Self-Supervised Multi-Frame Monocular Scene Flow

Junhwa Hur[1] Stefan Roth[1,2]
[1]Department of Computer Science, TU Darmstadt [2] hessian.AI

Abstract

Estimating 3D scene flow from a sequence of monocular images has been gaining increased attention due to the simple, economical capture setup. Owing to the severe ill-posedness of the problem, the accuracy of current methods has been limited, especially that of efficient, real-time approaches. In this paper, we introduce a multi-frame monocular scene flow network based on self-supervised learning, improving the accuracy over previous networks while retaining real-time efficiency. Based on an advanced two-frame baseline with a split-decoder design, we propose (i) *a multi-frame model using a triple frame input and convolutional LSTM connections,* (ii) *an occlusion-aware census loss for better accuracy, and* (iii) *a gradient detaching strategy to improve training stability. On the KITTI dataset, we observe state-of-the-art accuracy among monocular scene flow methods based on self-supervised learning.*

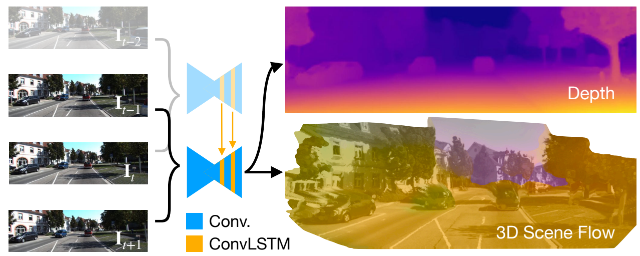


Figure 1. **Our multi-frame monocular scene flow approach** inputs three frames and estimates depth and scene flow (visualized in 3D [78] with scene flow color coding).

1. Introduction

Scene flow estimation, that is the task of estimating 3D structure and 3D motion of a dynamic scene, has been receiving increased attention together with a growing interest and demand for autonomous navigation systems. Many approaches have been proposed, based on various input data such as stereo images [3, 27, 38, 57, 68], RGB-D sequences [21, 37, 50], or 3D point clouds [4, 17, 35, 72].

Recently, *monocular* scene flow approaches [7, 23, 36, 73] have shown the possibility of estimating 3D scene flow from a pair of temporally consecutive monocular frames only, obviating complicated, expensive sensor setups such as a stereo rig, RGB-D sensors, or a LiDAR scanner. Only a simple, affordable monocular camera is needed. The availability of ground-truth data has been another key challenge for scene flow estimation in general. To address this, methods based on self-supervised learning [23, 36] have shown it possible to train CNNs for jointly estimating depth and scene flow without expensive 3D annotations. Yet, their accuracy is bounded by the limitation of only using two frames as input, their underlying proxy loss, and training instabilities due to the difficulty of optimizing CNNs for multiple tasks, particularly in a self-supervised manner [36].

Semi-supervised methods have demonstrated promising accuracy by combining CNNs with energy minimization [7] or sequentially estimating optical flow and depth to infer 3D scene flow [73]. Those methods, however, do not reach real-time efficiency due to iterative energy minimization [7] or the additional processing time from pre-computing depth and optical flow [73] beforehand. Yet, computational efficiency is important for autonomous navigation applications.

In this paper, we introduce a self-supervised monocular scene flow approach that substantially advances the previously most accurate real-time method of Hur *et al.* [23], while keeping its advantages (*e.g.*, computational efficiency and training on unlabeled data). We first analyze the technical design, revealing some limitations, and propose an improved two-frame backbone network to overcome them. Next, we introduce a multi-frame formulation that temporally propagates the estimate from the previous time step for more accurate and reliable results in consecutive frames. Previous monocular methods [7, 23, 36, 73] utilize only two frames as input, which is a minimal setup for demonstrating the underlying ideas. In contrast, our approach is the first to demonstrate how to exploit multiple consecutive frames, which are naturally available in most real-world scenarios.

We make the following main contributions: *(i)* We uncover some limitations of the baseline architecture of [23] and introduce an advanced two-frame basis with a *split-decoder design*. Contradicting the finding of [23] on using a single joint decoder, our split decoder is not only faster and more stable to train, but delivers competitive accuracy.

(ii) Next, we introduce our *multi-frame network* based on overlapping triplets of frames as input and temporally propagating the previous estimate via a convolutional LSTM [20, 59] (*cf*. Fig. 1). *(iii)* Importantly, we propagate the hidden states using *forward warping*, which is especially beneficial for handling occlusion; it is also more stable to train than using backward warping in a self-supervised setup. *(iv)* We propose an *occlusion-aware census transform* to take occlusion cues into account, providing a more robust measure for brightness difference as a self-supervised proxy loss. *(v)* Lastly, we introduce a *gradient detaching strategy* that improves not only the accuracy but also the training stability, which self-supervised methods for multi-task learning can benefit from. We successfully validate all our design choices through an ablation study.

Training on KITTI raw [11] in a self-supervised manner, our model improves the accuracy of the direct baseline of [23] by 15.4%. Owing to its self-supervised design, our approach also generalizes to different datasets. We can optionally perform (semi-)supervised fine-tuning on *KITTI Scene Flow Training* [43, 44], where we also outperform [23], reducing the accuracy gap to semi-supervised methods [7, 73] while remaining many times more efficient.

2. Related Work

Scene flow. First introduced by Vedula *et al*. [63, 64], scene flow aims to estimate dense 3D motion for each 3D point in the scene. Depending on the available input data, approaches differ in their objectives and formulations.

Stereo-based methods [1, 3, 38, 57, 71, 77] estimate a disparity map between stereo pairs to recover the 3D scene structure as well as a dense 3D scene flow field between a temporal frame pair. Earlier work mainly uses variational formulations or graphical models, which yields limited accuracy [2, 22] and/or slow runtime [42, 54, 65, 67, 68]. Recent CNN-based methods [24, 27, 40, 56] overcome these limitations: they attain state-of-the-art accuracy in real time by training CNNs on large synthetic datasets followed by fine-tuning on the target domain in a supervised manner. Un-/self-supervised approaches [31, 32, 69] aim to overcome the dependency on accurate, diverse labeled data, which is not easy to obtain. Approaches using sequences of RGB-D images [16, 19, 21, 37, 50, 51] or 3D point clouds [4, 17, 35, 45, 70, 72] have been also proposed, exploiting an already given 3D sparse point input.

In contrast, our approach jointly estimates both 3D scene structure and dense 3D scene flow from a monocular image sequence alone, which is a *more practical, yet much more challenging* setup.

Monocular scene flow. Estimating scene flow using a monocular image sequence has been gaining increased attention. Multi-task CNN approaches [9, 30, 36, 52, 74, 75, 79, 80] jointly predict optical flow, depth, and camera motion from a monocular sequence; scene flow can be reconstructed from those outputs. However, such approaches have a critical limitation in that they cannot recover scene flow for occluded pixels. Brickwedde *et al*. [7] propose to combine CNNs for monocular depth prediction with an energy-based formulation for estimating scene flow from the given depth cue. Yang *et al*. [73] introduce an integrated pipeline that obtains scene flow from given optical flow and depth cues via determining motion in depth from observing changes in object sizes. Using a single network, Hur *et al*. [23] directly estimate depth and scene flow with a joint decoder design, trained in a self-supervised manner.

All above methods [7, 23, 73] are limited to using two frames. In contrast, we demonstrate how to leverage *multiple consecutive frames* for more accurate and consistent results, which is desirable in real applications.

Multi-frame estimation. Multi-frame approaches to optical flow typically exploit a constant velocity or acceleration assumption [5, 10, 26, 29, 55] to encourage temporally smooth and reliable estimates. Using CNNs, propagation approaches have shown how to exploit previous predictions for the current time step, either by explicitly fusing the two outputs [39, 53] or using them as input for the current estimation [46]. Better temporal consistency has also been achieved using a bi-directional cost volume [25, 34] and convolutional LSTMs [14, 18]. Overall, these multi-frame approaches improve the optical flow accuracy, especially for occluded or out-of-bound areas.

For scene flow (stereo or RGB-D based), relatively few multi-frame methods have been introduced so far, all using classical energy minimization. A consistent rigid motion assumption has been proposed for temporal consistency [47, 65]. Other approaches include jointly estimating camera pose and motion segmentation [62], matching and visibility reasoning among multiple frames [58], or an integrated energy formulation [15]. These methods are robust against outliers and occlusion, improving the accuracy; yet, their runtime is slow due to iterative energy minimization.

We introduce a *CNN-based multi-frame scene flow approach in the challenging monocular setup*, ensuring real-time efficiency. Building on the two-frame network of [23], our method utilizes a bi-directional cost volume with convolutional LSTM connections, ensuring temporal consistency through overlapping frame triplets and temporal propagation of intermediate outputs (*cf*. Fig. 1). Moreover, we propose a forward-warping strategy for LSTMs.

3. Multi-frame Monocular Scene Flow

Given N temporally consecutive frames, $\{\mathbf{I}_{t-(N-2)}$, $\mathbf{I}_{t-(N-1)}$, ..., $\mathbf{I}_t$, $\mathbf{I}_{t+1}\}$, our main objective is to estimate 3D surface points $\mathbf{P} = (P_x, P_y, P_z)$ for each pixel $\mathbf{p} = (p_x, p_y)$ in the reference frame $\mathbf{I}_t$ and the 3D scene flow $\mathbf{s} = (s_x, s_y, s_z)$ of each 3D point to the target frame $\mathbf{I}_{t+1}$.

3.1. Refined backbone architecture

Advanced two-frame baseline. Our network architecture is based on the integrated two-frame network of Hur *et al.* [23], which uses PWC-Net [61] as a basis and runs in real time. The network constructs a feature pyramid for each input frame, calculates the cost volume, and estimates the residual scene flow and disparity with a joint decoder over the pyramid levels. While maintaining the core backbone, we first investigate whether recent advances in self-supervised optical flow can be carried over to monocular 3D scene flow.

Jonschkowski *et al.* [28] systematically analyze the key factors for highly accurate self-supervised optical flow, identifying crucial steps such as cost volume normalization, level dropout, data distillation, using a square resolution, *etc*. While we do not aim for a comprehensive review of such factors in the context of monocular scene flow, we performed a simple empirical study of their key findings.[1] We found cost volume normalization and using one less pyramid level (*i.e.* 6 instead of 7) to be helpful, and employ them for our advanced baseline. Other findings were less effective for monocular scene flow; hence we do not adopt them.

Moreover, we observed that the context network, a post-processing module with dilated convolutions [61], is a source of training instability in the self-supervised setup.[1] We thus discard the context network for stable convergence.

Split-decoder design. We further probe the decoder design in detail and introduce a split-decoder model that converges faster and more stably. Hur *et al.* [23] propose to use a single decoder (*cf*. Fig. 2a) that jointly predicts both scene flow and disparity based on the observation that separating the decoder for each task leads to balancing issues. This can result in a trivial prediction for the disparity (*e.g.*, outputting a constant value for all pixels). However, we observe that this issue mainly stems from the context network, which we discard (see above) due to stability concerns.

After discarding the context network (Fig. 2b), we find a better decoder configuration. We gradually split the decoder starting from the last layer into two separate decoders for each task and compare the scene flow accuracy in the experimental setting of [23]. Table 1 reports the result (lower is better). Discarding the context network degrades the accuracy by 4.1%, but in the end, splitting the decoder at the 2nd-to-last layer yields an accuracy competitive to the one of [23]. We choose this configuration (*i.e.* Fig. 2c) as our decoder design. Our findings suggest that the conclusions of [23] regarding the decoder design only hold in the presence of a context network. The benefit of our split decoder is that competitive accuracy is achieved more stably and in fewer training iterations (at 56% of the full training schedule), with a lighter network ($\sim$ 10% fewer parameters).[1]

[1] See supplementary material.

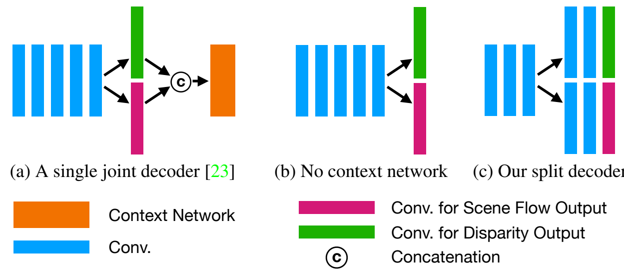


Figure 2. **Decoder configuration**: *(a)* A single joint decoder [23], *(b)* removing the context network, and *(c)* our split decoder design.

Configuration	D1-all	D2-all	Fl-all	SF-all
Single joint decoder (Fig. 2a, [23])	31.25	34.86	**23.49**	**47.05**
Removing the context network (Fig. 2b)	33.04	36.45	36.45	48.97
Splitting at the last layer	34.11	37.28	24.49	49.92
Splitting at the 2nd-to-last layer (Fig. 2c)	**30.50**	**34.72**	24.72	47.53
Splitting at the 3rd-to-last layer	31.04	35.79	24.57	47.85
Splitting at the 4th-to-last layer	32.21	36.28	24.65	48.84
Splitting into two separate decoders	32.12	36.14	25.02	48.66

Table 1. **Scene flow accuracy of split-decoder designs**: Removing the context network degrades the accuracy (*cf*. Sec. 4.2 for a description of the metrics), but splitting the decoder at the 2nd-to-last layer yields competitive accuracy while being stable to train.

3.2. Multi-frame estimation

Three-frame estimation. Toward temporally consistent estimation over multiple frames, we first utilize three frames at each time step [25, 34]. Fig. 3 illustrates our network for multi-frame estimation in detail. For simplicity, we only visualize one pyramid level (*i.e.* the dashed square in Fig. 3), noting that we iterate this across all pyramid levels. Given the feature maps from each frame at times $t-1, t$, and $t+1$, the forward cost volume (from t to $t+1$) and the backward cost volume (from t to $t-1$) are calculated from the correlation layer and fed into the decoder that estimates the forward scene flow $\mathbf{s}_t^{\mathrm{f}}$ and disparity $\mathbf{d}_t^{\mathrm{f}}$. The remaining inputs of the decoders are the feature map $\mathbf{x}_t^{\mathrm{feat}}$ from the encoder, upsampled estimates, and the hidden states in the convolutional LSTM (ConvLSTM) [59] module, see below, both from the previous pyramid level. For backward scene flow $\mathbf{s}_t^{\mathrm{b}}$ with disparity $\mathbf{d}_t^{\mathrm{b}}$, the same decoder with shared weights is used by switching the order of the inputs. We average the two disparity predictions for the final estimate, *i.e.* $\mathbf{d}_t = (\mathbf{d}_t^{\mathrm{f}} + \mathbf{d}_t^{\mathrm{b}})/2$, as they correspond to the same view and should be consistent forward and backward in time.

LSTM with forward warping. To further encourage temporal consistency, we employ a convolutional LSTM [59] in the decoder so that it can temporally propagate the hidden state across overlapping frame triplets (*cf*. Fig. 1) and implicitly exploit the previous estimates for the current time step. Fig. 5 shows our decoder in detail, visualizing only the forward scene flow case $\mathbf{s}_{t,l}^{\mathrm{f}}$ at pyramid level l for simplicity.

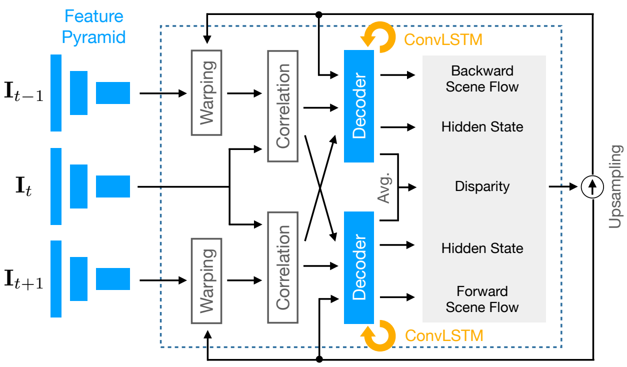


Figure 3. **Our network architecture for multi-frame monocular scene flow based on PWC-Net** [61]: At each pyramid level (*dashed square*), the two cost volumes are input to the decoder (shared for bi-directional estimation) to estimate the residual scene flow and disparity (averaged from the two decoders).

Inside the decoder, we place the ConvLSTM module right before splitting into two separate decoders so that we can temporally propagate the joint intermediate representation of scene flow and depth. The ConvLSTM is fed the feature map from previous layers, as well as cell state $\mathbf{c}^{\text{warp}}_{t-1,l}$ and hidden state $\mathbf{h}^{\text{warp}}_{t-1,l}$, both from the previous time step at the same pyramid level l. The module outputs the current cell state $\mathbf{c}_{t,l}$ and hidden state $\mathbf{h}_{t,l}$, which are fed into the subsequent split decoder that outputs residual scene flow and (non-residual) disparity, respectively. We use a leaky ReLU activation instead of tanh in the ConvLSTM, aiding faster convergence in our case.

Importantly, we forward-warp the previous states (*i.e.* $\mathbf{c}_{t-1,l}$ and $\mathbf{h}_{t-1,l}$) using the estimated scene flow at the previous time step $\mathbf{s}^{f}_{t-1}$ so that the coordinates of the states properly correspond. Without warping, each pixel in the current frame will attend to the previous state from mismatched pixels, which does not ensure proper propagation of corresponding states. Using backward warping based on backward scene flow at the previous pyramid level $\mathbf{s}^{b}_{t,l-1}$ may also be possible, but exhibits a challenge: using backward flow (at l–1) to warp the previous states to update itself (at l), which is not easy if the initial estimate is unreliable.

As an example, Fig. 4 shows the forward-warping result ($\mathbf{I}^{\text{warp}}_{t-1}$, Fig. 4c) of the previous frame $\mathbf{I}_{t-1}$ (Fig. 4a), which matches the current frame $\mathbf{I}_t$ (Fig. 4b) well. When a pixel $\mathbf{p}_1$ moves to $\mathbf{p}_2$ in the next frame, the pixel $\mathbf{p}_2$ in $\mathbf{I}_t$ should

(a) Previous frame $\mathbf{I}_{t-1}$ (b) Current frame $\mathbf{I}_t$ (c) Warping result $\mathbf{I}^{\text{warp}}_{t-1}$

Figure 4. **An example of forward warping**: *(a)* Previous frame, *(b)* current frame, and *(c)* forward-warped previous frame.

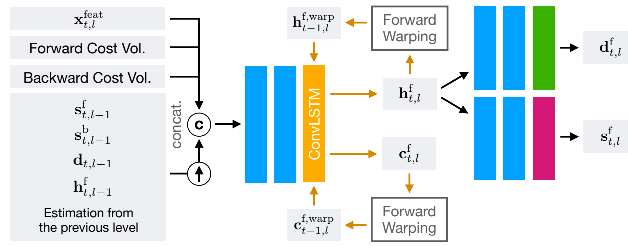


Figure 5. **A detailed view of the decoder at pyramid level** l, based on a convolutional LSTM with forward warping.

attend to the previous state of the corresponding pixel in $\mathbf{I}_{t-1}$, *i.e.* $\mathbf{h}_{t-1,l}(\mathbf{p}_1)$. To do so, we forward-warp the previous states using the estimated scene flow and disparity. Furthermore, we use a validity mask M to filter out states from mismatched pixels based on the affinity score of CNN feature vectors from corresponding pixels:

$$\mathbf{h}^{\text{warp}}_{t-1,l} = f_{\text{w}}(\mathbf{h}_{t-1,l}) M(\mathbf{x}^{\text{feat}}_{t-1}, \mathbf{x}^{\text{feat}}_{t}) \tag{1a}$$

$$\mathbf{c}^{\text{warp}}_{t-1,l} = f_{\text{w}}(\mathbf{c}_{t-1,l}) M(\mathbf{x}^{\text{feat}}_{t-1}, \mathbf{x}^{\text{feat}}_{t}) \tag{1b}$$

with

$$M(\mathbf{x}^{\text{feat}}_{t-1}, \mathbf{x}^{\text{feat}}_{t}) = (conv_{1\times1}(f_{\text{w}}(\mathbf{x}^{\text{feat}}_{t-1}) \cdot \mathbf{x}^{\text{feat}}_{t}) > 0.5), \tag{1c}$$

where f_{w} is the forward-warping operation with (implicitly) given estimated scene flow and disparity. To generate the per-pixel mask M, we forward-warp the previous (normalized) feature map ($\mathbf{x}^{\text{feat}}_{t-1}$), dot-product with the current (normalized) feature map ($\mathbf{x}^{\text{feat}}_{t}$) to calculate the affinity score, pass it through a 1×1 convolution layer, and apply a fixed threshold. Here, the 1×1 convolution is used to learn to scale the affinity score before thresholding.

For forward warping, we adopt the softmax splatting strategy of Niklaus *et al.* [48], which resolves conflicts between multiple pixels mapped into the same pixel location when forward-warping. In our case, we utilize the estimated disparity as a cue to compare the depth orders, determine visible pixels, and preserve their hidden states.

3.3. Self-supervised loss

Given the scene flow and disparity estimates over the multiple frames, we apply a self-supervised loss on each pair of temporally neighboring estimates that establish a bi-directional relationship. This allows us to exploit occlusion cues. As shown in Fig. 6, given the estimates for two time steps as $\{\mathbf{s}^{\text{f}}_{t-1}, \mathbf{s}^{\text{b}}_{t-1}, \mathbf{d}_{t-1}\}$ and $\{\mathbf{s}^{\text{f}}_{t}, \mathbf{s}^{\text{b}}_{t}, \mathbf{d}_{t}\}$, we apply the proxy loss to $\{\mathbf{s}^{\text{f}}_{t-1}, \mathbf{d}_{t-1}, \mathbf{s}^{\text{b}}_{t}, \mathbf{d}_{t}\}$. We adopt the self-supervised loss from Hur *et al.* [23], which consists of a view synthesis loss and a 3D reconstruction loss, guiding the disparity and scene flow output to be consistent with the given input images. The total self-supervised loss is a weighted sum of disparity loss L_{d} and scene flow loss L_{sf},

$$L_{\text{total}} = L_{\text{d}} + \lambda_{\text{sf}} L_{\text{sf}}. \tag{2}$$

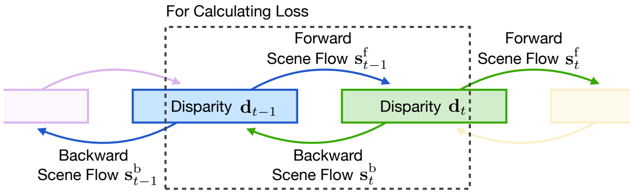
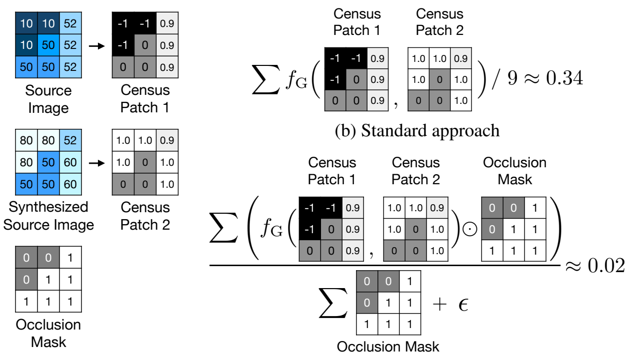


Figure 6. **Loss scheme.** The self-supervised loss is applied between each pair of temporally neighboring estimates.



(a) Census transform

(b) Standard approach

(c) Our occlusion-aware approach

Figure 7. **Occlusion-aware census transform**: *(a)* Computing the (continuous) census signature of two image patches and the corresponding occlusion mask, *(b)* standard approach of computing the Hamming distance of the census signature divided by the number of pixels, and *(c)* our occlusion-aware Hamming distance.

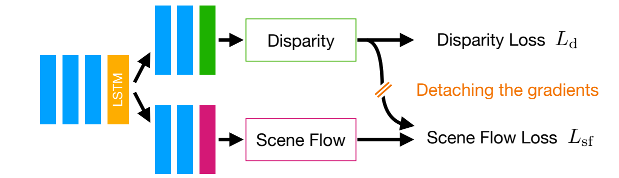
The main difference to [23] is that we newly propose an occlusion-aware census loss for penalizing the photometric difference. We only introduce our novel contribution here and provide details on the losses from [23] with our modifications in the supplementary material.

Occlusion-aware census transform. Carefully designing the proxy loss function matters for the accuracy of self-supervised learning [28]. For penalizing the photometric difference for the view-synthesis proxy task, the census transform [60, 76] has demonstrated its robustness to illumination changes, *e.g.*, in outdoor scenes [28, 33, 41, 66]. The conventional (ternary) census transform computes the local census patch (Fig. 7a) and calculates the Hamming distance between them to evaluate the brightness difference (Fig. 7b). However, it is vulnerable to outlier pixels (*e.g.*, occlusions) present in the patch, yielding a higher distance.

Taking into account the occlusion cue, we introduce an occlusion-aware census transform between images $\mathbf{I}$ and $\tilde{\mathbf{I}}$, which calculates the Hamming distance only for visible pixels:

$$\rho_{\text{census}}\big(\mathbf{I}, \tilde{\mathbf{I}}, \mathbf{O}\big) = \frac{1}{\sum_{\mathbf{p}} \mathbf{O}(\mathbf{p})} \cdot \tag{3a}$$

$$\sum_{\mathbf{p}} \frac{\sum_{\mathbf{y} \in [-3,3]^2} f_{\mathrm{G}}\Big(T(\mathbf{I}, \mathbf{p}, \mathbf{y}), T(\tilde{\mathbf{I}}, \mathbf{p}, \mathbf{y})\Big) \mathbf{O}(\mathbf{p}+\mathbf{y})}{\sum_{\mathbf{y} \in [-3,3]^2} \mathbf{O}(\mathbf{p}+\mathbf{y}) + \epsilon}$$



(a) Detaching the gradient between scene flow loss and disparity decoder

(b) Training loss

(c) Scene flow outlier rate

Figure 8. **Our gradient detaching strategy**: *(a)* Detaching the gradient between the scene flow loss and the disparity decoder in the early stages of the training improves *(b)* the training loss convergence and *(c)* the scene flow accuracy significantly.

with occlusion state $\mathbf{O}$ (with $\mathbf{O}(\mathbf{p}) = 1$ if visible) and

$$T(\mathbf{I}, \mathbf{p}, \mathbf{y}) = \frac{\mathbf{I}(\mathbf{p}+\mathbf{y}) - \mathbf{I}(\mathbf{p})}{\sqrt{(\mathbf{I}(\mathbf{p}+\mathbf{y}) - \mathbf{I}(\mathbf{p}))^2 + \sigma_{\mathrm{T}}^2}} \tag{3b}$$

$$f_{\mathrm{G}}(t_1, t_2) = \frac{(t_1 - t_2)^2}{(t_1 - t_2)^2 + \sigma_{\mathrm{G}}}, \tag{3c}$$

where $\sigma_{\mathrm{G}} = 0.1$ and $\sigma_{\mathrm{T}} = 0.9$. To facilitate differentiability, $T(\mathbf{I}, \mathbf{p}, \mathbf{y})$ in Eq. (3b) calculates a continuous approximation to the ternary value at pixel $\mathbf{p}$ with an offset $\mathbf{y}$ in image $\mathbf{I}$. f_G in Eq. (3c) is the Geman-McClure function [6] that scores the difference of the two input ternary values.

As shown in Fig. 7c, our occlusion-aware formulation can prevent from having a high score caused by occlusions in the census patch, thus providing a measure for the brightness difference that is more robust against outliers.

3.4. Improving the training stability

As shortly discussed in Sec. 3.1, discarding the context network improves the training stability. However, we found that integrating a ConvLSTM module [59] may still yield unstable training, resulting in trivial disparity predictions in the early stages of training.

To resolve the issue, we propose to detach the gradient from the scene flow loss (L_{sf} in Eq. (2)) to the disparity decoder in the early stages of training so that each split decoder focuses on its own task first. We conjecture that the gradient back-propagated from the scene flow loss to the disparity decoder strongly affects the disparity estimate, yielding a trivial prediction in the end. To prevent the scene flow from dominating, we detach the gradients, but only for the first 2 epochs of the training schedule as illustrated in Fig. 8a, and then continue to train in the normal setting.

Figures 8b and 8c demonstrate the effect of detaching the gradient in terms of the training loss and the scene flow outlier rate. Without detachment, the model outputs a constant disparity map in the early stage of training, thus yields higher scene flow error rates. In contrast, applying our gradient detaching strategy demonstrates faster and stable convergence, with much better accuracy (39.82% *vs*. 49.69%).

4. Experiments

4.1. Implementation details

For a fair comparison with the most closely related prior work [23], we use the same dataset (*i.e*. KITTI raw [11]) and the same training protocol, assuming a fixed stereo baseline. We use the *KITTI Split* [13] by splitting the 32 scenes total into 25 scenes for training and the remaining 7 for validation. Unlike [23], we divide the training/validation split at the level of entire scenes in order to exploit more continuous frames for our multi-frame setup and completely remove possible overlaps between the two splits. Then, we evaluate our model on *KITTI Scene Flow Training* [43, 44], using the provided scene flow ground truth. Note that *KITTI Split* and *KITTI Scene Flow Training* do not overlap. After our self-supervised training on *KITTI Split*, we optionally fine-tune our model on *KITTI Scene Flow Training* in a semi-supervised manner and compare with previous state-of-the-art monocular scene flow methods [7, 73].

Given that we use the network of [23] as the basis, we use the same augmentation schemes and training configurations (*e.g*., learning rate, training schedule, optimizer, *etc*.), except for the following changes. For training, we use one sequence of 5 temporally consecutive frames as a mini-batch. To ensure training stability, we detach the gradient between the scene flow loss and the disparity decoder during the first 2 epochs, as discussed in Sec. 3.4.[2,3]

4.2. Ablation study

We conduct a series of ablations to study the accuracy gain from our contributions over the two-frame baseline. We use *our* multi-frame train split of *KITTI Split* and evaluate on *KITTI Scene Flow Training* [43, 44] using the scene flow evaluation metric. The metric reports the outlier rate (in %, lower is better) among pixels with ground truth; a pixel is regarded an outlier if exceeding a threshold of 3 pixels or 5% w.r.t. the ground-truth disparity or motion. After evaluating the outlier rate of the disparity (D1-all), disparity change (D2-all), and optical flow (Fl-all), the scene flow outlier rate (SF-all) is obtained by checking if a pixel is an outlier on either of them.

Advanced two-frame baseline. In Table 2, we first conduct an ablation study of our advanced baseline described in Sec. 3.1. We first train the original implementation of Hur *et al*. [23] on *our* train split. Interestingly, the accuracy significantly drops by 29.4% (relative change) compared to training on their data split. This difference comes down to our train split containing less diverse scenes, which suggests that [23] may be somewhat sensitive to the choice of training data. Simply removing the context network already significantly improves the accuracy up to 17.1% by improving the training stability, almost closing the gap. As discussed in Sec. 3.1, we follow [28] and use one less pyramid level and feature normalization, which further improves the accuracy, even slightly beyond that of the original baseline trained on the split of [23] (covering more diverse scenes).

Baseline type	D1-all	D2-all	Fl-all	SF-all
Hur *et al*. [23] (on original split)	**31.25**	34.86	23.49	47.05
Hur *et al*. [23] (on our train split)	36.70	45.50	24.65	60.88
[23] − Context Net	34.24	37.32	25.06	50.49
[23] − Context Net + [28]	33.56	**34.49**	**22.92**	**46.70**

Table 2. The accuracy of the original baseline of [23] on our train split *(second row)* is significantly lower than on the original split *(first row)*. Removing the context network *(third row)* and exploiting the findings from [28] *(fourth row*, **our advanced two-frame baseline***)* improves both the accuracy and the training stability.

Multi-frame extension	Occ-aware census loss	D1-all	D2-all	Fl-all	SF-all
(Our advanced baseline)		33.56	34.49	22.92	46.70
✓		28.68	32.58	21.11	42.37
	✓	30.35	31.92	21.86	43.87
✓	✓	**27.33**	**30.44**	**18.92**	**39.82**

Table 3. **Ablation study of our main contributions**: Our key contributions (multi-frame extension, occlusion-aware census loss) consistently improve the accuracy over our advanced baseline.

Major contributions. In Table 3, we validate our major contributions, *i.e*. multi-frame estimation (Sec. 3.2) and the occlusion-aware census transform (Sec. 3.3), compared to our advanced two-frame baseline. Both contributions yield a relative accuracy improvement of 9.3% (multi-frame extension) and 6.1% (occlusion-aware census transform) over the baseline. Overall, our final model achieves 14.7% more accurate results than our baseline. Note that our advanced baseline already significantly outperforms the original implementation of [23] on our train split by 23.3%.

Multi-frame extension. We further ablate the technical details of our multi-frame estimation in Table 4. Inputting bidirectional cost volumes to the decoder (*i.e*. three-frame estimation) only marginally improves the accuracy. Adding a ConvLSTM that temporally propagates the hidden states (without using warping) only shows limited improvement as well. Using backward warping in the LSTM for temporal propagation with backward flow, interestingly, degrades the accuracy, possibly due to using less accurate backward flow from the previous level to warp the hidden states.

[2]Code is available at github.com/visinf/multi-mono-sf.

[3]See supplementary material for more details and analyses.

Frames	LSTM	Warping	D1-all	D2-all	Fl-all	SF-all
2	–	–	33.56	34.49	22.92	46.70
3	–	–	32.87	34.70	22.75	46.15
3	✓	–	**28.43**	33.75	25.22	45.43
3	✓	Backward	30.41	35.63	25.79	46.90
3	✓	**Forward**	28.68	**32.58**	**21.11**	**42.37**

Table 4. **Ablation study of our multi-frame estimation** (Sec. 3.2): Using a ConvLSTM with forward-warping the hidden states improves the accuracy up to 9.3% (relative improvement).

Loss type	D1-all	D2-all	Fl-all	SF-all
Brightness difference + SSIM	28.68	32.58	21.11	42.37
Standard census [28]	28.89	31.69	20.55	41.71
Occlusion-aware census (ours)	**27.33**	**30.44**	**18.92**	**39.82**

Table 5. **Ablation study of the occlusion-aware census transform**: Our occlusion-aware census loss provides a more discriminative proxy by taking occlusion cues into account.

However, when propagating the hidden states with forward warping with the estimated scene flow from the previous frame, we observe a significant relative accuracy improvement of up to 9.3% compared to two-frame estimation. Notably, this is the only setting in which there is a clear gain from going to more than two frames. This highlights the importance of propagating the hidden states and choosing a suitable warping method inside the ConvLSTM.

Occlusion-aware census transform. Lastly in Table 5, we compare our occlusion-aware census transform to the standard census transform and the widely used basic photometric loss consisting of brightness difference and SSIM [12, 13]. We conduct this experiment on top of our multi-frame architecture. Our occlusion-aware census further improves the scene flow accuracy by 6.0% (relative improvement) over the basic photometric loss and by 4.5% over the standard census transform [28].

4.3. Monocular scene flow

Table 6 compares our method with state-of-the-art monocular scene flow methods on *KITTI Scene Flow Training*. Our multi-frame architecture achieves the best scene flow accuracy among monocular methods [23, 36, 74] based on purely self-supervised learning. Our contributions yield a relative improvement over the direct baseline of [23] of 15.4%. Also note that our method closes a substantial part of the gap to the semi-supervised method of [7], but is significantly faster. Despite using multiple frames and having better accuracy, the runtime of our approach per time step is actually notably shorter (0.063s) than the direct baseline [23] (0.09s), benefiting from removing the context network and using one less pyramid level.

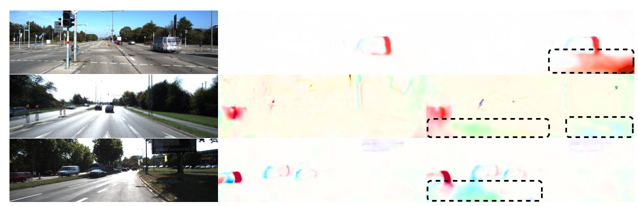
We further evaluate our model on the KITTI Scene Flow benchmark and compare with monocular scene flow approaches based on self- or semi-supervised learning in Table 7. Our model consistently outperforms [23]. Optionally fine-tuning (-ft) with 200 annotated pairs, our approach reduces the gap to semi-supervised methods that use a large amount of 3D LiDAR data [7] or multiple synthetic datasets for optical flow [73]. Yet, our model offers significantly faster runtime (*e.g.*, 650× faster than [7] and 4× faster than [73]). Our model can thus exploit available labeled datasets for accuracy gains while keeping the same runtime.

Method	D1-all	D2-all	Fl-all	SF-all	Runtime
EPC [74]	26.81	60.97	25.74	(>60.97)	0.05 s
EPC++ [36]	**23.84**	60.32	19.64	(>60.32)	0.05 s
Self-Mono-SF [23]	31.25	34.86	23.49	47.05	0.09 s
Multi-Mono-SF (ours)	27.33	**30.44**	**18.92**	**39.82**	0.063s
Mono-SF [7][§]	16.72	18.97	11.85	21.60	41 s

Table 6. **Evaluation on KITTI 2015 Scene Flow Training** [43, 44]: Our method achieves state-of-the-art accuracy among un-/self-supervised methods. [§]semi-supervised method.

Method	D1-all	D2-all	Fl-all	SF-all	Runtime
Mono-SF [7]	**16.32**	**19.59**	12.77	**23.08**	41 s
Mono expansion [73]	25.36	28.34	**6.30**	30.96	0.25 s
Self-Mono-SF-ft [23]	22.16	25.24	15.91	33.88	0.09 s
Multi-Mono-SF-ft (ours)	22.71	26.51	13.37	33.09	**0.063s**
Self-Mono-SF [23]	34.02	36.34	23.54	49.54	0.09 s
Multi-Mono-SF (ours)	**30.78**	**34.41**	**19.54**	**44.04**	**0.063s**

Table 7. **KITTI 2015 Scene Flow Test** [43, 44]: Our method consistently outperforms [23] and moves closer to the accuracy of semi-supervised methods with significantly faster runtime.

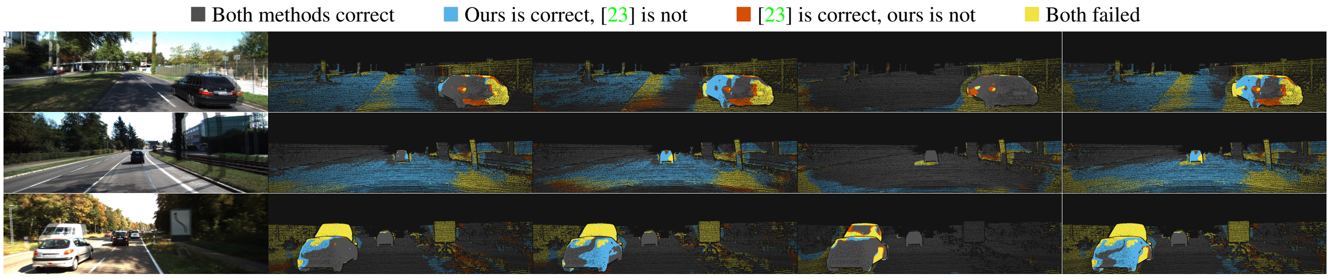


(a) Overlayed input images (b) Ours (c) Direct baseline [23]

Figure 9. **Qualitative comparison of temporal consistency**: Each scene shows *(a)* overlayed input images and scene flow difference maps of *(b)* our method, and *(c)* the direct baseline of [23], visualized using an optical flow color coding. The *dashed regions* highlight our more temporally consistent estimates.

4.4. Temporal consistency

We evaluate the temporal consistency of our model, comparing to the direct baseline of [23]. Lacking multi-frame metrics, we subtract two temporally consecutive scene flow estimates, project to optical flow, and visualize the result in Fig. 9. This shows how scene flow changes over time at the same pixel location. Our model produces visibly more temporally consistent scene flow, especially near out-of-bound regions and foreground objects.[3]



(a) Overlayed input images (b) Disparity error map (c) Disparity change error map (d) Optical flow error map (e) Scene flow error map

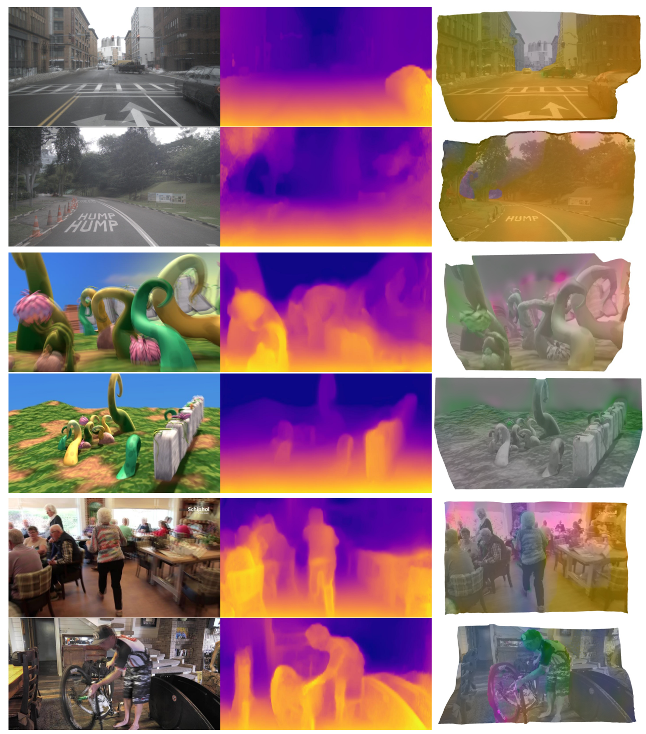
Figure 10. **Qualitative comparison with the direct baseline of [23]**: Each scene shows *(a)* overlayed input images and error maps for *(b)* disparity, *(c)* disparity change, *(d)* optical flow, and *(e)* scene flow. Please refer to the color code above the figure.

4.5. Qualitative comparison

In Fig. 10, we qualitatively compare our model with the direct baseline of [23] by visualizing where the accuracy gain mainly originates from. Our method outputs more accurate scene flow especially on planar road surfaces (1$^{\text{st}}$ and 2$^{\text{nd}}$ row), out-of-bound regions (2$^{\text{nd}}$ row), and foreground objects (1$^{\text{st}}$ and 3$^{\text{rd}}$ row). Especially the accuracy gain on foreground objects and planar road surfaces originates from more accurate estimates on the disparity and disparity change map.[3]

4.6. Generalization to other datasets

We test the generalization of our model trained on KITTI [11] to other datasets, such as the nuScenes [8], Monkaa [40], and DAVIS [49] datasets. Fig. 11 provides visual examples. Our method demonstrates good generalization to the real-world nuScenes dataset, but shows visually less accurate results on the synthetic domain (*i.e*. Monkaa), as can be expected. Interestingly on DAVIS, our method demonstrates reasonable performance on completely unseen domains (*e.g*., indoor, ego-centric).[3]



(a) Overlayed input images (b) Depth map (c) Scene flow visualization

Figure 11. **Generalization to nuScenes (top), Monkaa (middle), and DAVIS (bottom) datasets**: Each scene shows *(a)* overlayed input images, *(b)* depth, and *(c)* a 3D scene flow visualization.

5. Conclusion

We proposed a multi-frame monocular scene flow network based on self-supervised learning. Starting from the recent self-supervised two-frame network of [23], we first pointed out limitations of the decoder design and introduced an advanced two-frame baseline, which is stable to train and already improves the accuracy. Our multi-frame model then exploits the temporal coherency of 3D scene flow using overlapping triplets of input frames and temporally propagating previous estimates via a convolutional LSTM. Using forward warping in the ConvLSTM turned out to be crucial for accurate temporal propagation. An occlusion-aware census loss and a gradient detaching strategy further boost the accuracy and training practicality. Our model achieves state-of-the-art scene flow accuracy among self-supervised methods, while even yielding faster runtime, and reduces the accuracy gap to less efficient semi-supervised methods.

Future work should consider extending our self-supervised approach to challenging, uncontrolled capture setups with variable training baselines for even broader applicability. Also, while we do not explicitly model camera ego-motion mainly for the simplicity of the pipeline, additionally exploiting ego-motion can yield further benefits, *e.g*., for driving scenarios where rigid motion dominates.

Acknowledgements. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 866008).

References

[1] Filippo Aleotti, Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. Learning end-to-end scene flow by distilling single tasks knowledge. In *AAAI*, pages 10435–10442, 2020. 2

[2] Tali Basha, Yael Moses, and Nahum Kiryati. Multi-view scene flow estimation: A view centered variational approach. *Int. J. Comput. Vision*, 101(1):6–21, June 2013. 2

[3] Aseem Behl, Omid Hosseini Jafari, Siva Karthik Mustikovela, Hassan Abu Alhaija, Carsten Rother, and Andreas Geiger. Bounding boxes, segmentations and object coordinates: How important is recognition for 3D scene flow estimation in autonomous driving scenarios? In *ICCV*, pages 2574–2583, 2017. 1, 2

[4] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. PointFlowNet: Learning representations for rigid motion estimation from point clouds. In *CVPR*, pages 7962–7971, 2019. 1, 2

[5] Michael J. Black and P. Anandan. Robust dynamic motion estimation over time. In *CVPR*, pages 296–302, 1991. 2

[6] Michael J. Black and Anand Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *Int. J. Comput. Vision*, 19(1):57–91, July 1996. 5

[7] Fabian Brickwedde, Steffen Abraham, and Rudolf Mester. Mono-SF: Multi-view geometry meets single-view depth for monocular scene flow estimation of dynamic traffic scenes. In *ICCV*, pages 2780–2790, 2019. 1, 2, 6, 7

[8] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, pages 11621–11631, 2020. 8

[9] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *ICCV*, pages 7063–7072, 2019. 2

[10] Ravi Garg, Anastasios Roussos, and Lourdes Agapito. A variational approach to video registration with subspace constraints. *Int. J. Comput. Vision*, 104(3):286–314, Sept. 2013. 2

[11] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.*, 32(11):1231–1237, Aug. 2013. 2, 6, 8

[12] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *ICCV*, pages 3828–3838, 2019. 7

[13] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, pages 270–279, 2017. 6, 7

[14] Pierre Godet, Alexandre Boulch, Aurélien Plyer, and Guy Le Besnerais. STaRFlow: A spatiotemporal recurrent cell for lightweight multi-frame optical flow estimation. In *ICPR*, pages 181–186, 2020. 2

[15] Vladislav Golyanik, Kihwan Kim, Robert Maier, Matthias Nießner, Didier Stricker, and Jan Kautz. Multiframe scene flow with piecewise rigid motion. In *3DV*, pages 273–281, 2017. 2

[16] Andreas Görlitz, Jonas Geiping, and Andreas Kolb. Piecewise rigid scene flow with implicit motion segmentation. In *IROS*, pages 1758–1765, 2019. 2

[17] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. HPLFlowNet: Hierarchical permutohedral lattice FlowNet for scene flow estimation on large-scale point clouds. In *CVPR*, pages 3254–3263, 2019. 1, 2

[18] Shuosen Guan, Haoxin Li, and Wei-Shi Zheng. Unsupervised learning for optical flow estimation using pyramid convolution LSTM. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 181–186, 2019. 2

[19] Simon Hadfield and Richard Bowden. Kinecting the dots: Particle based scene flow from depth sensors. In *ICCV*, pages 2290–2295, 2011. 2

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[21] Michael Hornáček, Andrew Fitzgibbon, and Carsten Rother. SphereFlow: 6 DoF scene flow from RGB-D pairs. In *CVPR*, pages 3526–3533, 2014. 1, 2

[22] Frédéric Huguet and Frédéric Devernay. A variational method for scene flow estimation from stereo sequences. In *ICCV*, pages 1–7, 2007. 2

[23] Junhwa Hur and Stefan Roth. Self-supervised monocular scene flow estimation. In *CVPR*, pages 7396–7405, 2020. 1, 2, 3, 4, 5, 6, 7, 8

[24] Eddy Ilg, Tonmoy Saikia, Margret Keuper, and Thomas Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *ECCV*, volume 12, pages 614–630, 2018. 2

[25] Joel Janai, Fatma Güney, Anurag Ranjan, Michael J. Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *ECCV*, volume 16, pages 713–731. 2018. 2, 3

[26] Joel Janai, Fatma Güney, Jonas Wulff, Michael J. Black, and Andreas Geiger. Slow Flow: Exploiting high-speed cameras for accurate and diverse optical flow reference data. In *CVPR*, pages 1406–1416, 2017. 2

[27] Huaizu Jiang, Deqing Sun, Varun Jampani, Zhaoyang Lv, Erik Learned-Miller, and Jan Kautz. SENSE: A shared encoder network for scene-flow estimation. In *ICCV*, pages 3195–3204, 2019. 1, 2

[28] Rico Jonschkowski, Austin Stone, Jonathan T. Barron, Ariel Gordon, Kurt Konolige, and Anelia Angelova. What matters in unsupervised optical flow. In *ECCV*, volume 2, pages 557–572. 2020. 3, 5, 6, 7

[29] Ryan Kennedy and Camillo J. Taylor. Optical flow with geometric occlusion estimation and fusion of multiple frames. In *EMMCVPR*, pages 364–377, 2015. 2

[30] Hsueh-Ying Lai, Yi-Hsuan Tsai, and Wei-Chen Chiu. Bridging stereo matching and optical flow via spatiotemporal correspondence. In *CVPR*, pages 1890–1899, 2019. 2

[31] Seokju Lee, Sunghoon Im, Stephen Lin, and In So Kweon. Learning residual flow as dynamic motion from stereo videos. In *IROS*, pages 1180–1186, 2019. 2

[32] Liang Liu, Guangyao Zhai, Wenlong Ye, and Yong Liu. Unsupervised learning of scene flow estimation fusing with local rigidity. In *IJCAI*, pages 876–882, 2019. 2

[33] Pengpeng Liu, Irwin King, Michael R. Lyu, and Jia Xu. DDFlow: Learning optical flow with unlabeled data distillation. In *AAAI*, pages 8770–8777, 2019. 5

[34] Pengpeng Liu, Michael Lyu, Irwin King, and Jia Xu. SelFlow: Self-supervised learning of optical flow. In *CVPR*, pages 4571–4580, 2019. 2, 3

[35] Xingyu Liu, Charles R. Qi, and Leonidas J. Guibas. FlowNet3D: Learning scene flow in 3D point clouds. In *CVPR*, pages 529–537, 2019. 1, 2

[36] Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. Every pixel counts++: Joint learning of geometry and motion with 3D holistic understanding. *IEEE T. Pattern Anal. Mach. Intell.*, 2019. 1, 2, 7

[37] Zhaoyang Lv, Kihwan Kim, Alejandro Troccoli, Deqing Sun, James M. Rehg, and Jan Kautz. Learning rigidity in dynamic scenes with a moving camera for 3D motion field estimation. In *ECCV*, volume 5, pages 468–484, 2018. 1, 2

[38] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *CVPR*, pages 3614–3622, 2019. 1, 2

[39] Daniel Maurer and Andrés Bruhn. ProFlow: Learning to predict optical flow. In *BMVC*, 2018. 2

[40] Nikolaus Mayer, Eddy Ilg, Philip Häusser, 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*, pages 4040–4048, 2016. 2, 8

[41] Simon Meister, Junhwa Hur, and Stefan Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, pages 7251–7259, 2018. 5

[42] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, pages 3061–3070, 2015. 2

[43] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3D estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015. 2, 6, 7

[44] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 140:60–76, 2018. 2, 6, 7

[45] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *CVPR*, pages 11177–11185, 2020. 2

[46] Michal Neoral, Jan Šochman, and Jiří Matas. Continual occlusions and optical flow estimation. In *ACCV*, 2018. 2

[47] Michal Neoral and Jan Šochman. Object scene flow with temporal consistency. In *CVWW*, pages 273–281, 2017. 2

[48] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *CVPR*, pages 5436–5445, 2020. 4

[49] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, pages 724–732, 2016. 8

[50] Yi-Ling Qiao, Lin Gao, Yu-kun Lai, Fang-Lue Zhang, Ming-Ze Yuan, and Shihong Xia. SF-Net: Learning scene flow from RGB-D images with CNNs. In *BMVC*, 2018. 1, 2

[51] Julian Quiroga, Thomas Brox, Frédéric Devernay, and James Crowley. Dense semi-rigid scene flow estimation from RGBD images. In *ECCV*, volume 7, pages 567–582, 2014. 2

[52] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J. Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *CVPR*, pages 12240–12249, 2019. 2

[53] Zhile Ren, Orazio Gallo, Deqing Sun, Ming-Hsuan Yang, Erik B. Sudderth, and Jan Kautz. A fusion approach for multi-frame optical flow estimation. In *WACV*, pages 2077–2086, 2019. 2

[54] Zhile Ren, Deqing Sun, Jan Kautz, and Erik B. Sudderth. Cascaded scene flow prediction using semantic segmentation. In *3DV*, pages 225–233, 2017. 2

[55] Susanna Ricco and Carlo Tomasi. Dense Lagrangian motion estimation with occlusions. In *CVPR*, pages 1800–1807, 2012. 2

[56] Rohan Saxena, René Schuster, Oliver Wasenmüller, and Didier Stricker. PWOC-3D: Deep occlusion-aware end-to-end scene flow estimation. In *IV*, pages 324–331, 2019. 2

[57] René Schuster, Oliver Wasenmüller, Georg Kuschk, Christian Bailer, and Didier Stricker. SceneFlowFields: Dense interpolation of sparse scene flow correspondences. In *WACV*, pages 1056–1065, 2018. 1, 2

[58] René Schuster, Oliver Wasenmüller, Christian Unger, Georg Kuschk, and Didier Stricker. SceneFlowFields++: Multi-frame matching, visibility prediction, and robust interpolation for scene flow estimation. *Int. J. Comput. Vision*, 128(2):527–546, Feb. 2020. 2

[59] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM Network: A machine learning approach for precipitation nowcasting. In *NIPS*2015*, pages 802–810. 2, 3, 5

[60] Fridtjof Stein. Efficient computation of optical flow using the census transform. In *DAGM*, pages 79–86, 2004. 5

[61] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, pages 8934–8943, 2018. 3, 4

[62] Tatsunori Taniai, Sudipta N. Sinha, and Yoichi Sato. Fast multi-frame stereo scene flow with motion segmentation. In *CVPR*, pages 3939–3948, 2017. 2

[63] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *ICCV*, pages 722–729, 1999. 2

[64] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. *IEEE T. Pattern Anal. Mach. Intell.*, 27(3):475–480, Mar. 2005. 2

[65] Christoph Vogel, Stefan Roth, and Konrad Schindler. View-consistent 3D scene flow estimation over multiple frames. In *ECCV*, volume 4, pages 263–278, 2014. 2

[66] Christoph Vogel, Konrad Schindler, and Stefan Roth. An evaluation of data costs for optical flow. In *GCPR*, pages 343–353, 2013. 5

[67] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *ICCV*, pages 1377–1384, 2013. 2

[68] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3D scene flow estimation with a piecewise rigid scene model. *Int. J. Comput. Vision*, 115(1):1–28, Oct. 2015. 1, 2

[69] Yang Wang, Peng Wang, Zhenheng Yang, Chenxu Luo, Yi Yang, and Wei Xu. UnOS: Unified unsupervised optical-flow and stereo-depth estimation by watching videos. In *CVPR*, pages 8071–8081, 2019. 2

[70] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. FlowNet3D++: Geometric losses for deep scene flow estimation. In *WACV*, pages 91–98, 2020. 2

[71] Andreas Wedel, Thomas Brox, Tobi Vaudrey, Clemens Rabe, Uwe Franke, and Daniel Cremers. Stereoscopic scene flow computation for 3D motion understanding. *Int. J. Comput. Vision*, 95(1):29–51, Oct. 2011. 2

[72] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. PointPWC-Net: Cost volume on point clouds for (self-)supervised scene flow estimation. In *ECCV*, volume 5, pages 88–107. 2020. 1, 2

[73] Gengshan Yang and Deva Ramanan. Upgrading optical flow to 3D scene flow through optical expansion. In *CVPR*, pages 1331–1340, 2020. 1, 2, 6, 7

[74] Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, and Ram Nevatia. Every pixel counts: Unsupervised geometry learning with holistic 3D motion understanding. In *ECCV Workshops*, pages 691–709, 2018. 2, 7

[75] Zhichao Yin and Jianping Shi. GeoNet: Unsupervised learning of dense depth, optical flow and camera pose. In *CVPR*, pages 1983–1992, 2018. 2

[76] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV*, volume 2, pages 151–158, 1994. 5

[77] Ye Zhang and Chandra Kambhamettu. On 3D scene flow and structure estimation. In *CVPR*, pages 3526–3533, 2001. 2

[78] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847 [cs.CV]*, 2018. 1

[79] Alex Zihao Zhu, Wenxin Liu, Ziyun Wang, Vijay Kumar, and Kostas Daniilidis. Robustness meets deep learning: An end-to-end hybrid pipeline for unsupervised learning of egomotion. In *CVPR Workshops*, 2019. 2

[80] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. DF-Net: Unsupervised joint learning of depth and flow using cross-task consistency. In *ECCV*, volume 5, pages 36–53, 2018. 2