# Creative Flow+ Dataset
# Supplemental Material

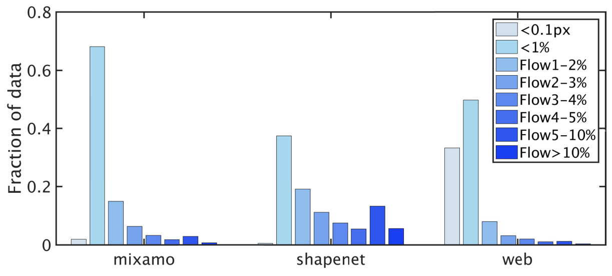
Maria Shugrina[1,2,3] www.shumash.com
Ziheng Liang[1,4] zhliang@cs.ubc.ca
Amlan Kar[1,2] amlan@cs.toronto.edu
Jiaman Li[1,2] ljm@cs.toronto.edu
Angad Singh[1,5] angad.singh@alum.utoronto.ca
Karan Singh[1] karan@dgp.toronto.edu
Sanja Fidler[1,2,3] fidler@cs.toronto.edu

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

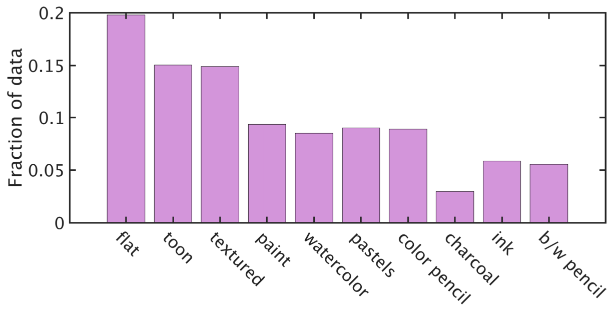
## 1. Detailed Dataset Statistics

This complements §5 in the paper. Because we used very different methods to obtain data in Mixamo, ShapeNet and Web sets, we break down speed distributions for each data source separately (Fig.2a), in addition to showing the overall dataset statistics in Fig.6a of the paper. The ShapeNet set contains more large-displacement flow examples due to fast object collisions and randomization in the rigid body simulation. This data is, therefore, likely to be more challenging.

We show style distribution in Fig.2b, where *flat*, *toon* and *textured* styles indicate corresponding Blender styles, and others correspond to Stylit examples drawn in a particular medium (refer to Fig.5a in the paper). As was our aim, we capture the diversity of styles found in the wild (paper §4.1), but do not necessarily aim to mimic the exact distribution of styles in Animation Show of Shows (paper Fig. 3), which is just one small set of stylized animated data. Anecdotal evidence suggests that flat and toon shading are common among other animation sources as well, and we ensure that these styles are well-represented in our dataset.



(a) Speed distribution by content source.



(b) Shading style distribution.

Figure 1: **Train Set Statistics:** speed distribution in % of frame width (1500px) in (a), and style distribution in (b).

## 2. Detailed Analysis

This complements §6 in the paper. We analyze the performance of several methods on the *foreground mask only* of our test set, as motion of featureless floor and background may be too ambiguous (See §3.2 in the paper). For quick reference, the methods analyzed are Horn-Schunck (HS) [4] implemented in [7], Classic+NLfast (Cl+NL) method from [7], and method by Brox et al. (brox) [1], Epic Flow (epic) [6], DC Flow (dcflow) [8], PWC-Net [7] trained on FlyingChairs (pwc-ch) [3] and on MPI Sintel [2] (pwc-sin), and LiteFlow Net [5] tuned for Sintel (liteflownet).

In in Fig.2, we analyze distributions of mean foreground endpoint error for all the samples in our 10K test set. The error distributions of all methods contain many outliers with extremely incorrect flow estimates and, in some cases, significantly lower error medians. This is not surprising, given that our data is designed to be noisy and does not meet standard assumptions of flow methods such as smoothness and

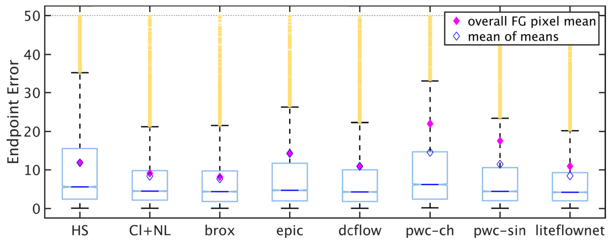


Figure 2: **Performance on Foreground Mask:** box plots show distribution of mean foreground pixel errors across all 10K test set samples, with box spanning the range between 25th and 75th percentile with median line shown and outliers appearing in yellow. Overall foreground error mean as reported in paper, Fig.7b, FG column is shown as filled magenta diamonds, mean of mean per-sample foreground endpoint error is shown as hollow diamonds.

temporal coherence of texture.

As an approximation of outliers, we collect the samples $S_{out}^M$ from our test set that performed in the worst 10% for each method $M$, with the total set of outliers defined as $S_{out} = \{S_{out}^1 \cup ... S_{out}^8\}$. If each method performed poorly on a unique subset of the test set, the size of $S_{out}$ would be about 8000. We find that $|S_{out}|$ is 2667, with 1010 outliers (38%) performing in the worst 10% for only one method, 466 outliers (17%) performing in the worst 10% for two methods, and the rest being in the worst 10% for three or more methods. The fact that 38% of outliers perform particularly poorly for only one method indicates that different methods find different aspects of our data difficult. We also discover that all methods find the ShapeNet component of the test set the most challenging, with 70% of outliers corresponding to frame pairs in these sequences. This could be due to the speed distribution of ShapeNet data (Fig.2a), symmetric featureless objects, specific motion profiles of the rigid body physics simulation, or the presence of the floor, which could confuse foreground pixel estimates even though it is excluded from the error computation.

**Universally Difficult Cases:** Among examples resulting in poor performance for 4 or more methods, we found that some sequences were particularly difficult and contributed many frames to $S_{out}$. For example, first three rows of Fig.3a had 20+ frames each which performed in the worst 10% for at least 4 methods. The first row is particularly challenging, due to highly stylized charcoal style and a complex ground truth motion pattern of a crumbling piece of paper. Even a human observer may need several frames to get the gist of the motion. In this web example, the floor was also labeled as an object and so included in the computation, increasing the difficulty. The symmetrical object in row 2 offers few features to suggest direction of rotation. The third row contains a large textured area, where static texture that does not move with the character confuses most methods in this foreground region, resulting in zero flow. Other sequences only contain a few difficult frames. For example, Fig.3a row 4 was the only difficult frame pair from that sequence, singled out due to extreme change in view angle.

**Uniquely Difficult Cases:** Other samples caused only one method to perform especially poorly. For example, first row of Fig.3b contains the same crumbling piece of paper (Fig.3a, row1) at another camera angle (Web data is rendered at two camera angles, see §5 in the paper) and toon shading. With these settings, the crisp outlines are enough to make most methods perform fairly well, with the exception of PWC-ch. This network is likewise most confused about the textured trumpet example in the second row of Fig.3b, where trumpet features are enough to make most methods do a reasonable job on the foreground, unlike the textured shirt in Fig.3a row 3, which confused many. Likewise, certain sequences are most confusing to specific classical methods, e.g. last row of Fig.3b most confuses HS.

**Easy Cases:** Examples where many methods' performance ranked in the top 10% tended to include textured background with rich features (Fig.4a, rows 2,3,4), or detailed shading (Fig.4a, rows 3,5). Some highly stylized examples showed good performance on the foreground (Fig.4a first row, Fig.4b rows 1,4,5), but caused confusion in the temporally-incoherent background, which would make optical flow results hard to use in practice. Among examples that performed in the top 10% for only one method, many still show good foreground performance for other methods, as 10% falls within narrow low error band (Fig.4b). It is interesting to note that chaotic background can confuse some learning-based methods and not others (e.g. Fig.4b rows 1,4,5 confuse PWC-ch, but not dcflow). Fig.4 suggests that even when performance on the foreground regions is good, untextured backgrounds present a problem and call for a notion of uncertainty in automatic optical flow estimation methods, especially if used for stylized content.

## 3. Summary

The diversity of styles and content in the Creative Flow+ Dataset (§1) generates a rich diversity of challenging examples (§2), with many combinations of factors contributing to example's difficulty or a method's robustness, leaving much room for future research.

## References

[1] 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. 1

[2] 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. 1

[3] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet:

Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 1

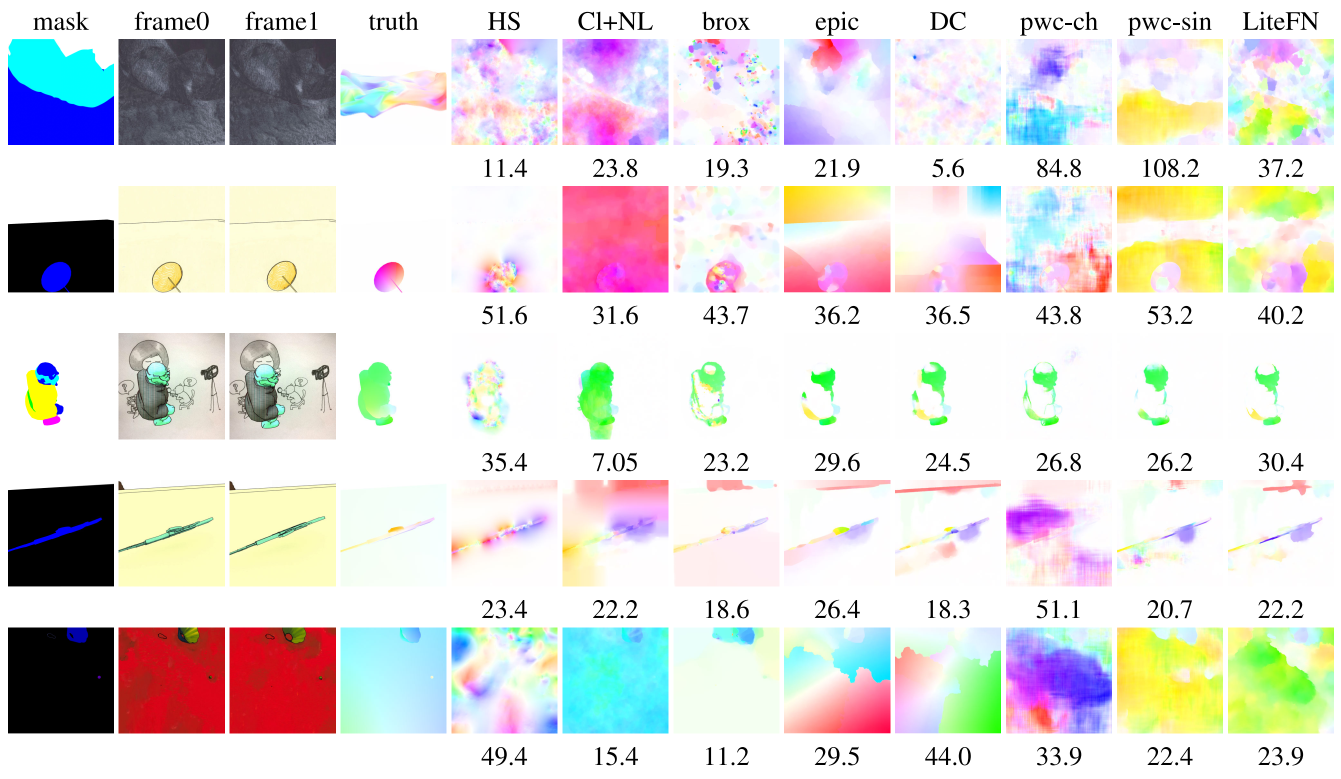
[4] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. 1

[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. 1

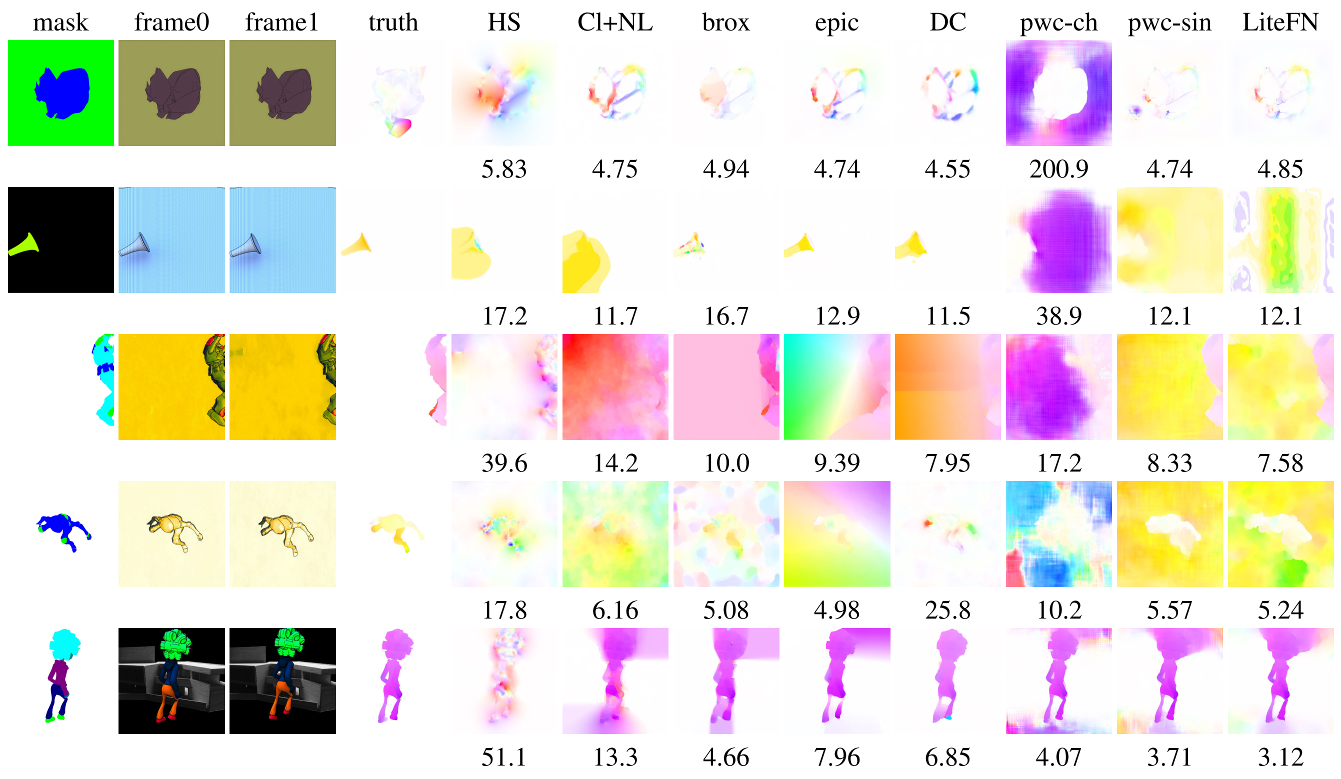
[6] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, pages 1164–1172, 2015. 1

[7] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, pages 2432–2439. IEEE, 2010. 1

[8] J. Xu, R. Ranftl, and V. Koltun. Accurate Optical Flow via Direct Cost Volume Processing. In *CVPR*, 2017. 1

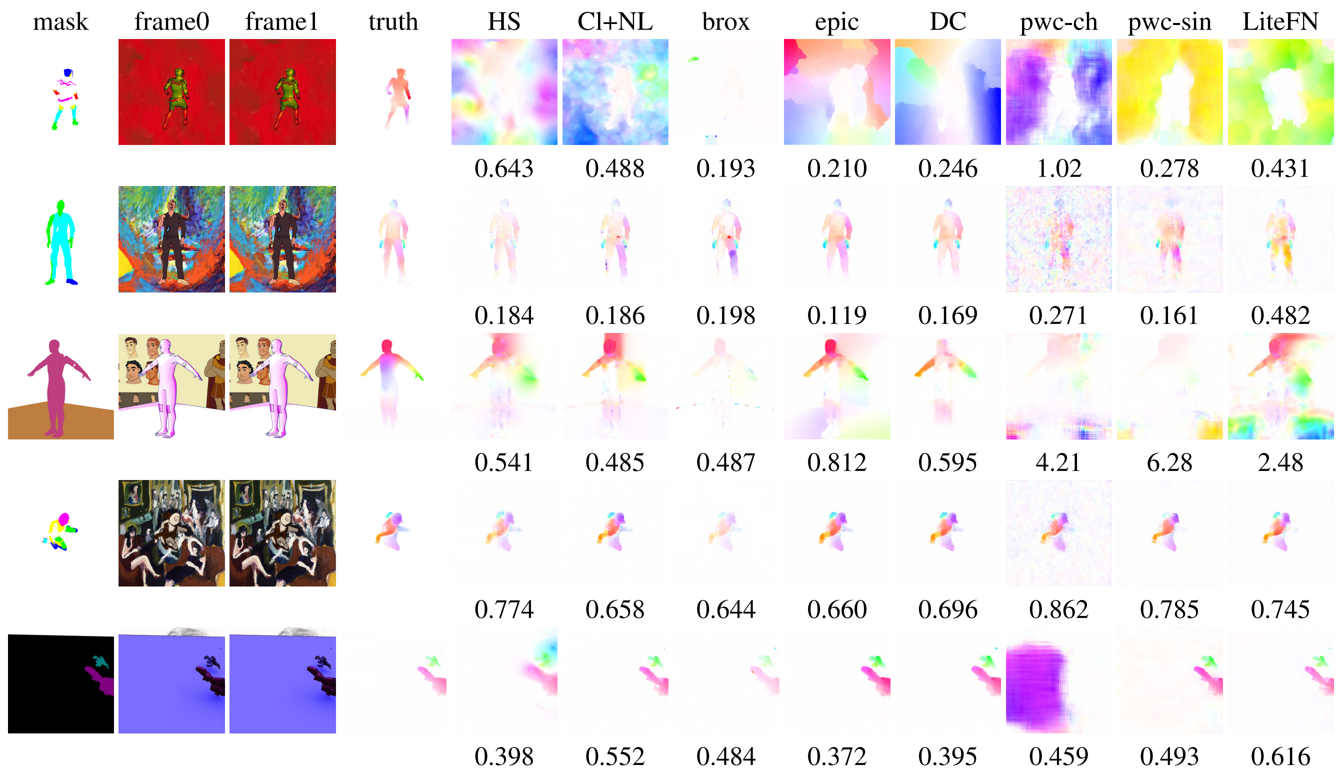


(a) **Universally Bad:** examples performing in the worst 10% for **at least 4** methods.

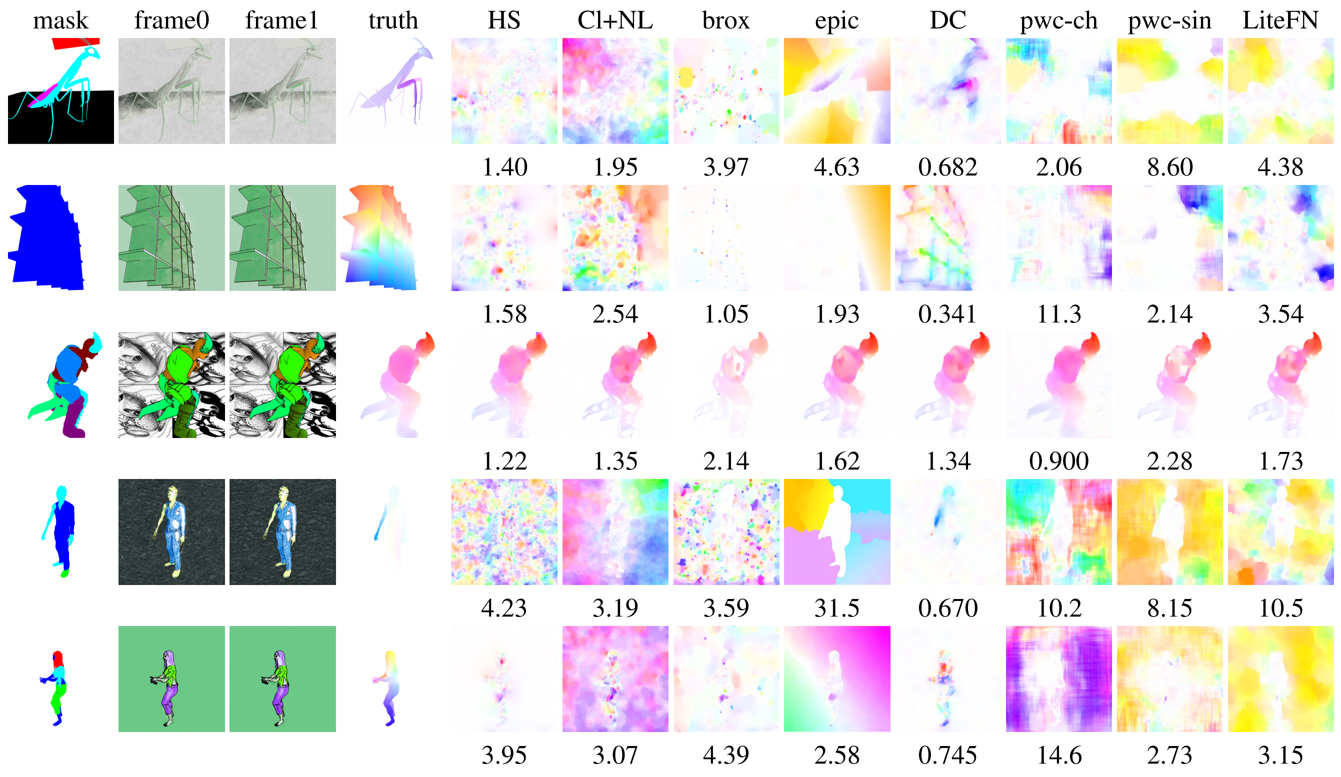


(b) **Uniquely Bad:** examples performing in the worst 10% for **exactly one** method.

Figure 3: **Samples of Poor Performance:** examples found challenging by many methods (a) and only one method (b). Average endpoint error for colorful (non-black non-white) pixels in the mask is shown below each example. Flow false coloring is computed individually for every method.



(a) **Universally Good:** examples performing in the best 10% for **at least 4** methods.



(b) **Uniquely Good:** examples performing in the best 10% for **exactly one** method.

Figure 4: **Sample of Good Performance:** examples with top optical flow performance by several (a) or one method (b). Average endpoint error for colorful (non-black non-white) pixels in the mask is shown below each example. Flow false coloring is computed individually for every method.