AnimeRun: 2D Animation Visual Correspondence from Open Source 3D Movies
Supplementary File

Li Siyao[1], **Yuhang Li**[2,3], **Bo Li**[1], **Chao Dong**[2,4], **Ziwei Liu**[1], **Chen Change Loy**[1]✉
[1]S-Lab, Nanyang Technological University
[2]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences [4]Shanghai AI Laboratory
{siyao002, libo0013, ziwei.liu, ccloy}@ntu.edu.sg
{yh.li5, chao.dong}@siat.ac.cn

Abstract

In this supplementary file, we introduce some implementation details in our work and discuss the limitations and potential social impacts of our proposed dataset. We also present a demo video at https://lisiyao21.github.io/projects/AnimeRun to illustrate the dataset more comprehensively. A datasheet is provided along with this supplementary file.

1 Implementation Details

1.1 Color Augmentation

An object in the frames of AnimeRun is rendered into flattened colors defined by the materials of this object under "*FLAT*" lighting choice of *Workbench* engine in Blender. To make the luminance/color distribution of AnimeRun closer to the that of real-world cartoons, we extract the representative colors from ATD-12K [15] dataset and randomly assign these colors to existing materials for augmentation. First, we manually cluster frames of ATD-12K to different groups based on subjective judgement on the analogy of styles and colorization, and acquire 46 groups for Disney styles and 26 for Japanese styles. Then, for each group g, we record all RGB colors that appear over a frequency of 0.05% $\{c_i\}^g$ as color candidates for this group as well as corresponding appearance times $\{N_i\}^g$. To reduce the storage for different RGB combinations, we quantize each color channel to 16 values during counting. When applying color aggregation to a film cut in AnimeRun, we select a group g and sample colors from $\{c_i\}^g$ to materials by using `random.choices` function of Python with corresponding sampling weights of $\ln(N_i^g/10)$. Examples of augmented frames and corresponding color sources are shown in Figure 1.

Augmented frame Samples of corresponding color source

Figure 1: **Examples of color augmented frames and the corresponding color source.**

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks.

1.2 Benchmark Experiments

Optical Flow. In our experiment, we finetune and evaluate four optical flow methods PWC-Net [17], RAFT [18], GMA [6] and GMFlow [19]. Since the officially released code of PWC-Net is incompatible to our environment settings, we use the implementation in [12], which achieves comparable scores on the MPI-Sintel benchmark [2]. For other three methods, we exploit the official model and pretrained weights as baselines. Similar to the training stage of "`sintel`" in [18], we finetune these networks on different datasets mixed with FlyingThings3D [10], with a mixing rate of around 10:1. During finetuning, images are cropped into the size of 368×768 when finetuning on Sintel and AnimeRun, and 480×480 for CreativeFlow+ [14]. Scale and color augmentations are applied as the same as [18]. To train PWC-Net, we adopt L_1 loss between the final predicted flow and the ground truth, while for other three methods, we also supervise the intermediate flow after each update using the sequential loss defined in [18]. We use AdamW optimizer [9] with weight decay of 5×10^{-5}, and apply gradient clipping to range $[-1, 1]$. All flow networks are finetuned by $20,000$ iterations with a batch size of 12 on two NVIDIA V100 GPUs, with initial learning rate of 10^{-4}. During evaluation, for RAFT [18] and GMA [6], we test after 32 flow updates .

Segment Matching. Since the official Animation Transformer (AnT) is not publicly available, we implement a similar version referring to the description in [3]. AnT is composited of a CNN-based segment descriptor, an MLP-based positional encoder, and a GNN-based feature aggregator realized in Transformer structure as SuperGlue [13], where the final mapping is computed from the segment-wise similarity matrix of aggregated features. Given a pair of input frames $I^0, I^1 \in \mathbb{R}^{H \times W \times 3}$ and the corresponding segmentation labels $S^0, S^1 \in \mathbb{R}^{H \times W}$, we first apply a three-layer CNN to extract the feature of each frame and apply super-pixel pooling [8] on the features to obtain region-wise feature $F^0 \in \mathbb{R}^{N \times D}$ and $F^1 \in \mathbb{R}^{M \times D}$ each segment S^0 and S^1, where N and M are the segment numbers of S^0 and S^1 respectively, and D is the channel number (128 in our experiment). Second, for each segment S_i^t $(t = 0, 1)$, we feed $[x_{min}, y_{min}, x_{max}, y_{max}]$, which represent the boundary of this segment, to an MPL to obtain positional encoding $P_i^t \in \mathbb{R}^D$ for this segment. Third, we adopt a Transformer $\mathcal{T}$ implemented in [13] to aggregate the segment features as $\hat{F}^0, \hat{F}^1 = \mathcal{T}(F^0 + P^0, F^1 + P^1)$. Next, we compute the correlation between features of S^0 and S^1 to be the similarity matrix $M = corr(\hat{F}^0, \hat{F}^1) \in \mathbb{R}^{N \times M}$. Finally, a softmax layer is applied alongside the rows of to obtain $\hat{M} = softmax(M, 1)$. We use weighted cross-entropy loss $\mathcal{L} = \sum_{i=0...N-1} w_i \cdot CE(\hat{M}_i, \boldsymbol{m}_{0\to1}[i])$ to train the network, where $w_i = |S^0 = i|/(H \times W)$. During training, images are randomly cropped to 368×368. We use Adam optimizer [7] with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ to train the networks for $3,000$ iterations. The learning rate is set to 10^{-4}. The whole training process is conducted on one NVIDIA V100 GPU. Code is available at https://github.com/lisiyao21/AnimeRun/.

2 Limitations

In this work, we present a new dataset, *AnimeRun*, which to our knowledge is the first 2D animation visual correspondence dataset composed from full scenes of industry-level 3D movies. Compared to most existing datasets [1, 5, 11, 2, 4, 10, 16] that are built for natural scenarios, our dataset features the style of 2D cartoons, including explicit contour lines and flat color pieces. Meanwhile, in contrast to previous datasets [14] designed for 2D animations, AnimeRun not only resembles the real anime more in image composition, but also possesses richer and more complex motion patterns.

Besides the advantages, it is worthy to note that for real cartoon films, different producers have their unique painting styles on processing details (*e.g.*, brightness and color distributions), and various special effects including different 2D shading and lighting. The cartoon style in AnimeRun is designed to standard capture the generic visual style of 2D animation instead of producer- or studio-specific styles. Therefore, compared to real 2D cartoon films, the colored frames in AnimeRun are more like an mid product after colorization stage of 2D animation pipeline without special effects as shown in Figure 2. However, lacking of such effect does not mean AnimeRun cannot generalize to finished animations – we show its usefulness for real cartoon frame interpolation in the Discussion section of the main paper.

If there is need for the painting style of a particular studio, one can add special effects into our data in the pre-processing for more robust industrial usage.

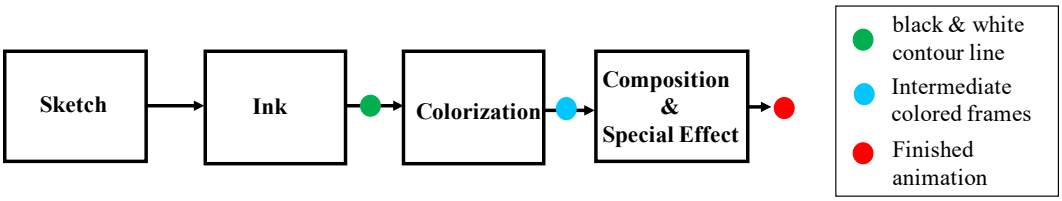

Figure 2: **Simplified diagram of making 2D animation.** In AnimeRun data, contour lines in gray scale are targeted towards -intermediate animation at the stage marked as green dot. While the colored frames resembles those marked as blue dot after the colorization step. Special effects, including shading and 2D style lighting, which are represented as the red dot are not included in AnimeRun.

3 Social Impacts

As to potential social impacts, since our data are made from public and open-source movies, AnimeRun is not subject to ethical risks, nor are its contents causing public discomfort. We believe the proposed dataset could facilitate some crucial steps in the processing of 2D animation, benefiting the development of relevant industries.

4 Demo Video

We present a video to demonstrate the content of our dataset and show its advantages compared to existing correspondence datasets. We display the comparison between the original 3D movies and the converted 2D cartoons side by side, and present the cartoon frames and the correspondence labels to show both the rich image components and complex motion patterns of our data.

References

[1] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92:1–31, 2011. 2

[2] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012. 2

[3] Evan Casey, Víctor Pérez, and Zhuoru Li. The animation transformer: Visual correspondence via segment matching. In *ICCV*, 2021. 2

[4] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 2

[5] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 2

[6] Shihao Jiang, Dylan Campbell, Yao Lu, Hongdong Li, and Richard Hartley. Learning to estimate hidden motions with global motion aggregation. In *ICCV*, 2021. 2

[7] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014. 2

[8] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE TPAMI*, 38(10):2024–2039, 2015. 2

[9] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[10] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016. 2

[11] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, 2015. 2

[12] Simon Niklaus. A reimplementation of PWC-Net using PyTorch. https://github.com/sniklaus/pytorch-pwc, 2018. 2

[13] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 2

[14] Maria Shugrina, Ziheng Liang, Amlan Kar, Jiaman Li, Angad Singh, Karan Singh, and Sanja Fidler. Creative flow+ dataset. In *CVPR*, 2019. 2

[15] Li Siyao, Shiyu Zhao, Weijiang Yu, Wenxiu Sun, Dimitris Metaxas, Chen Change Loy, and Ziwei Liu. Deep animation video interpolation in the wild. In *CVPR*, 2021. 1

[16] Deqing Sun, Daniel Vlasic, Charles Herrmann, Varun Jampani, Michael Krainin, Huiwen Chang, Ramin Zabih, William T Freeman, and Ce Liu. Autoflow: Learning a better training set for optical flow. In *CVPR*, 2021. 2

[17] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 2

[18] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 2

[19] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *CVPR*, 2022. 2

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Supplementary file.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Supplementary file.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] In datasheet.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]