Appendix

Table of Contents

A. Method details

A.1. Network structures

Why ARFlow as backbone? We choose ARFlow [5] as our backbone in light of the following considerations.

- Our main goal is to investigate how SAM [4] can help with unsupervised optical flow instead of pushing the best possible performances by all means. Therefore, adopting a simple yet effective model, such as ARFlow, works better for our research.
- ARFlow is especially light-weight and easy to train. Previous related work, SemARFlow [12] also adopts the ARFlow backbone, so we follow them to borrow similar ideas from SemARFlow as well.
- Previous research has shown that ARFlow can be easily adapted to have close to UPFlow performances while maintaining simplicity [11, 12]. We follow those suggestions and build our own baseline model, evaluated in the experiments section.

Why is there no comparisons with SMURF? Admittedly, SMURF [9] has achieved outstanding performances on unsupervised optical flow estimation. However, we do not compare with them in the experiments section for the following reasons.

- As mentioned above, our goal is to see how SAM [4] can help with unsupervised optical flow instead of pushing the best possible performances by all means. We compare our models with the baseline model that does not apply SAM to show how SAM is effective, while other previous methods are shown as references to help understand in absolute terms how our adapted version performs.
- SMURF uses a larger architecture RAFT [10], and arguably, its success highly relies on its technical designs such as full-image warping, multi-frame self-supervision (which requires training a tiny model for each training sample separately), as well as its extensive data augmentations. We could definitely keep these technical designs in our model as well to enhance performances. However, they add great complexities to our network and may greatly increase the computational costs of our experiments. Therefore, we choose to stay simple and focus more on how to inject SAM information effectively.

Detailed structures Our detailed network structures are shown in Figs. 1 and 2.

A.2. Decoder computation

As shown in Fig. 2b in the main paper. For each iteration on level l, the decoder takes in image features $f_1^{(l)}$, $f_2^{(l)}$ and the flow estimate from the previous level $\hat{F}_{1\to2}^{(l+1)}$ (and also masks M_1, M_2 if the mask feature module is turned on), and outputs the refined flow estimate at the current level $\hat{F}_{1\to2}^{(l)}$.

A 1-by-1 convolution layer is first applied to $f_1^{(l)}$ to unify the feature channel sizes of different levels to be the same number 32. This module enables us to reuse the same following modules on all levels. We first upsample $\hat{F}_{1\to2}^{(l+1)}$ by 2 times to the same resolution $\tilde{F}_{1\to2}^{(l)}$ as the features at this level through a simple bilinear interpolation. The upsampled flow is then used to warp $f_2^{(l)}$ as

$$\hat{f}_1^{(l)}(\boldsymbol{p}) = f_2^{(l)}(\boldsymbol{p} + \tilde{F}_{1\to2}^{(l)}), \quad \forall \boldsymbol{p} \tag{1}$$

which can be implemented through a grid sampling process.

We then compute the correlation between $f_1^{(l)}$ and $\hat{f}_1^{(l)}$ through a 9×9 neighborhood window, yielding a flattened 81-channel correlation output. The correlation is then concatenated with the upsampled flow $\tilde{F}_{1\to2}^{(l)}$ and the 1-by-1 convolutional features as the input to the flow estimator network. Note that if the mask feature module is turned on, we also do warping and correlation to the mask feature $g_t^{(l)}$ in the same way.

The flow estimator estimates a flow residual added to the current estimate $\tilde{F}_{1\to2}^{(l)}$. Subsequently, a context net is also applied similarly to obtain the refined flow of this level $\hat{F}_{1\to2}^{(l)}$. The learned upsampler, adapted from the one in RAFT [10], outputs the parameters for the convex upsampling of $\hat{F}_{1\to2}^{(l)}$, yielding the upsampled final output $F_{1\to2}^{(l)}$.

A.3. Semantic augmentation

Heuristic Our heuristic for choosing key objects is that a key object may have many object parts that could be also detected by SAM, so the key object masks may overlap with many other object masks.

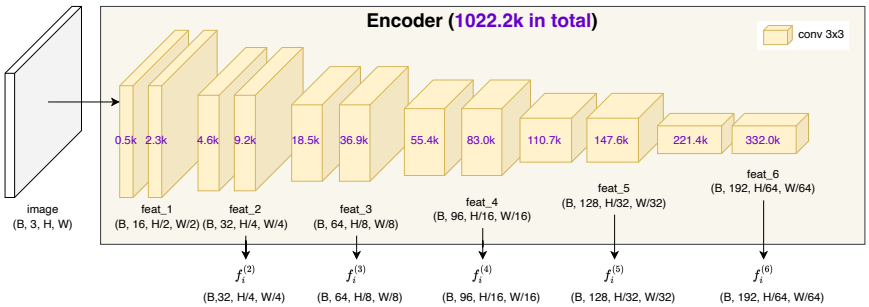

Figure 1. Detailed encoder structure (figure adapted from [11]); numbers in purple refer to the parameter sizes of each module.

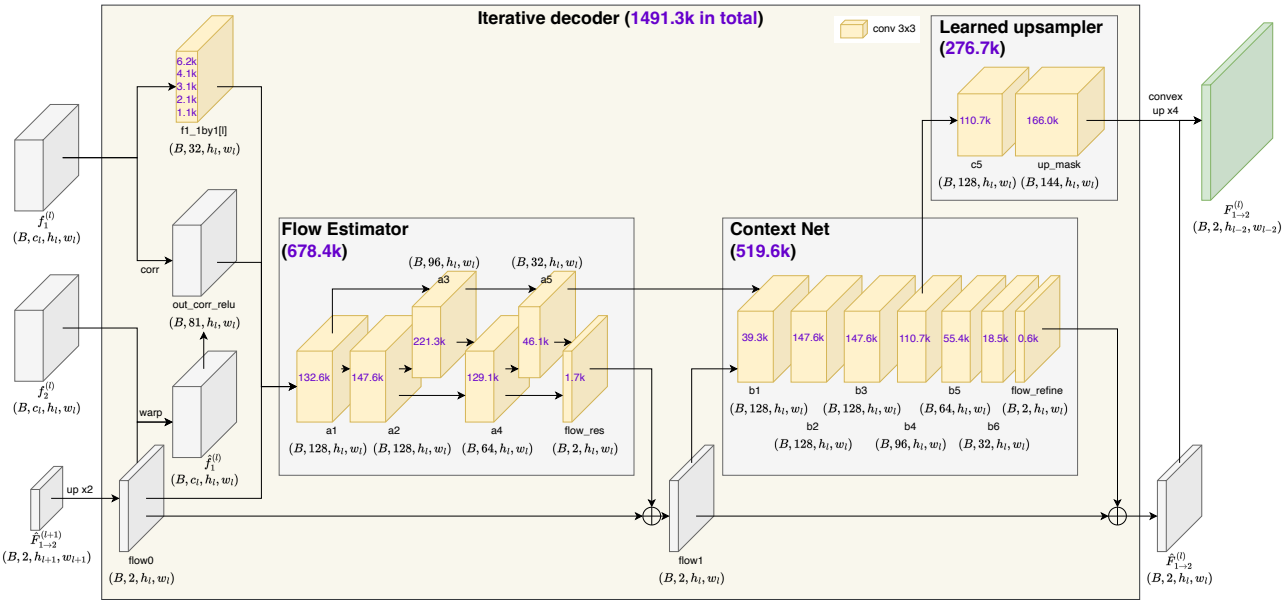

Figure 2. Detailed decoder structure (figure adapted from [11]); numbers in purple refer to the parameter sizes. The mask feature modules are not shown in the figure for conciseness. We use the same warping and correlation computation for mask feature and image features.

One example is shown in Fig. 3. Not only has the whole car object been detected by SAM, but also its components such as front and rear wheels, car doors and windows, lights, bumpers, and even the gas cap. As a result, the mask of the whole car object overlaps with all those component masks, whereas each component mask only overlaps with the car mask. Thus, the car object will be selected due to its high degree of mask overlap. Empirically, our heuristic works on all car objects pretty well, which are indeed key objects in autonomous driving.

Figure 3. Example of the SAM masks computed for a car patch

Key object selection We discuss more details on the process that we select key objects from the SAM masks. Suppose for the input image I, a number of n masks are detected by SAM, constituting masks $M \in \{0,1\}^{n\times H\times W}$. Denote $M(k) \in \{0,1\}^{H\times W}$ as the k-th mask, and $M(k,i,j) = 1$ means the pixel (i,j) is on the k-th object. For each mask $M(k)$, we examine the following.

- We first filter masks at a certain dimension. Suppose the bounding box of $M(k)$ has dimension $h \times w$, we only accept masks with $50 \le h \le 200, 50 \le w \le 400$. We avoid too large masks because they may not fit in the new sample well in our augmentation. We avoid too small masks as they make little difference in the augmentation.
- We drop the mask if the area of the mask is smaller than 50% of its bounding box area, *i.e.* $\sum_{i,j} M(k,i,j) < 50\% \cdot hw$. This rule is used to exclude those severely occluded objects.
- We accept the mask if it overlaps with at least 5 other

masks. The number of overlaps can be counted efficiently though matrix computation of M.

Training steps During key object selection, we save the selected masks for each training sample on the disk before starting to train, so this step adds little time or memory consumption during training. For each training sample, we load three key objects from the object cache for augmentation. For more details about semantic augmentation, we refer readers to the original paper of SemARFlow [12] and ARFlow [5].

A.4. Homography smoothness loss

Selecting object regions of interest Before selecting object regions, we first transform our raw SAM masks M_t to its full segmentation representation as described in Sec. 3.5 in the main paper. This makes sure that we do not refine the same pixel multiple times. Also, the segmentation is usually smaller pieces of objects, where homography is more likely to work well.

We estimate the occlusion region using forward-backward consistency check [6], as we did when computing photometric loss. The estimated occlusion region is a good cue of where the current flow estimate is less reliable. Then, we count the number of occlusion pixels for each segmentation in the full segmentation representation and pick the top six as candidates. Empirically, we find that six segmentation regions can generally cover most of the occluded pixels. Although including more candidates can improve performance, the improvement comes at a larger computational cost.

Refining each selected regions For each of the candidate regions selected above, we first find all the correspondences in that region from flow. We define the reliable flow as those non-occluded parts estimated above. We only proceed if the reliable flow part accounts for at least 20% of the whole region.

Using the reliable flow correspondences, we estimate homography using RANSAC [2] and compute the inlier percantage of this computation based on reprojection error. We only accept the homography if inlier percentage is at least 50%.

Consequently, using the accepted homography, we refine the correspondences of every pixel in the object region and generate refined flow.

Examples Some examples of the refined flow using homography are shown in Figs. 4 to 6.

Alleviating limitations of homography Admittedly, homography is not precise for all objects. It mostly works well on planar, rigid regions where no deformation occurs. In our method, we alleviate his issue through the following rules.

- We use full segmentation regions mentioned above, which generally refer to small object parts instead of the large object. Although the whole object may has very complex motions, its small parts are more likely to follow homography constraints.
- We introduce many rigorous accept/reject criteria mentioned above. If there is any sign that the homography

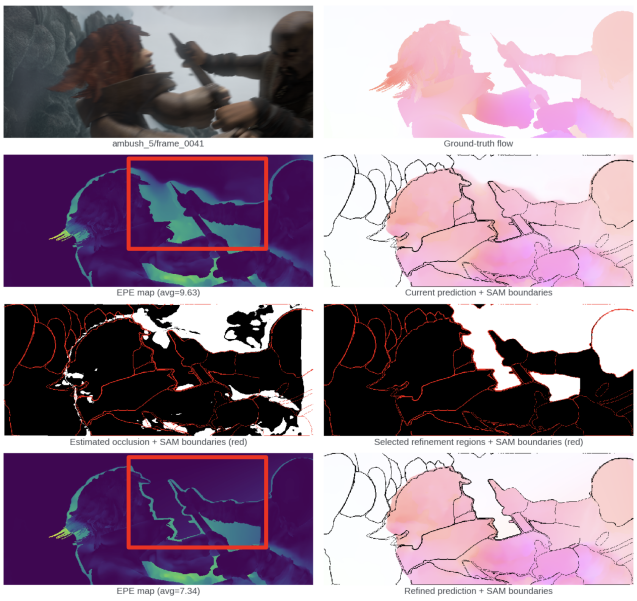

Figure 4. An example for homography refinement (Sintel)

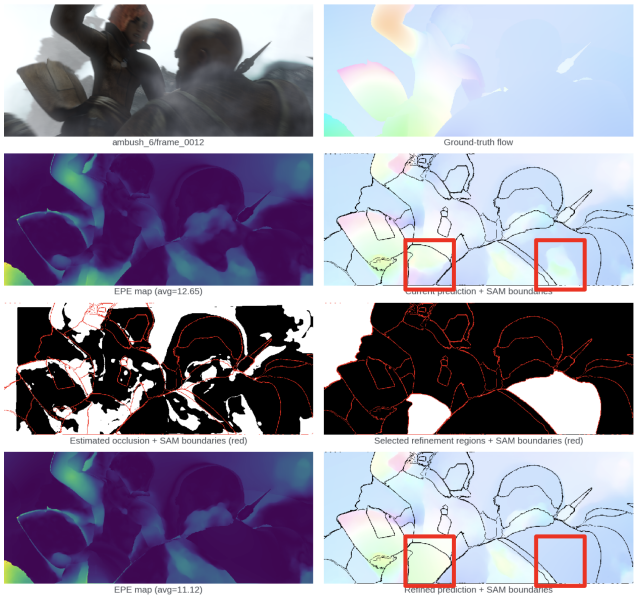

Figure 5. Another example for homography refinement (Sintel)

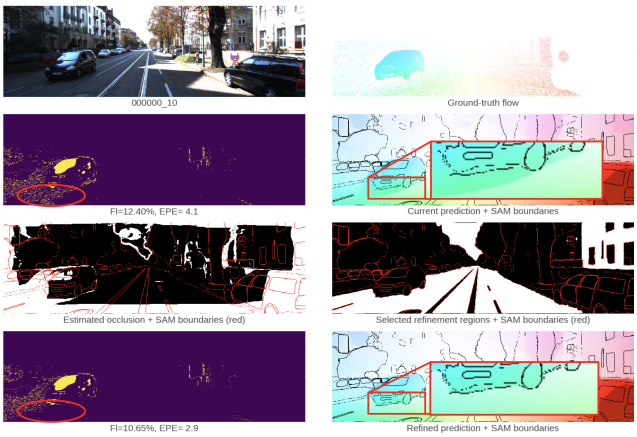

Figure 6. An example for homography refinement (KITTI)

relationship does not hold for the specific region, we stop using it. Only the most reliable homogrphies are used in refinement.

- We apply homography in the smoothness loss definition instead of direct post-processing. This allows our network to leverage between homography and other motion cues such as photometric constraints. Thus, a poor homography (if any) may not have large impacts if other signs/losses do not agree.

In addition, to better resolve this issue, it may be better if we could also obtain the semantic class of each object mask or if we could use text prompt to find masks, which may be updated in the later SAM versions.

A.5. Mask feature and correlation

Below are certain points that we need to take care when designing the mask feature module.

- The raw SAM masks are discrete and arbitrary (see Sec. 3.1 for explanations). The number of masks in each sample is not fixed. The masks can have different shapes and sizes. The masks can overlap or leave holes (parts that do not belong to any masks) in the frame. Therefore, we first standardize the mask representation using a full segmentation representation described in Sec. 3.5.
- Our mask feature module should be independent of the order of masks, *i.e.* our mask feature should be invariant against any permutation of masks. The mask/object IDs in the masks can be permuted without changing the segmentation map. Therefore, in our proposed module, we extract features for each mask separately regardless of its order.
- When we extract mask feature for each mask, the shape and size may vary, so our designed module should be well-defined for inputs of any size. This is why traditional convolutional layers may not work directly. Inspired by PointNet [8], for which the input size can also vary, we adopt operators like averaging or min/max to aggregate features of variable sizes. We apply max pooling in favor of average pooling because it adds non-linearity to the network and is often used in image classification networks. Apart from that, we need a new feature space where the max operation works. That is why we add the 1-by-1 convolutional layer at the front.
- The pooled feature is the same for every pixel in the same mask. This may cause numerical issues in optmization. Therefore, we concatenate and add another 1-by-1 convolutional layer to make sure the output mask feature is not exactly the same everywhere in the same mask.

A.6. Additional explanation on Fig. 4f in the paper

Why is there no curve for our proposed smoothness loss as a comparison? The way how our homography loss works is different. For traditional loss, since its gradients only concentrate around the flow boundary (Fig. 4d), the gradients push boundaries towards the optimal solution step by step, so we draw this landscape in Fig. 4f as if the flow boundary is moving. However, for our homography loss, the gradients apply on the whole region directly and instantly (Fig. 4e), so they are not just pushing the boundaries. Therefore, the same analysis in Fig. 4f may not apply.

B. Result details

B.1. Benchmark test screenshots

In Figs. 7 and 8, we show the benchmark test screenshots of our final model on KITTI and Sintel with more detailed evaluation metrics.

Error	Fl-bg	Fl-fg	Fl-all
All / All	6.40	14.98	7.83
All / Est	6.40	14.98	7.83
Noc / All	4.30	11.83	5.67
Noc / Est	4.30	11.83	5.67

This table as LaTeX

(a) KITTI-2015 [7] results

Error	Out-Noc	Out-All	Avg-Noc	Avg-All
2 pixels	6.39 %	10.63 %	0.9 px	1.4 px
3 pixels	3.79 %	7.05 %	0.9 px	1.4 px
4 pixels	2.67 %	5.29 %	0.9 px	1.4 px
5 pixels	2.06 %	4.21 %	0.9 px	1.4 px

This table as LaTeX

(b) KITTI-2012 [3] test results

Figure 7. Detailed test results of our final model on KITTI

B.2. More qualitative examples

We show more qualitative examples from the test set of KITTI-2015 (Fig. 9) and Sintel (Fig. 10).

B.3. Experiment timing

Inference As shown in Sec 4.7 in the main paper, our model inference is very efficient.

Training Training is fast because we only turn on semantic augmentation and homography smoothness loss after

		EPE all	EPE matched	EPE unmatched	d0-10	d10-60	d60-140	s0-10	s10-40	s40+
UnSAMFlow [233]	Final	5.200	2.558	26.747	4.699	2.260	1.644	0.912	3.390	30.842
UnSAMFlow [253]	Clean	3.926	1.671	22.341	3.785	1.602	0.707	0.628	2.100	24.744

Figure 8. Detailed test results of our final model on Sintel [1]

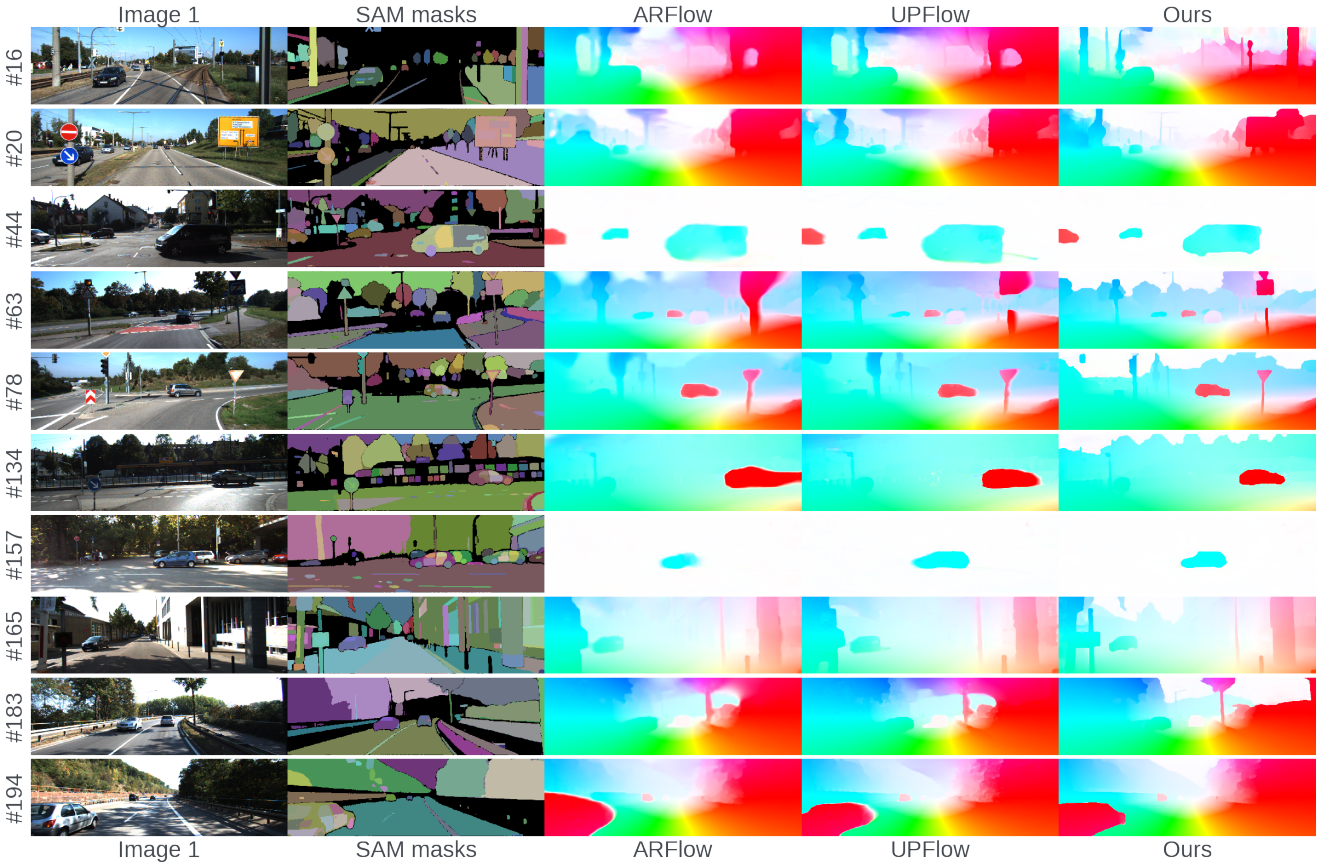

Figure 9. More qualitative results on KITTI-2015 test set [7]

150k iterations (200k in total), similar to what have been done in SemARFlow [12]. The reasons are as follows.

- Both semantic augmentation and homography smoothness loss rely on the current flow estimate to generate self-supervised loss signals, so we need to use flow at a later checkpoint to make sure they are reliable. Otherwise, the loss signals could be misleading.
- Semantic augmentation can generate very challenging self-supervised samples, which is better to be used at a later stage.

For the mask feature module adaptation, the added network size is very small (111.9k) since most of the added modules are 1-by-1 convolutions. Our typical full experiment training time is around 64 hours on 8 V100 GPUs.

We would like to emphasize that our goal is to investigate how SAM-style segmentations can benefit optical flow estimation. Optimizing SAM efficiency is outside the scope of this paper.

Figure 10. More qualitative results on Sintel test set [1]

References

[1] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, pages 611–625. Springer-Verlag, 2012. 5, 6

[2] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 3

[3] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, 32(11):1231–1237, 2013. 4

[4] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *ICCV*, pages 4015–4026, 2023. 1

[5] Liang Liu, Jiangning Zhang, Ruifei He, Yong Liu, Yabiao Wang, Ying Tai, Donghao Luo, Chengjie Wang, Jilin Li, and Feiyue Huang. Learning by analogy: Reliable supervision from transformations for unsupervised optical flow estimation. In *CVPR*, pages 6489–6498, 2020. 1, 3

[6] Simon Meister, Junhwa Hur, and Stefan Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, 2018. 3

[7] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, 2015. 4, 5

[8] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas.

Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017. 4

[9] Austin Stone, Daniel Maurer, Alper Ayvaci, Anelia Angelova, and Rico Jonschkowski. Smurf: Self-teaching multi-frame unsupervised raft with full-image warping. In *CVPR*, pages 3887–3896, 2021. 1

[10] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, pages 402–419. Springer, 2020. 1

[11] Shuai Yuan and Carlo Tomasi. Ufd-prime: Unsupervised joint learning of optical flow and stereo depth through pixel-level rigid motion estimation. *arXiv preprint arXiv:2310.04712*, 2023. 1, 2

[12] Shuai Yuan, Shuzhi Yu, Hannah Kim, and Carlo Tomasi. Semarflow: Injecting semantics into unsupervised optical flow estimation for autonomous driving. In *ICCV*, pages 9566–9577, 2023. 1, 3, 5