SENSE: a Shared Encoder Network for Scene-flow Estimation

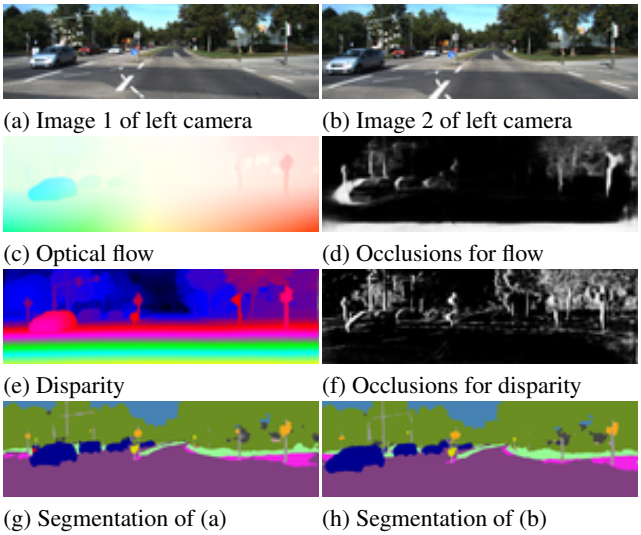
Huaizu Jiang[1†] Deqing Sun[2*] Varun Jampani[2*]
Zhaoyang Lv[3†] Erik Learned-Miller[1] Jan Kautz[2]
[1]UMass Amherst [2]NVIDIA [3]Georgia Tech

Abstract

We introduce a compact network for holistic scene flow estimation, called SENSE, which shares common encoder features among four closely-related tasks: optical flow estimation, disparity estimation from stereo, occlusion estimation, and semantic segmentation. Our key insight is that sharing features makes the network more compact, induces better feature representations, and can better exploit interactions among these tasks to handle partially labeled data. With a shared encoder, we can flexibly add decoders for different tasks during training. This modular design leads to a compact and efficient model at inference time. Exploiting the interactions among these tasks allows us to introduce distillation and self-supervised losses in addition to supervised losses, which can better handle partially labeled real-world data. SENSE achieves state-of-the-art results on several optical flow benchmarks and runs as fast as networks specifically designed for optical flow. It also compares favorably against the state of the art on stereo and scene flow, while consuming much less memory.

1. Introduction

Scene flow estimation aims at recovering the 3D structure (disparity) and motion of a scene from image sequences captured by two or more cameras [52]. It generalizes the classical problems of optical flow estimation for monocular image sequences and disparity prediction for stereo image pairs. There has been steady and impressive progress on scene flow estimation, as evidenced by results on the KITTI benchmark [39]. State-of-the-art scene flow methods outperform the best disparity (stereo) and optical flow methods by a significant margin, demonstrating the benefit of additional information in the stereo video sequences. However, the top-performing scene flow methods [5, 54] are based on the energy minimization framework [18] and are thus computationally expensive for real-time applications, such as 3D motion capture [11] and autonomous driving [27].



(a) Image 1 of left camera (b) Image 2 of left camera
(c) Optical flow (d) Occlusions for flow
(e) Disparity (f) Occlusions for disparity
(g) Segmentation of (a) (h) Segmentation of (b)

Figure 1. Given stereo videos, we train compact networks for several holistic scene understanding problems by sharing features.

Recently, a flurry of convolutional neural network (CNN)-based methods have been developed for the sub-problems of stereo and optical flow. These methods achieve state-of-the-art performance and run in real-time. However, while stereo and flow are closely-related, the top-performing networks for stereo and flow adopt significantly different architectures. Further, existing networks for scene flow stack sub-networks for stereo and optical flow together [37, 25], which does not fully exploit the structure of the two tightly-coupled problems.

As both stereo and flow rely on pixel features to establish correspondences, will the same features work for these two or more related tasks? To answer this question, we take a modular approach and build a Shared Encoder Network for Scene-flow Estimation (SENSE). Specifically, we share a feature encoder among four closely-related tasks: optical flow, stereo, occlusion, and semantic segmentation. Sharing features makes the network compact and also leads to better feature representation via multi-task learning.

The interactions among closely-related tasks further con-

†The work was begun while the author was an intern at NVIDIA.
*Currently affiliated with Google.
Code is available at: https://github.com/NVlabs/SENSE

strain the network training, ameliorating the issue of sparse ground-truth annotations for scene flow estimation. Unlike many other vision tasks, it is inherently difficult to collect ground-truth optical flow and stereo for real-world data. Training data-hungry deep CNNs often relies on synthetic data [7, 10, 37], which lacks the fine details and diversity ubiquitous in the real world. To narrow the domain gap, fine-tuning on real-world data is necessary, but the scarcity of annotated real-world data has been a serious bottleneck for learning CNN models for scene flow.

To address the data scarcity issue, we introduce a semi-supervised loss for SENSE by adding distillation and self-supervised loss terms to the supervised losses. First, no existing dataset provides ground truth annotations for all the four tasks we address. For example, the KITTI benchmark has no ground truth annotations for occlusion and semantic segmentation.[1] Thus, we train separate models for tasks with missing ground truth annotations using other annotated data, and use the pre-trained models to "supervise" our network on the real data via a distillation loss [17]. Second, we use self-supervision loss terms that encourage corresponding visible pixels to have similar pixel values and semantic classes, according to either optical flow or stereo. The self-supervision loss terms tightly couple the four tasks together and are critical for improvement in regions without ground truth, such as sky regions.

Experiments on both synthetic and real-world benchmark datasets demonstrate that SENSE achieves state-of-the-art results for optical flow, while maintaining the same run-time efficiency as specialized networks for flow. It also compares favorably against state of the art on disparity and scene flow estimation, while having a much smaller memory footprint. Ablation studies confirm the utility of our design choices, and show that our proposed distillation and self-supervised loss terms help mitigate issues with partially labeled data.

To summarize, we make the following contributions:

- We introduce a modular network design for holistic scene understanding, called SENSE, to integrate optical flow, stereo, occlusion, and semantic segmentation.
- SENSE shares an encoder among these four tasks, which makes networks compact and also induces better feature representation via multi-task learning.
- SENSE can better handle partially labeled data by exploiting interactions among tasks in a semi-supervised approach; it leads to qualitatively better results in regions without ground-truth annotations.
- SENSE achieves state-of-the-art flow results while running as fast as specialized flow networks. It compares favorably against state of the art on stereo and scene flow, while consuming much less memory.

[1] Segmentation is only available for left images of KITTI 2015 [1].

2. Related Work

A comprehensive survey of holistic scene understanding is beyond our scope and we review the most relevant work.

Energy minimization for scene flow estimation. Scene flow was first introduced by Vedula *et al*. [52] as the dense 3D motion of all points in an observed scene from several calibrated cameras. Several classical methods adopt energy minimization approaches, such as joint recovery of flow and stereo [20] and decoupled inference of stereo and flow for efficiency [56]. Compared with optical flow and stereo, the solution space of scene flow is of higher dimension and thus more challenging. Vogel *et al*. [53] reduce the solution space by assuming a scene flow of piecewise rigid moving planes over superpixels. Their work first tackles scene flow from a holistic perspective and outperforms contemporary stereo and optical flow methods by a large margin on the KITTI benchmark [12].

Joint scene understanding. Motion and segmentation are chicken-and-egg problems: knowing one simplifies the other. While the layered approach has long been regarded as an elegant solution to these two problems [55], existing solutions tend to get stuck in local minima [47]. In the motion segmentation literature, most methods start from an estimate of optical flow as input, and segment the scene by jointly estimating (either implicitly or explicitly) camera motion, object motion, and scene appearance, e.g. [6, 51]. Lv *et al*. [35] show that motion can be segmented directly from two images, without first calculating optical flow. Taylor *et al*. [50] demonstrate that occlusion can also be a useful cue.

Exploiting advances in semantic segmentation, Sevilla *et al*. [46] show that semantic information is good enough to initialize the layered segmentation and thereby improves optical flow. Bai *et al*. [2] use instance-level segmentation to deal with a small number of traffic participants. Hur and Roth [22] jointly estimate optical flow and temporally consistent semantic segmentation and obtain gains on both tasks. The object scene flow algorithm [39] segments a scene into independently moving regions and enforces superpixels within each region to have similar 3D motion. The "objects" in their model are assumed to be planar and initialized via bottom-up motion estimation. Behl *et al*. [5], Ren *et al*. [42], and Ma *et al*. [36] all show that instance segmentation helps scene flow estimation in the autonomous setting. While assuming a rigid motion for each individual instance works well for cars, this assumption tends to fail in general scenes, such as Sintel, on which our holistic approach achieves state-of-the-art performance.

The top-performing energy-based approaches are too computationally expensive for real-time applications. Here we present a compact CNN model to holistically reason about geometry (disparity), motion (flow), and semantics, which runs much faster than energy-based approaches.

End-to-end learning of optical flow and disparity. Recently CNN based methods have made significant progress on optical flow and disparity, two sub-problems of scene flow estimation. Dosovitskiy *et al*. [10] first introduce two CNN models, FlowNetS and FlowNetC, for optical flow and bring about a paradigm shift to optical flow and disparity estimation. Ilg *et al*. [24] propose several technical improvements, such as dataset scheduling and stacking basic models into a big one, *i.e*., FlowNet2. FlowNet2 has near real-time performance and obtains competitive results against hand-designed methods. Ilg *et al*. [25] stack networks for flow, disparity together for the joint task of scene flow estimation. However, there is no information sharing between the networks for flow and disparity. Ranjan and Black [41] introduce a spatial pyramid network that performs on par with FlowNetC but has more than 100 times fewer parameters, due to the use of two classical principles: pyramids and warping. Sun *et al*. [48] develop a compact yet effective network, called PWC-Net, which makes frequent use of three principles to construct the network: pyramids of learnable features, warping operations, and cost volume processing. PWC-Net obtains state-of-the-art performance on two major optical flow benchmarks.

The FlowNet work also inspired new CNN models for stereo estimation [30, 8, 60]. Kendall *et al*. [30] concatenate features to construct the cost volume, followed by 3D convolutions. The 3D convolution becomes commonly-used for stereo but is computationally expensive in speed and memory. Chang and Chen [8] introduce a pyramid pooling module to exploit context information for establishing correspondences in ambiguous regions. Yang *et al*. [60] incorporate semantic cues to tackle textureless regions. Yin *et al*. cast optical flow and disparity estimations as probabilistic distribution matching problems [61] to provide uncertainty estimation. They do not exploit the shared encoder of the two tasks as we do.

Existing scene flow networks [25, 36, 38] stack independent networks for disparity and flow together. We are interested in exploiting the interactions among multiple related tasks to design a compact and effective network for holistic scene understanding. Our holistic scene flow network performs favorably against state of the art while being faster for inference and consuming less memory. In particular, we show the benefit of sharing the feature encoder between different tasks, such as flow and disparity.

Self-supervised learning from videos. Supervised learning often uses synthetic data, as it is hard to obtain ground truth optical flow and disparity for real-world videos. Recently self-supervised learning methods have been proposed to learn scene flow by minimizing the data matching cost [65] or interpolation errors [29, 32]. However, the self-supervised methods have not yet achieved the performance of their supervised counterparts.

3. Semi-Supervised Scene Flow Estimation

We follow the problem setup of the KITTI scene flow benchmark [39], as illustrated in Fig. 2. The inputs are two stereo image pairs over time $\left(\mathbf{I}^{1,l}, \mathbf{I}^{2,l}, \mathbf{I}^{1,r}, \mathbf{I}^{2,r}\right)$, where the first number in the superscript indicates the time step and the second symbol denotes the left or right camera. To save space, we will omit the superscript if the context is clear. We want to estimate optical flow $\mathbf{F}^{1,l}$ from the first left image to the second left image and disparity $\mathbf{D}^{1,l}$ and $\mathbf{D}^{2,l}$ from the left image to the right image at the first and second frames, respectively. We also consider occlusion between two consecutive frames $\mathbf{O}_F^{1,l}$ and between the two sets of stereo images $\mathbf{O}_D^{1,l}$ and $\mathbf{O}_D^{2,l}$, as well as semantic segmentation for the reference (first left) image, *i.e*., $\mathbf{S}^{1,l}$. These extra outputs introduce interactions between different tasks to impose more constraints in the network training. Further, we hypothesize that sharing features among these closely-related tasks induces better feature representations.

We will first introduce our modular network design in Section 3.1, which shares an encoder among different tasks and supports flexible configurations during training. We will then explain our semi-supervised loss function in Section 3.2, which enables learning with partially labeled data.

3.1. Modular Network Design

To enable feature sharing among different tasks and allow flexible configurations during training, we design the network in a modular way. Specifically, we build our network on top of PWC-Net [48], a compact network for optical flow estimation. PWC-Net consists of an encoder and a decoder, where the encoder takes the input images and extracts features at different hierarchies of the network. The decoder is specially designed with domain knowledge of optical flow. The encoder-decoder structure allows us to design a network in a modular way, with a single shared encoder and several decoders for different tasks.

Shared encoder. The original encoder of PWC-Net, however, is not well-suited to multiple tasks because of its small capacity. More than 80% of the parameters of PWC-Net are concentrated in the decoder, which uses DenseNet [19] blocks at each pyramid level. The encoder consists of plain convolutional layers and uses fewer than 20% of the parameters. While sufficient for optical flow, the encoder does not work well enough for disparity estimation. To make the encoder versatile for different tasks, we make the following modifications. First, we reduce the number of feature pyramid levels from 6 to 5, which reduces the number of parameters by nearly 50%. It also allows us to borrow the widely-used 5-level ResNet-like encoder architecture [8, 16], which has been proven to be effective in a variety of vision tasks. Specifically, we replace plain CNN layers with residual blocks [16] and add Batch Normaliza-

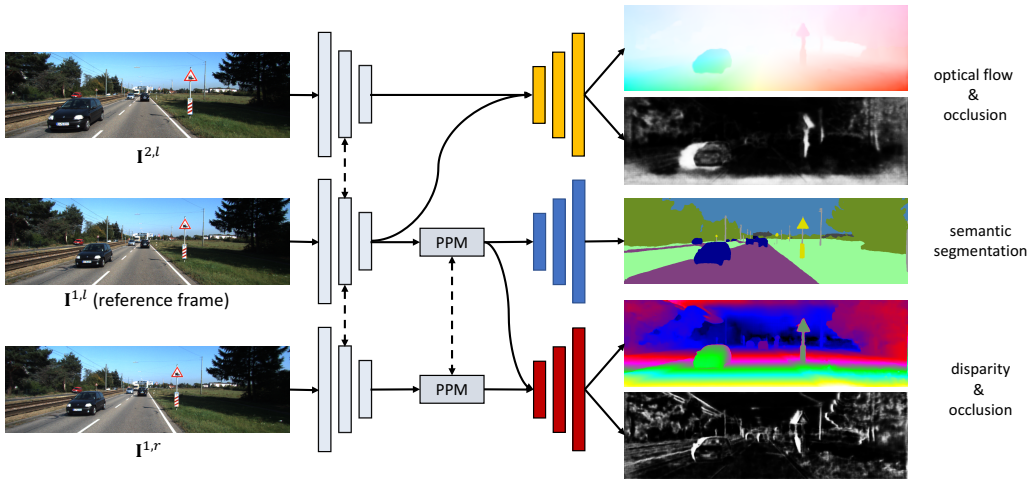


Figure 2. Illustration of network design. Dashed arrows indicate shared weights. We have a single encoder for all input images and all different tasks and keep different decoders for different tasks. On the right, from top to bottom are: optical flow, forward occlusion mask, semantic segmentation, disparity, and disparity occlusion. The PPM (Pyramid Pooling Module) is not helpful for optical flow estimation. But thanks to the modular network design, we can flexibly configure the network.

tion layers [26] in both encoder and decoder. With these modifications, the new model has slightly fewer parameters but gives better disparity estimation results and also better flow (Table 1).

Decoder for disparity. Next we explain how to adapt PWC-Net to disparity estimation between two stereo images. Disparity is a special case of optical flow computation, with correspondences lying on a horizontal line. As a result, we need only to build a 1D cost volume for disparity, while the decoder of the original PWC-Net constructs a 2D cost volume for optical flow. Specifically, for optical flow, a feature at $p=(x,y)$ in the first feature map is compared to features at $q \in [x-k, x+k]\times[y-k, y+k]$ in the warped second feature map. For disparity, we need only to search for correspondences by comparing p in the left feature map to $q \in [x-k, x+k]\times y$ in the warped right feature map. We use $k=4$ for both optical flow and disparity estimations. Across the feature pyramids, our decoder for disparity adopts the same warping and refinement process as PWC-Net.

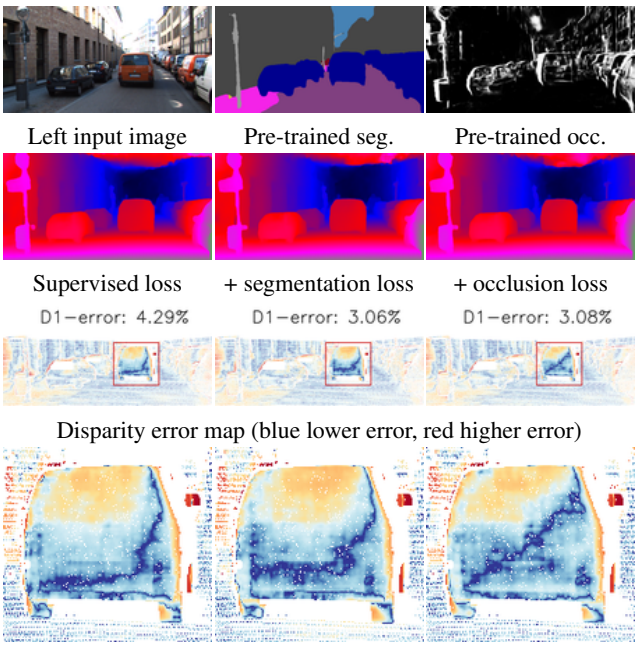
To further improve disparity estimation accuracy, we investigate more design choices. First, we use the Pyramid Pooling Module (PPM) [64] to aggregate the learned features of input images across multiple levels. Second, the decoder outputs a disparity map one fourth the size of the input resolution, which tends to have blurred disparity boundaries. As a remedy, we add a simple hourglass module widely used in disparity estimation [8]. It takes a twice upsampled disparity, a feature map of the first image, and a warped feature map of the second image to predict a residual disparity that is added to the upsampled disparity. Both the PPM and hourglass modifications lead to significant improvements in disparity estimation. They are not helpful for optical flow estimation though, indicating that the original PWC-Net is well designed for optical flow. The modular design allows us to flexibly configure networks that work for different tasks, as shown in Fig. 2.

Decoder for segmentation. To introduce more constraints to network training, we also consider semantic segmentation. It encourages the encoder to learn some semantic information, which may help optical flow and disparity estimations. For semantic segmentation decoder, we use the UPerNet [58] for its simplicity.

Occlusion estimation. For occlusion predictions, we add sibling branches to optical flow or disparity decoders to perform pixel-wise binary classification, where 1 means fully occluded. Adding such extra modules enables holistic scene understanding that helps us to induce better feature representations in the shared encoder and use extra supervision signals for network training to deal with partially labeled data, which is discussed in Section 3.2. Critically, for scene flow estimation, the shared encoder results in a more compact and efficient model. For optical flow and disparity estimations, we can combine modules as needed during training, with no influence on the inference time. For scene flow estimation, extra modules can be used optionally, depending on configuration. See explanations in Section 4.2.

3.2. Semi-Supervised Loss

No fully labeled datasets are available to directly train our holistic scene flow network. For example, KITTI has no ground-truth occlusion masks. Even for optical flow and disparity ground-truths, only around 19% of pixels of the KITTI data have annotations due to the difficulty in data capturing. The synthetic SceneFlow dataset [38] has



Enlarged view of error map for the car (best viewed in color)

Figure 3. Effects of adding distillation losses for semantic segmentation (middle) and occlusion (right) to the supervised loss.

no ground truth for semantic segmentation. To address these issues, we introduce our semi-supervised loss functions, which consist of supervised, distillation, and self-supervised loss terms.

Supervised loss. When corresponding ground-truth annotations are available, we define our supervised loss as

$$\mathcal{L}_{sp} = (\mathcal{L}_F + \mathcal{L}_{O_F}) + (\mathcal{L}_D + \mathcal{L}_{O_D}), \tag{1}$$

where $\mathcal{L}_F$ and $\mathcal{L}_{O_F}$ are loss terms for estimating optical flow and its corresponding occlusion. $\mathcal{L}_D$ and $\mathcal{L}_{O_D}$ are the loss terms for estimating disparity and its corresponding occlusion. $\mathcal{L}_F$ is defined across multiple pyramid levels as

$$\mathcal{L}_F = \sum_{i=1}^{N_F} \omega_i \sum_p \rho\left(\mathbf{F}_i(p), \hat{\mathbf{F}}_i(p)\right), \tag{2}$$

where ω_i denotes optical flow and disparity weights at pyramid level i, N_F is the number of pyramid levels, and $\rho(\cdot,\cdot)$ is a loss function measuring the similarity between the ground-truth $\mathbf{F}_i(p)$ and estimated optical flow $\hat{\mathbf{F}}_i(p)$ at pixel p. Disparity and occlusion loss functions, $\mathcal{L}_D$, $\mathcal{L}_{O_F}$, and $\mathcal{L}_{O_D}$ are defined in a similar way. We use L_2 and `smooth_l1` [13, 8] loss for optical flow and disparity estimations, respectively. For the occlusions, we use binary cross entropy loss when ground-truth annotations are available (*e.g.*, on FlyingThings3D [37]). For semantic segmentation, only ground-truth annotations of the left images are available for KITTI2015. We empirically found using distillation loss only introduced below yields better accuracy.

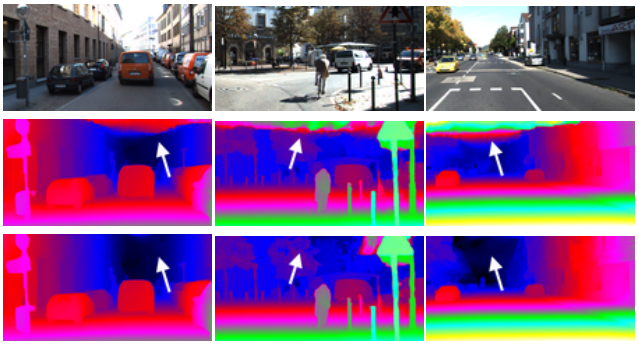


Figure 4. Illustration of effectiveness of self-supervised loss. From top to bottom: input images, disparity estimations *without* using self-supervised loss, and disparity estimations *with* using self-supervised loss. We can see self-supervised loss helps greatly reduce artifacts in the sky region.

Distillation loss. For occlusion estimation and semantic segmentation tasks, ground-truth annotations are not always available. They are important, however, during network training. For instance, on KITTI, supervised loss can only be computed on sparsely annotated pixels. Adding extra supervision for occlusion estimation is helpful for the network to extrapolate optical flow and disparity estimations to regions where ground-truth annotations are missing, yielding visually appealing results.

We find the occlusion estimations provided by a pre-trained model on synthetic data are reasonably good, as shown in Fig. 3. As a soft supervision, we encourage the occlusion estimations of the network during training do not deviate much from what it learned in the pre-training stage. Therefore, we simply use the estimations of a pre-trained network as pseudo ground-truth and `smooth_l1` loss function during training, computed in multiple pyramid levels as $\mathcal{L}_F$ and $\mathcal{L}_D$. Adding extra supervision using distillation loss for occlusion is helpful for reducing artifacts in disparity estimation, as shown in Fig. 3.

For semantic segmentation, we use the distillation loss formulation proposed in [17]. Specifically, semantic segmentation distillation loss $\mathcal{L}_{S_d}$ for a single pixel p (omitted here for simplicity) is defined as

$$\mathcal{L}_{S_d} = T \sum_{i=1}^{C} \tilde{y}_i \log \hat{y}_i, \quad \tilde{y}_i = \frac{\exp^{-z_i/T}}{\sum_k \exp^{-z_k/T}}, \tag{3}$$

where C is the number of segmentation categories. z_i and $\tilde{y}_i$ come from a more powerful teacher segmentation model, where z_i is the output for the i-th category right before the `softmax` layer, also known as `logit`. $\tilde{y}_i$ is "softened" posterior probability for the i-th category, controlled by the hyper-parameter T [17]. We empirically found $T=1$ works well on a validation set. $\hat{y}_i$ is the estimated posterior probability of our model. The distillation is aggregated over all pixels in training images.

Self-supervised loss. To further constrain the network training, we also define self-supervised loss. Optical flow and disparity are defined as correspondence between two input images. We can therefore compare two corresponding pixels defined by either optical flow or disparity as supervision for network training.

The most straightforward metric is to compare values between two corresponding pixels that are visible in both frames, known as photometric consistency. In a single pyramid level, it is defined as $\mathcal{L}_{PC} =$

$$\|\mathbf{I}^l - g(\mathbf{I}^r, \mathbf{D}^l)\|_1 \odot \bar{\mathbf{O}}_D + \|\mathbf{I}^1 - g(\mathbf{I}^2, \mathbf{F}^1)\|_1 \odot \bar{\mathbf{O}}_F, \quad (4)$$

where $g(\cdot,\cdot)$ is the differentiable warping function, $\bar{\mathbf{O}} = 1 - \mathbf{O}$, $\odot$ denotes element-wise multiplication followed by summation, and we omit some superscripts when the context is clear. This loss term reasons about occlusion by modulating the consistency loss using the occlusion map and tightly couples occlusion with optical flow and stereo.

As photometric consistency is not robust to lighting changes, we further introduce semantic consistency, encouraging two corresponding pixels to have similar semantic segmentation posterior probability. Specifically, this semantic consistency is defined as $\mathcal{L}_{SC} =$

$$\|\tilde{\mathbf{y}}^l - g(\tilde{\mathbf{y}}^r, \mathbf{D}^l)\|_1 \odot \bar{\mathbf{O}}_D + \|\tilde{\mathbf{y}}^1 - g(\tilde{\mathbf{y}}^2, \mathbf{F}^1)\|_1 \odot \bar{\mathbf{O}}_F, \quad (5)$$

where $\tilde{\mathbf{y}}$ denotes a posterior probability image coming from the teacher segmentation network used in Eq.(3). Unlike raw pixel values, the segmentation posterior probability is more robust to lighting changes.

Finally, we consider the structural similarity loss

$$\mathcal{L}_{SS} = \gamma_D \left(1 - SS(\mathbf{I}^l, \mathbf{I}^l \otimes \mathbf{O}_D + g(\mathbf{I}^r, \mathbf{D}^l) \otimes \bar{\mathbf{O}}_D)\right) + \gamma_F \left(1 - SS(\mathbf{I}^1, \mathbf{I}^1 \otimes \mathbf{O}_F + g(\mathbf{I}^2, \mathbf{F}^1) \otimes \bar{\mathbf{O}}_F)\right), \quad (6)$$

where $\otimes$ indicates element-wise multiplications only. $SS(\cdot,\cdot)$ is a differentiable function that outputs a single scalar value to measure the structural similarity between two input images [63]. Note that for occluded pixels in the warped image, their values are replaced with values of pixels at the same position in the left/first image.

There exist trivial solutions for minimizing Eq.(4) and Eq.(5) by setting $\mathbf{O}_D$ and $\mathbf{O}_F$ to all ones. We thus add regularization terms

$$\mathcal{L}_{REG} = \beta_D \sum_p \mathbf{O}_D(p) + \beta_F \sum_p \mathbf{O}_F(p), \quad (7)$$

Although the self-supervised photometric and structural similarity loss terms have been studied in previous work [28, 14], our definition differs from theirs in that we model occlusions. On one hand, we avoid defining loss terms in the occluded regions. On the other hand, these self-supervised terms provide modulation for the occlusion estimation as well. Thus, our networks tightly couple these four closely-related tasks together.

Our final semi-supervised loss consists of supervised, distillation, and self-supervised loss terms. More details can be found in the supplementary material.

Table 1. Average EPE results on MPI Sintel optical flow dataset. "-ft" means fine-tuning on the MPI Sintel *training* set and the numbers in parentheses are results on the data the methods have been fine-tuned on.

Methods	Training		Test		Time
	Clean	Final	Clean	Final	(s)
FlowFields [3]	-	-	3.75	5.81	28.0
MRFlow [57]	1.83	3.59	**2.53**	5.38	480
FlowFieldsCNN [4]	-	-	3.78	5.36	23.0
DCFlow [59]	-	-	3.54	5.12	8.60
SpyNet-ft [41]	(3.17)	(4.32)	6.64	8.36	0.16
FlowNet2 [24]	2.02	3.14	3.96	6.02	0.12
FlowNet2-ft [24]	(1.45)	(2.01)	4.16	5.74	0.12
LiteFlowNet [21]	(1.64)	(2.23)	4.86	6.09	0.09
PWC-Net [48]	2.55	3.93	-	-	**0.03**
PWC-Net-ft [48]	(1.70)	(2.21)	3.86	5.13	**0.03**
FlowNet3 [25]	2.08	3.94	3.61	6.03	0.07
FlowNet3-ft [25]	(**1.47**)	(2.12)	4.35	5.67	0.07
SENSE	1.91	3.78	-	-	**0.03**
SENSE-ft	(1.54)	(**2.05**)	3.60	**4.86**	**0.03**

4. Experiments

4.1. Implementation Details

Pre-training of stereo and optical flow. We use the synthetic SceneFlow dataset [37], including FlyingThings3D, Monkaa, and Driving, for pre-training. All three datasets contain optical flow and disparity ground-truth. Occlusion labels are only available in FlyingThings3D. During training, we uniformly sample images from all three datasets and compute occlusion loss when the ground-truths are available. During training, we use color jittering for both optical flow and disparity training. Additionally, we use random crops and vertical flips for stereo training images. The crop size is 256×512. For optical flow training images, we perform extensive data augmentations including random crop, translation, rotation, zooming, squeezing, and horizontal and vertical flip, where the crop size is 384×640. The network is trained for 100 epochs with a batch size of 8 using the Adam optimizer [31]. We use synchronized Batch Normalization [58] to ensure there are enough training samples for estimating Batch Normalization layers' statistics when using multiple GPUs. The initial learning rate is 0.001 and decreased by factor of 10 after 70 epochs.

Fine-tuning. For Sintel, we use a similar learning rate schedule as used in [48]. On KITTI 2012 [12] and KITTI 2015 [40] , we use longer learning rate schedule, where the model is trained for 1.5K epochs with an initial learning rate is 0.001. We perform another 1K-epoch training with an ini-

Table 2. Results on the KITTI optical flow dataset. "-ft" means fine-tuning on the KITTI *training* set and the numbers in the parenthesis are results on the data the methods have been fine-tuned on.

Methods	KITTI 2012 AEPE *train*	KITTI 2012 AEPE *test*	KITTI 2012 Fl-Noc *test*	KITTI 2015 AEPE *train*	KITTI 2015 Fl-all *train*	KITTI 2015 Fl-all *test*
FlowFields [3]	-	-	-	-	-	19.80%
MRFlow [57]	-	-	-	-	14.09 %	12.19 %
DCFlow [59]	-	-	-	-	15.09 %	14.83 %
SDF [2]	-	2.3	3.80%	-	-	11.01 %
MirrorFlow [23]	-	2.6	4.38%	-	9.93%	10.29%
SpyNet-ft [41]	(4.13)	4.7	12.31%	-	-	35.07%
FlowNet2 [24]	4.09	-	-	10.06	30.37%	-
FlowNet2-ft [24]	(1.28)	1.8	4.82%	(2.30)	(8.61%)	10.41 %
LiteFlowNet [21]	(1.26)	1.7	-	(2.16)	(8.16%)	10.24 %
PWC-Net [48]	4.14	-	-	10.35	33.67%	-
PWC-Net-ft [48]	(1.45)	1.7	4.22%	(2.16)	(9.80%)	9.60%
FlowNet3 [25]	3:69	-	-	9.33	-	-
FlowNet3-ft [25]	(1.19)	-	3.45%	(**1.79**)	-	8.60%
SENSE	2.55	-	-	6.23	23.29%	-
SENSE-ft	(**1.14**)	**1.5**	**3.00**%	(2.01)	(**9.20**%)	8.38%
SENSE-ft+semi	(1.18)	1.5	3.03%	(2.05)	(9.69%)	**8.16**%

Table 3. Results on KITTI stereo datasets (test set).

Methods	KITTI 2012 All Out-All	KITTI 2012 Non-Occ Out-Noc	KITTI 2015 All D1-fg	KITTI 2015 All D1-all	KITTI 2015 Non-Occ D1-fg	KITTI 2015 Non-Occ D1-all	Time (s)
Content-CNN [33]	3.07	4.29	8.58	4.54	7.44	4.00	1.0
DispNetC [37]	-	-	4.41	4.34	3.72	4.05	**0.06**
MC-CNN [62]	2.43	3.63	8.88	3.89	7.64	3.33	67
PBCP [45]	2.36	3.45	8.74	3.61	7.71	3.17	68
Displets v2 [15]	2.37	3.09	5.56	3.43	4.95	3.09	265
GC-Net [30]	1.77	2.30	6.16	2.87	5.58	2.61	0.9
PSMNet [8]	**1.49**	**1.89**	4.62	2.32	4.31	2.14	0.41
SegStereo [60]	1.68	2.03	3.70	**2.08**	4.07	2.25	0.6
FlowNet3 [25]	1.82	-	-	2.19	-	-	0.07
SENSE	1.77	2.18	3.13	2.33	2.79	2.13	**0.06**
SENSE+semi	1.73	2.16	**3.01**	2.22	**2.76**	**2.05**	**0.06**

tial learning rate of 0.0002. We use a crop size of 320×768 for both disparity and optical flow training images and a batch size of 8. More training details are provided in the supplementary material due to limited space here.

Training semantic segmentation. We jointly train all parts of the entire network, including pre-trained encoder and decoders for optical flow and disparity, as well as a randomly initialized segmentation decoder. We empirically found using a randomly initialized segmentation decoder yields better performance.

For the segmentation distillation loss and semantic consistency loss computation, we first train the teacher segmentation model. We use the ResNet101-UPerNet [58] pre-trained on CityScapes [9] using its training set with fine annotations only, which achieves 75.4% IoU on the validation set. We fine-tune the model on KITTI 2015 [1], where the segmentation annotations, consistent with CityScapes' annotation style, for the left images are provided.

Table 4. Results on KITTI2015 Scene flow dataset. CNN-based approaches need to deal with refinement of D2, where N and R indicates network and rigidity-based refinement, respectively.

Method	D1-all	D2-all	Fl-all	SF-all	D2 ref.	Time (s)
ISF [5]	4.46	5.95	6.22	8.08	-	600
CSF [34]	5.98	10.06	12.96	15.71	-	80
SGM+FF[43]	13.37	27.80	22.82	33.57	-	29
SceneFF[44]	6.57	10.69	12.88	15.78	-	65
FlowNet3 [25]	2.16	6.45	8.60	11.34	N	0.25
SENSE	2.23	7.37	8.38	11.71	N	0.16
SENSE+semi	2.22	6.57	8.16	11.35	N	0.16
SENSE+semi	2.22	5.89	7.64	9.55	R+N	0.32

4.2. Main Results

Optical flow results. Table 1 shows the results for optical flow estimation on the MPI Sintel benchmark dataset. Our approach outperforms CNN-based approaches without or with fine-tuning. On the more photorealistic (final) pass of the test set, which involves more rendering details such as lighting change, shadow, motion blur, etc, our approach outperforms both CNN-based and traditional hand-designed approaches by a large margin.

Table 2 shows the results on both KITTI2012 and KITTI2015. Our approach significantly outperforms both hand-designed and CNN-based approaches on KITTI 2012 with and without fine-tuning. On KITTI 2015, our model achieves much lower error rates than CNN-based approaches without pre-training (including ours). After fine-tuning, it outperforms all other approaches.

We note that better optical flow results are reported in an improved version of PWC-Net [49], which uses FlyingChairs followed by FlyingThings3D for pre-training. It also uses much longer learning rate schedules for fine-tuning, so the results are not directly comparable to ours.

Disparity results. For disparity estimation, SENSE significantly outperforms previous CNN-based approaches including DispNetC [37] and GC-Net [30] and achieves comparable accuracy with state-of-the-art approaches like PSMNet [8], SegStereo [60], and FlowNet3 [25]. Notably, our approach performs the best on the foreground region in both all and non-occluded regions on KITTI2015.

Scene flow results. Table 4 shows Scene flow results on KITTI 2015. SENSE performs the best in general CNN-based scene flow methods, compared to FlowNet3 [25]. Compared to ISF [5], SENSE is 2K times faster and can handle general nonrigid scene motions.

To remove artifacts introduced by the second frame disparity warping operation, we use a refinement network of a encoder-decoder structure with skip connections. It takes $\mathbf{I}^{1,l}$, $\mathbf{O}_F^{1,l}$, $\mathbf{D}^{1,l}$, and $g(\mathbf{D}^{2,l}, \mathbf{F}^{1,l})$ to generate a residual that is added to the warped disparity. From our holistic outputs,

Table 5. Effectiveness of different tasks.

Tasks			Results		
flow	disp	seg	flow (F1-occ) ↓	disp (D1-occ) ↓	seg (mIoU) ↑
✓			11.37%	-	-
	✓		-	2.73%	-
		✓	-	-	47.51%
✓	✓		11.59%	2.61%	-
✓		✓	11.39%	-	49.54%
	✓	✓	-	2.62%	49.12%
✓	✓	✓	11.19%	2.59%	48.25%

Table 6. Ablation study of different loss terms.

Distillation		Self-supervised			Flow	Disp	Seg
seg.	occ.	sem.	pho.	ss	F1-Occ↓	D1-Occ↓	mIoU↑
					11.16%	2.52%	-
✓					10.96%	2.44%	51.48%
	✓				11.07%	2.38%	-
✓	✓				11.17%	2.33%	51.26%
		✓			11.11%	2.38%	-
			✓		11.04%	2.55%	-
				✓	11.16%	2.47%	-
		✓	✓	✓	11.21%	2.58%	-
✓	✓	✓	✓	✓	11.12%	2.49%	50.92%

we can refine the background scene flow using a rigidity refinement step. We first determine the static rigid areas according to semantic segmentation outputs. We then calculate the ego-motion flow by minimizing the geometry consistency between optical flow and disparity images using the Gauss-Newton algorithm. Finally, we compute the warped scene flow using the disparity of the reference frame and the ego-motion to substitute the raw scene flow only in the rigid background region. This step additionally produces camera motion and better scene flow with minimal costs. Details of refinement steps are provided in supplementary material.

Running time. SENSE is an efficient model. SENSE takes 0.03s to compute optical flow between two images of size 436×1024. For disparity, SENSE is an order of magnitude faster than PSMNet and SegStereo, and slightly faster than FlowNet3. For scene flow using KITTI images, SENSE takes 0.15s to generate one optical flow and two disparity maps. The additional warping refinement network takes 0.01s and the rigidity refinement takes 0.15s.

Model size and memory. SENSE is small in size. It has only 8.8M parameters for the optical flow model, and 8.3M for the disparity model. The scene flow model with shared encoder has 13.4M parameters. In contrast, FlowNet3 has a flow model (117M) and a disparity model (117M), which is 20 times larger. SENSE also has a low GPU memory footprint. FlowNet3 costs 7.4GB while SENSE needs 1.5GB RAM only. Although PSMNet has fewer parameters (5.1M), it costs 4.2GB memory due to 3D convolutions.

4.3. Ablation Studies

Performance of different tasks. We report results of different tasks using different combinations of encoder and decoders. Our models are trained using 160 images of KITTI 2015 with a half of the aforementioned learning rate schedule. Results are reported on the rest 40 images in Table 5. We can see that the shared encoder model performs better than models trained separately.

Semi-supervised loss. To study the effects of distillation and self-supervised loss terms, we perform ablation studies using all images of KITTI 2012 and 160 images of KITTI 2015 for training with a half of full learning rate schedule. The rest 40 ones of KITTI 2015 are used for testing. We finetune the baseline model using sparse flow and disparity annotations only. Table 6 shows the quantitative comparisons and Fig. 4 highlights the effects qualitatively.

Regarding distillation loss, both segmentation and occlusion distillation loss terms are useful for disparity and optical flow estimation. However, distillation loss is not helpful for reducing the artifacts in sky regions. Thus, the self-supervised loss is essential, as shown in Fig. 4, though quantitatively self-supervised loss is not as effective as the distillation loss. Finally, combining all loss terms yields the best optical flow and disparity accuracies. We also test SENSE trained using semi-supervised loss on KITTI, as summarized in Tables 2, 3, and 4. We can see it improves disparity and optical flow accuracy on KITTI 2015 and also leads to better disparity on KITTI 2012.

5. Conclusion

We have presented a compact network for four closely-related tasks in holistic scene understanding: Sharing an encoder among these tasks not only makes the network compact but also improves performance by exploiting the interactions among these tasks. It also allows us to introduce distillation and self-supervision losses to deal with partially labeled data. Our holistic network has similar accuracy and running time as specialized networks for optical flow. It performs favorably against state-of-the-art disparity and scene flow methods while being much faster and memory efficient. Our work shows the benefits of synergizing closely-related tasks for holistic scene understanding and we hope the insights will aid new research in this direction.

Acknowledgement

Huaizu Jiang and Erik Learned-Miller acknowledge support from AFRL and DARPA (#FA8750- 18-2-0126) and the MassTech Collaborative grant for funding the UMass GPU cluster. The U.S. Gov. is authorized to reproduce and distribute reprints for Gov. purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the AFRL and DARPA or the U.S. Gov.

References

[1] Hassan Alhaija, Siva Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *IJCV*, 2018. 2, 7

[2] Min Bai, Wenjie Luo, Kaustav Kundu, and Raquel Urtasun. Exploiting semantic information and deep matching for optical flow. In *Proc. ECCV*, 2016. 2, 7

[3] Christian Bailer, Bertram Taetz, and Didier Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *Proc. ICCV*, 2015. 6, 7

[4] Christian Bailer, Kiran Varanasi, and Didier Stricker. CNN-based patch matching for optical flow with thresholded hinge embedding loss. In *Proc. CVPR*, 2017. 6

[5] 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 *Proc. ICCV*, 2017. 1, 2, 7

[6] Pia Bideau, Aruni RoyChowdhury, Rakesh R Menon, and Erik Learned-Miller. The best of both worlds: Combining cnns and geometric constraints for hierarchical motion segmentation. In *Proc. CVPR*, pages 508–517, 2018. 2

[7] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *Proc. ECCV*, 2012. 2

[8] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proc. CVPR*, 2018. 3, 4, 5, 7

[9] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. CVPR*, 2016. 7

[10] Alexey Dosovitskiy, Philipp Fischery, Eddy Ilg, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, Thomas Brox, et al. FlowNet: Learning optical flow with convolutional networks. In *Proc. ICCV*, 2015. 2, 3

[11] Yasutaka Furukawa and Jean Ponce. Dense 3d motion capture from synchronized video streams. In *Image and Geometry Processing for 3-D Cinematography*, pages 193–211, 2010. 1

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. CVPR*, pages 3354–3361. IEEE, 2012. 2, 6

[13] Ross B. Girshick. Fast R-CNN. In *Proc. ICCV*, 2015. 5

[14] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proc. CVPR*, 2017. 6

[15] Fatma Guney and Andreas Geiger. Displets: Resolving stereo ambiguities using object knowledge. In *Proc. CVPR*, 2015. 7

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016. 3, 13

[17] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. 2, 5

[18] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 1981. 1

[19] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proc. CVPR*, 2017. 3

[20] Frédéric Huguet and Frédéric Devernay. A variational method for scene flow estimation from stereo sequences. In *Proc. ICCV*, 2007. 2

[21] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proc. CVPR*, 2018. 6, 7

[22] Junhwa Hur and Stefan Roth. Joint optical flow and temporally consistent semantic segmentation. In *Proc. ECCV*. Springer, 2016. 2

[23] Junhwa Hur and Stefan Roth. MirrorFlow: Exploiting symmetries in joint optical flow and occlusion estimation. In *Proc. ICCV*, Oct 2017. 7

[24] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proc. CVPR*, 2017. 3, 6, 7

[25] 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 *Proc. ECCV*, 2018. 1, 3, 6, 7, 11

[26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015. 3

[27] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *arXiv preprint arXiv:1704.05519*, 2017. 1

[28] J Yu Jason, Adam W Harley, and Konstantinos G Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *ECCV*, pages 3–10. Springer, 2016. 6

[29] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In *Proc. CVPR*, 2018. 3

[30] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proc. ICCV*, 2017. 3, 7

[31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015. 6, 11

[32] Ziwei Liu, Raymond Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *Proc. ICCV*, 2017. 3

[33] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *Proc. CVPR*, 2016. 7

[34] Zhaoyang Lv, Chris Beall, Pablo F. Alcantarilla, Fuxin Li, Zsolt Kira, and Frank Dellaert. A continuous optimization

approach for efficient and accurate scene flow. In *Proc. ECCV*, 2016. 7

[35] Zhaoyang Lv, Kihwan Kim, Alejandro Troccoli, Deqing Sun, James Rehg, and Jan Kautz. Learning rigidity in dynamic scenes with a moving camera for 3d motion field estimation. In *Proc. ECCV*, 2018. 2

[36] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *Proc. CVPR*, 2019. 2, 3

[37] 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 *Proc. CVPR*, 2016. 1, 2, 5, 6, 7

[38] 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 *Proc. CVPR*, 2016. 3, 4

[39] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proc. CVPR*, pages 3061–3070, 2015. 1, 2, 3

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

[41] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proc. CVPR*, 2017. 3, 6, 7

[42] Zhile Ren, Deqing Sun, Jan Kautz, and Erik Sudderth. Cascaded scene flow prediction using semantic segmentation. In *3DV*, 2017. 2

[43] René Schuster, Christian Bailer, Oliver Wasenmüller, and Didier Stricker. Combining stereo disparity and optical flow for basic scene flow. In *CVTS*, 2018. 7

[44] René Schuster, Oliver Wasenmuller, Georg Kuschk, Christian Bailer, and Didier Stricker. Sceneflowfields: Dense interpolation of sparse scene flow correspondences. In *WACV*, 2018. 7

[45] Akihito Seki and Marc Pollefeys. Patch based confidence prediction for dense disparity map. In *BMVC*, 2016. 7

[46] Laura Sevilla-Lara, Deqing Sun, Varun Jampani, and Michael J. Black. Optical flow with semantic segmentation and localized layers. In *Proc. CVPR*, 2016. 2

[47] Deqing Sun, Erik B Sudderth, and Michael J Black. Layered image motion with explicit occlusions, temporal consistency, and depth ordering. In *Advances in Neural Information Processing Systems*, pages 2226–2234, 2010. 2

[48] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, June 2018. 3, 6, 7, 13

[49] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of cnns for optical flow estimation. *IEEE TPAMI*, 2019. 7

[50] Brian Taylor, Vasiliy Karasev, and Stefano Soatto. Causal video object segmentation from persistence of occlusions. In *Proc. CVPR*, pages 4268–4276, 2015. 2

[51] P. Tokmakov, K. Alahari, and C. Schmid. Learning motion patterns in videos. In *Proc. CVPR*, 2017. 2

[52] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Proc. ICCV*, 1999. 1, 2

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

[54] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a piecewise rigid scene model. *IJCV*, 115(1), 2015. 1

[55] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, Sept. 1994. 2

[56] Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke, and Daniel Cremers. Efficient dense scene flow from sparse or dense stereo data. In *Proc. ECCV*, pages 739–751. Springer, 2008. 2

[57] Jonas Wulff, Laura Sevilla-Lara, and Michael J Black. Optical flow in mostly rigid scenes. In *Proc. CVPR*, 2017. 6, 7

[58] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proc. ECCV*, 2018. 4, 6, 7, 11

[59] Jia Xu, René Ranftl, and Vladlen Koltun. Accurate optical flow via direct cost volume processing. In *Proc. CVPR*, 2017. 6, 7

[60] Guorun Yang, Hengshuang Zhao, Jianping Shi, Zhidong Deng, and Jiaya Jia. SegStereo: Exploiting semantic information for disparity estimation. In *Proc. ECCV*, 2018. 3, 7

[61] Zhichao Yin, Trevor Darrell, and Fisher Yu. Hierarchical discrete distribution decomposition for match density estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3

[62] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *JMLR*, 2016. 7

[63] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Trans. Computational Imaging*, 3(1):47–57, 2017. 6

[64] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proc. CVPR*, 2017. 4

[65] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *Proc. ECCV*, 2018. 3

Appendix A. Training Details

We perform pre-training on the synthetic SceneFlow dataset and fine-tuning on Sintel and KITTI, respectively.

Synthetic SceneFlow Dataset. We use the subset of FlyingThings3D used in [25], Monkaa, and Driving for pre-training. We remove images whose maximum optical flow magnitude is greater than 500. We end up using 128,753 samples for training.

Supervised training is performed for the pre-training with ground-truth annotations of optical flow, disparity, and their associated occlusions. The loss function is defined as

$$\mathcal{L}_{sp} = (\mathcal{L}_F + \mathcal{L}_{O_F}) + 0.25 \times (\mathcal{L}_D + \mathcal{L}_{O_D}) \,. \tag{8}$$

For Monkaa and Driving, since only optical flow and disparity annotations are available, we only set $\mathcal{L}_{O_D}$ and $\mathcal{L}_{O_F}$ to 0 for training data sampled from Monkaa and Driving.

During training, we use color jittering, including randomly changing gamma value, changing brightness, changing contrast, and adding Gaussian noise, for both optical flow and disparity training. Additionally, we use random crops and vertical flips for stereo training images. The crop size is 256×512. For optical flow training images, we perform extensive data augmentations including random crop, translation, rotation, zooming, squeezing, and horizontal and vertical flip, where the crop size is 384×640. The network is trained for 100 epochs with a batch size of 8 using the Adam optimizer [31]. We use synchronized Batch Normalization [58] to ensure there are enough training samples for estimating Batch Normalization layers' statistics when using multiple GPUs. The initial learning rate is 0.001 and decreased by factor of 10 after 70 epochs.

Sintel. We fine-tune the pre-trained model on Sintel. Sintel training data provides optical flow, disparity, and their corresponding occlusion annotations. We therefore use the same loss function as used for the pre-training.

During training, we apply the same color jittering used for pre-training. Similarly we use random crops and vertical flips for stereo training images with crop size of 384×768. For optical flow training images, we perform extensive data augmentations as well including random crop, translation, rotation, zooming, squeezing, and horizontal and vertical flip, where the crop size is 384×768.

Synchronized Batch Normalization is used with batch size of 8. The model is first trained for 500 epochs using the Adam optimizer with an initial learning rate of 0.0005, which is decreased by factor of 2 after every 100 epochs. The weight decay is 0.0004. After 500-epoch training is finished, we keep fine-tuning the model for another 500 epochs using Adam with an initial learning rate of 0.0002, which is decreased by factor of 2 after every 100 epochs. The weight decay remains 0.0004.

KITTI. On KITTI (including KITTI2012 and KITTI2015), we use both supervised loss and semi-supervised loss. The final loss is defined as

$$\mathcal{L} = \underbrace{\mathcal{L}_F + \mathcal{L}_D}_{\text{supervised loss}} + \underbrace{\alpha_O\big(\mathcal{L}_{O_{Fd}} + \mathcal{L}_{O_{Dd}}\big) + \alpha_{S_d}\mathcal{L}_{S_d}}_{\text{distillation loss}} + \underbrace{\alpha_{PC}\mathcal{L}_{PC} + \alpha_{SC}\mathcal{L}_{SC} + \mathcal{L}_{SS} + \mathcal{L}_{REG}}_{\text{self-supervised loss}}, \tag{9}$$

where $\mathcal{L}_{O_{Fd}}$ and $\mathcal{L}_{O_{Dd}}$ are distillation loss for optical flow occlusion and disparity occlusion, respectively. They are defined as `smooth_L1` loss between the pseudo ground-truth (*i.e.*, estimations from a model pre-trained on synthetic SceneFlow dataset) and estimations from the model being trained. On the validation set, we empirically found $\alpha_O = 0.05, \alpha_{S_d} = 1, \alpha_{PC} = 0.5, \alpha_{SC} = 0.5$ work well. For the SSIM loss, we use $\gamma_D = 0.005 \times C_H \times C_W$ and $\gamma_F = 0.01 \times C_H \times C_W$[2], where C_H and C_W are crop height and width, respectively. For the regularization term, we empirically set $\beta_F = \beta_D = 0.5$.

During training, we use similar color jittering used in pre-training but with a probability of 0.5. Similarly we use random crops and vertical flips for stereo training images with crop size of 320×768. For optical flow training images, we perform extensive data augmentations as well including random crop, translation, rotation, zooming, squeezing, and horizontal and vertical flip, where the crop size is 320×768.

Synchronized Batch Normalization is used with batch size of 8. The model is fine-tuned for 1,500 epochs using the Adam optimizer with an initial learning rate of 0.001, which is decreased by factor of 2 at epochs of 400, 800, 1,000, 1200, and 1,400. The weight decay is 0.0004. Another round of fine-tuning is followed with an initial learning rate of 0.0002, which is decreased by factor of 2 at epochs of 400, 600, 800, and 900.

Appendix B. Rigidity-based Warped Disparity Refinement for Scene Flow Estimation

Determine rigidity area. Given the estimated semantic segmentation labels of the first left frame $\mathbf{S}^{1,l}$, we select pixels as static rigid regions by removing pixels which have a semantic label of vehicle, pedestrian, cyclist, or sky. This step gives a

[2] In our definition of SSIM loss, the function $SS(\cdot, \cdot)$ gives a single scalar value.

conservative selection of static regions with points not at infinity. The output is a binary mask $\mathbf{B}$ with the label 1 indicating static rigid region. Since the semantic segmentation can be inaccurate at object boundary, we further perform an erosion operation with a size of 10 on the static rigid region mask $\mathbf{B}$.

Estimate rigid flow induced by camera motion. Given the estimated flow $\mathbf{F}^1$ and disparity $\mathbf{D}^1$ of the left frame, we calculate the ego-motion flow induced by the rigid camera motion by minimizing the weighted errors between predicted rigid flow $\mathbf{F}^1_{\mathbf{R}}$ and optical flow $\mathbf{F}^1$ in the background region pixels $\mathbf{x} \in \mathbf{R}^2$:

$$\arg\min_{\xi} \quad \mathbf{r}^T(\xi;\mathbf{x})\mathbf{W}\mathbf{r}(\xi;\mathbf{x}) \tag{10}$$

$$\mathbf{r}(\xi;\mathbf{x}) = \mathbf{F}^1(\mathbf{x}) - \mathbf{F}^1_{\mathbf{R}}(\xi;\mathbf{x}) \tag{11}$$

$$\mathbf{F}^1_{\mathbf{R}}(\xi;\mathbf{x}) = \mathcal{W}(\xi;\mathbf{x},\mathbf{D}^1(\mathbf{x})) - \mathbf{x} \tag{12}$$

where $\mathbf{x} \in \mathbf{R}^2$ denotes the pixels in 2D image space which are within the rigid areas $\mathbf{B}$. $\mathcal{W}(\xi;\mathbf{x},\mathbf{D}^1)$ is the warping function which transforms the pixels $\mathbf{x}$ and its corresponding disparity $\mathbf{D}^1(\mathbf{x})$ with an estimated transform $\xi \in \mathbf{SE}(3)$. $\mathbf{W}$ is a diagonal weight matrix that depends on residuals using Huber weight function.

We solve equation 10 as an iteratively reweighted least-square problem using Gauss-Newton update:

$$\delta\xi = (\mathbf{J}^T\mathbf{W}\mathbf{J})^{-1}\mathbf{J}^T\mathbf{W}\mathbf{r} \tag{13}$$

$$\xi = \xi \circ \delta\xi \tag{14}$$

where $\circ$ indicates the right composition of $\xi \in \mathbf{SE}(3)$. $\mathbf{J}$ is the Jacobian matrix of $\partial\mathbf{F}^1_{\mathbf{R}}(\xi)/\partial\xi$.

Suppose $\mathbf{K}$ is the intrinsic matrix for a pin-hole camera without distortion, which can be parameterized as (f_x, f_y, c_x, c_y) with f_x, f_y as its focal length and c_x, c_y as its offset along the two axes. The baseline of the stereo pair is b. We define the 3D point $\mathbf{p} = (p_x, p_y, p_z)$ as $\mathbf{p} = (f_x b/\mathbf{D}^1(\mathbf{x}))\mathbf{K}^{-1}\mathbf{x}$. Through chain-rule, we can derive the analytical form of the Jacobian matrix $\mathbf{J}$. To simplify the computation, we use the inverse depth parameterization $\mathbf{p} = (p_u/p_d, p_v/p_d, 1/p_d)$ in which $\mathbf{x} = (p_u, p_v) \in \mathbf{R}^2$ is the pixel of coordinate of $\mathbf{x}$ and p_d is the inverse depth as $p_d = \mathbf{D}^1(\mathbf{x})/(f_x b)$. Thus, we obtain the Jacobian matrix at a pixel $\mathbf{x}$ as:

$$\begin{bmatrix} -p_u p_v f_x & (1+p_u^2)f_x & -p_v f_x & p_d f_x & 0 & -p_u p_d f_x \\ -(1+p_v^2)f_y & p_u p_v f_y & p_u f_y & 0 & p_d f_y & -p_u p_d f_y \end{bmatrix} \tag{15}$$

We perform the Gauss-Newton update if the absolute residual error is bigger than 10^{-6} with a maximum of 20 iterations. All operations are implemented in Pytorch and executed in GPU. The running time of the total optimization varies between 0.03s and 0.2s, according to the number of iterations. In average, the optimization step takes 0.1s for KITTI image of resolution 375x1242.

The final optical flow $\mathbf{F}$ is an element-wise linear composition of $\mathbf{F}^1$ and $\mathbf{F}^1_{\mathbf{R}}$ as:

$$\mathbf{F} = (\mathbf{1} - \mathbf{B}) \otimes \mathbf{F}^1 + \mathbf{B} \otimes \mathbf{F}^1_{\mathbf{R}} \tag{16}$$

where $\otimes$ indicates element-wise multiplications.

Estimate warped second frame rigid disparity. Given the estimated optimal $\xi^\star$, we define the disparity $\mathbf{D}^{1\to2}_{\mathcal{W},\mathbf{R}}$ of the second frame warped from the first frame following the optimal rigid transform $\xi^\star$:

$$\mathbf{D}^{1\to2}_{\mathcal{W},\mathbf{R}} = \mathcal{W}^{1\to2}_{\mathbf{D}}(\xi^\star;\mathbf{D}^1) \tag{17}$$

where $\mathcal{W}^{1\to2}_{\mathbf{D}}(\cdot)$ defines the disparity channel output from the warping function $\mathcal{W}(\cdot)$ through a forward warping. Given the forward optical flow $\mathbf{F}^1$, the warped disparity of the second frame can be computed through an inverse warping $\mathcal{W}^{2\to1}_{\mathbf{D}}$ as:

$$\mathbf{D}^{2\to1}_{\mathcal{W}} = \mathcal{W}^{2\to1}_{\mathbf{D}}(\mathbf{F}^1, \mathbf{D}^2) \tag{18}$$

We find that the disparity through forward warping $\mathbf{D}^{1\to2}_{\mathcal{W},\mathbf{R}}$ gives more accurate disparity in static region and can better handle occlusions. The final warped disparity $\mathbf{D}^2_{\mathcal{W}}$ is a element-wise linear composition of $\mathbf{D}^{2\to1}_{\mathcal{W}}$ and $\mathbf{D}^{1\to2}_{\mathcal{W},\mathbf{R}}$ as:

$$\mathbf{D}^2_{\mathcal{W}} = (\mathbf{1} - \mathbf{B}) \otimes \mathbf{D}^{2\to1}_{\mathcal{W}} + \mathbf{B} \otimes \mathbf{D}^{1\to2}_{\mathcal{W},\mathbf{R}} \tag{19}$$

Note that both warping function cannot deal with out-of-boundary pixels due to two-view occlusion. This can be resolved by the additional refinement network detailed in the following section.

Table 7. Definition of our shared encoder. H and W denote the height and width of the input images. $[\cdot]$ indicates a residual block [16]. We use convolution with stride of 2 to perform downsampling. The first downsampling is performed in the layer conv1_1.

layer name	output size	layer setting
input	$H \times W \times 3$	-
conv1_1 conv1_2 conv1_3	$\frac{H}{2} \times \frac{W}{2} \times 32$	$3 \times 3, 32$ $3 \times 3, 32$ $3 \times 3, 32$
conv2	$\frac{H}{4} \times \frac{W}{4} \times 32$	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$
conv3	$\frac{H}{8} \times \frac{W}{8} \times 64$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 16$
conv4	$\frac{H}{16} \times \frac{W}{16} \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$
conv5	$\frac{H}{32} \times \frac{W}{32} \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$

Appendix C. Details of Network Architecture

- Table 7 provides the detailed network architecture of our shared encoder, which is a ResNet-like [16] architecture.

- Table 8 provides details of the pyramid pooling module (PPM), which aggregates multi-scale feature maps to enhance disparity estimation and semantic segmentation.

- The Hourglass module used for disparity estimation refinement is illustrated in Table 9. The input is a concatenation of upsampled disparity (by a factor of 2), the feature map of the first image (128-dimensional), and the warped feature map of the second image (128-dimensional). The output is a residual disparity estimation that is added to the twice upsampled disparity.

- The refinement network for warped disparity, which is used for scene flow estimation, can be found in Table 10. The input is a concatenation of $\mathbf{I}^{1,l}$, $\mathbf{O}_F^{1,l}$, $\mathbf{D}^{1,l}$, and $g(\mathbf{D}^{2,l}, \mathbf{F}^{1,l})$, with 6 (=3+1+1+1) channels in total. The network consists of an encoder, a decoder, and skip connections between them. It contains 9.1M parameters in total and takes 0.01s for inference for a KITTI image with resolution of 375×1242. We perform supervision for intermediate layers of the decoder, in a manner similar to optical flow and disparity estimations.

Appendix D. More Ablation Studies

New network design. As shown in Table 11, we study the effectiveness of new network designs for disparity estimation. The baseline model has exactly the same architecture as PWC-Net [48] for optical flow estimation, except we construct a 1D cost volume for disparity estimation. It is a compact model with 7.1M parameters. However, most of the parameters concentrate in the decoder due to DenseNet blocks. By removing the last pyramid in both encoder and decoder and adding Batch Normalization layers, we obtain significant improvement in disparity while halving the parameters. By replacing the original encoder consisting of plain CNN layers with deeper residual blocks, we obtain further improvements and yet still have fewer parameters. Adding PPM and hourglass refinement keeps improving the accuracy. Our final model has slightly more parameters than the baseline, but the performance on both synthetic and real-world benchmark datasets increases substantially.

Shared encoder. We report optical flow and disparity errors on Sintel, KITTI 2012, and KITTI 2015 using both separate and shared encoders in Table 12, where a model trained on synthetic SceneFlow dataset is used. As we can see, a shared encoder leads to better EPE metrics on both KITTI 2012 and KITTI 2015.

Table 8. Definition of our PPM head, where branch1, branch2, branch3, and branch4 are all parallel branches on top of the conv5 layer in the encoder.

layer name	output size	layer setting
branch1	$\frac{H}{32} \times \frac{W}{32} \times 128$	1×1 adaptive avg. pool $1 \times 1, 128$ bilinear interpolation
branch2	$\frac{H}{32} \times \frac{W}{32} \times 128$	2×2 adaptive avg. pool $1 \times 1, 128$ bilinear interpolation
branch3	$\frac{H}{32} \times \frac{W}{32} \times 128$	3×3 adaptive avg. pool $1 \times 1, 128$ bilinear interpolation
branch4	$\frac{H}{32} \times \frac{W}{32} \times 128$	6×6 adaptive avg. pool $1 \times 1, 128$ bilinear interpolation
fusion	$\frac{H}{32} \times \frac{W}{32} \times 128$	concat of conv5, branch1 branch2, branch3, and branch4 $3 \times 3, 128$

Table 9. Definition of our disparity Hourglass refinement model.

layer name	output size	layer setting
input	$\frac{H}{2} \times \frac{W}{2} \times 257$	-
conv1	$\frac{H}{4} \times \frac{W}{4} \times 514$	$3 \times 3, 514$
conv2	$\frac{H}{8} \times \frac{W}{8} \times 514$	$3 \times 3, 514$
conv3	$\frac{H}{8} \times \frac{W}{8} \times 514$	$3 \times 3, 514$
conv4	$\frac{H}{4} \times \frac{W}{4} \times 514$	bilinear interpolation $3 \times 3, 514$
conv5	$\frac{H}{2} \times \frac{W}{2} \times 257$	bilinear interpolation $3 \times 3, 257$
output	$\frac{H}{2} \times \frac{W}{2} \times 1$	$3 \times 3, 1$

Appendix E. Visual Results of Optical Flow and Disparity Estimations

We provide more visual results of optical flow and disparity estimations of the test set in Fig. 5 and Fig. 6 for KITTI 2012 and in Fig. 7 and Fig. 8 for KITTI 2015.

We can clearly see our full model (supervised loss plus semi-supervised loss) produces visually better results on both KITTI 2012 and KITTI 2015.

Acknowledgement

Huaizu Jiang and Erik Learned-Miller acknowledge support from AFRL and DARPA (#FA8750- 18-2-0126) and the MassTech Collaborative grant for funding the UMass GPU cluster. The U.S. Gov. is authorized to reproduce and distribute reprints for Gov. purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the AFRL and DARPA or the U.S. Gov.

Table 10. Definition of our warped disparity refinement model. output5 is on top of decoder_layer_5. output4, output3, and output2 are computed similarly. output1 is on top of decoder_layer1_2. We compute loss for all five output during training and sum them up as the final loss. During inference, we only compute output1.

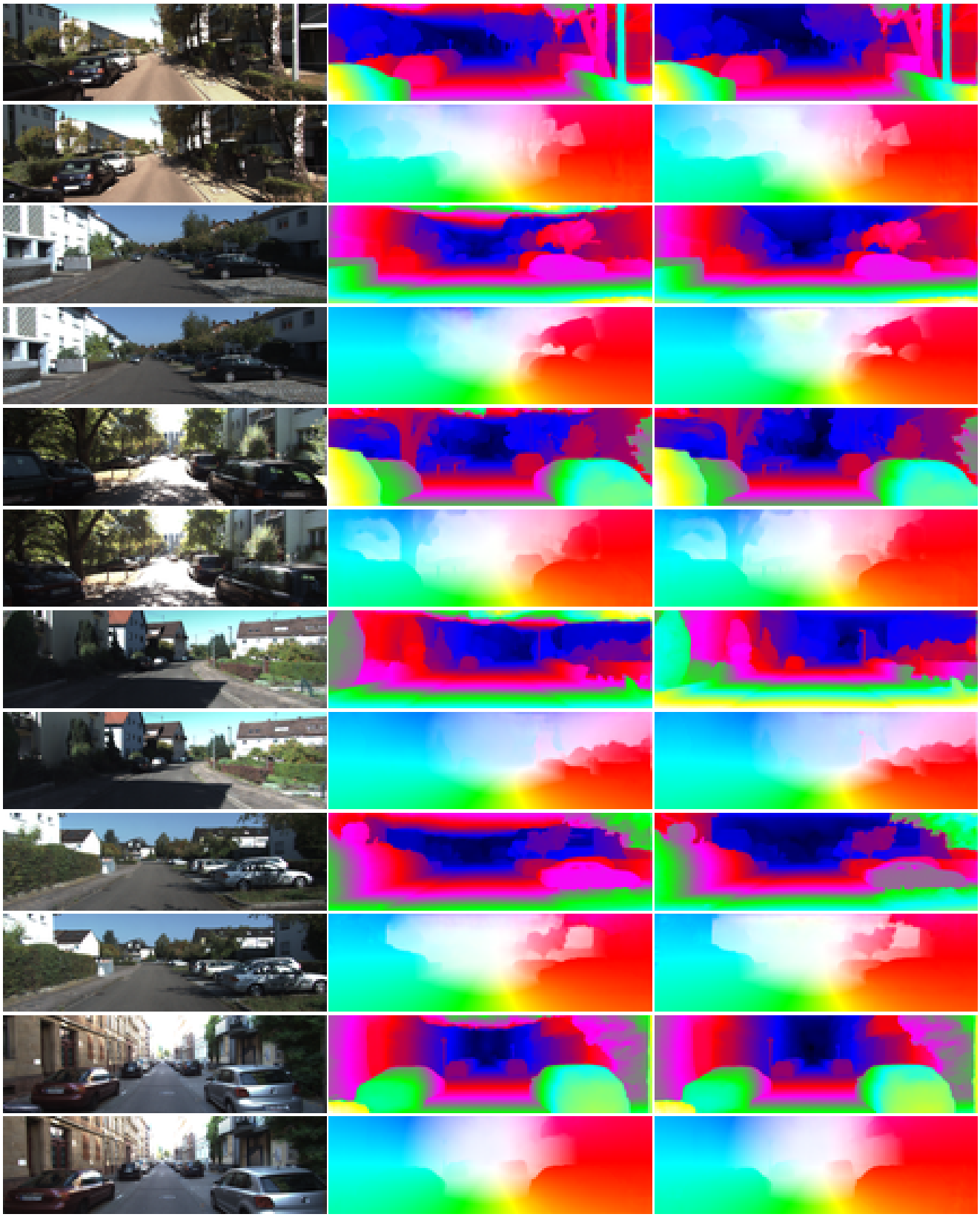
layer name	output size	layer setting
input	$H \times W \times 6$	-
encoder_layer1	$H \times W \times 32$	$3 \times 3, 32$
encoder_layer2	$\frac{H}{2} \times \frac{W}{2} \times 32$	2×2 avg. pool, stride of 2 $3 \times 3, 32$
encoder_layer3	$\frac{H}{4} \times \frac{W}{4} \times 32$	2×2 avg. pool, stride of 2 $3 \times 3, 32$
encoder_layer4	$\frac{H}{8} \times \frac{W}{8} \times 32$	2×2 avg. pool, stride of 2 $3 \times 3, 32$
encoder_layer5	$\frac{H}{16} \times \frac{W}{16} \times 32$	2×2 avg. pool, stride of 2 $3 \times 3, 32$
bottleneck	$\frac{H}{32} \times \frac{W}{32} \times 32$	2×2 avg. pool, stride of 2 $3 \times 3, 32$
decoder_layer5	$\frac{H}{16} \times \frac{W}{16} \times 512$	$3 \times 3, 512$ $2\times$ bilinear interpolation
decoder_layer4	$\frac{H}{8} \times \frac{W}{8} \times 256$	concat. with encoder_layer_5 $3 \times 3, 256$ $2\times$ bilinear interpolation
decoder_layer3	$\frac{H}{4} \times \frac{W}{4} \times 128$	concat. with encoder_layer_4 $3 \times 3, 128$ $2\times$ bilinear interpolation
decoder_layer2	$\frac{H}{2} \times \frac{W}{2} \times 64$	concat. with encoder_layer_3 $3 \times 3, 64$ $2\times$ bilinear interpolation
decoder_layer1_1	$H \times W \times 32$	concat. with encoder_layer_2 $3 \times 3, 32$ $2\times$ bilinear interpolation
decoder_layer1_2	$H \times W \times 32$	concat. with encoder_layer_1 $3 \times 3, 32$
output5	$\frac{H}{16} \times \frac{W}{16} \times 1$	$3 \times 3, 1$
output4	$\frac{H}{8} \times \frac{W}{8} \times 1$	$3 \times 3, 1$
output3	$\frac{H}{4} \times \frac{W}{4} \times 1$	$3 \times 3, 1$
output2	$\frac{H}{2} \times \frac{W}{2} \times 1$	$3 \times 3, 1$
output1	$H \times W \times 1$	$3 \times 3, 1$

Table 11. Ablation study of design choices for disparity.

5 layers+BN	ResNet encoder	PPM	hourglass	#params	FlyThings3D (EPE)	KITTI2015 (EPE)
				7.1M	2.10	1.01
✓				3.6M	1.61	0.85
✓	✓			6.9M	1.40	0.77
✓	✓	✓		7.7M	1.32	0.77
✓	✓	✓	✓	8.3M	1.15	0.71

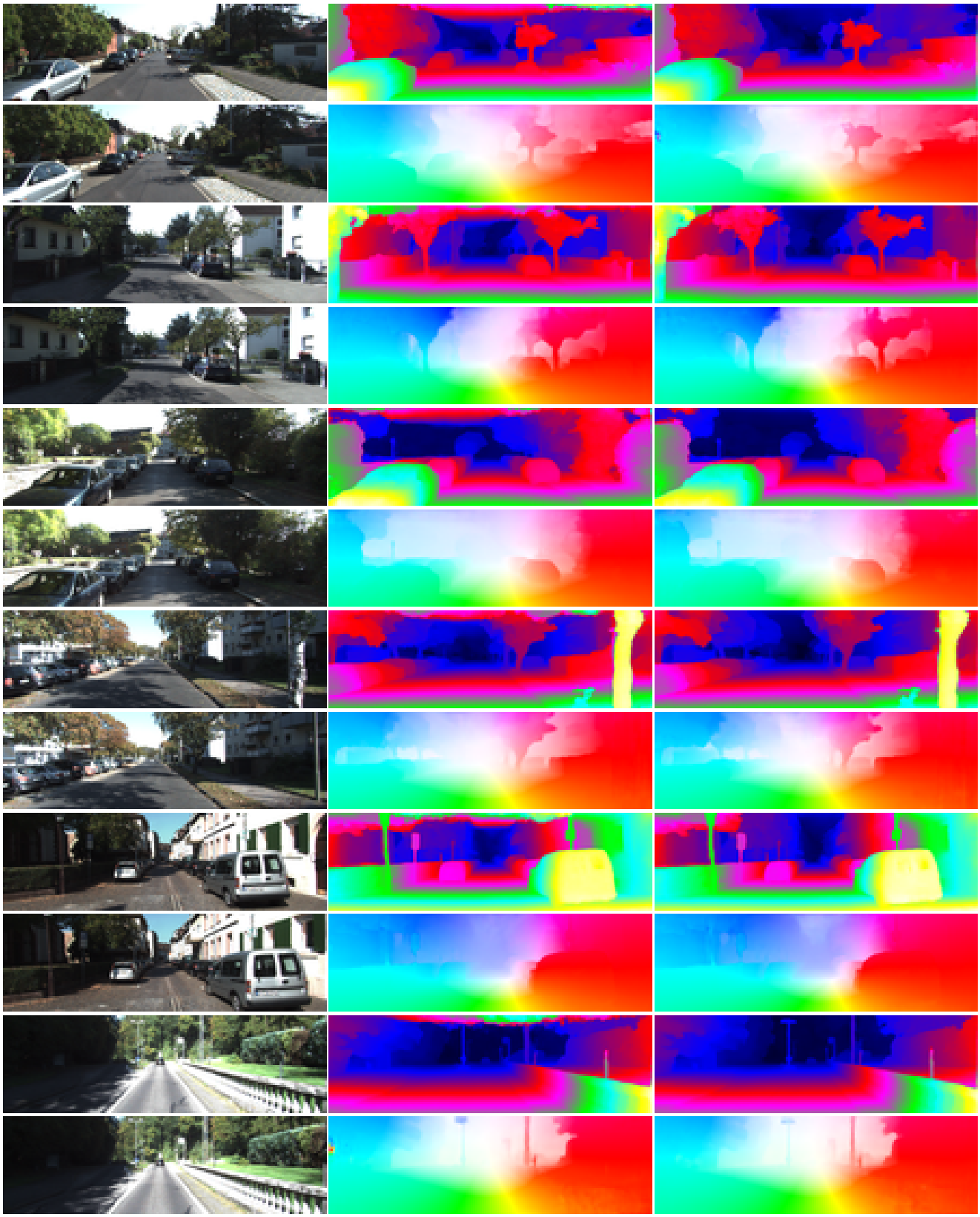
Table 12. Effectiveness of the shared encoder for optical flow and disparity estimations.

		Sintel (EPE)		KITTI 2012		KITTI 2015	
		clean	final	EPE	D1/F1-occ	EPE	D1/F1-occ
optical flow	separate encoder	1.97	3.34	2.63	11.72%	6.37	21.15%
	shared encoder	1.91	3.78	2.55	12.56%	6.23	23.29%
disparity	separate encoder	1.56	2.99	1.09	6.17%	1.26	6.62%
	shared encoder	1.70	3.20	1.04	5.42%	1.22	6.38%



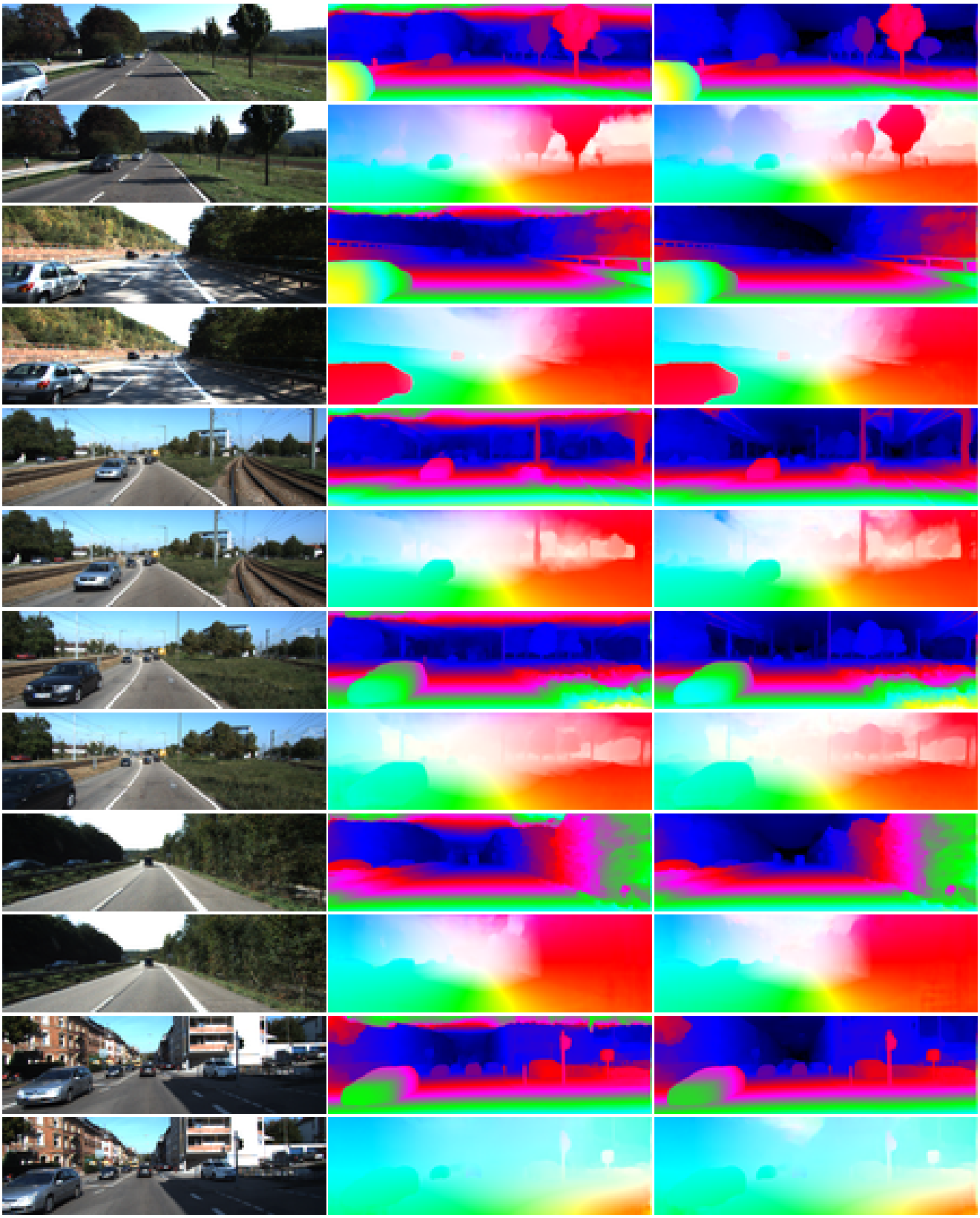
(a) input images (b) supervised loss (c) full loss

Figure 5. Visual results on the test set of KITTI 2012. We show two consecutive video frames in the first column. In the second and third columns, we show disparity in every first row and optical flow in the other one. Best viewed in color.



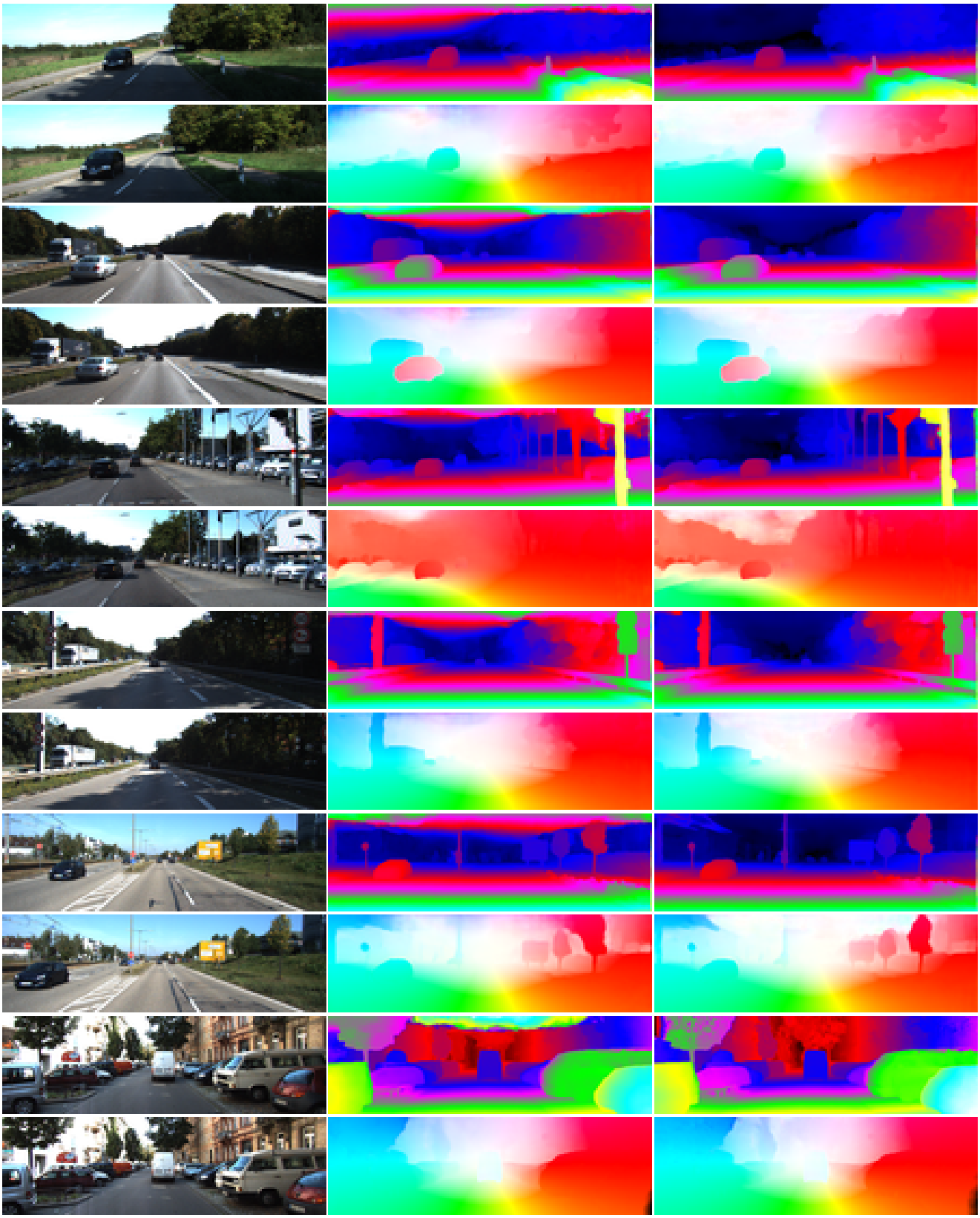
(a) input images (b) supervised loss (c) full loss

Figure 6. Visual results on the test set of KITTI 2012. We show two consecutive video frames in the first column. In the second and third columns, we show disparity in every first row and optical flow in the other one. Best viewed in color.



(a) input images (b) supervised loss (c) full loss

Figure 7. Visual results on the test set of KITTI 2015. We show two consecutive video frames in the first column. In the second and third columns, we show disparity in every first row and optical flow in the other one. Best viewed in color.



(a) input images (b) supervised loss (c) full loss

Figure 8. Visual results on the test set of KITTI 2015. We show two consecutive video frames in the first column. In the second and third columns, we show disparity in every first row and optical flow in the other one. Best viewed in color.