Joint Stroke Tracing and Correspondence for 2D Animation

HAORAN MO, Sun Yat-sen University, China
CHENGYING GAO[*], Sun Yat-sen University, China
RUOMEI WANG, Sun Yat-sen University, China

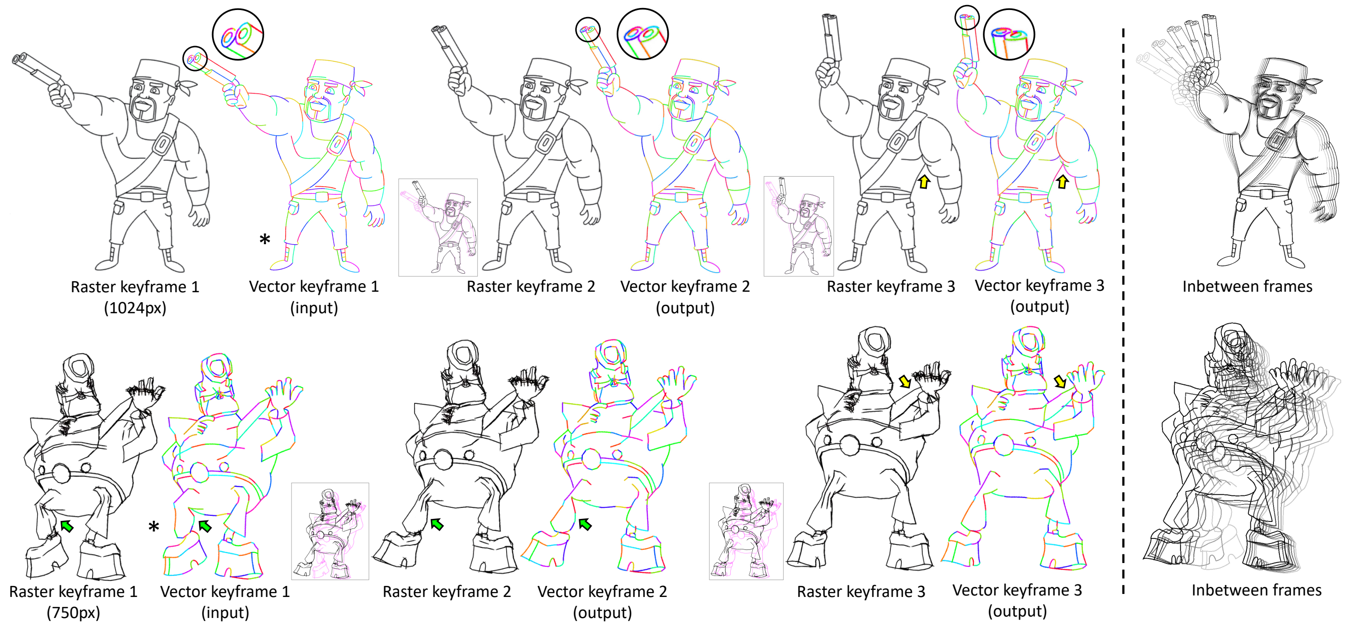

Fig. 1. Given consecutive raster keyframes and a single vector drawing from the starting keyframe only (with marks '∗'), our method generates vector images for the remaining keyframes with one-to-one stroke correspondence. The framework trained with clean line drawings generalizes well to rough sketches. The generated results can be directly imported into an inbetweening system to produce inbetween frames to form 2D animation (see supplemental video). Sub-figures in boxes show differences between consecutive frames, with magenta for the previous and black for the current. Gunman from paper [Yang et al. 2018] is courtesy of Eugene Babich ©2018 John Wiley & Sons Ltd. Bigvegas2 is from [Shugrina et al. 2019].

To alleviate human labor in redrawing keyframes with ordered vector strokes for automatic inbetweening, we for the first time propose a joint stroke tracing and correspondence approach. Given consecutive raster keyframes along with a single vector image of the starting frame as a guidance, the approach generates vector drawings for the remaining keyframes while ensuring one-to-one stroke correspondence. Our framework trained on clean line drawings generalizes to rough sketches and the generated results can be imported into inbetweening systems to produce inbetween sequences. Hence, the method is compatible with standard 2D animation workflow. An adaptive spatial transformation module (ASTM) is introduced to handle non-rigid motions and stroke distortion. We collect a dataset for training, with 10k+ pairs of raster frames and their vector drawings with stroke correspondence. Comprehensive validations on real clean and rough animated frames manifest the effectiveness of our method and superiority to existing methods.

[*]Corresponding author.

Authors' addresses: Haoran Mo, Sun Yat-sen University, Guangzhou, China, mohaor@mail2.sysu.edu.cn; Chengying Gao, Sun Yat-sen University, Guangzhou, China, mcsgcy@mail.sysu.edu.cn; Ruomei Wang, Sun Yat-sen University, Guangzhou, China, isswrm@mail.sysu.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 0730-0301/2024/1-ART1
https://doi.org/XXXXXXX.XXXXXXX

CCS Concepts: • **Computing methodologies** → **Parametric curve and surface models**.

Additional Key Words and Phrases: stroke tracing, stroke correspondence, automatic inbetweening, 2D animation

ACM Reference Format:
Haoran Mo, Chengying Gao, and Ruomei Wang. 2024. Joint Stroke Tracing and Correspondence for 2D Animation. *ACM Trans. Graph.* 1, 1, Article 1 (January 2024), 17 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Generating inbetween frames from keyframes is a fundamental component in keyframe-based 2D animation production. As shown in Fig. 2, in a conventional workflow with an inbetweening product (*e.g.*, CACANi [2020]), artists usually draw raster images or load prepared ones as drafts. These images are mostly rough, requiring the artists to redraw or clean up with vector strokes as keyframes.

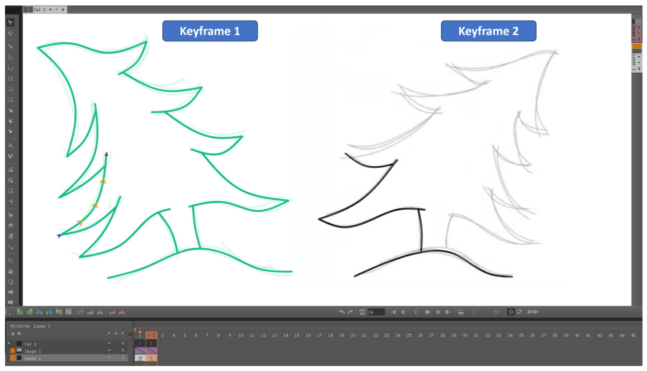

Fig. 2. Interface of an inbetweening product CACANi [2020], where users are required to redraw all the drafts with vector strokes in the same order as keyframes. It provides a navigator for each stroke (arrows in keyframe 1) to guide the redrawing of a corresponding stroke in another keyframe. Tree images ©2011 ACM, Inc.

While tracing, one-to-one correspondence between strokes in the keyframes should be ensured to allow subsequent stroke interpolation process to produce the inbetweens. Given that a real animation may contain quite a few keyframes and each consists of plentiful strokes [Jiang et al. 2020], the process above is tedious and time-consuming. To reduce the human labor in this fundamental step for the automatic inbetweening, we for the first time introduce a task called *joint stroke tracing and correspondence.* As shown in Fig. 1, given a series of raster keyframes, this task requires only a single vector image manually traced for the starting frame, and then automatically generates vector drawings for the remaining keyframes with one-to-one stroke correspondence. While the single vector input is necessary as a reference, the workload from the artists is largely reduced compared with the conventional full-sequence redrawing process, which accelerates the 2D animation pipeline.

We extend a learning-based line drawing vectorization framework [Mo et al. 2021] to this fundamental task of inbetweening and animation, which simultaneously computes vectorization and, more importantly, stroke correspondence. Although trained on synthetic clean line drawing data, the framework generalizes well to real animated frames with high complexity, and even rough sketches with overdrawn and uneven lines. The generated results, albeit with minor artifacts induced by the topology structure change or occlusion (see yellow arrows in Fig. 1), can be imported into interactive inbetweening systems to make a quick modification (for 1–3 strokes) and produce natural inbetweens. Hence, our approach is practical and compatible with the standard 2D animation workflow to promote the productivity.

Continuous non-rigid motions in consecutive keyframes tend to give rise to incremental spatial variation (see boxes in Fig. 1) and geometric distortion of strokes (see green arrows). The issues notably increase the difficulty in predicting corresponding strokes with proper proportion of length and orientation. To tackle the challenges, we propose an adaptive spatial transformation module (ASTM), which helps to find correspondence by reducing the variation and increasing the similarity between strokes in local patches via predicted transformations. The module is trained in a self-supervised manner without relying on ground-truth transformation parameters.

Due to the lack of a dataset with vector stroke correspondence for training, we collect one containing consecutive line drawing frames and their vector drawings along with the annotation of stroke correspondence. The data are synthesized via an automatic algorithm that mimics the real animated keyframes with non-rigid motions. Then, manual review is done to filter out undesired synthesis, which guarantees the quality of the data while alleviating the burden of manual tracing and correspondence identification. The dataset is expected to facilitate the research of automatic inbetweening.

We evaluate our approach with real clean and rough animated frames of various resolutions and with fairly complex motions. The results in the ablation study and comparisons with existing methods corroborate the effectiveness of our framework. The generated results are imported into an inbetweening product CACANi [2020] to obtain inbetween frames forming the 2D animation [1].

The main contributions of this work are summarized as follows:

(1) A joint stroke tracing and correspondence framework for consecutive raster keyframes according to a vector input of the starting frame. While trained with clean line drawings, it generalizes well to rough sketches.
(2) An adaptive spatial transformation module (ASTM) based on self-supervised learning that helps to cope with keyframes with fairly large motions or stroke distortion.
(3) A dataset including 10k+ pairs of raster frames and their vector drawings with stroke correspondence, collected via an automatic algorithm followed by manual review.
(4) Extensive validations on high-resolution real drawings with clean or rough sketches. By importing the results into an inbetweening product, 2D animation is created.

2 RELATED WORK

2.1 2D Animation and Inbetweening

The problem of computer-aided 2D animation and automatic inbetweening has been studied for about fifty years [Catmull 1978; Dalstein et al. 2014, 2015; Even et al. 2023; Fekete et al. 1995; Kort 2002; Miura et al. 1967; Reeves 1981], and is still an open and challenging problem. While a number of commercial solutions have been developed, most are designed for either hand-drawn animation without support for automatic inbetweening [Blender 2023; DWANGO 2023; Toon Boom 2022; TVPaint 2023], or specific use cases such as dynamic expressions or rigged characters [Adobe 2022; Live2D 2023; Reallusion 2023]. CACANi [2020] is tailored for automatic inbetweening, while requiring users to draw the corresponding strokes in order between keyframes to facilitate stroke interpolation. As a result, these systems are not directly applicable to our task. In academia, plenty of works have been introduced and they can be classified into two main lines [Jiang et al. 2020]: raster-based and vector-based approaches.

The *raster-based* methods first construct associated templates or embeddings (*e.g.*, triangular meshes or lattice grids) for objects in the raster keyframes, and then adopt as-rigid-as-possible (ARAP)

[1]The source code can be found at https://github.com/MarkMoHR/JoSTC.

techniques [Alexa et al. 2000; Igarashi et al. 2005; Sýkora et al. 2009] for shape deformation [Dvorožnák et al. 2018; Smith et al. 2023]. Methods in this category are more suitable for rigid animations [Dvorožňák et al. 2017; Su et al. 2018], while limited in typical 2D animations with non-rigid nature and occlusions.

The *vector-based* approaches focus on two basic problems: correspondence and interpolation of the parametric strokes in 2D line drawings. Here we mainly discuss the correspondence problem our approach targets. For the interpolation, we refer readers to the literatures [Even et al. 2023; Jiang et al. 2020; Yang 2017]. Early systems require users to manually identify correspondences of the strokes between keyframes [Burtnyk and Wein 1975; Durand 1991; Reeves 1981]. To alleviate the tedious work, a number of automatic correspondence and inbetweening systems have been developed and introduce different measurements of stroke similarity and matching algorithms [Even et al. 2023; Kort 2002; Whited et al. 2010; Yang 2017; Yang et al. 2018]. They require users to input keyframes with vector strokes, which are incompatible with our task where only a single vector drawing from the starting frame is provided. Our setting largely reduces the workload of manually redrawing all the keyframes and increases productivity.

A recent geometrized inbetweening method named AnimeInbet [Siyao et al. 2023] extracts endpoints from raster line drawings to build graphs, and reformulates the inbetweening task as a vertex correspondence and reposition problem. Due to imperfect predictions of correspondence and visibility for a large number of vertices, it easily produces discontinuous lines and flickering noisy points. In contrast, vector strokes in our approach are more compact and benefit the line continuity.

2.2 Pixel-wise Correspondence

Per-pixel correspondence based on feature matching [Jiang et al. 2021b; Sarlin et al. 2020] is also commonly used in the field of 2D animation [Navarro et al. 2021; Xu et al. 2022a; Yu et al. 2020]. Each pixel is encoded into a feature descriptor for a similarity-based registration. Other works rely on optical flow prediction [Sun et al. 2018; Teed and Deng 2020] to associate the pixels between keyframes [Chen and Zwicker 2022; Li et al. 2021; Narita et al. 2019; Siyao et al. 2021]. Inbetweens are produced by warping the keyframes with the predicted pixel-wise correspondence maps. These solutions possibly give rise to discontinuous strokes and noises for line drawings due to the separated prediction of discrete pixels. In contrast, the continuous and neat representation of vector strokes benefits high-quality inbetweening.

2.3 Vectorization and Sketch Simplification

The methods of vectorization [Bessmeltsev and Solomon 2019; Favreau et al. 2016; Guo et al. 2019; Guţan et al. 2023; Liu et al. 2022; Mo et al. 2021; Noris et al. 2013; Puhachov et al. 2021; Stanko et al. 2020] and rough sketch simplification [Liu et al. 2023, 2018, 2015; Simo-Serra et al. 2018a,b, 2016; Xu et al. 2019; Yan et al. 2020] are proposed for a single input image, and probably produce vector drawings with large variations in stroke number and order for different inputs. In our task, such variations tend to be detrimental to the one-to-one stroke correspondence.

A vectorization approach Virtual-Sketching [Mo et al. 2021] is close to our framework in the recurrent manner for stroke prediction and the patch-based modeling scheme. However, our approach works towards a largely different task on stroke tracing while ensuring stroke correspondence simultaneously. The additional correspondence process requires computation of appropriate length proportion of the strokes especially when large geometric distortion occurs. Hence, modeling more comprehensive information in the patches, *e.g.*, context of the strokes, is crucial in our task. We propose a self-supervised learning-based transformation module to tackle this challenge.

2.4 Datasets for 2D Animation

Several datasets exist in the field of 2D animation, while most are constructed for specific use cases, such as cartoon generation [Siarohin et al. 2019], animation video interpolation [Siyao et al. 2021], animation head reenactment [Kim et al. 2022], figure detection and pose estimation [Smith et al. 2023], etc. CreativeFlow+ dataset [Shugrina et al. 2019] contains animated sequences rendered from animated 3D scenes or characters, along with per-pixel correspondence labels such as optical flow and correspondence maps. AnimeRun dataset [Siyao et al. 2022] synthesizes 2D animation frames from open-source 3D movies and provides pixel-wise (optical flow) and region-wise (segment matching) correspondences. Compared to them, our collected dataset is the first large-scale line drawing dataset with one-to-one vector stroke correspondence annotated for consecutive raster frames, which applies to the automatic inbetweening task.

3 JOINT STROKE TRACING AND CORRESPONDENCE

3.1 Overview

Given consecutive raster keyframes and a vector drawing of the starting frame as a guidance, our approach predicts parametrized strokes for the remaining keyframes while ensuring one-to-one stroke correspondence simultaneously, which is a fundamental step in automatic inbetweening and 2D animation. We follow the same terminology that a drawing is formed by several *stroke chains* (*i.e.*, long curves), each of which consists of one or more continuously connected *strokes* [Jiang et al. 2020]. As illustrated in Fig. 3-(a), our proposed framework essentially applies to two successive keyframes, denoted as reference and target. When taking more keyframes into account, the generated vector result is adopted as a second reference and propagated to the next frame.

The framework processes the stroke chains one after another. It is mainly comprised of a starting point matching model and a stroke tracing and correspondence model. The former works once at the beginning of each stroke chain, and uses the starting point in the reference as a guidance to locate the matchup in the target image. With the matched starting position, the latter model produces the corresponding vector strokes on the chain one by one according to the reference vector drawing. At the end of a stroke chain, we switch to next chain and repeat the two kinds of operations above. We integrate the concept of window-based processing in [Mo et al. 2021] into our framework by modeling local patches cropped from

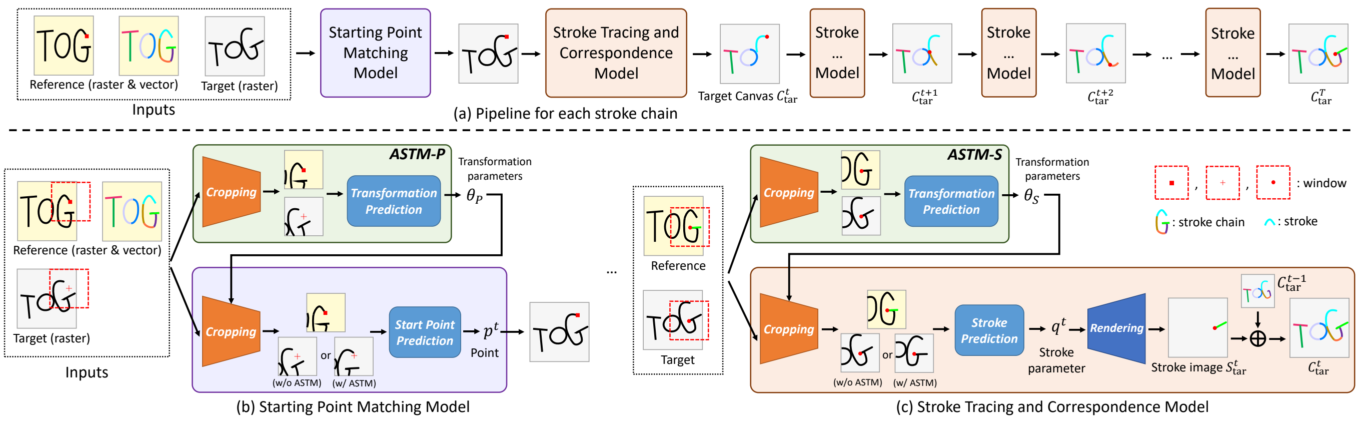
Reference (raster & vector)
Target (raster)
Inputs
Starting Point Matching Model
Stroke Tracing and Correspondence Model
Target Canvas C^t_{tar}
Stroke ... Model
C^{t+1}_{tar}
C^{t+2}_{tar}
C^T_{tar}
(a) Pipeline for each stroke chain
ASTM-P
Cropping
Transformation Prediction
Transformation parameters θ_P
Start Point Prediction
p^t Point
(w/o ASTM) or (w/ ASTM)
(b) Starting Point Matching Model
Reference
Target
ASTM-S
Transformation parameters θ_S
Stroke Prediction
q^t Stroke parameter
Rendering
Stroke image S^t_{tar}
C^{t-1}_{tar}
C^t_{tar}
: window
: stroke chain
: stroke
(c) Stroke Tracing and Correspondence Model

Fig. 3. Our framework takes as inputs consecutive raster frames, denoted as reference and target, along with a vector drawing of the reference containing several stroke chains (*i.e.*, long curves), each of which comprises connected strokes. It performs a joint stroke tracing and correspondence task by generating corresponding vector strokes one by one (a). It consists of two models: one for matching starting point of each stroke chain (b), the other for predicting parameters of the associated strokes (c). The whole process works in a local view based on patches cropped by windows. A proposed plug-and-play adaptive spatial transformation module (ASTM) is integrated into the two models to handle large motions or stroke distortion.

the input images (Fig. 3-(b & c)), allowing our approach to work on images of arbitrary resolutions and high complexity.

To handle fairly complex motions that may induce spatial variations or deformations between strokes in the reference and target patches, we propose an adaptive spatial transformation module (ASTM) to reduce the variations by ensuring content alignment between the patches through spatial transformations, which brings obvious performance boosts to the subsequent predictions of starting points or strokes. The ASTM is a plug-and-play module trained separately, so we introduce the starting point model and the stroke model without the ASTM in Sec. 3.2 and 3.3 first, and then details of the ASTM in Sec. 3.4.

3.1.1 Stroke Representation. Following the stroke types in inbetweening products (*e.g.*, CACANi [2020]), we use cubic Bézier curves $q^t = (x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3)$, comprised of two endpoints and two control points, to represent a stroke at step t. The strokes lie within the processing window, so we define a local coordinate system $[-1, +1]$ with respect to the window area for the stroke parameters, while they can be mapped back to the global coordinates given center position and size of the window.

3.2 Starting Point Matching Model

Due to the non-rigid motions, starting points of the stroke chains tend to vary between the reference and the target frames. Therefore, this model performs a starting point matching process per stroke chain. Below we first introduce the original pipeline of this model without the ASTM-P in Fig. 3-(b).

Given two raster frames and a starting point from a stroke chain in the reference, we first crop the patches using the aligned cropping technique from [Mo et al. 2021], via a window with a center position and a pre-defined size proportional to the image size. We use the given reference starting point as the center to locate the cropped patches of both the reference and the target images. They are then used as inputs for predicting the matched starting point in the target

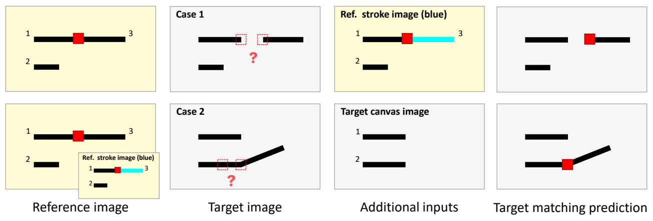

Fig. 4. In starting point matching model, some visual signals should be adopted as inputs to help in determining the predictions of matching point. Here we show target images in two cases using the same reference. In case 1 with stroke connectivity change, when given only the reference image and the target one, it is difficult to choose the left or the right point in the target image as the matchup. Here a reference ('ref.' for short) stroke image connecting to the reference starting point helps to reject the left candidate. In case 2 with changes in both stroke connectivity and shape, even after using the reference stroke image as the indicator, it is still not easy to choose between the candidates. Here the target canvas image (recording all the predicted strokes thus far) helps as it avoids the matchup in drawn area, which therefore filters out the left candidate and remains the right one.

patch. As illustrated in Fig. 4, we notice additional visual signals can benefit the prediction. In the presence of stroke connectivity change (case 1), a *reference stroke image* helps to indicate where the matched starting point should be by telling where the subsequent stroke tracing should start from. In more complex cases where both stroke connectivity and shape changes exist (case 2), the adopted reference stroke image is insufficient here and a *target canvas image* facilitates the prediction by limiting the search space within the undrawn area. The two kinds of images are cropped into patches and adopted as additional inputs to the model.

A starting point prediction network is employed to encode the cropped patches, which is built upon convolutional layers followed by multi-layer perceptrons (MLP). The reference and the target patches are processed by separated convolutional layers, and their features are fused in the MLP. The network predicts the matching starting point p^t in the target patch. It lies in a local coordinate system $[-1, +1]$ with respect to the window area, which can be mapped back to the global coordinate system for usage in the following stroke tracing and correspondence process.

3.3 Stroke Tracing and Correspondence Model

This model produces vector strokes in correspondence with those in the reference in a recurrent manner, by generating the strokes one by one. Below we first introduce its original pipeline for each stroke without the ASTM-S in Fig. 3-(c).

We integrate the ideas of window-based mechanism and differentiable rendering in Virtual-Sketching [Mo et al. 2021] into this model. With the reference and the target frames as inputs, a window is used to crop local patches from them, and then the vector stroke q^t is predicted based on the patches. Afterwards, the parametrized stroke is rendered onto a raster canvas and the window moves to the end point of the stroke to start a new iteration. The canvas after the final iteration stores all the predicted strokes. This simulates the tracing procedure for the target frame.

Specifically, at each step t, we use a cropping window with a size proportional to the length of the reference stroke. Similar to the starting point model, we also consider patches from the reference stroke and the target canvas to provide necessary information. A stroke prediction network encodes these patches with separated convolutional layers and fuses the features through a recurrent neural network (RNN). It finally predicts the parameters $q^t = (x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3)$ of the corresponding stroke in a local coordinate system $[-1, +1]$ with respect to the window area. After mapping q^t back to the global coordinate system, a differentiable renderer DiffVG [Li et al. 2020] is employed to convert the vector parametrization into a full-size stroke image S^t_{tar}, which is then added to the previous canvas image C^{t-1}_{tar} to form a new one C^t_{tar}. The pipeline repeats until all the strokes on the chain are processed.

3.4 Adaptive Spatial Transformation Module (ASTM)

While our framework is able to work on simple cases with the original pipeline introduced above, it tends to suffer noticeable performance degradation in real cases with complex motions involved in the strokes. As shown in Fig. 5, in prediction of matching starting point (top row), the contents in the cropped patches between the reference and the target vary dramatically due to the large motion, and the expected matchup in the target is even outside the patch (a & b). Consequently, the derived prediction inevitably fails due to the limited window view (c). When predicting the corresponding stroke (bottom row), the model with the original pipeline is found to be fragile in cases with large geometric deformation (g & h), and tends to predict a stroke that misaligns with the target drawing (i).

To overcome the issues caused by the complicated motions and the limited view of local windows, we propose a self-supervised learning-based *adaptive spatial transformation module (ASTM)*, which predicts a spatial transformation, including a combination of translation, scaling and rotation, for the initial processing window in the target to enable an adaptive local view (Fig. 5-(d or j)). With the transformed window, the cropped patch of the target is similar in content to the reference patch. Such a content alignment process reduces the spatial variations caused by the deformation and provides consistent contexts to find corresponding starting points with similar spatial characteristics (a & e) or corresponding strokes with correct orientation and proportion of length (g & k). The transformation is invertible and thus allows the prediction to be mapped back to coordinates of the original target image (f or l).

3.4.1 *Architecture of ASTM.*

The ASTM is designed in a plug-and-play manner and can be integrated into both the starting point matching model and the stroke tracing and correspondence model, as shown in Fig. 3-(b & c). We denote the module for the point model as ASTM-P and that for the stroke model as ASTM-S, as they differ in output transformation parameters θ. ASTM-P outputs the parameters θ_P including translation, scaling and rotation, while θ_S from ASTM-S contains only the scaling and rotation, since additional shifting for the first endpoint is not needed when generating connected strokes. The two modules share exactly the same pipeline and network architecture.

As shown in Fig. 3, the ASTMs take as inputs the reference and the target images, and crop patches with the initial window akin to the start-up processes of the original starting point matching model and stroke tracing and correspondence model. Then, a transformation prediction network built with convolutional layers followed by an MLP generates the transformation parameters $\theta_P = (t_x, t_y, s_x, s_y, \alpha) \in \mathbb{R}^5$ or $\theta_S = (s_x, s_y, \alpha) \in \mathbb{R}^3$. $(t_x, t_y) \in [-1, +1]$ are the translation offsets with respect to the window area. $(s_x, s_y) \in [0.2, 2.0]$ are the non-isotropic scaling parameters with pre-defined upper and lower bounds. $\alpha \in [-\pi, +\pi]$ is the rotation angle. Prediction of correct transformation parameters is key to the patch alignment. We train both ASTM modules in a self-supervised scheme based on a shape similarity measurement, without relying on ground-truth transformation parameters. The training details are depicted in Sec. 3.5.1.

3.4.2 *Integration into Main Workflow.*

With the predicted transformation, we incorporate it into the main workflow. The integration includes two processes: (1) *transformed cropping* that works with the cropping operation for the target image by using a transformed window to crop a patch with similar content to the reference, as illustrated in Fig. 5-((d to e) or (j to k)), and (2) *inverse transformation* that maps the prediction, *i.e.*, the starting point or the stroke, from a transformed coordinate system (local orthogonal) back to the global one, as illustrated in Fig. 5-((e to f) or (k to l)). Both processes are differentiable and thus amenable to integration in an end-to-end training model.

Transformed Cropping. In this process, we rely on two operations in Spatial Transformer Networks (STN) [Jaderberg et al. 2015]: parameterized sampling grid generation and differentiable image sampling. First, a transformation $\mathcal{T}_\theta$ of any parameterized form is used to generate a sampling grid, *i.e.*, a set of points, as illustrated in Fig. 6. Then, by sampling from the input image at the grid points in a differentiable manner, an output image forming a warping of the input is obtained. We utilize the transformation predicted by our ASTMs to form $\mathcal{T}_\theta$.

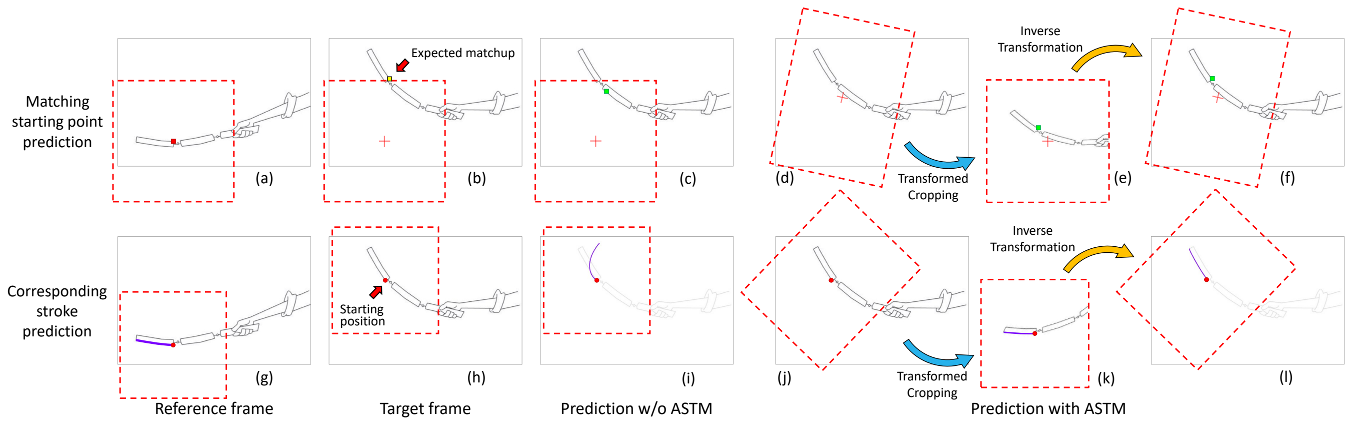

Fig. 5. Issues of large motions between consecutive keyframes, which lead to situations where the expected matched starting point in the target is outside the processing window (b), and geometric deformation induces spatial variations (h). Our adaptive spatial transformation module (ASTM) handles the motions by predicting transformations to align the patches in content (a & e, g & k). Green dots denote the predicted points. Stick images from paper [Yang 2017] are courtesy of Jie Li ©2018, IEEE.

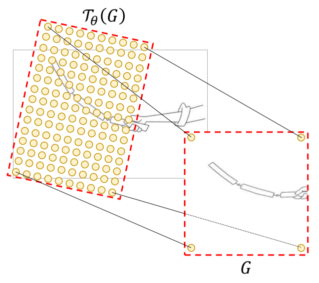

Fig. 6. Sampling grid generation in the transformed cropping process. Stick image from paper [Yang 2017] is courtesy of Jie Li ©2018, IEEE.

Specifically, a regular grid $G = \{G_i\} \in \mathbb{R}^{h\times w}$ of pixels $G_i = (x_i', y_i')$ in a local orthogonal coordinate system is defined for the cropped target patch via the transformed window. h and w are the height and width from the initial window. The sampling grid $\mathcal{T}_\theta(G)$ in the original target image is a warping of the regular grid G with transformation $\mathcal{T}_\theta$. In our approach, $\mathcal{T}_\theta$ is represented with the transformation parameters $\theta = (t_x, t_y, s_x, s_y, \alpha)$ predicted by the ASTMs. For brevity, we omit the subscript of θ denoting ASTM-P or ASTM-S and use the 5-fold parameters as $(t_x, t_y) = (0, 0)$ can be used for θ_S. We define the transformation matrices for translation, scaling and rotation as:

$$M_t = \begin{bmatrix} 1 & 0 & \hat{t}_x \\ 0 & 1 & \hat{t}_y \\ 0 & 0 & 1 \end{bmatrix}, M_s = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}, M_r = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

respectively. As the predicted $(t_x, t_y) \in [-1, +1]$ is a relative offset with respect to the center position (p_x, p_y) of the initial cropping window in size $h \times w$, it should be mapped back to the global coordinate system before formulating the M_t:

$$\hat{t}_x = t_x \times w/2 + p_x, \qquad \hat{t}_y = t_y \times h/2 + p_y. \tag{2}$$

The combination of these transformations forms a 2D affine transformation $A_\theta = M_r \cdot M_s \cdot M_t$, representing $\mathcal{T}_\theta$. Then, we generate the sampling grid by establishing the point-wise transformation:

$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} = \mathcal{T}_\theta(G_i) = A_\theta \begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix}, \tag{3}$$

where (x_i, y_i) is the position in the original target image that defines the sampling grid point. With this sampler, we subsequently perform a differentiable image sampling following the operation in the STN [Jaderberg et al. 2015], which takes as inputs the original target image and the set of sampling points, and produces the cropped patch. We recommend readers to refer to the STN paper for details.

Inverse Transformation. The cropped target patch formed by the regular grid G lies in a local orthogonal coordinate system (Fig. 5-(e or k)). Hence, the predictions based on the cropped patch, including the matched starting point p^t or the control/end points in q^t of the corresponding strokes, ought to be mapped back to the global coordinate system. As in the sampling grid generation operation above, we have established a point-wise transformation (Eq.(3)) from a point in the transformed patch (*i.e.*, the regular grid) to a point in the original target image, the inverse transformation can be performed following Eq.(3) directly on the predicted points. The starting point $p^t \in [-1, +1]$ and the control/end points in $q^t \in [-1, +1]$ are relative positions to the center of the window, and thus they are converted into global positions prior to the inverse transformation.

3.5 Training

The starting point and stroke models in our framework are trained separately. The ASTMs (ASTM-P and ASTM-S) are plug-and-play and the integration is fully differentiable, so we train them separately prior to the two models. When training the stroke tracing and correspondence model, the ASTM-S is loaded and frozen. While the starting point matching model is trained without using ASTM-P in order to make it robust to training data. Then in inference phase on real complicated cases, the ASTM-P is integrated to improve generalization ability and gain further performance boosts.

3.5.1 *Training of ASTMs.*

The ASTM-P and ASTM-S are trained in a similar procedure based on a *self-supervised learning* scheme, without relying on ground-truth transformation parameters. Since the goal is to make the patches similar in content, we perform the transformed cropping with the predicted transformation to obtain a new target patch and then measure its similarity to the original reference patch. The key problem here is to impose a proper shape similarity measurement. We find a perceptual distance introduced in [Mo et al. 2021], a kind of structural loss, works very well. It relies on a VGG-16 model [Simonyan and Zisserman 2015] trained on a sketch classification task on QuickDraw dataset [Ha and Eck 2018], so it is sensitive to line drawing data.

The perceptual network ϕ takes two images I_a and I_b and produces intermediate activation map $\phi_j(\cdot) \in \mathbb{R}^{D_j \times H_j \times W_j}$ of layer j, where H_j and W_j are the height and width of the activation map and D_j is the number of channels. The perceptual loss of layer j is formulated as:

$$\mathcal{L}_{\text{perc}}^{j}(I_a, I_b) = \frac{1}{D_j \times H_j \times W_j} \left\| \phi_j(I_a) - \phi_j(I_b) \right\|_1. \tag{4}$$

Given a cropped reference patch P_{ref} and a target patch $A_\theta(P_{\text{tar}})$ derived from the transformation A_θ and the original cropping P_{tar}, we obtain the perceptual loss $\mathcal{L}_{\text{perc}}^{j}(P_{\text{ref}}, A_\theta(P_{\text{tar}}))$ at layer j. The loss computed from a single layer `relu3_3` is shown to be sufficient to ensure the rough alignment and additional layers lead to degradation. While trained on clean line drawing images, the ASTMs are generalizable to rough drawings and thus empower the whole framework to cope with real rough sketches.

3.5.2 *Training of Starting Point Matching Model.*

This model predicts the matched starting point $p^t \in [-1, +1]$ in the target patch. As mentioned above, it is trained in the absence of ASTM-P. Our collected dataset contains vector drawings of the raster frames along with stroke correspondence information, so ground-truth matchups exist and allow a supervised training. Given the ground-truth point $\hat{p}^t$, we adopt L1 distance as the loss:

$$\mathcal{L}_{\text{match}}(p^t, \hat{p}^t) = \left\| p^t - \hat{p}^t \right\|_1. \tag{5}$$

Besides, we introduce a *non-stroke point penalty* that encourages the model to predict points lying on stroke pixels by penalizing predictions in the empty space. We simply use the differentiable cropping operation in our framework to produce a single-pixel patch I_{p^t} from the target given the predicted point, which indicates the pixel value in that position. When assuming a value 0 for the stroke in black and 1 the empty space in white, we define the penalty based on the pixel value as:

$$\mathcal{L}_{\text{penalty}}(p^t) = \begin{cases} I_{p^t}, & \text{if } I_{p^t} \geq \tau \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where τ is a threshold to distinguish between a stroke pixel or an empty one, and set to 0.9 empirically. The total loss is:

$$\mathcal{L}_{\text{start\_point}} = \mathcal{L}_{\text{match}} + \lambda_{\text{penalty}} \mathcal{L}_{\text{penalty}}, \tag{7}$$

where λ_{penalty} is a scalar and set to 0.1 during training.

3.5.3 *Training of Stroke Tracing and Correspondence Model.*

At each step t, starting from the center $q_0^t = (x_0, y_0) = (0, 0)$, the model predicts the control points $q_1^t = (x_1, y_1)$ and $q_2^t = (x_2, y_2)$ as well as the end point $q_3^t = (x_3, y_3)$ of the corresponding stroke q^t. They are in a local coordinate system $[-1, +1]$ and mapped back to global coordinates $Q^t = (Q_0^t, Q_1^t, Q_2^t, Q_3^t)$ for training. The vector parametrization data with stroke correspondence in our dataset enables a supervised training.

Instead of predicting and penalizing all the four points, we fix Q_0^t by using a real one from the dataset and making the model predict the remaining three points. This training strategy is compatible with the inference mode producing connected strokes, where Q_0^t is from the predicted Q_3^{t-1}. With this strategy, the training efficiency is significantly increased, and the framework generalizes well when switching to the inference mode. We adopt L1 distance to measure the difference between the prediction Q^t and ground truth $\hat{Q}^t$ at all steps $t = 1, ..., T$:

$$\begin{aligned} &\mathcal{L}_{\text{tracing-L1}}(\{Q^t\}, \{\hat{Q}^t\}) = \\ &\qquad \frac{1}{T} \sum_{t=1}^{T} \left\| (Q_1^t - \hat{Q}_1^t) + (Q_2^t - \hat{Q}_2^t) + (Q_3^t - \hat{Q}_3^t) \right\|_1, \end{aligned} \tag{8}$$

where $\hat{Q}_1^t$, $\hat{Q}_2^t$ and $\hat{Q}_3^t$ are the points in $\hat{Q}^t$.

In addition, we also adopt a raster-level loss to improve the fidelity of fine details in the generated stroke renderings, which further promotes performance of the parameter prediction. We follow the raster-level supervision in Virtual-Sketching [Mo et al. 2021], which passes the original target image I_{tar} and the final canvas C_{tar}^T into a QuickDraw-trained VGG-16 model to compute a perceptual loss with intermediate activation maps (Eq.(4)). We compute the loss from several layers including `relu1_2`, `relu2_2`, `relu3_3`, `relu4_3` and `relu5_1`, and adopt the loss value normalization technique in [Mo et al. 2021] to balance them, which forms the final raster-level loss $\mathcal{L}_{\text{tracing-raster}}(C_{\text{tar}}^T, I_{\text{tar}})$.

To maintain a balance between the vector-level and the raster-level losses, we also apply the value normalization technique to the $\mathcal{L}_{\text{tracing-L1}}$ to produce a normalized one $\mathcal{L}_{\text{tracing-L1-norm}}$. Finally, the total loss is formulated as:

$$\mathcal{L}_{\text{stroke}} = \mathcal{L}_{\text{tracing-L1-norm}} + \mathcal{L}_{\text{tracing-raster}}. \tag{9}$$

4 DATASET

4.1 Overview

To the best of our knowledge, we for the first time collect a large-scale line drawing dataset containing 10k+ consecutive raster frames along with their vector drawings with the annotations of one-to-one stroke correspondence. The frames simulate animated keyframes with corresponding strokes in 2D animation process. Thus, the dataset is expected to facilitate research of automatic inbetweening.

Collecting such a dataset fully with human effort, *i.e.*, by manually drawing keyframes with vector strokes and then identifying the stroke correspondence, is laborious and time-consuming. To reduce the human labor, we turn to an automatic algorithm followed by manual review. The presented *automatic algorithm* applies transformations to existing vector line drawings (denoted as reference) to produce deformed ones with stroke correspondence (denoted as target). The resulting target frames exhibit dynamics with respect to

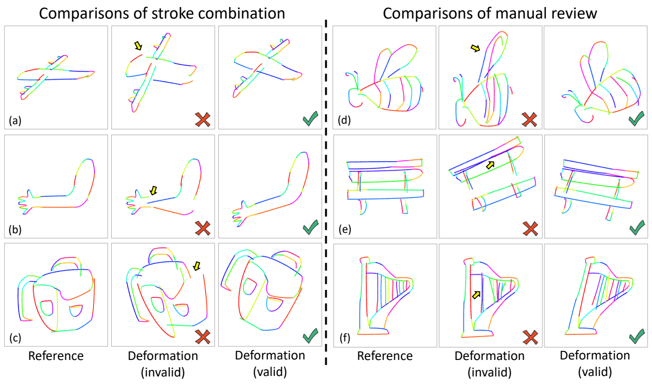

Fig. 7. In our dataset, we deform the reference vector drawings through an automatic algorithm followed by manual review, to mimic the consecutive animated keyframes with vector stroke correspondence.

the reference, as shown in Fig. 7. Artifacts such as improper stroke distribution and stroke ambiguity may be induced by the automatic generation, so subsequent *manual review* is done to filter out undesired deformations. Compared with manual stroke tracing and correspondence identification, the reviewing process requires much less workload from human.

We build our dataset upon a vector sketch dataset TU-Berlin [Eitz et al. 2012], which contains 20,000 sketches from 250 object categories and stores temporal order of strokes in SVG with cubic Bézier curves. In this dataset, the strokes are short and a sketch tends to store a great number of strokes. To eliminate unnecessarily short strokes, we merge every two of them into one. After the pre-processing, we obtain 11,000+ sketches with 4 to 50 strokes. To increase the training efficiency, we define a maximum sequence number as 40. Thus, strokes exceeding the maximum length are directly deleted. We randomly select 11,000 pairs, and split them into a training and a validation sets with 10,000 and 1,000 pairs, respectively.

After applying the automatic algorithm and the manual review, the paired frames and the ground-truth vector stroke correspondence are obtained. The vector strokes are rendered into raster images of size 320×320 (with a padding 40) via DiffVG [Li et al. 2020] with a stroke thickness 1.2. Please refer to supplemental document for more statistics of the dataset. During testing and inference, we collect real line drawings (both clean and rough) from research papers and existing datasets with animated frames (*e.g.*, CreativeFlow+ [Shugrina et al. 2019]), and create the vector drawings for the starting reference frames via a digital painting software Krita [2023].

4.2 Automatic Algorithm

Our automatic algorithm applies two types of deformations to the vector sketches to produce the corresponding frames: entirety-level and group-level, which mimics the real animated frames typically with non-rigid motions. The entirety-level deformation is first applied to transform the sketch globally, followed by the group-level one that deforms a group of strokes (*e.g.*, a stroke chain). Both deformations are sampled according to a probability $\mathcal{P}_{pr}$, as listed in Table 1. We use four types of transformations: translation, rotation, scaling and shearing. A random number of n (= 1, 2, 3 or 4) types are combined to form the 2D affine transformation. The translation moves the strokes by a random offset $(\mathcal{P}_{t_x}, \mathcal{P}_{t_y})$. The rotation, scaling or shearing is done based on a center of any point in the sketch/group, with a random rotation angle $\mathcal{P}_r$, scaling factors $(\mathcal{P}_{s_x}, \mathcal{P}_{s_y})$ or shearing angle $\mathcal{P}_{sh}$, respectively. The shearing is done randomly in x or y direction. Transformation that leads to out-of-bound strokes is considered invalid and a new one is sampled repeatedly until a valid one is generated.

Table 1. Parameters of the deformations for generating the dataset.

	Entirety-level	Group-level
Probability $\mathcal{P}_{pr}$	$\mathcal{P}_{pr} = 0.95$	$\mathcal{P}_{pr} = 0.8$
Translation $\mathcal{P}_{t_x}, \mathcal{P}_{t_y}$ (px)	$-20 \le \mathcal{P}_{t_x}, \mathcal{P}_{t_y} \le 20$	$-10 \le \mathcal{P}_{t_x}, \mathcal{P}_{t_y} \le 10$
Rotation angle $\mathcal{P}_r$ (°)	$-30 \le \mathcal{P}_r \le 30$	$-15 \le \mathcal{P}_r \le 15$
Scaling factor $\mathcal{P}_{sx}, \mathcal{P}_{sy}$	$0.7 \le \mathcal{P}_{sx}, \mathcal{P}_{sy} \le 2.0$	$0.8 \le \mathcal{P}_{sx}, \mathcal{P}_{sy} \le 1.3$
Shearing angle $\mathcal{P}_{sh}$ (°)	$-20 \le \mathcal{P}_{sh} \le 20$	$-10 \le \mathcal{P}_{sh} \le 10$

The transformation may lead to an issue that visually connected strokes in the reference are disconnected in the target due to the individual deformations of the stroke chains, as shown in Fig. 7-left. To overcome this issue, stroke chains with close endpoints are combined as a group and undertake the same transformation.

Note that the sampled deformation parameters are not stored into our dataset as the supervision data for the ASTMs. The reason is that the sampled transformation is not optimal to align patches when other strokes with individual deformations change the content inside the patches simultaneously.

4.3 Manual Review

As shown in Fig. 7-right, the automatic algorithm tends to induce artifacts, *e.g.*, improper stroke placement that leads to unreasonable structure variation (d), stroke ambiguity and occlusion (e & f). They are detrimental to the judgment of stroke correspondence. Hence, we improve the quality of the data through manual review that keeps human in the loop. To this end, we invite 15 participants to filter out the undesired deformations, each assigned 700 to 750 pairs of the frames. Once a deformed drawing is marked undesired, a new deformation based on another sampled transformation is produced for reviewing again. This procedure repeats until all the assigned examples are regarded valid.

5 EXPERIMENTS

5.1 Implementation Details

The definition of initial window size for both reference and target is critical in the cropping operations commonly used in the framework. In ASTM-P, a random size ranging from 224 to 320 is used during training to increase the data diversity. While in inference, we use 0.6× the input image size to accommodate to various input resolutions. In the starting point matching model, a fixed size of 256 is used as we assume ASTM-P has reduced the spatial variations. In ASTM-S and the stroke tracing and correspondence model, we use a dynamic window size that is 1.5× the length of each reference stroke.

Table 2. Ablation study in starting point matching model. Values of the metric are in e-2.

Target canvas	Reference stroke	Penalty	ASTM	PE(↓)
×	✓	✓	×	2.18
✓	×	✓	×	1.72
✓	✓	×	×	1.63
✓	✓	✓	×	1.58
✓	✓	✓	✓	**1.14**

Table 3. Ablation study in stroke tracing and correspondence model. Values of the metrics are in e-2.

	ASTM	Loss	pre-trained ASTM	PS(↓)	EPE(↓)	APE(↓)
#1	×	Ras. & Vec.	-	0.84	1.75	4.66
#2	✓	**Ras.**	✓	0.82	1.64	9.50
#3	✓	**Vec.**	✓	0.78	**0.92**	**2.83**
#4	✓	Ras. & Vec.	×	0.69	1.24	3.82
#5	✓	Ras. & Vec.	✓	**0.59**	0.93	2.95

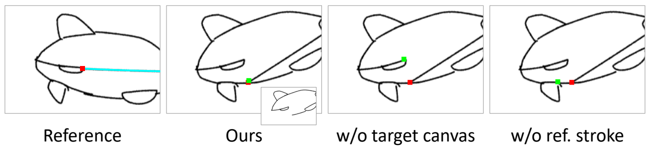

Fig. 8. Visual results of ablation study in starting point matching model. The blue stroke in reference indicates the stroke that the starting point belongs to. Red dots denote the ground truth and green ones are the predictions. Sub-figure shows the previous target canvas.

The prediction networks in the two models and the ASTMs are built with CNN and MLP/RNN, so we define a fixed input image size as 256 for those in the ASTM-P and the starting point model, and 192 for those in the ASTM-S and the stroke model. Please refer to supplemental document for network details.

5.2 Quantitative Evaluation Metrics

We use our validation set for quantitative evaluations of our design choices. For the starting point matching task, we adopt *point error* (PE) as the metric, which measures the Euclidean distance between the predicted point and the ground truth. In terms of the stroke tracing and correspondence task, we use three metrics: (1) *perceptual score* (PS) from [Mo et al. 2021] that measures the raster-level similarity between the final canvas and the input target frame, (2) *endpoint error* (EPE) which assesses the vector-level accuracy of one-to-one stroke correspondence by computing the Euclidean distance between the predicted endpoint and the ground-truth one, and (3) *all-point error* (APE), another vector-level metric, that additionally accounts for the prediction accuracy of the control points with Euclidean distance on the basis of EPE.

5.3 Ablation Study

In the starting point matching model, we validate the effectiveness of the inputs and the introduced non-stroke point penalty, and quantitative results are shown in Table 2. When target canvas is not used as input, the model suffers the worst performance, because it may predict points incorrectly in the already drawn areas (see Fig. 8). Without reference stroke as the guidance, the model probably lacks the ability to resolve stroke topology ambiguity and fails to locate the matching position when stroke connectivity changes.

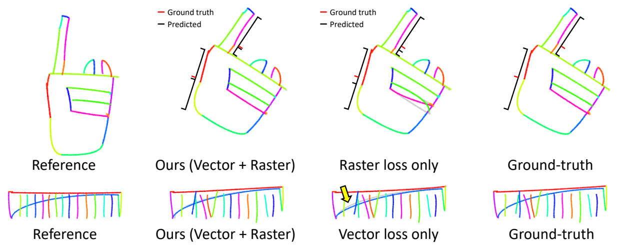

Fig. 9. Ablation study of loss type in stroke tracing and correspondence model. Length proportions of strokes are shown in top row.

In the stroke tracing and correspondence process, we study the loss type. As shown in Table 3, when using the raster loss only, the performance decreases, especially in the vector-level metrics. Typically, it is difficult for the raster loss to ensure the vector correspondence, *i.e.*, the length proportion of the strokes, even with a redrawing consistent with the target frame. Figure 9-top shows an example where the produced proportion is worse than ours. When using the vector loss only, the vector metrics are slightly better than ours, while the raster metric is much worse. We observe the vector loss fails to account for the fidelity of pixel-level fine-grained details, especially when a stroke is long (Fig. 9-bottom).

5.4 Comparisons with Existing Approaches

5.4.1 Evaluation Settings. To the best of our knowledge, we for the first time propose a framework on predicting vector stroke correspondence for consecutive raster keyframes according to the only vector drawing from the starting frame. Therefore, we adapt existing approaches for this task to form baseline methods, mainly including methods based on (1) raster correspondence or (2) vector stroke correspondence.

Based on Raster Correspondence. The raster correspondence predicted by two raster keyframes can be used to advect the vector control points from the starting drawing, such that the strokes are moved towards the next frame. To obtain the raster correspondence, we take into account two lines of methods: optical flow or image registration. For the optical flow-based method, we try different approaches, both classic (PWC-Net [Sun et al. 2018] and RAFT [Teed and Deng 2020]) and the latest (GMA [Jiang et al. 2021a] and GMFlow [Xu et al. 2022b]), trained on different 2D animation datasets (CreativeFlow+ [Shugrina et al. 2019] or AnimeRun [Siyao et al. 2022]). GMA trained on the AnimeRun dataset is found to work the best. In terms of the image registration algorithms, we try both a deep learning-based method APES [Xu et al. 2022a] designed for 2D cartoon frames and a non-deep learning algorithm introduced by Wang *et al.* [2021] tailored for non-rigid registration of line drawings [Xiao et al. 2022]. The latter is an energy-based optimization

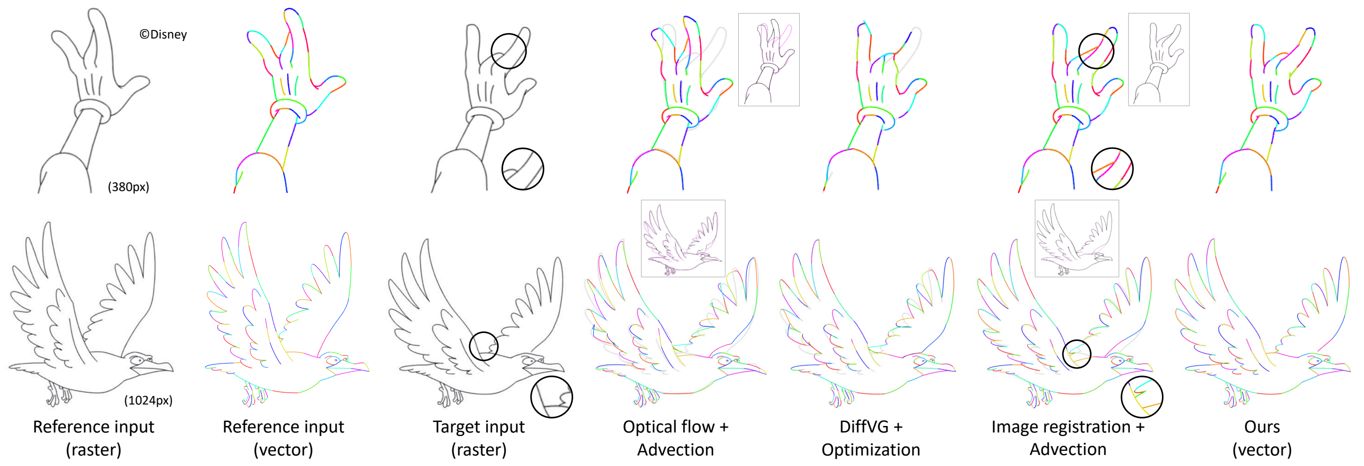

Fig. 10. Comparisons with baseline methods based on raster correspondence. Sub-figures in square box are warped reference images by the predicted optical flows or registration maps (magenta drawing denote the target). Please refer to supplemental document for more results. Hand from paper [Whited et al. 2010] ©2010 Blackwell Publishing Ltd. Eagle from paper [Yang 2017] is courtesy of Jie Li ©2018, IEEE.

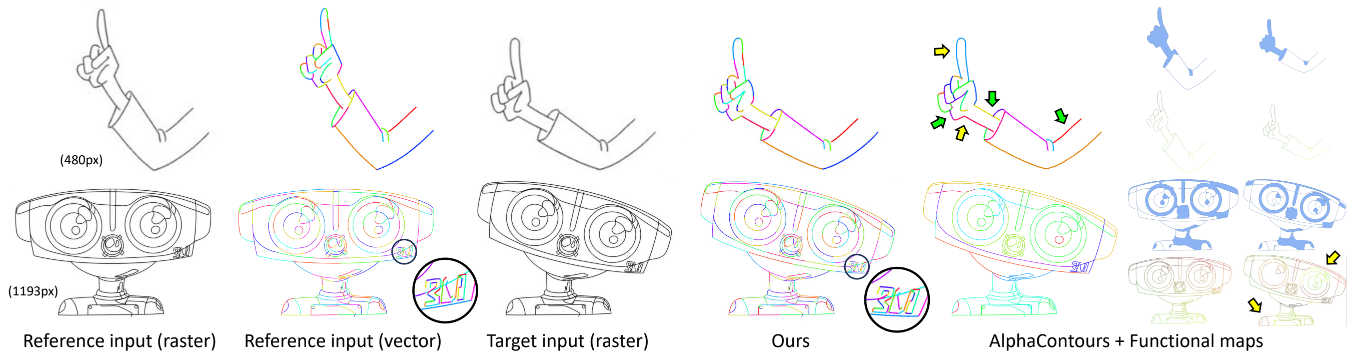

Fig. 11. Comparisons with a baseline method based on vector stroke correspondence. Sub-figures on the rightmost are sketch shapes computed by AlphaContours [Myronova et al. 2023] (top) and functional maps indicating the correspondence [Ovsjanikov et al. 2012] (bottom). Arm from paper [Yang 2017] is courtesy of Jie Li ©2018, IEEE. Robot from paper [Yang et al. 2018] is courtesy of Yovanny Ramirez ©2018 John Wiley & Sons Ltd.

algorithm that aligns raster drawings by maximizing their cross correlation over a dense displacement field. It is done in a coarse-to-fine manner, employing an affine transformation and several B-spline transformations first to initialize the displacement field used for subsequent dense deformation. The non-deep learning method is found to work better.

We additionally compare to another optimization algorithm that uses a differentiable renderer DiffVG [Li et al. 2020] to directly optimize the reference vector strokes to best align with the drawing in the target keyframe. A raster loss plus a Laplacian regularization term for preventing severely distorted strokes are adopted during the optimization. To speed up convergence, we advect the strokes along the optical flows mentioned above as the initialization.

Based on Vector Stroke Correspondence. We first apply a vectorization algorithm to the raster target frame, and then compute the stroke correspondence at vector level. Note that a potential issue of this solution is the vectorization process may generate a vector drawing with a different number of strokes from that in the reference. This issue tends to disable the one-to-one correspondence required by our task.

In the vectorization step, we exploit PolyVectorization [Bessmeltsev and Solomon 2019] to generate vector drawings consisting of polylines. Then, we use a combination of AlphaContours [Myronova et al. 2023] and Functional Maps [Ovsjanikov et al. 2012], which is designed for the stroke correspondence between vector drawings even with different numbers of strokes. This solution generates vertex-wise functional correspondence between sampling points on the strokes, and a discrete optimization [Ren et al. 2021] technique is exploited to compute point-wise correspondence. Based on the result, we adopt a simple strategy to compute the stroke-level correspondence. For each target stroke, its correspondence is voted by its points with associated points on the reference strokes. We do not try a complex strategy as we find the predicted functional maps are somewhat less than satisfactory.

5.4.2 Results. The comparisons with raster correspondence-based approaches are shown in Fig. 10. The optical flow-based method struggles with large non-rigid motions (index finger in top-row example) and repeated patterns (wings in bottom-row example), and thus the results with strokes advecting along the problematic flows are largely misaligned with the target drawings. Using such results as initialization, DiffVG optimization produces results with more strokes aligned, although it tends to fall into local optimum (*i.e.*, strokes at both sides of the index finger move to a single side in the target image). This is probably because the independent optimization of all the strokes without interactions with each other is so non-convex that every stroke tends to converge towards the nearest line in the target drawing. While the image registration-based method works better than the previous two baselines, it fails to warp strokes with non-rigid deformations. When the stroke shape or connectivity varies between the reference and the target, the results still inherit those from the reference. In sharp contrast, our results exhibit high alignment with the target drawings in the presence of large motions, as well as good stroke correspondence.

The comparison with the solution built on vector stroke correspondence is shown in Fig. 11. Even with similar sketch shapes computed by the AlphaContours algorithm [Myronova et al. 2023], the functional maps exhibit incorrect vertex-wise correspondence (bottom-row example) that is detrimental to the final stroke-level correspondence. This is attributed to the varying numbers and lengths of the strokes between the input reference vector drawing and the vectorized target one consisting of a large number of short polylines. The inconsistent geometric composition induces varying triangulations even within the similar sketch shapes, which harm the performance of the functional maps (please refer to the supplemental document for more analyses). Another issue is that even if the functional correspondence is fine, the different stroke numbers hinder one-to-one correspondence by definition. As shown in the top-row example, while some short strokes are correctly matched (see green arrows), the long ones fail (see yellow arrows). In contrast, our approach, without the need of a combination of separated techniques, directly produces vector drawings with consistent stroke numbers and reasonable one-to-one correspondence that is desired in the automatic inbetweening.

5.5 Generalization to Rough Drawings

While our approach is trained with clean line drawing images, it generalizes well to rough ones. We try different types of rough drawings, including line-based renderings of 3D characters from CreativeFlow+ dataset [Shugrina et al. 2019], sketches drawn with stylus with *overdrawn strokes*, and pencil drawing photographs with *shadow and non-pure background*. The vector inputs for the starting frames are manually traced with simplified strokes. As shown in Fig. 12, the rough drawings are notably different from the clean ones, which exhibit sketchy strokes and different amounts of perturbations of stroke thickness. The result shows that our approach still works well on these unseen drawing styles, even in the presence of non-rigid motions. We attribute the success to our proposed ASTM, which is robust enough to align the rough patches and reduce the drawing variation (please refer to Section 5.6 for more analyses and justification). With such a promising property, our approach is compatible with the standard 2D animation workflow mostly starting with rough sketching and thus helps to promote the production efficiency.

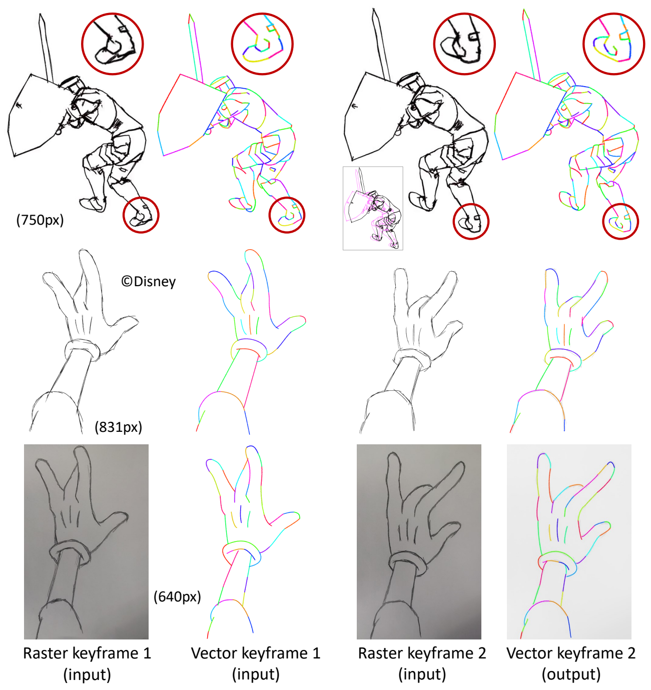
(750px)
©Disney
(831px)
(640px)
Raster keyframe 1 (input)
Vector keyframe 1 (input)
Raster keyframe 2 (input)
Vector keyframe 2 (output)

Fig. 12. Result on different types of rough drawings. Sub-figure in box shows differences between consecutive frames, with magenta for the previous and black for the current. Paladin is from [Shugrina et al. 2019]. Hand images from paper [Whited et al. 2010] ©2010 Blackwell Publishing Ltd.

5.6 Effectiveness of ASTM

Our proposed adaptive spatial transformation module (ASTM) takes effect in content alignment that benefits subsequent starting point and stroke models. As shown in Fig. 13, we test its robustness to both near-linear transformations (*e.g.*, rotation (b1), scaling (b2) and translation (b3)) and more commonly seen non-linear ones (*e.g.*, (b4), (b5) and (b6)). Albeit with non-linear or non-rigid deformations such as shape distortion and topology change, the ASTM is able to adaptively predict linear transformations (see (a)) to roughly align the patches and reduce spatial variations. This is because the ASTM learns to make use of all the visual cues in the patches to make them as similar as possible. Although the alignment is sub-optimal, it brings obvious improvements to the subsequent predictions of starting points or strokes while helping to improve their fault tolerance and reliability. We also show patches ((a)/(b4) and (b5)) of rough drawings from Fig. 12, and the reasonable transformations confirm that the ASTM trained on clean drawings is also applicable to rough ones. This aids our framework in generalizing to rough sketches.

Table 2 shows quantitative comparisons between starting point matching models with and without the ASTM-P, where we can see

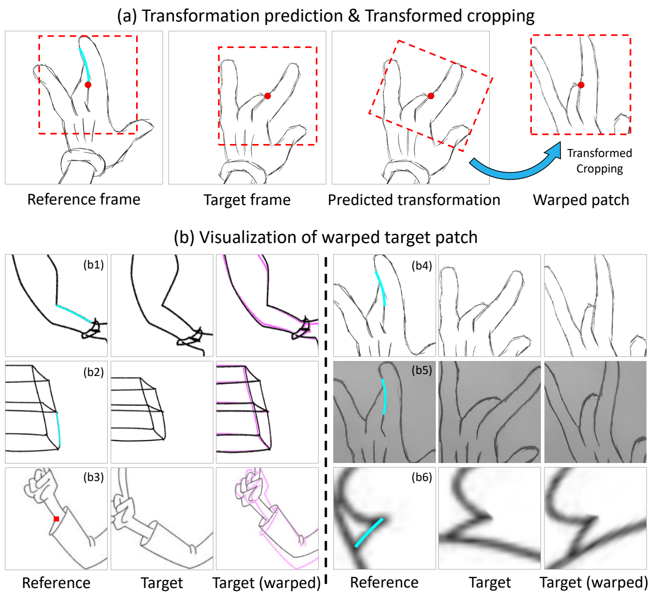

Fig. 13. (a) Effectiveness of ASTM that predicts a transformation to roughly align the target patch with the reference. (b) More examples of warped patches from the target. Drawings in magenta underneath are the references. Reference strokes at current step are highlighted in blue. See the predicted transformations and more results in supplemental document.

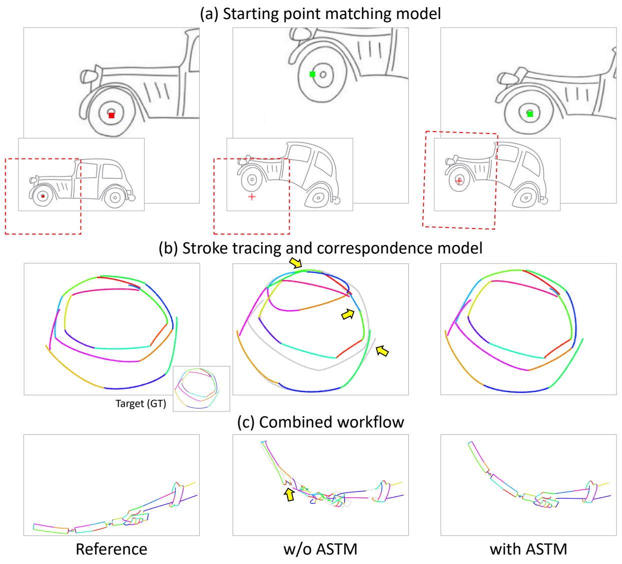
(a) Starting point matching model
(b) Stroke tracing and correspondence model
Target (GT)
(c) Combined workflow
Reference
w/o ASTM
with ASTM

Fig. 14. Comparisons between models with and without the ASTM. Car and Stick from paper [Yang 2017] are courtesy of Jie Li ©2018, IEEE.

ASTM-P offers a significant performance boost. Qualitative results are shown in Fig. 5-top row and Fig. 14-(a). Both examples indicate that the ASTM-P benefits the matchup prediction by transforming the target patches into those roughly aligned with the reference.

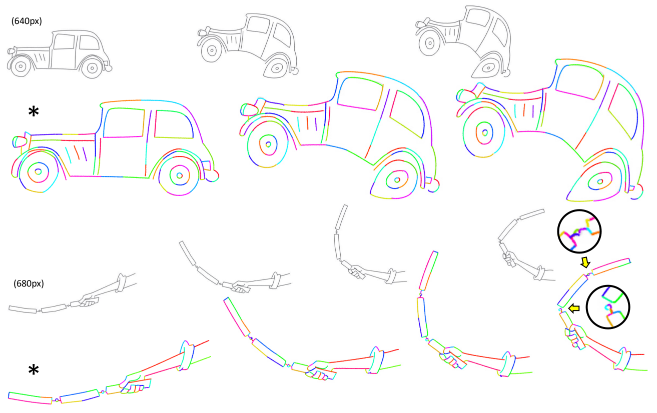

Fig. 15. Results on multiple clean keyframes. Only vector drawings of the starting frames (with marks '∗') are provided. Car and Stick from paper [Yang 2017] are courtesy of Jie Li ©2018, IEEE.

To evaluate the stroke tracing and correspondence model independent of the starting point matching model, we use real starting points from the dataset for each stroke chain to conduct this ablation study. Quantitative results in Table 3 manifest the significance of the ASTM-S, which notably boosts the performance in both raster and vector metrics. Qualitative results in Fig. 5-bottom row and Fig. 14-(b) show that in the absence of the ASTM-S, the model fails to handle large geometric deformations, and draws strokes with incorrect orientation or proportion of length. Since we use a pre-trained ASTM-S when training the tracing model, we also study the effectiveness of the pre-training. As can be seen in Table 3-#4, when we use a randomly initialized ASTM-S and train it with the tracing model jointly, the performance degrades to a certain degree.

When combining the two models, *i.e.*, using the predicted starting points for each stroke chain, we show how a starting point matching model without the ASTM-P affects the whole process. As shown in Fig. 14-(c), with an incorrectly predicted starting point (see the arrow), it is too difficult to redraw the strokes correctly. Thus, a terrible result is produced. In contrast, correct starting point predictions with the ASTM-P promote the overall performance of joint stroke tracing and correspondence.

5.7 Results on Multiple Raster Keyframes

Our approach also works with more than one target keyframes, dependent exclusively on the starting vector drawing, by forward propagating the generated vector results, *i.e.*, using the generation as a second reference. This task is much more challenging due to the continuous motion difference that induces incremental topology variations. A high prediction accuracy is required, otherwise the error accumulation probably leads to worse and worse results in the latter frames. As demonstrated in Fig. 15 and Fig. 16, our approach produces promising results on both clean and rough drawings, which corroborate the effectiveness of our framework in high fidelity of vectorization and accuracy of stroke correspondence. Furthermore, they also indicate our approach suits well with the automatic inbetweening process with a series of keyframes.

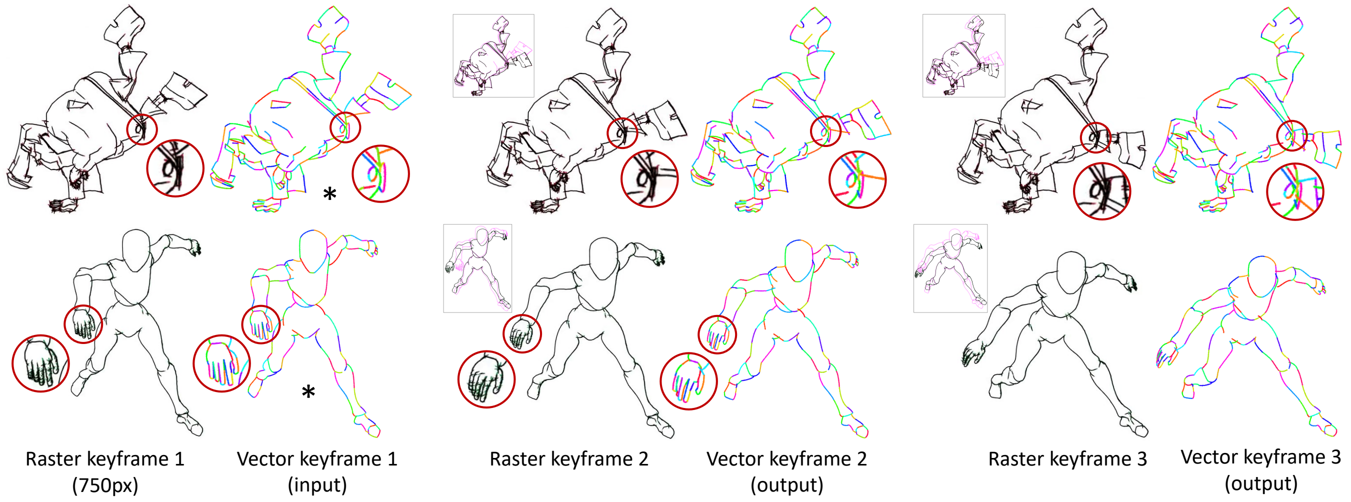
*
*
Raster keyframe 1 (750px)
Vector keyframe 1 (input)
Raster keyframe 2
Vector keyframe 2 (output)
Raster keyframe 3
Vector keyframe 3 (output)

Fig. 16. Results on multiple rough keyframes. Only vector drawings of the starting frames (with marks '∗') are provided. Sub-figures in boxes show differences between consecutive frames, with magenta for the previous and black for the current. Bigvegas1 and Xbot are from [Shugrina et al. 2019].

Due to the imperfect prediction, error accumulation does exist and gives rise to degradation, as shown in the second example in Fig. 15. While 3 strokes are incorrectly predicted (see arrows), they could be adjusted with a few manual editings quickly in an interactive system (see Section 5.9) and then used for inbetweening. This requires much less labor and time comparing to redrawing the entire keyframe.

5.8 Sensitivity to Initial Vectorization

Evaluation Settings. Our framework relies on the initial vector input of the starting keyframe, so we evaluate its robustness and sensitivity to the input vectorization in two aspects: (1) stroke order and (2) stroke number or length. For the *stroke order*, we randomize the order of the stroke chains of the input vector drawings. Moreover, we randomly reverse the drawing order of each stroke chain. These operations also perturb the choice of initial starting point. We generate 5 random results for the evaluation. As for the sensitivity to *stroke number or length*, we create an input vector drawing with longer strokes for each original vector input (*e.g.*, short strokes are merged into a single continuous line), so that the number of strokes is decreased.

We conduct the evaluation with real clean drawings and real rough ones. Due to the lack of ground-truth, we manually count the number of incorrect strokes, including those in wrong correspondences or wrong trajectories that need manual fix. We invite 3 people to count the incorrect strokes of the outputs above, and report the averaged results.

Robustness to Stroke Order. As can be seen in Fig. 17, our approach shows equally good performance on vector drawings with different stroke orders or initial starting points. The averaged numbers of incorrect strokes over the 5 results with randomized stroke orders are shown in block 'Short strokes' of Table 4 (column #incorrect). Albeit with order perturbations, the numbers of incorrectness are

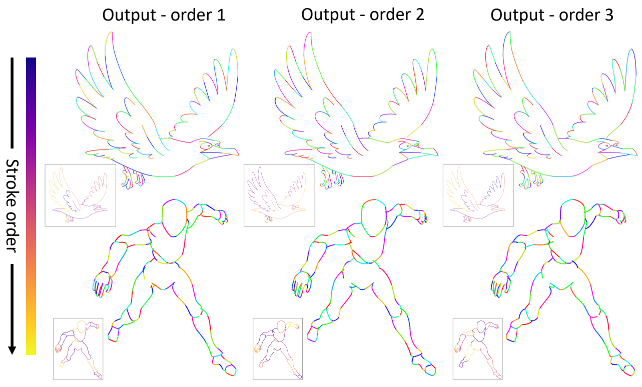

Fig. 17. Results with different stroke orders. Sub-figures show the stroke orders. We show 3 out of 5 orders and omit reference vector frames here for brevity. Please refer to supplemental document for complete and more results. Eagle from paper [Yang 2017] is courtesy of Jie Li ©2018, IEEE. Xbot is from [Shugrina et al. 2019].

small in a low error rate (most less than 3%). Those wrong strokes can be fixed quickly in an interactive user interface (see Section 5.9). The results indicate that our approach is robust enough to be independent of input vector drawings with a reasonable stroke order or a well-defined initial starting point.

Sensitivity to Stroke Number or Length. A vector drawing with longer strokes tends to yield a small number of strokes. As shown in Fig. 18, our method also works with some long strokes merged from several shorted ones (see yellow arrows). For the rough sketch, we use a more simplified input drawing with about 60% strokes, and the resulting output is corresponded to the respective input with skipped details and longer strokes. We do notice the framework performs less than satisfactory on some other long strokes with worse alignment with the target, as pointed out by red arrows in Fig. 18

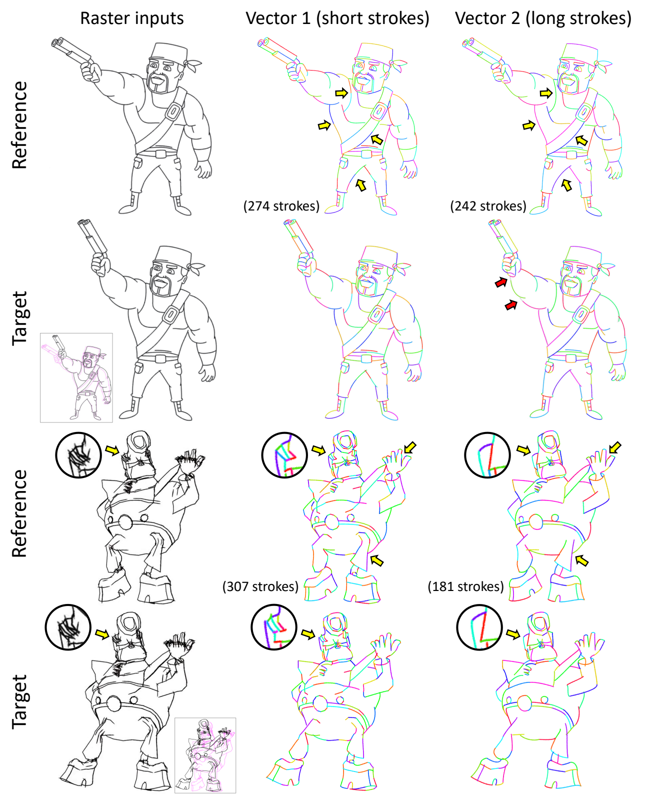

Fig. 18. Results with different numbers or lengths of strokes. See more in supplemental document. Gunman from paper [Yang et al. 2018] is courtesy of Eugene Babich ©2018 John Wiley & Sons Ltd. Bigvegas2 is from [Shugrina et al. 2019].

Table 4. Statistics of drawings and numbers of incorrect strokes counted manually. Marks '#' denote the number. For short strokes, the numbers of incorrect strokes are averaged over 5 random orders as introduced in Section 5.8, and are rounded up to integers. Percentages denote error rate.

Image	Location	Short strokes		Long strokes	
		#stroke	#incorrect	#stroke	#incorrect
Arm	Fig. 11-top	38	2 (5.3%)	31	5 (+3)
Hand	Fig. 10-top	45	1 (2.2%)	29	4 (+3)
Stick	Fig. 15-bottom	61	2 (3.3%)	55	3 (+1)
Car	Fig. 15-top	97	4 (4.3%)	80	6 (+2)
Eagle	Fig. 10-bottom	129	3 (2.3%)	108	9 (+6)
Robot	Fig. 11-bottom	239	4 (1.7%)	205	9 (+5)
Gunman	Fig. 1-top	274	4 (1.5%)	242	8 (+4)
Paladin	Fig. 12-top	153	4 (2.6%)	125	5 (+1)
Xbot	Fig. 16-bottom	154	2 (1.3%)	118	3 (+1)
Bigvegas1	Fig. 16-top	214	1 (0.5%)	156	1 (+0)
Bigvegas2	Fig. 1-bottom	307	5 (1.6%)	181	4 (−1)

where the gray drawings underneath indicate the misalignment. The performance drop is also suggested by the increased numbers of incorrect strokes shown in block 'Long strokes' of Table 4. In a word, our approach copes with long strokes to a certain extent, while still suffering from slight performance degradation. This is probably because the framework is trained on a relatively simple dataset with insufficient complexity, where long lines are mostly split into short segments as can be seen in Fig. 7. As we make the first attempt to assemble such a dataset with less human workload to tackle the more challenging joint stroke tracing and correspondence task, we believe the applicability of our approach to long strokes could be improved with more complex training data.

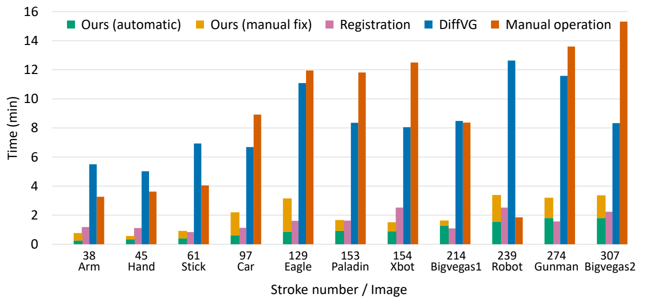

Fig. 19. Runtime comparison. Please refer to supplemental document for numerical ones.

5.9 Runtime Comparison

We report the runtime of our approach and the baseline methods introduced in Section 5.4 to evaluate how much our method helps the 2D animation pipeline. An experienced CACANi user is invited to manually fix our automatically generated results, and the time he took is summated to that of the model inference. We compare to methods DiffVG optimization [Li et al. 2020] and image registration [Wang et al. 2021]. Optical flow method [Jiang et al. 2021a] (near-real-time) and vector stroke correspondence-based one [Myronova et al. 2023] are ignored due to their much worse performance. The user is also asked to operate on the CACANi software to manually produce the target vector drawings with stroke correspondence, by either tracing the target images from scratch or editing the reference vectors. This derives a time of doing by hand. The automatic algorithms above are tested under the same environment on a Ubuntu machine with an Intel Xeon Platinum 8375C @ 2.90GHz CPU, 252GB RAM, and an NVIDIA GeForce RTX 3090 GPU.

As illustrated in Fig. 19, the image registration method takes comparable runtime to ours, while its overall visual performance is worse. DiffVG takes a lot of time to optimize. The runtime of our automatic algorithm is proportional to the number of strokes. Thanks to the high accuracy in stroke tracing and correspondence, it does not take a long time for users to fix our results manually. Thus, our method combined with manual fixing is much faster than if done by hand in general, except for the case "Robot" in Fig. 11, in which the deformation is exactly linear (*i.e.*, the bottom is invariant and the head rotates around the center) so that the user edited the reference with a built-in rotation tool quickly. The runtime comparisons above suggest that our approach is effective and friendly enough to practical use in real-life 2D animation workflow.

5.10 Inbetweening and 2D Animation

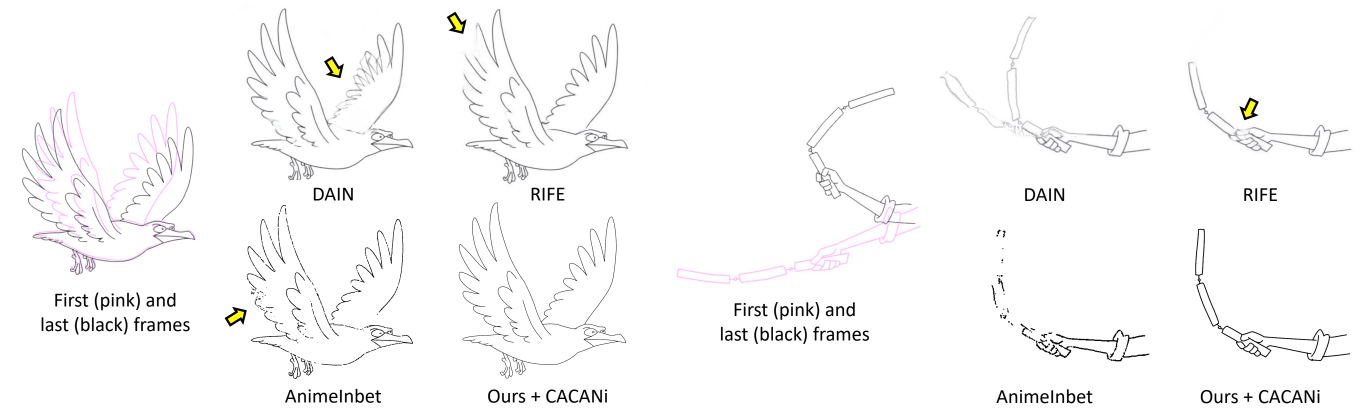

Fig. 20. Comparisons of inbetweening results. Please refer to the supplemental video for animations and more results. Eagle and Stick from paper [Yang 2017] are courtesy of Jie Li ©2018, IEEE.

The keyframes with vector stroke correspondence produced by our framework can be directly imported into an existing inbetweening product, *e.g.*, CACANi [2020], to generate inbetween frames. Albeit with minor artifacts as shown in Fig. 15, they can be manually corrected with a few interactions readily (*e.g.*, adjusting 3 strokes) in the inbetweening software. Afterwards, we use built-in stroke interpolation tools in CACANi to generate the inbetweens.

We also compare with representative framerate upsampling methods DAIN [Bao et al. 2019] and RIFE [Huang et al. 2022], and a recent line drawing inbetweening method named AnimeInbet [Siyao et al. 2023]. We adopt their officially released model weights and parameters. For AnimeInbet that requires vertices of vector drawings as inputs, we try vectorization methods that generate polylines, namely PolyVectorization [Bessmeltsev and Solomon 2019] and SingularityFree [Guţan et al. 2023], and show the better results. The comparisons are shown in Fig. 20, where DAIN performs poorly with duplication of drawings and blurry artifacts in the presence of repeated patterns and large motion. RIFE works better than DAIN, while producing blur and disappearance of strokes. The issues above arise commonly in raster inbetweening approaches that predict discrete pixels instead of compact strokes. AnimeInbet tends to produce inbetween frames with discontinuous lines because its visibility prediction for stroke vertices struggles with real drawings instead of line renderings of 3D models with which it trains. Compared with these alternatives, our method together with the interpolation algorithms in CACANi yields clean inbetweens with high curve continuity, which confirms that our approach suits well with the standard automatic inbetweening process.

6 LIMITATIONS AND DISCUSSION

While our approach produces promising one-to-one vector stroke correspondence in a fully automatic way, it may fail in cases with occlusion, topology change and complex non-rigid motions. The first two issues are common while challenging in automatic inbetweening, and the majority of existing systems resolve them through user interactions [Even et al. 2023; Jiang et al. 2022; Whited et al. 2010; Yang 2017; Yang et al. 2018].

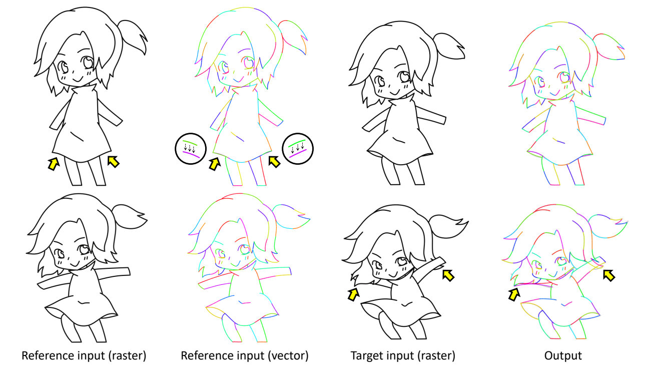

Fig. 21. Results on examples with occlusion. Images from paper [Yang et al. 2018] are courtesy of Zirong Low ©2018 John Wiley & Sons Ltd.

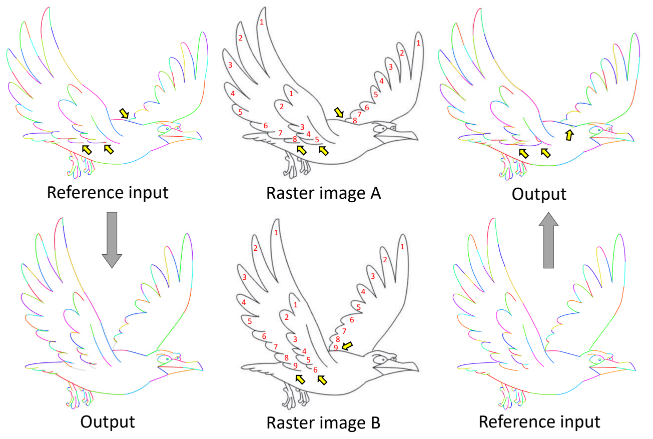

Fig. 22. Failure cases with topology (line structure) change. We use image A as reference and B as target, and then reverse. Numbers in the images indicate the stroke variations. Note that image A is different from those displayed before, with manual modification to yield topology variance. Eagle from paper [Yang 2017] is courtesy of Jie Li ©2018, IEEE.

Figure 21 shows two examples involved with occlusions, which make strokes appear in the first one and disappear in the second. In the first example, our method works based on a common solution by drawing additional occluded lines that overlap with other strokes in the reference vector input (see arrows). When the strokes disappear as shown in the second example, our approach fails inevitably due to the lack of depth information and draws unreasonable strokes. User interactions are required to resolve the occlusion, such as defining visibility toggles [Yang 2017; Yang et al. 2018] or boundary strokes [Jiang et al. 2022].

Topology change also leads to stroke appearing or disappearing, as indicated by the numbers in Fig. 22. When using image A as reference and B as target with appearing strokes (the left-most column), we try the same solution with occluded lines (overlaps are pointed by arrows) but it fails in this case. This is because the shapes and orientations between the occluded lines and the appearing strokes vary significantly. Here a few manual adjustments could refine the results. In the reversed scenario (B as reference and A as target, the right-most column) where strokes disappear, similar to the occlusion case, the framework draws unnecessary strokes. Besides the solutions with visibility toggles or boundary strokes, automatically predicting a visibility variable for each stroke could be a future extension of our approach.

Our introduced ASTM predicts affine transformations for rough alignment between patches, which is shown to work on some cases with non-linear deformations (Section 5.6). However, the linear transformations are still insufficient to handle drawings with complex non-rigid motions, such as a straight line evolving into a bump or a highly curved (near-circular) stroke. We show such a failure case in supplemental document. Non-linear transformations, such as Thin-Plate Spline (TPS) [Zhao and Zhang 2022] could be incorporated as an extension. Given that our ASTM is required to be invertible and TPS is generally irreversible analytically, there still exist open problems that are out of scope for our paper and worth addressing in future work.

7 CONCLUSION

We focus on a fundamental step in automatic inbetweening, redrawing raster keyframes with vector strokes while ensuring one-to-one stroke correspondence, and propose an automatic method to reduce the tedious manual efforts. An adaptive spatial transformation module (ASTM) is introduced to handle non-rigid motions and stroke distortion, which also aids the framework in coping with rough animated frames. A line drawing dataset with annotation of vector stroke correspondence is collected, and is expected to advance the research in inbetweening community. While the approach shows compelling results, it performs less than satisfactory in cases with occlusion and topology change, which are common issues in real inbetweening scenario. The incorporation with interactive operations could be a promising future direction.

ACKNOWLEDGMENTS

This work was supported by the Natural Science Foundation of Guangdong Province, China (Grant No. 2022A1515011425).

REFERENCES

Adobe. 2022. *Adobe Animate.* https://www.adobe.com/products/animate.html

Marc Alexa, Daniel Cohen-Or, and David Levin. 2000. As-rigid-as-possible shape interpolation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques.* 157–164.

Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. 2019. Depth-aware video frame interpolation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 3703–3712.

Mikhail Bessmeltsev and Justin Solomon. 2019. Vectorization of line drawings via polyvector fields. *ACM Transactions on Graphics (TOG)* 38, 1 (2019), 1–12.

Blender. 2023. *Blender.* https://www.blender.org/

Nestor Burtnyk and Marceli Wein. 1975. Computer animation of free form images. In *Proceedings of the 2nd annual conference on Computer graphics and interactive techniques.* 78–80.

CACANi. 2020. *CACANi.* https://cacani.sg

Edwin Catmull. 1978. The problems of computer-assisted animation. *ACM Siggraph Computer Graphics* 12, 3 (1978), 348–353.

Shuhong Chen and Matthias Zwicker. 2022. Improving the perceptual quality of 2d animation interpolation. In *European Conference on Computer Vision.* Springer.

Boris Dalstein, Rémi Ronfard, and Michiel Van De Panne. 2014. Vector graphics complexes. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–12.

Boris Dalstein, Rémi Ronfard, and Michiel Van De Panne. 2015. Vector graphics animation with time-varying topology. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–12.

Charles X Durand. 1991. The "TOON" project: requirements for a computerized 2D animation system. *Computers & graphics* 15, 2 (1991), 285–293.

Marek Dvorožňák, Pierre Bénard, Pascal Barla, Oliver Wang, and Daniel Sỳkora. 2017. Example-based expressive animation of 2d rigid bodies. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–10.

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), 1–11.

DWANGO. 2023. *OpenToonz.* http://opentoonz.github.io/e/

Mathias Eitz, James Hays, and Marc Alexa. 2012. How do humans sketch objects? *ACM Transactions on graphics (TOG)* 31, 4 (2012), 1–10.

Melvin Even, Pierre Bénard, and Pascal Barla. 2023. Non-linear Rough 2D Animation using Transient Embeddings. In *Computer Graphics Forum*, Vol. 42. 411–425.

Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. 2016. Fidelity vs. simplicity: a global approach to line drawing vectorization. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–10.

Jean-Daniel Fekete, Érick Bizouarn, Éric Cournarie, Thierry Galas, and Frédéric Taillefer. 1995. TicTacToon: A paperless system for professional 2D animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques.* 79–90.

Yi Guo, Zhuming Zhang, Chu Han, Wenbo Hu, Chengze Li, and Tien-Tsin Wong. 2019. Deep line drawing vectorization via line subdivision and topology reconstruction. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 81–90.

Olga Guţan, Shreya Hegde, Erick Jimenez Berumen, Mikhail Bessmeltsev, and Edward Chien. 2023. Singularity-Free Frame Fields for Line Drawing Vectorization. In *Computer Graphics Forum*, Vol. 42. e14901.

David Ha and Douglas Eck. 2018. A Neural Representation of Sketch Drawings. In *International Conference on Learning Representations.*

Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. 2022. Real-time intermediate flow estimation for video frame interpolation. In *European Conference on Computer Vision.* 624–642.

Takeo Igarashi, Tomer Moscovich, and John F Hughes. 2005. As-rigid-as-possible shape manipulation. *ACM transactions on Graphics (TOG)* 24, 3 (2005), 1134–1141.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. 2015. Spatial transformer networks. *Advances in neural information processing systems* 28 (2015).

Jie Jiang, Hock Soon Seah, and Hong Ze Liew. 2022. Stroke-Based Drawing and Inbetweening with Boundary Strokes. In *Computer Graphics Forum*, Vol. 41.

Jie Jiang, Hock Soon Seah, Hong Ze Liew, and Quan Chen. 2020. Challenges in designing and implementing a vector-based 2D animation system. In *The Digital Gaming Handbook.* CRC Press, 245–274.

Shihao Jiang, Dylan Campbell, Yao Lu, Hongdong Li, and Richard Hartley. 2021a. Learning to estimate hidden motions with global motion aggregation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 9772–9781.

Wei Jiang, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi. 2021b. Cotr: Correspondence transformer for matching across images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 6207–6217.

KDE. 2023. *Krita.* https://krita.org/

Kangyeol Kim, Sunghyun Park, Jaeseong Lee, Sunghyo Chung, Junsoo Lee, and Jaegul Choo. 2022. AnimeCeleb: Large-Scale Animation CelebHeads Dataset for Head Reenactment. In *European Conference on Computer Vision.* Springer, 414–430.

Alexander Kort. 2002. Computer aided inbetweening. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering.* 125–132.

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. 2020. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–15.

Xiaoyu Li, Bo Zhang, Jing Liao, and Pedro V Sander. 2021. Deep sketch-guided cartoon video inbetweening. *IEEE Transactions on Visualization and Computer Graphics* 28, 8 (2021), 2938–2952.

Chenxi Liu, Toshiki Aoki, Mikhail Bessmeltsev, and Alla Sheffer. 2023. StripMaker: Perception-driven Learned Vector Sketch Consolidation. *ACM Transactions on Graphics* 42, 4 (2023).

Chenxi Liu, Enrique Rosales, and Alla Sheffer. 2018. Strokeaggregator: Consolidating raw sketches into artist-intended curve drawings. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–15.

Hanyuan Liu, Chengze Li, Xueting Liu, and Tien-Tsin Wong. 2022. End-to-End Line Drawing Vectorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 4559–4566.

Xueting Liu, Tien-Tsin Wong, and Pheng-Ann Heng. 2015. Closure-aware sketch simplification. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–10.

Live2D. 2023. *Live2D*. https://www.live2d.com/en/

Takeo Miura, Junzo Iwata, and Junji Tsuda. 1967. An application of hybrid curve generation: cartoon animation by electronic computers. In *Proceedings of the April 18-20, 1967, spring joint computer conference*. 141–148.

Haoran Mo, Edgar Simo-Serra, Chengying Gao, Changqing Zou, and Ruomei Wang. 2021. General virtual sketching framework for vector line art. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14.

Mariia Myronova, William Neveu, and Mikhail Bessmeltsev. 2023. Differential Operators on Sketches via Alpha Contours. *ACM Transactions on Graphics* 42, 4 (2023).

Rei Narita, Keigo Hirakawa, and Kiyoharu Aizawa. 2019. Optical flow based line drawing frame interpolation using distance transform to support inbetweenings. In *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 4200–4204.

Pablo Navarro, J Ignacio Orlando, Claudio Delrieux, and Emmanuel Iarussi. 2021. SketchZooms: Deep Multi-view Descriptors for Matching Line Drawings. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 410–423.

Gioacchino Noris, Alexander Hornung, Robert W Sumner, Maryann Simmons, and Markus Gross. 2013. Topology-driven vectorization of clean line drawings. *ACM Transactions on Graphics (TOG)* 32, 1 (2013), 1–11.

Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. 2012. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (ToG)* 31, 4 (2012), 1–11.

Ivan Puhachov, William Neveu, Edward Chien, and Mikhail Bessmeltsev. 2021. Keypoint-driven line drawing vectorization via PolyVector flow. *ACM Transactions on graphics* 40, 6 (2021).

Reallusion. 2023. *Cartoon Animator*. https://www.reallusion.com/cartoon-animator/

William T Reeves. 1981. Inbetweening for computer animation utilizing moving point constraints. *ACM SIGGRAPH Computer Graphics* 15, 3 (1981), 263–269.

Jing Ren, Simone Melzi, Peter Wonka, and Maks Ovsjanikov. 2021. Discrete optimization for shape matching. In *Computer Graphics Forum*, Vol. 40. 81–96.

Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. 2020. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4938–4947.

Maria Shugrina, Ziheng Liang, Amlan Kar, Jiaman Li, Angad Singh, Karan Singh, and Sanja Fidler. 2019. Creative flow+ dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5384–5393.

Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. 2019. Animating arbitrary objects via deep motion transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2377–2386.

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

Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. 2018b. Real-time data-driven interactive rough sketch inking. *ACM Transactions on Graphics (TOG)* 37, 4 (2018).

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.

Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

Li Siyao, Tianpei Gu, Weiye Xiao, Henghui Ding, Ziwei Liu, and Chen Change Loy. 2023. Deep Geometrized Cartoon Line Inbetweening. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7291–7300.

Li Siyao, Yuhang Li, Bo Li, Chao Dong, Ziwei Liu, and Chen Change Loy. 2022. AnimeRun: 2D Animation Visual Correspondence from Open Source 3D Movies. *Advances in Neural Information Processing Systems* 35 (2022), 18996–19007.

Li Siyao, Shiyu Zhao, Weijiang Yu, Wenxiu Sun, Dimitris Metaxas, Chen Change Loy, and Ziwei Liu. 2021. Deep animation video interpolation in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 6587–6595.

Harrison Jesse Smith, Qingyuan Zheng, Yifei Li, Somya Jain, and Jessica K Hodgins. 2023. A Method for Animating Children's Drawings of the Human Figure. *ACM Transactions on Graphics* 42, 3 (2023), 1–15.

Tibor Stanko, Mikhail Bessmeltsev, David Bommes, and Adrien Bousseau. 2020. Integer-Grid Sketch Simplification and Vectorization. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 149–161.

Qingkun Su, Xue Bai, Hongbo Fu, Chiew-Lan Tai, and Jue Wang. 2018. Live sketch: Video-driven dynamic deformation of static drawings. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

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. As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of the 7th International Symposium on Non-photorealistic Animation and Rendering*. 25–33.

Zachary Teed and Jia Deng. 2020. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 402–419.

Toon Boom. 2022. *Harmony*. https://www.toonboom.com/products/harmony

TVPaint. 2023. *TYPaint*. https://www.tvpaint.com/

Zeyu Wang, Sherry Qiu, Nicole Feng, Holly Rushmeier, Leonard McMillan, and Julie Dorsey. 2021. Tracing versus freehand for evaluating computer-generated drawings. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–12.

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.

Chufeng Xiao, Wanchao Su, Jing Liao, Zhouhui Lian, Yi-Zhe Song, and Hongbo Fu. 2022. DifferSketching: How Differently Do People Sketch 3D Objects? *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–16.

Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. 2022b. Gmflow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8121–8130.

Xuemiao Xu, Minshan Xie, Peiqi Miao, Wei Qu, Wenpeng Xiao, Huaidong Zhang, Xueting Liu, and Tien-Tsin Wong. 2019. Perceptual-aware Sketch Simplification Based on Integrated VGG Layers. *IEEE Transactions on Visualization and Computer Graphics* (2019).

Zhan Xu, Matthew Fisher, Yang Zhou, Deepali Aneja, Rushikesh Dudhat, Li Yi, and Evangelos Kalogerakis. 2022a. Apes: Articulated part extraction from sprite sheets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Chuan Yan, David Vanderhaeghe, and Yotam Gingold. 2020. A benchmark for rough sketch cleanup. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–14.

Wenwu Yang. 2017. Context-aware computer aided inbetweening. *IEEE transactions on visualization and computer graphics* 24, 2 (2017), 1049–1062.

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.

Deng Yu, Lei Li, Youyi Zheng, Manfred Lau, Yi-Zhe Song, Chiew-Lan Tai, and Hongbo Fu. 2020. SketchDesc: Learning local sketch descriptors for multi-view correspondence. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 5 (2020).

Jian Zhao and Hui Zhang. 2022. Thin-plate spline motion model for image animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3657–3666.