ABSTRACT

Title of Dissertation: MACHINE LEARNING FOR ANIME: ILLUSTRATION, ANIMATION, AND 3D CHARACTERS

Shuhong Chen
Doctor of Philosophy, 2024

Dissertation Directed by: Professor Matthias Zwicker
Department of Computer Science

As anime-style content becomes more popular on the global stage, we ask whether new vision/graphics techniques could contribute to the art form. However, the highly-expressive and non-photorealistic nature of anime poses additional challenges not addressed by standard ML models, and much of the existing work in the domain does not align with real artist workflows. In this dissertation defense, we will present several works building foundational 2D/3D infrastructure for ML in anime (including pose estimation, video frame interpolation, and 3D character reconstruction) as well as an interactive tool leveraging novel techniques to assist 2D animators.

MACHINE LEARNING FOR ANIME: ILLUSTRATION, ANIMATION, AND 3D CHARACTERS

by

Shuhong Chen

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2024

Preface

Anime is getting more popular in the global entertainment market. However, traditional animation is laborious. To create the expressive motions loved by millions, professional and amateur animators alike face the intrinsic cost of 12 illustrations per second. As the medium rapidly enters mainstream, the sheer manual line-mileage demanded continues to increase. This begs the question of whether modern data-driven computer vision/graphics methods can offer automation or assist the creative process. While some work exists for colorization, cleanup, in-betweening, etc., we're still missing foundational domain-specific infrastructure. In addition, much of the academic work around problems like in-betweening have only been studied in-vitro, without practical considerations for real animators. By studying industry practices, scaling data pipelines, bridging domain gaps, leveraging 3d priors, etc., we developed domain-specific ML infrastructure for anime, and demonstrated ways that modern techniques can assist existing workflows.

In addition, while 3d human priors are crucial for the above animation topic (form and surface anatomy are animator fundamentals), 3d character modeling itself may also benefit from new techniques. As AR/VR apps and virtual creators become more popular, there will soon be major demand for stylized 3d avatars. But current template-based designers are restrictive, with custom assets still requiring expert software to create. Novel representations and rendering techniques may help create realistic 3d humans, but comparatively little has been done to suit the design challenges of non-photorealistic characters. This work also tries to democratize 3d character creation, bringing customizable experiences to the next generation of social interaction.

Dedication

To Nagato Yuki.

Table of Contents

Preface ii

Dedication iii

Table of Contents iv

Chapter 1: Transfer Learning for Pose Estimation of Illustrated Characters 1
- 1.1 Introduction 1
- 1.2 Related Work 3
- 1.3 Method & Architectures 6
- 1.4 Data Collection 10
- 1.5 Experiments 13
- 1.6 Application: Pose-guided Retrieval 18
- 1.7 Conclusion & Future Work 19

Chapter 2: Improving the Perceptual Quality of 2D Animation Interpolation 21
- 2.1 Introduction 21
- 2.2 Related Work 24
- 2.3 Methodology 26
- 2.4 Experiments & Discussion 36
- 2.5 Limitations & Conclusion 41

Chapter 3: Match-free Inbetweening Assistant (MIBA): A Practical Animation Tool without User Stroke Correspondence 42
- 3.1 Introduction 43
- 3.2 Related Work 47
- 3.3 Methodology 49
  - 3.3.1 Input/Output Representations 49
  - 3.3.2 Match-free Inbetweening 50
  - 3.3.3 User Interaction 54
- 3.4 User Study Evaluation 55
  - 3.4.1 Data 55
  - 3.4.2 Participants 58
  - 3.4.3 Procedure 59
  - 3.4.4 Metrics 60
  - 3.4.5 Analysis 61

3.5 Limitations and Future Work 62
3.6 Conclusions 63

Chapter 4: Stylized Single-view 3D Reconstruction from Portraits of Anime Characters 67
4.1 Introduction & Related Work 67
4.2 Methodology 71
4.3 Data 74
4.4 Results & Evaluation 78
4.5 Limitations & Future Work 82

Chapter 1: Transfer Learning for Pose Estimation of Illustrated Characters

Human pose information is a critical component in many downstream image processing tasks, such as activity recognition and motion tracking. Likewise, a pose estimator for the illustrated character domain would provide a valuable prior for assistive content creation tasks, such as reference pose retrieval and automatic character animation. But while modern data-driven techniques have substantially improved pose estimation performance on natural images, little work has been done for illustrations. In our work, we bridge this domain gap by efficiently transfer-learning from both domain-specific and task-specific source models. Additionally, we upgrade and expand an existing illustrated pose estimation dataset, and introduce two new datasets for classification and segmentation subtasks. We then apply the resultant state-of-the-art character pose estimator to solve the novel task of pose-guided illustration retrieval. All data, models, and code will be made publicly available.

1.1 Introduction

Human pose estimation is a foundational computer vision task with many real-world applications, such as activity recognition [81], 3D reconstruction [47], motion tracking [102], virtual try-on [30], person re-identification [78], etc. The generic formulation is to find, in a given image containing people, the positions and orientations of body parts; typically, this means locating landmark and joint keypoints on 2D images, or regressing for bone transformations in 3D.

The usefulness of pose estimation is not limited to the natural image domain; in particular, we focus on the domain of illustrated characters. As pose-guided motion retargeting of realistic humans rapidly advances [37], there is growing potential for automatic pose-guided animation [43], a traditionally labor-intensive task for both 2D and 3D artists. Pose information may also serve as a valuable prior in illustration colorization [137], keyframe interpolation [106], 3D character reconstruction [13] and rigging [128], etc.

With deep computer vision, we have been able to leverage large-scale datasets [70, 4, 116] to train robust estimators of human pose [45, 17, 33]. However, little work has been done to solve pose estimation for illustrated characters. Previous pose estimation work on illustrations by Khungurn et al [56] presented a 2D keypoint detector, but relied on a publicly-unavailable synthetic dataset and an ImageNet-trained backbone. In addition, the dataset they collected for supervision lacked variation, and was missing keypoints and bounding boxes required for evaluation under the more modern COCO standard [70].

Facing these challenges, we constructed a 2D keypoint detector with state-of-the-art performance on illustrated characters, built upon domain-specific components and efficient transfer learning architectures. We demonstrate the effectiveness of our methods by implementing a novel illustration retrieval system. Summarizing, we contribute:

- A state-of-the-art pose estimator for illustrated characters, transfer-learned from both domain-specific and task-specific source models. Despite the absence of synthetic supervision, we outperform previous work by 10-20% PDJ@20 [56].

- An application of our proposed pose estimator to solve the novel task of pose-guided character illustration retrieval.

- Datasets for our model and its components, including: an updated COCO-compliant version of Khungurn et al's [56] pose dataset with 2x the number of samples and more diverse poses; a novel 1062-class Danbooru [5] tagging rulebook; and a character segmentation dataset 20x larger than those currently available.

1.2 Related Work

The Illustration Domain Though there has been work on caricatures and cartoons [15, 91], we focus on anime/manga-style drawings where characters tend to be less abstract. While there is work for more traditional problems like lineart cleaning [103] and sketch extraction [66], more recent studies include sketch colorization [137], illustration segmentation [135], painting relighting [136], image-to-image translation with photos [58], and keyframe interpolation [106].

Available models for illustrated tasks typically rely on small manually-collected datasets. For example, the AniSeg [68] character segmenter is trained on less than $1{,}000$ examples. While larger datasets are becoming available (e.g. Danbooru [5] now with 4.2m tagged illustrations), the labels are noisy and long-tailed, leading to poor model performance [6, 60]. Works requiring pose information may use synthetic renders of anime-style 3D models [56, 43], but the models are usually not publicly available. In this work, we present a cleaner tag classification task, a large character segmentation dataset, and an upgraded COCO keypoint dataset; these will all be made available upon publication, and may serve as a valuable prior for other tasks.

Transfer Learning & Domain Adaptation Transfer learning and domain adaptation have been defined somewhat inconsistently throughout the vision and natural language processing literature [119, 27], though generally the former is considered broader than the latter. In this paper, we use the terms interchangeably, referring to methods that leverage information from a number of related source domains and tasks, to a specific target domain and task. Typically, much more data is available for the source than the target, motivating us to transfer useful related source knowledge in the absence of sufficient target data [119]. For deep networks, the simplest practice is to pretrain a model on source data, and fine-tune its parameters on target data; however, various techniques have been studied that work with different levels of target data availability.

Much of the transfer learning work in vision focuses on extreme cases with significantly limited target domain data, with emphasis around the task of image classification. In the few-shot learning case, we may be given as few as ten (or even one) samples from the target, inviting methods that embed prototypical target data into a space learned through prior source knowledge [121]. In particular, it is common to align parameters of feature extractors across domains, by directly minimizing pairwise feature distances or by adversarial domain discrimination [74, 115]. If the source and target are similar enough, it is possible to perform domain adaptation in the complete absence of labeled target data. This can be achieved by matching statistical properties of extracted features [109], or by converting inputs between domains through cycle-consistent image translation [46].

Pose Estimation With the availability of large-scale human pose datasets [70, 4], the vision community has recently been able to make great strides in pose estimation. A naive baseline was demonstrated by Mask R-CNN [45], which extended their detection and segmentation framework to predict single-pixel masks of joint locations. Other work such as RMPE take an approach tailored to pose estimation, deploying spatial transformer networks with pose-guided NMS and region proposal [33]. Around the same time, OpenPose proposed part affinity fields as a bottom-up alternative to the more common heatmap representation of joints [17]. Human pose estimation work continues to make headway, extending beyond keypoint localization to include dense body part labels [42] and 3D pose estimation [51, 64, 80].

Pose Estimation Transfer Most transfer learning for pose estimation adapts from synthetically-rendered data to natural images. For example, by using mocaps and 3D human models, SURREAL [116] provides 6 million frames of synthetic video, complete with a variety of datatypes (2D/3D pose, RGB, depth, optical flow, body parts, etc.). CNNs may be able to directly generalize pose from synthesized images [116], and can further close the domain gap using other priors like motion [28]. Outside of synthetic-to-real, Cao et al [65] explore domain adaptation for quadruped animal pose estimation, achieving generalization from human pose through adversarial domain discrimination with pseudo-label training.

The closest prior work to our topic was done by Khungurn et al [56], who collected a modest AnimeDrawingsDataset (ADD) of 2k character illustrations with joint keypoints, and a larger synthetic dataset of 1 million frames rendered from MikuMikuDance (MMD) 3D models and mocaps. Unfortunately, the MMD dataset is not publicly available, and ADD contains mostly standard forward-facing poses. In addition, ADD is missing bounding boxes and several face keypoints, which are necessary for evaluation under the modern COCO standard [70]. We remedy these issues by training a bounding box detector from our new character segmentation dataset, labeling missing annotations in ADD, and labeling 2k additional samples in more varied poses.

Khungurn et al perform transfer from an ImageNet-pretrained GoogLeNet backbone [111] and synthetic MMD data. In the absence of MMD, we instead transfer from a stronger backbone trained on a new illustration-specific classification task, as well as from a task-specific model pretrained on COCO keypoints. We use our subtask models and data to implement a number of transfer techniques, from naive fine-tuning to adversarial domain discrimination. In doing so, we significantly outperform Khungurn et al on their reported metrics by 10-20%.

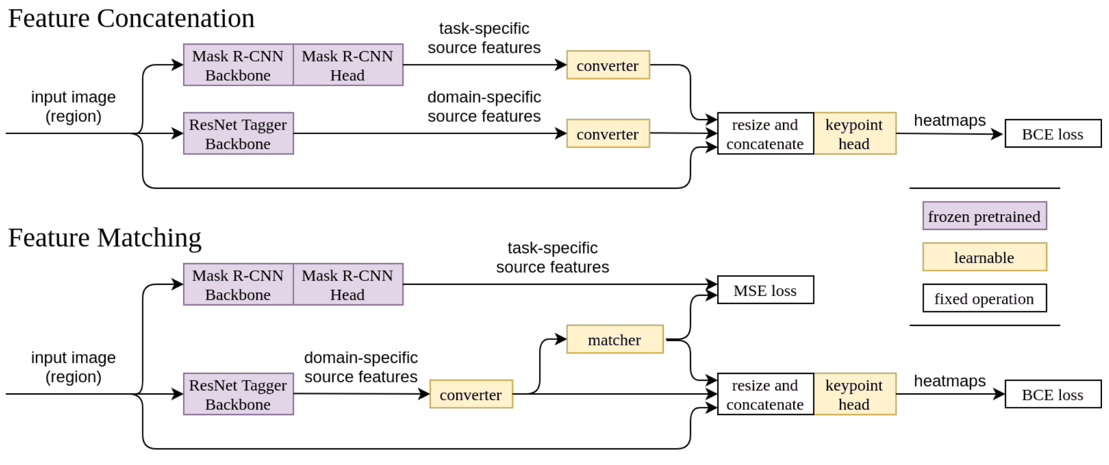


Figure 1.1: A schematic outlining our two transfer learning architectures: feature concatenation, and feature matching. Note that source feature specificity is with respect to the target; i.e. task-specific means "related to pose estimation" and domain-specific means "related to illustrations". Feature converters and matchers are convolutional networks that learn to mimic or re-appropriate pretrained features, respectively. While both designs require the pretrained Mask R-CNN components during training, feature matching discards them during inference, instead relying on the trained matcher network. "BCE" refers to binary cross-entropy loss.

1.3 Method & Architectures

We provide motivation and architecture details for two variants of our proposed pose estimator (feature concatenation and feature matching), as well as two submodules critical for their success (a class-balanced tagger backbone and a character segmentation model). Architectures for baseline comparison models are described in Sec. 1.5.

Pose Estimation Transfer Model We present two versions of our final model: feature concatenation, and feature matching. In this section, we assume that region proposals are given by a separate segmentation model (Sec. 1.3), and that the domain-specific backbone is already available (Sec. 1.3); here, we focus on combining source features to predict keypoints (Fig. 4.1).

The goal is to perform transfer simultaneously from both a domain-specific classification backbone (Sec 1.3) and a task-specific keypoint model (Mask R-CNN [45]). Here, we chose Mask R-CNN as it showed significantly better out-of-the-box generalization to illustrations than OpenPose [17] (Tab. 1.1). Taking into account that the task-specific model already achieves mediocre performance on the target domain, the feature concatenation model simply stacks features from both sources (Fig. 4.1). In order to perform the concatenation, it learns shallow feature converters for each source to decrease the feature channel count and allow bilinear sampling to a common higher resolution. The combined features are fed to the head, consisting of a shallow converter and two ResNet blocks.

The final output is a stack of 25 heatmaps, 17 for COCO keypoints and 8 for auxiliary appendage midpoints (following Khungurn et al [56]). We apply pixel-wise binary cross-entropy loss on each heatmap, targeting a normal distribution centered on the ground-truth keypoint location with standard deviation proportional to the keypoint's COCO OKS sigma [70]; the sigmas for auxiliary midpoints are averaged from endpoints of the body part. At inference, we gaussian-smooth the heatmaps and take the maximum pixel value index as the keypoint prediction.

Although feature concatenation produces the best results (Tab. 1.1), it is very inefficient. At inference, it must maintain the parameters of both source models, and run both forward models for each prediction; Mask R-CNN is particularly expensive in this regard. We thus also provide a feature matching model, inspired by the methods used in Luo et al [74]. As shown in Fig. 4.1, we simultaneously train an additional matching network that predicts features from the expensive task-specific model using features from the domain-specific model. Though matching may be optimized with self-supervision signals such as contrastive loss [126], we found that featurewise mean-squared error is suitable. Given the matcher, the pretrained Mask R-CNN still helps training, but is not necessary at inference. Despite its simplicity, feature matching retains most performance benefits from both source models, while also being significantly lighter and faster than the concatenation architecture.

ResNet Tagger The domain-specific backbone for our model (Fig. 4.1) is a pretrained ResNet50 [**ResNet**] fine-tuned as an illustration tagger. The tagging task is equivalent to multi-label classification, in this case predicting the labels applied to an image by the Danbooru image-board moderators [5]. The 392k unique tags cover topics including colors, clothing, interactions, composition, and even copyright metainfo.

Khungurn et al [56] use an ImageNet-trained GoogLeNet [111] backbone for their illustrated pose estimator, but we find that Danbooru fine-tuning significantly boosts transfer performance. There are publicly-available Danbooru taggers [6, 60], but both their classification performance and feature learning capabilities are hindered by uninformative target tags and severe class imbalance. By alleviating these issues, we achieve significantly better transfer to pose estimation.

Most available Danbooru taggers [6, 60] take a coarse approach to defining classes, simply predicting the several thousand (6-7k) most frequent tags. However, many of these tags represent contextual information not present in the image; e.g. neon_genesis_evangelion (name of a franchise), or alternate_costume (fanmade/non-canon clothes). We instead only allow tags explicitly describing the image (clothing, body parts, etc.). Selecting tags by frequency also introduces tag redundancy and annotator disagreement. There are many high-frequency tags that share similar concepts, but are annotated inconsistently; e.g. hand_in_hair, adjusting_hair, and hair_tucking have vague wiki definitions for taggers, and many color tags are subjective (aqua_hair vs. blue_hair). To address these challenges, we survey Danbooru wikis to manually develop a rulebook of tag groups that defines more explicit and less redundant classes.

Danbooru tag frequencies form a long-tailed distribution, posing a severe class imbalance problem. In addition to filtering out under-tagged images (detailed in Sec. 1.4), we implement an inverse square-root frequency reweighing scheme to emphasize the learning of less-frequent classes. More formally, the loss on a sample is:

$$\mathcal{L}(y, \hat{y}) = \frac{1}{C} \sum_{i=0}^{C-1} w_i(y_i) BCE(y_i, \hat{y}_i) \tag{1.1}$$

$$w_i(z) = \frac{1}{2} \left(\frac{z}{r_i} + \frac{1-z}{1-r_i} \right) \tag{1.2}$$

$$r_i = \frac{\sqrt{N_i}}{\sqrt{N_i} + \sqrt{N - N_i}} \tag{1.3}$$

where C is the number of classes, $\hat{y} \in [0,1]^C$ is the prediction, $y \in \{0,1\}^C$ is the ground truth label, BCE is binary cross entropy loss, N is the total number of samples, and N_i is the number of positive samples in the i^{th} class. We found that plain inverse frequency weighing caused numerical instability in training, necessitating the square root.

Character Segmentation & Bounding Boxes In order to produce bounding boxes around each subject in the image, we first train an illustrated character segmenter. As we assume one subject per image, we can derive a bounding box by enclosing the thresholded segmentation output. The single-subject assumption also removes the need for region proposal and NMS infrastructure present in available illustrated segmenters [68], so that our model may focus on producing clean segmentations only. Our segmentation model is based on DeepLabv3 [22], with three additional layers at the end of the head for finer segmentations at the input image resolution. We initialize with pretrained DeepLabv3 weights from PyTorch [89], and fine-tune the full model using pixel-wise binary cross-entropy loss.

Model	OKS@50	OKS@75	PCKh@50	PDJ@20	PCPm@50	params	ms/img
Feature Concatenation (+new data)	**0.8982**	**0.7930**	**0.7866**	0.8403	0.8551	86.8m	217.7
Feature Concatenation	0.8827	0.7723	0.7762	0.8282	0.8435	86.8m	217.7
Feature Matching (+new data)	0.8953	0.7907	0.7851	**0.8423**	**0.8599**	9.9m	147.8
Feature Matching	0.8769	0.7680	0.7675	0.8251	0.8343	9.9m	147.8
Task Fine-tuning Only	0.8026	0.6481	0.7032	0.7666	0.7446	77.5m	174.5
Domain Features Only	**0.8607**	**0.7467**	0.7444	0.8076	**0.8215**	9.6m	143.7
Task Fine-tuning w/ Domain Features	0.8548	0.7209	**0.7544**	**0.8181**	0.8084	41.1m	147.8
Adversarial (DeepFashion2)	0.8321	0.6804	0.7108	0.7823	0.7778	9.9m	147.8
Adversarial (COCO)	0.8065	0.6362	0.6788	0.7607	0.7350	9.9m	147.8
Task-Pretrained (R-CNN)	**0.7584**	**0.6724**	**0.6960**	**0.7357**	**0.6679**	77.5m	174.5
Task-Pretrained (OpenPose)	0.4922	0.4222	0.4447	0.4796	0.4381	52.3m	128.2
Ours (equiv. to feat. concat.)	**0.8827**	**0.7723**	**0.7762**	**0.8282**	**0.8435**	86.8m	217.7
RF5 Backbone	0.8547	0.7358	0.7427	0.8015	0.8005	86.8m	217.7
ImageNet-pretrained Backbone	0.8218	0.6919	0.7060	0.7649	0.7571	86.8m	217.7

Table 1.1: Performance of different architectures and ablations described in Sec. 1.5. Note that the parameter count and speed are measured in inference mode with batch size one; “m” refers to “millions of parameters”.

1.4 Data Collection

Unless mentioned otherwise, we train with random image rotation, translation, scaling, flipping, and recoloring.

Pose Data We extend the AnimeDrawingsDataset (ADD), first collected by Khungurn et al [56]. The original dataset had 2000 illustrated full-body single-character images from Danbooru, each annotated with joint keypoints. However, ADD did not follow the now popularized COCO standard [70]; in particular, it was missing facial keypoints (eyes and ears) and bounding boxes. In order to evaluate and compare with modern pose estimators, we manually labeled the missing keypoints using an open-source COCO annotator [12] and automatically generated bounding boxes using the character segmenter described in Sec. 1.3. We also manually remove 57 images with multiple characters, or without the full body in view.

In addition, we improve the diversity of poses in ADD by collecting an additional 2043 samples. A major weakness of ADD is its lack of backwards-facing characters; only 5.45% of the entire 2k dataset had a back-related Danbooru tag (e.g. back, from_behind, looking_back, etc.). We specifically filtered for back-related images when annotating, resulting in a total of 850 in the updated dataset (21.25%). We also selected for other notably under-represented poses, like difficult leg tags (soles, bent_over, leg_up, crossed_legs, squatting, kneeling, etc.), arm tags (stretch, arms_up, hands_clasped, etc.), and lying tags (on_side, on_stomach).

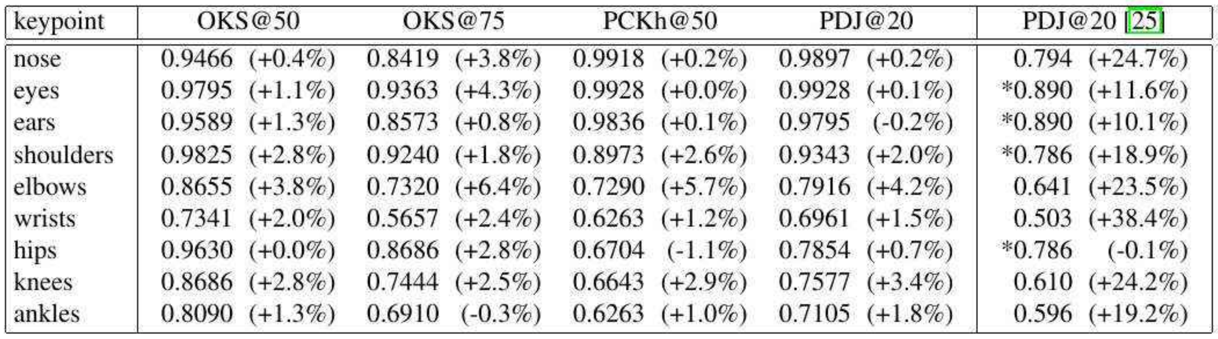
Our final updated dataset contains 4000 illustrated character images with all 17 COCO keypoints and bounding boxes. We designate 3200 images for training (previously 1373), 313 for validation (previously 97), and 487 for testing (same as original ADD). For each input image, we first scale and crop such that the bounding box is centered and padded by at least 10% of the edge length on all sides. We then perform augmentations; flips require swapping left-right keypoints, and full 360-degree rotations are allowed.

ResNet Tagger Data Our ResNet50 tagger is trained on a new subset of the 512px SFW Danbooru2019 dataset [5]. The original dataset contains 2.83m images with over 390k tags, but after filtering and retagging we arrive at 837k images with 1062 classes. The new classes are derived from manually-selected union rules over 2027 raw tags, as described in Sec. 1.3; the rulebook has 314 body-part, 545 clothing, and 203 miscellaneous (e.g. image composition) classes.

To combat the class imbalance problem described in Sec. 1.3, we also rigorously filtered the dataset. We remove all images that are not single-person (solo, 1girl, or 1boy), are comics (comic, 4koma, doujinshi, etc.), or are smaller than 512px. Most critically, we remove all images with less than 12 positive tags; these images are very likely under-tagged, and would have introduced many false-negatives to the ground truth. The final subset of 837k images has significantly reduced class imbalance (median class frequency 0.38%, minimum 0.04%) compared to the datasets of available taggers (median 0.07%, min 0.01%) [6].

We split the dataset 80-10-10 train-val-test. As some tags are color-sensitive, we do not jitter the hue; similarly as some tags are orientation-sensitive, we allow up to 15-degree rotations and horizontal flips only.

Character Segmentation Data To obtain character bounding boxes, we train a character segmentation model and enclose output regions at 0.5 threshold (Sec. 1.3). The inputs to our segmentation system are augmented composites of RGBA foregrounds (with transparent backgrounds) onto RGB backgrounds; the synthetic ground truth is the foreground alpha. The available AniSeg dataset [68] has only 945 images, with manually-labeled segmentations that are not pixel-perfectly aligned. We thus collect our own larger synthetic compositing dataset. Our background images are a mix of illustrated scenery (5.8k Danbooru images with scenery and no_humans tag) and stock textures (2.3k scraped [2] from the Pixiv Dataset [67]). We collect single-character foreground images from Danbooru with the transparent_background tag; 18.5k samples are used, after filtering images with text, non-transparency, or more than one connected component in the alpha channel. Counting each foreground as a single sample, this makes our new dataset roughly 20x larger than AniSeg. The foregrounds and backgrounds are randomly paired for compositing during training, with 5% chance of having no foreground. We hold out 2048 deterministic foreground-background pairs for validation and testing (1024 each).



keypoint	OKS@50	OKS@75	PCKh@50	PDJ@20	PDJ@20 [25]
nose	0.9466 (+0.4%)	0.8419 (+3.8%)	0.9918 (+0.2%)	0.9897 (+0.2%)	0.794 (+24.7%)
eyes	0.9795 (+1.1%)	0.9363 (+4.3%)	0.9928 (+0.0%)	0.9928 (+0.1%)	*0.890 (+11.6%)
ears	0.9589 (+1.3%)	0.8573 (+0.8%)	0.9836 (+0.1%)	0.9795 (-0.2%)	*0.890 (+10.1%)
shoulders	0.9825 (+2.8%)	0.9240 (+1.8%)	0.8973 (+2.6%)	0.9343 (+2.0%)	*0.786 (+18.9%)
elbows	0.8655 (+3.8%)	0.7320 (+6.4%)	0.7290 (+5.7%)	0.7916 (+4.2%)	0.641 (+23.5%)
wrists	0.7341 (+2.0%)	0.5657 (+2.4%)	0.6263 (+1.2%)	0.6961 (+1.5%)	0.503 (+38.4%)
hips	0.9630 (+0.0%)	0.8686 (+2.8%)	0.6704 (-1.1%)	0.7854 (+0.7%)	*0.786 (-0.1%)
knees	0.8686 (+2.8%)	0.7444 (+2.5%)	0.6643 (+2.9%)	0.7577 (+3.4%)	0.610 (+24.2%)
ankles	0.8090 (+1.3%)	0.6910 (-0.3%)	0.6263 (+1.0%)	0.7105 (+1.8%)	0.596 (+19.2%)

Table 1.2: Keypoint breakdown of our most performant "feature concatenation" model trained on our extended ADD dataset. In the center, we list the relative improvement of each metric when training on additional data. On the right, we display the PDJ@20 from Khungurn et al [56], and report the relative difference from our best model. *Note that due to keypoint incompatibilities, we fill missing keypoint results from [56] using the most similar keypoints reported: "head" for eyes and ears, and "body" for shoulders and hips.

Model	F-1	pre.	rec.	IoU
Ours	**0.9472**	**0.9427**	**0.9576**	**0.9326**
YAAS SOLOv2	0.9061	0.9003	0.9379	0.9077
YAAS CondInst	0.8866	0.8824	0.8999	0.9158
AniSeg	0.5857	0.5877	0.5954	0.6651

Table 1.3: Comparison of our character segmentation and bounding box performance, described in Sec. 1.5.

1.5 Experiments

We used PyTorch [89] wrapped in Lightning [3]; some models use the R101-FPN keypoint detection R-CNN from Detectron2 [125]. All models can be trained with a single GTX1080ti (11GB VRAM). Unless otherwise mentioned, we trained models using the Adam [61] optimizer, with 0.001 learning rate and batch size 32, for 1,000 epochs.

The ResNet backbone is trained on the Danbooru tag classification task using our new manual tagging rulebook (Sec. 1.4). The character segmenter used for bounding boxes is trained with our new character segmentation dataset (Sec. 1.4). Using the previous two submodules, we train the pose estimator using our upgraded version of the ADD dataset (Sec. 1.4). All data and code will be released upon publication.

Pose Estimation Transfer Table 1.1 shows the performance of different architectures. We report COCO OKS [70], PCKh and PCPm [4], and PDJ (for comparison with Khungurn et al [56]). From the top four rows, we see that our proposed feature concatenation and matching models perform the best out overall, and that the addition of our new data increases performance. We also observe that while concatenation performs marginally better than matching, matching is 8.8x more parameter efficient and one-third faster at inference.

The second group of Table 1.1 shows other architectures, roughly in order of method complexity. Here, as in Fig. 4.1, "task" source features refer to Mask R-CNN pose estimation features, and "domain" source features refer to illustration features extracted by our ResNet50 tag classifier.

"Task Fine-tuning Only" fine-tunes the pretrained Mask R-CNN head with its frozen default backbone; the last head layer is re-initialized to accommodate auxiliary appendage keypoints. This is vanilla transfer by fine-tuning a task-specific source network on a small task-specific target domain dataset.

"Domain Features Only" is our frozen ResNet50 backbone with a keypoint head. This is vanilla transfer by adding a new task head to a domain-specific source network.

"Task Fine-tuning w/ Domain Features" fine-tunes the pretrained Mask R-CNN head as above, but replaces the R-CNN backbone with our frozen ResNet50 backbone. This is a naive method of incorporating both sources, attempting to adapt the task source's pretrained prediction component to new domain features.

"Adversarial (DeepFashion2)" reuses the feature matching architecture, but performs ad-

versarial domain discrimination instead of MSE matching. The discriminator is a shallow 2-layer convnet, trained to separate Mask R-CNN features of randomly sampled DeepFashion2 [38] images from ResNet features of Danbooru illustrations. As the feature maps to discriminate are spatial, we are careful to employ only 1x1 kernels in the discriminator; otherwise, the discriminator could pick up intrinsic anatomical differences. The matching network now fools the discriminator by adversarially aligning the feature distributions.

"Adversarial (COCO)" is the same adversarial architecture as above, but using COCO [70] images containing people instead of Deepfashion2.

While domain-features-only is the cheapest architecture overall, it is only slightly more efficient than feature matching, and loses all benefits of task-specific transfer. However, the performance drop from feature concatenation to domain-features-only and task-with-domain-features is not very large (2-3% OKS@50); meanwhile, there is a wide gap to task-fine-tuning-only. This shows that the domain-specific ResNet50 backbone trained on our new body-tag rulebook provides much more predictive power than the task-specific pretrained Mask R-CNN.

It is important to note that the adversarial models exhibited significant instability during training. After extensive hyperparameter tuning, the best DeepFashion2 model returns NaN loss at epoch 795, and the best COCO model fails at epoch 354; all other models safely exited at epoch $1{,}000$. DeepFashion2 likely outperforms COCO because the image composition is much more similar to that of Danbooru; images are typically single-person portraits with most of the body in view. Adversarial losses are notoriously difficult to optimize, and in our case destabilized training so as to perform worse than not having been used at all.

The fourth group of Table 1.1 shows out-of-the-box generalization to illustrations for Mask R-CNN [45] and OpenPose [17]. We use Mask R-CNN as our task-specific source, as it is less-

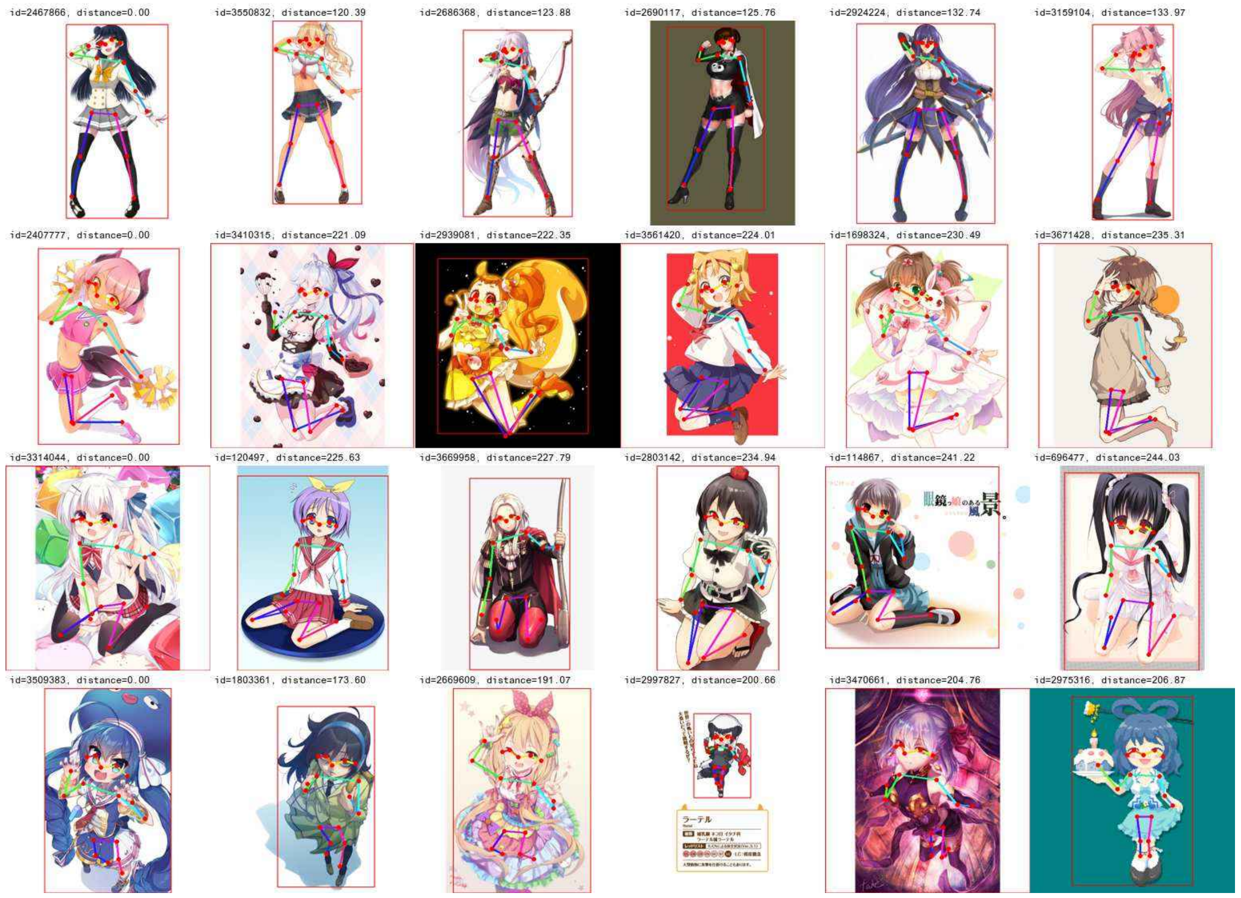

id=2467866, distance=0.00
id=3550832, distance=120.39
id=2686368, distance=123.88
id=2690117, distance=125.76
id=2924224, distance=132.74
id=3159104, distance=133.97
id=2407777, distance=0.00
id=3410315, distance=221.09
id=2939081, distance=222.35
id=3561420, distance=224.01
id=1698324, distance=230.49
id=3671428, distance=235.31
id=3314044, distance=0.00
id=120497, distance=225.63
id=3669958, distance=227.79
id=2803142, distance=234.94
id=114867, distance=241.22
id=696477, distance=244.03
id=3509383, distance=0.00
id=1803361, distance=173.60
id=2669609, distance=191.07
id=2997827, distance=200.66
id=3470661, distance=204.76
id=2975316, distance=206.87

Figure 1.2: Pose-based retrieval. From left to right, we show the query image (descriptor distance zero) followed by its five nearest neighbors (duplicate and NSFW images removed). Each illustration is annotated with its Danbooru ID, descriptor distance to the query, and the predicted bounding box with COCO keypoints. Please see supplementary materials for full artist attribution and additional examples.

overfit to natural images than OpenPose.

Table 1.2 gives a keypoint breakdown and comparison with Khungurn et al [56]. The results demonstrate that training on our additional more varied data improves the overall model performance; this is especially true for appendage keypoints, which are more variable than the head and torso. We also see significant improvement from results reported in Khungurn et al. The exception is the hips, for which we compare to their "body" keypoint at the navel. While this is not a direct comparison, our PDJ on hips is nevertheless low relative to other keypoints. This is because PDJ does not account for the intrinsic ambiguity of the hips; looking at the OKS, which accounts for annotator disagreement, we see that hip performance is actually quite high.

An important caveat is that the metrics are generally not comparable with those reported in human pose estimation. COCO OKS, for example, was designed using annotator disagreement on natural images [70]; however, illustrated character proportions deviate widely from the standard human form (i.e. bigger head and eyes). Characters also tend to take up more screen space proportional to body size (i.e. big hair and clothing), leading to looser thresholds normalized by bounding box size.

ResNet Tagger Backbone We train our ResNet50 tagger backbone to produce illustration-specific source features (Fig. 4.1). Taking into account the class imbalance, we accumulate gradients for an effective batch size of 512. Considering the minimum (0.04%) and median (0.38%) class frequencies, we may expect the smallest class to appear 0.2 times per batch, and the median class to appear 1.9 times per batch.

To demonstrate the effectiveness of our tag rulebook and class reweighing strategy, we report performance on pose estimation using two other ResNet50 backbones: the RF5 tagger [6], and the default ImageNet-pretrained ResNet50 from PyTorch [89]. While there are several Danbooru taggers available [6, 60], we chose to compare our backbone to the RF5 tagger [6] because it is the most architecturally similar to our ResNet50, and relatively better-documented. The backbones all share the same architecture and parameter count, and are all placed into our feature concatenation transfer model for the ablation.

The backbone ablation results are shown in the last three rows of Table 1.1. As expected, a classifier trained with our novel body-part-specific tagging rulebook and class-balancing techniques significantly improves transfer to pose estimation. Note that our tagger also outperforms

RF5 at classification (on shared target classes); please refer to the supplementary materials for more details.

Character Segmentation & Bounding Boxes We compare the segmentation and bounding box performance of our system with that of publicly-available models. AniSeg [68] is a Faster-RCNN [95], and YAAS [141] provides SOLOv2 [120] and CondInst [114] models. These detectors may detect more than one character, and their bounding boxes are not necessarily tight around segmentations; for simplicity, we union all predicted segmentations of an image, and redraw a tight bounding box around the union. We evaluate all models on the same test set described in Sec. 1.4. Table 1.3 shows that training with our new 20x larger dataset outperforms available models in both mean F-1 (segmentation) and IoU (bounding boxes); we thus use it in our pipeline for bounding box prediction.

1.6 Application: Pose-guided Retrieval

An immediate application of our illustrated pose estimator is a pose-guided character retrieval system. We construct a proof-of-concept retriever that takes a query character (or user-specified keypoints and bounding box) and searches for illustrated characters in a similar pose. This system can serve as a useful search tool for artists, who often use reference drawings while illustrating.

Our pose retriever performs a simple nearest-neighbor search. The support images consist of single-character Danbooru illustrations with the full_body tag. Using our best-performing model, we extract bounding boxes and keypoint locations for each character, normalize the keypoints by the longest bounding box dimension, and finally store the pairwise euclidean distances between the normalized keypoints. This process ensures the pairwise-distance descriptor is invariant to translation, rotation, and image scale. At inference, we extract the descriptor from the query, and find the euclidean k-nearest neighbors from the support set.

In practice, we compute descriptors using all 25 predicted keypoints (17 COCO and 8 additional appendage midpoints). This makes the descriptor 300-dimensional (25 choose 2), which is generally too large for tree-based nearest neighbors [14]. However, since our support set consists of 136k points, we are still able to brute force search in reasonable time. Empirically, each query takes about 0.1341s for keypoint extraction (GPU) and 0.0638s for search (CPU).

To demonstrate the effectiveness of our pose estimator, we present several query results in Fig. 1.2; while there is no ground-truth to measure quantitative performance, qualitative inspection suggests that our model works well. We can retrieve reasonably similar illustrations for standard poses as shown in the first row, as well as more difficult poses for which illustrators would want references. Note that while our system has no awareness of perspective, it is able to effectively leverage keypoint cues to retrieve similarly foreshortened views in the last row. For more examples, please refer to our supplementary materials.

1.7 Conclusion & Future Work

While we may continue to improve the transfer performance through methods like pseudo-labeling [65] or cycle-consistent image translation [46], we can also begin extending our work to multi-character detection and pose estimation. While it is possible to construct a naive instance-based segmentation and keypoint estimation dataset by compositing background-removed ADD samples, we cannot expect a system trained on such data to perform well in-the-wild. Character interactions in illustrations are often much more complex than human interactions in real life, with much more frequent physical contact. For example, Danbooru has 43.6k images tagged with holding_hands and 59.1k with hugging, already accounting for 2.8% of the entire dataset. Simply compositing independent characters together would not be able to model the intricacies of the illustration domain; we would again need to expand our datasets with annotated instances of character interactions.

As a fundamental vision task, pose estimation also provides a valuable prior for numerous other novel applications in the illustrated domain. Our pose estimator opens the door to pose-guided retargeting for automatic character animation, better keyframe interpolation, pose-aware illustration colorization, 3D character reconstruction, etc.

In conclusion, we demonstrate state-of-the-art pose estimation on the illustrated character domain, by leveraging both domain-specific and task-specific source models. Our model significantly outperforms prior art [56] despite the absence of synthetic supervision, thanks to successful transfer from our new illustration tagging subtask focused on classifying body-related tags. In addition, we provide a single-region proposer trained on a novel character segmentation dataset 20x larger than those currently available, as well as an updated illustration pose estimation dataset with twice the number of samples in more diverse poses. Our model performance allows for the novel task of pose-guided character illustration retrieval, and paves the way for future applications in the illustrated domain.

Chapter 2: Improving the Perceptual Quality of 2D Animation Interpolation

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.

2.1 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 [106]) approached some of these challenges by improving the optical flow estimation component of a deep video interpolation system by Niklaus et. al. (Softsplat [83]); 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 [138] 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.

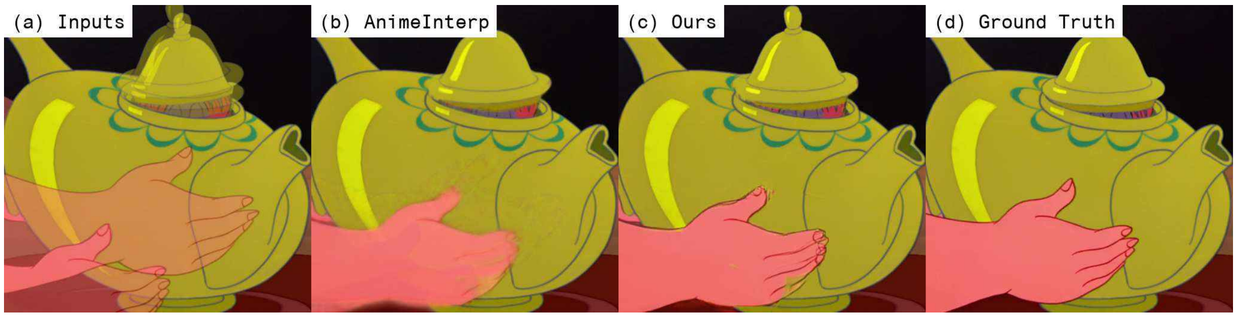


Figure 2.1: We improve the perceptual quality of 2D animation interpolation from previous work. **(a)** Overlaid input images to interpolate; **(b)** AnimeInterp by Li et. al. [106]; **(c)** Our proposed method; **(d)** Ground truth interpolation. Note the destruction of lines in (b) compared to (c), and the patchy artifacts ghosted on the teapot in (b). Our user study validates our focus on perceptual metrics and artifact removal.

2.2 Related Work

Much recent work has been published on photorealistic video interpolation. Broadly, these works fall into phase-based [77, 76], kernel-based [85, 84], and flow-based methods [83, 53, 88, 127], with others using a mix of techniques [7, 8, 25]. The most recent state-of-the-art has seen more flow-based methods [83, 88], following corresponding advancements in optical flow estimation [50, 110, 52, 112]. Flow-based methods can be further split by forward [83], or backward [88] warping. The prior art most directly related to ours is AnimeInterp, by Li et. al. [106]. 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 [105, 104] is a popular topic with applications to speeding up animation "tie-downs" and "cleanups". There are systems for syn-

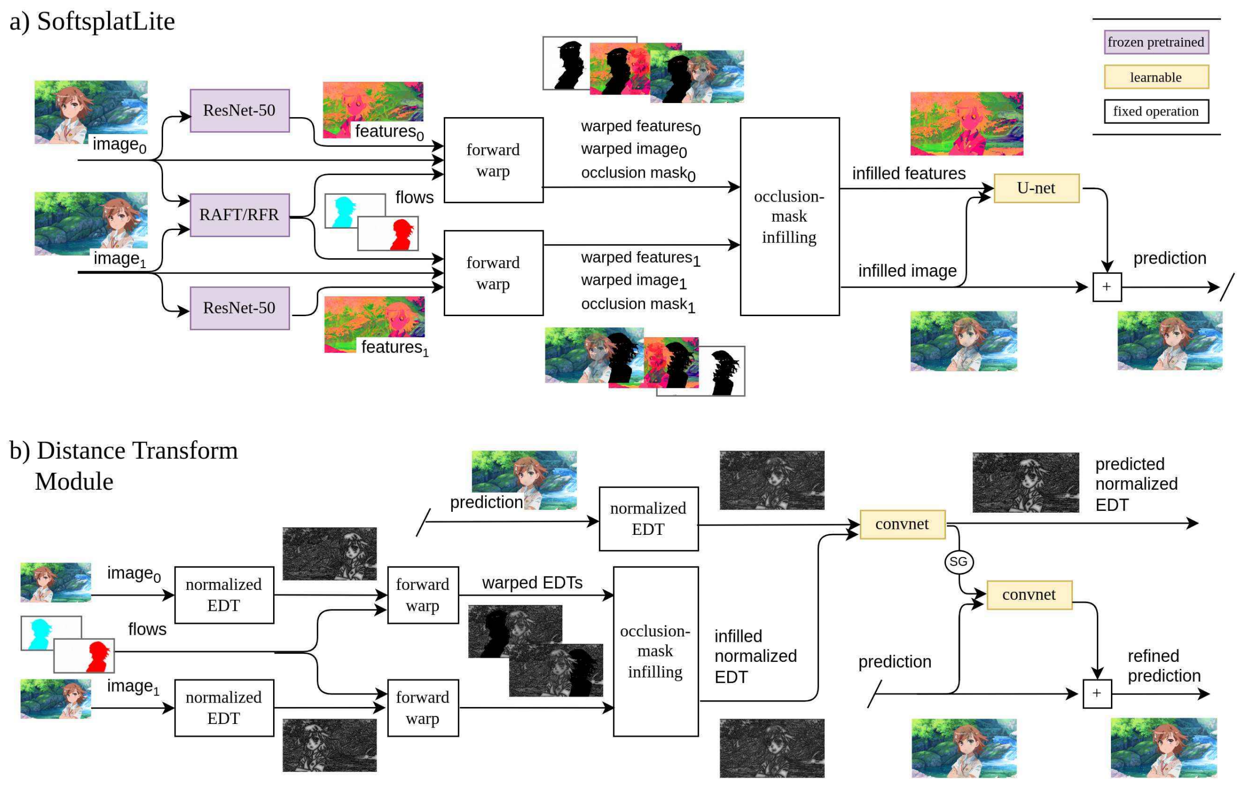


Figure 2.2: Schematic of our proposed system. SoftsplatLite (SSL, Sec. 2.3) passes a prediction to the Distance Transform Module (DTM, Sec. 2.3) for refinement. SSL uses many fewer trainable parameters than AnimeInterp [106] to reduce overfitting, and introduces an infilling step to avoid ghosting artifacts. DTM leverages domain knowledge about line drawings to achieve more uniform solid-color regions. Artists: hariken, k.k.[1]

thesizing "in-between" line drawings from sketch keyframes in both raster [82, 130] and vector [123, 26] form. While the flow-based in-betweening done by Narita et. al. [82] 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 [137] and animations [92, 75, 19]. 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.

2.3 Methodology

SoftsplatLite

As with AnimeInterp [106], we base our model on the state-of-the-art Softsplat [83] 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 (the same RAFT [112] 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 [72]) aims to improve convergence on LPIPS [138] while also being parameter- and training-efficient. Please see Fig. 4.1a for an overview of SSL.

We first note that the feature extractors in AnimeInterp [106] and Softsplat [83] 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 [44] 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 [106] output in Fig. 2.3b). Additionally, subtle gradients at the edge of moving objects in the optical flow field may result in a spread of dots after forward warping; these later manifest

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

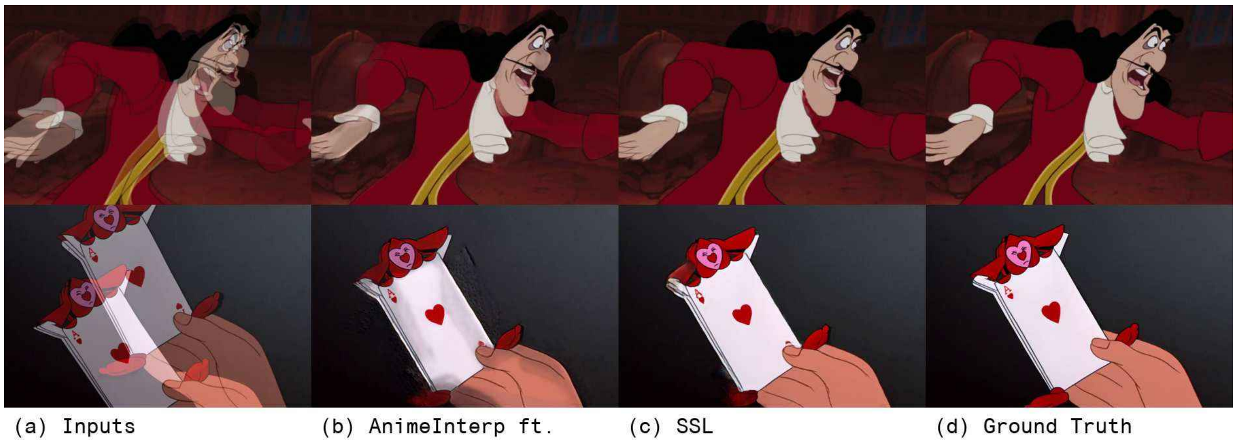


Figure 2.3: SSL vs. AnimeInterp ft. [106]. Trained on the same ATD data [106] and LPIPS loss [138], AnimeInterp encounters many “ghosting” artifacts, which we resolve in SSL by proposing an inpainting technique.

as blurry patches in AnimeInterp predictions (see Fig. 2.1b). 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. 4.1a):

$$\begin{aligned} F = \frac{1}{2}\left(M_{0\to t}W_{0\to t}(f(I_0)) + (1 - M_{0\to t})W_{1\to t}(f(I_1))\right) \\ + \frac{1}{2}\left(M_{1\to t}W_{1\to t}(f(I_1)) + (1 - M_{1\to t})W_{0\to t}(f(I_0))\right) \end{aligned} \tag{2.1}$$

$$Z_{1\to 0} = -0.1 \times ||LAB(I_1) - W'_{0\to 1}(LAB(I_0))|| \tag{2.2}$$

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 [106], 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 [83]). 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 [98] instead of the GridNet [36] used in the original Softsplat [83]. 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 images produces a strong initial guess for the output; this is achieved by replacing feature extractor f in Eq. 2.1 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. 2.2) that lighter training and artifact reduction allow SSL to score better on perceptual metrics like LPIPS and chamfer distance.

Distance Transform Module

As seen in Fig. 2.4b, 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 tex-

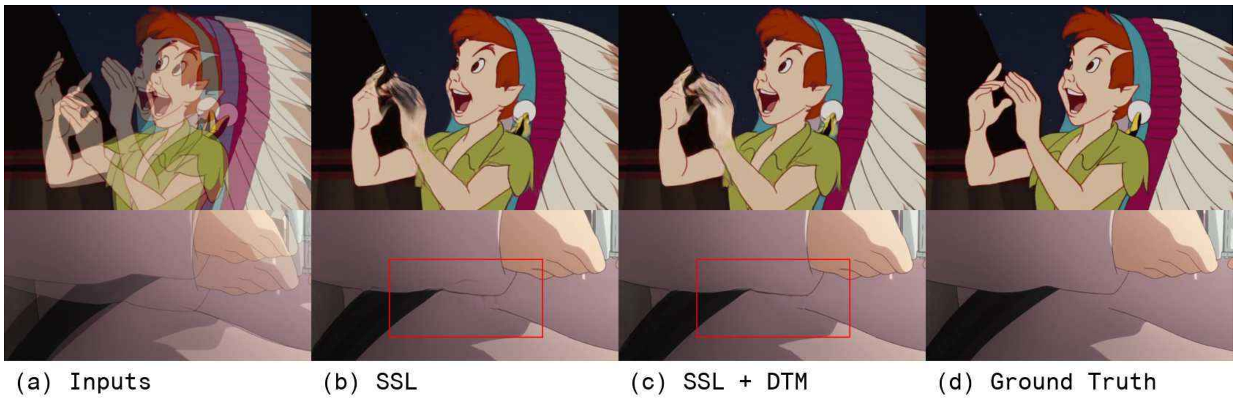


Figure 2.4: Effect of DTM. DTM effectively leverages line proximity cues (distance transform) to refine SSL outputs. DTM not only removes minor aberrations from solid-color regions (bottom), but also corrects entire enclosures if needed (top).

ture bias of convolutional models [39]; 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. 4.1b), 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 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

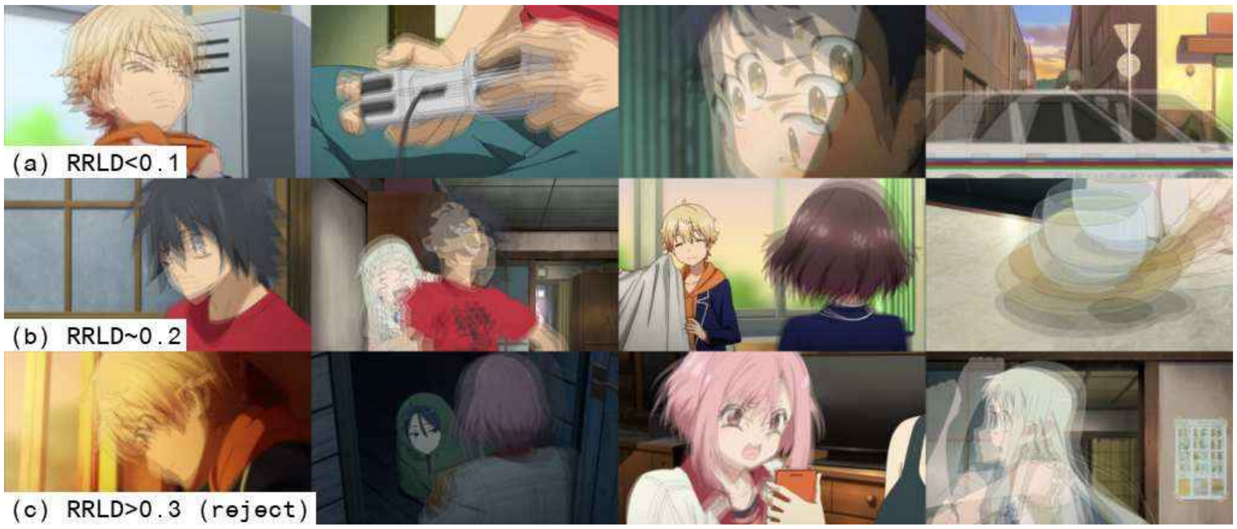


Figure 2.5: RRLD filtering. RRLD quantifies whether a triplet is evenly-spaced. We show several overlaid triplets from our additional dataset ranked by RRLD; higher RRLD (bottom) indicates deviation from the halfway assumption. As RRLD is fully automatic, appropriate training data can be filtered from raw video at scale.

effective difference of gaussians (DoG) edge detector,

$$DoG(I) = \frac{1}{2} + t(G_{k\sigma}(I) - G_{\sigma}(I)) - \epsilon, \tag{2.3}$$

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. 2.6 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. [82],

$$NEDT(I) = 1 - \exp\{\frac{-EDT(DoG(I) > 0.5)}{\tau d}\}, \tag{2.4}$$

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. 2.1; 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. 4.1b), 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. 4.1b) 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. 4.1b), 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. 2.2).

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 [106] 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. 2.2), 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. 4.3). Please see Fig. 2.5 for example triplets accepted or rejected by RRLD. The rest of this section provides specifics of the filtering method. We define RRLD as follows,

$$RRLD(\omega_{0\to t}, \omega_{1\to t}) = \frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} \frac{||\omega_{0\to t}[i,j] + \omega_{1\to t}[i,j]||/2}{||\omega_{0\to t}[i,j] - \omega_{1\to t}[i,j]||}, \tag{2.5}$$

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. 2.5 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, 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

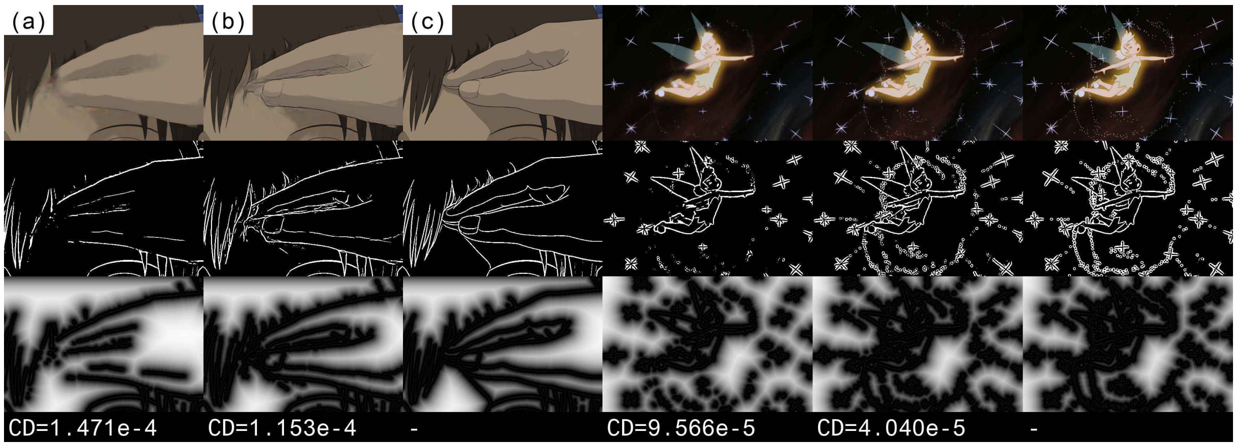


Figure 2.6: Line and detail preservation. **(a)** AnimeInterp prediction; **(b)** our full model (SSL+DTM); **(c)** ground truth; **(middle)** extracted DoG lines; **(bottom)** normalized Euclidean distance transform. AnimeInterp blurs lines and details that are critical to animation; by focusing on perceptual metrics like LPIPS and chamfer distance (CD), we improve the generation quality.

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.

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 [106]. 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 [138], 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 [106].

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

Table 2.1: Comparison with baselines. Our full proposed method achieves the best perceptual performance, followed by AnimeInterp [106]. We show in our user study (Sec. 2.4) 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 [7]	4.695	5.288	28.840	95.28	5.499	6.537	4.204	4.524
DAIN ft. [7]	4.137	4.851	29.040	95.27	4.734	5.888	3.771	4.217
RIFE [48]	4.451	5.488	28.515	95.14	4.933	6.618	4.156	4.796
RIFE ft. [48]	4.233	5.411	27.977	93.70	4.788	6.643	3.894	4.658
ABME [88]	5.731	7.244	29.177	*95.54*	7.000	10.010	4.955	5.552
ABME ft. [88]	4.208	4.981	29.060	95.19	4.987	6.092	3.732	4.302
AnimeInterp [106]	5.059	5.564	<u>29.675</u>	<u>95.84</u>	5.824	7.017	4.590	4.674
AnimeInterp ft. [106]	*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>

our data using DoG (Eq. 2.3), 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. 2.6 for examples of CD evaluation. In this work, we define chamfer distance as:

$$CD(X_0, X_1) = \frac{1}{2HWD} \sum X_0 DT(X_1) + X_1 DT(X_0) \tag{2.6}$$

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.

Table 2.2: 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. 4.3 for comparison; the last row here and in Tab. 4.3 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. [106]	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>

2.4 Experiments & Discussion

We implement our system in PyTorch [90] wrapped in Lightning [3], with Kornia [96]. Our model uses the same RFR/RAFT with SGM flows as AnimeInterp for fairer comparison [106, 112], and forward splatting is done with the official Softsplat [83] module. We train with the

Table 2.3: 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 [106] 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%

Adam [62] 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 [138] package, with the AlexNet [63] backbone. All training minimizes the total loss $L = \lambda_{lpips} L_{lpips} + \lambda_{dt} L_{dt}$, where $\lambda_{lpips} = 30$; depending 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. [106], 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 [86] that achieves upwards of 3000x speedup from the SciPy CPU implementation [117]; the algorithm is a simpler version of Felzenszwalb et. al. [34], where we calculate the minimum of the lower envelope through brute iteration. While more efficient GPU algorithms are known [16], we found our implementation sufficient.

RRLD Data Collection

As RRLD was designed to replicate the manual selection of training data, we applied RRLD to AnimeInterp's ATD dataset [106] 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 [50] and RFR flows [106] 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. Here, RRLD was calculated using FlowNet2 [50] as inference was faster than RFR [106]. 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 [108]. 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. 2.5 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.

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 [88], RIFE [48], DAIN [7], and AnimeInterp [106]) 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. 4.3, 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.

Ablation Studies

We perform several ablations in Tab. 2.2. In the first group, each of the modifications to Softsplat [83] (frozen ResNet [44] feature extractor, infilling, U-net [98] replacing GridNet [36]) contributes to SSL outperforming AnimeInterp [106]. The infilling technique improves performance the most.

In the second group of Tab. 2.2, we ablate the addition of new data filtered by RRLD (Sec. 4.3). 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 [106]: 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. 2.2, 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. 4.1b 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.

User Study Results

We summarize the user study results in Tab. 2.3, and provide the full breakdown with sample animations in the supplementary. Our study had 5 participants, meaning each entry of Tab. 2.3 has support 1615 (323 compared pairs per participant). We confirm the observations made by Niklaus et. al. and Blau et. al. [10], 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 [138, 100], 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. 2.6). 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 [10], 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 4.3 and 2.2, 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 2.3 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.

2.5 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 [127], 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 [82], we can expand our scope to work on line drawings directly.

To summarize, we identify and overcome shortcomings of previous work [106] 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.

Chapter 3: Match-free Inbetweening Assistant (MIBA): A Practical Animation Tool without User Stroke Correspondence

In traditional 2D frame-by-frame animation, inbetweening (interpolating line drawings, abbr. "IB") is still a manual and labor-intensive task. Despite the abundance of literature and software offering automation and claiming speedups, animators and the industry as a whole have been hesitant to adopt these new tools. Upon inspection, we find prior work often unreasonably expects adoption of novel stroke-matching workflows, naively assumes access to adequate center-line vectorization, and lacks rigorous evaluation with professional users on real production data. Facing these challenges, we leverage optical flow estimation and differentiable vector graphics to design a "Match-free Inbetweening Assistant" (MIBA). Unrestricted by the need for user stroke correspondence, MIBA integrates into the existing IB workflow without introducing additional requirements, and makes the raster input case feasible thanks to its robustness to vectorization quality. MIBA's simplicity and effectiveness is demonstrated in our comprehensive user study, where users with professional IB experience achieved 4.2x average speedup and better chamfer distance scores on real-world production data, given only a 5-minute tutorial of new functionality.

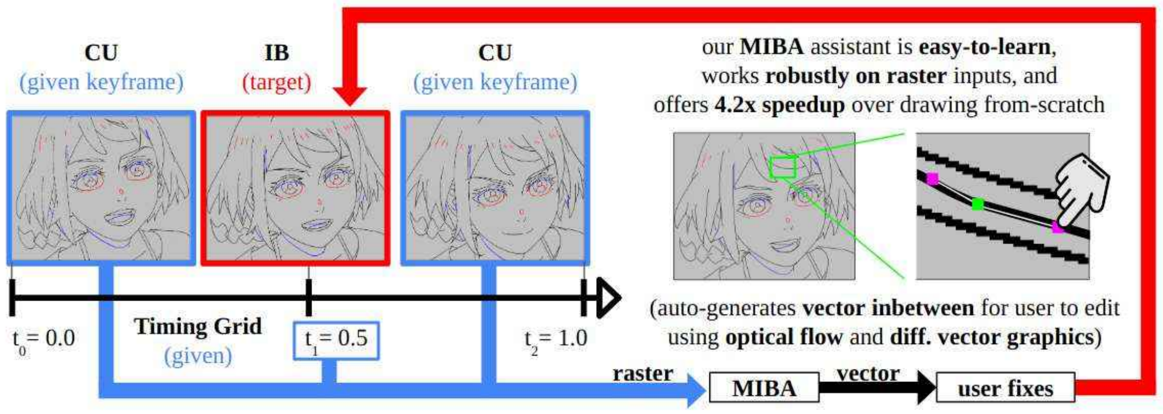

CU
(given keyframe)
IB
(target)
CU
(given keyframe)
our MIBA assistant is easy-to-learn, works robustly on raster inputs, and offers 4.2x speedup over drawing from-scratch
t_0 = 0.0
Timing Grid
(given)
t_1 = 0.5
t_2 = 1.0
(auto-generates vector inbetween for user to edit using optical flow and diff. vector graphics)
raster
MIBA
vector
user fixes

Figure 3.1: **Our MIBA system assists the inbetweening (IB) task**. In traditinoal 2D frame-by-frame animation, animators often draw "pose-to-pose", finishing **"cleaned-up keyframes" (CU)** at critical poses of a sequence first, before completing the **"inbetweens" (IB)** that spatio-temporally interpolate based on a specified **"timing grid"**. IB is notoriously labor-intensive and is still drawn manually in many productions, since existing automatic methods often fail on raster inputs and are incompatible with artist workflows. Leveraging state-of-the-art optical flow [112] and differentiable vector graphics [69], our MIBA system works **robustly on scanned-in raster** inputs, and integrates into the existing workflow so well that it can be **learned in 5 minutes** by experienced animators in the industry. Tested by professional inbetweeners on real production data, **MIBA sped up IB drawing by 4.2x** on average during our user study.

3.1 Introduction

In traditional 2D frame-by-frame animation, animators often draw "pose-to-pose": first completing "cleaned-up keyframes" (or "CU") at several critical poses of a sequence, before spatially interpolating them with "inbetweens" (or "IB") at intervals specified by a "timing grid" (Fig. 3.1). CUIB (a.k.a. "douga" in Japanese) can be drawn as vectors or rasters, but they are typically saved as aliased rasters to then be bucket-filled by digital ink and paint staff (DIP). The IB process demands precise linework and is notoriously time-consuming, taking anywhere between 5-40 minutes per frame depending on the difficulty. Across a single 24-minute episode at 12 frames per second, thousands of such inbetweens are drawn by hand; this scales to tens of thousands of drawings for seasonal shows and feature films.

The academic community has proposed many systems for offering computational IB assistance, typically in the form of matching vector strokes across keyframes to interpolate control points [123, 131, 54, 18, 133, 82, 132, 26]. But despite the abundance of work, there is still little adoption of automatic methods in industry. While there are many artistic, economic, and cultural factors contributing to this lack of progress, we observe that the research community has repeatedly overlooked major design considerations that have significant impact on practical usability and system evaluation. Specifically:

Consideration 1: Artists should not be expected to alter their workflow. It is prohibitively expensive for artists to make major changes to their established way of doing things, and invest time and effort into learning the intricacies of new tooling. Yet, the systems proposed in the literature consistently introduce additional requirements and functions that burden the animator (see summary in Table 3.1). By learning the existing workflow from professionals, we designed MIBA to automate specific steps without intrusively adding new ones (Fig. 3.2).

Consideration 2: Animators do not have vector keyframe inputs adequate for prior vector-based methods. Prior work has mostly focused on interpolation between matched vector strokes, but implicitly assumes properly-connected, noise-free, and/or nearly-identical vector topology on both input keyframes (Tab. 3.1). Unfortunately, we find that these requirements are satisfied by neither off-the-shelf vectorizations nor by artist-drawn vectors (Fig. 3.5, Fig. 3.6). MIBA on the other hand is robust to input vectorization quality, and so can feasibly handle the raster input case.

Consideration 3: Evaluation should be precise and representative of real-world animation production. Relevant IB user studies are few; most have participants without professional IB experience, tend to evaluate qualitatively on simple animations, lack details about

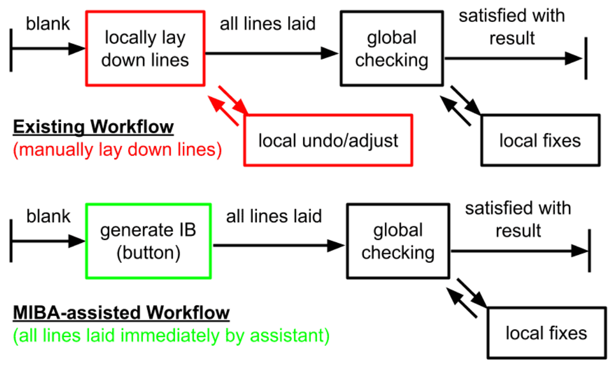


Figure 3.2: **MIBA non-intrusively assists the existing workflow**. Our proposed system (bottom) seamlessly integrates into the existing workflow as described by professionals (top). Laying down all predicted lines with a simple click advances users to the "check+fix" stage of their familiar workflow, characterized by zooming globally/locally to find/fix errors and missing lines. By replacing the time-consuming step of zooming in to lay lines, we save significant effort without deviating from established practices.

procedures and training, and neglect to report key metrics like speedup and evaluation against ground truth (see summary in Tab. 3.3). We evaluate MIBA with a comprehensive user study addressing all these points.

Taking into account these three key considerations, we make the following contributions:

- **MIBA**, a "Match-free Inbetweening Assistant" leveraging deep optical flow and differentiable vector graphics [112, 69]. Unrestricted by the need for user stroke correspondence, MIBA seamlessly integrates into the existing IB workflow, and works well on raster input thanks to its robustness to vectorization quality.

- A **comprehensive user study** evaluating our MIBA system, in which users with professional IB experience achieved both 4.2x average task speedup and better chamfer distance scores w.r.t. ground-truth on real-world production data, given only a 5-minute tutorial of MIBA's functionality.

	Additional requirements / tools to learn
MIBA (ours)	- **none** (raster input, no vector requirements) (new assistant operated by button click)
BetweenIT [123]	- requires adequate vector input for 1-to-1 stroke correspondence (Fig. 3.5) - lasso tool for stroke correspondence - point tool for vector correspondence - path tool for trajectory guidelines
VGC [26]	- requires learning completely new time-vector topology data primitives
DiLight [18]	- requires well-connected vectors (Fig. 3.6) - path tool for stroke correspondence - must set stroke matching tolerance param.
FTP-SC [133]	- requires adequate vector input for 1-to-1 stroke correspondence (Fig. 3.5) - matching tool for stroke correspondence
Narita et. al. [82]	- cannot generate vector output, only rasters are supported (Fig. 3.4)
CACANI [54]	- requires adequate vector input for 1-to-1 stroke correspondence (Fig. 3.5) - requires stroke depth layering - requires stroke occlusion orientation - new grouping, linking, inverting tools
AnimeInbet [107]	- requires strictly straight-line vectors (does not support curved lines) - requires well-connected and densely-sampled vectors (Fig. 3.6)

Table 3.1: **Comparison of new requirements and tooling in prior work**. It is costly for animators to make major changes to their established workflows, and invest time and effort into learning new tooling. Yet, systems proposed in academia consistently introduce additional requirements and functions that burden the animator. Our MIBA framework on the other hand is designed to accelerate IB with a simple button click.

3.2 Related Work

Prior vector-based inbetweening works first and foremost address the problem of vector stroke correspondence [123, 18, 133]. They assume two vector keyframe inputs with extremely similar and well-connected topologies, between which they match strokes. Based on the discovered 1-to-1 correspondence, control points are interpolated to derive the target IB vector representation. BetweenIT [123] and FTP-SC [133] provide a number of semi-automatic user tools to achieve this exact vector match, while DiLight [18] proposes guideline tools and loosens the strict correspondence requirement. More recently, AnimeInbet [107] proposed a method of fusing the two graph topologies. However, we often see that neither off-the-shelf vectorizers [122, 9, 79] nor users provide adequate vectorization quality for these methods (Fig. 3.5 & 3.6). In order to break free of the limitations imposed by vectorization, our match-free methodology completely discards the need to correspond disparate vectors, achieving a simple-to-use tool robust enough to work on vectorizations of raster scan-ins.

Stroke occlusion resolution and aesthetic non-linear curve interpolation are other aspects of IB work. CACANI [54] for example proposes a layered and oriented representation of strokes, and is able to infer occlusions automatically. BetweenIT [123], FTP-SC [133], and DiLight [18] all propose their own methods for allowing user-specified non-linear trajectories. In our work on MIBA, however, we have found that simple linear interpolation works well for real-world production IB data, and that improper occlusion is relatively quick for users to fix. We thus focus on removing the stroke correspondence paradigm; however, as the occlusion and interpolation techniques are orthogonal to our match-free contribution, they can be added after our workflow if desired.

Another very different approach to IB is taken by Narita et. al. [82], who discard the vector representation altogether in favor of rasters. They propose the direct use of optical flow to warp between keyframes (similar to video frame interpolation systems commonly used for natural RGB videos [23, 7, 48]), and improve flow estimation on sparse line drawings with the distance transform. While this approach indeed relieves the user of vector considerations, the results are heavily dependent on optical flow performance. As failure cases are common for animations with large displacements and sparse lines, the user would need to edit an inflexible raster representation (Fig. 3.4). Our MIBA on the other hand tackles vector representation issues, without resorting to rasters.

Yet another alternative is proposed by Boris et. al. [26], who introduce a novel "vector animation complex" data structure to manage interpolation of topologies across both space and time. However, the new representation is a departure from the conventional vector graphics paradigm, and inhibitively requires the artist to learn a new framework of thought in order to inbetween.

As summarized in Tab. 3.1 and illustrated in Fig. 3.2, our MIBA system distinguishes itself from prior work by not requiring significant changes to animator tooling or workflow. Additionally, we provide one of the most comprehensive user studies for IB in the literature to evaluate MIBA (Tab. 3.3); we report metrics with respect to ground-truth production data from the real world, provide explicit details on speed and interaction gains, and test with professional IB animators.

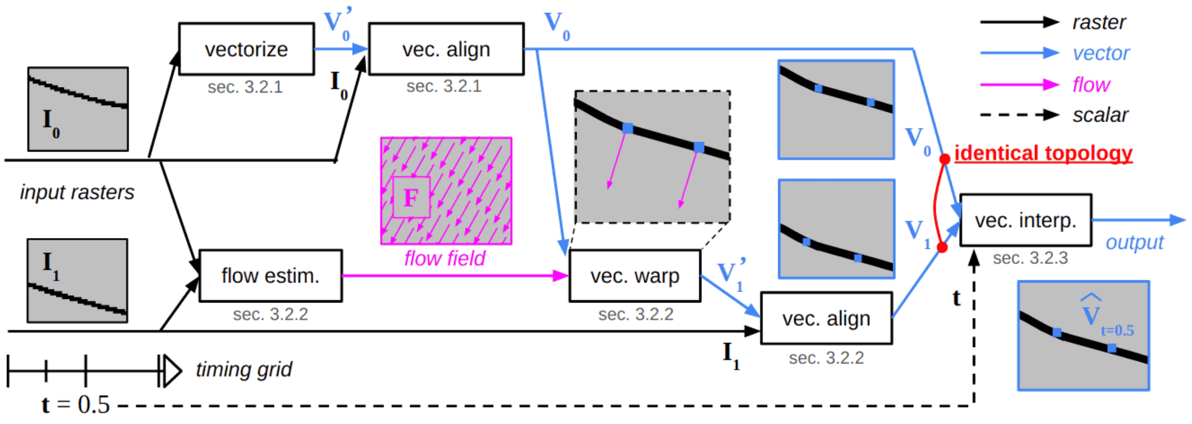


Figure 3.3: **Schematic of our proposed system**. Given the left two raster keyframes (I_0, I_1) and the interpolating position on the timing grid ($t = 0.5$), our system produces a vector inbetween ($\hat{V}_t$, right) that can then be adjusted as needed, without requiring the user to specify stroke correspondences between vectorized versions of the input frames. The key to achieving this match-free framework lies in our ability to robustly warp and align the vectorization of the first frame (V_0, top) to a raster of the second frame (I_1, bottom). This way, we obtain topologically identical vectors (red) aligned to respective frames, ready for interpolation. The system leverages optical flow estimation and differentiable vector graphics to produce reasonable stroke-raster alignment.

3.3 Methodology

Our system operates on a single timing grid sequence at a time (Fig. 3.1). As illustrated in Fig. 3.2, MIBA lays down predictions of where the vector lines of IB should be placed. Our system uses a match-free method to return a vector IB drawing for each interval specified by the timing grid. The user may then use common vector editing tools, or choose to rasterize at any time and use common raster tools. Below, we define the input and output representations more formally, before describing our algorithm and provided user interaction tools.

3.3.1 Input/Output Representations

The inputs to the MIBA system are: a timing grid T, and two cleaned keyframes (Fig. 4.1, left-most side). The timing grid T is a sorted array of unique values $t_i \in [0, 1]$, where $t_0 = 0$ and

$t_1 = 1$. The two CU keyframes are assumed to be rasters (I_0, I_1 in Fig. 4.1). In the case where a vector representation is available, we still rasterize before inputting to our system; this ensures consistency of outputs, and relieves any mental burden on the animators to consider line topology when drawing CU. Similar to the IB to be generated, the CU keyframes are binary-aliased for eventual digital coloring with paint-bucket tools. The CU often only have few colors aside from black; by default in the Japanese pipeline, red denotes highlights, blue is shadow, and green is a special effect or second shadow/highlight.

The output representation of MIBA is a vector graphics representation ($\hat{V}_t$ in Fig. 4.1), which can be rasterized if desired. We select to use cubic Bézier curves, with a polar representation for control points. The representation backend supports arbitrary graph connectivity, though for our experiments we only work with acyclic graphs, similar to SVGs. In practice, our system renders curves as short piecewise-linear line segments; we thus support variable line width across a single curve, although we found that a single global thickness worked well enough for our specific data.

3.3.2 Match-free Inbetweening

Our proposed match-free inbetweening method is illustrated in Fig. 4.1. Similar to previously proposed frameworks, each curve of the output is derived by interpolating between the vertex/handle coordinates of two existing curves (V_0 and V_1, one representing each vectorized input image). However, to fit this paradigm, prior work struggles at finding the two correct corresponding strokes to inbetween from each input, because the input vectorizations have either unreasonably different topology (Fig. 3.5) or inadequately noisy connectivity (Fig. 3.6).

The key insight of our MIBA method is that we can warp one frame's vector (V_0) to align with the other's raster (I_1). This way we do not need to match two disparate vectorizations, and instead can directly interpolate between two coordinate-value configurations of the same vector topology (Fig. 4.1, red). Put another way, we find offsets from one vector to look like the other raster; interpolation equates to moving by a specified fraction of those offsets.

Naturally, since no matching occurs between different vectorizations, MIBA is robust to the connectivity and noisiness of the input vectors; by discarding the stroke correspondence paradigm, we make IB assistance on vectorized scanned-ins a feasible reality (Fig. 3.6). In addition, animators can easily operate MIBA without the intricate stroke correspondence and match guidance tooling required by prior work (Tab. 3.1).

From the perspective of solving the stroke occlusion problem, MIBA intrinsically resolves occlusion at the warp and align stages. At these steps, strokes are effectively occluded by shortening curves to the occlusion boundary, in order to satisfy alignment to the opposing raster frame. Note that this occurs independently of local topology at the junction, as the optimization condition is imposed directly on a raster render of the stroke graph. While we found that improper occlusions still may appear from imperfect alignments, animators are able to fix them relatively quickly.

Note that MIBA is asymmetric with respect to the input frame order; in other words, the system output for interpolating $t = 0.5$ will be different if the two input CU keyframes are swapped. Users can choose which direction to use MIBA, although in practice we find many users simply stick with the default forward direction.

The subsections below describe in more detail the steps outlined in Fig. 4.1, and how we leveraged state-of-the-art optical flow and differentiable vector graphics to achieve this non-trivial vector-raster alignment across frames.

Vectorization with Alignment Post-processing

As previously mentioned, MIBA only works with a single vector topology that does not need to be matched; thus the method is robust to the input vector representation quality. In Fig. 3.6, we demonstrate our system with Weber AutoTrace [122], PolyVectorization [9] (simplified with [101]), and Virtual Sketching [79]; in all three cases, our system was able to deliver similar reasonable results. We default to using AutoTrace in our program for its quick processing speed.

Across the different vectorization algorithms, we inevitably found imperfections in the linework that did not match with the input raster; usually, this came in the form of "wobbly" curves. To improve the base vectorizations (V_0'), we optimize the vector image to match the input raster (I_0) using DiffVG [69] by minimizing

$$V_0 = \underset{\theta}{\operatorname{argmin}}\, L_2\left(R(\theta),\ I_0\right), \tag{3.1}$$

where θ represents the vertices and Bézier control points initialized at V_0', and R denotes the DiffVG differentiable vector rendering operation. The optimization is run for 32 iterations, with Adam [62] on MSE image loss with unit learning rate. To facilitate alignment of initially distant lines, we additionally apply Gaussian blur to the differentiable render before loss evaluation for the first half of optimization (with a ramping sigma).

Vector Warping & Alignment

The goal of warping here is to roughly position the post-processed vectorization (V_0) over the other raster frame (I_1), to initialize the more expensive alignment optimizations. We estimate the optical flow (F) between the two raster inputs using an off-the-shelf RAFT model [112];

inspired by previous work on raster inbetweening of line art [82], we preprocess the sparse line drawings into dense distance transform images to improve the flow estimation.

To perform the warp, we simply offset each vertex by the flow sampled at its image coordinates, denoted as $V_1' = V_0 + F[V_0]$. Even without modifying the polar Bézier handle values, we found that the warp gave results that were reasonable, but not yet acceptable for interpolation without further alignment.

With the vectorization of the first frame roughly warped to the second raster, we remove remaining alignment imperfections by DiffVG optimization [69]. The optimization process is the same as vector postprocessing previously described in Sec. 3.3.2, but θ is initialized at V_1' and the raster target is instead I_1:

$$V_1 = \operatorname*{argmin}_{\theta} L_2\Big(R(\theta),\ I_1\Big). \tag{3.2}$$

Note that there still may be imperfect alignments, since the inherent topology of the two frames is often different (Fig. 3.4, 3.5, 3.6). However, we find that in many cases these incompatibilities can be quickly fixed by the animator after interpolation, still at a time discount compared to manually laying down all the lines.

Vector Interpolation

Despite the emphasis that prior work puts on interpolating aesthetically between two curves [123, 133, 18], we find that a simple linear interpolation of vertex coordinates and polar Bézier handle values is sufficient enough to satisfy professional IB animators. As the choice of interpolation method here is orthogonal to our contribution of match-free assistance, this step can be freely interchanged with interpolation schemes from other work. For timing grids with more than one IB, we simply cache the alignment offsets and lerp to new intervals as needed. Once the rasterization of the first frame is appropriately offset to the target IB, the user is free to correct any imperfections of the MIBA output using vanilla vector manipulation tools.

3.3.3 User Interaction

We implemented an interactive web browser app with Vue.js. Our app has basic features typical to modern IB software, including a navigable viewport, toolbar with options, raster and vector layers, frame/layer selection panels, onionskinning and frame-flipping, tool keybindings, undo/redo history, etc. Pen/stylus input is supported, and is functionally equivalent to the mouse.

MIBA assistance is implemented as buttons attached to each timing grid sequence the user is asked to IB. The user clicks on the provided "assist" button, which provides a preview of the generated IBs, and then clicks "use" on the previews they would like to use. This inserts the MIBA output as a vector layer on the frame in question, which the user is then free to edit or rasterize. As MIBA is asymmetric with respect to input frame order, each timing grid is equipped with two "assist" buttons, one for each direction; the user can elect to preview and use either as they please. In order to disable the assistant for certain parts of the user study, we simply remove the "assist" buttons from the interface; all other parts of the program work as a typical piece of 2D animation software.

Standard vector, raster, and layer editing tools are provided: selection, copy/paste, layer transformation, vector drawing (with line color, width, and curvature), vector manipulation (vertices and Bézier handles), stroke deletion, and raster brush (with color, width, and erase). We additionally provide an "unpeg" function; this allows users to temporarily shift and align other

usr.	**sample-A1**										**sample-A2**										**third sample (time permitting)**										
ID	time				interaction			chamfer distance			time				interaction			chamfer distance			samp.	time				interaction			chamfer distance		
	est.	man.	asst.	**spd+**	man.	asst.	**gain**	man.	asst.	**gain**	est.	man.	asst.	**spd+**	man.	asst.	**gain**	man.	asst.	**gain**	ID	est.	man.	asst.	**spd+**	man.	asst.	**gain**	man.	asst.	**gain**
1	5:00	13:50	4:50	**2.8x**	23	39	**-70%**	26.0	7.5	**71%**	5:00	12:27	3:40	**3.3x**	17	23	**-35%**	25.9	6.6	**74%**	B	30:00	56:14	6:50	**8.2x**	170	37	**78%**	20.1	16.0	**20%**
2	-	16:25	4:10	**3.9x**	142	63	**56%**	10.2	8.2	**20%**	-	11:24	4:15	**2.6x**	147	75	**49%**	9.4	7.3	**23%**	C	-	8:28	2:09	**3.9x**	108	19	**82%**	8.8	6.8	**23%**
3	10:00	16:54	7:50	**2.1x**	175	118	**33%**	14.0	7.1	**50%**	10:00	15:28	9:15	**1.6x**	103	143	**-39%**	11.2	6.3	**44%**	D	10:00	19:13	9:23	**2.0x**	160	124	**23%**	5.8	5.2	**11%**
4	10:00	18:33	1:16	**14.6x**	77	17	**78%**	10.1	8.1	**19%**	10:00	13:46	1:10	**11.8x**	68	2	**97%**	10.2	7.2	**29%**	E	20:00	45:38	9:12	**4.9x**	297	105	**65%**	6.8	6.7	**1%**
5	5:00	28:56	5:55	**4.8x**	372	114	**69%**	8.9	7.1	**20%**	5:00	29:21	9:19	**3.1x**	459	123	**73%**	8.4	6.5	**22%**	-	-	-	-	-	-	-	-	-	-	-
6	20:00	29:05	4:59	**5.8x**	210	81	**61%**	20.5	8.1	**61%**	20:00	24:07	2:05	**11.5x**	239	16	**93%**	18.4	6.4	**65%**	-	-	-	-	-	-	-	-	-	-	-
7	2:00	39:19	16:04	**2.4x**	210	163	**22%**	9.5	7.3	**23%**	2:00	29:10	5:42	**5.1x**	142	57	**60%**	7.3	7.5	**-3%**	-	-	-	-	-	-	-	-	-	-	-
8	20:00	40:49	16:15	**2.5x**	249	137	**45%**	21.2	8.1	**62%**	20:00	29:37	7:23	**4.0x**	279	97	**65%**	17.9	8.2	**54%**	-	-	-	-	-	-	-	-	-	-	-

Table 3.2: **User study results**. A total of N=8 users participated in our study, of which all but one have over 2 years of professional IB experience. Participants were asked to inbetween two frames between the keys of sample A (first two column groups), and a third sample if time permitted. We ask each participant to draw each IB twice, once manually and once using our MIBA output as a starting point (man. vs. asst.). We consistently observe not only the significant time speedups and reduction in required interactions, but also a reduction in chamfer distance w.r.t. the ground truth IB when using the assistant (positive delta is good); this is usually due to the assistant missing fewer lines than users. The user study demonstrates that MIBA can successfully reduce the workload and improve accuracy on IB tasks in real production situations.

images in the sequence to visually reduce the gap between the lines to inbetween.

Note that the entirety of our proposed MIBA system can be operated with simple button clicks; as summarized in Tab. 3.1, this contrasts with prior work, in which additional requirements must be considered, workflows must be rearranged, and new intricate tooling must be learned. Aside from our straight-forward MIBA buttons, all the other functions in our program as described above can be considered "vanilla" tooling for 2D animation software that digital animators are already familiar with. This simplicity reflects how MIBA integrates seamlessly with the existing manual workflow, without burdening the artist with an intrusive new framework.

3.4 User Study Evaluation

3.4.1 Data

We evaluate our system on real production data provided by Japanese anime production studios. The CUIB line drawings (a.k.a. "douga") were aliased scanned-in rasters with at most

	used prod. data	reports metrics w.r.t. GT	reports exact times & speedup	reports interact. eff. gain	N users	training time for new tools	user professional experience
MIBA (ours)	Y	Y	4.2x	45.3%	8	5 min	2+ years of IB (all except one user)
BetweenIT [123]	Y	-	-	83%	unclear	an afternoon	tonal department (different from IB)
VGC [26]	-	-	-	-	-	-	-
DiLight [18]	-	-	-	-	unclear	unclear	professional animators (unclear if IB)
FTP-SC [133]	Y	Y	-	93.6%	5	unclear	1 animator, 4 graduate students
Narita et. al. [82]	Y	Y	-	-	-	-	-
CACANI [54]	-	-	3.18x	-	-	-	-
AnimeInbet [107]	-	Y	-	-	36*	-	-

Table 3.3: **Comparison to evaluation performed in prior work**. Whereas prior work generally lacks critical details and measurements for their user studies, we provide comprehensive descriptions of the context and results of our study. Our speedup and gain are averages calculated from Tab 3.2; note that the numerical reports are not directly comparable, as the interactions and task loads are different in other frameworks. Notice that unlike in other studies, professional IB animators were able to achieve improved results with MIBA after only a 5-minute tutorial of new features. (*) Note that AnimeInbet's user study consisted of only perceptual ranking, without any user-tool interaction.

user	pref.*	IB xp.	check xp.	device	intu.	qual.
1	CSP	$>$ 2yrs	$>$ 2yrs	pen	5.0	4.0
2	Stylos	$>$ 2yrs	$>$ 2yrs	pen	5.0	5.0
3	CSP	$<$ 6mo	-	mouse	4.0	4.0
4	CSP	$>$ 2yrs	$>$ 2yrs	mouse	5.0	5.0
5	TB	$>$ 2yrs	$>$ 2yrs	pen	3.0	4.0
6	Flash	$>$ 2yrs	$>$ 2yrs	pen	2.5	3.0
7	CSP	$>$ 2yrs	-	pen	3.0	4.0
8	Moho	$>$ 2yrs	$>$ 2yrs	mouse	3.5	4.5

Table 3.4: **User metadata and questionnaire**. All but one user has over two years of professional IB experience, and six of eight have over two years of IB checking experience (a more senior supervisory role); for context, IB training can take between one to six months. Three participants used primarily the mouse, due to tablet unavailability, pen incompatibility, or by choice. While most were relatively satisfied with the assistant output quality (average 4.2 on 1-5 scale), the perception of our program's intuitiveness to use scored lower (average 3.9 on 1-5 scale). We believe the "intuitiveness" variation depends largely on our app's similarity to the artist's most comfortable software ("pref" column); this is a testament to the importance of fitting the user's existing workflow and tool preferences. (*) Abbreviations: Clip Studio Paint, RETAS Stylos, ToonBoom, Adobe Flash/Animate, Moho.

four line colors (black and RGB) excluding the white background. Five CU pairs (A-E) were inbetweened during our user study, and three separate sequences were displayed as examples in the tutorial but not fully inbetweened. Sample A had two IB at $t_1 = 0.5$ and $t_2 = 0.75$; all the other samples B-E had only $t = 0.5$. Each sequence included the timing grid, the two raster CU keyframes, and the ground-truth raster IB ultimately used in production (used only for evaluation, not shown to participants). As the samples were fixed, we automatically load a sample's data upon page load, and use precomputed results (V_0, V_1 in Fig. 4.1).

Note that many prior works neglect to evaluate on real-world production data (Tab. 3.3, first column). Previous methods evaluating real production data either strongly assumed adequate-quality vectorizations [123, 133], or do not support vector output representations at all [82]; our method on the other hand is able to process raw raster CU keyframe inputs, and give vector outputs. Another distinguishing point of our data is that our lines are colored (i.e. shadow and highlight lines are present); while previous methods may have been extended to more than one line color, none have shown results on production data with this common characteristic.

3.4.2 Participants

All $N = 8$ participants recruited were professional animators with either industry (7) or freelance (1) experience (Tab. 3.4). All users had at least two years of IB experience, with the exception of one user with under 6 months experience. Six of the eight participants also had over two years of IB checking experience; for context, CUIB checking (a.k.a. "douga-kensa") is a supervisory task typically given to experienced CUIB/douga animators. For comparison, previous studies have often given vague descriptions of users' experience levels, recruited participants without direct IB experience, or even fail to mention the total number of users (Tab. 3.3).

3.4.3 Procedure

The study typically lasted two hours for each participant. Users were first brought to the tutorial page with the assistant disabled, and were given a 25-minute tutorial walking through the program basics (pen tablet setup, navigation, display options, and vector/raster tools excluding the proposed assistant). Three of eight participants used primarily the mouse, due to tablet unavailability, pen incompatibility, or by choice (Tab. 3.4).

After familiarizing with the vanilla tools, we enabled the MIBA assistant buttons, and gave a 5-minute explanation on how to use them. Studies in prior work often neglect to report the amount of time it took to teach participants the new tool (Tab. 3.3). However, we believe the training time is a key factor in determining whether animators can adopt the assistants. Tools with additional requirements (Tab. 3.1) take time to habituate, and discourage usage; our MIBA on the other hand is easy enough to explain in five minutes.

At this point, users have spent roughly 30 minutes getting to know the program features. Participants then moved on to timed tasks. For each sequence, we first asked users their time estimate for one of the IB frames in their most comfortable program. Users were then asked to draw the IBs manually (with the MIBA assistant disabled), and then to start over again using the assistant. All participants started with sample A (which had two IBs to complete, marked "A1" and "A2" in Tab. 3.2), and if time permitting were given a second sample from B-E (all with only one IB frame).

At the end of the study, users were asked a brief questionnaire (Tab. 3.4) rating the program's intuitiveness to use (on a 1-to-5 scale), the quality of the assistant's predictions (also 1-to-5), and what additional features they would need to use the program for IB work.

3.4.4 Metrics

We make several measurements for each frame drawn by the users, and are mainly interested in comparing between manual and MIBA-assisted performance on a per-frame basis. While comparisons could be made across different users, there are many significant factors that cannot be controlled for, such as the similarity between our program and each user's most comfortable software.

In Tab. 3.2, we show in bold for each frame of each user: the time speedup from using the assistant, the effort gain in terms of tool interaction count (history length), and the reduction in chamfer distance with respect to the ground truth. The chamfer distance here is calculated as defined in prior work showing its relevance to perceptual line quality [23], and is measured between black-and-white binarized ground truth and an aliased rasterization of the user drawing. In addition to the drawing results, we include qualitative responses to the questionnaire at the end of the study in Tab. 3.4 (last two columns, on a 1-to-5 scale).

Note that prior studies repeatedly neglected to report the exact time speedup afforded by their proposed methods (Tab. 3.3). Some may extrapolate based on the interaction effort gain, but the reduction in number of interactions does not map directly to time savings, nor does it allow generalized comparison between methods. Our results explicitly report the manual vs. assisted times, interactions, and chamfer distance broken down by frame and by user (Tab. 3.2); we believe this evaluation gives a much more complete picture of our tool's practicality.

3.4.5 Analysis

The most striking result of the user study in Tab. 3.2 is that MIBA assistance significantly speeds up task completion compared to manual drawing (average 4.2x, min 1.7x, max 14.6x). We found that except for 4 out of 17 cases (where users provided an estimate), users were even able to finish in less time than their estimate in their most familiar software (Tab. 3.2, time est. vs. asst. columns). We suspect that improving the user experience and allowing more time to familiarize with our specific program would further improve the speedups from our assistant. The reduced time afforded by the assistant is also reflected in the overall reduction of interaction count compared to manual drawing.

The results in Tab. 3.2 show that users additionally achieved better chamfer distance with respect to ground-truth using our assistant, on all but 1 of 20 frames drawn. In other words, in the vast majority of cases, our assistant also lets animators draw closer to the actual IB used in production. The ground truth frames, while also drawn by professional IB animators like our participants, went through an additional IB checking procedure by another staff member; notably, many missing lines and imperfect shapes are caught by checkers at this stage. We observe that as the manual IB drawn in our study are not checked by a second person, they sometimes are missing lines and do not preserve shape volumes as exactly as our assistant; these factors all contribute to superior chamfer distance when using MIBA, and are reflected qualitatively in Fig. 3.7.

Note also from Fig. 3.7 that most participants asked to disregard line color for the study. The UI for switching colors was not optimal and hindered the users; some participants preferred to change colors with a separate tool after drawing all lines, but our UI did not have this function.

Thus we allowed users to use all-black and discarded color considerations from all quantitative evaluation. However, as MIBA initializes the workflow with colored lines, all our users were able to produce correctly-colored IB on assisted frames.

The results also hint that workflow habits from previous software set different user expectations, and take time to unlearn. The user IDs in Tab. 3.2 are sorted by the time to manually complete the first frame A1; we find that this is roughly representative of a user's comparative speed using our program. Looking at Tab. 3.4, we see that the first four fastest users of our program all gave high intuitive scores above 4.0 and preferred using CSP or Stylos (popular animation software) for their work; on the other hand, the others gave lower intuitive scores between 2.5-3.5 and used various other software. While we did not explicitly design our program to match any particular existing software, we can clearly see that users' experiences with our app are related to their prior experiences. This demonstrates that even seemingly trivial parts of the existing workflow like keybindings and specific tool modes of vanilla drawing functionality play an important role in an artist's perception of the program. We expect that proposing completely new tooling and requirements found in prior work (Tab. 3.1) may be a significant deterrent to adoption, and re-emphasize the need to integrate into existing workflows like MIBA does.

3.5 Limitations and Future Work

A major limitation of our work is the computational cost. Each cached alignment for a sequence requires one minute of compute on a GTX 1080Ti; AutoTrace [122] vectorization takes around 2 seconds, both DiffVG vector alignments take 20-30 seconds, and all other steps are negligible (below one second each). While optical flow may be compute intensive, the GPU forward pass (even with distance transform [23]) is still less than one second. In future work, it is possible to improve the performance of vectorization and DiffVG.

Another limitation of our current system is the lack of trajectory control. Many prior works [123, 18, 54] have stressed the importance of supporting user-editable trajectories by non-linear arcs. However, in practice, we have found the majority of inbetweens in our datasets to be linear interpolations of CU keyframes. This may be due to differences between the western and eastern pipelines (our data comes from the latter). Note that most trajectory editing techniques in prior work can be implemented alongside MIBA, as they are orthogonal to our contribution of match-free assistance.

3.6 Conclusions

In conclusion, we present an assistant for inbetweening traditional 2D animation. We found that prior work often unreasonably expects artists to adopt new workflows, naively assumes access to adequate vectorization, and lacks evaluation with professional users on real production data. Facing these challenges, we leverage optical flow and differentiable vector graphics to design a "Match-free Inbetweening Assistant" (MIBA). Unrestricted by the need for stroke correspondence, MIBA integrates into the existing IB workflow without introducing additional requirements, and makes the raster input case feasible thanks to its robustness to vectorization quality. MIBA's simplicity and effectiveness is demonstrated in our user study, where users with professional IB experience achieved 4.2x average speedup and better chamfer scores on real-world production data, given only a 5-minute tutorial of new functionality.

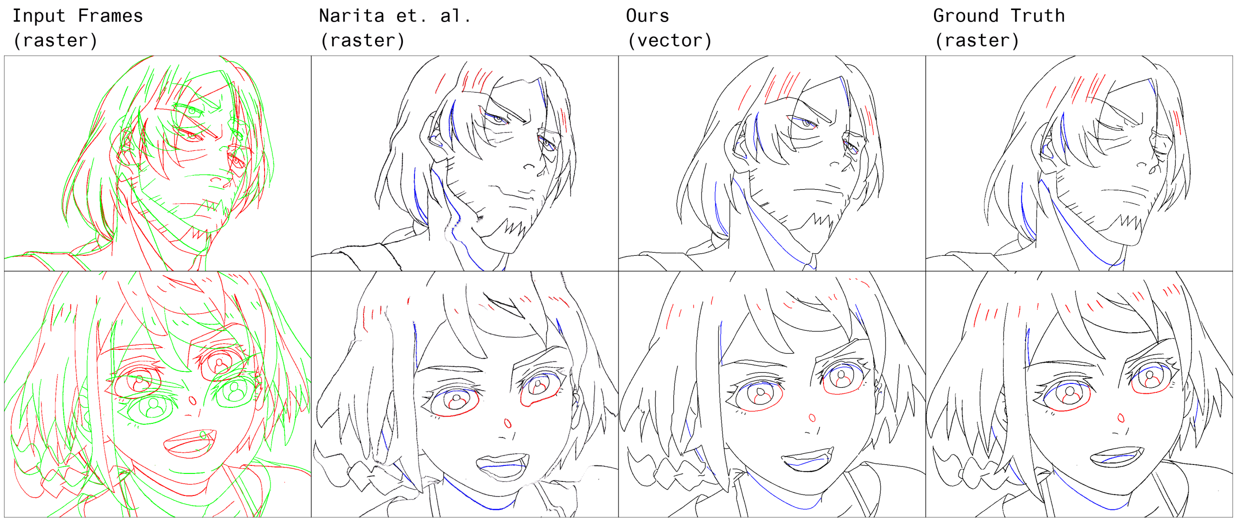


Figure 3.4: **Limitations of raster output**. While allowing raster output removes concerns of vector topology [82], we lose vector editability, which is critical to the "check+fix" stage of the existing IB workflow (Fig. 3.2). Raster outputs like the ones generated by Narita et. al. are often plagued with artifacts and poor line quality, which the artist must completely erase and redraw, even for minor imperfections. Our MIBA system addresses the vector topology concerns directly without resorting to less editable rasters, and often performs even better than raster warping due to our additional alignment optimizations (Sec. 3.3.2).

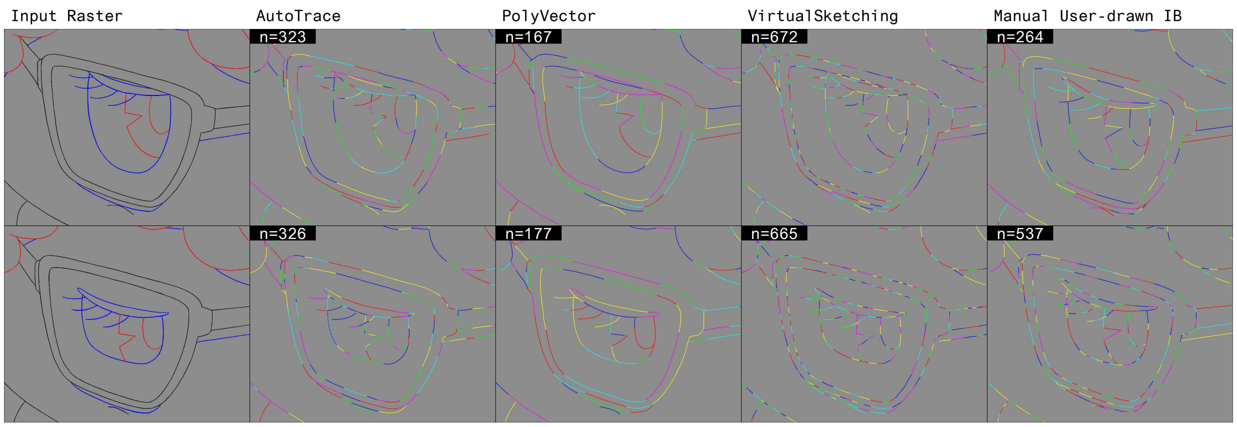


Figure 3.5: **Challenges of 1-to-1 stroke correspondence**. Prior work [123, 133] unreasonably expects vectorizers and users to produce 1-to-1 topology for their IB systems to function. Even for somewhat similar frames (top vs. bottom rows), off-the-shelf vectorizers [122, 9, 79] will return incompatible topologies with highly variant stroke count. In the last column, we show the two corresponding IBs manually drawn by a user in our study; this shows that topology correspondence is not considered in the existing manual workflow. Our MIBA system is free of vector-vector correspondence assumptions, and thus does not require new curve-matching tools for animators to operate.

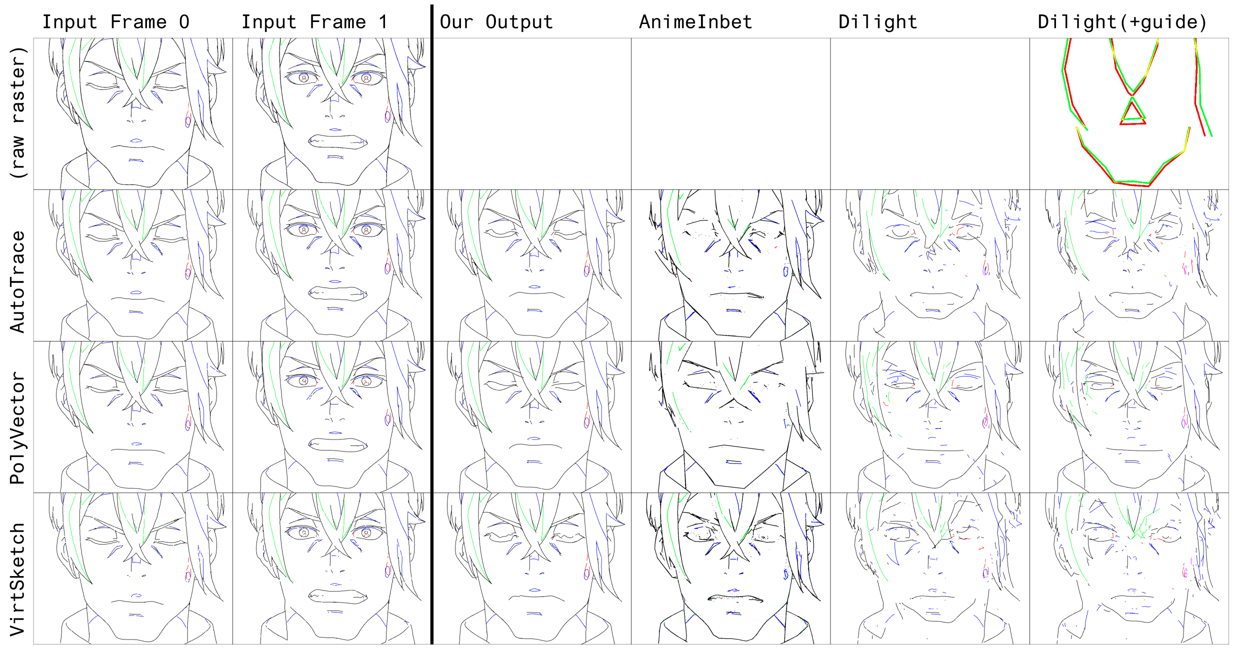


Figure 3.6: **Inadequate vectorization quality for prior work**. Prior vector-based methods without strict 1-to-1 stroke matching requirements (like AnimeInbet [107] and Dilight [18]) are highly sensitive to the quality of input vectors. Both AnimeInbet and Dilight (rightmost columns) produce unusable artifact-ridden results, regardless of which off-the-shelf vectorizer is used to preprocess the raster input (left columns, AutoTrace [122], PolyVector [9], VirtualSketching [79]). This demonstrates the impracticality of prior methods on the very common raster input use case. On the other hand our robust MIBA system innately handles vectorization, achieving consistently reasonable results across different vectorizers.

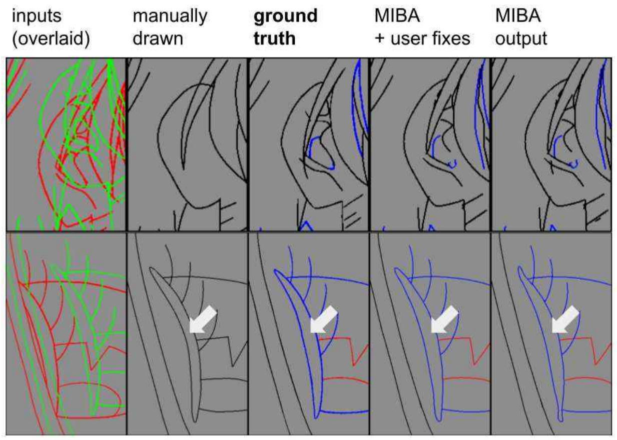


Figure 3.7: **MIBA-assisted drawings are closer to ground-truth**. We see in Tab. 3.2 that MIBA assistance reduces the chamfer distance to the ground truth IB (which had been additionally checked for correctness before use in real production). We find two potential reasons for this: Firstly (top row), it is easier to miss lines/details when manually drawing from-scratch opposed to starting with our assistant. Secondly (bottom row), MIBA is sometimes more precise than the human hand at correctly preserving volumes as they move. Another point of interest is that the manual column above is uncolored; this is because most users asked to disregard line color when manually drawing (the color switch UI slowed them down). MIBA brings the added benefit of already having line colors at initialization.

Chapter 4: Stylized Single-view 3D Reconstruction from Portraits of Anime Characters

We propose PAniC-3D, a system to reconstruct stylized 3D character heads directly from illustrated (p)ortraits of (ani)me (c)haracters. Our anime-style domain poses unique challenges to single-view reconstruction; compared to natural images of human heads, character portrait illustrations have hair and accessories with more complex and diverse geometry, and are shaded with non-photorealistic contour lines. In addition, there is a lack of both 3D model and portrait illustration data suitable to train and evaluate this ambiguous stylized reconstruction task. Facing these challenges, our proposed PAniC-3D architecture crosses the illustration-to-3D domain gap with a line-filling model, and represents sophisticated geometries with a volumetric radiance field. We train our system with two large new datasets (11.2k Vroid 3D models, 1k Vtuber portrait illustrations), and evaluate on a novel AnimeRecon benchmark of illustration-to-3D pairs. PAniC-3D significantly outperforms baseline methods, and provides data to establish the task of stylized reconstruction from portrait illustrations.

4.1 Introduction & Related Work

With the rise of AR/VR applications, there is increased demand for not only high-fidelity human avatars, but also non-photorealistic 3D characters, especially in the “anime” style. Most

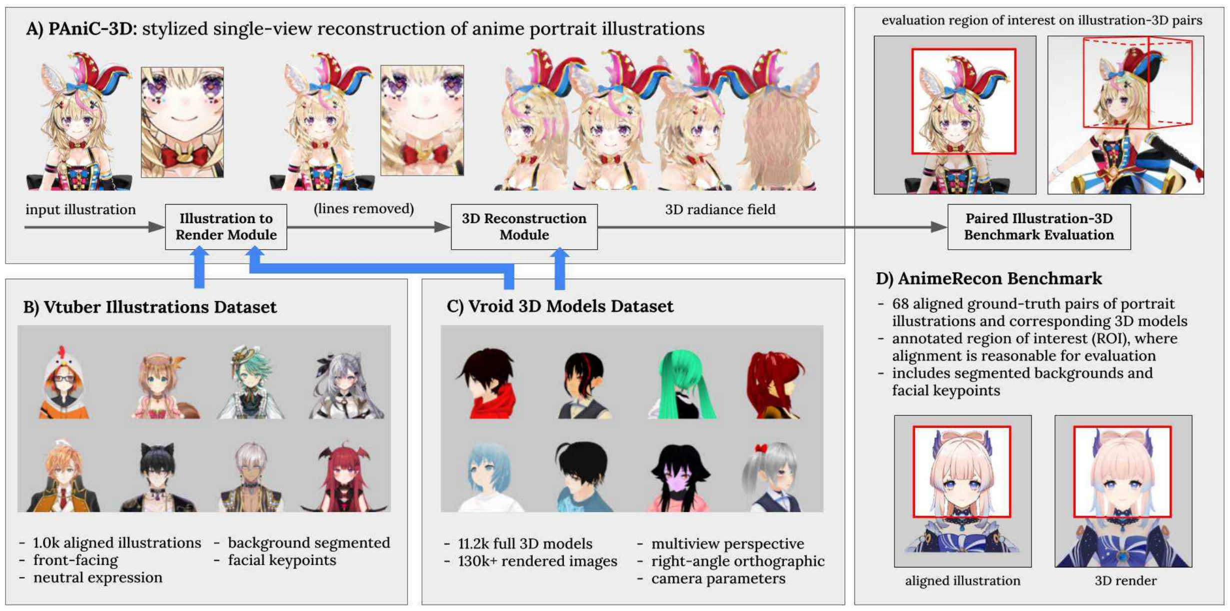

A) PAniC-3D: stylized single-view reconstruction of anime portrait illustrations
input illustration
Illustration to Render Module
(lines removed)
3D Reconstruction Module
3D radiance field
Paired Illustration-3D Benchmark Evaluation
evaluation region of interest on illustration-3D pairs
B) Vtuber Illustrations Dataset
- 1.0k aligned illustrations
- front-facing
- neutral expression
- background segmented
- facial keypoints
C) Vroid 3D Models Dataset
- 11.2k full 3D models
- 130k+ rendered images
- multiview perspective
- right-angle orthographic
- camera parameters
D) AnimeRecon Benchmark
- 68 aligned ground-truth pairs of portrait illustrations and corresponding 3D models
- annotated region of interest (ROI), where alignment is reasonable for evaluation
- includes segmented backgrounds and facial keypoints
aligned illustration
3D render

Figure 4.1: Overview of contributions. Our (A) PAniC-3D system is able to reconstruct a 3D radiance field directly from a line-based portrait illustration. We gather a new (B) Vtuber illustration dataset and (C) Vroid 3D models dataset in order to cross the illustration-render domain gap and supervise reconstruction. To evaluate, we provide a new (D) AnimeRecon benchmark of paired illustrations and 3D models, establishing the novel task of stylized single-view reconstruction of anime characters. (Art attributions in suppl.)

character designers typically create concept illustrations first, allowing them to express complex and highly diverse characteristics like hair, accessories, eyes, skins, headshapes, etc. Unfortunately, the process of developing illustrated concept art into an AR/VR-ready 3D asset is expensive, requiring professional 3D artists trained to use expert modeling software. While template-based character creators democratize 3D avatar creation to an extent, they are often restricted to existing 3D assets compatible with a specific internal body model.

We propose PAniC-3D, a system to automatically reconstruct a stylized 3D character head directly from illustrated (p)ortraits of (ani)me (c)haracters. We formulate our problem in two parts: 1) implicit single-view head reconstruction, 2) from across an illustration-3D domain gap. To summarize our contributions:

- **PAniC-3D**: a system to reconstruct the 3D radiance field of a stylized character head from a single line-based portrait illustration.
- The **Vroid 3D dataset** of 11.2k character models and renders, the first such dataset in the anime-style domain to provide 3D assets with multiview renders.
- The **Vtuber dataset** of 1.0k reconstruction-friendly portraits (aligned, front-facing, neutral-expression) that bridges the illustration-render domain gap through the novel task of line removal from drawings.
- The **AnimeRecon benchmark** with 68 pairs of aligned 3D models and corresponding illustrations, enabling quantitative evaluation of both image and geometry metrics for stylized reconstruction.

Implicit 3D Reconstruction

While there has been much work on mesh-based reconstruction from images [73], these systems are not expressive enough to capture the extreme complexity and diversity of topology of our 3D characters. Inspired by the recent successes in generating high-quality 3D radiance fields [20, 21, 32, 129], we instead turn to implicit representations. However, to achieve high-quality results, recent implicit reconstruction work such as PixelNerf [134] tend to operate solely from 2D images, due to the lack of publicly-available high-quality 3D data. Some implicit reconstruction systems like Pifu [99] employing complex 3D assets have shown reasonable success using point-based supervision, but require careful point sampling techniques and loss balancing to work properly.

In the presence of complex high-quality 3D assets, as in our case, we demonstrate the superiority of differentiable volumetric rendering for reconstruction. We build off of recent unconditional generative work (EG3D [20]), formulating the problem of reconstruction as a conditional generation, proposing several architecture improvements, and applying direct 2.5D supervision signal as afforded by our 3D dataset.

Anime-style 3D Avatars and Illustrations

It is a fairly common task for 3D character artists to produce a 3D model from a portrait illustration; however from the computer graphics standpoint, this stylized reconstruction setup adds additional ambiguity to an already ill-posed problem. In addition, while there's work in the popular anime/manga domain using 3D character assets (for pose estimation [57], re-targetting [59, 55], and reposing [71], etc.), there's a lack of publicly-available 3D character assets with multi-view renders that allow scalable training (Tab. 4.1). In light of these issues, we propose AnimeRecon (Fig. 4.1d) to formalize the stylization task with a paired illustration-to-3D benchmark, and provide the Vroid dataset (Fig. 4.1c) of 3D assets to enable large-scale training.

Within the problem of stylized reconstruction, we solve the unique task of contour removal from illustrations. There is much work on line extraction [124], sketch simplification [104], reconstruction from lines [31], and other line exploits for artistic imagery [23]; however, the removal of lines has not yet seen an application for study. We examine this novel contour deletion task in the context of adapting drawings to render-like images more conducive to 3D reconstruction; we find that naive image-to-image translation [140, 58] is unsuited to the task, and propose a simple yet effective adversarial training setup with facial feature awareness. Lastly, we provide a Vtuber dataset (Fig. 4.1b) of portraits to train and evaluate contour removal for 3D reconstruction.

3D datasets	Vroid	AniRec.	AC	CoNR	ADD
3D avail.	Y	Y	-	-	-
renders avail.	Y	Y	Y	-	-
multiview	Y	Y	-	Y	Y
paired illustr.	-	Y	-	-	-

Table 4.1: 3D anime datasets comparison. Our new Vroid dataset is the first to make 3D anime models and multiview renders available. The AniRecon benchmark allows quantitative evaluation of both 2D image and 3D geometry metrics. Others left-to-right: AnimeCeleb [59], CoNR [71], Anime Drawings Dataset [57].

Portraits	Vtuber	AC	AP	iCF	BP
illustrations	Y	-	Y	Y	Y
aligned face	Y	Y	Y	Y	-
front-facing	Y	Y	-	-	-
neutral expr.	Y	Y	-	-	-
segmented	Y	Y	-	-	Y
face kpts.	Y	-	-	-	Y

Table 4.2: Anime image datasets comparison. Our new Vtuber dataset allows us to examine the domain gap between line-based illustrations and 3D renders, and is filtered/standardized specifically for characteristics desirable in 3D head reconstruction. Others left-to-right: AnimeCeleb [59], AnimePortraits [11], iCartoonFace [139], BizarrePose [24].

4.2 Methodology

PAniC-3D is composed of two major components (Fig. 4.1a): a 3D reconstructor trained with direct supervision to predict a radiance field from a given front render, and an illustration-to-render module that translates images to the reconstructor's training distribution. The two parts are trained independently, but are used sequentially at inference.

3D Reconstruction Module

The 3D reconstruction module Fig. 4.2 is trained with direct supervision to invert a frontal render into a volumetric radiance field. We build off of recent unconditional generative work

(EG3D [20]), formulating the problem of reconstruction as that of conditional generation, proposing several architecture improvements, and applying direct 2.5D supervision signal as afforded by our 3D dataset.

Conditional input: The given front orthographic view to reconstruct is resized and appended to the intermediate feature maps of the Stylegan2 backbone used in EG3D [20]. In addition, at the earliest feature map we give the model high-level domain-specific semantic information about the input by concatenating the penultimate features of a pre-trained Resnet-50 anime tagger [24].

Feature pooling: As the spatial feature maps are to be reshaped into a 3D triplane as in EG3D [20], we found it beneficial to pool a fraction of each feature map's channels along the image axes (see Fig. 4.2 left). This simple technique helps distribute information along common triplane axes, improving performance on geometry metrics.

Multi-layer triplane: As proposed in concurrent work [87], we improve the EG3D triplane by stacking more channels along each plane (see Fig. 4.2 center). The method may be interpreted as a hybrid between a triplane and a voxel grid (they are equivalent if the number of layers equals the spatial size). Setting three layers per plane allows better spatial disambiguation when bilinearly sampling the volume, and particularly helps our model generate more plausible backs of heads (an additional challenge not faced by EG3D).

Losses: We take full advantage of the ground-truth 2.5D representations afforded to us by our available 3D assets. Our reconstruction losses include: RGB L_1, LPIPS [138], silhouette L_1, and depth L_2; these are applied to the front, back, right, and left orthographic views, as shown in Fig. 4.2. A discriminative loss is applied to improve the detail quality, in addition to maintaining the generation orientation. We also keep the R1 and density regularization losses from EG3D training. Our 2.5D representations and adversarial setup allow us to surpass similar single-view reconstructors such as PixelNerf [134] which work only with color losses.

Post-processing: We leverage our assumption that front-orthographic views are given as input, by stitching the given input onto the generated radiance field at inference. The xy-coordinates of each pixel's intersection within the volume are used to sample the input as a uv-texture map; we cast few additional rays from each intersection to test for visibility from the front, and apply the retexturing accordingly. This simple yet effective method improves detail preservation from the input at negligible cost.

Illustration-to-Render Module

In order to remove non-realistic contour lines present in the input illustration, but absent in a diffusely-lit radiance field, we design an illustration-to-render module (Fig. 4.3). Assuming access to unpaired illustrations and renders (our Vtuber and Vroid datasets, respectively), the shallow network re-generates pixel colors near lines in the drawing in order to adversarially match the render image distribution.

Similar to unpaired image-to-image models like CycleGAN and UGATIT [140, 58], we also impose a small identity loss; while this may seem counter-productive for our infilling case where identity is preserved in non-generated regions, we found that this stabilizes the GAN training. Note that our setup also differs from other infilling models, in that we inpaint to match a distribution different from the input.

Following prior work extracting sketches from line-based animations [23], we use the simple difference of gaussians (DoG) operator in order to prevent double-line extraction around each stroke.

While most lines present in the drawing should be removed, certain lines around key facial

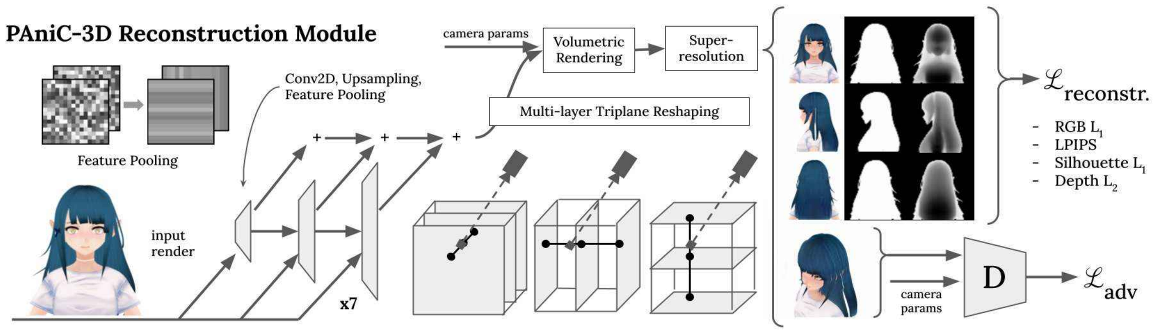


Figure 4.2: Schematic of the 3D reconstruction module. The front-orthographic input rendering is fed to a series of upsampling convolutions, with intermediate feature pooling to help distribute information along common triplane axes. The final feature stack is reshaped into a multi-layer triplane, which is volumetrically rendered and super-resolutioned to the final output. Reconstruction losses are applied to front, left, right, and back views (right view omitted in figure), and adversarial loss is applied to a random perspective view. (Art attributions in suppl.)

features must be preserved as they indeed appear in renderings (eyes, mouth, nose, etc.). We employ an off-the-shelf anime facial landmark detector [49] to create convex hulls around critical structures, where infilling is disallowed.

We show that this line removal module indeed achieves a more render-like look; it performs image translation more accurately than baseline methods when evaluated over our AnimeRecon pairs (Tab. 4.4), and removes line artifacts from the ultimate radiance field renders (Fig. 4.5).

4.3 Data

Unless otherwise mentioned, we use 80-10-10 splits for training, validation, and testing.

Vroid 3D Dataset

We collect a large dataset of 11.2k 3D anime characters from VroidHub, in order to train both the reconstruction and image-to-image translation modules of PAniC-3D. Unlike previous work in the 3D anime domain using MikuMikuDance PMD/PMX models [57, 59, 55], Vroid VRM models conform to the GLTF2 standard [41] with several extensions [118], allowing us to

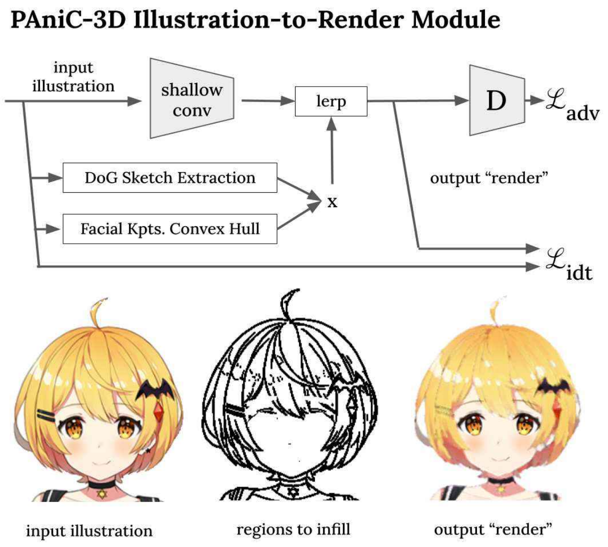


Figure 4.3: Schematic of the illustration-to-render module. We design a simple yet effective network to cross the domain gap by removing illustration contour lines absent in a diffuse 3D render, while retaining lines present around facial features like the eyes and mouth. (Art attributions in suppl.)

render using ModernGL [29, 35]. As the data is crowd-sourced, we filter against a variety of undesirable properties, such as: texture corruption, too much or too little transparency, extreme character sizes, missing bones, etc. The 11.2k renders we use represent 70% retension of our original 16.0k scraped models.

All our image data are rendered with unit ambient lighting (using diffuse surface color only), unit distance away from the "neck" bone common to VRMs [118]. Linear blend skinning is used to lower characters' arms 60-degrees from their resting T-pose position. We supersample at 1024px resolution, before bilinear downsampling to 512px for training and testing.

The dataset for PAniC-3D's reconstruction module consists of both random perspective views for adversarial training, and fixed orthographic views for the input and reconstruction losses. Each 3D model is rendered from 8 random perspective views (with uniformly-sampled 360-degree azimuth, normally-sampled elevation of 20-degree standard deviation, and fixed 30-

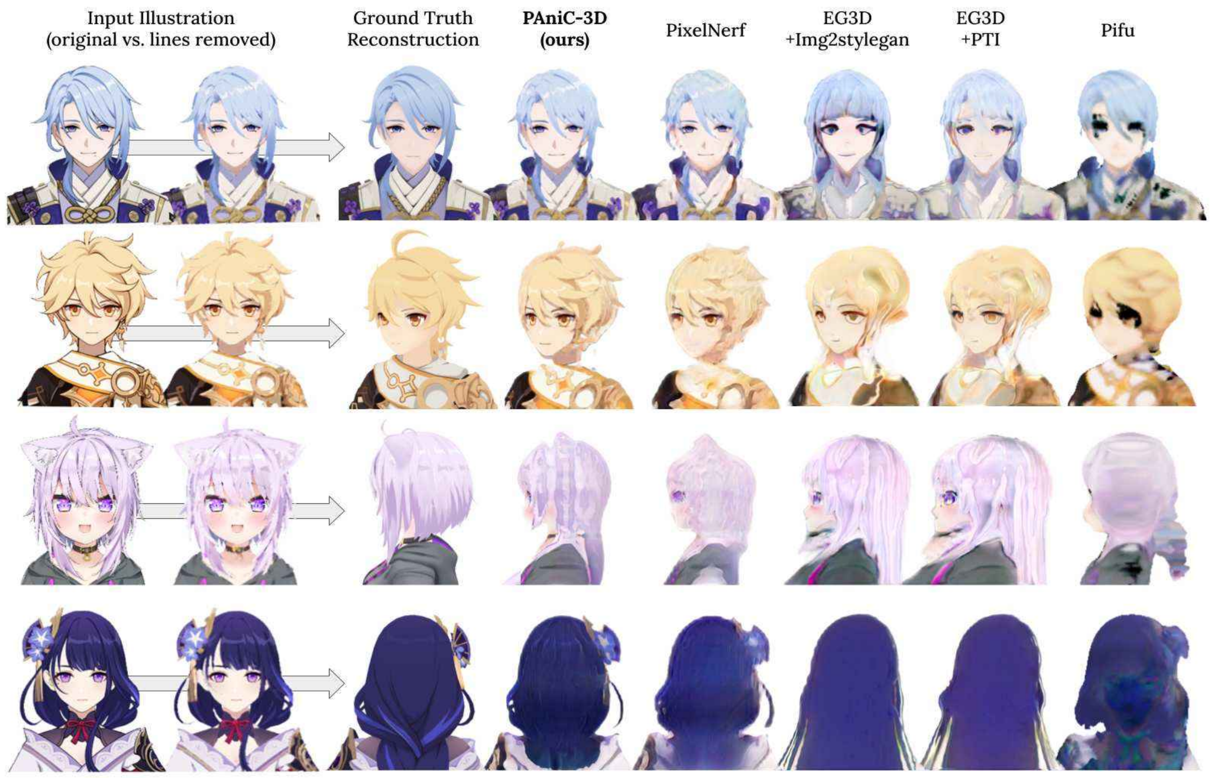


Figure 4.4: Qualitative comparison of baselines. Illustrations with lines removed by our illustration-to-render module are fed to various reconstruction frameworks; our PAniC-3D system delivers plausible reconstructions, while other methods struggle to preserve identity and predict reasonable geometry. Note that metrics (Tab. 4.3) between the displayed ground-truth and prediction are restricted to an ROI bounding box/rectangular prism (unshown) during evaluation. (Art attributions in suppl.)

degree full field-of-view); this yields a total of 89.6k perspective images, each with known camera parameters. The four orthographic views are taken at fixed 90-degree angles from the front, sides, and back. The unpaired 3D data for our image-to-image translation module is simply the front orthographic view of each character identity.

Vtuber Portraits Dataset

We gather 1004 portraits from the VirtualYoutuber Fandom Wiki as the unpaired illustration dataset for our image-to-image module. These portraits were manually filtered from 15.2k scraped images for desired properties, high-resolution, front-facing, neutral-expression,

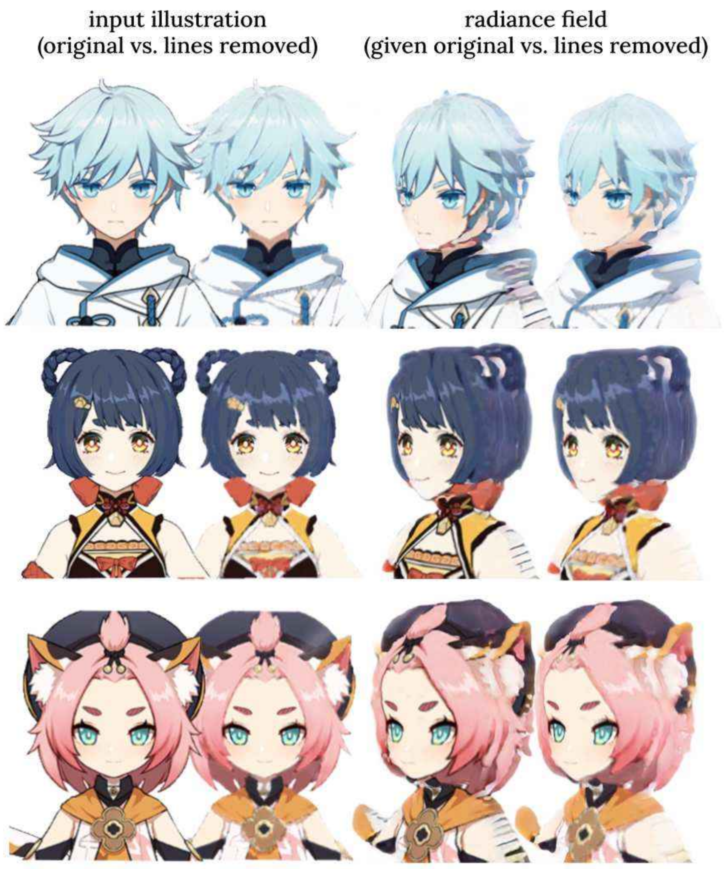

input illustration
(original vs. lines removed)
radiance field
(given original vs. lines removed)

Figure 4.5: The effect of our illustration-to-render module on reconstruction. Without our proposed line-infilling method, the reconstruction module trained on contour-less renders is unable to cross the illustration domain gap. Notice the line artifacts along the shoulders, and improper contours along the chins. (Art attributions in suppl.)

uncropped, etc. In order to mimic the image distribution of the orthographic front-view Vroid renders, we white out the backgrounds using the character segmenter from Chen [24] and select the largest connected component. In addition, we align the facial keypoints of each illustration (extracted with a pretrained YOLOv3 [49, 94]) to match the height and scale distributions of keypoint detections on the Vroid renders.

AnimeRecon Benchmark

We collect a benchmark set for stylized reconstruction by finding 68 characters with both 3D models and closely-aligned illustrations. Specifically, we source from the 3D mobile game

Genshin Impact (for which we can match character portraits from the Fandom Wiki) and the virtual talent agency Hololive (from which several members have both 3D avatars and a Fandom Wiki portrait). As the raw 3D data from both sources comes in MMD format, we convert to VRM using the DanSingSing converter; the rendering process is the same as that of Vroid.

The portraits are aligned to their corresponding front-view orthographic renders by a manually-decided mixture of YOLOv3 facial keypoints [49, 94] and ORB detections. We segment out backgrounds using the same model [24] and procedure as with Vtuber portraits. In order to maintain separation from training data, we remove all Hololive identities present in this benchmark set from the Vtuber portraits set.

Inevitably, the illustrations do not all align perfectly with their corresponding renders, and many Genshin illustrations are cropped when aligned. In order to perform more reasonable evaluation, we additionally label a rectangular region of interest (ROI) for each aligned pair, within which the alignment is appropriate; all 2D image and 3D geometry metrics reported are restricted to the ROI when calculated.

4.4 Results & Evaluation

In this section, we provide a breakdown of reconstruction and image-to-image translation performance, with both qualitative and quantitative comparison to other baselines.

3D Reconstruction Results

Metrics

As shown in Tab. 4.3, we evaluate over our new AnimeRecon benchmark using both 2D image and 3D geometry metrics. All the illustration inputs went through our illustration-to-render

	front			back			360			geom.		
	CLIP	LPIPS	PSNR	CLIP	LPIPS	PSNR	CLIP	LPIPS	PSNR	CD	F-1@5	F-1@10
PAniC-3D (full)	**94.66**	**19.37**	16.91	**85.08**	**30.02**	15.51	**84.68**	**25.25**	**15.98**	**1.33**	**37.73**	65.50
no feature pooling	**94.64**	**19.26**	**16.95**	84.06	**30.23**	**15.53**	84.30	**25.42**	15.96	1.38	35.26	65.51
no multi-layer triplane	94.39	19.99	**16.94**	84.28	30.37	15.41	**84.49**	26.13	15.82	1.44	34.55	**65.95**
no side/back loss	94.54	19.93	16.86	**85.50**	32.18	14.73	83.98	27.34	15.39	1.56	36.82	58.67
PixelNerf [30]	91.07	22.96	16.76	81.02	32.01	**15.74**	80.42	24.86	**16.31**	1.45	35.07	65.50
EG3D+Img2SG [2]	85.90	30.78	13.92	77.78	39.84	12.68	79.78	31.10	13.86	2.05	11.88	23.55
EG3D+PTI [23]	89.90	25.93	15.50	77.37	39.16	13.37	79.78	32.62	14.32	2.19	11.38	23.54
Pifu [24]	75.12	41.62	11.94	75.01	43.63	12.51	73.62	32.45	13.81	**1.35**	**37.37**	**68.13**

Table 4.3: Ablations and baselines. We evaluate and compare our system using our new AnimeRecon benchmark of illustrations with their ground-truth 3D reconstructions. Image reconstruction metrics are listed for the front (reconstruction of the given input) and back orthographic views, as well as averaged over twelve perspective spin-around views. On the right, chamfer distances and F-1 scores [40] capture the correctness of reconstructed meshes. For fair comparison, lines are preprocessed out of all input images using our illustration-to-render model, and evaluation is performed within our annotated ROIs. The top two of each category are bolded.

module before being fed to the respective method; we believe this is a fairer comparison, as all the reconstructors were trained on renders.

Image metrics are measured by comparing the predicted radiance field's integrated image with the ground-truth render from the same camera viewpoint. We show such measures for the front orthographic view (which should match the input), the back orthographic view, and an average of 12 perspective cameras circling the character at 30-degree intervals. For the front and back views, we restrict evaluation to our AnimeRecon ROI bounding box (Fig. 4.1d); for the 12 circling views, we crop to the horizontal strip of the bounding box.

We show standard color metrics like PSNR for completeness, but emphasize that as there are inevitably imperfections on the AnimeRecon illustration-render pairs (even within the ROI), perceptual metrics like CLIP image cosine similarity [93] and LPIPS [138] are generally more relevant to quality.

The geometry metrics are on the right of Tab. 4.3. We extract meshes from both the ground-truth 3D asset and the predicted radiance field (through marching cubes), and delete faces with vertices outside the rectangular prism defined by the ROI annotation (see Fig. 4.1d top). We then sample 10k points randomly from each mesh subset, to compute the point-cloud chamfer distance and F-1 scores at 5cm and 10cm [40].

Baselines

Comparisons are shown between our method and several other implicit reconstruction methods (Fig. 4.4). Of the methods shown, only Pifu [99] receives point-wise signals for optimization; we see that this works to its detriment, as the near-surface point sampling strategies bias the model towards certain geometric structures (such as black eyelashes that often protrude from the face). The other methods using volumetric rendering inherently weigh the supervision signal such that the final rendered product is consistent.

Image2stylegan [1] and Pivotal Tuning Inversion [97] are optimization-based methods of performing single-view reconstruction, the latter of which was used in EG3D to demonstrate reconstruction by projection [20]. We train an EG3D with similar hyperparameters as the Stylegan2 backbone used in our reconstruction module, and allow the two respective baselines to optimize features/weights in the prior to match the given input. Unfortunately, the EG3D prior is not sufficient to regularize the optimization, leading to implausible results; while PTI worked reasonably well for EG3D, it may not work as well on our setup where the back of the head must also be projected.

Lastly we compare to our model with the PixelNerf [134] single-view setup. The key difference in this comparison is that PixelNerf does not use adversarial nor 2.5D reconstruction losses (for fairer comparison with us, we trained PixelNerf with LPIPS in addition to L_1, resulting in significantly less blurry results). We see that without these signals provided by our available

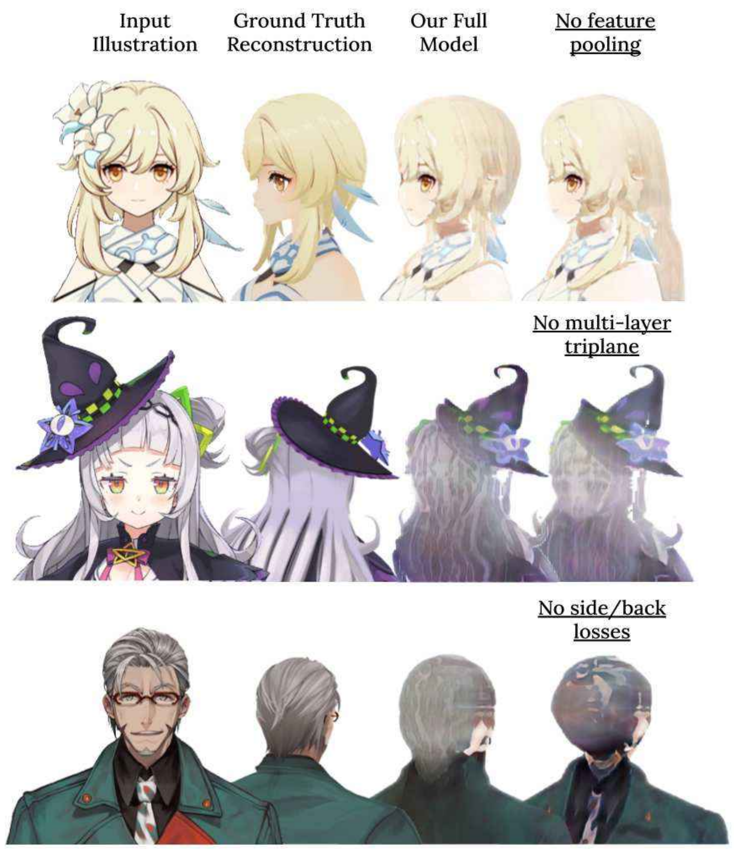


Figure 4.6: 3D reconstruction module ablations. Row 1: feature pooling helps distribute information across the triplane, and correctly structures the hair as short. Rows 2+3: the multi-layer triplane helps disambiguate the front/back, and side/back-view losses significantly improve both geometry and texture. (Art attributions in suppl.)

3D assets, the quality of details and geometry does not match up to PAniC-3D.

Ablations

From Tab. 4.3 and Fig. 4.6, we conclude that our architecture decisions improve both the qualitative and quantitative performance of our reconstruction system. It is shown that feature pooling is able to propagate information along triplane axes to improve generated geometries, the multi-layer triplane adds additional model capacity to further disambiguate locations for features, and the addition of fixed side/back-view losses significantly improves geometry and texture.

Illustration-to-Render Results

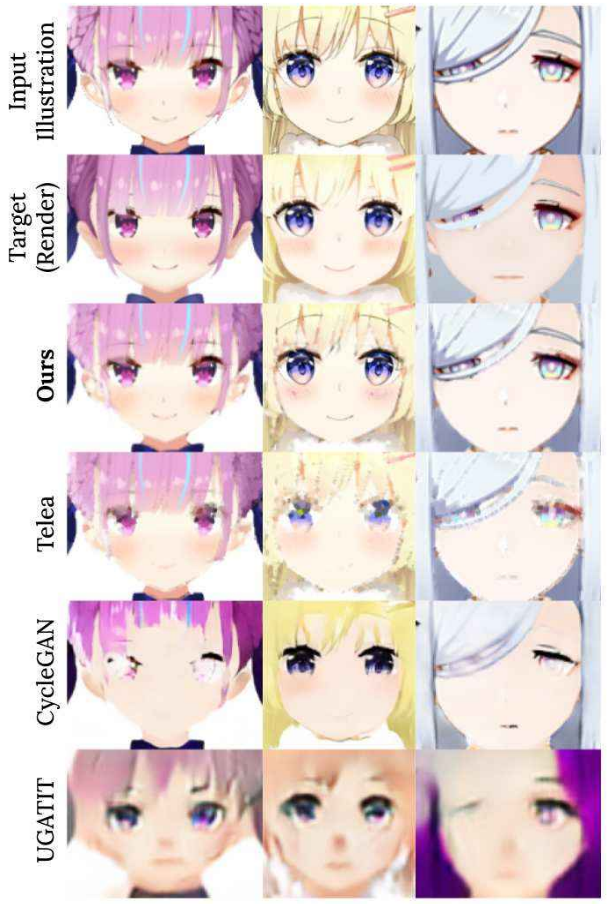
As shown in Tab. 4.4 and Fig. 4.5, our illustration-to-render model was able to effectively remove contours that would not appear in renders. We evaluate in Tab. 4.4 how closely the translated illustrations match their rendered counterpart from the AnimeRecon benchmark (restricted to the ROI annotation), and find that we are able to significantly outperform naive inpainting [113] as well as off-the-shelf image-to-image translation systems [58, 140]. The others struggled to retain key semantic structures like eyes/mouths, and often failed to retain the original identity.

In Fig. 4.5, it is shown that removing lines indeed alleviates the effects of domain transfer between illustrations and their output predicted radiance field; artifacts like extra contours and bands along shoulders are significantly reduced using our line removal strategy.

4.5 Limitations & Future Work

From the figures comparing the ground-truth 3D model renders and generated radiance fields, it is evident that this task is incredibly difficult. Although PAniC-3D performs better than other baseline methods, there is still a large gap in quality between our reconstructions and real character assets; this is still particularly evident for the hind occluded regions, as well as the areas around the ears connecting the front and back. Our model is usually able to prevent faces from appearing on the backside, but still largely relies on copying the visible portions of accessories to cover the occluded parts. In the future, we may look into ways of decomposing the radiance field in an object-centric manner, and generate accessories through a separate more expressive process. An obvious extension of this work would be to model full-body characters and exploit more opportunities given by the Vroid dataset, which supports blendshape facial expressions, hair physics, full-body rigging, and more.

In conclusion, we propose PAniC-3D, a system to reconstruct stylized 3D character heads directly from illustrated portraits of anime characters. Our anime-style domain poses unique challenges to single-view reconstruction when compared to natural images of human heads, such as hair and accessories with more complex and diverse geometry, and non-photorealistic contour lines. Furthermore, there is a lack of both 3D model and portrait illustration data suitable to train and evaluate this ambiguous stylized reconstruction task. Facing these challenges, our proposed PAniC-3D architecture crosses the illustration-to-3D domain gap with a line-filling model, and represents sophisticated geometries with a volumetric radiance field. We train our system with two large new datasets (11.2k Vroid 3D models, 1k Vtuber portrait illustrations), and evaluate on a novel AnimeRecon benchmark of illustration-to-3D pairs. PAniC-3D significantly outperforms baseline methods, and can generate plausible fully-textured geometries from a single input drawing. We hope that our proposed system and provided datasets may help establish the stylized reconstruction task for future work in both the academic community and the entertainment industry.



	LPIPS↓	CLIP↑	PSNR↑
Ours	**18.26**	**94.97**	**16.96**
Telea [113]	23.91	93.88	16.67
CycleGAN [140]	21.39	93.81	15.23
UGATIT [58]	39.64	85.48	13.18

Table 4.4: Illustration-to-render results. Over our AnimeRecon benchmark, we calculate perceptual and color metrics between translated illustrations and their corresponding 3D renders, restricted to the ROI annotations. Our method crosses the illustration-render domain gap better than both naive inpainting (which struggles to retain facial features) and deep image2image models (which fail to retain identity). LPIPS is scaled by 1e2.

Bibliography

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. “Image2stylegan: How to embed images into the stylegan latent space?” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4432–4441.

[2] Nandaka et al. *PixivUtil2*. `https://github.com/Nandaka/PixivUtil2`. 2021.

[3] William Falcon et al. “PyTorch Lightning”. In: 3 (2019).

[4] Mykhaylo Andriluka et al. “2D Human Pose Estimation: New Benchmark and State of the Art Analysis”. In: *IEEE CVPR (CVPR)*. June 2014.

[5] Anonymous, Danbooru community, and Gwern Branwen. *Danbooru2020: A Large-Scale Crowdsourced and Tagged Anime Illustration Dataset*. `https://www.gwern.net/Danbooru2020`. dataset. Jan. 2021. URL: `https://www.gwern.net/Danbooru2020`.

[6] Matthew Baas. *Danbooru2018 pretrained resnet models for PyTorch*. `https://rf5.github.io`. pretrained model. July 2019. URL: `https://rf5.github.io/2019/07/08/danbuuro-pretrained.html`.

[7] Wenbo Bao et al. “Depth-aware video frame interpolation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3703–3712.

[8] Wenbo Bao et al. “Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement”. In: *IEEE transactions on pattern analysis and machine intelligence* (2019).

[9] Mikhail Bessmeltsev and Justin Solomon. “Vectorization of line drawings via polyvector fields”. In: *ACM Transactions on Graphics (TOG)* 38.1 (2019), pp. 1–12.

[10] Yochai Blau and Tomer Michaeli. “The perception-distortion tradeoff”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6228–6237.

[11] Gwern Branwen, Anonymous, and Danbooru Community. *Danbooru2019 Portraits: A Large-Scale Anime Head Illustration Dataset*. dataset. Mar. 2019. URL: `https://www.gwern.net/Crops#danbooru2019-portraits`.

[12] Justin Brooks. *COCO Annotator*. `https://github.com/jsbroks/coco-annotator/`. 2019.

[13] Philip Buchanan, Ramakrishnan Mukundan, and Michael Doggett. “Automatic single-view character model reconstruction”. In: *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*. 2013, pp. 5–14.

[14] Lars Buitinck et al. “API design for machine learning software: experiences from the scikit-learn project”. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[15] Kaidi Cao, Jing Liao, and Lu Yuan. “Carigans: Unpaired photo-to-caricature translation”. In: *arXiv preprint arXiv:1811.00222* (2018).

[16] Thanh-Tung Cao et al. “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*. 2010, pp. 83–90.

[17] Zhe Cao et al. “OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields”. In: *IEEE transactions on pattern analysis and machine intelligence* 43.1 (2019), pp. 172–186.

[18] Leonardo Carvalho, Ricardo Marroquim, and Emilio Vital Brazil. “DiLight: Digital light table–Inbetweening for 2D animations using guidelines”. In: *Computers & Graphics* 65 (2017), pp. 31–44.

[19] Evan Casey, Víctor Pérez, and Zhuoru Li. “The Animation Transformer: Visual Correspondence via Segment Matching”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 11323–11332.

[20] Eric R Chan et al. “Efficient geometry-aware 3D generative adversarial networks”. In: *CVPR* (2022).

[21] Eric R Chan et al. “pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis”. In: *CVPR*. 2021.

[22] Liang-Chieh Chen et al. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).

[23] Shuhong Chen and Matthias Zwicker. “Improving the Perceptual Quality of 2D Animation Interpolation”. In: *arXiv preprint arXiv:2111.12792* (2021).

[24] Shuhong Chen and Matthias Zwicker. “Transfer Learning for Pose Estimation of Illustrated Characters”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 793–802.

[25] Myungsub Choi et al. “Channel attention is all you need for video frame interpolation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 10663–10671.

[26] Boris Dalstein, Rémi Ronfard, and Michiel Van De Panne. “Vector graphics animation with time-varying topology”. In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), pp. 1–12.

[27] Hal Daume. *Domain adaptation vs. transfer learning*. Nov. 2007. URL: `https://nlpers.blogspot.com/2007/11/domain-adaptation-vs-transfer-learning.html`.

[28] Carl Doersch and Andrew Zisserman. “Sim2real transfer learning for 3D human pose estimation: motion to the rescue”. In: *arXiv preprint arXiv:1907.02499* (2019).

[29] Szabolcs Dombi. “ModernGL, high performance python bindings for OpenGL 3.3+”. In: *GitHub repository* (May 1, 2020).

[30] Haoye Dong et al. “Towards multi-pose guided virtual try-on network”. In: *Proceedings of the IEEE/CVF ICCV*. 2019, pp. 9026–9035.

[31] Marek Dvorožňák et al. “Monster mash: a single-view approach to casual 3D modeling and animation”. In: *ACM Transactions on Graphics (TOG)* 39.6 (2020), pp. 1–12.

[32] Roy Or-El et al. “StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation”. In: *CVPR* (2022).

[33] Hao-Shu Fang et al. “Rmpe: Regional multi-person pose estimation”. In: *Proceedings of the IEEE ICCV*. 2017, pp. 2334–2343.

[34] Pedro F Felzenszwalb and Daniel P Huttenlocher. “Distance transforms of sampled functions”. In: *Theory of computing* 8.1 (2012), pp. 415–428.

[35] Einar Forselv. “moderngl-window, a cross-platform windowing/utility library for ModernGL”. In: *GitHub repository* (May 1, 2020).

[36] Damien Fourure et al. “Residual conv-deconv grid network for semantic segmentation”. In: *arXiv preprint arXiv:1707.07958* (2017).

[37] Oran Gafni, Oron Ashual, and Lior Wolf. “Single-Shot Freestyle Dance Reenactment”. In: *arXiv preprint arXiv:2012.01158* (2020).

[38] Yuying Ge et al. “Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images”. In: *Proceedings of the IEEE/CVF CVPR*. 2019, pp. 5337–5345.

[39] Robert Geirhos et al. “ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness”. In: *arXiv preprint arXiv:1811.12231* (2018).

[40] Georgia Gkioxari and Jitendra Malik. “JJ: Mesh r-cnn”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019.

[41] The Khronos Group. “GLTF Specification”. In: (2022).

[42] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. “Densepose: Dense human pose estimation in the wild”. In: *Proceedings of the IEEE CVPR*. 2018, pp. 7297–7306.

[43] Koichi Hamada et al. “Full-body high-resolution anime generation with progressive structure-conditional generative adversarial networks”. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0.

[44] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE CVPR*. 2016, pp. 770–778.

[45] Kaiming He et al. “Mask r-cnn”. In: *Proceedings of the IEEE ICCV*. 2017, pp. 2961–2969.

[46] Judy Hoffman et al. “Cycada: Cycle-consistent adversarial domain adaptation”. In: *ICML*. PMLR. 2018, pp. 1989–1998.

[47] Zeng Huang et al. “Arch: Animatable reconstruction of clothed humans”. In: *Proceedings of the IEEE/CVF CVPR*. 2020, pp. 3093–3102.

[48] Zhewei Huang et al. “RIFE: Real-Time Intermediate Flow Estimation for Video Frame Interpolation”. In: *arXiv preprint arXiv:2011.06294* (2020).

[49] hysts. *Anime Face Detector*. `https://github.com/hysts/anime-face-detector`. 2021.

[50] Eddy Ilg et al. “Flownet 2.0: Evolution of optical flow estimation with deep networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2462–2470.

[51] Catalin Ionescu et al. “Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (July 2014), pp. 1325–1339.

[52] Andrew Jaegle et al. “Perceiver io: A general architecture for structured inputs & outputs”. In: *arXiv preprint arXiv:2107.14795* (2021).

[53] Huaizu Jiang et al. “Super slomo: High quality estimation of multiple intermediate frames for video interpolation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9000–9008.

[54] Jie Jiang, Hock Soon Seah, and Hong Ze Liew. “Stroke-Based Drawing and Inbetweening with Boundary Strokes”. In: *Computer Graphics Forum*. Vol. 41. 1. Wiley Online Library. 2022, pp. 257–269.

[55] Pramook Khungurn. “Talking Head Anime from a Single Image 2: More Expressive”. In: (2021).

[56] Pramook Khungurn and Derek Chou. “Pose estimation of anime/manga characters: a case for synthetic data”. In: *Proceedings of the 1st International Workshop on coMics ANalysis, Processing and Understanding* (2016).

[57] Pramook Khungurn and Derek Chou. “Pose estimation of anime/manga characters: a case for synthetic data”. In: (2016), pp. 1–6.

[58] Junho Kim et al. “U-GAT-IT: unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation”. In: *arXiv preprint arXiv:1907.10830* (2019).

[59] Kangyeol Kim et al. “AnimeCeleb: Large-Scale Animation CelebHeads Dataset for Head Reenactment”. In: (2022).

[60] Kichang Kim, Rachmadani Haryono, and Arthur Guo. *DeepDanbooru*. `https://github.com/KichangKim/DeepDanbooru`. 2019.

[61] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[62] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[63] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[64] Christoph Lassner et al. "Unite the people: Closing the loop between 3d and 2d human representations". In: *Proceedings of the IEEE CVPR*. 2017, pp. 6050–6059.

[65] Chen Li and Gim Hee Lee. "From Synthetic to Real: Unsupervised Domain Adaptation for Animal Pose Estimation". In: *arXiv preprint arXiv:2103.14843* (2021).

[66] Chengze Li, Xueting Liu, and Tien-Tsin Wong. "Deep extraction of manga structural lines". In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), pp. 1–12.

[67] Jerry Li. *Pixiv Dataset*. `https://github.com/jerryli27/pixiv_dataset`. 2019.

[68] Jerry Li and Tazik Shahjahan. *AniSeg*. `https://github.com/jerryli27/AniSeg`. 2020.

[69] Tzu-Mao Li et al. "Differentiable vector graphics rasterization for editing and learning". In: *ACM Transactions on Graphics (TOG)* 39.6 (2020), pp. 1–15.

[70] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[71] Zuzeng Lin et al. "Collaborative Neural Rendering using Anime Character Sheets". In: *arXiv preprint arXiv:2207.05378* (2022).

[72] Liang Liu et al. "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*. 2020, pp. 6489–6498.

[73] Shichen Liu et al. "Soft rasterizer: A differentiable renderer for image-based 3d reasoning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7708–7717.

[74] Zelun Luo et al. "Label efficient learning of transferable representations across domains and tasks". In: *arXiv preprint arXiv:1712.00123* (2017).

[75] Akinobu Maejima et al. "Anime Character Colorization using Few-shot Learning". In: *SIGGRAPH Asia 2021 Technical Communications*. 2021, pp. 1–4.

[76] Simone Meyer et al. "Phase-based frame interpolation for video". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1410–1418.

[77] Simone Meyer et al. "Phasenet for video frame interpolation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 498–507.

[78] Jiaxu Miao et al. "Pose-guided feature alignment for occluded person re-identification". In: *Proceedings of the IEEE/CVF ICCV*. 2019, pp. 542–551.

[79] Haoran Mo et al. "General Virtual Sketching Framework for Vector Line Art". In: *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2021)* 40.4 (2021), 51:1–51:14.

[80] Gyeongsik Moon and Kyoung Mu Lee. "I2L-MeshNet: Image-to-lixel prediction network for accurate 3D human pose and mesh estimation from a single RGB image". In: *arXiv preprint arXiv:2008.03713* (2020).

[81] Tewodros Legesse Munea et al. “The progress of human pose estimation: a survey and taxonomy of models applied in 2D human pose estimation”. In: *IEEE Access* 8 (2020), pp. 133330–133348.

[82] Rei Narita, Keigo Hirakawa, and Kiyoharu Aizawa. “Optical Flow Based Line Drawing Frame Interpolation Using Distance Transform to Support Inbetweenings”. In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 4200–4204.

[83] Simon Niklaus and Feng Liu. “Softmax splatting for video frame interpolation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5437–5446.

[84] Simon Niklaus, Long Mai, and Feng Liu. “Video frame interpolation via adaptive convolution”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 670–679.

[85] Simon Niklaus, Long Mai, and Feng Liu. “Video frame interpolation via adaptive separable convolution”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 261–270.

[86] Ryosuke Okuta et al. “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.

[87] “PanoHead: Geometry-aware 3D Full-head Generative Adversarial Networks”. In: (2022).

[88] Junheum Park, Chul Lee, and Chang-Su Kim. “Asymmetric Bilateral Motion Estimation for Video Frame Interpolation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 14539–14548.

[89] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[90] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019), pp. 8026–8037.

[91] Omid Poursaeed et al. “Neural puppet: Generative layered cartoon characters”. In: *Proceedings of the IEEE/CVF WACV*. 2020, pp. 3346–3356.

[92] Zhang Qian et al. “Line Art Correlation Matching Network for Automatic Animation Colorization”. In: *arXiv e-prints* (2020), arXiv–2004.

[93] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763.

[94] Joseph Redmon and Ali Farhadi. “Yolov3: An incremental improvement”. In: *arXiv preprint arXiv:1804.02767* (2018).

[95] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *arXiv preprint arXiv:1506.01497* (2015).

[96] E. Riba et al. “A survey on Kornia: an Open Source Differentiable Computer Vision Library for PyTorch”. In: 2020.

[97] Daniel Roich et al. “Pivotal tuning for latent-based editing of real images”. In: *ACM Transactions on Graphics (TOG)* 42.1 (2022), pp. 1–13.

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

[99] Shunsuke Saito et al. “Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2304–2314.

[100] Mehul P Sampat et al. “Complex wavelet structural similarity: A new image similarity index”. In: *IEEE transactions on image processing* 18.11 (2009), pp. 2385–2401.

[101] Philip J Schneider. “An algorithm for automatically fitting digitized curves”. In: *Graphics gems* 1 (1990), pp. 612–626.

[102] Jamie Shotton et al. “Real-time human pose recognition in parts from single depth images”. In: *CVPR 2011*. Ieee. 2011, pp. 1297–1304.

[103] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. “Mastering sketching: adversarial augmentation for structured prediction”. In: *ACM Transactions on Graphics (TOG)* 37.1 (2018), pp. 1–13.

[104] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. “Mastering sketching: adversarial augmentation for structured prediction”. In: *ACM Transactions on Graphics (TOG)* 37.1 (2018), pp. 1–13.

[105] Edgar Simo-Serra et al. “Learning to simplify: fully convolutional networks for rough sketch cleanup”. In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), pp. 1–11.

[106] Li Siyao et al. “Deep Animation Video Interpolation in the Wild”. In: *arXiv preprint arXiv:2104.02495* (2021).

[107] Li Siyao et al. “Deep Geometrized Cartoon Line Inbetweening”. In: *ICCV* (2023).

[108] Tomáš Souček and Jakub Lokoč. “TransNet V2: An effective deep network architecture for fast shot transition detection”. In: *arXiv preprint arXiv:2008.04838* (2020).

[109] Baochen Sun and Kate Saenko. “Deep coral: Correlation alignment for deep domain adaptation”. In: *European conference on computer vision*. Springer. 2016, pp. 443–450.

[110] Deqing Sun et al. “Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8934–8943.

[111] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE CVPR*. 2015, pp. 1–9.

[112] Zachary Teed and Jia Deng. “Raft: Recurrent all-pairs field transforms for optical flow”. In: *European conference on computer vision*. Springer. 2020, pp. 402–419.

[113] Alexandru Telea. “An image inpainting technique based on the fast marching method”. In: *Journal of graphics tools* 9.1 (2004), pp. 23–34.

[114] Zhi Tian, Chunhua Shen, and Hao Chen. “Conditional convolutions for instance segmentation”. In: *arXiv preprint arXiv:2003.05664* (2020).

[115] Eric Tzeng et al. “Adversarial discriminative domain adaptation”. In: *Proceedings of the IEEE CVPR*. 2017, pp. 7167–7176.

[116] Gul Varol et al. “Learning from synthetic humans”. In: *Proceedings of the IEEE CVPR*. 2017, pp. 109–117.

[117] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: `10.1038/s41592-019-0686-2`.

[118] VRoid. “VRM Specification”. In: (2022).

[119] Mei Wang and Weihong Deng. “Deep visual domain adaptation: A survey”. In: *Neurocomputing* 312 (2018), pp. 135–153.

[120] Xinlong Wang et al. “SOLOv2: Dynamic and fast instance segmentation”. In: *Advances in Neural Information Processing Systems* (2020).

[121] Yaqing Wang et al. “Generalizing from a few examples: A survey on few-shot learning”. In: *ACM Computing Surveys (CSUR)* 53.3 (2020), pp. 1–34.

[122] Martin Weber. “AutoTrace - converts bitmap to vector graphics”. In: (). URL: `https://autotrace.sourceforge.net/`.

[123] Brian Whited et al. “Betweenit: An interactive tool for tight inbetweening”. In: *Computer Graphics Forum*. Vol. 29. 2. Wiley Online Library. 2010, pp. 605–614.

[124] Holger Winnemöller, Jan Eric Kyprianidis, and Sven C Olsen. “XDoG: An eXtended difference-of-Gaussians compendium including advanced image stylization”. In: *Computers & Graphics* 36.6 (2012), pp. 740–753.

[125] Yuxin Wu et al. *Detectron2*. `https://github.com/facebookresearch/detectron2`. 2019.

[126] Xiangyu Xu et al. “3d human shape and pose from a single low-resolution image with self-supervised learning”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 284–300.

[127] Xiangyu Xu et al. “Quadratic video interpolation”. In: *arXiv preprint arXiv:1911.00627* (2019).

[128] Zhan Xu et al. “Rignet: Neural rigging for articulated characters”. In: *arXiv preprint arXiv:2005.00559* (2020).

[129] Yang Xue et al. “GIRAFFE HD: A High-Resolution 3D-aware Generative Model”. In: *CVPR*. 2022.

[130] Yuichi Yagi. “A filter based approach for inbetweening”. In: *arXiv preprint arXiv:1706.03497* (2017).

[131] Wenwu Yang. “Context-aware computer aided inbetweening”. In: *IEEE transactions on visualization and computer graphics* 24.2 (2017), pp. 1049–1062.

[132] Wenwu Yang, Jieqing Feng, and Xun Wang. “Structure Preserving Manipulation and Interpolation for Multi-element 2D Shapes”. In: *Computer Graphics Forum* 31 (2012).

[133] Wenwu Yang et al. “FTP-SC: Fuzzy Topology Preserving Stroke Correspondence”. In: *Computer Graphics Forum* 37 (2018).

[134] Alex Yu et al. “pixelnerf: Neural radiance fields from one or few images”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4578–4587.

[135] Lvmin Zhang, Yi Ji, and Chunping Liu. “DanbooRegion: An Illustration Region Dataset.” In: *ECCV (13)*. 2020, pp. 137–154.

[136] Lvmin Zhang et al. “Generating Digital Painting Lighting Effects via RGB-space Geometry”. In: *ACM Transactions on Graphics (TOG)* 39.2 (2020), pp. 1–13.

[137] Lvmin Zhang et al. “Two-stage sketch colorization”. In: *ACM Transactions on Graphics (TOG)* 37.6 (2018), pp. 1–14.

[138] Richard Zhang et al. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

[139] Yi Zheng et al. “Cartoon Face Recognition: A Benchmark Dataset”. In: *Proceedings of the 28th ACM International Conference on Multimedia*. 2020, pp. 2264–2272.

[140] Jun-Yan Zhu et al. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.

[141] zymk9. *Yet-Another-Anime-Segmenter*. `https://github.com/zymk9/Yet-Another-Anime-Segmenter`. 2020.