Deep Sketch-guided Cartoon Video Synthesis

XIAOYU LI, Hong Kong UST
BO ZHANG, Microsoft Research Asia
JING LIAO, City University of Hong Kong
PEDRO V. SANDER, Hong Kong UST

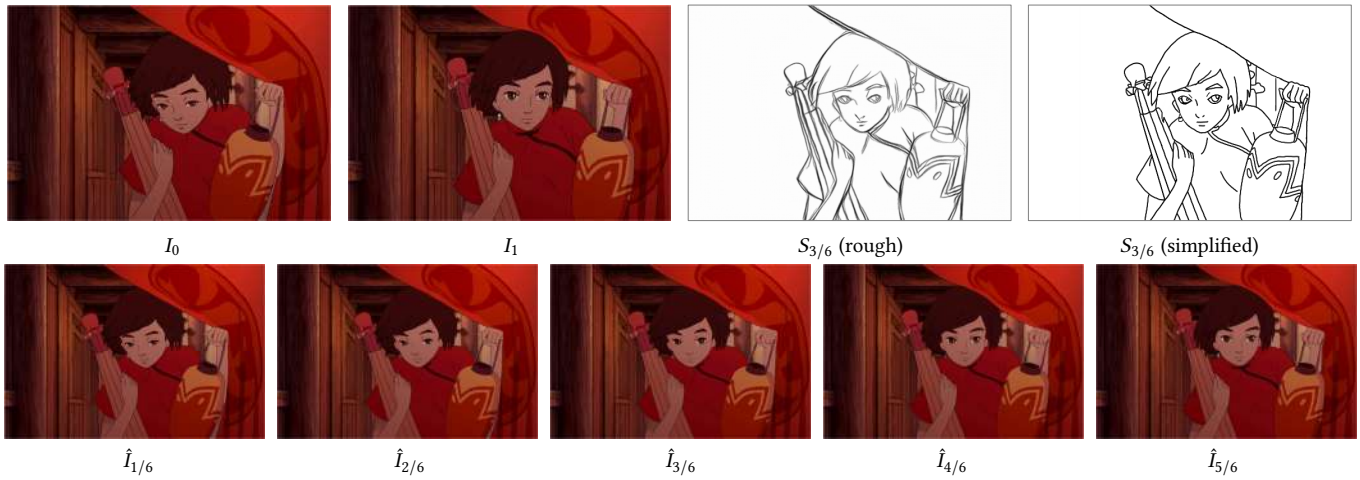


Fig. 1. Our method synthesizes the frame $\hat{I}_{3/6}$ using two input keyframes $\{I_0, I_1\}$ and a guided sketch $S_{3/6}$ which was simplified from a rough input sketch. Furthermore, the approach can automatically interpolate additional inbetween frames ($\hat{I}_{1/6}$, $\hat{I}_{2/6}$, $\hat{I}_{4/6}$,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

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, following by a blending module with occlusion estimation to synthesize the middle frame guided by the sketch. After that, the inputs 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 video post-processing approach is used to further improve the result. 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, our methods can achieve high-quality synthetic results compared with image synthesis methods. Our results show that our system generalizes well to different movie frames, achieving better results than existing solutions.

CCS Concepts: •**Computing methodologies** → **Image manipulation;** *Computer vision; Matching;*

Additional Key Words and Phrases: sketch-guided cartoon synthesis, frame interpolation

1 INTRODUCTION

Creating 2D cartoon animations can be divided 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 with skilled professionals, 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.

Research attempts have been made in helping users produce cartoon animations more easily. Sỳkora et al. [2009] 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. [2010] 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 [Zhu et al. 2016] or a color-coded skeleton [Dvorožnák et al. 2018] to guide synthetic animations, but Dvorožnák et al. [2018] focus on one specific category of object and Zhu et al. [2016] 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 between two

keyframes directly. However, these methods [Bao et al. 2019a; Jiang et al. 2018] mainly focus on live-action videos which makes it challenging to get adequate results due to the large differences between live-action videos and cartoon animations. Recent image synthesis methods, either for general purpose [Isola et al. 2017; Wang et al. 2018] or specifically for sketch colorization [Liu et al. 2018; Zhang et al. 2018], support automatically colorizing sketches with given frames as the reference. But without establishing correspondence between the sketch and the frame, color bleeding artifacts are obvious and the temporal consistency is also hard to maintain.

The reason why it is hard to produce good results is because the problem of synthesizing videos from a sketch and cartoon keyframes is highly challenging. First, a cross-domain cartoon-to-sketch correspondence need to be established. However, the cartoon frames are usually texture-less and lack the features for matching. 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 makes occlusion estimation especially difficult due to large motions. Moreover, the unique contours in 2D cartoon frames can be easily distorted by operations such as warping or resampling, causing color bleeding and blurry outline artifacts in results. Finally, the actual frame rate (by removing the duplicated frames) of 2D cartoon animation is often low, i.e. 8-12 FPS, which makes it more challenging for temporally smooth video generation.

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 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 conditioning 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, video post-processing 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 post-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. And our major contributions can be summarized as follows:

(1) A sketch-guided cartoon video synthesis approach utilizing sketches to guide the motion and synthesize a full video with an arbitrary frame rate. We show that our method outperforms these existing 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 occlusion estimation using flow consistency checking, which is robust to the errors in estimated flows and occlusion masks.
(4) An arbitrary-time frame interpolation pipeline and video post-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 [Eitz et al. 2012]. 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. [2009] 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. [2011] and Bansal et al. [2019] 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 [Chen and Hays 2018; Güçlütürk et al. 2016; Sangkloy et al. 2017]. Yet these works typically overfit a certain type of scene and usually produce low-resolution results with noticeable artifacts. Portenier et al. [2018] 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 [Isola et al. 2017; Wang et al. 2018; Zhang et al. 2020]. 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. Sỳkora et al. [2009] propose the first interactive tool that fills colors for sketch images. Zhang et al. [2018] and Liu et al. [2018] 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. [2015] 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. Dvorožnák et al. [2018] 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. [2010] 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 [Dosovitskiy et al. 2015; Ilg et al. 2017; Ranjan and Black 2017; Sun et al. 2018], the semantic dense correspondence for general images remains challenging. Liu et al. [2010] and Yang et al. [2014] rely on manually-crafted features to obtain the correspondence of scenes under large appearance variation. Ben-Zvi et al. [2016] and Yang et al. [2018] study the matching for stroke correspondence. Zhu et al. [2016], 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. [2017] perform PatchMatch [Barnes et al. 2009] in a deep feature pyramid to compute the semantic dense correspondence. Aberman et al. [2018], 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.

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 [Huang and Forchhammer 2012; Wu et al. 2015] and slow motion effects [Jiang et al. 2018]. Classic frame interpolation algorithms are based on optical flow [Barron et al. 1994; Werlberger et al. 2011] and the quality of frame interpolation heavily depends on the flow accuracy. These methods usually require computationally expensive optimization and well-designed regularization [Mahajan et al. 2009]. 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. [2016] first attempt to use deep neural network to directly synthesize the intermediate frames. Liu et al. [2017] 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 [Bao et al. 2019b; Niklaus et al. 2017a,b], context maps [Niklaus and Liu 2018], and incorporation of depth information [Bao et al. 2019a] can effectively improve the interpolation quality. Other methods focus on novel interpolation scenarios such as multi-frame interpolation for high frame rate videos [Jiang et al. 2018], high resolution frame interpolation [Peleg et al. 2019] and the interpolation under camera shake [Choi and Kweon 2019]. 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 post-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 preprocessing 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 [Simo-Serra et al. 2018, 2016] 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 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

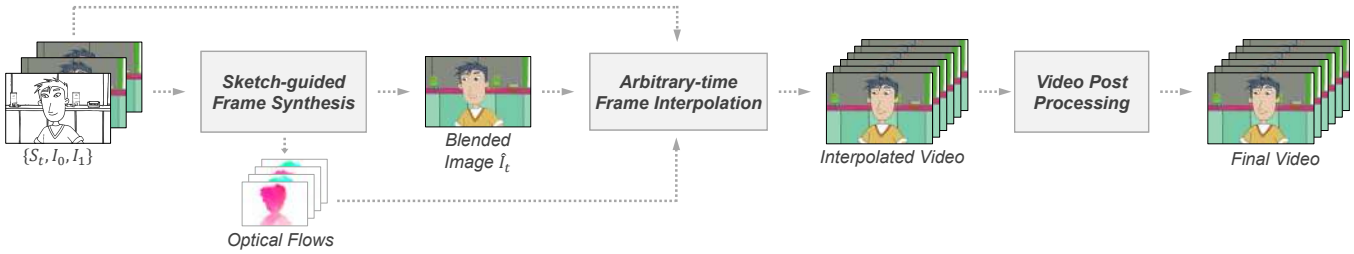
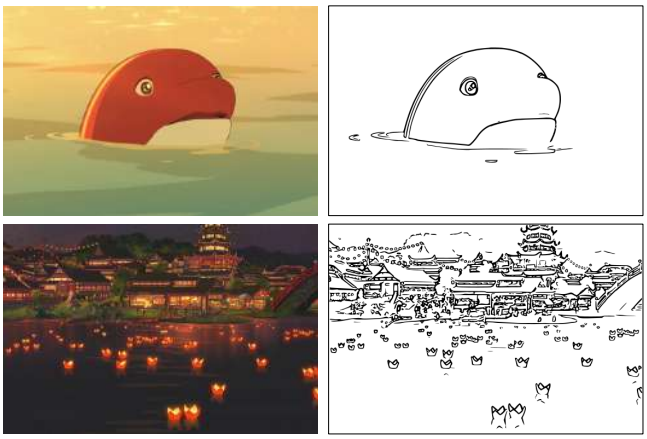


Fig. 2. Our sketch-guided cartoon video synthesis consists of three stages. We first establish the correspondence between the sketch S_t and keyframes $\{I_0, I_1\}$ and synthesize a blended image $\hat{I}_t$ corresponding to S_t. Then, we use the estimated flow from the first stage to interpolate additional inbetween frames. Finally, we use a post-processing network to further reduce the artifacts by considering the generated frames as an entire sequence and produce a temporally consistent video output.



Video Frames Synthetic Sketches

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

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. [2018]. Specifically, we first extract contour maps using the holistically-nested contour detection (HED) method [Xie and Tu 2015], 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 [Selinger 2015] and smooth the curvature by manipulating control points as suggested in [Portenier et al. 2018]. 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 in fact non-trivial. Directly using or fine-tuning an established flow estimation model as shown in Figure 5 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 "grow" 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. Only after enhancing the structure of the sketch can we compute the correspondence in the deep features. We introduce this transformer only at the sketch branch, so the correspondence network has two asymmetric branches. This is also in line with the finding in recent work [Zhao et al. 2020].

Architecture. The transformer consists of several dilated residual layers [Yu et al. 2017] 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 [Sun et al. 2018] 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

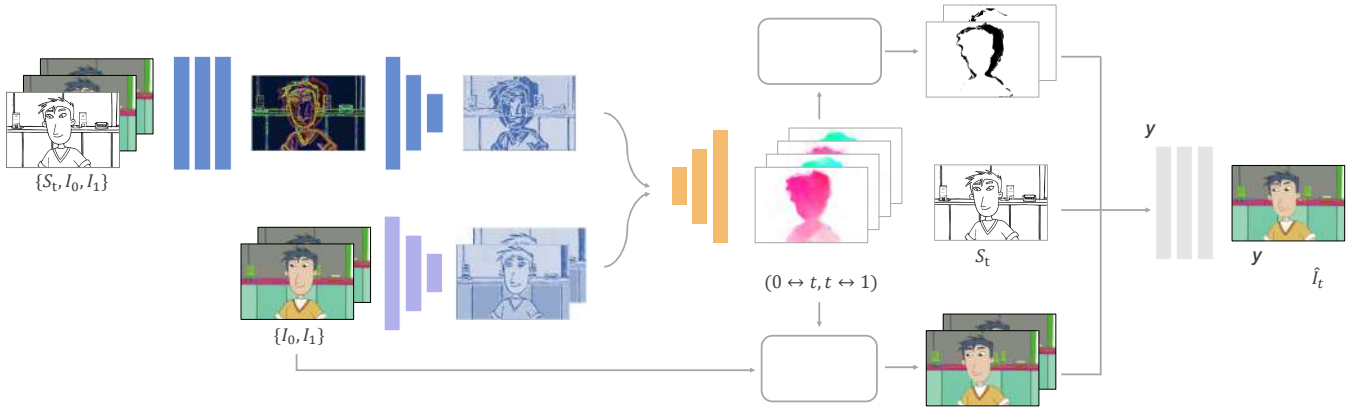


Fig. 4. Overview of the sketch-guided frame synthesis pipeline.

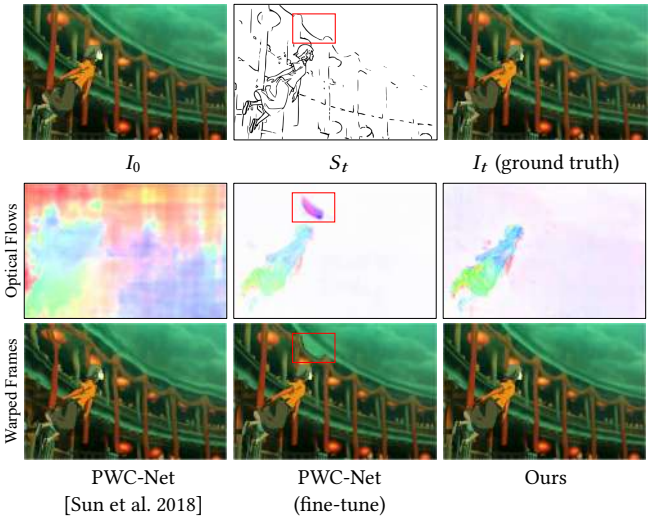


Fig. 5. Comparison of different flow estimation methods. Our method can handle movement in the presence of sparse sketches or empty regions. ©B&T.

network computes the correlation in the cost volume [Sun et al. 2018] and estimates the bidirectional flow $f_{t\leftrightarrow 0}$ and $f_{t\leftrightarrow 1}$ for the two cartoon-sketch pairs.

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, 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:

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

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

Fig. 6. The consistency checking module calculates the forward-backward flow consistency to estimate the occlusion mask ($O_{t\to 0}$). Occlusion is denoted by darker regions.

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 (Figure 6). 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:

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

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:

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

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.

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.

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 [Yi et al. 2019]. 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 $E(\hat{I}_t) \in \mathbb{R}^{H \times W}$ 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:

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

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 [Xie and Tu 2015] 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. The overall objective function to train the entire synthesis network (Figure 4) is:

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

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. For this stage we assume linear motion. 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 can and should be synthesized before performing interpolation. Nonetheless, frame interpolation in 2D cartoon video is very useful in many scenarios and automating this final stage is non-trivial. 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 process for generating a frame at an arbitrary intermediate time is shown in Figure 7. 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. [2018], 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 7. 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 [Ronneberger et al. 2015] that has a broader receptive field for flow refinement and thus gives globally consistent prediction.

3.3.3 Frame interpolation. As shown in Figure 7, 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

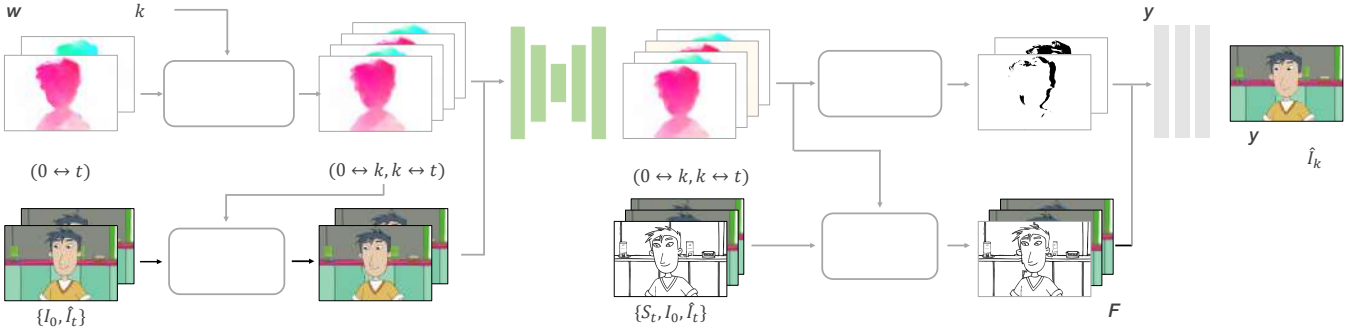


Fig. 7. Overview of the arbitrary-time frame interpolation pipeline.

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 Video Post-processing

The inbetween outputs are generated independently frame-by-frame. In order to further improve temporal consistency, we propose to optimize them using a post-processing network that considers the entire space-time volume. Specifically, we use Unet [Ronneberger et al. 2015] to digest the concatenation of all the inbetween frames $\{I_k\}$ and provide a video output with improved consistency. This post-processing network adopts deformable convolutions [Dai et al. 2017] 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 video 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 post-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 [Sun et al. 2018]. 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.

For arbitrary-time frame interpolation pipeline and video post-processing, smaller smooth motion videos are required. Unfortunately, the common 2D cartoon animations can not meet these requirements. Because fione-shot three framesfi or fione-shot two framesfi 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 to optimize 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 [Bao et al. 2019a; Xia et al. 2018; Zhang et al. 2018] 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

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

Model	2D Cartoon Clips		
	ℓ_1 Loss	PSNR	SSIM
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 post-processing	0.0104	**30.11**	0.952
full model	**0.0102**	**30.11**	**0.953**

remaining four middle frames. During inference, we can interpolate an arbitrary number of frames and post-process 5 middle frames in each batch.

Our method is implemented in PyTorch [Paszke et al. 2019] 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 [Kingma and Ba 2014] 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.

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

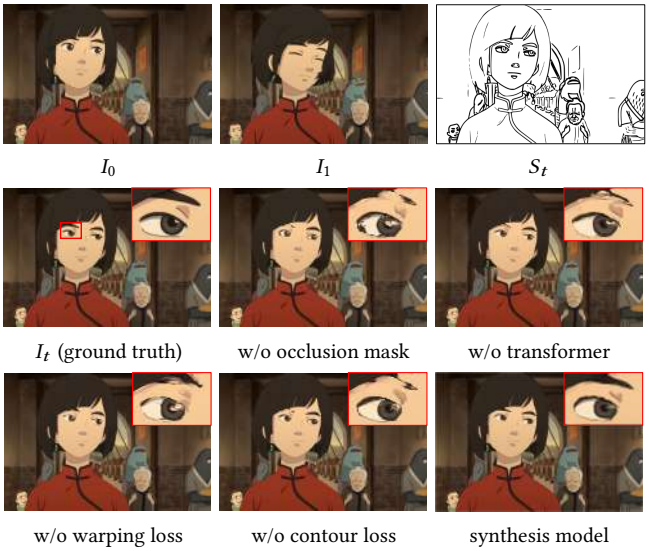


Fig. 8. Sample frames for the ablation study of frame synthesis stage. Removing any of the components causes performance degradation of varying degrees. ©B&T.

stage. We verify the effectiveness of four components in the first stage: transformer, occlusion mask, warping loss and contour loss. 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 8. 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.

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 post-processing network (referred to as *w/o joint training*). Then we selectively remove the flow refinement and post-processing stage to show their effect in results. Finally, we show the results using

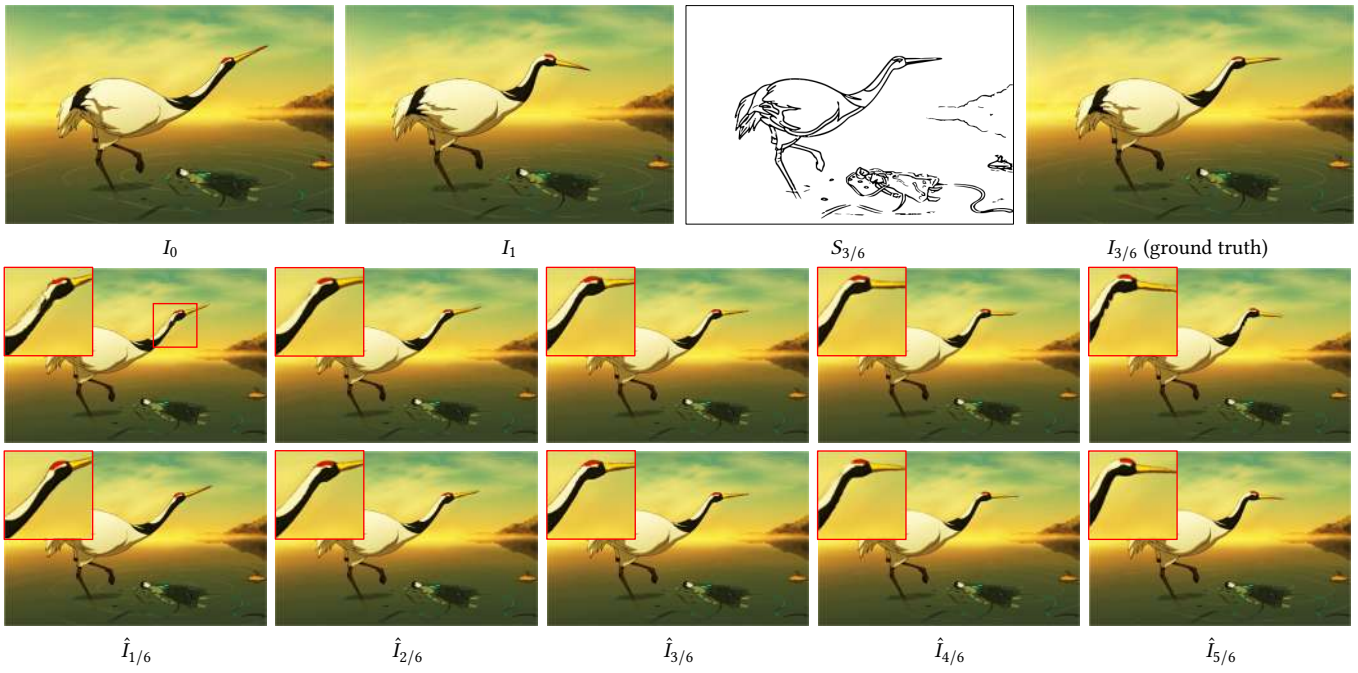


Fig. 9. Example results comparing the model without refinement and post-processing (second row) with our full model with joint training of the entire framework (third row). We can see that the full model can produce higher quality results equipped with the refinement and post-processing. ©B&T.

the 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 9. We can see that with the refinement and post-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 [Sun et al. 2018]. 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 [Butler et al. 2012] 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.

We first directly adopt PWC-Net [Sun et al. 2018] to estimate the cartoon-to-sketch correspondence. In the second experiment, we fine-tune their model in our training set with cartoon sketch pairs. For our model, we remove the consistency checking and blending in the synthesis pipeline and only keep the modules related to flow estimation. The warping loss is utilized to train this model and fine-tune the PWC-Net. The quantitative results are shown in Table 3 and one example can be found in Figure 5, which has been analysed in Section 3.2. 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.

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 [Sun et al. 2018]	22.55	0.0596	0.0336
PWC-Net (fine-tune)	10.93	0.0265	0.0112
our flow estimation model	**10.52**	**0.0247**	**0.0104**

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. [2018] (SloMo) and the depth-aware interpolation method by Bao et al. [2019a] (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 quantitative results measuring PSNR and SSIM are shown in Figure 10. 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 lower the input frame

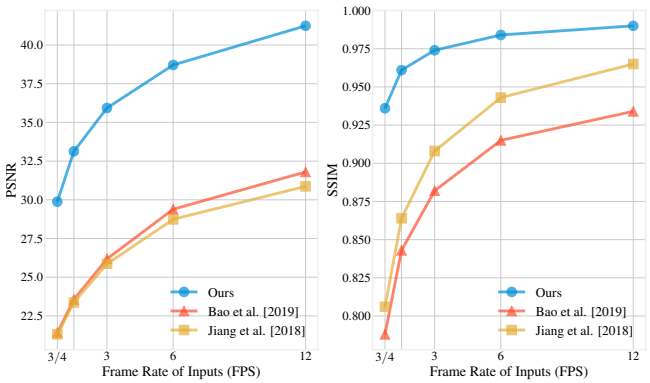


Fig. 10. Quality comparison of our approach with other methods at different input frame rates. Lower frame rate means larger displacement between successive frames. Our method outperforms all other methods by a large margin by taking advantage of the guided sketch.

rate, the lower the resulting quality). 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. Figure 12 shows an example of our results with different input frame rates. Moreover, our method has the flexibility to generate different inbetween frames by using different sketches from the same two input frames, whereas other methods can only generate one deterministic result. Another challenge for our method is to estimate the flow between two images in different domains (i.e., sketch and cartoon image). We show a qualitative comparison in Figure 11. Both Jiang et al. [2018] and Bao et al. [2019a] introduce artifacts in the results while our method achieves higher interpolation quality.

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 [Wang et al. 2018], a state-of-the-art conditional generative adversarial networks and a sketch colorization method [Zhang et al. 2018] 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 choose the cartoon frame as the reference to colorize the sketch.

As shown in Figure 13, the sketch colorization method is unable to handle the complex background and tends to use a fixed color within a region, ignoring shading variation. Moreover, 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 folds 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 14. Our training set only contains one single sketch style synthesized by our algorithm. However, it still has the capability to generalize to rough sketches after simplification (Figure 1) or hand-drawn sketches (Figure 14, 15). Furthermore, our method has the flexibility to generate different results by providing different guided sketches as shown in Figure 15. 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.

The capabilities of our method can be summarized as follows:

(1) It generalizes to different cartoon styles;
(2) It generalizes to different sketch styles;
(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;
(5) It supports synthesizing the video frame by frame by drawing each individual sketch.

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 post-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

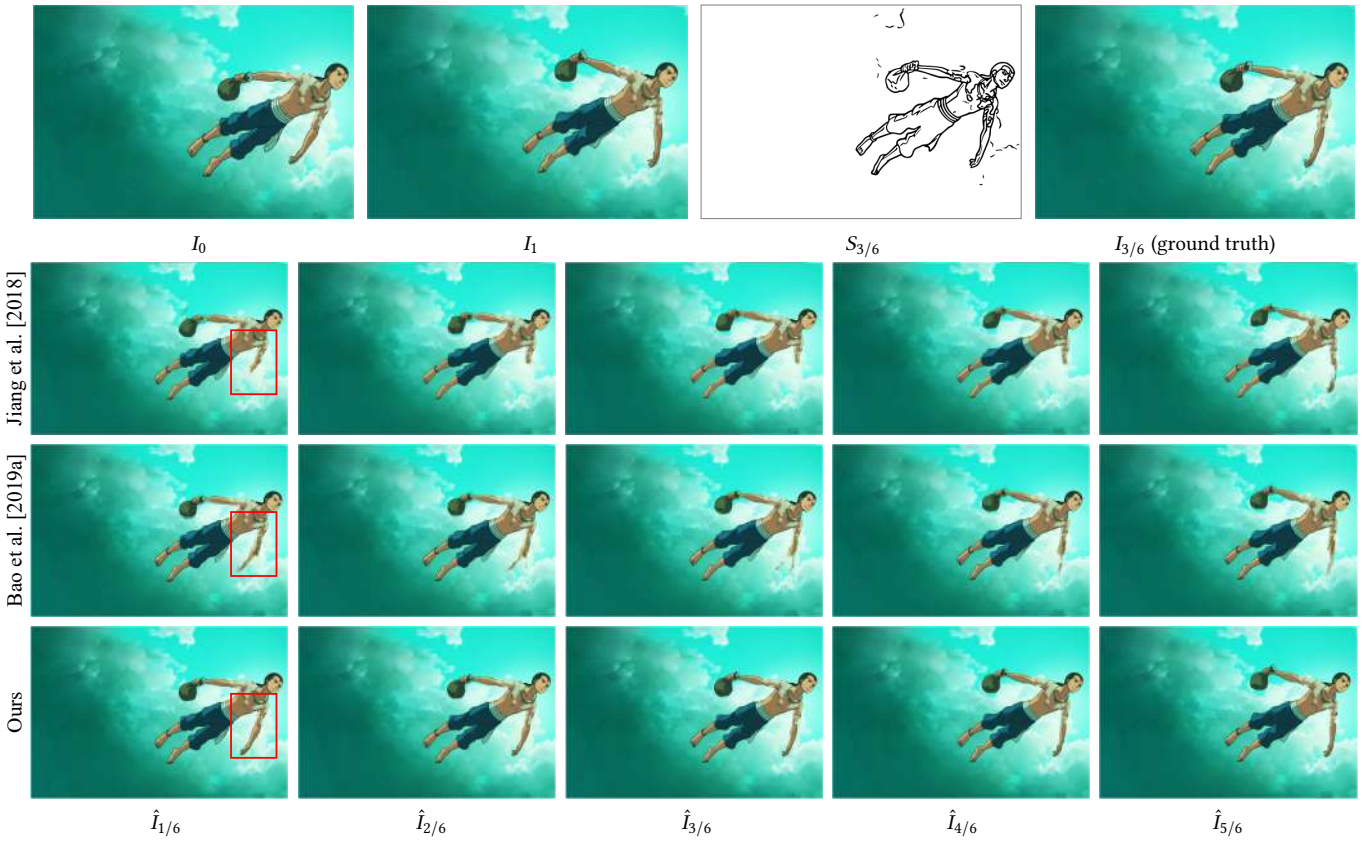


Fig. 11. Example results comparing our method (fourth row) with other frame interpolation approaches: Jiang et al. [2018] (second row) and Bao et al. [2019a] (third row). Compared to off-the-shelf methods, our method tends to better preserves the image content. ©B&T.

Fig. 12. Example results with different input frame rates (deformation ranges). For each result $\hat{I}_k$ in the second row, I_0 and I_{2k} from the first row are used as inputs. As the interval between input frames increases, artifacts begin to appear due to the larger motion. ©2:10 AM Animation.

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

Kfir Aberman, Jing Liao, Mingyi Shi, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. 2018. Neural best-buddies: Sparse cross-domain correspondence. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 69.

Fig. 13. Example results comparing image synthesis methods with our approach. ©B&T.

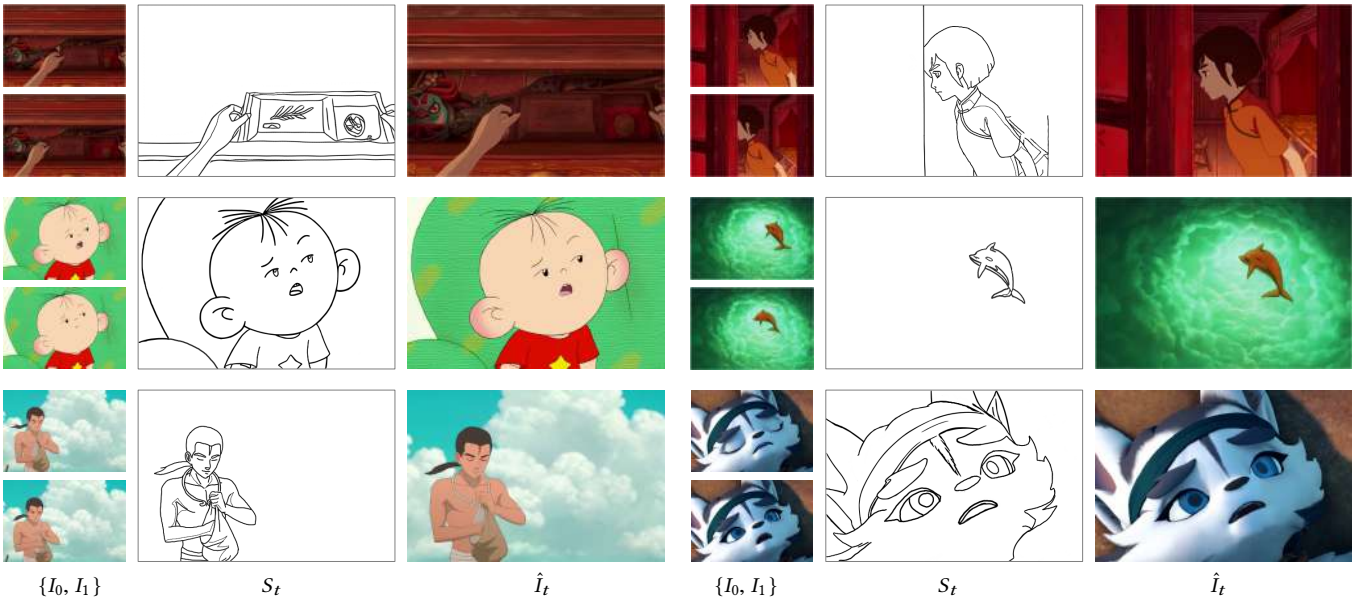


Fig. 14. Results from hand-drawn sketches with different cartoon and drawing styles. ©B&T, 2:10 AM Animation, SAFS.

Aayush Bansal, Yaser Sheikh, and Deva Ramanan. 2019. Shapes and context: in-the-wild image synthesis & manipulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2317–2326.

Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. 2019a. Depth-aware video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3703–3712.

Wenbo Bao, Wei-Sheng Lai, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. 2019b. MEMC-Net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. *IEEE transactions on pattern analysis and machine intelligence* (2019).

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. In *ACM Transactions on Graphics (ToG)*, Vol. 28. ACM, 24.

John L Barron, David J Fleet, and Steven S Beauchemin. 1994. Performance of optical flow techniques. *International journal of computer vision* 12, 1 (1994), 43–77.

Nir Ben-Zvi, Jose Bento, Moshe Mahler, Jessica Hodgins, and Ariel Shamir. 2016. Line-Drawing Video Stylization. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 18–32.

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

Tao Chen, Ming-Ming Cheng, Ping Tan, Ariel Shamir, and Shi-Min Hu. 2009. Sketch2photo: Internet image montage. In *ACM transactions on graphics (TOG)*, Vol. 28. ACM, 124.

Wengling Chen and James Hays. 2018. Sketchygan: Towards diverse and realistic sketch to image synthesis. In *Proceedings of the IEEE Conference on Computer Vision*

Fig. 15. 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.

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.

and Pattern Recognition. 9416–9425.

Jinsoo Choi and In So Kweon. 2019. Deep Iterative Frame Interpolation for Full-frame Video Stabilization. *arXiv preprint arXiv:1909.02641* (2019).

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. 2017. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 764–773.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. 2015. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 2758–2766.

Marek Dvorožnák, Wilmot Li, Vladimir G Kim, and Daniel Sỳkora. 2018. Toonsynth: example-based synthesis of hand-colored cartoon animations. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 167.

Mathias Eitz, James Hays, and Marc Alexa. 2012. How do humans sketch objects? *ACM Trans. Graph.* 31, 4 (2012), 44–1.

Mathias Eitz, Ronald Richter, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. 2011. Photosketcher: interactive sketch-based image synthesis. *IEEE Computer Graphics and Applications* 31, 6 (2011), 56–66.

Yağmur Güçlütürk, Umut Güçlü, Rob van Lier, and Marcel AJ van Gerven. 2016. Convolutional sketch inversion. In *European Conference on Computer Vision*. Springer, 810–824.

Xin Huang and Søren Forchhammer. 2012. Cross-band noise model refinement for transform domain Wyner–Ziv video coding. *Signal Processing: Image Communication* 27, 1 (2012), 16–30.

Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. 2017. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2462–2470.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. 2018. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9000–9008.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. 2017. Visual attribute transfer through deep image analogy. *arXiv preprint arXiv:1705.01088* (2017).

Ce Liu, Jenny Yuen, and Antonio Torralba. 2010. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence* 33, 5 (2010), 978–994.

Yifan Liu, Zengchang Qin, Tao Wan, and Zhenbo Luo. 2018. Auto-painter: Cartoon image generation from sketch by using conditional Wasserstein generative adversarial networks. *Neurocomputing* 311 (2018), 78–87.

Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. 2017. Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE International Conference on Computer Vision*. 4463–4471.

Gucan Long, Laurent Kneip, Jose M Alvarez, Hongdong Li, Xiaohu Zhang, and Qifeng Yu. 2016. Learning image matching by simply watching video. In *European Conference on Computer Vision*. Springer, 434–450.

Dhruv Mahajan, Fu-Chung Huang, Wojciech Matusik, Ravi Ramamoorthi, and Peter Belhumeur. 2009. Moving gradients: a path-based method for plausible image interpolation. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 42.

Simon Niklaus and Feng Liu. 2018. Context-aware synthesis for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1701–1710.

Simon Niklaus, Long Mai, and Feng Liu. 2017a. Video frame interpolation via adaptive convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 670–679.

Simon Niklaus, Long Mai, and Feng Liu. 2017b. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*. 261–270.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*. 8024–8035.

Tomer Peleg, Pablo Szekely, Doron Sabo, and Omry Sendik. 2019. IM-Net for High Resolution Video Frame Interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2398–2407.

Tiziano Portenier, Qiyang Hu, Attila Szabo, Siavash Arjomand Bigdeli, Paolo Favaro, and Matthias Zwicker. 2018. Faceshop: Deep sketch-based face image editing. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 99.

Anurag Ranjan and Michael J Black. 2017. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4161–4170.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. 2017. Scribbler: Controlling deep image synthesis with sketch and color. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5400–5409.

Peter Selinger. 2015. Portrace. 6, 7 (2015). http://potrace.sourceforge.net

Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. 2018. Mastering sketching: adversarial augmentation for structured prediction. *ACM Transactions on Graphics (TOG)* 37, 1 (2018), 1–13.

Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. 2016. Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.

Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. 2018. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8934–8943.

Daniel Sỳkora, John Dingliana, and Steven Collins. 2009. Lazybrush: Flexible painting tool for hand-drawn cartoons. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 599–608.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8798–8807.

Manuel Werlberger, Thomas Pock, Markus Unger, and Horst Bischof. 2011. Optical flow guided TV-L 1 video interpolation and restoration. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer, 273–286.

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

Jiyan Wu, Chau Yuen, Ngai-Man Cheung, Junliang Chen, and Chang Wen Chen. 2015. Modeling and optimization of high frame rate video transmission over wireless networks. *IEEE Transactions on Wireless Communications* 15, 4 (2015), 2713–2726.

Menghan Xia, Xueting Liu, and Tien-Tsin Wong. 2018. Invertible grayscale. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 246.

Saining Xie and Zhuowen Tu. 2015. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*. 1395–1403.

Jun Xing, Li-Yi Wei, Takaaki Shiratori, and Koji Yatani. 2015. Autocomplete hand-drawn animations. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–11.

Hongsheng Yang, Wen-Yan Lin, and Jiangbo Lu. 2014. Daisy filter flow: A generalized discrete approach to dense correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3406–3413.

Wenwu Yang, Hock-Soon Seah, Quan Chen, Hong-Ze Liew, and Daniel Sỳkora. 2018. FTP-SC: Fuzzy Topology Preserving Stroke Correspondence. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 125–135.

Ran Yi, Yong-Jin Liu, Yu-Kun Lai, and Paul L Rosin. 2019. APDrawingGAN: Generating Artistic Portrait Drawings from Face Photos with Hierarchical GANs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 10743–10752.

Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. 2017. Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 472–480.

Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. 2018. Two-stage sketch colorization. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 261.

Pan Zhang, Bo Zhang, Dong Chen, Lu Yuan, and Fang Wen. 2020. Cross-domain Correspondence Learning for Exemplar-based Image Translation. *arXiv preprint arXiv:2004.05571* (2020).

Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I Chang, Yan Xu, et al. 2020. MaskFlownet: Asymmetric Feature Matching with Learnable Occlusion Mask. *arXiv preprint arXiv:2003.10955* (2020).

Haichao Zhu, Xueting Liu, Tien-Tsin Wong, and Pheng-Ann Heng. 2016. Globally optimal toon tracking. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–10.