Creative Flow+ Dataset
Errata and Data Details

Maria Shugrina[1,2,3] www.shumash.com
Ziheng Liang[1,4] zhliang@cs.ubc.ca
Amlan Kar[1,2] amlan@cs.toronto.edu
Jiaman Li[1,2] ljm@cs.toronto.edu
Angad Singh[1,5] angad.singh@alum.utoronto.ca
Karan Singh[1] karan@dgp.toronto.edu
Sanja Fidler[1,2,3] fidler@cs.toronto.edu

[1]University of Toronto [2]Vector Institute [3]NVIDIA
[4] University of British Columbia [5] Evertz Microsystems

Overview

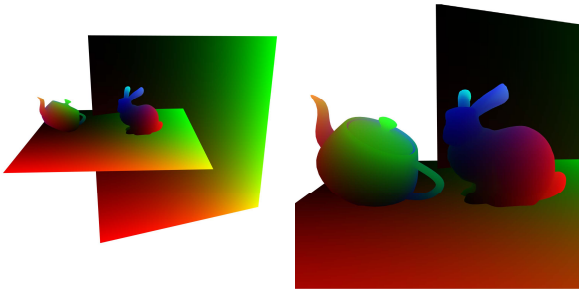
Creative Flow+ Dataset is large and was computationally intensive to generate. We have done our best to ensure the integrity of our data, but some minor things went wrong (§1). We encourage all users of our data to at least read (§1). Details about included ground truth (§2), data consistency checks (§3), a newly discovered bug in blender (§4), and our decision not to withhold the test set for a benchmark (§5) are included below. Here are links to data and code.

1. Data Errata Summary

While rendering our training set, we discovered a bug in Blender affecting under 5% of our optical flow data. Erroneous data is excluded from our downloads and has only minor effects on the results we reported in the paper (§4).

Due to an operational error, for an unknown fraction of our *training* set, the color randomization in the stylized renderings (but not the style itself) abruptly switches at frame number 5 (i.e. `frame000006` and following frames have different color settings from all previous 1-based frames in the decompressed dataset). We believe that for most tasks, including these frame pairs will actually make training more robust. However, if you wish to exclude this data, we suggest simply ignoring this particular frame pair for the entire training set (about 2% of the data).

Back flow for the last frame in every sequence has been mistakenly omitted from the compressed dataset. As this supplemental metadata is not often required, we will not be correcting this mistake.



(a) Frame I_0 (b) Frame I_i

Figure 1: **Correspondence** renderings allow identifying closest matching points across widely disparate views by color, when coupled with the object id maps.

2. Data Details

2.1. Half Resolution Downloads

While compiling Creative Flow+ Dataset we have experienced the strain this scale of data can put on research infrastructure. To make our dataset more attainable, we are also providing lower resolution (750x750) downloads at half of the original dimensions (1500x1500). To obtain this lower resolution data, all compressed renderings were downsampled using `ffmpeg` command line utility using Lanczos resampling with default parameters. All the metadata was re-rendered from scratch at a lower resolution using Blender.

2.2. Sequence Lists

Sequence lists help navigate the actual frames in our dataset, e.g. by using our python utilities. The test set se-

quences contain more than 40K frames, some with no motion due to the character or object moving out of camera view. In order to obtain the original 10K test set, we subsampled these frames and ensured that they include motion. We provide both subsampled and full sequence lists. Prior to rendering the training set, we trimmed sequences to only parts containing motion, so no frame filtering is necessary.

2.3. About Occlusions

Just like the MPI Sintel dataset [3], we mark the pixel $\bar{x}$ in frame I_i as occluded, if optical flow $F_i(\bar{x})$ differs from back flow B_{i+1} interpolated for location $\bar{x}+F_i(\bar{x})$ by some threshold τ. For our resolution of 1500x1500 we used $\tau = 0.5$, larger than that for MPI Sintel dataset[1], to reflect our larger frame dimensions.

2.4. About Correspondences

To our knowledge, ours is the only dataset to render dense correspondences as color persisting across frames. This allows looking for closest matching color to find correspondences in widely disparate views, where tracking using optical flow could become impossible due to occlusions (e.g. Fig.1). Robustly finding correspondences in widely disparate views across different drawing styles can be useful for an array of creative applications, e.g. involving image morphing and registration [9], autocompletion [11] and frame interpolation [10, 1] of animation. Our data allows formulating the correspondence problem not as finding "the one and only matching point", but as finding "the closest matching point" on the underlying 3D object across 2 views.

To render correspondences, we embed the mesh of each object in the reference pose (first frame) into a coordinate system defined by its bounding box where each side spans the range of $[0..1]$. The xyz location of each mesh vertex in this coordinate system also becomes its RGB color. This color assignment persists across the entire sequence and all object deformations. In order to find the best corresponding point in frame I_i for a point in frame I_0, one has to look for the closest matching color in correspondence renderings, *where the object id maps agree* (correspondence colors repeat across objects).

This method is not perfect. The color distance of vertices depends on the initial pose and orientation of the object, and is not normalized across different objects. However, this is a fast and compact way of encoding correspondences across the entire moving sequence.

We provide compressed correspondences encoded in an `mp4` video as a part of the core metadata download. We have not quantified how much precision is lost due to this compression, and therefore recommend downloading lossless correspondences if you are planning to heavily rely on the precision of this data.

[1]We thank the authors of [3] for sharing details about their pipeline

2.5. About Depth

We provide depth maps, i.e. distance from the image plane, in two different formats. The first format is depth rendered as images, with white and black representing the minimum and maximum depth values, respectively, across the entire sequence. The actual unnormalized values in this range are included in an accompanying text file. We also provide depth as unnormalized `numpy` arrays (better precision) in a different, larger, download. Note that our sequences have not been normalized to have consistent depth.

2.6. About Compressed Downloads

The size of our dataset makes uncompressed distribution prohibitive. Most renders are provided as `mp4` files, including lossy normals, depth images and correspondences. Object id images are simply zipped for every animated sequence. Flows, lossless depth, lossless correspondence and lossless normals are packed into special zips, one per sequence, which yield better compression in many cases.

Decompression utilities are provided in our github repo. Following decompression, flow is written as Middlebury `.flo` files, most renders as `.png` images and lossless depth as serialized `numpy` arrays. Utilities for reading this data are included in our python library.

3. Evaluating Data Consistency

While developing our pipeline we discovered that the correctness of ground truth renderings (e.g. optical flow) from Blender can be affected by various Blender settings, a behavior that is not well-documented or predictable. Unlike the authors of [3], we were not able to hand-curate our dataset due to its larger size (100x). Further, optical flow data cannot be easily checked for correctness. This prompted us to write consistency checks for our data. To our knowledge, ours is the first synthetic optical flow dataset that comes with such "sanity" checks, as we are the first to render correspondences (§2.4) that can be cross-checked with the flow values.

3.1. Sanity Computation

A "sane" pixel $\bar{x}$ in frame I_i should either be marked as occluded, or its object id label $O_i(\bar{x})$ and correspondence color $C_i(\bar{x})$ should agree with the next frame's object id label O_{i+1} and correspondence color C_{i+1} at location $\bar{x} + F_i(\bar{x})$, where F_i is flow. While object ids must agree exactly, we require correspondence color to be within $\beta = 4$ Euclidean distance in RGB ranging between 0 and 255. After rendering each sequence we run sanity checks for a sample of *foreground* pixels in a subset of the frames and compute overall "sanity", or the number of sane pixels divided by the number of tested pixels. These checks are approximate, where aliasing artifacts in correspondence

rendering (especially of meshes rendered further away from the camera), quantization artifacts at object boundaries in object id renderings and other rendering particulars could cause disagreement and impact the overall "sanity".

3.2. Sanity Results

Our pipeline was configured to only flag sequences with sanity values lower than 0.8, i.e. with more than 20% of pixels failing the test. We found that even these permissive settings can flag benign inconsistencies, for example in frames where the object rendering is small and there is a larger than $\tau = 4$ difference between adjacent pixels in correspondence renderings. However, these rough checks also led us to discover an important and much more subtle bug in Blender (next section). Following this discovery, we performed due diligence and computed per-frame sanity for 2000 random foreground pixels in *every frame* of every sequence. We make these values available in case some other inconsistencies may be discovered later.

4. Blender Bug

A debugging saga (§4.2) led us to discover a bug in the way Blender renders optical flow for clips that have animation of the *camera focal length*. We included focal length animation for a **small fraction of our sequences** to simulate the zoom effect, and thus a fraction of our optical flow ground truth is affected by this bug. We have filed a Bug Report (T69731) with Blender[2] and analyzed our data and results, reporting **minor** corrections (§4.1).

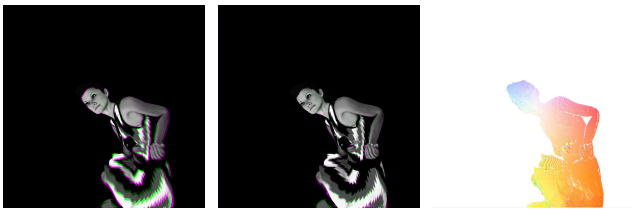
4.1. Implications for Data and Results

Training set: In the training set, the focal length Blender bug affected 104 out of 1379 Mixamo sequences and none of the other data. We are excluding optical flow and occlusions for these sequences in our downloads.

Test set: In the test set, the focal length Blender bug affected 19 out of 268 Mixamo sequences and none of the other data. This amounts to 475 frames out of 10,031 test set frames, or 4.7%. The results reported in Fig.7b of the paper were only slightly affected by excluding these frames from the evaluation, and all the conclusions in our original analysis still hold. Despite the small differences, we include the corrected results here for completeness (Tb.1). As with the training set, we are excluding optical flow and occlusions for the test set sequences affected by the bug.

4.2. Debugging Insane Clips

Debugging errors in data such as flow, which cannot be hand-checked manually, is difficult, and we include details in the case this is of interest. While most sanity failures resulted from simple misconfiguration or benign disagreement, 4% of our Mixamo training set failed for reasons that left us stumped: flow disagreed with correspondence colors for large areas of the frame and incorrect areas were marked as occluded. After trying various debugging approaches, we eventually decided to directly compute correspondence-based flow F_i^c by finding closest matching color in sequential correspondence renderings C_i, C_{i+1}. To our surprise, though noisy and time-consuming to compute, F_i^c resulted in superior alignment of frame I_{i+1} with the flow-morphed frame I_i than the original flow F_i for one of the failing sequences (Fig.2a,b). Visualization of the error between two flow fields $F_i^c - F_i$ revealed a rainbow-colored plot, indicating that the error vectors pointed radially out, mimicking the camera zoom present in this sequence. This finally led us to create a small one-triangle example that demonstrated the bug in the Blender pipeline. Note that camera motion does not result in incorrect output, only the animation of the camera focal length does.



(a) $W(I_i, F_i), I_{i+1}$ (b) $W(I_i, F_i^c), I_{i+1}$ (c) $F_i^c - F_i$

Figure 2: **Debugging:** Frame I_{i+1} overlaid with false color on the previous frame I_i warped **(a)** using original flow field $W(I_i, F_i)$ (clear mis-alignment) and **(b)** using correspondence-based flow $W(I_i, F_i^c)$ (better alignment). The difference in the two flow values **(c)** visualized using standard flow color (paper, Fig.7a inset) reveals structured error.

All sequences that failed the sanity checks for inexplicable reasons contained animation of the focal length. We were also able to catch additional buggy sequences not flagged by the randomized checks.

5. Why No Benchmark

At the onset of this project we intended to host a public benchmark for the optical flow task and to keep the test set private. However, as we expanded the scope to include other ground truth, such as normals and depth, it became clear that we cannot formulate the full range of tasks that can be tackled with the Creative Flow+ Dataset. Discussion with the members of the research community during our CVPR poster session reinforced this realization. We, therefore, make a decision to release the test set and to trust the integrity and creativity of researchers to use our data well.

[2]Note that this bug is very different from the bug corrected by the authors of [3] in a custom Blender build, which has since then been fixed in the Blender releases.

		Sintel	Creative Flow+									
				median		Styles				Speeds		
			All	All	FG	FG:flat	FG:toon	FG:tex.	FG:stylit	FG:1%	FG:1-3%	FG:3%
Horn-Schunck	[4]	9.64	8.34	3.49	12.17	11.70	11.12	13.83	12.18	3.45	17.23	60.03
Classic+NLfast	[7]	10.12	13.35	7.05	9.27	9.09	6.95	11.36	9.67	5.68	11.04	29.81
Brox2011	[2]	9.15	9.05	3.27	8.28	7.33	6.18	11.63	8.29	3.99	11.20	30.74
EpicFlow	[6]	6.29	64.31	10.46	14.80	9.42	6.78	11.47	23.81	11.20	15.97	36.99
DC Flow	[12]	5.12	41.08	3.37	11.21	7.69	9.32	12.57	13.39	3.90	17.98	44.76
PWC(chrs.)	[8]	-	66.71	40.98	22.73	41.20	10.85	15.94	23.67	23.10	17.89	32.77
PWC(snt.)	[8]	4.60	73.98	33.04	18.17	24.75	7.01	17.29	21.72	17.45	15.20	30.93
LiteFlowNet	[5]	5.06	35.14	13.69	11.19	6.77	6.37	13.68	14.95	8.37	12.61	27.25

Table 1: **Minor Corrections to Fig.7b**: New results exclude 4.7% of the test set, affected by the discovered Blender bug. Most values differ from the original by less than 1.0, having no effect on the conclusions of our analysis.

References

[1] Y. Bai, D. M. Kaufman, C. K. Liu, and J. Popovic. Artist-directed dynamics for 2D animation. *ACM Transactions on Graphics*, 35(4):1–10, July 2016. 2

[2] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2011. 4

[3] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conf. on Computer Vision (ECCV)*, pages 611–625, Oct. 2012. 2, 3

[4] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. 4

[5] T.-W. Hui, X. Tang, and C. C. Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *CVPR*, pages 8981–8989, 2018. 4

[6] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, pages 1164–1172, 2015. 4

[7] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, pages 2432–2439. IEEE, 2010. 4

[8] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018. 4

[9] D. Sýkora, J. Dingliana, and S. Collins. *As-rigid-as-possible image registration for hand-drawn cartoon animations*. ACM, New York, New York, USA, Aug. 2009. 2

[10] B. Whited, G. Noris, M. Simmons, R. W. Sumner, M. H. Gross, and J. Rossignac. BetweenIT: An Interactive Tool for Tight Inbetweening. *Comput. Graph. Forum ()*, 29(2):605–614, 2010. 2

[11] J. Xing, L.-Y. Wei, T. Shiratori, and K. Yatani. Autocomplete hand-drawn animations. *ACM Trans. Graph.*, 34(6):1–11, 2015. 2

[12] J. Xu, R. Ranftl, and V. Koltun. Accurate Optical Flow via Direct Cost Volume Processing. In *CVPR*, 2017. 4