Learning optical flow from still images – Supplementary material

Filippo Aleotti* Matteo Poggi* Stefano Mattoccia
Department of Computer Science and Engineering (DISI)
University of Bologna, Italy
{filippo.aleotti2, m.poggi, stefano.mattoccia }@unibo.it

This document reports additional details concerning CVPR 2021 paper – "Learning optical flow from still images". Section 1 shows a further comparison between RAFT models trained on depthstilled data with models trained on real images with proxy labels obtained by a hand-made flow algorithm, while most of the remaining material concerns visualizations of the data generated by our *depthstillation* pipeline (Sections 2-6), as well as qualitative results of RAFT models on standard datasets (Sintel in Section 7, KITTI 2012 and 2015 in Section 8) and videos freely available online (Section 9).

1. Comparison with proxy-supervision from hand-crafted optical flow algorithms

As a further experiment, we compare the accuracy yield by depthstilled labels with respect to what achieved by generating proxy flow labels with a hand-crafted algorithm, although this latter requires image pairs – as well for training self-supervised methods. To this aim, we select RICFlow [1] as a competitor and we generate two training datasets, namely ricDAVIS and ricKITTI, respectively counterparts of dDAVIS and dKITTI used for the experiments in Tables 7 and 8 in the main paper.

As shown in the following tables, the RAFT models trained on depthstilled data always outperform the others. Thus, real images and *imperfect* labels obtained with RICFlow seem worse than *imperfect* images and depthstilled labels for training.

	Model	Dataset	KITTI12		KITTI15	
			EPE	Fl	EPE	Fl
(A)	RAFT	ricDAVIS	2.72	11.52	6.21	20.61
(B)	RAFT	dDAVIS	**1.78**	**6.85**	**3.80**	**13.22**

	Model	Dataset	KITTI12		KITTI15	
			EPE	Fl	EPE	Fl
(A)	RAFT	ricKITTI	2.34	8.50	6.02	18.51
(B)	RAFT	dKITTI	**1.76**	**5.91**	**4.01**	**13.35**

2. Depth is crucial for good Depthstillation

Given an input image $\mathcal{I}_0$, we aim to generate a ground-truth optical flow map $\mathcal{F}_{0\to1}$ and a new image $\mathcal{I}_1$. Knowing the depth $\mathcal{D}_0$ plays a crucial role to obtain a realistic flow field.

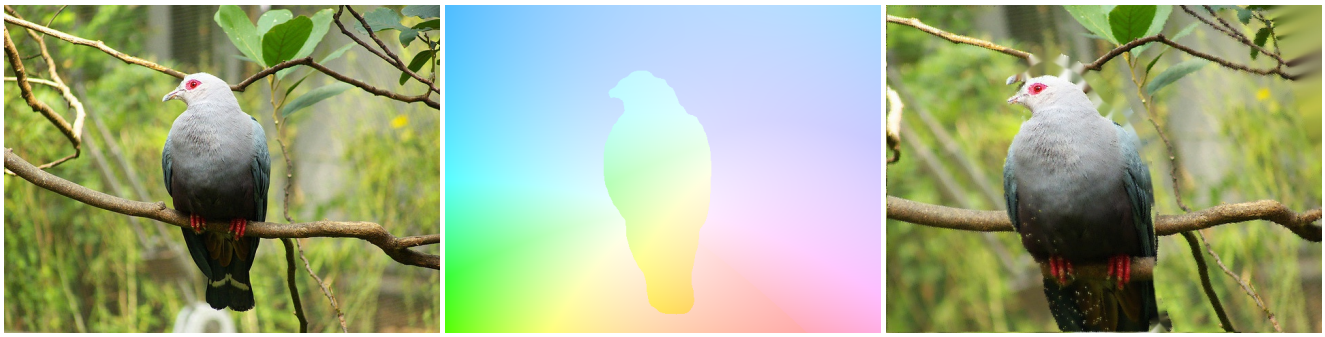
$\mathcal{I}_0$ $\mathcal{F}_{0\to1}$ $\mathcal{I}_1$

*Joint first authorship.

It is worth observing that, without a depth estimator, we could assume a constant depth for each image pixel generating planar motions. However, this would limit considerably the variety of dense flow fields we can generate. Consequently, optical flow networks trained on such data would yield low accuracy, as discussed in the submitted manuscript.

$\mathcal{I}_0$ $\mathcal{F}_{0\to1}$ $\mathcal{I}_1$

The reader might suppose that avoiding depth estimation, yet segmenting objects in the scene could be enough to obtain adequate training data. For instance, by assuming a constant depth and extracting the bird from the previous example, we could apply two different virtual motions to pixels belonging to the background and the foreground.



$\mathcal{I}_0$ $\mathcal{F}_{0\to1}$ $\mathcal{I}_1$

Although segmenting the image allows to model more complex motions with respect to not using either depth or segmentation, a RAFT model trained on data generated by knowing depth achieve much lower error on all metrics and datasets, as shown in the following tables, either without (left) or with (right) hole filling.

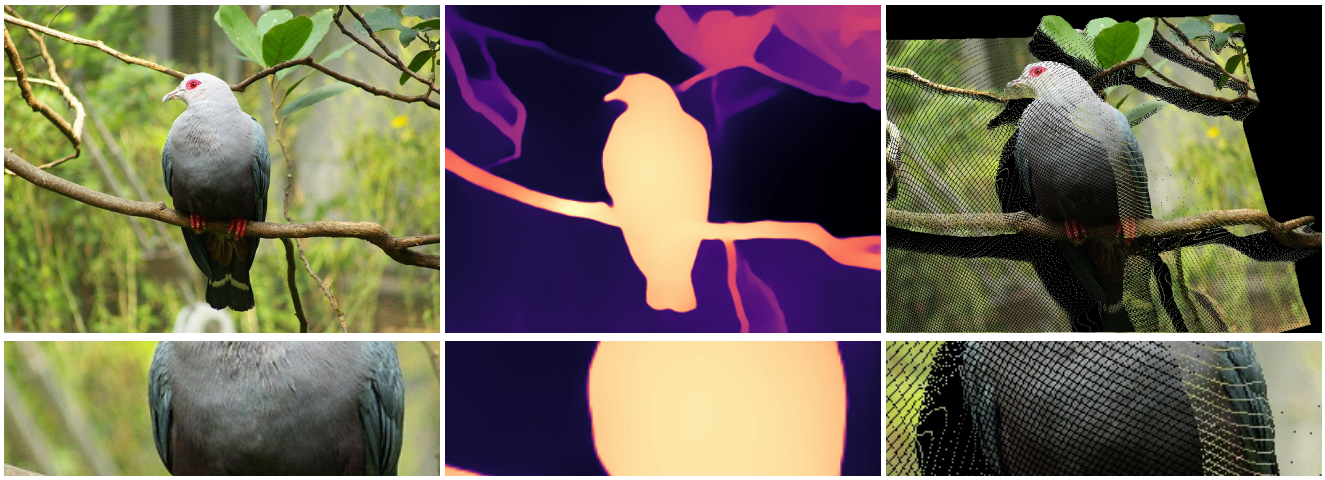
Depth est.	Hole fill.	Moving obj.	Sintel C.		Sintel F.		KITTI 12		KITTI 15	
			EPE	Fl	EPE	Fl	EPE	Fl	EPE	Fl
✓	✗	✗	**2.52**	**7.17**	**3.72**	**11.04**	**2.02**	**7.53**	**4.84**	**16.26**
✗	✗	✓	3.02	8.68	4.19	12.48	3.54	11.24	6.59	19.61

Depth est.	Hole fill.	Moving obj.	Sintel C.		Sintel F.		KITTI 12		KITTI 15	
			EPE	Fl	EPE	Fl	EPE	Fl	EPE	Fl
✓	✓	✗	**2.63**	**7.00**	**3.90**	**11.31**	**1.82**	**6.62**	**3.81**	**12.42**
✗	✓	✓	2.73	7.63	4.01	11.64	2.14	8.63	4.66	16.09

3. Components impacting view synthesis

We discuss the impact of the design choices introduced in our pipeline on the generated view $\mathcal{I}_1$.

Depth sharpening. Unfortunately, edges in monocular predictions are often blurred (A), causing *flying pixel* artefacts after warping, as those isolate pixels visible in $\mathcal{I}_1$ (B) on the dis-occlusion between the bird and the background.



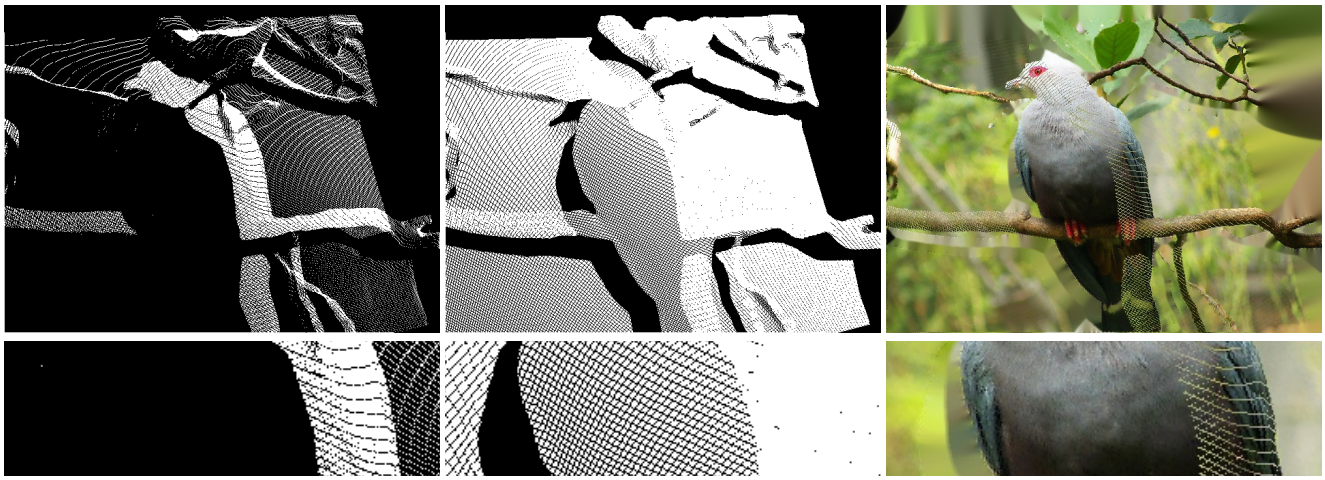
$\mathcal{I}_0$ (A) (B)

To alleviate this problem, we apply the bilateral filter [3] to $\mathcal{D}_0$, obtaining a depth map (C) with sharper edges that generates an image $\mathcal{I}_1$, displayed in (D), where most of the flying pixels have been removed.

$\mathcal{I}_0$ (C) (D)

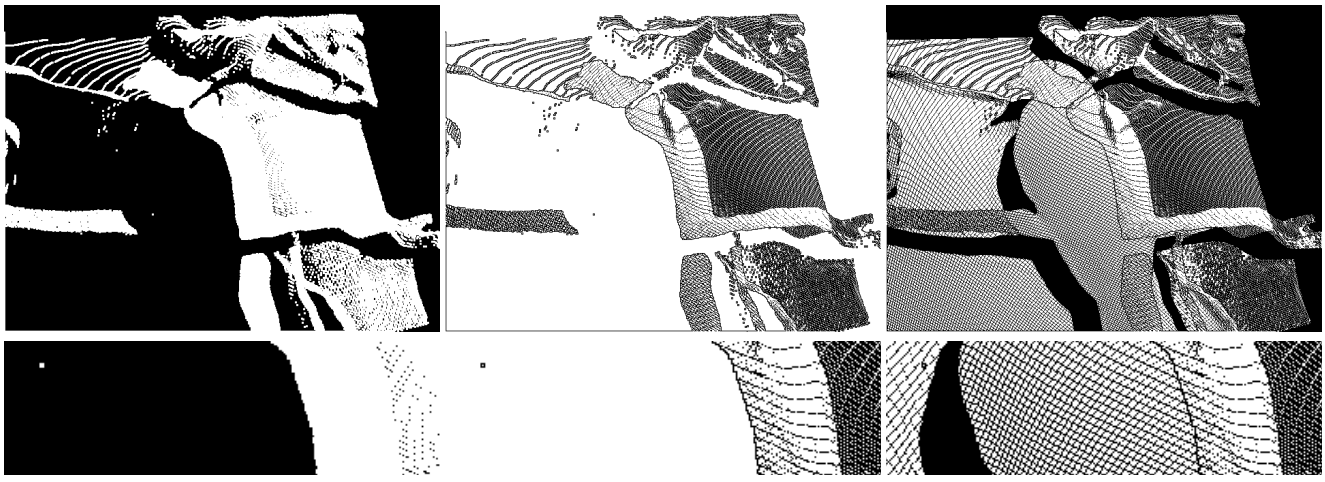
Collisions and holes. After applying the virtual camera motion, multiple pixels could reproject at the same coordinates in $\mathcal{I}_1$. When these *collisions* occur, we assume that the closer 3D point occludes all the others. Thus, we easily solve this ambiguity by assigning to the pixel in $\mathcal{I}_1$ the RGB value of the pixel with the lowest depth. We also keep a trace of colliding pixels for following computational steps, as we will see later. In particular, we build a binary mask $\mathcal{M}$, showed in (E), assigning 1 to all pixels for which collisions occur (*i.e.*, multiple pixels from $\mathcal{I}_0$ are reprojected there), 0 otherwise.

Another issue occurring after forward warping is the presence of *holes* or *stretching* artefacts. For these pixels, we need to *inpaint* an RGB value to make the overall image more realistic. To this aim, we build a binary mask $\mathcal{H}$, showed in (F), in which we assign 1 to pixels in $\mathcal{I}_1$ for which an RGB value has been retrieved from $\mathcal{I}_0$, 0 otherwise. This mask is paramount since it allows to determine which pixels have to be inpainted after warping: each pixel in $\mathcal{I}_1$ is inpainted with [5] if labeled with 0 in $\mathcal{H}$. At the end of the inpainting, we obtain (G), in which holes have been filled.



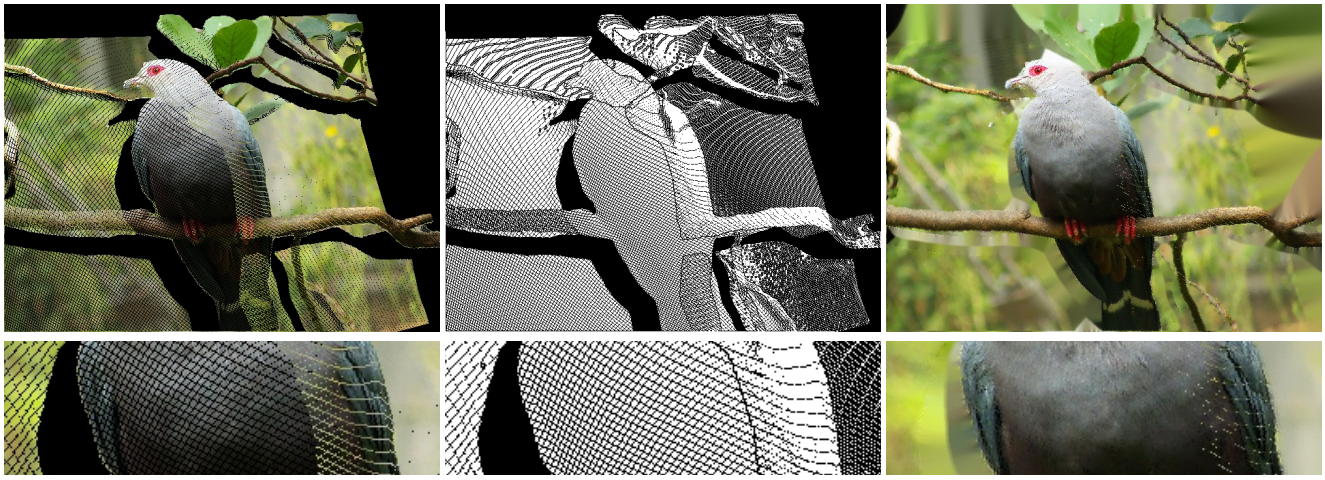
(E) (F) (G)

Bleeding artefacts. However, it is worth noticing that some regions in (G), such as the bird's wing on the right part of the image, are affected by different kinds of artefacts. For instance, due to the camera motion, some pixels of the bird in $\mathcal{I}_1$ are not filled with RGB values of the bird itself from $\mathcal{I}_0$, but with background ones. To obtain more realistic images, we need to track such pixels and then inpaint them as we do for holes. Unfortunately, we are not able to trace them in $\mathcal{H}$. However, we can observe that many of these artefacts have a 0 value in $\mathcal{M}$ (E), while being surrounded by pixels with 1 value. We leverage this behavior to detect these artefacts: by dilating $\mathcal{M}$ we obtain $\mathcal{M}'$ (H), on which several of these 0 values become 1. Now, we can compare $\mathcal{M}$ (E) with $\mathcal{M}'$ (H) to detect artefacts, in particular by looking for pixels that switched from 0 to 1. We store this information in a new mask, called $\mathcal{P}$ and displayed in (I), in which all the pixels with the same value in $\mathcal{M}$ and $\mathcal{M}'$ are marked with 1, 0 otherwise. Finally, we can update $\mathcal{H}$ including all the invalid pixels found in $\mathcal{P}$ by simply multiplying the two, thus obtaining our final mask $\mathcal{H}'$ depicted in (J).



(H) (I) (J)

By using $\mathcal{H}'$ (J) instead of $\mathcal{H}$ during the inpainting phase, we can obtain more realistic images, as in (K), removing most of the bleeding artefacts.



(D) (J) (K)

4. Components impacting flow generation

We now discuss the key components impacting the flow vectors angle and magnitude during the depthstillation process.

Camera intrinsics. Since any single picture is suitable for our method, K is most of the times unknown. However, our purpose is not to accurately reconstruct the 3D scene (that would be barely feasible with a monocular depth network, even knowing K). Thus, we can assume a *plausible* intrinsics matrix K, plausible because we want to avoid unrealistic settings (*i.e.*, points that are mapped infinitely far away, behind the camera reference frame, etc.). By playing on the focal length, we increase/decrease the depth scale and generate a smaller or larger magnitude of the flow vectors.

Camera intrinsics K as defined in the main paper

Different camera intrinsics K with focal length multiplied by 2

Virtual camera motion. Sampling different random motion parameters $(R|t)$ impacts, of course, on both the angle and magnitude of the generated optical flow field.

Randomly generated motion #1

Randomly generated motion #2

Moving objects. Segmenting objects and applying different camera motions to them allows for further increase the variety of the generated flow fields.

Randomly generated motion #1

Randomly generated motion #1 + randomly generated motion #2 applied to bird only

5. Traditional vs learned inpainting

Finally, we show the impact of adopting more advanced inpainting strategies to fill holes in $\mathcal{I}_1$. Most recent inpainting methods are based on deep learning [2] and, of course, require additional supervision that would add complexity to our pipeline, although not introducing sensible improvements on the quality of $\mathcal{I}_1$.

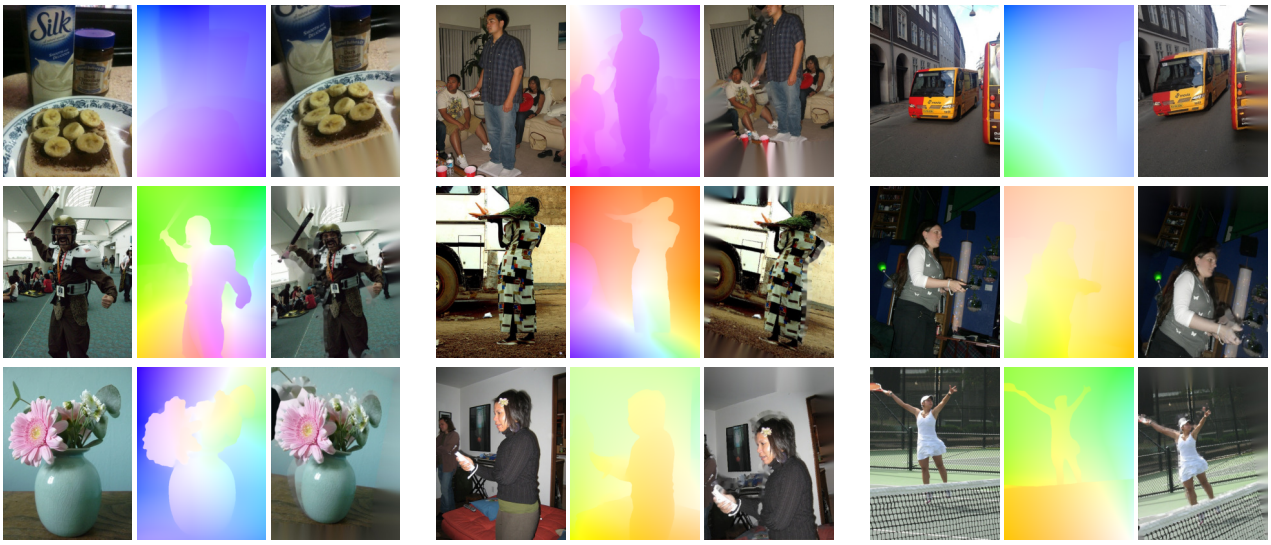
We show the outcome of some of these strategies, sorted by increasing complexity. From left to right, the results of 1) background texture filling [6], that consists of filling invalid pixels with RGB values taken from another image after color alignment, 2) the traditional image inpainting [5] used in our main paper, 3) the predictions of a GAN model [2] pre-trained for image inpainting and 4) the prediction of a Fourier Features Network [4], trained directly on the image itself and thus optimized for each single $\mathcal{I}_1$. For the latter [4], we train a compact MLP to predict RGB values given as input the 2D coordinates, remapped into Fourier Features, of valid pixels in $\mathcal{I}_1$. Then, we use the trained model to infer the RGB for invalid pixels in $\mathcal{I}_1$. Notice that, in our pipeline, this setting would require a standalone training over each image in the dataset, dramatically increasing the complexity of our solution.

These qualitative examples highlight that background filling, although useful when generating stereo images [6], is not enough in the case of 2D motions. Moreover, large occluded regions result challenging to fill even for deep learning models [2]. The Fourier Features Network [4] results in more visually pleasant images, yet turns out to be prohibitive in terms of time required to depthstill thousand of images. Hence, in our pipeline, we rely on [5] since it provides comparable results with minimum complexity.

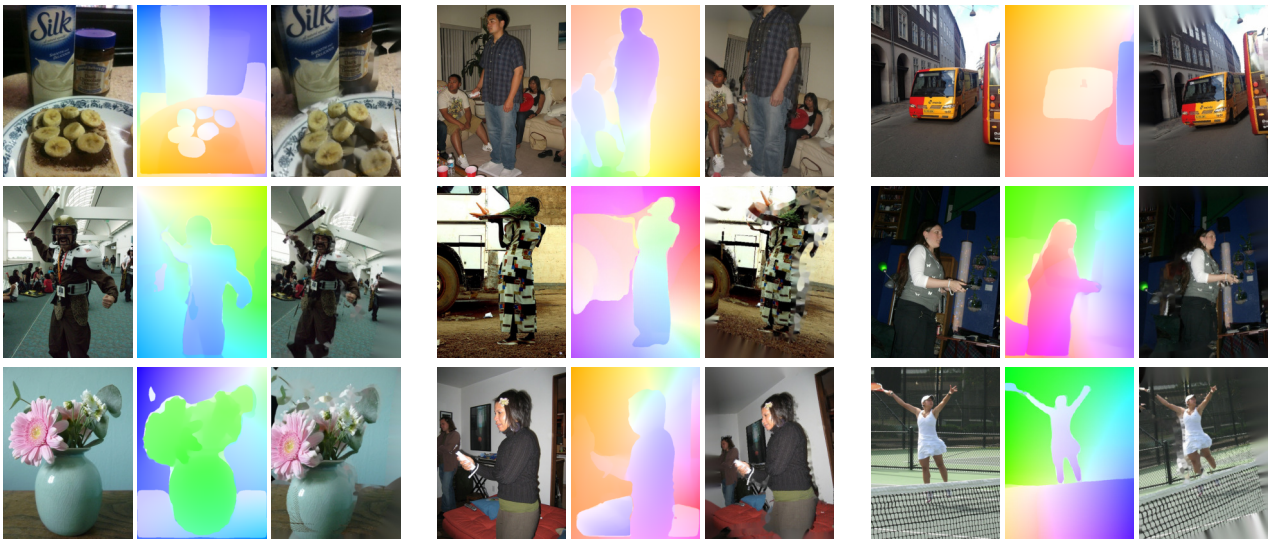
6. Qualitative examples – dCOCO

To conclude our analysis on the depthstillation pipeline, we show some qualitative examples from dCOCO. We first show images generated without taking into account depth, *i.e.* assuming a constant distance from the camera enabling only planar motions and rotations. We can notice how this produces a meager variety of flow vectors on a single frame.

Leveraging depth, we can model more complex flow fields, leading to much more variegate vectors occurring on the same scene. As shown in the submitted paper, optical flow networks trained on these images are dramatically more accurate.



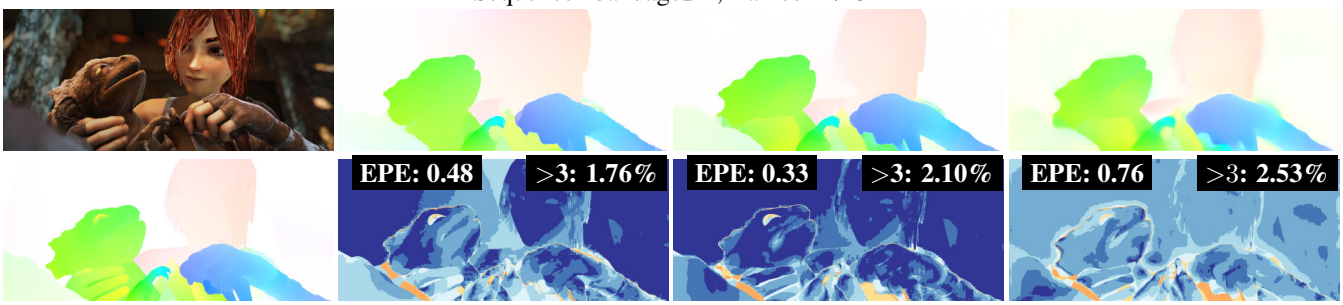
Finally, instance segmentation allows for extracting objects from the scene and simulate independent motions. This additional strategy results in even more variegate flow fields, although increasing the accuracy of optical flow networks marginally compared to the previous case.



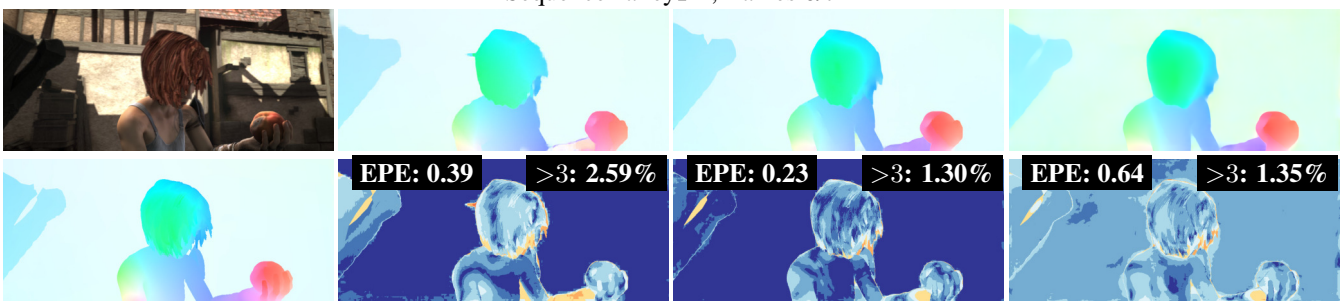
7. Qualitative results – RAFT on synthetic benchmarks

We now report some qualitative results on four image pairs taken from the Sintel-final dataset. For each, we report on two rows the reference frame (top) and the ground-truth flow (bottom), followed in order by optical flow maps estimated by RAFT when trained on Ch, Ch→Th and Ch→Th→dCOCO (top) and their corresponding error maps (bottom), with EPE and > 3 metrics overimposed.

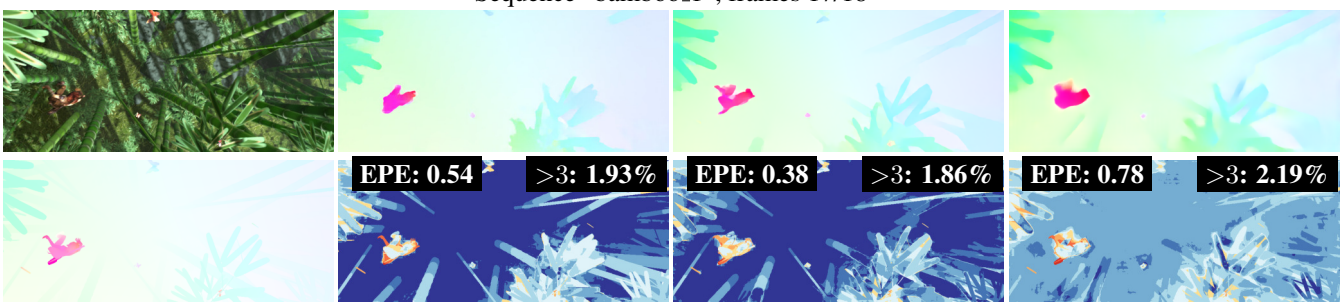
Sequence “bandage_2”, frames 24/25



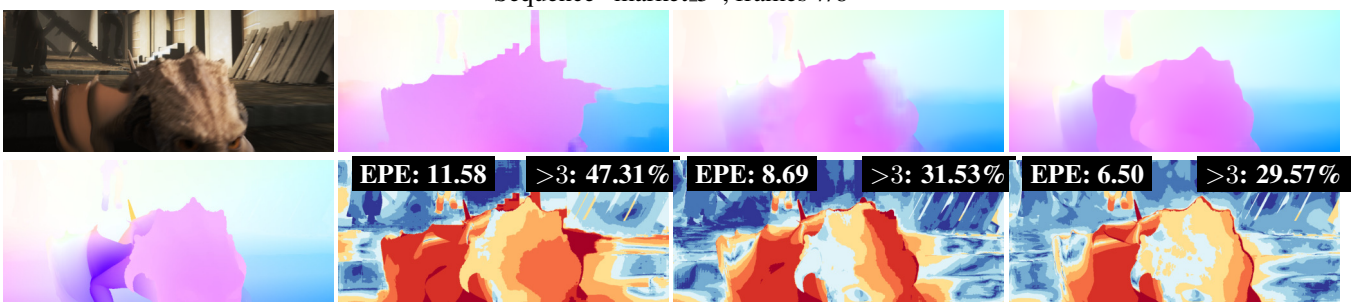
Sequence “alley_1”, frames 6/7



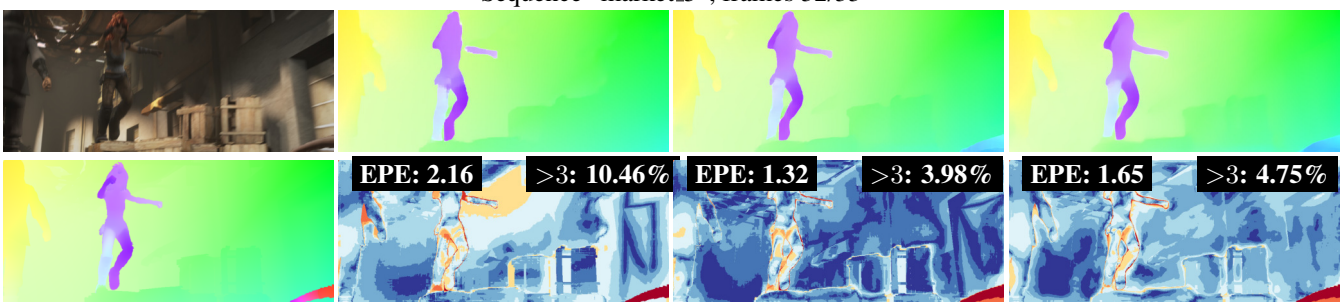
Sequence “bamboo_1”, frames 17/18



Sequence “market_5”, frames 7/8



Sequence “market_5”, frames 32/33

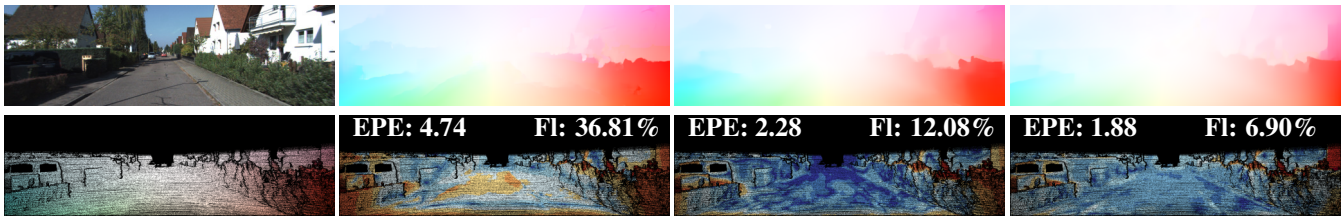


8. Qualitative results – RAFT on real benchmarks

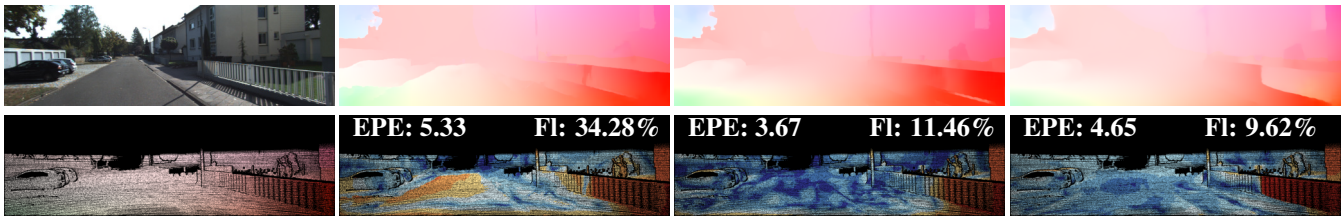
We also report some qualitative results on eight image pairs taken from KITTI 2012 and KITTI 2015 datasets. For each, we report on two rows the reference frame (top) and the ground-truth flow (bottom), followed in order by optical flow maps estimated by RAFT when trained on Ch, Ch→Th and dCOCO (top) and their corresponding error maps (bottom), with EPE and Fl metrics overimposed.

8.1. KITTI 2012

000012

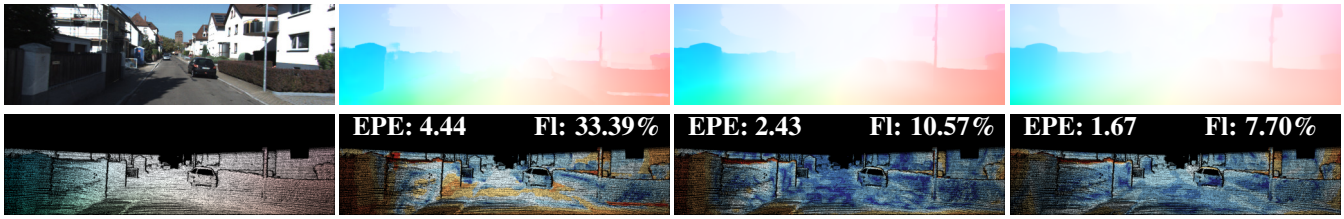


000027

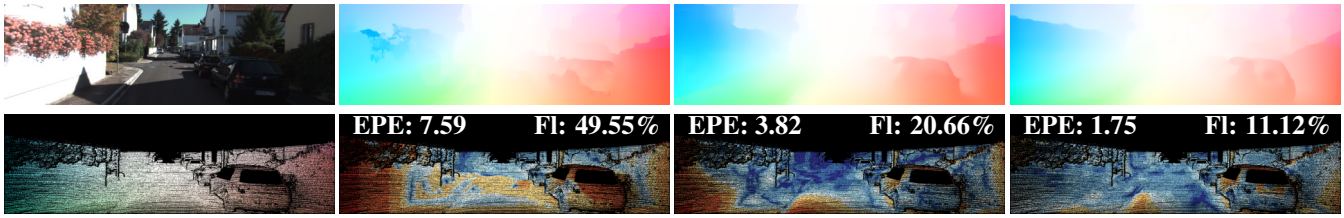


000040

000104

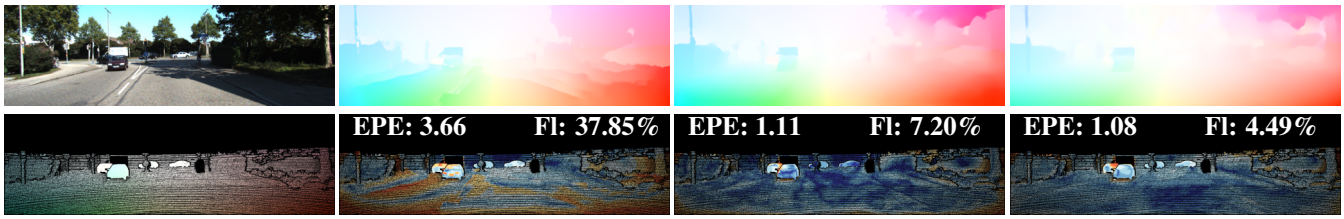


000141

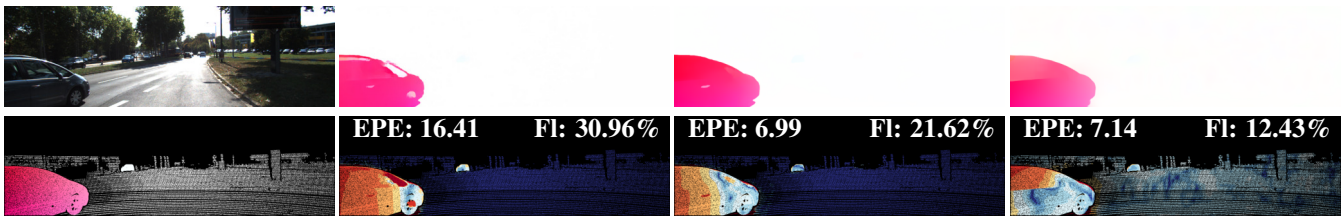


8.2. KITTI 2015

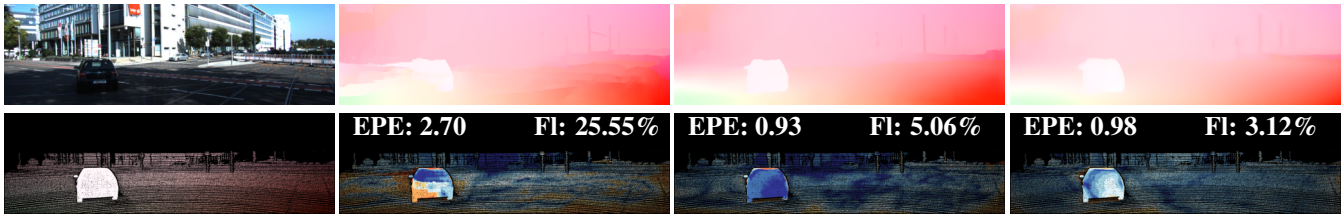
000007



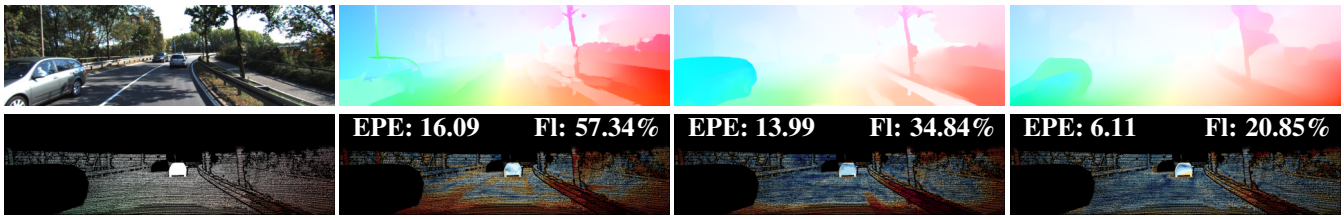
000091



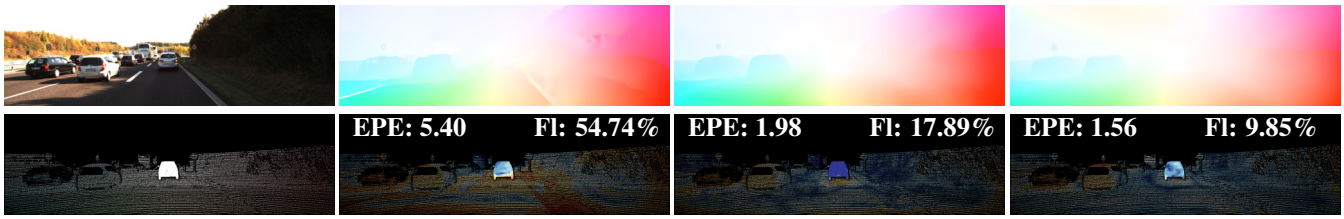
000171



000175

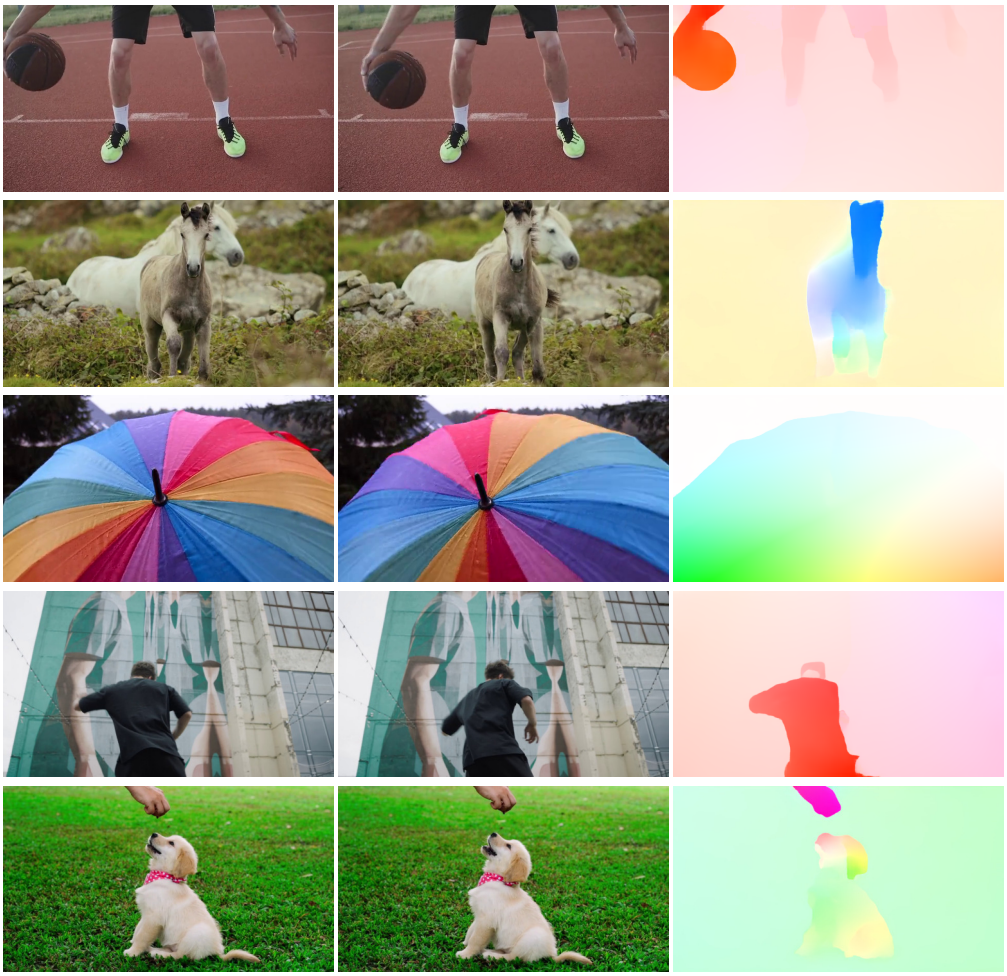


000199



9. Qualitative results – freely available web videos

Finally, since we primarly aim at generalization to unseed real videos, we show qualitative results produced by RAFT trained on Ch→Th→dCOCO on web videos freely available on the *pexels.com* portal. We report input images $\mathcal{I}_0$ and $\mathcal{I}_1$, followed by estimated optical flow $\mathcal{F}_{0\to1}$.



References

[1] Yinlin Hu, Yunsong Li, and Rui Song. Robust interpolation of correspondences for large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1

[2] Yuqing Ma, Xianglong Liu, Shihao Bai, Lei Wang, Dailan He, and Aishan Liu. Coarse-to-fine image inpainting via region-wise convolutions and non-local correlation. In *IJCAI*, 2019. 7

[3] Ziyang Ma, Kaiming He, Yichen Wei, Jian Sun, and Enhua Wu. Constant time weighted median filtering for stereo matching and beyond. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 49–56, 2013. 3

[4] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. 2020. 7

[5] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of graphics tools*, 9(1):23–34, 2004. 4, 7

[6] Jamie Watson, Oisin Mac Aodha, Daniyar Turmukhambetov, Gabriel J. Brostow, and Michael Firman. Learning stereo from single images. In *European Conference on Computer Vision (ECCV)*, 2020. 7