisfying results due to the large differences between live-action videos and cartoon animations [5], [6]. More importantly, the artists hope to control the inbetweening by drawing rather than use the deterministic result from interpolation. Recent image synthesis methods, either for general purpose [7], [8] or specifically for sketch colorization [9], [10], support automatically colorizing sketches with given frames as the reference. But without establishing the correspondence between the sketch and the frame, color bleeding artifacts may appear and the temporal consistency may also be hard to maintain.

The reason why it is hard to produce good results is that the problem of synthesizing videos from a sketch and cartoon keyframes is highly challenging. First, a cross-domain cartoon-to-sketch correspondence needs to be established. However, the cartoon frames are usually texture-less and the features in cartoon frames are unique and different from photographs, which is the target domain most previous matching methods are designed for. The situation is even worse for sketches, which makes establishing cartoon-to-sketch correspondence difficult. Second, the cartoon animations are more choppy and vigorous than live-action videos, which usually have unique object shape deformations and make occlusion estimation more difficult. Moreover, the unique contours in 2D cartoon frames can be easily de-

• X. Li is with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong.
E-mail: xliea@connect.ust.hk.
• B. Zhang is with Microsoft Research Asia, Beijing 100080, China.
E-mail: Tony.Zhang@microsoft.com.
• J. Liao is with the Department of Computer Science, City University of Hong Kong, Hong Kong. E-mail: jingliao@cityu.edu.hk.
• P. V. Sander is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong.
E-mail: psander@cse.ust.hk.

| I_0 | I_1 | $S_{3/6}$ (rough) | $S_{3/6}$ (simplified) |

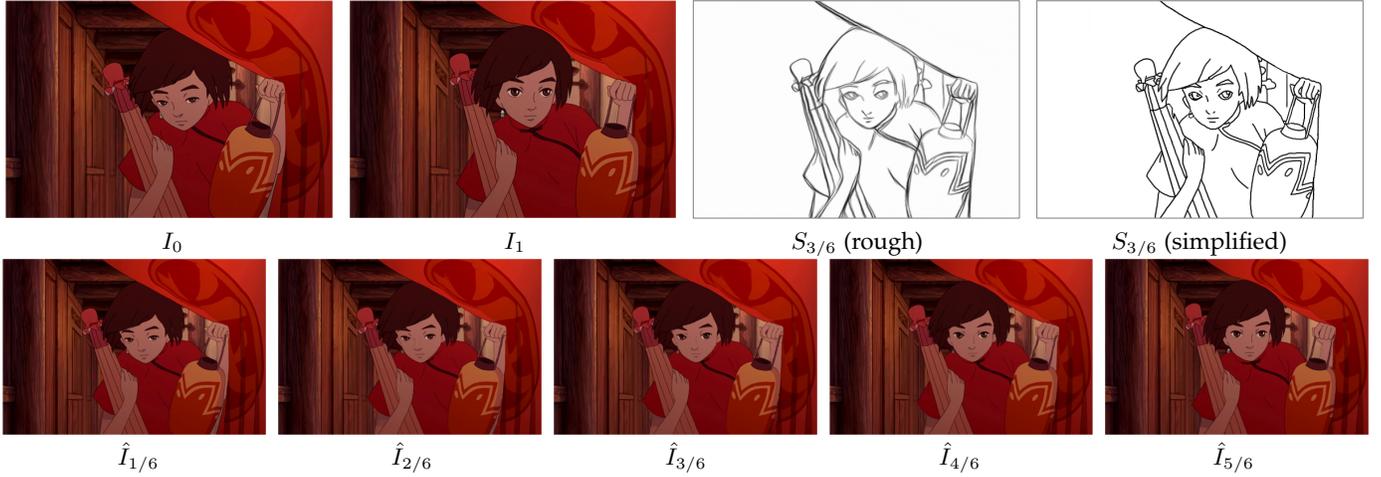| $\hat{I}_{1/6}$ | $\hat{I}_{2/6}$ | $\hat{I}_{3/6}$ | $\hat{I}_{4/6}$ | $\hat{I}_{5/6}$ |

Fig. 1. Our method synthesizes the frame $\hat{I}_{3/6}$ using two input keyframes $\{I_0, I_1\}$ and a guided sketch $S_{3/6}$ which was simplified from a rough input sketch drawn by artists. Furthermore, the approach can automatically interpolate additional inbetween frames $\{\hat{I}_{1/6}, \hat{I}_{2/6}, \hat{I}_{4/6}, \hat{I}_{5/6}\}$ producing a smooth video with motion prescribed by the user-given sketch. ©B&T.

stroyed by operations such as warping or resampling. It remains an open problem to generate cartoon frames without causing color bleeding or contour blurring. Finally, the valid frame rate (by removing the duplicated frames) of 2D cartoon animation is often low, i.e. 8-12 FPS, making it more challenging to achieve smooth interpolation results.

To address the above challenges and help users to automatically complete the inbetween frames, we propose a novel sketch-guided video synthesis system that can generate a sequence of inbetween frames controlled by one user-input sketch. Since the initial sketches from artists usually are very rough, casual, and potentially stylized, a pre-processing sketch cleanup needs to be performed to convert rough sketches into simplified clean line drawings. Then, the simplified sketches that contain the main contours or outlines of the objects are taken as the input guidance. To solve the cross-domain correspondence between a sketch and a cartoon frame, we first fill the large empty regions in the sketch with meaningful details by a transformation module conditioned on two keyframes. Then, two independent feature extractors are used to map the cartoon and sketch features into a common space that can be used to estimate the correspondence directly while maintaining the semantics of the original images. For occlusion handling, we estimate the occlusion mask by checking flow consistency and use a blending module to dynamically select and combine the pixels from two keyframes with these masks. Once the correspondence is established and the sketch frame is synthesized, an arbitrary-time frame interpolation module is used to generate and refine more inbetween frames. Finally, a temporal processing step is applied to further improve the result. Considering that the frame rate of 2D cartoon animation is low, we leverage the 3D cartoon movies which have smooth motions in nature to help training the interpolation and temporal processing module to produce temporally smooth 2D cartoon video results.

We demonstrate that our system can generate high-quality results in a broad range of scenes even containing some relatively large motions and works for cartoon movies with different styles. Moreover, with the sketch as guidance,

our system allows the users to easily control the motion trajectory of the generated video by drawing sketches, thus increasing the flexibility. One example can be seen in Figure 1. Our major contributions can be summarized as follows:

1) A system to resolve the demand in inbetweening by allowing users to specify the motion by drawing the sketch in a way that is compatible with the traditional animation workflow. We propose this new scenario and show that our method outperforms these existing possible solutions by a large margin.

2) A cross-domain correspondence estimation method for sketches and cartoon frames matching, achieving more accurate flow results than current optical flow estimation methods finetuned for this problem.

3) A blending method with a novel contour loss that better leverages the motion boundary clue to alleviate color bleeding, and an occlusion estimation module using flow consistency checking which is robust to the errors in estimated flows and occlusion masks.

4) An arbitrary-time frame interpolation pipeline and temporal processing module to produce and refine more inbetween frames with temporal coherence learned from 3D cartoon movies.

2 RELATED WORK

While there is no prior work that also tries to guide the synthesis of the whole 2D cartoon video using only one sketch, there are several techniques that can potentially be used to achieve this goal. In this section, we give an overview of those methods as well as the related works in cartoon animation.

2.1 Sketch-guided Image Synthesis

Sketches have been used to depict the visual world since prehistoric times and are deemed as a convenient art form to all humans [11]. Due to its simplicity, it can serve as

a user-friendly control input for image synthesis. How to convert these easily acquired sketches to colorful images is thus a significant problem in both computational photography and cartoon animation. Chen *et al.* [12] compose a realistic picture from a freehand sketch annotated with text labels, which is realized by stitching several text-related photographs discovered online. Eitz *et al.* [13] and Bansal *et al.* [14] adopt a similar approach which composite in-the-wild shapes and parts. However, methods in this category are not suitable to synthesize complex images due to the limited image database, and may often produce disharmonious results as it is hard to unify the style of different parts. Recently, with the emergence of deep learning, sketches can be directly mapped to realistic photographs by learning from data [15]–[17]. Yet these works typically overfit a certain type of scene and usually produce low-resolution results with noticeable artifacts. Portenier *et al.* [18] present a sketch-guided image editing system that is specialized for faces. Moreover, recent image to image translation techniques can also be used to translate sketches to cartoon images [7], [8], [19]. However, applying these methods directly to our video task cannot give satisfactory result as the appearance of the output may deviate from the user-given keyframe and the generated video may introduce temporal flickering due to the nature of frame-by-frame processing. Furthermore, these methods are incapable to synthesize the frame at arbitrary intermediate time as in our approach.

There are methods specifically designed for cartoon generation. Sykora *et al.* [1] propose the first interactive tool that fills colors for sketch images. Zhang *et al.* [9] and Liu *et al.* [10] use deep neural networks to colorize sketches, but these methods target single images rather than video frames, and do not consider spatio-temporal consistency. The method proposed by Xing *et al.* [20] is similar to our scheme, which utilizes an artist-drawn sketch to animate a cartoon image. Nonetheless, they only consider 2D deformation to warp the input frame, and fail to address occlusions. We use two successive frames as input and can leverage richer information to address the issue. Dvorožňák *et al.* [4] also attempt to animate the cartoon frames, but their work is specialized to body skeletons, which limits their application to cartoon characters rather than the general genre. Instead of using a user-drawn image as guidance, Whited *et al.* [2] propose to interpolate two keyframes by asking users to interactively match the outlines of input frames and manually adjust the motion trajectories. The method can achieve impressive results. However, the correspondence between frames has to be established manually, and the interpolated uniformly varying motion is not flexible enough. In comparison, our solution is more compatible to the cartoon inbetweening workflow and provides more freedom for artists to create desired motions.

2.2 Cross-domain Correspondence

While significant advances have been made to estimate the optical flow for temporally adjacent frames [21]–[24], the semantic dense correspondence for general images remains challenging. Liu *et al.* [25] and Yang *et al.* [26] rely on manually-crafted features to obtain the correspondence of scenes under large appearance variation. Ben-Zvi *et al.* [27]

and Yang *et al.* [28] study the matching for stroke correspondence. Zhu *et al.* [3], on the other hand, identify region correspondence between consecutive cartoon frames by solving a graph problem. However, this method assumes that the cartoon frame is composed of multiple flat regions with homogeneous color, and is thus not suitable to contemporary cartoon movies that usually contain complex shading and textures. There have been works that use deep neural networks for semantic correspondence. Liao *et al.* [29] perform PatchMatch [30] in a deep feature pyramid to compute the semantic dense correspondence. Aberman *et al.* [31], on the other hand, focus on finding reliable sparse correspondence. However, both of them rely on a pre-trained classification model, e.g. VGG network, as feature extractor, and cannot capture semantics for sketch images. In our case, we wish to densely match the frame with the sketch image, where the latter lacks textures in most parts and only has semantic clues around the outlines. We solve this cross-domain correspondence problem in a self-supervised manner.

Recently image translation methods provides the ability to translate the images across multiple domains by learning the domain-invariant representation from data [32]–[35]. Liu *et al.* [32] map images in different domains to a latent code in a sheared-latent space. Huang *et al.* [33] decomposes the image representation into a domain-invariant content code and a domain-specific style code. Liu *et al.* [34] propose a model to learn disentangled features for describing cross-domain data to perform continuous cross-domain image translation and manipulation. All of these methods also inspire us to map the features of sketch image and cartoon frames to a domain-invariant space for correspondence estimation or alignment.

2.3 Video Frame Interpolation

Video frame interpolation increases the video frame rate by inferring smooth motion and can be used for frame recovery in video streaming [36], [37] and slow motion effects [5]. Classic frame interpolation algorithms are based on optical flow [38], [39] and the quality of frame interpolation heavily depends on the flow accuracy. These methods usually require computationally expensive optimization and well-designed regularization [40]. Recently, deep neural networks were proven to be a powerful hammer for frame interpolation and outperforms traditional methods in both quality and speed. Long *et al.* [41] first attempt to use deep neural network to directly synthesize the intermediate frames. Liu *et al.* [42] propose to learn a 3D optical flow in the space-time domain for frame warping and can support both frame interpolation and extrapolation. Many other learning strategies including interpolation kernels [43]–[45], context maps [46], and incorporation of depth information [6] can effectively improve the interpolation quality. Other methods focus on novel interpolation scenarios such as multi-frame interpolation for high frame rate videos [5], high resolution frame interpolation [47] and the interpolation under camera shake [48]. Yet all these methods can only produce deterministic results that appear plausible without any user control. In this work, we allow the user to explicitly control the motion path by drawing sketch, which

we believe is the most convenient way for artist interaction. Since motion ambiguity is greatly reduced, our method demonstrates superior quality especially when processing long interval keyframes.

3 SKETCH-GUIDED VIDEO SYNTHESIS

We propose a sketch-guided cartoon video synthesis that utilizes one user-input sketch between a pair of keyframes to guide the motion in generated videos. Figure 2 shows an overview of our method. Given two consecutive cartoon keyframes $\{I_0, I_1\} \in \mathbb{R}^{H \times W \times 3}$ (H and W are image height and width respectively) and a sketch image $S_t \in \mathbb{R}^{H \times W}$ at the time $t \in (0, 1)$, we first seek to synthesize an inbetween frame $\hat{I}_t$ that is geometrically aligned with the structure in S_t and photometrically consistent with the input keyframes. Occlusions are also properly handled for $\hat{I}_t$ at this stage. Then, we use the estimated flow in the first stage to generate more inbetween frames at arbitrary intermediate times. Finally, a temporal processing network further reduces the artifacts by considering all the synthesized frames in spatio-temporal space, and finally produce smooth video results. We subsequently elaborate on each module.

3.1 Sketch Simplification and Generation

Since the sketches drawn by artists can be rough and casual, developing a generic approach to process them directly is challenging. Therefore, a sketch simplification or cleanup is required as a pre-processing procedure, which removes superfluous details of sketches and leaves a clean line drawing to characterize the motion. In this work, we adopt existing simplification algorithms [49], [50] which are robust in producing good sketch simplification for unseen styles by leveraging unsupervised data during training. In our work, we use simplified sketches as the network input and focus more on video synthesis. We will use the term sketches to refer to the simplified ones which have a clean line drawing style unless otherwise specified.

In order to conduct supervised learning, we create a video dataset which contains cartoon frames $\{I_t\}$ and the corresponding synthetic sketch images $\{S_t\}$. It is well known that deep neural networks tend to overfit the training data and may generalize poorly to images that slightly deviate from the training samples. Therefore, it is crucial to generate synthetic data as close as the hand-drawn sketches as possible. Our sketch generation procedure mostly follows Portenier *et al.* [18]. Specifically, we first extract contour maps using the holistically-nested contour detection (HED) method [51], which provides multi-level contour map predictions. We choose the second level of its predictions as we empirically find that this level of output demonstrates good visual resemblance to real simplified sketches while maintaining high contour completeness. We further remove short contours by performing morphological operations. Additionally, we fit splines for the contour maps using Potrace [52] and smooth the curvature by manipulating control points as suggested in Portenier *et al.* [18]. Such curve smoothing is essential for improving the generalization since it allows better tolerance to the potentially inaccurate sketch simplification and helps the network to

learn the synthesis based on rough contour locations. We show two examples of synthetic sketches in Figure 3. One can see that the synthetic sketches outline the major content in the cartoon frame and closely mimic the simplified sketch used during inference.

3.2 Sketch-guided Frame Synthesis

Given a simplified sketch S_t, we now aim to hallucinate the corresponding frame $\hat{I}_t$ which is also conditioned on the content images $\{I_0, I_1\}$. The framework of this sketch-guided frame synthesis is illustrated in Figure 4. We first establish the dense correspondence between the sketch image and each of the input frames. Unlike conventional optical flow methods, we are trying to densely match images of distinct types. Then, we explicitly estimate a mask which accounts for the occluded region due to the foreground movement. This occlusion mask will guide the network to properly choose the non-occluded pixels from the warped frames and finally produce the blended result. Note that we learn these tasks in a self-supervised manner without any external labeling.

3.2.1 Cartoon-to-sketch Correspondence

Learning the cartoon-to-sketch correspondence is a non-trivial problem. As we will show in our experiments, directly using or fine-tuning an established flow estimation model fails to give accurate flow estimation since the sketch is a sparse representation and the correspondence for large areas of blank regions is essentially ill-posed. To accomplish reliable dense correspondence, both the sketch and the cartoon frame are expected to be mapped to a space where feature maps demonstrate detailed structures. To help the sketch to add these structures, we use cartoon frames as conditional inputs when extracting the features of sketch image. Specifically, we propose a transformer network with input S_t and $\{I_0, I_1\}$ to hallucinate the missing structures of the sketch. After enhancing the structure of the sketch, we compute the correspondence in the deep features extracted from two independent mapping functions. We introduce this transformer only at the sketch branch, so the correspondence network has two asymmetric branches as shown in Figure 4.

3.2.1.1 Architecture: The transformer consists of several dilated residual layers [53] so that the receptive field is large enough to accommodate displacement between S_t and I_0 (or I_1). After transforming the sketch image to a proper feature space, we adopt PWC-Net [21] as the flow estimator. This approach is capable of dealing with large motion by coarse-to-fine matching. Specifically, the PWC-Net estimates the flow in a feature pyramid, where the low-level flow is refined from a higher-level estimation. Here we initialize the feature extractor of PWC-Net with pre-trained weights but let the two branches independently update during training. The network computes the correlation in the cost volume [21] and estimates the bidirectional flow $f_{t \leftrightarrow 0}$ and $f_{t \leftrightarrow 1}$ for the two cartoon-sketch pairs.

3.2.1.2 Loss Functions: The ground truth flows are not available in our dataset, and flows computed by off-the-shelf flow estimation models may introduce errors. Instead,
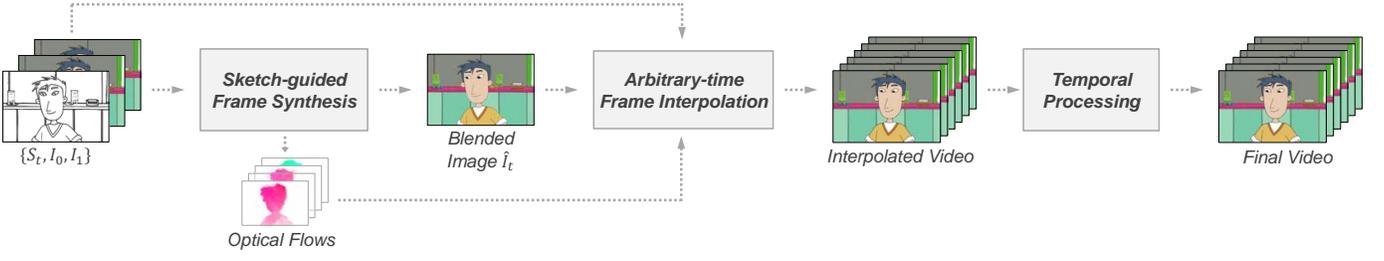
Fig. 2. Our sketch-guided cartoon video synthesis consists of three stages. We first establish the correspondence between the sketch S_t and keyframes $\{I_0, I_1\}$ and synthesize a blended image $\hat{I}_t$ corresponding to S_t. Then, we use the estimated flow from the first stage to interpolate additional inbetween frames and produce video results with an arbitrary frame rate. Finally, a temporal processing module is used to improve temporal consistency of the video.
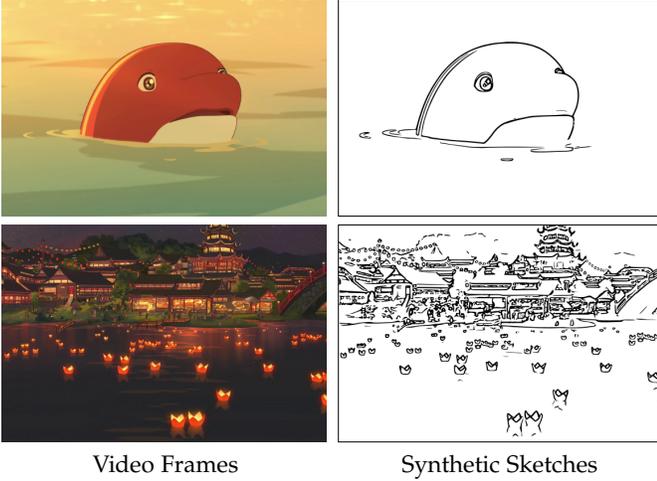

Fig. 3. Two synthetic sketch examples. The synthetic sketches outline the major content in the cartoon frame. ©B&T.

we use warping loss $\mathcal{L}_{warping}$ which calculates the ℓ_1 difference between the ground truth and the warped frames according to the flow estimation. Albeit slight errors within occlusion, this loss suffices to serve as a rough guidance for flow training. The warping loss is defined as:

$$\mathcal{L}_{warping} = \left\| I_t - w(I_0, f_{t\to 0}) \right\|_1 + \left\| I_t - w(I_1, f_{t\to 1}) \right\|_1 \\ + \left\| I_0 - w(I_t, f_{0\to t}) \right\|_1 + \left\| I_1 - w(I_t, f_{1\to t}) \right\|_1 \tag{1}$$

where $w(\cdot, \cdot)$ denotes the backward warping function.

3.2.2 Consistency Checking

The foreground objects may undergo large displacements in two adjacent keyframes and inaccurate flows in the occlusion may severely degrade the warping quality. To alleviate this, we perform occlusion estimation by flow consistency checking. Occluded points cannot find corresponding counterparts in the other image, so the cyclic mapping will unlikely map them back to the original location. Formally, we use the spatial Euclidean distance to measure such consistency. Therefore the mask $O_{t\to 0} \in \mathbb{R}^{H\times W}$ accounting for the visibility in I_0 can be computed as:

$$O_{t\to 0}(p) = 2\sigma\left(\left\| v(v(p, f_{t\to 0}), f_{0\to t}) - p \right\|_2 \right) - 1 \tag{2}$$

where σ denotes the sigmoid function which is used to map the value in occlusion mask to $(0, 1)$, and v is the mapping

function: $v(p, f) = p + f(p)$. The visibility of I_1 is computed similarly. Since the mask calculation is differentiable, it will in turn improve the flow prediction in the subsequent blending network.

3.2.3 Blending

We propose a blending network which predicts a soft blending mask $M \in \mathbb{R}^{H\times W}$ and fuses the warped cartoon frames $I_{t\to 0}$ and $I_{t\to 1}$ accordingly:

$$\hat{I}_t = M \odot I_{t\to 0} + (1 - M) \odot I_{t\to 1} \tag{3}$$

where $I_{t\to 0} = w(I_0, f_{t\to 0})$, $I_{t\to 1} = w(I_1, f_{t\to 1})$, and $\odot$ denotes the Hadamard product. The network takes as input the warped cartoon frames $\{I_{t\to 0}, I_{t\to 1}\}$, the occlusion masks $\{O_{t\to 0}, O_{t\to 1}\}$, and the sketch guidance S_t, and implicitly predicts the mask M during the final blending. The blending mask should range in $[0, 1]$ so it can be regarded as an attention map which properly selects from either frames. This blending mask not only considers the occlusion, but also resolves the blending artifacts due to the flow error. As each pixel in the blended image rigorously comes from the content frames, the output appears sharper than using a network that directly predicts a blended image.

3.2.3.1 Architecture: As the occlusion masks serve as a rough estimate, the network can predict the blending mask with a local receptive field. We determined that three convolutional layers are sufficient for a good estimation.

3.2.3.2 Loss function: The blending network needs to output I_t, so we introduce a blending loss to penalize the photometric ℓ_1 error:

$$\mathcal{L}_{blend} = \left\| \hat{I}_t - I_t \right\|_1 \tag{4}$$

The blended image, however, may still miss the contours that differentiate the neighboring color blocks, making the results appear blurry. This is because the contours are too thin to be penalized by the pixel-wise ℓ_1 loss. In order to improve the perceptual sharpness and maintain the cartoon style, we propose to promote the contours by adopting a contour loss based on Chamfer matching. A similar loss function has previously been adopted for artist drawing synthesis [54]. The idea is to transform the target contour maps into distance maps through Euclidean distance transform, where each pixel value stores the distance to the closest contour, e.g., a larger value in the distance map means a further distance to the contours. Let $E(\hat{I}_t) \in \mathbb{R}^{H\times W}$
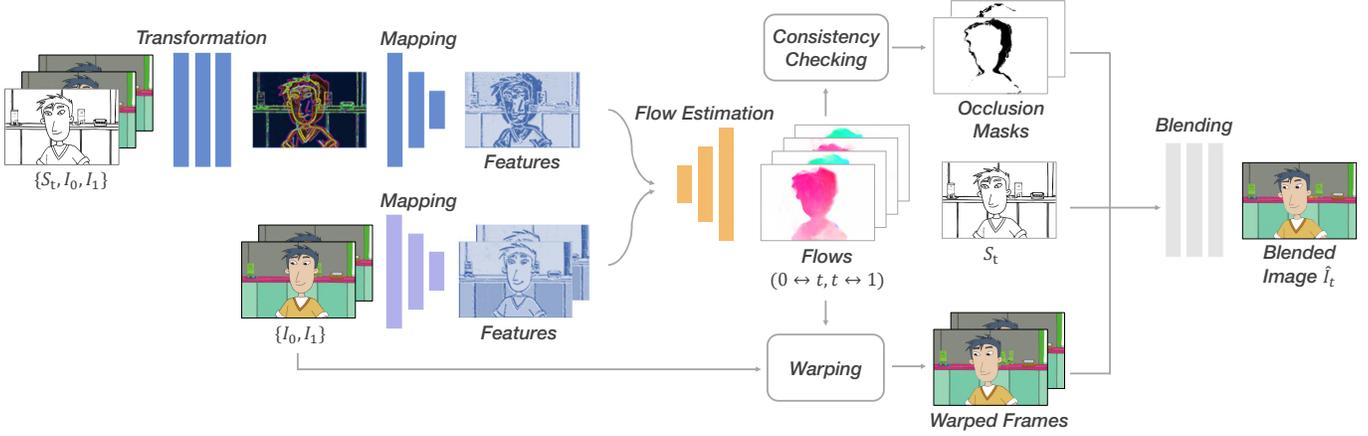
Fig. 4. Overview of the sketch-guided frame synthesis pipeline, including cartoon-to-sketch correspondence estimation, occlusion handling by flow consistency checking and frame blending.
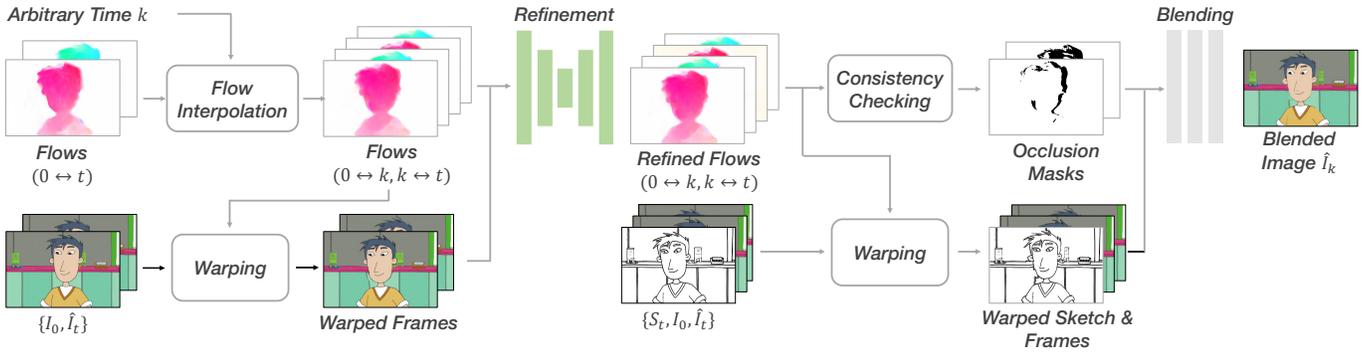

Fig. 5. Overview of the arbitrary-time frame interpolation pipeline, including flow interpolation, flow refinement, occlusion handling by flow consistency checking and frame blending.

be the detected contour map of $\hat{I}_t$, and $D \in \mathbb{R}^{H \times W}$ be the distance map of the ground truth contours. In order to match contours to the ground truth, $(1 - E(\hat{I}_t))$ should always sample small values in D. Formally, we penalize:

$$\mathcal{L}_{contour} = \left\| (1 - E(\hat{I}_t)) \odot D \right\|_1 \qquad (5)$$

If $\hat{I}_t$ fails to produce contours at the expected location, it will induce a higher contour loss. In our implementation, we use HED [51] to detect contours for the outputs and the ground truth. Since the contour extraction E is differentiable, the contour loss can guide the blending network to improve $\hat{I}_t$.

So far, we have shown how to synthesize $\hat{I}_t$, by cartoon-sketch correspondence, occlusion handling by flow consistency checking, and frame blending as shown in Figure 4. The overall objective function to train the entire synthesis network is:

$$\mathcal{L}_{syn} = \mathcal{L}_{blend} + \lambda_1 \mathcal{L}_{warping} + \lambda_2 \mathcal{L}_{contour} \qquad (6)$$

where we empirically set $\lambda_1 = 0.5$ and $\lambda_2 = 0.01$.

3.3 Arbitrary-time Frame Interpolation

At this stage, the guided sketch frame has been synthesized and the correspondences have been established through the synthesis pipeline. Next, we will leverage this information in order to automatically interpolate more inbetween frames. We note that not all motions in cartoon animation can be interpolated this way due to the free drawing nature of cartoon frames. Thus, for more complex and larger motions, additional frames with guided sketches should be synthesized before performing interpolation. Nonetheless, frame interpolation in 2D cartoon video is very useful in many scenarios and simply using methods for live-action videos does not achieve satisfactory results. Instead we can directly leverage the already obtained flow information as well as some of the building blocks used for consistency checking and blending in the cartoon synthesis stage to produce a more accurate final result.

The interpolation pipeline for generating a frame at an arbitrary intermediate time is shown in Figure 5. Without loss of generality, we illustrate the interpolation of I_k at time $k \in (0, t)$. We assume linear motion within $(0, t)$, so we approximate the bidirectional flow $f_{0 \leftrightarrow k}$ and $f_{k \leftrightarrow t}$ at time k by scaling $f_{0 \leftrightarrow t}$ proportionally. These flows are then refined so as to suppress the motion artifacts near the object boundaries. Equipped with the estimated flow, the cartoon frame at that time can be synthesized with a procedure similar to that in Section 3.2.

3.3.1   Flow Interpolation

Since we assume linear motion from I_0 to I_t, for an arbitrary intermediate time k, the flow can be estimated by

$$f_{0 \to k} = \frac{k}{t} f_{0 \to t}, \quad f_{t \to k} = \frac{t-k}{t} f_{t \to 0} \tag{7}$$

Solving for the flows $f_{k \to 0}$ and $f_{k \to t}$ in opposite directions, however, is more problematic. Inspired by the work of Jiang *et al.* [5], we assume the optical flow is locally smooth. To compute the flow at time k, we can borrow the flow at the same position at time 0 and t, and scale the magnitude proportionally. This way, we have the following two approximations:

$$f^0_{k \to t} \approx \frac{t-k}{t} f_{0 \to t}, \quad f^1_{k \to t} \approx -\frac{t-k}{t} f_{t \to 0} \tag{8}$$

Given these, we can combine them according to the temporal distance:

$$f_{k \to t} = \frac{t-k}{t} f^0_{k \to t} + \frac{k}{t} f^1_{k \to t} \approx \frac{(t-k)^2}{t^2} f_{0 \to t} - \frac{k(t-k)}{t^2} f_{t \to 0} \tag{9}$$

Similarly, we derive the estimation of $f_{k \to 0}$ as

$$f_{k \to 0} \approx -\frac{k(t-k)}{t^2} f_{0 \to t} + \frac{k^2}{t^2} f_{t \to 0} \tag{10}$$

One advantage of such interpolation scheme is that we can ensure the interpolated motion is temporally smooth.

3.3.2   Flow Refinement

The flow approximation works well for smooth motion, but may fail when points undergo non-linear motion or lie near motion boundaries. Thus, we train a flow refinement network to improve the flow estimation as shown in Figure 5. Specifically, the network takes as input the warped frames $\{I_{0 \to k}, I_{t \to k}\}$ and the rough flow estimation $\{f_{0 \leftrightarrow k}, f_{k \leftrightarrow t}\}$, and learns the flow residual for correction. Such residual learning helps to accelerate convergence and improve the flow quality. This flow refinement network adopts a U-Net structure [55] that has a broader receptive field for flow refinement and thus gives globally consistent prediction.

3.3.3   Frame interpolation

As shown in Figure 5, frame interpolation also performs consistency checking and final blending. The blending network shares the same weights those used for sketch-guided synthesis (Section 3.2). One should notice that we use the warped sketch S_t, *i.e.*, $S_{t \to k}$. This sketch information also improves this interpolation stage, as it helps to produce sharp results with clearly defined contours.

3.4   Temporal Processing

The inbetween outputs are generated independently frame-by-frame. In order to further improve temporal consistency, we propose to optimize them using a temporal processing network that considers the entire space-time volume. Specifically, we use Unet [55] to digest the concatenation of all the inbetween frames $\{I_k\}$ and provide a video output with improved consistency. This processing network adopts deformable convolutions [56] since the convolutional filters are performed on displaced grid points with learnable offsets, which can compensate the spatial misalignment of input video frames. We consistently observed improved loss curves using deformable convolution for temporal processing. The network is optimized to reduce the ℓ_1 loss between the ground truth and the generated video. Later we will see that this processing step also helps reduce the spatial artifacts, since the networks can rely on motion trajectory to propagate more information to regions with unreliable colors.

4   TRAINING

We next present how we construct the dataset and some strategies we adopt for training, both of which play an important role in our methods.

4.1   Data Preparation

We first extract all the frames in a 2D cartoon movie called Spirited Away, a 24fps HD animated film with a variety of different scenes. Since the movie usually has some repeated frames or frames with only subtle variations that have limited contribution to learning the model, we prune neighbor frames with an SSIM of 0.95 or higher. Finally, we produce a temporally downsampled movie with an average frame rate of approximately 8fps.

We then sort these frames by scenes. However, not all scenes are applicable for frame synthesis. Scenes without distinct semantic correspondences like rainfall and rising smoke are very challenging. Keeping these examples in the training data reduces the training performance. As such, we calculate matching costs for each pixel with its corresponding pixels at adjacent frame to filter the scenes. More specifically, we divide each scene into multiple triples of frames. In every triple, the first and third frames are warped to the second frame using optical flow PWC-Net [21]. We select the pixels which are closer to the second frame based on ℓ_1 error to form the final warped second frame. We then calculate the number of pixels which have a ℓ_1 error less than 5% of the color range between the final warped frame and the ground truth frame. Finally, the scenes with a pixel matching rate of less than 65% will be removed from the data. These selective frames will be used to generate their corresponding sketch images using the method described in Section 3.1.

For arbitrary-time frame interpolation pipeline and temporal processing, smaller smooth motion videos are required. Unfortunately, the common 2D cartoon animations can not meet these requirements. Because "one-shot three frames" or "one-shot two frames" are used for saving cost, most 2D animations are limited animated to 8-12 fps if the repetitive frames are removed. Therefore, it can not be used as the dataset to help our system to generate the video with an arbitrary high frame rate. To address this problem, we create a smooth motion dataset based on 3D cartoon videos. We explored with different options and used a 3D cartoon called The Octonauts, which is a 25fps full animated video. Next, we introduce how we use these two datasets to train the system.

4.2 Training Procedure

The entire system can be trained end-to-end by optimizing the networks from scratch in a single stage. However, it is difficult to get a good result in practice due to a large number of components and intermediate results. Moreover, recent works [6], [9], [57] have shown that multi-stage training and pre-trained models can be beneficial in this scenario. Therefore, we adopt a two-stage training to optimize the full model, which is shown to be more effective in our ablation study. We first train the frame synthesis pipeline with the correspondence network and the blending network as the two tasks mutually benefit each other using our 2D cartoon dataset and loss function in Equation 6. Then, we jointly train all the modules using the 3D dataset and ℓ_1 loss between generated video and the ground truth video. For the first stage, we use 3 consecutive frames as one sample to synthesize the middle frame. For the second stage, we use 7 consecutive frames as a sample to synthesize one middle frame and interpolate the remaining four middle frames. During inference, we can interpolate an arbitrary number of frames.

Our method is implemented in PyTorch [58] and the code will be made publicly available. We utilize approximately 36,000 2D cartoon frames and 10,000 3D cartoon frames for training. While all the frames come from one 2D cartoon movie and one 3D cartoon teleplay, we will show that it has the ability to generalize to many different movies and cartoon styles. We train our network on frames with resolution of 384×576 using the Adam optimizer [59] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, a learning rate of 0.0001 without any decay schedule, and batch size of 4 samples. It takes approximately 60 epochs to converge for the first stage and 100 epochs for the jointly trained second stage. The entire training procedure takes approximately three days on 4x Tesla M40 GPUs.

5 RESULTS

We next present ablation studies to verify the benefits of key components of our method, followed by comparisons to related techniques, including flow estimation, frame interpolation, and image synthesis methods. Finally, we present additional results to show the generalization ability of our method.

5.1 Model Analysis

We construct two test datasets for our ablation study. For *2D Cartoon Clips*, we collect 30 cartoon clips from 2D cartoon animations which are not seen during training. These clips cover different styles and scenes. We also downsample these clips temporally and generate a sketch for each middle frame of each triple as we did for training data preparation. Finally, we get approximately 500 triples for testing in this dataset. For *3D Cartoon Clips*, we use 20 cartoon clips from a different 3D TV animation to evaluate final generated videos with a smooth transition. We take every 7 consecutive frames as a group and only generate one sketch in the middle frame for each group having a total of 120 groups. We use PSNR, SSIM, and ℓ_1 between the estimated frames and the ground truth frames as the evaluation criteria.

TABLE 1
Ablation study for the first stage (frame synthesis pipeline).

Model	2D Cartoon Clips		
	ℓ_1 Loss	PSNR	SSIM
w/o sketch image	0.0219	25.97	0.871
w/o transformer	0.0101	32.40	0.945
w/o occlusion mask	0.0104	32.02	0.944
w/o warping loss	0.0107	31.74	0.937
w/o contour loss	0.0102	32.19	0.945
full synthesis model	**0.0095**	**32.70**	**0.950**

TABLE 2
Ablation study for joint training the entire framework.

Model	3D Cartoon Clips		
	ℓ_1 Loss	PSNR	SSIM
w/o joint training	0.0113	29.90	0.949
w/o refinement	0.0106	30.04	0.951
w/o temporal processing	0.0104	**30.11**	0.952
full model	**0.0102**	**30.11**	**0.953**

We first conduct an ablation study to determine the optimal settings for our sketch-guided frame synthesis pipeline in the first stage of training. The second stage that jointly trains all the modules starts from the first stage results by utilizing its synthetic middle frame and established correspondences. Therefore, the key empirical observation is that the better results the first stage can achieve, the better performance we get from the joint training of the second stage. We verify the effectiveness of four components in the first stage: transformer, occlusion mask, warping loss and contour loss. We also do an ablation study without the sketch image as the guidance. To keep all the other modules intact, we remove the sketch image by using a blank one. As shown in Table 1, the best results are achieved in terms of ℓ_1 loss, EPE and PSNR when the full synthesis model is used. Removing any of them causes performance degradation of varying degrees. An image example from the test dataset is shown in Figure 6. In this example, our method can produce results that have fewer visible artifacts and a high-quality synthesis result. The small deviation from the ground truth frame I_t is due to the guided sketch S_t does not follow the edges of I_t accurately. We find that warping loss and occlusion masks can increase the overall performance by a relatively large margin. The transformer can improve the region without dense guided strokes and contour loss can help maintain boundaries. Without the sketch as the guidance, the network has difficulty learning the motion in the 2D cartoon animation correctly, resulting in misalignment with respect to the ground truth.

Next, we evaluate the training strategies and key components for the second stage in which we jointly train all the modules after initialing the parameters trained from stage one. As the first experiment, we load the parameters of the first stage but do not update them and only train the interpolation network and temporal processing network (referred to as *w/o joint training*). Then we selectively remove the flow refinement and temporal processing stage to show their effect in results. Finally, we show the results using the

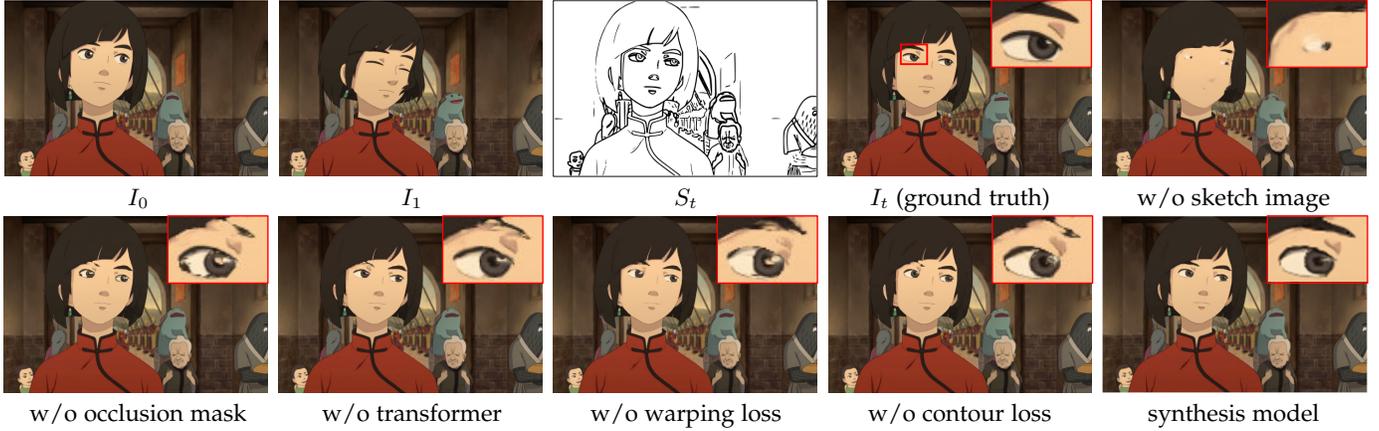| I_0	I_1	S_t	I_t (ground truth)	w/o sketch image
w/o occlusion mask	w/o transformer	w/o warping loss	w/o contour loss	synthesis model

Fig. 6. Sample frames for the ablation study of frame synthesis stage. Removing any of the components causes performance degradation of varying degrees. ©B&T.

I_0	I_1	$S_{3/6}$	$I_{3/6}$ (ground truth)

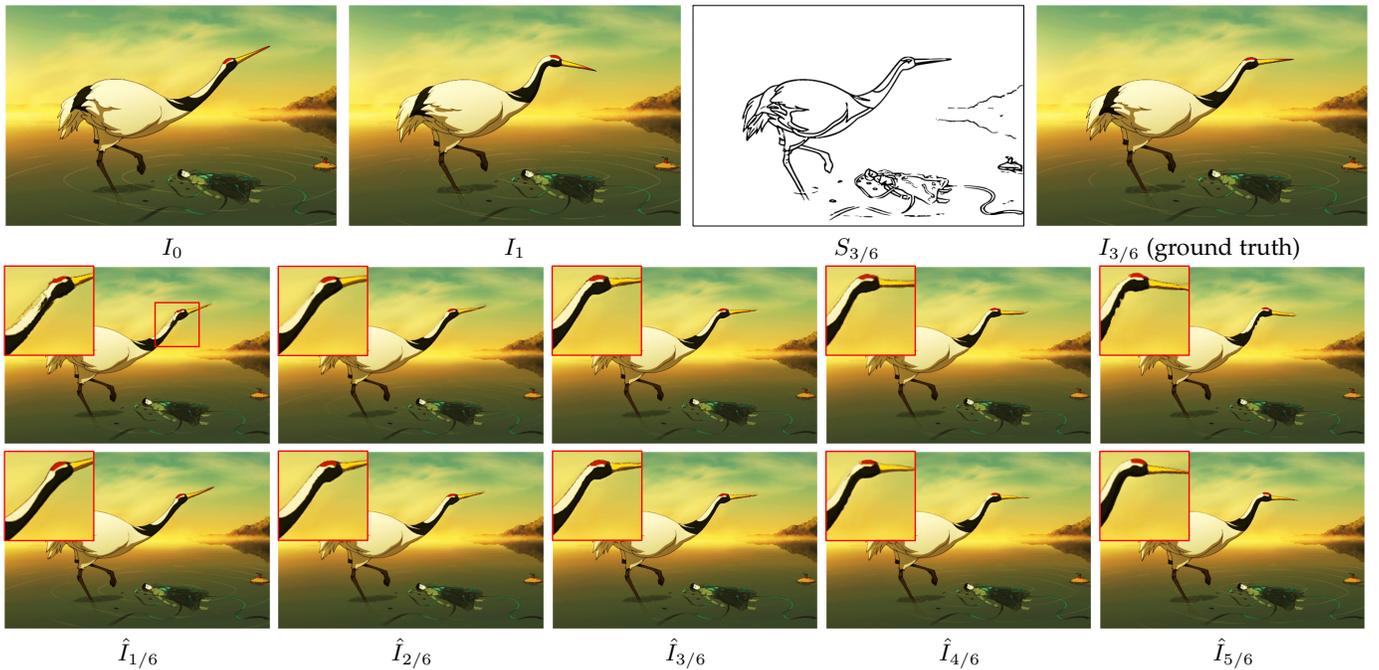| $\hat{I}_{1/6}$	$\hat{I}_{2/6}$	$\hat{I}_{3/6}$	$\hat{I}_{4/6}$	$\hat{I}_{5/6}$

Fig. 7. Example results comparing the model without refinement and temporal processing (second row) with our full model with joint training of the entire framework (third row). We can see that the full model can produce higher quality results equipped with the refinement and temporal processing. ©B&T.

full model. All of the results are summarized in Table 2. Note that the full model outperforms all the other settings. An image example is shown in Figure 7. We can see that with the refinement and temporal processing, our method can produce higher quality and more temporally coherent results.

5.2 Comparison to Flow Estimation Methods

We first compare our cartoon-to-sketch correspondence method to recent advanced flow estimation methods PWC-Net [21]. Since the ground truth optical flows for 2D cartoon animations are unavailable, we use the warping loss to train and evaluate the flows in 2D cartoon clips. More specifically, we use the estimated flows to warp one frame to another and calculate the ℓ_1 loss between warped frames and ground truth frames. We also use the MPI *Sintel* Flow Dataset [60] which is used for the evaluation of optical flow derived from the open-source 3D animated short film, Sintel. Notice that this dataset has a significantly different style from the training set. Since the clips from this dataset already have large motion, we do not temporally downsample the clips and only generate the sketch to provide it as input. The Sintel dataset contains 341 triples for our testing. Because the ground truth flow is available, we also use the endpoint error (EPE) as a metric. For each pair of frames, we take a frame image and a sketch image from the other frame as inputs to estimate their optical flow. We use our sketch generation method in Section 3.1 to generate the corresponding sketch images.

We first directly adopt PWC-Net [21] to estimate the cartoon-to-sketch correspondence. In the second experiment, we fine-tune their model in our training set with

TABLE 3
Quantitative evaluation of cartoon-to-sketch correspondence estimation
with different methods.

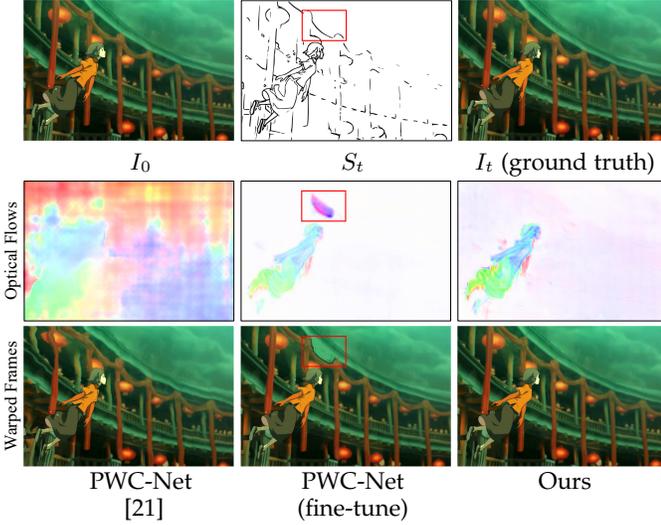| Model	Sintel		2D Cartoon Clips
	EPE	ℓ_1 Loss	ℓ_1 Loss
PWC-Net [21]	22.55	0.0596	0.0336
PWC-Net (fine-tune)	10.93	0.0265	0.0112
our model	**10.52**	**0.0247**	**0.0104**

Fig. 8. Comparison of different flow estimation methods. Our method can handle movement in the presence of sparse sketches or empty regions. ©B&T.

cartoon sketch pairs as input and keep the network structure and the mechanism unchanged. For our model, we remove the consistency checking and blending in the synthesis pipeline and only keep the cartoon-to-sketch correspondence network. The warping loss is utilized for both of these experiments. The quantitative results are shown in Table 3 and one example can be found in Figure 8. As the results show, our cartoon-to-sketch flow estimation method outperforms the common flow method by addressing challenging issues in texture-less cartoon frames and sketches with large empty regions.

5.3 Comparison to Frame Interpolation Methods

We next compare our method with some recent frame interpolation techniques that have source code available and are also able to interpolate arbitrary intermediate frames. These are the slow motion method by Jiang *et al.* [5] (SloMo) and the depth-aware interpolation method by Bao *et al.* [6] (DAIN). For a fairer comparison with these learning methods, we fine-tune their model on our training set. We use the *3D Cartoon Clips* temporally downsampled to different frame rates as inputs for testing.

The original frame rate for the clips is 24fps. We show results reconstructing the 24fps video using the input video temporally downsampled to different rates. The other interpolation methods will take two adjacent frames of the downsampled video as input. For our method, in addition to the two input frames, we also utilize the sketch generated from the middle frame as additional guidance. The
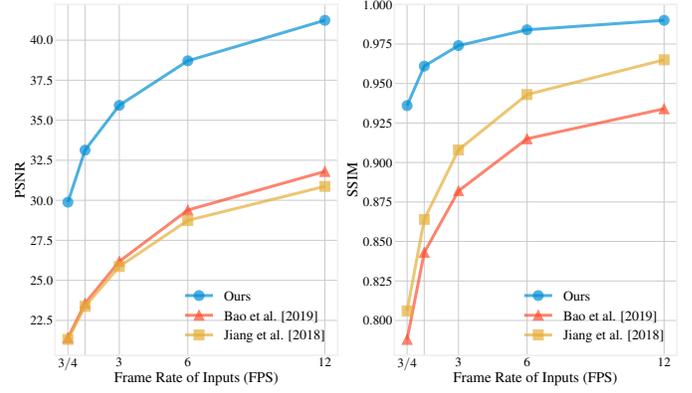

Fig. 9. Quality comparison of our approach with other methods at different input frame rates. Lower frame rate means larger displacement between successive frames. Our method outperforms all other methods by a large margin by taking advantage of the guided sketch.
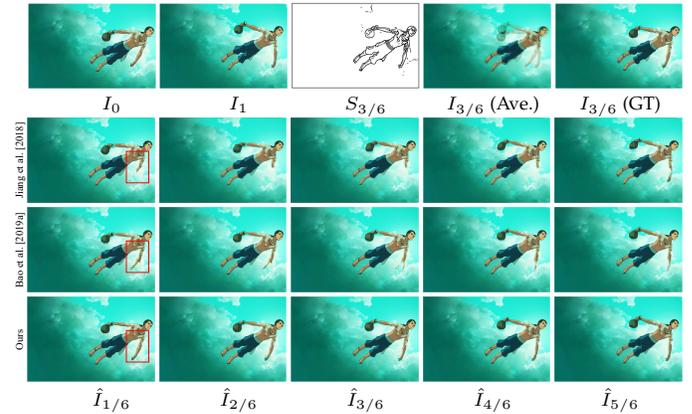

Fig. 10. Example results comparing our method (fourth row) with other frame interpolation approaches: Jiang *et al.* [5] (second row) and Bao *et al.* [6] (third row). Compared to off-the-shelf methods, our method tends to better preserves the image content. "Ave." represents average input frames and "GT" represents ground truth frame. ©B&T.

quantitative results measuring PSNR and SSIM are shown in Figure 9. Results show that our method outperforms all other methods by a large margin since it can take advantage of the sketch for guidance. The advantage is even more obvious when we interpolate frames with longer intervals. We also show a qualitative comparison in Figure 10, where our method takes $\{I_0, S_{3/6}, I_1\}$ as input to synthesize $I_{3/6}$ and interpolate four more frames $\{I_{1/6}, I_{2/6}, I_{4/6}, I_{5/6}\}$, and the other two methods take the two input frames $\{I_0, I_1\}$ to interpolate these five frames directly. We can see that both Jiang *et al.* [5] and Bao *et al.* [6] cannot always get satisfactory results due to the specific situations in cartoon frames, like large motion, texture-less style, unique contours. Our method takes advantage of the sketch as guidance, thus improving the the result by a large margin and also following the real-life workflow of cartoon animation. More importantly, the artists hope to control the inbetweening by drawing rather than using the deterministic result from interpolation. The sketch input provides the flexibility to control the interpolation result. We also shows an example of our results with different input frame rates in Figure 11 and it indicates that the lower the input frame rate, the lower

Fig. 11. Example results with different input frame rates (deformation ranges). For each result $\hat{I}_k$ in the third row, I_0 and I_{2k} from the first row are used as inputs. We also show the averaged input frames for reference in the second row. As the interval between input frames increases, artifacts begin to appear due to the larger motion. ©2:10 AM Animation.

TABLE 4
Quantitative evaluation of frame interpolation only, providing ground truth to previous methods.

Method	PSNR	SSIM
Averaged frames	33.09	0.952
Jiang *et al.* [5]	29.85	0.955
Bao *et al.* [6]	36.03	**0.977**
Ours	**36.64**	0.971

the resulting quality.

The additional sketch input of our method provides an advantage when we do the comparisons to earlier work, showing the need for such a sketch in order to achieve the artist's intent. However, this is an advantageous comparison since the previous methods do not use a sketch and are just focused on interpolation. Thus, we provide another experiment to test just the interpolation ability of our method in a disadvantageous condition to our approach. We use the *3D Cartoon Clips* in Section 5.1 as the test data for this experiment. For these clips, we take every 7 consecutive frames $I_{0/6}$ to $I_{6/6}$ as a sample. We use $\{I_0, S_{3/6}, I_1\}$ to interpolate $\{I_{1/6}, I_{2/6}, I_{4/6}, I_{5/6}\}$ for our method and use $\{I_0, I_{3/6}, I_1\}$ to interpolate $\{I_{1/6}, I_{2/6}, I_{4/6}, I_{5/6}\}$ for the two interpolation methods. We also compared the averaged input frames. The result is shown in Table 4. We can see that even if this experiment is disadvantageous for our method since we only have $S_{3/6}$ while the others have ground truth $I_{3/6}$ for interpolation, our method can still produce competitive results in terms of interpolation ability.

5.4 Comparison to Image Synthesis Methods

We compare our method with recent image generation or synthesis techniques. More specifically, we compare it with pix2pixHD [8], a state-of-the-art conditional generative adversarial networks and a sketch colorization method [9] which also targets cartoon images and utilizes two-stage

training strategy. For the pix2pixHD, we use the sketch image as the input and two cartoon frames as the image translation condition. The model is trained using our training set until convergence. For the sketch colorization method, we directly use their pre-trained model for inference as their method is trained on a large-scale cartoon dataset. We show two results of their method. One is to use the manually selected color hints to colorize the sketch. Another is to use the cartoon frame as the reference to colorize the sketch.

As shown in Figure 12, the sketch colorization method is unable to handle the complex color styles and tends to use a relatively smooth color within a region even using many color hints as additional inputs. Moreover, their method does not utilize the temporal information of the video, thus produce a relatively low-quality result. Due to the nature of colorization, it faithfully follows edges from the input sketch but cannot recover any structure details that are missing the sketch. Our method has the ability to address such variations, such as the shadow on the clothes in the second example. On the other hand, pix2pixHD suffers from bleeding artifacts in the regions with large motion. Furthermore, it severely overfits the cartoon style in the training set: when we test on a movie beyond the training set, it suffers from color shifting.

5.5 Generalization Ability and Flexibility

We try to maximize the generalization of our method by constructing a dataset containing diverse scenes and large motions. Furthermore, we attempt to synthesize simplified sketches as realistically as possible. Though ultimately we trained the network solely using the training samples from only one 2D cartoon movie and one 3D cartoon animations, our method can still perform well on frames in movies with different styles, even a 3D cartoon that does not have obvious contours and 2D animations as shown in Figure 13. Our training set only contains one single sketch style synthesized by our algorithm. However, it still has the capability to

| I_0 I_1 | S_t and hints (only for Zhang *et al.* [9]) | Zhang *et al.* [9] by using hints | Zhang *et al.* [9] by using references | Wang *et al.* [8] | Ours |

Fig. 12. Example results comparing image synthesis methods with our approach. ©B&T.

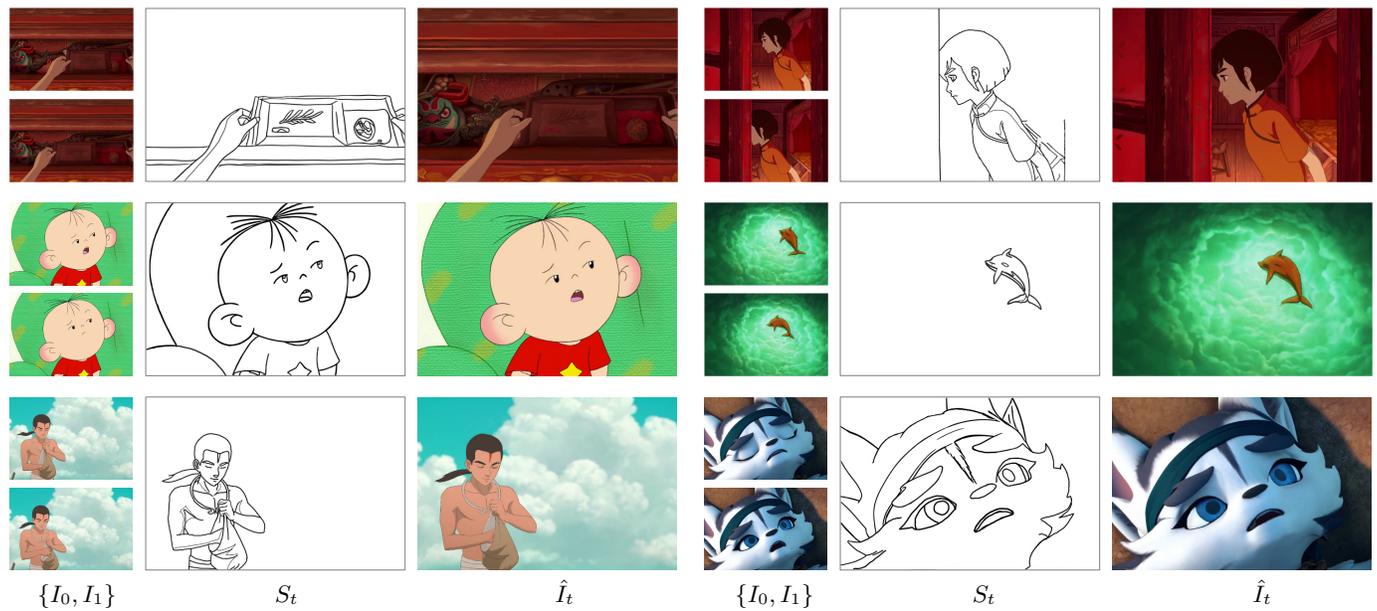| $\{I_0, I_1\}$ | S_t | $\hat{I}_t$ | $\{I_0, I_1\}$ | S_t | $\hat{I}_t$ |

Fig. 13. Results from hand-drawn sketches with different cartoon and drawing styles. ©B&T, 2:10 AM Animation, SAFS.
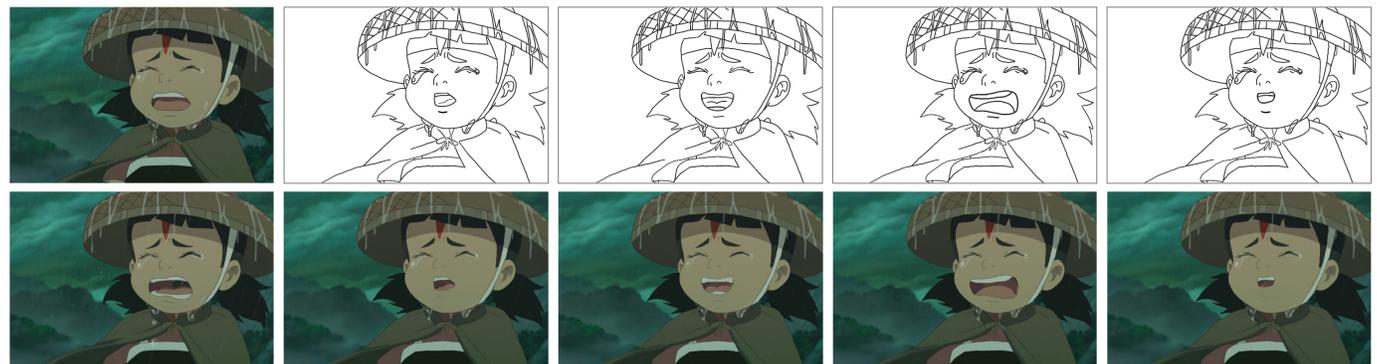

Fig. 14. Our system can synthesize different middle frames by drawing different guided sketches. In this example, the proposed method takes two input frames (first column) and synthesizes the cartoon frames (second row) guided by the corresponding user's sketches (first row). ©B&T.

$\{I_0, I_1\}$ First level   Second level   Third level

Fig. 15. Results using sketches with different levels of detail. In this example, the proposed method takes two input frames (first column) and synthesizes the cartoon frames (second row) guided by the corresponding sketches (first row). ©B&T.

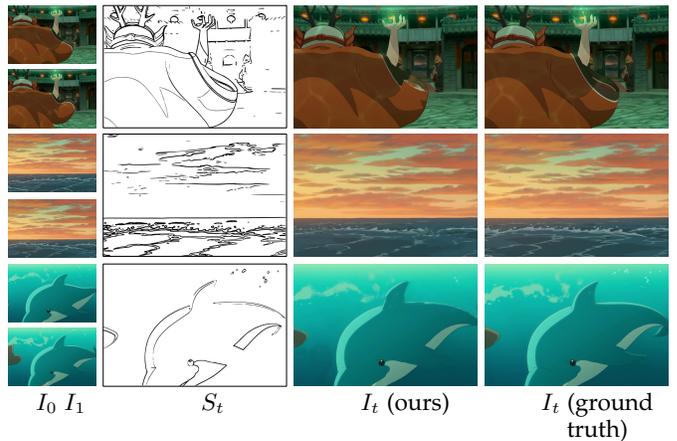I_0 I_1 S_t I_t (ours) I_t (ground truth)

Fig. 16. Examples where our approach did not yield satisfactory results, including pixels being occluded in both two input frames, unclear semantic correspondence, and artifacts due to the incomplete sketch for indicating motion. ©B&T.

generalize to rough sketches after simplification (Figure 1) or hand-drawn sketches (Figure 13, 14). Furthermore, our method has the flexibility to generate different results by providing different guided sketches as shown in Figure 14. Users can choose to automatically interpolate the whole video by just drawing one middle sketch if the motion is relatively simple, or drawing more sketches to synthesize frame by frame if the motion is complex and can not be interpolated. The user can also use a combination of the two approaches. In Figure 15, we use an example to show how the level of detail affects the output of our result. To do so, we progressively remove some lines from the first level of the real sketch and show their results in the second row. Note that our method can still produce reasonable results when omitting some lines (e.g. remove the shadow of the hair on the face or change the nose according to the lines). But artifacts may appear if some important lines indicating the motion are omitted. The model can produce a better result if more details are provided in the sketch. Moreover, due to the fully convolutional networks we use, our method can address video frames with different resolutions without a drop in performance. One example can be found in Figure 13, whose original frames come from different movies at different resolutions (e.g., 480p, 720p, and 1080p).

The capabilities of our method can be summarized as follows:

1) It generalizes to different cartoon styles;
2) It generalizes to sketches with some variations;
3) It supports the generation of different video results from the same input by drawing different sketches;
4) It supports drawing and synthesizing one (or more) sketch and interpolating the remaining frames;

6 LIMITATIONS AND CONCLUSION

In conclusion, we present a novel framework that synthesizes cartoon videos by using the color information from two input frames while following the animated motion guided by a sketch. Our approach first estimates the dense cross-domain correspondence between a sketch and video frames by transforming the cartoon and sketch into feature representations in the same domain, followed by applying a blending module for occlusion handling considering flow consistency. Then, the inputs and the synthetic frame equipped with established correspondence is fed into an

arbitrary-time interpolation pipeline to generate and refine more inbetween frames. Finally, a video temporal processing approach is used to further improve the result. We perform several experiments to verify each component of our system, show side-by-side comparisons with related methods, and demonstrate the generalization ability and flexibility of our system. Our results show that our system generalizes well to different scenes and produce high-quality results.

However, there are some cases that our method cannot handle perfectly. First, our method is based on warping and blending. If the pixels in the middle frame are occluded in both two input frames, artifacts will appear as shown in the first example of Figure 16. Second, some scenes in a 2D cartoon without accurate semantic correspondence, such as the waves in Figure 16 which can appear and disappear suddenly with different shapes, also cannot be addressed by our method. Third, when the contours that are vital to indicate motion are missing in the sketch image, it is hard for our method to infer that information from the two input frames accurately, e.g., the fins of the dolphin in the third example of Figure 16. To address this limitation, our method allows the user to interactively drawing more strokes in the sketch. It would be worthwhile to explore how to solve these artifacts automatically.

REFERENCES

[1] D. Sỳkora, J. Dingliana, and S. Collins, "Lazybrush: Flexible painting tool for hand-drawn cartoons," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 599–608.

[2] B. Whited, G. Noris, M. Simmons, R. W. Sumner, M. Gross, and J. Rossignac, "Betweenit: An interactive tool for tight inbetweening," in *Computer Graphics Forum*, vol. 29, no. 2. Wiley Online Library, 2010, pp. 605–614.

[3] H. Zhu, X. Liu, T.-T. Wong, and P.-A. Heng, "Globally optimal toon tracking," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–10, 2016.

[4] M. Dvorožňák, W. Li, V. G. Kim, and D. Sỳkora, "Toonsynth: example-based synthesis of hand-colored cartoon animations," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 167, 2018.

[5] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz, "Super slomo: High quality estimation of multiple intermediate frames for video interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9000–9008.

[6] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang, "Depth-aware video frame interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3703–3712.

[7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[8] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8798–8807.

[9] L. Zhang, C. Li, T.-T. Wong, Y. Ji, and C. Liu, "Two-stage sketch colorization," in *SIGGRAPH Asia 2018 Technical Papers*. ACM, 2018, p. 261.

[10] Y. Liu, Z. Qin, T. Wan, and Z. Luo, "Auto-painter: Cartoon image generation from sketch by using conditional wasserstein generative adversarial networks," *Neurocomputing*, vol. 311, pp. 78–87, 2018.

[11] M. Eitz, J. Hays, and M. Alexa, "How do humans sketch objects?" *ACM Trans. Graph.*, vol. 31, no. 4, pp. 44–1, 2012.

[12] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu, "Sketch2photo: Internet image montage," in *ACM transactions on graphics (TOG)*, vol. 28, no. 5. ACM, 2009, p. 124.

[13] M. Eitz, R. Richter, K. Hildebrand, T. Boubekeur, and M. Alexa, "Photosketcher: interactive sketch-based image synthesis," *IEEE Computer Graphics and Applications*, vol. 31, no. 6, pp. 56–66, 2011.

[14] A. Bansal, Y. Sheikh, and D. Ramanan, "Shapes and context: in-the-wild image synthesis & manipulation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2317–2326.

[15] Y. Güçlütürk, U. Güçlü, R. van Lier, and M. A. van Gerven, "Convolutional sketch inversion," in *European Conference on Computer Vision*. Springer, 2016, pp. 810–824.

[16] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays, "Scribbler: Controlling deep image synthesis with sketch and color," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5400–5409.

[17] W. Chen and J. Hays, "Sketchygan: Towards diverse and realistic sketch to image synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9416–9425.

[18] T. Portenier, Q. Hu, A. Szabo, S. A. Bigdeli, P. Favaro, and M. Zwicker, "Faceshop: Deep sketch-based face image editing," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 99, 2018.

[19] P. Zhang, B. Zhang, D. Chen, L. Yuan, and F. Wen, "Cross-domain correspondence learning for exemplar-based image translation," *arXiv preprint arXiv:2004.05571*, 2020.

[20] J. Xing, L.-Y. Wei, T. Shiratori, and K. Yatani, "Autocomplete hand-drawn animations," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, pp. 1–11, 2015.

[21] 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*, 2018, pp. 8934–8943.

[22] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[23] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[24] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4161–4170.

[25] C. Liu, J. Yuen, and A. Torralba, "Sift flow: Dense correspondence across scenes and its applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 5, pp. 978–994, 2010.

[26] H. Yang, W.-Y. Lin, and J. Lu, "Daisy filter flow: A generalized discrete approach to dense correspondences," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3406–3413.

[27] N. Ben-Zvi, J. Bento, M. Mahler, J. Hodgins, and A. Shamir, "Line-drawing video stylization," in *Computer Graphics Forum*, vol. 35, no. 6. Wiley Online Library, 2016, pp. 18–32.

[28] W. Yang, H.-S. Seah, Q. Chen, H.-Z. Liew, and D. Sỳkora, "Ftp-sc: Fuzzy topology preserving stroke correspondence," in *Computer Graphics Forum*, vol. 37, no. 8. Wiley Online Library, 2018, pp. 125–135.

[29] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang, "Visual attribute transfer through deep image analogy," *arXiv preprint arXiv:1705.01088*, 2017.

[30] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," in *ACM Transactions on Graphics (ToG)*, vol. 28, no. 3. ACM, 2009, p. 24.

[31] K. Aberman, J. Liao, M. Shi, D. Lischinski, B. Chen, and D. Cohen-Or, "Neural best-buddies: Sparse cross-domain correspondence," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 69, 2018.

[32] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in neural information processing systems*, 2017, pp. 700–708.

[33] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 172–189.

[34] A. H. Liu, Y.-C. Liu, Y.-Y. Yeh, and Y.-C. F. Wang, "A unified feature disentangler for multi-domain image translation and manipulation," in *Advances in neural information processing systems*, 2018, pp. 2590–2599.

[35] A. Gonzalez-Garcia, J. Van De Weijer, and Y. Bengio, "Image-to-image translation for cross-domain disentanglement," in *Advances in neural information processing systems*, 2018, pp. 1287–1298.

[36] X. Huang and S. Forchhammer, "Cross-band noise model refinement for transform domain wyner–ziv video coding," *Signal Processing: Image Communication*, vol. 27, no. 1, pp. 16–30, 2012.

[37] J. Wu, C. Yuen, N.-M. Cheung, J. Chen, and C. W. Chen, "Modeling and optimization of high frame rate video transmission over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2713–2726, 2015.

[38] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *International journal of computer vision*, vol. 12, no. 1, pp. 43–77, 1994.

[39] M. Werlberger, T. Pock, M. Unger, and H. Bischof, "Optical flow guided tv-l 1 video interpolation and restoration," in *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, 2011, pp. 273–286.

[40] D. Mahajan, F.-C. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur, "Moving gradients: a path-based method for plausible image interpolation," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3. ACM, 2009, p. 42.

[41] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu, "Learning image matching by simply watching video," in *European Conference on Computer Vision*. Springer, 2016, pp. 434–450.

[42] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala, "Video frame synthesis using deep voxel flow," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4463–4471.

[43] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive convolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 670–679.

[44] ——, "Video frame interpolation via adaptive separable convolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 261–270.

[45] W. Bao, W.-S. Lai, X. Zhang, Z. Gao, and M.-H. Yang, "Memc-net: Motion estimation and motion compensation driven neural net-

work for video interpolation and enhancement," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[46] S. Niklaus and F. Liu, "Context-aware synthesis for video frame interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1701–1710.

[47] T. Peleg, P. Szekely, D. Sabo, and O. Sendik, "Im-net for high resolution video frame interpolation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2398–2407.

[48] J. Choi and I. S. Kweon, "Deep iterative frame interpolation for full-frame video stabilization," *arXiv preprint arXiv:1909.02641*, 2019.

[49] E. Simo-Serra, S. Iizuka, K. Sasaki, and H. Ishikawa, "Learning to simplify: fully convolutional networks for rough sketch cleanup," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–11, 2016.

[50] E. Simo-Serra, S. Iizuka, and H. Ishikawa, "Mastering sketching: adversarial augmentation for structured prediction," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 1, pp. 1–13, 2018.

[51] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1395–1403.

[52] P. Selinger, "Portrace," vol. 6, no. 7, 2015. [Online]. Available: http://potrace.sourceforge.net

[53] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 472–480.

[54] R. Yi, Y.-J. Liu, Y.-K. Lai, and P. L. Rosin, "Apdrawinggan: Generating artistic portrait drawings from face photos with hierarchical gans," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10743–10752.

[55] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[56] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 764–773.

[57] M. Xia, X. Liu, and T.-T. Wong, "Invertible grayscale," in *SIGGRAPH Asia 2018 Technical Papers*. ACM, 2018, p. 246.

[58] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

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

[60] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)*, ser. Part IV, LNCS 7577, A. Fitzgibbon et al. (Eds.), Ed. Springer-Verlag, Oct. 2012, pp. 611–625.

Bo Zhang received his Ph.D. degree with the Department of Electronic and Computer Engineering at the Hong Kong University of Science and Technology at 2019. Prior to that, he received a Bachelor of Engineering degree at Zhejiang University. Now he is a researcher at visual computing group of Microsoft research asia. His research interests involve low-level computer vision, image synthesis, computational photography and imaging system.

Jing Liao is an Assistant Professor with the Department of Computer Science, City University of Hong Kong (CityU) since Sep 2018. Prior to that, she was a Researcher at Visual Computing Group, Microsoft Research Asia from 2015 to 2018. She received the B.Eng. degree from Huazhong University of Science and Technology and dual Ph.D. degrees from Zhejiang University and Hong Kong UST. Her primary research interests fall in the fields of Computer Graphics, Computer Vision, Image/Video Processing, Digital Art and Computational Photography.

Pedro V. Sander received a Bachelor of Science in Computer Science from Stony Brook University, and Master of Science and Doctor of Philosophy degrees from Harvard University. He was a senior member of the Application Research Group of ATI Research, where he conducted real-time rendering and general-purpose computation research with latest generation and upcoming graphics hardware. Currently, he is a Professor in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology. His research interests lie mostly in real-time rendering, graphics hardware, geometry processing, and imaging.

Xiaoyu Li received a Bachelor of Engineering degree in Electronic Information Engineering from Huazhong University of Science and Technology, in 2017. He is currently pursuing a Ph.D. degree with the Electronic and Computer Engineering, Hong Kong University of Science and Technology. His research interests include computer vision and deep learning with an emphasis on computational photography.