

Deep Geometrized Cartoon Line Inbetweening

Li Siyao¹ Tianpei Gu^{2*} Weiye Xiao³ Henghui Ding¹ Ziwei Liu¹ Chen Change Loy¹ [✉]
¹S-Lab, Nanyang Technological University ²Lexica ³Southeast University

{siyao002, henghui.ding, ziwei.liu, ccloy}@ntu.edu.sg, gutianpei@ucla.edu, 230189776@seu.edu.cn

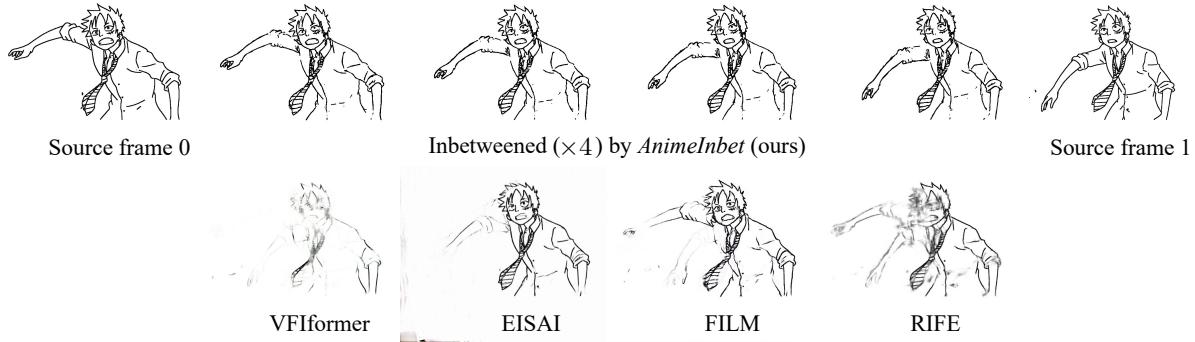


Figure 1: **Inbetweening on two source cartoon line drawings of *Monkey D. Luffy* extracted from *ONE PIECE*.** We compare our proposed *AnimeInbet* with state-of-the-art frame interpolation methods VFIfomer [14], EISAI [5], FILM [23] and RIFE [6].

Abstract

We aim to address a significant but understudied problem in the anime industry, namely the inbetweening of cartoon line drawings. Inbetweening involves generating intermediate frames between two black-and-white line drawings and is a time-consuming and expensive process that can benefit from automation. However, existing frame interpolation methods that rely on matching and warping whole raster images are unsuitable for line inbetweening and often produce blurring artifacts that damage the intricate line structures. To preserve the precision and detail of the line drawings, we propose a new approach, *AnimeInbet*, which geometrizes raster line drawings into graphs of endpoints and reframes the inbetweening task as a graph fusion problem with vertex repositioning. Our method can effectively capture the sparsity and unique structure of line drawings while preserving the details during inbetweening. This is made possible via our novel modules, i.e., vertex geometric embedding, a vertex correspondence Transformer, an effective mechanism for vertex repositioning and a visibility predictor. To train our method, we introduce Mixamo-Line240, a new dataset of line drawings with ground truth vectorization and matching labels. Our experiments demonstrate that *AnimeInbet* synthesizes high-quality, clean, and

complete intermediate line drawings, outperforming existing methods quantitatively and qualitatively, especially in cases with large motions. Data and code are available at <https://github.com/lisiyao21/AnimeInbet>.

1. Introduction

Cartoon animation has undergone significant transformations since its inception in the early 1900s, when consecutive frames were manually drawn on paper. Although automated techniques now exist to assist with some specific procedures during animation production, such as colorization [22, 32, 10, 39, 4] and special effects [38], the core element – the line drawings of characters – still needs hand-drawing each frame individually, making 2D animation a labor-intensive industry. Developing an automated algorithm that can produce intermediate line drawings from two input key frames, commonly referred to as “inbetweening”, has the potential to significantly improve productivity.

Line inbetweening is not a trivial subset of general frame interpolation, as the structure of line drawings is extremely sparse. Unlike full-textured images, line drawings contain only around 3% black pixels, with the rest of the image being white background. As illustrated in Figure 2, this poses two significant challenges for existing raster-image-based frame interpolation methods. **1)** The lack of texture in line drawings makes it challenging to compute pixel-wise correspondence

* Corresponding author. *Work completed at UCLA.

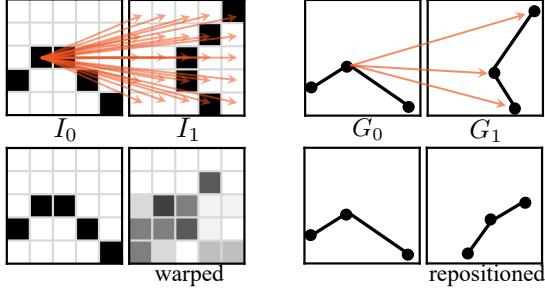


Figure 2: **Raster vs geometrized inbetweening.** Top: search space of a pixel (left) vs a vertex (right) in matching. Bottom: pixel warping/sampling (left) vs vertex repositioning (right).

accurately in frame interpolation. One pixel can have many similar matching candidates, leading to inaccurate motion prediction. **2)** The warping and blending used in frame interpolation can blur the salient boundaries between the line and the background, leading to a significant loss of detail.

To address the challenges posed by line inbetweening, we propose a novel deep learning framework called *AnimeInbet*, which inbetrwens line drawings in a geometrized format instead of raster images. Specifically, the source images are transformed into vector graphs, and the goal is to synthesize an intermediate graph. This reformulation can overcome the challenges discussed earlier in this paper. As illustrated in Figure 2, the matching process in the geometric domain is conducted on concentrated geometric endpoint vertices, rather than all pixels, reducing potential ambiguity and leading to more accurate correspondence. Moreover, the repositioning does not change the topology of the line drawings, enabling preservation of the intricate and meticulous line structures. Compared to existing methods, our proposed *AnimeInbet* framework can generate clean and complete intermediate line drawings, as demonstrated in Figure 1.

The core idea of our proposed *AnimeInbet* framework is to find matching vertices between two input line drawing graphs and then reposition them to create a new intermediate graph. To achieve this, we first design a vertex encoding strategy that embeds the geometric features of the endpoints of sparse line drawings, making them distinguishable from one another. We then apply a vertex correspondence Transformer to match the endpoints between the two input line drawings. Next, we propagate the shift vectors of the matched vertices to unmatched ones based on the similarities of their aggregated features to realize repositioning for all endpoints. Finally, we predict a visibility mask to erase the vertices and edges occluded in the inbetweened frame, ensuring a clean and complete intermediate frame.

To facilitate supervised training on vertex correspondence, we introduce *MixamoLine240*, the first line art dataset with ground truth geometrization and vertex matching labels. The

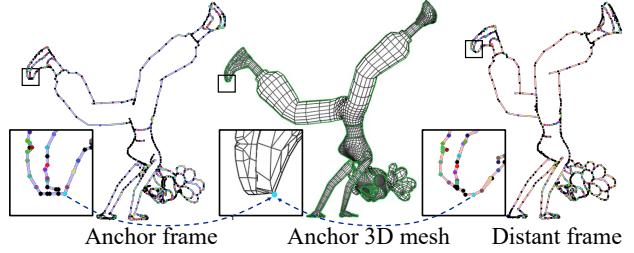


Figure 3: **Geometrized line art in *MixamoLine240*.** 2D endpoints and connected lines are projected from vertices and edges of orinal 3D mesh. Endpoints indexed to unique 3D vertices are matched (marked in the same colors).

2D line drawings in our dataset are selectively rendered from specific edges of a 3D model, with the endpoints indexed from the corresponding 3D vertices. By using 3D vertices as reference points, we ensure that the vertex matching labels in our dataset are accurate and consistent at the vertex level.

In a conclusion, our work contributes a new and challenging task of line inbetweening, which could facilitate one of the most labor-intensive art production processes. We also propose a new method that outperforms existing solutions, and introduce a new dataset for comprehensive training.

2. Related Work

Frame Interpolation. Frame interpolation is a widely studied task in recent years, involving synthesizing intermediate frames from existing ones. Many approaches have been proposed [13, 19, 20, 7, 17, 34, 18, 21, 26, 6, 23, 5, 14, 11], such as those that use optical flows or deep networks to search for matching areas and warp them to proper intermediate locations. Among the most recent algorithms, RIFE [6] directly predicts intermediate flows to warp the input frames and blends the warped frames into intermediate ones by a visible mask. VFformer [14] adopts the same idea to predict the intermediate flows but proposes a Transformer to synthesize the intermediate from both warped images and features. Reda *et al.* [23] design a scale-agnostic feature pyramid to predict the intermediate flows and warp frames in a hierarchical manner to handle extreme large motions. Siyao and Zhao *et al.* [30] propose a frame interpolation pipeline specific for 2D cartoon in the wild, while Chen and Zwicker [5] improves the perceptual quality by embedding an optical-flow based line aggregator. While these methods achieve impressive performance on raster natural or cartoon videos, their pixel-oriented nature are not suitable for inbetweening concise and sparse line arts, which can yield severe artifacts and are not feasible for real usage in anime creation.

Research on Anime. There has been increasing research interest in techniques to facilitate 2D cartoon creation, including sketch simplification [28, 27], vectorization

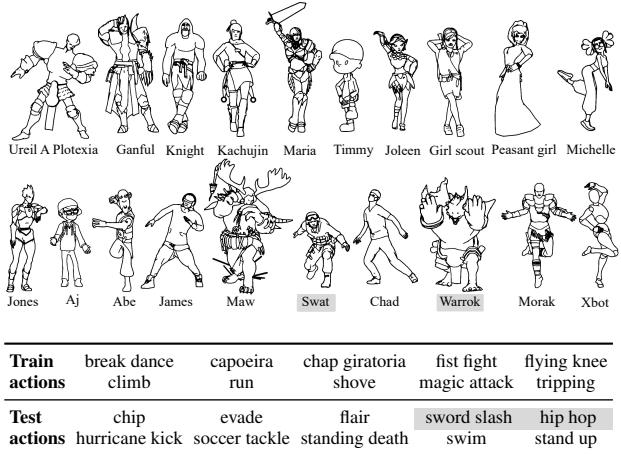


Figure 4: **Data composition.** Training and test sets are separately composed by 10 characters \times 10 actions. First & second rows are training & test characters, respectively. Shaded are for validation.

[40, 36, 15, 12], colorization [22, 32, 10, 39, 4], shading [38], head reenactment [8] and line-art-based cartoon generation [37]. While these studies may improve specific aspects of animation creation, the core line arts still rely on manual frame-by-frame drawing. Some sporadic rule-based methods have been developed for stroke inbetweening under strict conditions, but these methods lack the flexibility required for wider applications [35, 3]. Our work is the first to propose a deep learning-based method for inbetweening geometrized line arts. Additionally, we introduce vertex-wise correspondence datasets on line arts. It is noteworthy that existing datasets are not sufficiently ‘clean’ for our task since cartoon contour lines can cross the boundaries of motion, leading to incorrect corresponding labels at the vertex level [25, 29].

3. Mixamo Line Art Dataset

To facilitate training and evaluation of geometrized line inbetweening, we develop a large-scale dataset, named *MixamoLine240*, which consists of 240 sequences of consecutive line drawing frames, with 100 sequences for training and 140 for validation and testing. To obtain this vast amount of cartoon line data, we utilize a “Cel-shading” technique, *i.e.*, to use computer graphics software (Blender in this work) to render 3D resources into an anime-style appearance that mimics the hand-drawn artistry. Unlike previous works [25, 29] that only provide raster images, *MixamoLine240* also provides ground-truth geometrization labels for each frame, which include the coordinates of a group of vertices (V) and the connection topology (T). Additionally, we assign an index number ($R[i]$) to each 2D endpoint ($V[i]$) that refers to a unique vertex in the 3D mesh of the character, as illustrated in Figure 3, which can be further used to deduce the vertex-level correspondence. Specifically, given two frames I_0 and

Table 1: **Difficulty statistics with various frame gaps.**

	Frame gap →	0 (60 fps)	1 (30 fps)	5 (10 fps)	9 (6 fps)
Train	Occlusion rate (%)	14.8	21.5	37.8	46.6
	Avg. vtx shift	8.6	16.4	42.6	62.8
	Avg. max vtx shift	26.0	48.9	129.7	192.3
Test	Occlusion rate (%)	18.4	26.5	44.2	53.5
	Avg. vtx shift	7.8	14.9	38.9	57.0
	Avg. max vtx shift	23.8	45.0	119.3	173.5

I_1 in a sequence, the 3D reference IDs reveal the vertex correspondence $\{(i, j)\}$ for those vertices i in I_0 and j in I_1 having $R_0[i] = R_1[j]$, while the rest unmatched vertices are marked as occluded. This strategy allows us to produce correspondence pairs with arbitrary frame gaps to flexibly adjust the input frame rate during training. Next, we discuss the construction and challenges inherent in the data.

Data Construction. In Blender, the mesh structure of a 3D character remains stable, *i.e.*, the number of 3D vertex and the edge topology keep constant, when moving without additional subdivision modifier. We employ this property to achieve consistent line art rendering and accurate annotations for geometrization and vertex matching. As shown in Figure 3, the original 3D mesh contains all the necessary line segments required to represent the character in line art. During rendering, the visible outline from the camera’s perspective is selected based on the material boundary and the object’s edge. This process ensures that every line segment in the resulting raster image corresponds to an edge in the original mesh. The 2D endpoints of each line segment are simply the relevant 3D vertices projected onto the camera plane, referenced by the unique and consistent index of the corresponding 3D vertex. Meanwhile, since the 3D mesh naturally defines the vertex connections, the topology of the 2D lines can be transferred from the selective edges used for rendering. To prevent any topological ambiguity that may be caused by overlapped vertices in 3D space, we merge the endpoints that are within a Euclidean distance of 0.1 in the projected 2D space. This enables us to obtain both the raster line drawings and the accurate labels of each frame.

To create a diverse dataset, we used the open-source 3D material library Mixamo [1] and selected 20 characters and 20 actions, as shown in Figure 4. Each action has an average of 191 frames. We combined 10 characters and 10 actions to render 100 sequences, with a total of 19,930 frames as the training set. We then used the remaining 10 characters and 10 actions to render an 18,230-frame test set, ensuring that the training and testing partitions are exclusive. We also created a 44-sequence validation set, consisting of 20 unseen characters, 20 unseen actions, and 4 with both unseen character and action. To create this set, we combined the test characters “Swat” and “Warrok” and actions “sword slash” and “hip hop” with the training characters and actions. The

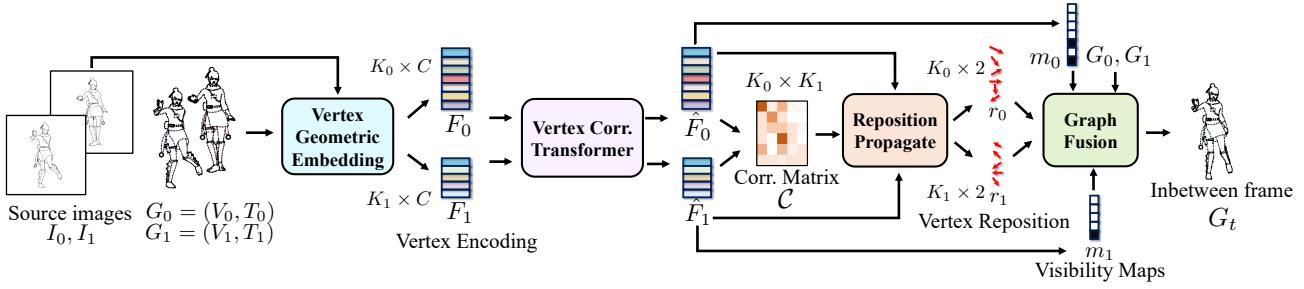


Figure 5: **Pipeline of proposed *AnimeInbet*.** Our framework is composed of four main parts: the vertex geometric embedding, the vertex correspondence Transformer, repositioning propagation and graph fusion. Given a pair of line images I_0 and I_1 and their vector graphs G_0 and G_1 , our method generates the intermediate frame G_t in geometrized format.

validation set contains 11,102 frames and was also rendered at 1080p resolution with a frame rate of 60 fps. To ensure consistency across all frames, we cropped and resized each frame to a unified 720×720 character-centered image.

Challenges. Table 1 summarizes the statistics that reflect the difficulty of the line inbetweening task under various input frame rates. With an increase in frame gaps, the inbetweening task becomes more challenging with larger motion magnitudes and higher occlusion percentages. For instance, when the frame gap is 9, the input frame rate becomes 6 fps, and the average vertex shift is 62.8 pixels. The mean value of the maximum vertex shift in a frame (“Avg. max vtx shift”) reaches 192.3 pixels, which is 27% of the image width. Additionally, nearly half of the vertices are unmatched in such cases, making line inbetweening a tough problem. Furthermore, the image composition of the test set is more complex than that of the training set. A training frame has an average of 1,256 vertices and 1,753 edges, while a test frame has an average of 1,512 vertices and 2,099 edges since the test set has more complex characters such as “Maw”.

4. Our Approach

An overview of the proposed line inbetweening framework, *AnimeInbet*, is depicted in Figure 5. Unlike existing frame interpolation methods that use raw raster images I_0 and I_1 , we process vector graphs $G_0 = \{V_0, T_0\}$ and $G_1 = \{V_1, T_1\}$ instead. The vertex coordinates in the images are represented by $V \in \mathbb{R}^{K \times 2}$, and the binary adjacency matrix is denoted by $T \in \{0, 1\}^{K \times K}$, where K denotes the number of vertices. The goal is to generate the intermediate graph G_t at time $t \in (0, 1)$. To this end, we first design a CNN-based vertex geometric embedding to encode V_0 and V_1 to features F_0 and F_1 , respectively, as detailed in Section 4.1. Along with the embeddings, a vertex correspondence Transformer is proposed to aggregate the mutuality of vertex features to \hat{F}_0 and \hat{F}_1 by alternating self- and cross-attention layers (Section 4.2). The aggregated features are used to compute the correlation matrix $C \in \mathbb{R}^{K_0 \times K_1}$ and to induce the vertex matching by row-wise and column-wise argmax.

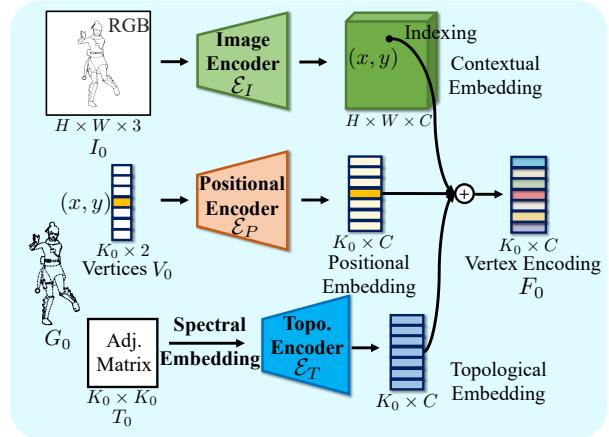


Figure 6: **Vertex Geometric Embedding.** The goal is to obtain discriminative and meaningful features to describe each vertex.

In cases where vertices are occluded during large motion, we adopt a self-attention-based layer to propagate the vertex shifts from matched vertices to the unmatched, and obtain repositioning vectors $r_0 \in \mathbb{R}^{K_0 \times 2}$ and $r_1 \in \mathbb{R}^{K_1 \times 2}$ for all vertices (Section 4.3). Finally, we superpose the two input graphs based on the predicted correspondence, and we further refine the output by predicting visibility maps $m_0 \in \{0, 1\}^{K_0}$ and $m_1 \in \{0, 1\}^{K_1}$ to mask off those vertices of V_0 and V_1 that disappear in the intermediate frame, respectively, to obtain the final inbetweened line drawing G_t , as explained in Section 4.5.

Geometrizing Line Drawings. The process of creating artwork has become largely digital, allowing for direct export in vectorized format. However, for line drawings that only appear in raster images, there are various commercial software and open-source research projects available [40, 36, 15, 12] that can be used to convert the raster images into the required vectorized input format. We will ablate the performance of line vectorization in our experiments.

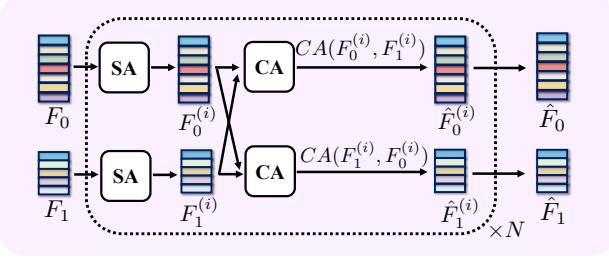


Figure 7: **Vertex Correspondence Transformer.** SA and CA represent self-attention and cross-attention, respectively.

4.1. Vertex Geometric Embedding

Discriminative features for each vertex are desired to achieve accurate graph matching. Line graphs are different from general graphs as the spatial position of endpoint vertices, in addition to the topology of connections, determines the geometric shape of the line. The geometric graph embedding for line art is hence designed to comprise three parts: **1) image contextual embedding, 2) positional embedding, and 3) topological embedding**, as shown in Figure 6.

For image contextual embedding, we use a 2D CNN \mathcal{E}_I to extract deep contextual features within the same size of the input raster image I . Then, for each vertex $V_0[i] := (x, y)$ we index feature $\mathcal{E}_I(I)[(x, y)]$ as the image embedding for the i -th vertex. As to the positional embedding, we employ a 1D CNN \mathcal{E}_P to map each vertex coordinate (x, y) to a C -dimensional feature. To include the topological information into a lower dimensional feature, we first conduct spectral embedding [2] \mathcal{S} on the binary adjacency matrix T , which involves an eigenvector decomposition on the Laplacian matrix of the graph, then feed the spectral embedding to a subsequent 1D CNN \mathcal{E}_T . The final geometric graph embedding is formulated as

$$F_0 = \mathcal{E}_I(I_0)[V_0] + \mathcal{E}_P(V_0) + \mathcal{E}_T(\mathcal{S}(T_0)). \quad (1)$$

We obtain F_1 in the same way.

4.2. Vertex Correspondence Transformer

We use geometric features F_0 and F_1 to establish a vertex-wise correspondence between G_0 and G_1 . Specifically, we compute a correlation matrix between vertex features and identify the matching pair as those with the highest value across both the row and the column of the matrix. Prior to this step, we apply a Transformer that aggregates the mutual consistency both intra- and inter-graph.

Mutual Aggregation. Following [24, 31], we employ a cascade of alternating self- and cross-attention layers to aggregate the vertex feature. In a self-attention layer, all queries, keys and values are derived from the single source feature,

$$SA(F_0) = \text{softmax} \left(\frac{\mathcal{Q}(F_0)\mathcal{K}^T(F_0)}{\sqrt{C}} \right) \mathcal{V}(F_0), \quad (2)$$

where \mathcal{Q} , \mathcal{K} and \mathcal{V} represent MLPs for query, key and value, respectively; while in the cross-attention layer, the keys and values are computed from another feature:

$$CA(F_0, F_1) = \text{softmax} \left(\frac{\mathcal{Q}(F_0)\mathcal{K}^T(F_1)}{\sqrt{C}} \right) \mathcal{V}(F_1). \quad (3)$$

After N layers of rotating self- and cross-attention layers as shown in Figure 7, we obtain aggregated feature \hat{F}_0 and \hat{F}_1 . In the aggregation, each vertex is represented as an attentional pooling of all other vertices within the same graph and across the two graphs achieving a full fusion of information with mutual dependencies.

Correlation Matrix and Vertex Matching. We compute the correlation matrix \mathcal{P} as $\mathcal{P} = \frac{\hat{F}_0\hat{F}_1^T}{\sqrt{C}}$. We further apply a differentiable optimal transport (OT) [24] to improve the dual selection consistency and obtain $\hat{\mathcal{P}} = OT(\mathcal{P})$. Then, we predict the one-way matching from G_0 to G_1 and vice versa as $\arg \max$ indices across rows and columns:

$$\begin{cases} \mathcal{M}_{0 \rightarrow 1} = \{(i, j) | j = \arg \max \hat{\mathcal{P}}_{i,:}, i = 0, \dots, K_0 - 1\} \\ \mathcal{M}_{1 \rightarrow 0} = \{(i, j) | i = \arg \max \hat{\mathcal{P}}_{:,j}, j = 0, \dots, K_1 - 1\}. \end{cases} \quad (4)$$

A vertex pair is selected into the final correspondence if it is mutually consistent and its correlation value is larger than θ :

$$\hat{\mathcal{M}} = \{(i, j) | (i, j) \in \mathcal{M}_{0 \rightarrow 1} \cap \mathcal{M}_{1 \rightarrow 0}, \hat{\mathcal{P}}_{i,j} > \theta\}. \quad (5)$$

Otherwise, vertices will be considered to be occluded.

4.3. Repositioning Propagation

Fused vertices (i, j) from vertex correspondence can be linearly relocated to $tV_0[i] + (1 - t)V_1[j]$ in intermediate graph G_t based on time t . However, the positions of the unmatched vertices in G_t are still unknown. To reposition these vertices, we design an attention-based scheme similar to Xu *et al.* [33] to predict bidirectional shift vectors $r_{0 \rightarrow 1}$ and $r_{1 \rightarrow 0}$ for V_0 and V_1 , respectively. Formally,

$$\begin{cases} r_{0 \rightarrow 1} = \text{softmax} \left(\frac{\hat{F}_0\hat{F}_0^T}{\sqrt{C}} \right) (\text{softmax}(\hat{\mathcal{P}})V_1 - V_0) \\ r_{1 \rightarrow 0} = \text{softmax} \left(\frac{\hat{F}_1\hat{F}_1^T}{\sqrt{C}} \right) (\text{softmax}(\hat{\mathcal{P}}^T)V_0 - V_1) . \end{cases} \quad (6)$$

We then compute the final repositioning vectors as follows:

$$r_0[i] = \begin{cases} V_1[j] - V_0[i], & \text{if } \exists j \text{ s.t. } (i, j) \in \hat{\mathcal{M}}, \\ r_{0 \rightarrow 1}[i], & \text{otherwise,} \end{cases} \quad (7)$$

while r_1 is computed in a similar way.

In this step, the motion vector $r_{0 \rightarrow 1}$ of an unmatched vertex $V_0[i]$ is computed as a softmax average of shifts to all vertices in G_1 , *i.e.*, $\text{softmax}(\hat{\mathcal{P}}_{i,:})V_1 - V_0$. It is then refined by attention pooling from matched vertices, based on self-similarity given by $\hat{F}_0\hat{F}_0^T/\sqrt{C}$. Vertices are reasonably repositioned in the new vector graph after this step.

Table 2: **Quantitative evaluations of state-of-the-art frame interpolation methods using Chamfer Distance** (reported in units of $\times 10^{-5}$, with lower values indicating better performance). The first place and runner-up are highlighted in bold and underlined, respectively.

Method	Validation Set				Test Set			
	gap = 1	gap = 5	gap = 9	Avg.	gap = 1	gap = 5	gap = 9	Avg.
VFIformer [14]	7.82	26.04	50.71	28.19	7.62	27.55	50.68	28.62
RIFE [6]	5.02	27.79	49.81	27.54	5.85	28.91	51.08	28.61
EISAI [5]	5.66	27.64	49.43	27.57	6.02	29.14	52.36	29.17
FILM [23]	3.18	16.84	30.74	16.92	3.50	17.94	33.51	18.31
<i>AnimeInbet</i> (ours)	2.20	11.12	21.27	11.53	2.80	12.69	23.21	12.90
<i>AnimeInbet-VS</i> (ours)	<u>2.62</u>	<u>11.43</u>	<u>22.36</u>	<u>12.14</u>	<u>3.44</u>	<u>13.41</u>	<u>23.67</u>	<u>13.51</u>

4.4. Visibility Prediction and Graph Fusion

To handle occlusions in the source line arts, we use a three-layer MLP to predict binary visibility maps m_0 and m_1 for the input graphs, obtained as $m_0 = \text{MLP}(\hat{F}_0)$ and $m_1 = \text{MLP}(\hat{F}_1)$. Then, we merge the vertices to V_t in the two graphs according to the following rule:

$$\begin{aligned} V_t = & \left\{ (1-t)V_0[i] + tV_1[j] \mid (i, j) \in \hat{\mathcal{M}} \right\} \\ & \cup \left\{ V_0[i] + t \cdot r_0[i] \mid i \notin \hat{\mathcal{M}}, m_0[i] = 1 \right\} \\ & \cup \left\{ V_1[j] + (1-t)r_1[j] \mid j \notin \hat{\mathcal{M}}, m_1[j] = 1 \right\}, \end{aligned} \quad (8)$$

where we implement the repositioning that is compatible with arbitrary time $t \in (0, 1)$. As to T_t , we union all original connections if both endpoint vectors are both visible in G_t . Or formally, $T_t[\tilde{i}][\tilde{j}] = T_t[\tilde{j}][\tilde{i}] = 1$ if $T_0[i][j] = 1$ or $T_1[i][j] = 1$, where (i, j) and (\tilde{i}, \tilde{j}) are the vertex indices in the original graph and the merged one.

4.5. Learning

The training objective of *AnimeInbet* composes of three terms: $\mathcal{L} = \mathcal{L}_c + \mathcal{L}_r + \mathcal{L}_m$, where the \mathcal{L}_c , \mathcal{L}_r and \mathcal{L}_m are used to supervise the learning of vertex matching $\hat{\mathcal{M}}$, repositioning vectors r_0 and r_1 , and visibility masks m_0 and m_1 , respectively. \mathcal{L}_c is to enlarge the correlation values of ground truth pairs and is defined as:

$$\mathcal{L}_c = -\frac{1}{|\mathcal{M}^{GT}|} \sum_{(i,j) \in \mathcal{M}^{GT}} \log \hat{\mathcal{P}}_{i,j}, \quad (9)$$

where \mathcal{M}^{GT} is the ground truth matching labels. For \mathcal{L}_r and \mathcal{L}_m , we regress $r_{0 \rightarrow 1}$, $r_{1 \rightarrow 0}$, m_0 , and m_1 as follows:

$$\begin{aligned} \mathcal{L}_r &= \frac{1}{K_0} \|r_{0 \rightarrow 1} - r_{0 \rightarrow 1}^{GT}\|_1 + \frac{1}{K_1} \|r_{1 \rightarrow 0} - r_{1 \rightarrow 0}^{GT}\|_1 \\ \mathcal{L}_m &= \text{BCE}^w (\sigma(m_0), m_0^{GT}) + \text{BCE}^w (\sigma(m_1), m_1^{GT}), \end{aligned} \quad (10)$$

where σ represents the sigmoid function, and BCE^w is the binary cross-entropy loss with bias weight w . However, since

the shift vectors of occluded vertices cannot be obtained directly by subtracting the matched vertices, we conduct a frame-by-frame backtrack to generate pseudo labels to support the point-wise supervision of the repositioning vector and visibility maps.

Pseudo Labels of Repositioning and Visibility. Assume $G^{(0)}$ and $G^{(Z)}$ are the 0-th and the Z -th frames in a training sequence, which are used for two input line sources. Although there can exist many unmatched vertices in the two graphs when the gap Z is large, the matching rate between adjacent frames (gap = 0) is relatively high according to Table 1. Based on this, we iteratively backtrack a shift vector $r^{(z)}$ from the $G^{(Z)}$ to $G^{(0)}$:

$$r^{(z)}[i] = \begin{cases} V^{(z+1)}[j] - V_{(z)}[i] + r^{(z+1)}, & \text{if } i, j \text{ is matched} \\ \frac{1}{|\mathcal{N}_i|} \sum_{k \in \mathcal{N}_i} r^{(z)}[k], & \text{otherwise} \end{cases} \quad (11)$$

where \mathcal{N}_i regards to the neighbors of the i -th vertex in $G^{(z)}$ and $r^{(Z)}$ is initialized to be 0. The termination $r^{(0)}$ of the backtrack is regarded as the pseudo repositioning label $r_{0 \rightarrow 1}^{GT}$. As to the visibility labels, we first deuce $r_{0 \rightarrow t}^{GT}$ as above and compute m_0^{GT} as

$$m_0^{GT}[i] = \begin{cases} 1, & \text{if } V_0[i] + r_{0 \rightarrow t}^{GT} \in \tilde{I}_t, \\ 0, & \text{otherwise,} \end{cases} \quad (12)$$

where \tilde{I}_t is I_t dilated by a 3×3 kernel. $r_{1 \rightarrow 0}^{GT}$ and m_1^{GT} are computed in reversed order.

5. Experiments

Implementation Details. In the vertex geometric embedding module, the image encoder \mathcal{E}_I is implemented as a three-layer 2D CNN, while the positional encoder \mathcal{E}_P and the topological encoder \mathcal{E}_T are 1D CNNs with a kernel size of 1. Encoding feature C is 128 in our experiments. Before feeding vertex coordinates V into \mathcal{E}_P , V are first normalized to the scale between $(-1, 1)$; the dimension of the spectral embedding feature is 64. Threshold θ in Equation 5 is 0.2. In both training and evaluation, intermediate time t is 0.5, which regards the center frame between I_0 and I_1 . The detailed network structures are provided in the supplementary

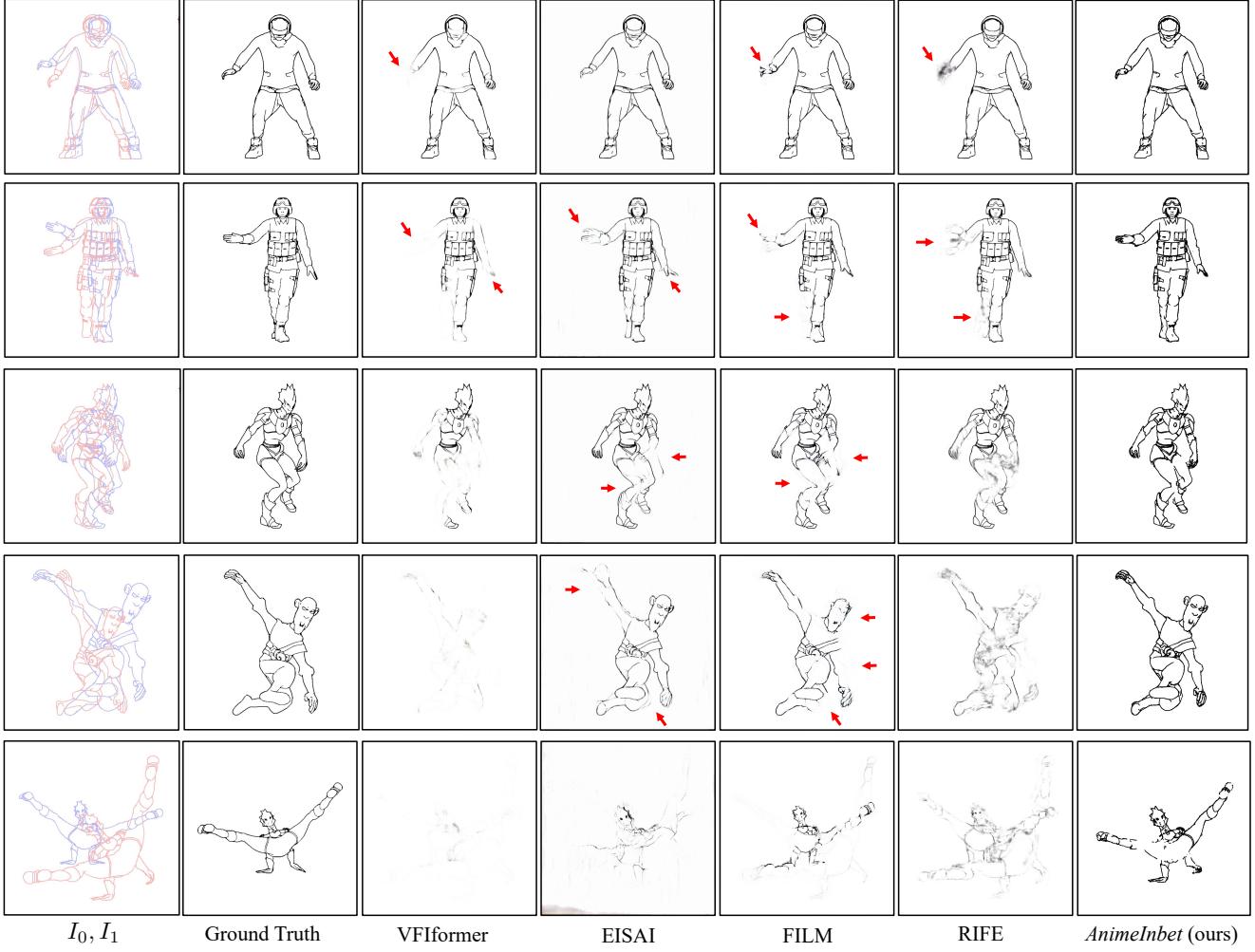


Figure 8: **Inbetweening results on MixamoLine240 test set.** Examples are arranged from small (top) to large (bottom) motion magnitudes.

file. We use Adam [9] optimizer with a learning rate of 1×10^{-4} to train the *AnimeInbet* for 70 epochs, where we first solely supervise the network using the correspondence loss \mathcal{L}_c for the 50 epochs, and then adopt the full loss \mathcal{L} for the rest 20 epochs. Bias weight w in \mathcal{L}_m is 0.2. Since vertex numbers differ in frames, we feed one pair of input frames each time but adopt gradient accumulation for a mini-batch size of 8. The model is trained with an NVIDIA Tesla V100 GPU for about five days. During the test, G_t is visualized as a raster image by `cv2.line` function with a line width of 2 pixels. We evaluate our model on both ground truth vectorization labels (noted as “*AnimeInbet*”) and those vectorized from VirtualSketcher [15] (noted as “*AnimeInbet-VS*”), to simulate the cases when input anime drawing are vector and raster, respectively.

Evaluation Metric. Following [16, 5], we adopt the chamfer distance (CD) as the evaluation metric, which has been initially introduced to measure the similarity between two

point clouds. Formally, CD is computed as:

$$CD(I_t, I_t^{GT}) = \frac{1}{HWd} \sum (I_t DT(I_t^{GT}) + I_t^{GT} DT(I_t)), \quad (13)$$

where I_t and I_t^{GT} are predicted binary lines and ground truth, while H , W and d are image height, width, and a search diameter [5], respectively. DT denotes the Euclidean distance transform. To transfer predicted raster images into binary sketches, we threshold pixels smaller than 0.99 times the maximum value to 0.

5.1. Comparison to Existing Methods

Since there is no existing geometrized line inbetweening study that we can directly compare our proposed model with, we set several state-of-the-art raster-image-based frame interpolation methods as baselines, including VFformer [14], RIFE [6], EISAI [5], FILM [23]. Specifically, EISAI is originally intended for 2D animation and embeds an opti-

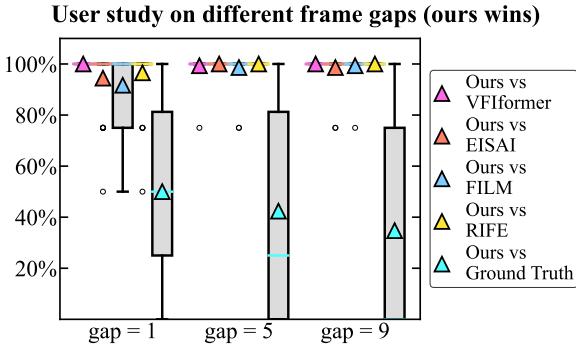


Figure 9: **Statistics of user study.** In the boxplot, triangles and colored lines represent mean and median values, respectively. Circles are outliers beyond $1.5 \times$ interquartile range (3σ in a normal distribution).

cal flow-based contour aggregator. We test each model’s performance on frame pairs within frame gaps of 1, 5 and 9, respectively. For fairness, we finetune each compared method on the training set of *MixamoLine240* with relative frame gaps using a learning rate of 1×10^{-6} for five epochs.

As shown in Table 2, our *AnimeInbet* favorably outperforms all compared methods on both the validation set and the test set of *MixamoLine240*. On the validation set, our approach achieves an average CD value of 11.53, representing a significant improvement over the best-performing compared method, FILM, with over 30% enhancement. Upon closer inspection, the advantage of *AnimeInbet* becomes more pronounced as the frame gap increases (0.98, 5.72 and 9.47 for gaps of 1, 5, and 9, respectively), indicating that our method is more robust in handling larger motions. On the test set, our method maintains its lead over the other compared methods, with improvements of 0.70 (20%), 5.25 (29%), and 10.30 (31%) from the best-performing compared method FILM for the frame gaps of 1, 5, and 9, respectively. Given that both the characters and actions in the test set are new, our method’s superiority on the test set provides more convincing evidence of its advantages over the existing frame interpolation methods.

To illustrate the advantages of our method, we present several inbetweening results in Figure 8. We arranged these examples in increasing levels of difficulty from top to bottom. When the motion is simple, compared methods can interpolate a relatively complete shape of the main body of the drawing. However, they tend to produce strong blurring (RIFE) or disappearance (VFIfomer, EISAI, and FILM) of noticeable moving compositions (indicated by red arrows). In contrast, our method maintains a concise line structure in these key areas. When the input frames involve the whole body’s movement within large magnitudes, the intermediate frames predicted by the compared methods become indistinguishable and patchy, rendering the results invalid for further

Table 3: **Ablation study on vertex encoding.**

\mathcal{E}_I	\mathcal{E}_P	\mathcal{E}_T	Acc. (%)	Valid Acc. (%)	CD (\downarrow)
✓	✗	✗	51.66	31.01	12.30
✓	✓	✗	61.87	55.62	11.55
✓	✗	✓	59.28	45.45	11.86
✓	✓	✓	65.51	61.28	11.12

Table 4: **Ablation study on repositioning and visibility mask.**

Method	CD (\downarrow)
w/o. repositioning propagation	23.62
w/o. visibility mask	12.81
full model	11.12

use. However, our *AnimeInbet* method can still preserve the general shape in the correct positions, even with a partial loss of details, which can be easily fixed with minor manual effort.

User Study. To further evaluate the visual performance of our methods, we conduct a user study among 36 participants. For each participant, we randomly show 60 pairs, each composed of a result of *AnimeInbet* and that of a compared method, and ask the participant to select the better. To allow participants to take temporal consistency into the decision, we display these results in GIF formats formed by triplets of input frames and the inbetweened one. The winning rates of our method are shown in Figure 9, where *AnimeInbet* wins over 92% versus the compared methods. Notably, for “gap = 5” and “gap = 9” slots, the winning rates of our methods are close to 100% with smaller deviations than “gap = 1”, suggesting the advantages of our method on cases within large motions.

5.2. Ablation Study

Embedding Features. To investigate the effectiveness of the three types of embeddings mentioned in Section 4.1, we trained several variants by removing the corresponding modules. As shown in Table 3, for each variant, we list the matching accuracy for all vertices (“Acc.”), the accuracy for non-occluded vertices (“Valid Acc.”) and the final CD values of inbetweening on the validation set (gap = 5). If removing the positional embedding \mathcal{E}_P , the “Valid Acc.” and the CD value drop 15.83% and 0.74, respectively; while the lacking of topological embedding \mathcal{E}_T lowers “Valid Acc.” by 5.66% and worsens CD by 0.43, which reveals the importance of these two components.

Repositioning Propagation and Visibility Mask. We demonstrate the contribution of repositioning propagation (prepos. prop.) and visibility mask (vis. mask) both quantitatively and qualitatively. As shown in Table 4, without repositioning propagation, the CD value will be sharply wors-

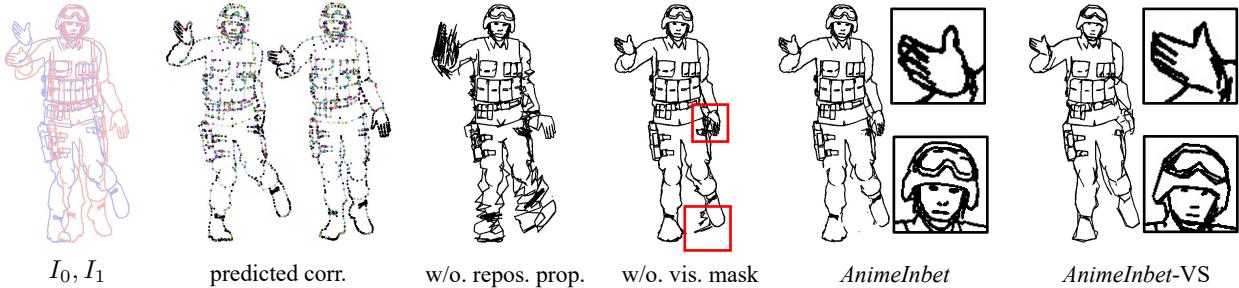


Figure 10: **Visualization of ablation study.** In predicted correspondence, matched vertices are marked in the same colors, while unmatched are black (please zoom in).

Table 5: Ablation study on data influence.

Validation data (gap = 5)	Occ. (%)	Shift	CD (\downarrow)
Unseen characters (2×10)	34.30	44.59	14.70
Unseen actions (10×2)	37.71	31.53	8.98
Unseen both (2×2)	34.10	29.62	9.98

ened by 12.50 (112%), while the lacking of visibility mask will also make a drop of 1.69 (15%). An example is shown in Figure 10, where “w/o. repos. prop.” appears within many messy lines due to undefined positions for those unmatched vertices, while “w/o. vis. mask” shows some redundant segments (red box) after repositioning; the complete *AnimeInbet* can resolve these issues and produce a clean yet complete result.

Geometrizer. As shown in Table 2, the quantitative metrics of *AnimeInbet-VS* are generally worse by around 0.6 compared to *AnimeInbet*. This is because VirtualSketcher [15] does not vectorize the line arts as precisely as the ground truth labels (average vertex number 587 vs 1,351). As shown in Figure 10, the curves in “*AnimeInbet-VS*” become sharper and lose some details, which decreases the quality of the inbetweening frame. Using a more accurate geometrizer would lead to higher quality inbetweening results for raster image inputs.

Data Influence. As mentioned in Section 3, we created a validation set composed of 20 sequences of unseen characters but seen actions, 20 of unseen actions but seen characters and 4 of unseen both to explore the influence on data. Our experiment finds that whether the characters or the actions are seen does not fundamentally influence the inbetweening quality, while the motion magnitude is the key factor. As shown in Table 5, the CD value of unseen characters is 14.70, which is over 47% worse than that of unseen both due to larger vertex shifts (44.59 vs 29.62), while the difference between the CD values of unseen actions and unseen both is around 10% under similar occlusion rates and shifts.

6. Conclusion

In this study, we address the practical problem of cartoon line inbetweening and propose a novel approach that treats line arts as geometrized vector graphs. Unlike previous frame interpolation tasks on raster images, our approach formulates the inbetweening task as a graph fusion problem with vertex repositioning. We present a deep learning-based framework called *AnimeInbet*, which shows significant gains over existing methods in terms of both quantitative and qualitative evaluation. To facilitate training and evaluation on cartoon line inbetweening, we also provide a large-scale geometrized line art dataset, *MixamoLine240*. Our proposed framework and dataset facilitate a wide range of applications, such as anime production and multimedia design, and have significant practical implications.

Acknowledgement. This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-PhD/2021-01-031[T]). It is also supported under the RIE2020 Industry Alignment Fund Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s). This study is partially supported by NTU NAP, MOE AcRF Tier 1 (2021-T1-001-088).

References

- [1] Mixamo. <https://www.mixamo.com/>. 3
- [2] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003. 5
- [3] Leonardo Carvalho, Ricardo Marroquim, and Emilio Vital Brazil. Dilight: Digital light table–inbetweening for 2d animations using guidelines. *Computers & Graphics*, 2017. 3
- [4] Evan Casey, Víctor Pérez, and Zhuoru Li. The animation transformer: Visual correspondence via segment matching. In *ICCV*, 2021. 1, 3
- [5] Shuhong Chen and Matthias Zwicker. Improving the perceptual quality of 2d animation interpolation. In *ECCV*, 2022. 1, 2, 6, 7

- [6] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *ECCV*, 2022. 1, 2, 6, 7
- [7] Huaiyu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In *CVPR*, 2018. 2
- [8] Kangyeol Kim, Sunghyun Park, Jaeseong Lee, Sunghyo Chung, Junsoo Lee, and Jaegul Choo. Animeceleb: Large-scale animation celebheads dataset for head reenactment. In *ECCV*, 2022. 3
- [9] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 7
- [10] Johannes Kopf and Dani Lischinski. Digital reconstruction of halftoned color comics. *ACM TOG*, 31(6), 2012. 1, 3
- [11] Xiaoyu Li, Bo Zhang, Jing Liao, and Pedro V. Sander. Deep sketch-guided cartoon video inbetweening. *TVCG*, 2020. 2
- [12] Songtao Liu, Jin Huang, and Hao Zhang. End-to-end line drawing vectorization. In *AAAI*, 2022. 3, 4
- [13] Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *CVPR*, 2017. 2
- [14] Liying Lu, Ruizheng Wu, Huajia Lin, Jiangbo Lu, and Jiaya Jia. Video frame interpolation with transformer. In *CVPR*, 2022. 1, 2, 6, 7
- [15] Haoran Mo, Edgar Simo-Serra, Chengying Gao, Changqing Zou, and Ruomei Wang. General virtual sketching framework for vector line art. In *SIGGRAPH*, 2021. 3, 4, 7, 9
- [16] Rei Narita, Keigo Hirakawa, and Kiyoharu Aizawa. Optical flow based line drawing frame interpolation using distance transform to support inbetweenings. In *ICIP*, 2019. 7
- [17] Simon Niklaus and Feng Liu. Context-aware synthesis for video frame interpolation. In *CVPR*, 2018. 2
- [18] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *CVPR*, 2020. 2
- [19] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *CVPR*, 2017. 2
- [20] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *ICCV*, 2017. 2
- [21] Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. In *ECCV*, 2020. 2
- [22] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. Manga colorization. *ACM TOG*, 25(3), 2006. 1, 3
- [23] Fitsum Reda, Janne Kontkanen, Eric Tabellion, Deqing Sun, Caroline Pantofaru, and Brian Curless. Film: Frame interpolation for large motion. In *ECCV*, 2022. 1, 2, 6, 7
- [24] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 5
- [25] Maria Shugrina, Ziheng Liang, Amlan Kar, Jiaman Li, Angad Singh, Karan Singh, and Sanja Fidler. Creative flow+ dataset. In *CVPR*, 2019. 3
- [26] Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. Xvfi: extreme video frame interpolation. In *ICCV*, 2021. 2
- [27] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. Mastering sketching: Adversarial augmentation for structured prediction. *ACM TOG*, 37(1), 2018. 2
- [28] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. Learning to simplify: Fully convolutional networks for rough sketch cleanup. *ACM TOG*, 35(4), 2016. 2
- [29] Li Siyao, Yuhang Li, Bo Li, Chao Dong, Ziwei Liu, and Chen Change Loy. Animerun: 2d animation visual correspondence from open source 3d movies. In *NeurIPS*, 2022. 3
- [30] Li Siyao, Shiyu Zhao, Weijiang Yu, Wenxiu Sun, Dimitris Metaxas, Chen Change Loy, and Ziwei Liu. Deep animation video interpolation in the wild. In *CVPR*, 2021. 2
- [31] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *CVPR*, 2021. 5
- [32] D. Šýkora, J. Buriánek, and J. Žára. Unsupervised colorization of black-and-white cartoons. In *Int. Symp. NPAR*, 2004. 1, 3
- [33] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *CVPR*, 2022. 5
- [34] Xiangyu Xu, Li Siyao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. Quadratic video interpolation. In *NeurIPS*, 2019. 2
- [35] Wenwu Yang. Context-aware computer aided inbetweening. *IEEE TVCG*, 24(2):1049–1062, 2017. 3
- [36] Chih-Yuan Yao, Shih-Hsuan Hung, Guo-Wei Li, I-Yu Chen, Reza Adhitya, and Yu-Chi Lai. Manga vectorization and manipulation with procedural simple screentone. *IEEE TVCG*, 23(2), 2016. 3, 4
- [37] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023. 3
- [38] Lvmin Zhang, Jinyue Jiang, Yi Ji, and Chunping Liu. Smartshadow: Artistic shadow drawing tool for line drawings. In *ICCV*, 2021. 1, 3
- [39] Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. Two-stage sketch colorization. In *SIGGRAPH*, 2018. 1, 3
- [40] Song-Hai Zhang, Tao Chen, Yi-Fei Zhang, Shi-Min Hu, and Ralph R. Martin. Vectorizing cartoon animations. *IEEE TVCG*, 15(4), 2009. 3, 4