Improving the Perceptual Quality of 2D Animation Interpolation

Shuhong Chen[1] and Matthias Zwicker[1]

University of Maryland, College Park, MD 20742, USA
{shuhong,zwicker}@cs.umd.edu

Abstract. Traditional 2D animation is labor-intensive, often requiring animators to manually draw twelve illustrations per second of movement. While automatic frame interpolation may ease this burden, 2D animation poses additional difficulties compared to photorealistic video. In this work, we address challenges unexplored in previous animation interpolation systems, with a focus on improving perceptual quality. Firstly, we propose SoftsplatLite (SSL), a forward-warping interpolation architecture with fewer trainable parameters and better perceptual performance. Secondly, we design a Distance Transform Module (DTM) that leverages line proximity cues to correct aberrations in difficult solid-color regions. Thirdly, we define a Restricted Relative Linear Discrepancy metric (RRLD) to automate the previously manual training data collection process. Lastly, we explore evaluation of 2D animation generation through a user study, and establish that the LPIPS perceptual metric and chamfer line distance (CD) are more appropriate measures of quality than PSNR and SSIM used in prior art.

Keywords: animation, video frame interpolation

1 Introduction

Traditional 2D animators typically draw each frame manually; this process is incredibly labor-intensive, requiring large production teams with expert training to sketch and color the tens of thousands of illustrations required for an animated series. With the growing global popularity of the traditional style, studios are hard-pressed to deliver high volumes of quality content. We ask whether recent advancements in computer vision and graphics may reduce the burden on animators. Specifically, we study video frame interpolation, a method of automatically generating intermediate frames in a video sequence. In the typical problem formulation, a system is expected to produce a halfway image naturally interpolating two given consecutive video frames. In the context of animation, an animator could potentially achieve the same framerate for a sequence (or "cut") by manually drawing only a fraction of the frames, and use an interpolator to generate the rest.

Though there is abundant work on video interpolation, 2D animation poses additional difficulties compared to photorealistic video. Given the high manual

cost per frame, animators tend to draw at reduced framerates (e.g. "on the twos" or at 12 frames/second), increasing the pixel displacements between consecutive frames and exaggerating movement non-linearity. Unlike in natural videos with motion blur, the majority of animated frames can be viewed as stand-alone cel illustrations with crisp lines, distinct solid-color regions, and minute details. For this non-photorealistic domain with such different image and video features, even our understanding of how to evaluate generation quality is limited.

Previous animation-specific interpolation by Li et. al. (AnimeInterp [37]) approached some of these challenges by improving the optical flow estimation component of a deep video interpolation system by Niklaus et. al. (Softsplat [24]); in this paper, we build upon AnimeInterp by addressing some remaining challenges. Firstly, though AnimeInterp improved optical flow, it trained with an L_1 objective and did not modify the Softsplat feature extraction, warping, or synthesis components; this results in blurred lines/details and ghosting artifacts in supposedly solid-color regions. We alleviate these issues with architectural improvements in our proposed SoftsplatLite (SSL) model, as well as with an additional Distance Transform Module (DTM) that refines outputs using domain knowledge about line drawings. Secondly, though AnimeInterp provided a small ATD12k dataset of animation frame triplets, the construction of this dataset required intense manual filtering of evenly-spaced triplets with linear movement. We instead automate linear triplet collection from raw animation by introducing Restricted Relative Linear Discrepancy (RRLD), enabling large-scale dataset construction. Lastly, AnimeInterp only focused on PSNR/SSIM evaluation, which we show (through an exploratory user study) are less indicative of percieved quality than LPIPS [45] and chamfer line distance (CD). We summarize the contributions of this paper:

1. **SoftsplatLite (SSL)**: a forward-warping interpolation architecture with fewer trainable parameters and better perceptual performance. We tailor the feature extraction and synthesis networks to reduce overfitting, propose a simple infilling method to remove ghosting artifacts, and optimize LPIPS loss to preserve lines and details.
2. **Distance Transform Module (DTM)**: a refinement module with an auxiliary domain-specific loss that leverages line proximity cues to correct aberrations in difficult solid-color regions.
3. **Restricted Relative Linear Discrepancy (RRLD)**: a metric to quantify movement non-linearity from raw animation; this automates the previously manual training data collection process, allowing more scalable training.
4. **Perceptual user study**: we explore evaluation of 2D animation generation, establishing the LPIPS perceptual metric and chamfer line distance (CD) as more appropriate quality measures than PSNR/SSIM used in prior art.

2 Related Work

Much recent work has been published on photorealistic video interpolation. Broadly, these works fall into phase-based [21, 22], kernel-based [26, 25], and

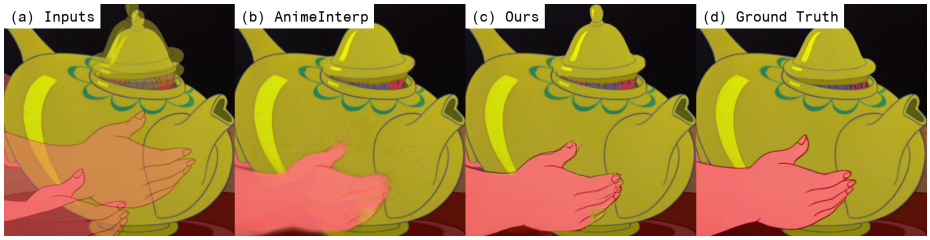

Fig. 1. We improve the perceptual quality of 2D animation interpolation from previous work. **(a)** Overlaid input images to interpolate; **(b)** AnimeInterp by Li et. al. [37]; **(c)** Our proposed method; **(d)** Ground truth interpolation. Note the destruction of lines in (b) compared to (c), and the patchy artifacts ghosted on the teapot in (b). Our user study validates our focus on perceptual metrics and artifact removal.

flow-based methods [24, 16, 28, 43], with others using a mix of techniques [1, 2, 6]. The most recent state-of-the-art has seen more flow-based methods [24, 28], following corresponding advancements in optical flow estimation [14, 39, 15, 40]. Flow-based methods can be further split by forward [24], or backward [28] warping. The prior art most directly related to ours is AnimeInterp, by Li et. al. [37]. While they laid the groundwork for the problem specific to the traditional 2D animation domain, their system had many shortcomings that we overcome as described in the introduction section.

Even though we focus on animations "post-production" (i.e. interpolating complete full-color sequences), there is also a body of work on automating more specific components of animation production itself. For example, sketch simplification [36, 35] is a popular topic with applications to speeding up animation "tie-downs" and "cleanups". There are systems for synthesizing "in-between" line drawings from sketch keyframes in both raster [23, 44] and vector [42, 7] form. While the flow-based in-betweening done by Narita et. al. [23] shares similarity to our work (such as the use of chamfer distance and forward warping), their system composed pretrained models without performing any form of training. Another related problem is sketch colorization, with application to both single illustrations [31] and animations [30, 20, 5]. These works unsurprisingly highlight the foundational role of lines and sketches in animation, and we continue the trend by introducing a Distance Transform Module to improve our generation quality.

3 Methodology

3.1 SoftsplatLite

As with AnimeInterp [37], we base our model on the state-of-the-art Softsplat [24] interpolation model, which uses bidirectional optical flow to differentiably forward-splat input image features for synthesis. Whereas AnimeInterp only focused on improving optical flow estimation, we assume a fixed flow estimator

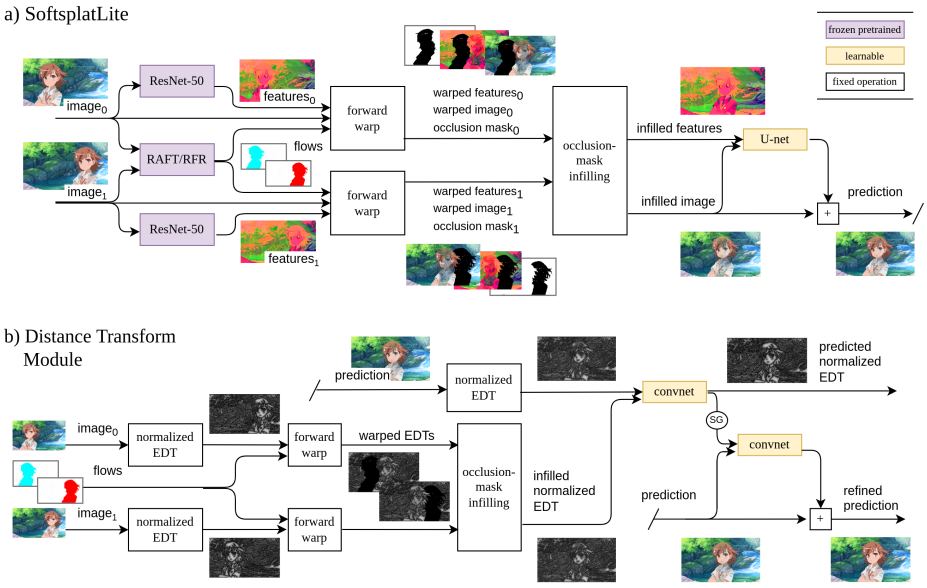

Fig. 2. Schematic of our proposed system. SoftsplatLite (SSL, Sec. 3.1) passes a prediction to the Distance Transform Module (DTM, Sec. 3.2) for refinement. SSL uses many fewer trainable parameters than AnimeInterp [37] to reduce overfitting, and introduces an infilling step to avoid ghosting artifacts. DTM leverages domain knowledge about line drawings to achieve more uniform solid-color regions. Artists: hariken, k.k.[1]

(the same RAFT [40] network from AnimeInterp, which they dub "RFR"). We instead look more closely at feature extraction, warping, and synthesis; our proposed SoftsplatLite (named similarly to PWC-Lite [19]) aims to improve convergence on LPIPS [45] while also being parameter- and training-efficient. Please see Fig. 2a for an overview of SSL.

We first note that the feature extractors in AnimeInterp [37] and Softsplat [24] are relatively shallow. The extractors must still be trained, and rely on backpropagation through the forward splatting mechanism. In practice, we found that replacing the extractor with the first four blocks of a frozen ImageNet-pretrained ResNet-50 [12] performs better; additionally, freezing the extractor contributes to reduced memory usage and compute during training, as no gradients must be backpropagated through the warping operations. Note that we also tried unfreezing the ResNet, but observed slight overfitting.

Next, we observe that forward splatting results in large empty occluded regions. If left unhandled during LPIPS training, these gaps often cause undesirable ghosting artifacts (see AnimeInterp [37] output in Fig. 3b). Additionally, subtle gradients at the edge of moving objects in the optical flow field may result

[1] hariken: https://danbooru.donmai.us/posts/5378938
k.k.: https://danbooru.donmai.us/posts/789765

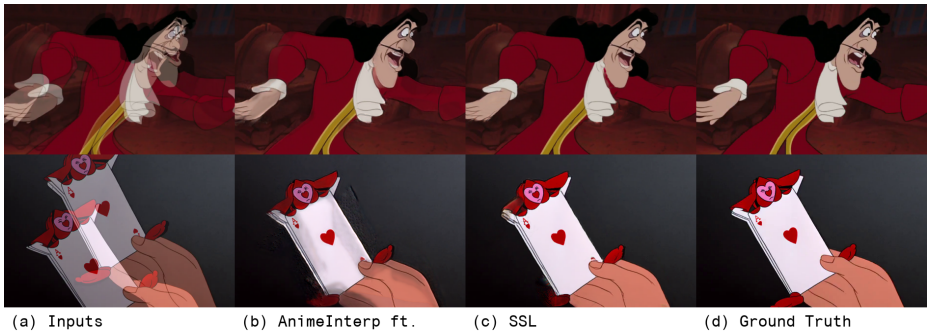

Fig. 3. SSL vs. AnimeInterp ft. [37]. Trained on the same ATD data [37] and LPIPS loss [45], AnimeInterp encounters many "ghosting" artifacts, which we resolve in SSL by proposing an inpainting technique.

in a spread of dots after forward warping; these later manifest as blurry patches in AnimeInterp predictions (see Fig. 1b). To remove these artifacts, we propose a simple infilling technique to generate a better warped feature stack F prior to synthesis ("occlusion-mask infilling" in Fig. 2a):

$$F = \frac{1}{2}\left(M_{0\to t}W_{0\to t}(f(I_0)) + (1 - M_{0\to t})W_{1\to t}(f(I_1))\right) + \frac{1}{2}\left(M_{1\to t}W_{1\to t}(f(I_1)) + (1 - M_{1\to t})W_{0\to t}(f(I_0))\right) \quad (1)$$

$$Z_{1\to 0} = -0.1 \times ||LAB(I_1) - W'_{0\to 1}(LAB(I_0))|| \quad (2)$$

where $W_{a\to b}$ denotes forward warping from timestep a to timestep b, W' denotes backwarping, M denotes the opened occlusion mask of the warp, I represents either input image, and f represents the feature extractor. In other words, occluded features are directly infilled with warped features from the other source image. The computation of mask M involves warping an image of ones, followed by a morphological image opening with kernel $k = 5$ to remove dotted artifacts; note that though opening is non-differentiable, no gradients are needed with respect to the flow field as our flow estimator is fixed. Unlike AnimeInterp [37], we do not use average forward splatting, and instead use the more accurate softmax weighting scheme with negative L_2 LAB color consistency as our Z-metric (similar as in Softsplat [24]). While it is not guaranteed that this infilling method will eliminate all holes (it is still possible for both warps to have shared occluded regions), we find that in practice the majority of image areas are covered.

Lastly, for the synthesis stage, we opt for a much more lightweight U-Net [33] instead of the GridNet [10] used in the original Softsplat [24]. We may afford this thrifty replacement by carefully placing a direct residual path from an initial warped guess to the final output. This follows the observation that directly applying our previously-described infilling method to the input RGB

Fig. 4. Effect of DTM. DTM effectively leverages line proximity cues (distance transform) to refine SSL outputs. DTM not only removes minor aberrations from solid-color regions (bottom), but also corrects entire enclosures if needed (top).

images produces a strong initial guess for the output; this is achieved by replacing feature extractor f in Eq. 1 with the identity function. Instead of requiring a large synthesizer to reconcile two sets of warped images and features into a single final image, we employ a small network to simply refine a single good guess. Under this architecture, the additional GridNet parameters become redundant, and even contribute to overfitting.

Note that while SoftsplatLite and Softsplat have comparable parameter counts at inference (6.92M and 6.21M respectively), the frozen feature extractor and smaller synthesizer significantly reduces the number of trainable parameters compared to the original (1.28M and 2.01M respectively). We later demonstrate through ablations (Tab. 2) that lighter training and artifact reduction allow SSL to score better on perceptual metrics like LPIPS and chamfer distance.

3.2 Distance Transform Module

As seen in Fig. 4b, SoftsplatLite may struggle to choose colors for certain regions, or have trouble with large areas of flat color. These difficulties may be partly attributed to the natural texture bias of convolutional models [11]; the big monotonous regions of traditional cel animation would expectedly require convolutions with larger perceptual fields to extract meaningful features. Instead of building much deeper or wider models, we take advantage of line information inherently present in 2D animation; hypothetically, providing line proximity information to convolutions may act as a form of "stand-in" texture that helps the processing of cel-colored image data.

We thus propose a Distance Transform Module (DTM) to refine the SSL outputs by leveraging a normalized version of the Euclidean distance transform (NEDT). At a high level (see Fig. 2b), DTM first attempts to predict the ground truth NEDT of the output (middle) frame, and then uses this prediction to refine the SSL output through a residual block. To train the prediction of NEDT, we introduce an auxiliary L_{dt} in addition to the L_{lpips} on the final prediction, and

Fig. 5. RRLD filtering. RRLD quantifies whether a triplet is evenly-spaced. We show several overlaid triplets from our additional dataset ranked by RRLD; higher RRLD (bottom) indicates deviation from the halfway assumption. As RRLD is fully automatic, appropriate training data can be filtered from raw video at scale.

optimize a weighted sum of both losses end-to-end. The rest of this section provides specifics on the implementation.

The first step is to extract lines from the input images; for this, we use the simple but effective difference of gaussians (DoG) edge detector,

$$DoG(I) = \frac{1}{2} + t(G_{k\sigma}(I) - G_{\sigma}(I)) - \epsilon, \tag{3}$$

where G_σ are Gaussian blurs after greyscale conversion, $k = 1.6$ is a factor greater than one, and $t = 2$ with $\epsilon = 0.01$ are hyperparameters. Please see Fig. 6 for examples of DoG extraction. Next, we apply the distance transform. To bound the range of values, we normalize EDT values to unit range similar to Narita et. al. [23],

$$NEDT(I) = 1 - \exp\{\frac{-EDT(DoG(I) > 0.5)}{\tau d}\}, \tag{4}$$

where $\tau = 15/540$ is a steepness hyperparameter, and d is the image height in pixels. Note that we thresholded DoG at 0.5 to get a binarized sketch.

This normalized EDT is extracted from both input images, and warped through the same inpainting procedure as Eq. 1; more precisely, f is replaced by $NEDT$. DTM then uses this, as well as the extracted NEDT of SSL's output, to estimate the NEDT of the ground truth output frame. This prediction occurs through a small convolutional network (first yellow box in Fig. 2b), and is trained to minimize an auxiliary L_{dt}, the L_1 Laplacian pyramid loss between predicted and ground truth NEDTs. A final convolutional network (second yellow box in Fig. 2b) then incorporates the predicted NEDT to residually refine the SSL output.

Note that we detach the predicted NEDT image from the final RGB image prediction gradients ("SG" for "stop-gradient" in Fig. 2b), in order to reduce potentially competing signals from L_{dt} and the final image loss. It is also important to mention that since both DoG sketch extraction and EDT are non-differentiable operations, the extraction of NEDT from the Softsplat output cannot be backpropagated. However, we found that we could still reasonably perform end-to-end training despite the required stop-gradient in this step.

Through this process, our DTM is able to predict the distance transform of the output, and utilize it in the final interpolation. Experiments show that this relatively cheap additional network is effective at improving perceptual performance (Tab. 2).

3.3 Restricted Relative Linear Discrepancy

Unlike in the natural video domain, where almost any three consecutive frames from a cut may be used as a training triplet, data collection for 2D animation is much more ambiguous. Animators often draw at variable framerates with expressive arc-like movements; when coupled with high pixel displacements, this results in a significant amount of triplets with non-linear motion or uneven spacing. However under the problem formulation, all middle frames of training triplets are assumed to be "halfway" between the inputs. While forward warping provides a way to control the interpolated $t \in [0, 1]$ at which generation occurs, it is ambiguous to label such ground truth for training. Li et. al. in AnimeInterp [37] manually filter through more than 130,000 triplets to arrive at their ATD dataset with 12,000 samples, a costly manual effort with less than 10% yield.

In order to automate the training data collection process from raw animation data, we quantify the deviance of a triplet from the halfway assumption with a novel Restricted Relative Linear Discrepancy (RRLD) metric, and filter samples based on a simple threshold. In our experiments (Tab. 2), we demonstrate that selecting additional training data with RRLD improves generalization error, whereas training on naively-collected triplets damages performance. We additionally show that RRLD largely agrees with ATD, and that RRLD is robust to choice of flow estimator (Sec. 4.1). Please see Fig. 5 for example triplets accepted or rejected by RRLD. The rest of this section provides specifics of the filtering method. We define RRLD as follows,

$$RRLD(\omega_{0\to t}, \omega_{1\to t}) = \frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} \frac{||\omega_{0\to t}[i,j] + \omega_{1\to t}[i,j]||/2}{||\omega_{0\to t}[i,j] - \omega_{1\to t}[i,j]||}, \tag{5}$$

where ω are forward flow fields extracted from consecutive frames I_0 and I_t and I_1, and Ω denotes the set of (i, j) pixel coordinates where both flows have norms greater than threshold 2.0 and point to pixels within the image.

RRLD takes as input flow fields from the middle frame I_t to the end frames, and assumes they are correct. The numerator of Eq. 5 represents the distance from pixel (i, j) to the midpoint between destination pixels, while the denominator describes the total distance between destination pixels. In other words,

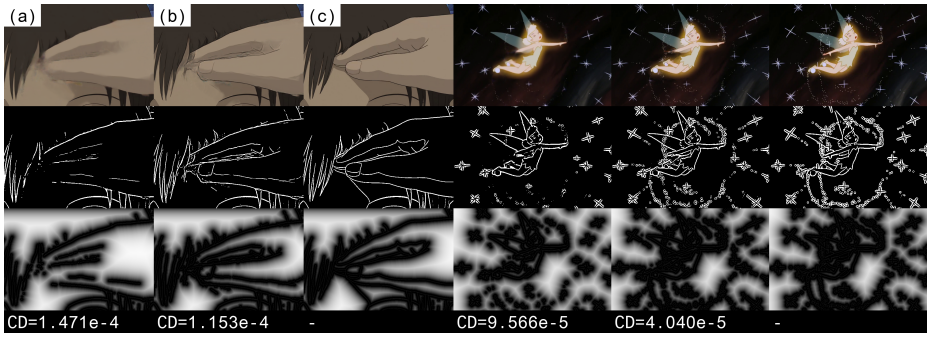

Fig. 6. Line and detail preservation. **(a)** AnimeInterp prediction; **(b)** our full model (SSL+DTM); **(c)** ground truth; **(middle)** extracted DoG lines; **(bottom)** normalized Euclidean distance transform. AnimeInterp blurs lines and details that are critical to animation; by focusing on perceptual metrics like LPIPS and chamfer distance (CD), we improve the generation quality.

the interior of the summand is half the ratio between the diameters of a parallelogram formed by two flow vectors; this measures the relative distance from the actual to the ideal halfway point. As the estimated flows are noisy, we average over a restricted set of pixels Ω. We first remove pixels with displacement close to zero, where a low denominator results in unrepresentatively high discrepancy measurement. Then, we also filter out pixels with flows pointing outside the image, which are often poor estimates. The final RRLD gives a rough measure of deviance from the halfway-frame assumption, for which we may define a cutoff (0.3 in this work).

One caveat to this method is that pans must be discarded. In some cases, a non-linear animation may be composited onto a panning background; RRLD would then include the linearly-moving background in Ω, lowering the overall measurement despite having a nonlinear region of interest. We simply remove triplets with large Ω, high average flow magnitude, and low flow variance. It is possible to reintroduce panning effects through data augmentation if needed, though we did not for our training.

Another important point is that even though animators may draw at framerates like 12 or 8, the final raw input videos are still at 24fps. Thus, many consecutive triplets in actuality contain two duplicates, which leads to RRLD values around 0.5; had the duplicate been removed, an adjacent frame outside the triplet may have had a qualifying RRLD. In order to maximize the data yield, we also train a simple duplicate frame detector, using linear regression over the mean and maximum L_2 LAB color difference between consecutive frames.

3.4 User Study & Quality Metrics

We perform a user study in order to evaluate our system and explore the relationship between metrics and perceived quality. To get a representative subset

of the ATD test set, on which we perform all evaluations, we select 323 random samples in accordance with Fischer's sample size formula (with population 2000, margin of error 5%, and confidence level 95%). For each sample triplet, users were given a pair of animations playing back and forth at 2fps, cropped to the region-of-interest annotation provided by ATD. The middle frame of each animation was a result generated either by our best model (on LPIPS), or by the pretrained AnimeInterp [37]. Participants were asked to pick which animation had: clearer/sharper lines, more consistent shapes/colors, and better overall quality. Complete survey results, including several random animation pairs compared, are available in the supplementary.

Our main metric of interest is LPIPS [45], a general measure of perceived image quality based on deep image classification features. We are interested in understanding its applicability to non-photorealistic domains like ours, especially in comparison with PSNR/SSIM used in prior work [37].

We additionally consider the chamfer distance (CD) between lines extracted from the ground truth vs. the prediction. The chamfer metric is typically used in 3D work, where the distance between two point clouds is calculated by averaging the shortest distances from each point of one cloud to a point on the other. In the context of binary line drawings extracted from our data using DoG (Eq. 3), the 3D points are replaced by all 2D pixels that lie on lines. As chamfer distance would intuitively measure how far lines are from each other in different images, we explore the importance of this metric for our domain with images based on line drawings. Please see Fig. 6 for examples of CD evaluation. In this work, we define chamfer distance as:

$$CD(X_0, X_1) = \frac{1}{2HWD} \sum X_0 DT(X_1) + X_1 DT(X_0) \tag{6}$$

where X are binary sketches with 1 on lines and 0 elsewhere, DT denotes the Euclidean distance transform, the summation is pixel-wise, and HWD is the product of height, width, and diameter. We normalize by both area and diameter to enforce invariance to image scale. Note that our definition is symmetric with respect to prediction and ground truth, zero if and only if they are equal, and strictly non-negative. Also observe that as neither DoG binarization nor DT is differentiable, CD cannot be optimized directly by gradient descent training; thus it is used for evaluation only.

4 Experiments & Discussion

We implement our system in PyTorch [29] wrapped in Lightning [8], with Kornia [32]. Our model uses the same RFR/RAFT with SGM flows as AnimeInterp for fairer comparison [37, 40], and forward splatting is done with the official Softsplat [24] module. We train with the Adam [17] optimizer at learning rate $\alpha = 0.001$ for 50 epochs, and accumulate gradients for an effective batch size of 32. Our code uses the official LPIPS [45] package, with the AlexNet [18] backbone. All training minimizes the total loss $L = \lambda_{lpips} L_{lpips} + \lambda_{dt} L_{dt}$, where $\lambda_{lpips} = 30$; depending

Table 1. Comparison with baselines. Our full proposed method achieves the best perceptual performance, followed by AnimeInterp [37]. We show in our user study (Sec. 4.4) that LPIPS/CD are better indicators of quality than the PSNR/SSIM focused on in previous work; we list them here for completeness. Models from prior work are fine-tuned on LPIPS for fairer comparison. Best values are underlined, runner-ups italicized; LPIPS is scaled by 1e2, CD by 1e5.

	All				Eastern		Western	
Model	LPIPS	CD	PSNR	SSIM	LPIPS	CD	LPIPS	CD
DAIN [1]	4.695	5.288	28.840	95.28	5.499	6.537	4.204	4.524
DAIN ft. [1]	4.137	4.851	29.040	95.27	4.734	5.888	3.771	4.217
RIFE [13]	4.451	5.488	28.515	95.14	4.933	6.618	4.156	4.796
RIFE ft. [13]	4.233	5.411	27.977	93.70	4.788	6.643	3.894	4.658
ABME [28]	5.731	7.244	29.177	*95.54*	7.000	10.010	4.955	5.552
ABME ft. [28]	4.208	4.981	29.060	95.19	4.987	6.092	3.732	4.302
AnimeInterp [37]	5.059	5.564	<u>29.675</u>	<u>95.84</u>	5.824	7.017	4.590	4.674
AnimeInterp ft. [37]	*3.757*	*4.513*	28.962	95.02	*4.113*	*5.286*	*3.540*	*4.039*
Ours	<u>3.494</u>	<u>4.350</u>	*29.293*	95.15	<u>3.826</u>	<u>4.979</u>	<u>3.291</u>	<u>3.966</u>

on whether DTM is trained, λ_{dt} is either 0 or 5. Evaluations are run over the 2000-sample test set from AnimeInterp's ATD12k dataset; however we only train on a random 9k of the remaining 10k in ATD, so that we can designate 1k for validation. Similar to Li et. al. [37], we randomly perform horizontal flips and frame order reversal augmentations during training. We use single-node training with at most 4x GTX1080Ti at a time, with mixed precision where possible. All models are trained and tested at 540x960 resolution.

We wrote a custom CUDA implementation for the distance transform and chamfer distance using CuPy [27] that achieves upwards of 3000x speedup from the SciPy CPU implementation [41]; the algorithm is a simpler version of Felzenszwalb et. al. [9], where we calculate the minimum of the lower envelope through brute iteration. While more efficient GPU algorithms are known [4], we found our implementation sufficient.

4.1 RRLD Data Collection

As RRLD was designed to replicate the manual selection of training data, we applied RRLD to AnimeInterp's ATD dataset [37] and achieved 95.3% recall (i.e. RRLD only rejected less than 5% of human-collected data); as the negative samples from the ATD collection process are not available, it is not possible to calculate RRLD's precision on ATD. Additionally we study the effect of flow estimation on RRLD, finding that filtering with FlowNet2 [14] and RFR flows [37] returns very similar results (0.877 Cohen's kappa tested over 34,128 triplets).

We use our automatic pipeline to collect additional training triplets. We source data from 14 franchises in the eastern "anime" style, with premiere dates ranging from 1989-2020, totalling 239 episodes (roughly 95hrs, 8.24M frames at 24fps); please refer to our supplementary materials for the full list of sources.

Table 2. Ablations of proposed methods. Firstly, each component of SSL contributes to performance (especially infilling). Secondly, new data filtered naively hurts performance, while new RRLD-filtered data helps. Lastly, DTM improvement is due to auxiliary supervision, not just increased parameter count. AnimeInterp ft. is copied from Tab. 1 for comparison; the last row here and in Tab. 1 are equivalent. Best values are underlined, runner-ups italicized; LPIPS is scaled by 1e2, CD by 1e5.

		All		Eastern		Western	
Model	Data	LPIPS	CD	LPIPS	CD	LPIPS	CD
AnimeInterp ft. [37]	ATD	3.757	4.513	4.113	5.286	3.540	4.039
SSL (no flow infill)	ATD	3.648	4.496	4.026	5.160	3.416	4.089
SSL (no U-net synth.)	ATD	3.614	4.579	3.982	5.288	3.389	4.146
SSL (no ResNet extr.)	ATD	3.605	4.739	3.957	5.429	3.391	4.317
SSL	ATD	3.586	4.572	3.940	5.248	3.369	4.158
SSL	ATD+naive	3.702	4.811	3.997	5.033	3.521	4.675
SSL	ATD+RRLD	3.535	4.431	3.873	5.089	3.329	*4.028*
SSL+DTM (no L_{dt})	ATD+RRLD	*3.531*	*4.430*	*3.865*	*4.995*	*3.327*	4.085
SSL+DTM	ATD+RRLD	<u>3.494</u>	<u>4.350</u>	<u>3.826</u>	<u>4.979</u>	<u>3.291</u>	<u>3.966</u>

Table 3. User study results. For each of the visual criteria we asked the users to judge (rows), we list the percentage of instances where users preferred the animation with a better metric score (columns). Values above 50% indicate agreement between queried criteria and metric score difference, and values under 50% indicate contradiction. "Pref. Ours" means percent of users preferring our output to AnimeInterp [37] for that criteria.

Criteria	Prefer Ours	Lower LPIPS	Lower CD	Higher PSNR	Higher SSIM
cleaner/sharper lines	86.01%	86.56%	78.20%	18.95%	15.48%
more consistent shape/color	78.82%	79.26%	73.99%	25.02%	22.66%
better overall quality	81.11%	81.55%	75.67%	22.97%	19.88%

Here, RRLD was calculated using FlowNet2 [14] as inference was faster than RFR [37]. While RRLD filtering presents us with 543.6k viable triplets, we only select one random triplet per cut to promote diversity; the cut detection was performed with a pretrained TransNet v2 [38]. This cuts down eligible samples to 49.7k. For the demonstrative purposes of this paper, we do not train on the full new dataset, and instead limit ourselves to doubling the ATD training set by randomly selecting 9k qualifying triplets. Please see Fig. 5 for examples of accepted and rejected triplets from franchises set aside for validation.

While we cannot release the new data collected in this work, our specific sources are listed in the supplementary and our RRLD data collection pipeline will be made public; this allows followup work to either recreate our dataset or assemble their own datasets directly from source animations.

4.2 Comparison with Baselines

The main focus of our work is to improve perceptual quality, namely LPIPS and chamfer distance (as validated later by our user study results). We gather four existing frame interpolation systems (ABME [28], RIFE [13], DAIN [1], and AnimeInterp [37]) for comparison to our full model incorporating all our proposed methods. For a fairer comparison, as other models may not have been trained on the same LPIPS objective or on animation data, we fine-tune their given pre-trained models with LPIPS on the ATD training set. As we can see from Tab. 1, our full proposed method achieves the best perceptual performance, followed by AnimeInterp. To provide more complete information on trainable parameters, our model has 1.28M (million) compared to: AnimeInterp 2.01M, RIFE 13.0M, ABME 17.5M, DAIN 24.0M. Breaking down further, our model consists of 1.266M for SSL and 0.011M for DTM.

4.3 Ablation Studies

We perform several ablations in Tab. 2. In the first group, each of the modifications to Softsplat [24] (frozen ResNet [12] feature extractor, infilling, U-net [33] replacing GridNet [10]) contributes to SSL outperforming AnimeInterp [37]. The infilling technique improves performance the most.

In the second group of Tab. 2, we ablate the addition of new data filtered by RRLD (Sec. 4.1). Training with RRLD-filtered data improves generalization as expected. To demonstrate the necessity of RRLD's specific filtering strategy, we train with an alternative dataset of equal size gathered from the same sources, but using a "naive" filtering approach. For simplicity, we directly follow the crude filter used in creating ATD [37]: no two frames of a triplet may contain SSIM outside [0.75, 0.95]. We see this naively-collected data actively damages model performance, validating the use of our proposed RRLD filter.

Splitting by eastern vs. western style, we clarify the distribution shift between sub-domains. Note that our new data is all anime, whereas 62.05% of ATD test set is in the western "Disney" style. From the LPIPS results, the eastern style is more difficult; adding eastern-only RRLD data has unexpectedly less of an effect on eastern testing than western. This may be because western productions tend to prioritize fluid motion (smaller displacements) over complex character designs (more details), contrary to the eastern style.

In the last group of Tab. 2, we train SoftsplatLite with DTM, but ablate the effect of additionally optimizing for L_{dt}; this way, we may see whether auxiliary supervision of NEDT improves performance under the same parameter count. Note that the upper yellow convnet of Fig. 2b receives no gradients in the ablation, effectively remaining at its random initialization. The results show that the prediction of line proximity information indeed contributes to performance.

4.4 User Study Results

We summarize the user study results in Tab. 3, and provide the full breakdown with sample animations in the supplementary. Our study had 5 participants,

meaning each entry of Tab. 3 has support 1615 (323 compared pairs per participant). We confirm the observations made by Niklaus et. al. and Blau et. al. [3], that PSNR/SSIM and perceptual metrics may be at odds with one another. Despite lower PSNR/SSIM scores, users consistently preferred our outputs to those of AnimeInterp. A possible explanation is that due to animations having larger displacements, the middle ground truth frames may be quite displaced from the ideal halfway interpolation. SSIM, as noted by previous work [45, 34], was not designed to assess these geometric distortions. Color metrics like PSNR and L_1 may penalize heavily for this perceptually minor difference, encouraging the model to reduce risk by blurring; this is consistent with behavior exhibited by the original AnimeInterp trained on L_1 (Fig. 6). LPIPS on the other hand has a larger perceptive field due to convolutions, and may be more forgiving of these instances. This study provides another example of the perception-distortion tradeoff [3], and establishes its transferability to 2D animation.

The user study also shows an imperfect match between LPIPS and CD. This mismatch is also reflected in Tables 1 and 2, where aggregate decreases in LPIPS do not correspond to reduced CD. This maybe because CD reflects only the line-structures of an image. However, Tab 3 shows LPIPS is unexpectedly more predictive of line quality. A possible explanation is that CD is still more sensitive to offsets than LPIPS; in fact, CD grows roughly proportionally to displacement for line drawings. Thus, it may suffer the same problems as PSNR but to a lesser extent, as PSNR would penalize across an entire displaced area opposed to across a thin line.

5 Limitations & Conclusion

Our system still has several limitations. By design, our model can only interpolate linearly between two frames, while real animations have non-linear movements that follow arcs across long sequences. In future work, we may incorporate non-linearity from methods like QVI [43], or allow user input from an artist. Additionally, we are limited to colored frames, which are typically unavailable until the later stages of animation production; following related work [23], we can expand our scope to work on line drawings directly.

To summarize, we identify and overcome shortcomings of previous work [37] on 2D animation interpolation, and achieve state-of-the-art interpolation perceptual quality. Our contributions include an effective SoftsplatLite architecture modified to improve perceptual performance, a Distance Transform Module leveraging domain knowledge of lines to perform refinement, and a Restricted Relative Linear Discrepancy metric that allows automatic training data collection from raw animation. We validate our focus on perceptual quality through a user study, hopefully inspiring future work to maintain this emphasis for the traditional 2D animation domain.

Acknowledgements The authors would like to thank Lillian Huang and Saeed Hadadan for their discussion and feedback, as well as NVIDIA for GPU support.

References

1. Bao, W., Lai, W.S., Ma, C., Zhang, X., Gao, Z., Yang, M.H.: Depth-aware video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3703–3712 (2019)
2. Bao, W., Lai, W.S., Zhang, X., Gao, Z., Yang, M.H.: Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. IEEE transactions on pattern analysis and machine intelligence (2019)
3. Blau, Y., Michaeli, T.: The perception-distortion tradeoff. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6228–6237 (2018)
4. Cao, T.T., Tang, K., Mohamed, A., Tan, T.S.: Parallel banding algorithm to compute exact distance transform with the gpu. In: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games. pp. 83–90 (2010)
5. Casey, E., Pérez, V., Li, Z.: The animation transformer: Visual correspondence via segment matching. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 11323–11332 (2021)
6. Choi, M., Kim, H., Han, B., Xu, N., Lee, K.M.: Channel attention is all you need for video frame interpolation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 10663–10671 (2020)
7. Dalstein, B., Ronfard, R., Van De Panne, M.: Vector graphics animation with time-varying topology. ACM Transactions on Graphics (TOG) **34**(4), 1–12 (2015)
8. Falcon, W., The PyTorch Lightning team: PyTorch Lightning (3 2019). https://doi.org/10.5281/zenodo.3828935, https://github.com/PyTorchLightning/pytorch-lightning
9. Felzenszwalb, P.F., Huttenlocher, D.P.: Distance transforms of sampled functions. Theory of computing **8**(1), 415–428 (2012)
10. Fourure, D., Emonet, R., Fromont, E., Muselet, D., Tremeau, A., Wolf, C.: Residual conv-deconv grid network for semantic segmentation. arXiv preprint arXiv:1707.07958 (2017)
11. Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv:1811.12231 (2018)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
13. Huang, Z., Zhang, T., Heng, W., Shi, B., Zhou, S.: Rife: Real-time intermediate flow estimation for video frame interpolation. arXiv preprint arXiv:2011.06294 (2020)
14. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2462–2470 (2017)
15. Jaegle, A., Borgeaud, S., Alayrac, J.B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., et al.: Perceiver io: A general architecture for structured inputs & outputs. arXiv preprint arXiv:2107.14795 (2021)
16. Jiang, H., Sun, D., Jampani, V., Yang, M.H., Learned-Miller, E., Kautz, J.: Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9000–9008 (2018)

17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25**, 1097–1105 (2012)
19. Liu, L., Zhang, J., He, R., Liu, Y., Wang, Y., Tai, Y., Luo, D., Wang, C., Li, J., Huang, F.: Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6489–6498 (2020)
20. Maejima, A., Kubo, H., Shinagawa, S., Funatomi, T., Yotsukura, T., Nakamura, S., Mukaigawa, Y.: Anime character colorization using few-shot learning. In: SIGGRAPH Asia 2021 Technical Communications, pp. 1–4 (2021)
21. Meyer, S., Djelouah, A., McWilliams, B., Sorkine-Hornung, A., Gross, M., Schroers, C.: Phasenet for video frame interpolation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 498–507 (2018)
22. Meyer, S., Wang, O., Zimmer, H., Grosse, M., Sorkine-Hornung, A.: Phase-based frame interpolation for video. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1410–1418 (2015)
23. Narita, R., Hirakawa, K., Aizawa, K.: Optical flow based line drawing frame interpolation using distance transform to support inbetweenings. In: 2019 IEEE International Conference on Image Processing (ICIP). pp. 4200–4204. IEEE (2019)
24. Niklaus, S., Liu, F.: Softmax splatting for video frame interpolation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5437–5446 (2020)
25. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive convolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 670–679 (2017)
26. Niklaus, S., Mai, L., Liu, F.: Video frame interpolation via adaptive separable convolution. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 261–270 (2017)
27. Okuta, R., Unno, Y., Nishino, D., Hido, S., Loomis, C.: Cupy: A numpy-compatible library for nvidia gpu calculations. In: Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS) (2017)
28. Park, J., Lee, C., Kim, C.S.: Asymmetric bilateral motion estimation for video frame interpolation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14539–14548 (2021)
29. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32**, 8026–8037 (2019)
30. Qian, Z., Bo, W., Wei, W., Hai, L., Hui, L.J.: Line art correlation matching network for automatic animation colorization. arXiv e-prints pp. arXiv–2004 (2020)
31. Ren, H., Li, J., Gao, N.: Two-stage sketch colorization with color parsing. IEEE Access **8**, 44599–44610 (2019)
32. Riba, E., Mishkin, D., Shi, J., Ponsa, D., Moreno-Noguer, F., Bradski, G.: A survey on kornia: an open source differentiable computer vision library for pytorch (2020)
33. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)

34. Sampat, M.P., Wang, Z., Gupta, S., Bovik, A.C., Markey, M.K.: Complex wavelet structural similarity: A new image similarity index. IEEE transactions on image processing **18**(11), 2385–2401 (2009)
35. Simo-Serra, E., Iizuka, S., Ishikawa, H.: Mastering sketching: adversarial augmentation for structured prediction. ACM Transactions on Graphics (TOG) **37**(1), 1–13 (2018)
36. Simo-Serra, E., Iizuka, S., Sasaki, K., Ishikawa, H.: Learning to simplify: fully convolutional networks for rough sketch cleanup. ACM Transactions on Graphics (TOG) **35**(4), 1–11 (2016)
37. Siyao, L., Zhao, S., Yu, W., Sun, W., Metaxas, D., Loy, C.C., Liu, Z.: Deep animation video interpolation in the wild. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6587–6595 (2021)
38. Souček, T., Lokoč, J.: Transnet v2: An effective deep network architecture for fast shot transition detection. arXiv preprint arXiv:2008.04838 (2020)
39. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8934–8943 (2018)
40. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: European conference on computer vision. pp. 402–419. Springer (2020)
41. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2
42. Whited, B., Noris, G., Simmons, M., Sumner, R.W., Gross, M., Rossignac, J.: Betweenit: An interactive tool for tight inbetweening. In: Computer Graphics Forum. vol. 29, pp. 605–614. Wiley Online Library (2010)
43. Xu, X., Siyao, L., Sun, W., Yin, Q., Yang, M.H.: Quadratic video interpolation. arXiv preprint arXiv:1911.00627 (2019)
44. Yagi, Y.: A filter based approach for inbetweening. arXiv preprint arXiv:1706.03497 (2017)
45. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)