Deep Rigid Instance Scene Flow

Wei-Chiu Ma[1,2] Shenlong Wang[1,3] Rui Hu[1] Yuwen Xiong[1,3] Raquel Urtasun[1,3]
[1]Uber Advanced Technologies Group
[2]Massachusetts Institute of Technology
[3]University of Toronto

Abstract

In this paper we tackle the problem of scene flow estimation in the context of self-driving. We leverage deep learning techniques as well as strong priors as in our application domain the motion of the scene can be composed by the motion of the robot and the 3D motion of the actors in the scene. We formulate the problem as energy minimization in a deep structured model, which can be solved efficiently in the GPU by unrolling a Gaussian-Newton solver. Our experiments in the challenging KITTI scene flow dataset show that we outperform the state-of-the-art by a very large margin, while being 800 times faster.

1. Introduction

Scene flow refers to the problem of estimating a threedimenional motion field from a set of two consecutive (in time) stereo pairs. It was first introduced in [40] to describe the 3D motion of each point in the scene. Through scene flow, we can gain insights into the geometry as well as the overall composition and motion of the scene. It is of particular importance for robotics systems, such as self-driving cars, as knowing the 3D motion of other objects in the scene can not only help the autonomous systems avoid collision while planing its own future movements, but also improve the understanding of the scene and predict the intent of others. In this work, we focus on estimating the 3D scene flow in autonomous driving scenarios.

In the world of self-driving, the motion of the scene can be mostly explained by the motion of the ego-car. The presence of dynamic objects which typically move rigidly can also be utilized as strong priors. Previous structure prediction approaches often exploit these facts and fit a piece-wise rigid representations of motion [41, 44, 27, 3]. While these methods achieve impressive results on scene flow estimation, they require minutes to process each frame, and thus cannot be employed in real-world robotics systems.

On the other hand, deep learning based methods have achieved state-of-the-art performance in real time on a variety of low level tasks, such as optical flow prediction [11, 32, 38] and stereo estimation [46, 26, 24]. While they produce 'accurate' results, their output is not structured and cannot capture the relationships between estimated variables. For instance, they lack the ability to guarantee that pixels on a given object produce consistent estimates. While this phenomenon may have little impact in photography editing applications, this can cathastrophic in the context of self-driving cars, where the motion of the full object is more important than the motion of each individual pixel.

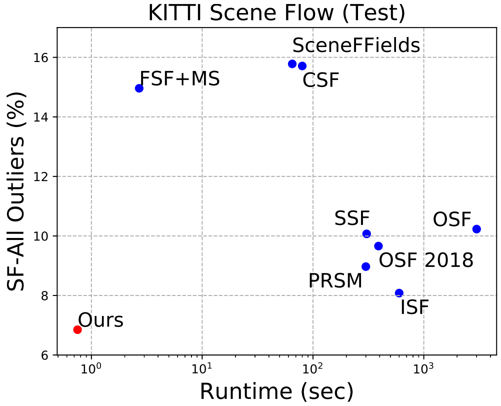

Figure 1: **Performance vs runtime on KITTI SceneFlow dataset:** Our approach is much faster and more accurate.

With these problems in mind, we develop a novel deep rigid instance scene flow (DRISF) model that takes the best of both worlds. The idea behind is that the motion of the scene can be composed by estimating the 3D rigid motion of each actor. The static background can also be modeled as a rigidly moving object, as its 3D motion can be described by the 'ego-car' motion. The problem is thus reduced to estimating the 3D motion of each traffic participant. Towards this gaol, we first capitalize on deep neural networks to estimate optical flow, disparity and instance segmentation. We then exploit multiple geometry based energy functions to encode the structural geometric relationship between these visual cues. Through optimizing the energy function, we can effectively reason about the 3D motion of each traffic participant. As the energy takes the form of weighted sum of squares, it can be efficiently minimized via Gaussian-Newton (GN) algorithm [5]. We implement the GN solver

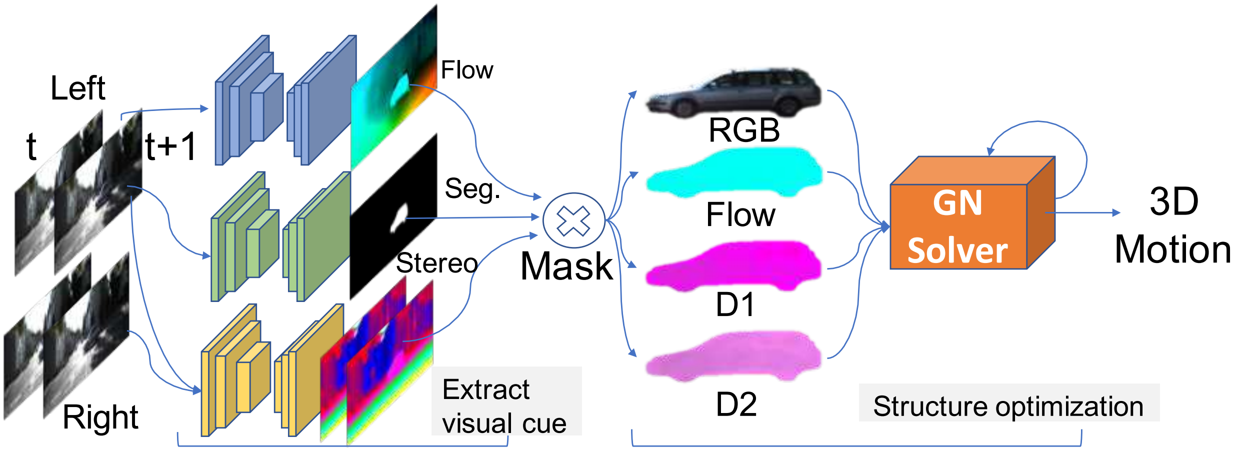

Figure 2: **Overview of our approach:** Given two consecutive stereo images, we first estimate the flow, stereo, and segmentation (Sec. 3.1). The visual cues of each instance are then encoded as energy functions (Sec. 3.2) and passed into the Gaussian-Newton (GN) solver to find the best 3D rigid motion (Sec. 3.3). The GN solver is unrolled as a recurrent network.

as layers in neural networks, thus all operations can be computed efficiently on the GPU in an end-to-end fashion.

We demonstrate the effectiveness of our approach on the KITTI scene flow dataset [27]. As shown in Fig. 1, our deep rigid instance scene flow model outperforms all previous methods by a significant margin in both runtime and accuracy. Importantly, it achieves state-of-the-art performance on almost every entry. Comparing to prior art, DRISF reduces the D1 outliers ratio by **43%**, the D2 outliers ratio by **32%**, and the flow outliers ratio by **24%**. Comparing to the existing best scene flow model [3], our scene flow error is **22%** lower and our runtime is **800** times faster.

2. Related Work

Optical flow: Optical flow is traditionally posed as an energy minimization task. It dates back to Horn and Schunck [17] where they define the energy as a combination of a data term and a smoothness term, and adopt variational inference to solve it. Since then, a variety of improvements have been proposed [6, 4, 30]. Recently, deep learning has replaced the variational approaches. Employing deep features for matching [1, 43] improved performance by a large margin. However, as the matching results are not dense, post-processing steps are required [35]. This not only reduces the speed, but also limits the overall performance.

Pioneered by Flownet [11], various end-to-end deep regression based methods have been proposed [21]. Flownet2 [20] stacks multiple networks to iteratively refine the estimated flow and introduces a differentiable warping operation to compensate for large displacements. As the resulting network is very large, SpyNet [32] propose to use spatial pyramid network to handle large motions. They reduce the model size greatly, yet at the cost of degrading performance. Lite-Flownet [19] and PWC-Net [38, 37] extend this idea and incorporate the traditional pyramid processing and cost volume concepts into the network. Comparing to previous approach, the resulting model is smaller and faster. In this work, we adapt the latest PWC-Net as our flow module.

Stereo: Traditional stereo methods [16, 22] follow three steps: compute patch-wise feature, construct cost volumes, and final post-processing. The representation of the patch plays an important role. Modern approaches leverage CNNs to predict whether two patches are a match [45, 46]. While they showed great performance in challenging benchmarks, they are computationally expensive. To speed up the matching process, Luo *et al.* [24] propose a siamese matching network which exploits a correlation layer [9] to extract marginal distributions over all possible disparities. While the usage of the correlation layer significantly improves efficiency, they still require post-processing techniques [15, 47] to smooth their estimation, which largely limits their speed. In light of this, networks that directly regress sub-pixel disparities from the given stereo image pair have been proposed. DispNet [26] exploits a 1D correlation layer to approximate the stereo cost volumes and rely on later layers for implicit aggregation. Kendall *et al.* [23] incorporate 3D conv for further regularization and propose a differentiable soft argmin to enable sub-pixel disparity from cost volumes. PSM-Net [8] later extend [23] by incorporating stacked hourglass [29] and Pyramid spatial pooling [48, 14]. In this work, we exploit PSM-Net as our stereo module.

Scene flow: Scene flow [40] characterizes the 3D motion of a point. Similar to optical flow estimation, the task is traditionally formulated as a variational inference problem [39, 31, 18, 2]. However, the performance is rather limited in real world scenarios due to errors caused by large motions. To improve the robustness, slanted-plane based methods [44, 27, 41, 25] propose to decompose the scene into small rigidly moving planes and solve the discrete-continuous optimization problem. Behl *et al.* [3] build upon [27], and incorporate recognition cues. With the help of fine-grained instance and geometric feature, they are able to establish correspondences across various challenging scenarios. Similar to our work, Ren *et al.* [34] exploit multiple

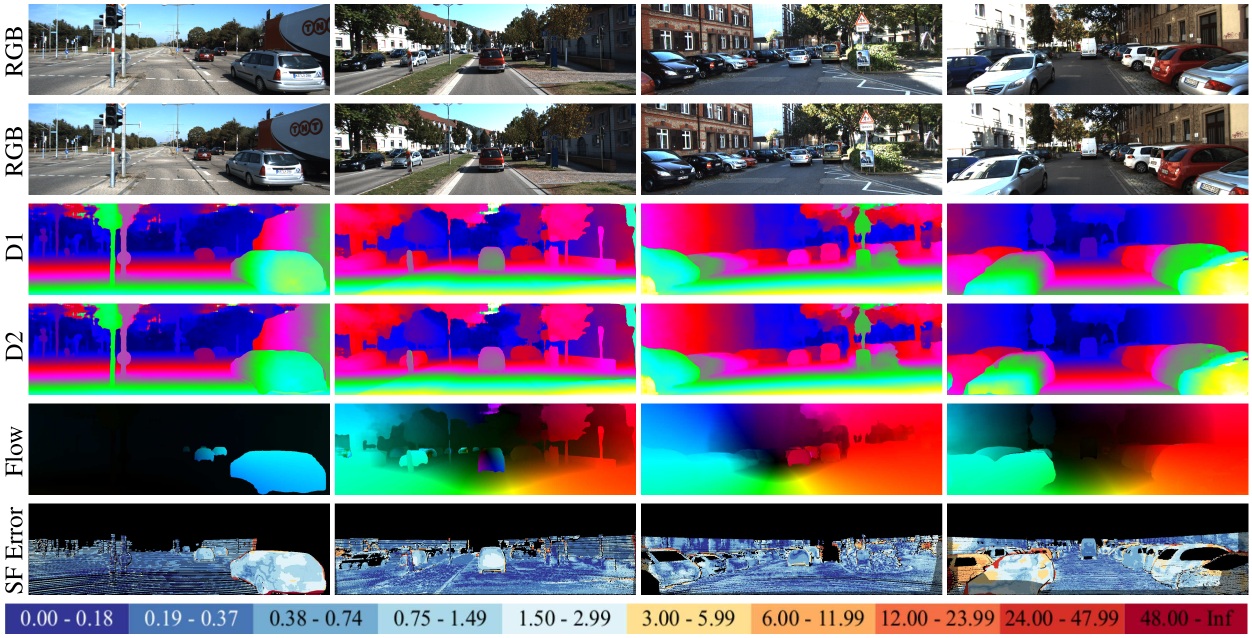

Figure 3: **Qualitative results on val set:** Our model can estimate the background motion very accurately. It is also able to estimate the 3D motion of foreground objects in most scenarios. It fails in challenging cases as show in last column.

visual cues for scene flow estimation. They encode the features via a cascade of conditional random fields and iteratively refine them. While these methods have achieved impressive performance, they are computationally expensive for practical usage. Most methods require minutes to compute one scene flow. This is largely due to the complicated optimization task. In contrast, our deep structured motion estimation model is able to compute scene flow in less than a second, which is two to three orders of magnitude faster.

3. Deep Rigid Instance Scene Flow

In this paper we are interested in estimating scene flow in the context of self-driving cars. We build our model on the intuition that in this scenario the motion of the scene can be formed by estimating the 3D motion of each actor. The static background can be also modeled as a rigidly moving object, as its 3D motion can be described by the 'ego-car' motion. Towards this goal, we proposed a novel deep structured model that exploits optical flow, stereo, as well as instance segmentation as visual cues. We start by describing how we employ deep learning to effectively estimate the geometric and semantic features. We then formulate the scene flow task as an energy minimization problem and discuss each energy term in details. Finally, we describe how to perform efficient inference and learning.

3.1. Visual Cues

We exploit three types of visual cues: instance segmentation, optical flow and stereo.

Instance Segmentation: We utilize Mask R-CNN [13] as our instance segmentation network, as it produces state-of-the-art results in autonomous driving benchmarks such as KITTI [12] and Cityscapes [10]. Mask R-CNN is a proposal based two stage network built upon Faster R-CNN [33]. For each object proposal, it predicts the object class, regresses its 2D box, and infers the bg/fg segmentation mask.

Stereo: We exploit the pyramid stereo matching network (PSM-Net) [8] to compute our stereo estimates. It consists of three main modules: fully convolutional feature module, spatial pyramid pooling [14, 48] and 3D cost volume processing. The feature module computes a high-dimensional feature map in a fully convolutional manner; the spatial pyramid pooling aggregates context in different scales and locations to construct the cost volume; the 3D cost volume module then performs implicit cost volume aggregation and regularizes it using stacked hourglass networks. Compared to previous disparity regression networks, PSM-Net learns to refine and produce sharp disparity images that respect object boundaries better. This is of crucial importance as over-smoothed results often deteriorates motion estimation.

Optical Flow: Our flow module is akin to PWC-Net [38], which is a state-of-the-art flow network designed based on three classical principles (similar to stereo networks): pyramidal feature processing, warping, and cost volume reasoning. Pyramidal feature processing encode visual features with large context; the progressive warping reduces the cost of building cost-volume through a coarse-to-fine scheme. Cost volume reasoning further boost performance by sharpening the boundaries. We implement PWC-net with one modification: during the warping operation, we use the feature of the nearest boundary pixel to pad if the sampling point falls outside the image, rather than 0. Empirically we found this to improve performance.

		Disparity 1			Disparity 2			Optical Flow			Scene Flow		
Methods	Runtime	*bg*	*fg*	all	*bg*	*fg*	all	*bg*	*fg*	all	*bg*	*fg*	all
CSF [25]	1.3 mins	4.57	13.04	5.98	7.92	20.76	10.06	10.40	25.78	12.96	12.21	33.21	15.71
OSF [27]	50 mins	4.54	12.03	5.79	5.45	19.41	7.77	5.62	18.92	7.83	7.01	26.34	10.23
SSF [34]	5 mins	3.55	8.75	4.42	4.94	17.48	7.02	5.63	14.71	7.14	7.18	24.58	10.07
OSF-TC* [28]	50 mins	4.11	9.64	5.03	5.18	15.12	6.84	5.76	13.31	7.02	7.08	20.03	9.23
PRSM* [42]	5 mins	3.02	10.52	4.27	5.13	15.11	6.79	5.33	13.40	6.68	6.61	20.79	8.97
ISF [3]	10 mins	4.12	6.17	4.46	4.88	11.34	5.95	5.40	**10.29**	6.22	6.58	**15.63**	8.08
Our DRISF	**0.75 sec**	**2.16**	**4.49**	**2.55**	**2.90**	**9.73**	**4.04**	**3.59**	10.40	**4.73**	**4.39**	15.94	**6.31**

Table 1: **Comparison against top 6 published approaches:** Our method acheives state-of-the-art performance on almost every entry while being two to three orders of magnitude faster. (*: Method uses more than two temporally adjacent images.)

3.2. Energy Formulation

We now describe the energy formulation of our deep structured model. Let $\mathcal{L}^0, \mathcal{R}^0, \mathcal{L}^1, \mathcal{R}^1$ be the input stereo pairs captured from two consecutive time steps. Let $\mathcal{D}^0, \mathcal{D}^1$ be the estimated stereo, and $\mathcal{F}_\mathcal{L}, \mathcal{F}_\mathcal{R}$ be the inferred flow. Denote $\mathcal{S}^0_\mathcal{L}$ as the instance segmentation computed on the left image $\mathcal{L}^0$. Assume all cameras are pre-calibrated with known intrinsics. We parametrize the 3D rigid motion with $\boldsymbol{\xi} \in \mathfrak{se}(3)$, the Lie-algebra associated with $SE(3)$. We use this parametrization as it is a minimal representation for 3D motion. For each instance $i \in \mathcal{S}^0_\mathcal{L}$, we aim to find the rigid 3D motion that minimizes the weighted combination of photometric error, rigid fitting and flow consistency, where the weights are denoted as $\lambda_{\cdot,i}$. For simplicity, let $\mathcal{I} = \{\mathcal{L}^0, \mathcal{R}^0, \mathcal{L}^1, \mathcal{R}^1, \mathcal{D}^0, \mathcal{D}^1, \mathcal{F}_\mathcal{L}, \mathcal{F}_\mathcal{R}\}$ be input images and visual cues. We denote the set of pixels belonging to instance i as $P_i = \{\boldsymbol{p} | S^0_\mathcal{L}(\boldsymbol{p}) = i\}$. Note that background can be considered as an 'instance' since all the pixels in it undergo the same rigid transform. We obtain the 3D motion of each instance by minimizing

$$\min_{\boldsymbol{\xi}} \{\lambda_{\text{photo},i} E_{\text{photo},i}(\boldsymbol{\xi}; \mathcal{I}) + \lambda_{\text{rigid},i} E_{\text{rigid},i}(\boldsymbol{\xi}; \mathcal{I}) + \lambda_{\text{flow},i} E_{\text{flow},i}(\boldsymbol{\xi}; \mathcal{I})\} \quad (1)$$

The three energy terms are complementary. They capture the geometry and appearance agreement between the observations and inferred rigid motion. Next, we describe the energy terms in more details.

Photometric Error: This energy encodes the fact that correspondences should have similar appearance across all images. In particular, for each pixel $\boldsymbol{p} \in P_i$ in the reference image, we compare its photometric value with that of the corresponding pixel in the target image:

$$E_{\text{photo},i}(\boldsymbol{\xi}; \mathcal{I}) = \sum_{\boldsymbol{p} \in P_i} \alpha_{\boldsymbol{p}} \rho(\mathcal{L}^0(\boldsymbol{p}) - \mathcal{L}^1(\boldsymbol{p}')) \quad (2)$$

where $\alpha_{\boldsymbol{p}} \in \{0, 1\}$ is an indicator function representing which pixel is an outlier. We refer the reader to section 3.3 for a discussion on how to estimate α_p. $\boldsymbol{p}$ is a pixel in the reference image and $\boldsymbol{p}'$ stands for the projected image coordinate on another image, given by inverse depth warping followed by a rigid transform $\boldsymbol{\xi}$. Specifically,

$$\boldsymbol{p}' = \pi_\mathbf{K}(\boldsymbol{\xi} \circ \pi_\mathbf{K}^{-1}(\boldsymbol{p}, \mathcal{D}(\boldsymbol{p}))) \quad (3)$$

where $\pi_\mathbf{K}(\cdot) : \mathbb{R}^3 \to \mathbb{R}^2$ is the perspective projection function given known intrinsic $\mathbf{K}$ and $\pi_\mathbf{K}^{-1}(\cdot, \cdot) : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$ is the inverse projection that convert a pixel and its associated disparity into a 3D point; $\boldsymbol{\xi} \circ \mathbf{x}$ transforms a 3D point $\mathbf{x}$rigidly with transformation $\exp(\boldsymbol{\xi})\mathbf{x}$. ρ is a robust error function that improves the overall robustness by reducing the influence of outliers on the non-linear least squares problems. Following Sun *et al.* [36], we adopt the generalized Charbonnier function $\rho(x) = (x^2 + \epsilon^2)^\alpha$ as our robust function and set $\alpha = 0.45$ and $\epsilon = 10^{-5}$. Similar to [36], we observe the slightly non-convex penalty improves the performance in practice.

Rigid Fitting: This term encourages the estimated 3D rigid motion to be similar to the point-wise 3D motion obtained from the stereo and flow networks. Formally, given correspondences $\{(\boldsymbol{p}, \boldsymbol{q} = \boldsymbol{p} + F_\mathcal{L}(\boldsymbol{p})) | \boldsymbol{p} \in P_i\}$ defined by the output of optical flow network and the disparity maps $\mathcal{D}^0, \mathcal{D}^1$, the energy measures rigid fitting error of $\boldsymbol{\xi}$:

$$E_{\text{rigid},i}(\boldsymbol{\xi}; \mathcal{I}) = \sum_{(\boldsymbol{p},\boldsymbol{q})} \alpha_{\boldsymbol{p}} \rho(\boldsymbol{\xi} \circ \pi_\mathbf{K}^{-1}(\boldsymbol{p}, \mathcal{D}^0(\boldsymbol{p})) - \pi_\mathbf{K}^{-1}(\boldsymbol{q}, \mathcal{D}^1(\boldsymbol{q}))),$$

where $\boldsymbol{q} = \boldsymbol{p} + \mathcal{F}_\mathcal{L}(\boldsymbol{p})$ and $\pi_\mathbf{K}^{-1}$ denotes the inverse projection function, and ρ is the same robust error function.

Flow Consistency: This term encourages the projection of the 3D rigid motion to be close to the original flow estimation. This is achieved by measuring the difference between our optical flow net, and the structured rigid flow, which is computed by warping each pixel using $\mathcal{D}^0$ and the rigid motion $\boldsymbol{\xi}$.

$$E_{\text{flow},i}(\boldsymbol{\xi}; \mathcal{I}) = \sum_{\boldsymbol{p} \in P_i} \rho(\underbrace{(\boldsymbol{p}' - \boldsymbol{p})}_{\text{2D Rigid flow}} - \underbrace{\mathcal{F}_\mathcal{L}(\boldsymbol{p})}_{\text{optical flow}}) \quad (4)$$

where $\boldsymbol{p}'$ is the rigid warping function defined in Eq. (3), and ρ is the same robust error function.

3.3. Inference

Uncertain Pixel Removal: Due to viewpoint change, flow/stereo prediction errors, etc, the visual cues of some

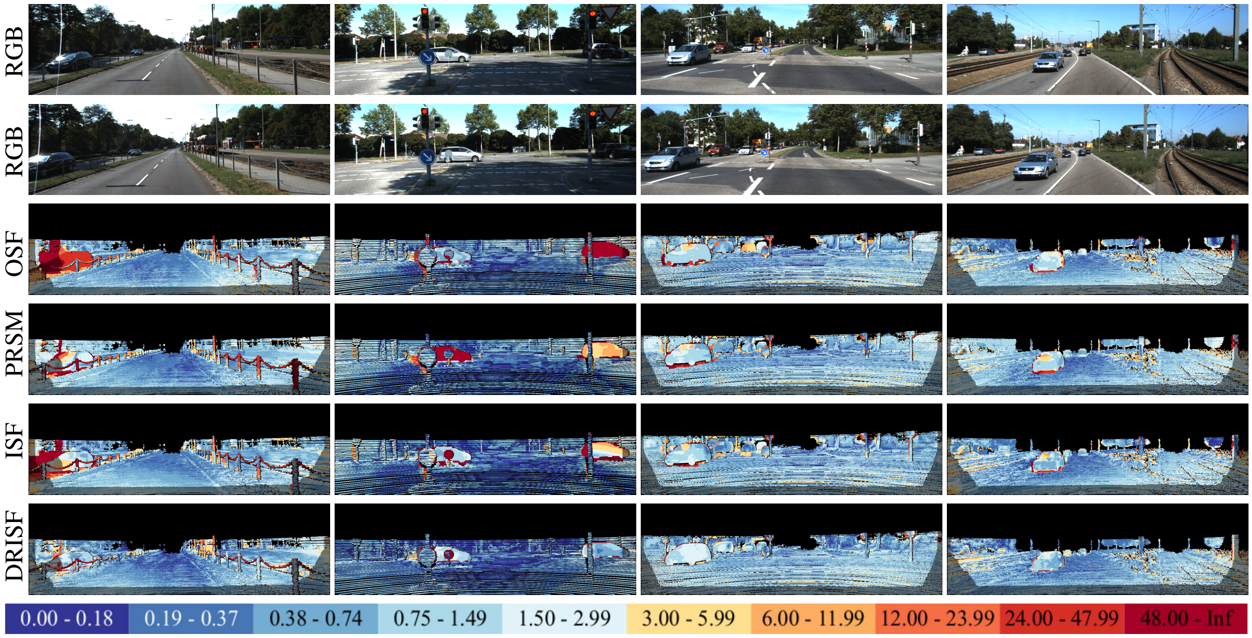

Figure 4: **Qualitative comparison on test sest:** Our method can effecitvely handle occlusion and texture-less regions. It is more robust to the illumination change as well as large displacement. Please refer to the supp. material for more results.

pixels are not reliable. For instance, pixels in one image may be occluded in another image due to viewpoint change. This motivates us to assign $\alpha_{\boldsymbol{p}}$ to each pixel $\boldsymbol{p}$ as an indication of outlier or not. Towards this goal, we first exclude pixels which are likely to be occluded in the next frame. Specifically, pixels are labeled as occluded if the warped 3D disparity of the second frame significantly differs from the disparity of the first frame. The intuition is that the disparity of a pixel cannot change drastically in real world due to the speed limit. We empirically set threshold to 30. Next, we employ the RANSAC scheme to fit a rigid motion for each instance. We only keep the inlier points and prune out the rest. Despite simple, we found this strategy very effective.

Initialization: Due to the highly non-convex structure of the energy model, a good initialization is critical to achieve good performance. As previous step already prune out most unreliable points, we directly exploit the rigid motion obtained by RANSAC as our robust initial guess.

Gaussian Newton Solver: The energy function is non-convex but differentiable w.r.t. $\boldsymbol{\xi}$ defined over continuous space. In order to handle the robust function, we adopt an iterative reweighted least square algorithm [7]. For each iteration, we can rewrite the original energy minimization problem of each instance i as a weighted sum of squares:

$$\boldsymbol{\xi}^{(n+1)} = \arg\min_{\boldsymbol{\xi}} E_{\text{total},i}(\boldsymbol{\xi}) = \arg\min_{\boldsymbol{\xi}} \sum_{\text{Eng}} w_i(\boldsymbol{\xi}^{(n)}) r_i^2(\boldsymbol{\xi}^{(n)}),$$

where r denotes the residual function, w reweights each sample based on the robust function ρ, and Eng refers to summing over the energy terms. We employ Gaussian-Newton algorithm to minimize the function. Thus we have

$$\boldsymbol{\xi}^{(n+1)} = \boldsymbol{\xi}^{(n)} \circ (\mathbf{J}^T \mathbf{W} \mathbf{J})^{(-1)} \mathbf{J}^T \mathbf{W} r(\boldsymbol{\xi}^{(n)}) \qquad (5)$$

where $\circ$ is a pose composition operator and $\mathbf{J} = \frac{\delta r(\boldsymbol{\epsilon} \circ \boldsymbol{\xi}^{(n)})}{\delta \boldsymbol{\epsilon}}|_{\boldsymbol{\epsilon}=0}$. In practice, we unroll the inference steps as a recurrent neural network and define its computation graph as in Eq. (5). The full pipeline including the matrix inverse is differentiable. Please refer to the supp. material for the derivation of the Jacobian matrix of each term and more details on the Gaussian-Newton solver.

Final Scene Flow Prediction: Given the final rigid motion estimation for each instance $\boldsymbol{\xi}_i^*$, we are able to compute the dense instance-wise rigid scene flow. Our scene flow consists of three component, namely the first frame's stereo $\mathcal{D}^0$, warped stereo to second frame $\mathcal{D}^{\text{warp}}$ as well as the instance-wise rigid flow estimation $\mathcal{F}^{\text{rigid}}$. Specifically, for each point $\boldsymbol{p}$ we have:

$$\begin{aligned} \mathcal{D}^0(\boldsymbol{p}) &= \mathcal{D}^0(\boldsymbol{p}) \qquad (6) \\ \mathcal{D}^{\text{warp}}(\boldsymbol{p}) &= z_{\mathbf{K}}(\boldsymbol{\xi}^*_{\mathcal{S}^0_{\mathcal{L}}(\boldsymbol{p})} \circ \pi_{\mathbf{K}}^{-1}(\boldsymbol{p}, \mathcal{D}^0(\boldsymbol{p}))) \\ \mathcal{F}^{\text{rigid}}(\boldsymbol{p}) &= \boldsymbol{p}' - \boldsymbol{p} = \pi_{\mathbf{K}}(\boldsymbol{\xi} \circ \pi_{\mathbf{K}}^{-1}(\boldsymbol{p}, \mathcal{D}^0(\boldsymbol{p}))) - \boldsymbol{p} \end{aligned}$$

where $z_{\mathbf{K}}(\cdot)$ computes the disparity of the 3D point; $\pi_{\mathbf{K}}^{-1}$ is the inverse projection function; and $\boldsymbol{\xi} \circ \mathbf{x}$ transforms a 3D point $\mathbf{x}$ using the rigid motion $\boldsymbol{\xi}$.

3.4. Learning

The whole deep structured network can be trained end-to-end. In practice, we train our instance segmentation,

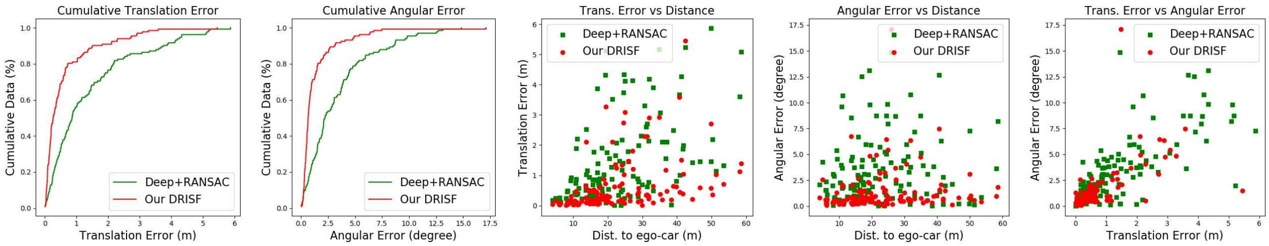

Figure 5: **3D rigid motion analysis:** Over 80% of the estimated 3D rigid motion has an error less than $1m$ and $1.3°$. Large errors often happen at farther distances where the vehicles are small and less points are observable.

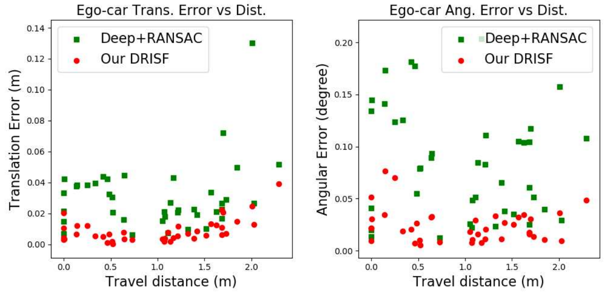

Figure 6: **Odometry from background motion:** On average, our ego-car drifts $0.9cm$ and $0.024°$ every $1m$ of drive.

flow estimation, and stereo estimation module respectively through back-propagation. To be more specific, Mask R-CNN model is pre-trained on Cityscapes and fine-tuned on KITTI. The loss function includes ROI classification loss, box regression loss as well as the mask segmentation loss. PSM-Net is pre-trained on Scene Flow [26] and fine-tuned on KITTI with L1 regression loss. PWC-Net is pre-trained on FlyingChairs [11] and FlyingThings [26] then fine-tuned over KITTI, with weighted L1 regression loss.

4. Experiments

In this section we first describe the experimental setup. Next we evaluate our model based on pixel-level scene flow metric and instance-level rigid motion metric. Finally we comprehensively study the characteristic of our model.

4.1. Dataset and Implementation Details

Data: We validate our approach on the KITTI scene flow dataset [27]. The dataset consists of 200 sets of training images and 200 sets of test images, captured on real world driving scenarios. Following [8], we divide the training data into *train*, *val* splits based on the 4:1 ratio.

Implementation details: For foreground objects, we use all energy terms. The weights are set to 1. For background, we only use photometric term (see ablation study). We run RANSAC 5 times and use the one with lowest mean energy as initialization. We unroll the GN solver for 50 steps. The solver terminates early if the energy reaches plateau. In practice, best energy are often reached within 10 iterations.

4.2. Scene Flow Estimation

Comparison to the state-of-the-art: We compare our approach against the leading methods on the benchmark[1]: ISF [3], PRSM [42], OSF+TC [28], SSF [34], OSF [27], and CSF [25]. Note that in addition to the standard two adjacent frames, PRSM and OSF+TC rely on extra temporal frames. As shown in Tab. 1, our approach (DRISF) outperforms all previous methods by a significant margin in both runtime and outliers ratio. It achieves state-of-the-art performance on almost every entry. DRISF reduces the D1 outliers ratio by **43%**, the D2 outliers ratio by **32%**, and the flow outliers ratio by **24%**. Comparing to ISF model [3], our scene flow error is **22%** lower and our runtime is **800** times faster. Fig. 1 compares the performance and runtime of all methods.

Qualitative results: To better understand the pros and cons of our approach, we visualize a few scene flow results on test set in Fig. 4. Scene flow estimation is challenging in these scenarios due to large vehicle motions, texture-less regions, occlusion, and illumination variation. For the left-most image, prior methods fail to estimate the vehicle's motion and adjacent area due to the sun reflection and occlusion. The saturated, high intensity pixels hinder photometric based approaches [27] from matching accurately. With the help of detection and segmentation, ISF [3] is able to improve the foreground estimation. Yet it still fails at the occluded background. In comparison, our approach is robust to illumination changes and is able to handle the occlusion by effectively separating the vehicle from the background. It can also accurately estimate the motion of the small car far away, as well as those of the traffic sticks aside. As we only train our Mask R-CNN on vehicles, it fails to segment the train and hence the failure of our model. For the middle image, the texture-less car has a large displacement and is occluded in the second frame. While previous approaches failed substantially, our method is able to produce accurate motion estimation through the inferred flow and disparity of

[1] As the validation performance of our PWC-Net (fine-tuned on 160 images) performs slightly worse than the official one (fine-tuned on all 200 images), we use their weights instead when submitting to the benchmark. All other settings remain intact. We thank Deqing Sun for his help.

Employed energy			Background outliers ($\%$)				Employed energy			Foreground outliers ($\%$)			
E_{pho}	E_{flow}	E_{rigid}	D1	D2	Fl	SF	E_{pho}	E_{flow}	E_{rigid}	D1	D2	Fl	SF
✓			1.92	2.69	**3.71**	**4.30**	✓			1.70	**4.25**	7.57	9.00
	✓	✓	1.92	**2.56**	4.72	5.28		✓	✓	1.70	4.58	6.98	8.67
✓	✓	✓	1.92	**2.56**	4.63	5.21	✓	✓	✓	1.70	4.56	**6.73**	**8.39**

Table 2: **Contributions of each energy:** As foreground objects sometimes are texture-less and have large displacement, simple photometric term is not enough. In contrast, background is full of disriminative cues. Simple photometric error would suffice. Adding extra terms will introduce noises and hurt the performance. Please refer to the supp. material for full table.

Methods	D1-all	D2-all	Fl-all	SF-all
PSM + PWC	1.89	(47.0)	11.0	(50.8)
Deep+RANSAC	1.89	**2.75**	7.65	8.26
Our Full DRISF	1.89	2.89	**4.10**	**4.84**

Table 3: **Improvement over original flow/stereo estimation on validation set:** The numbers in parenthis are obtained by simply warping the disparity output with optical flow, without interpolation, occlusion handling, etc.

Module	Stereo	Optical Flow	Segmentation
Inference time	409 ms / pair	30 ms / pair	251 ms / pair
Module	RANSAC	GN Solver	Total
Inference time	93 ms / instance	244 ms / instance	746 ms / pair

Table 4: **Runtime analysis.** Modules within each building block can be executed in parallel (see text for more details).

the remaining non-occluded part. The middle failure mode is again due to the inaccurate segmentation.

4.3. 3D Rigid Motion Estimation

We now evaluate how good our DRISF model is at estimating the 3D rigid motion. Towards this goal, we exploit the ground truth optical flow, disparity, and instance segmentation provided in the KITTI scene flow dataset to fit a least square rigid motion for each object instance in order to create the ground truth rigid motion.

Curating KITTI scene flow: During fitting, we discover two critical issues with KITTI: first, there are misalignments between GT flow/disparity and GT segmentation. Second, the scale fitting of the same 3D CAD model employed to compute ground truth changes sometimes across frames. The first issue is due to the fact that the GT are collected via different means and thus not consistent. While the GT flow and GT disparity are obtained from the fitted 3D CAD models, the GT segmentation are based on human annotation. To address this, we first use the GT segmentation mask to define each object instance. We then fit a rigid motion using the GT flow and GT disparity of each instance via least squares. Since some boundary pixels may be mis-labeled by the annotators, for each pixel around the boundary we search if there are other instances in the surrounding area, and if there are, we transform the pixel with their rigid motion. If their rigid motion better explains the pixel's 3D movement, *i.e.* the 3D distance is closer, then we assign the pixel to that instance. At the end, we perform the least square fitting again with the new pixel assignment. Unfortunately, even after re-labeling, there are still a few vehicle instances where the rigid motion cannot be explained. After careful diagnose, we notice that this is because the scale of the CAD model changes across frames. To verify our hypothesis, we compute the eigen decomposition for the same instance across frames. Ideally if the scale of the instance does not change much, the eigen value should be roughly the same. Yet we discover a few examples where the largest eigen value changes by 7%. We simply prune those instances as the GT is not accurate.

3D Motion evaluation: Most scene flow methods are pixel-based or adopted a piece-wise rigid setting. It is unclear how to aggregate their estimation into instance-based motion model without affecting their performance. In light of this, we exploit the motion initialization of our GN Solver as baseline. We take the output of the deep nets and apply RANSAC to find the best rigid motion. We denote it as Deep+RANSAC. As shown in Tab. 3, this baseline is very competitive. Its performance is comparable to, or even better than prior state-of-the-art. We evaluate our motion model based on translation error and angular error. As shown in Fig. 5, over 80% of the vehicles have translation error less than $1m$ and angular error less than 1.3°. Furthermore, most vehicles with translation error larger than $1m$ is at least $20m$ away. In general, both error slightly increase with distance. This is expected as the farther the vehicle is, the less observations we have. The translation error and angular error are also strongly correlated.

Visual odometry: The odometry of the 'ego-car' can be computed by estimating the background movement. As a proof-of-concept, we compute the per frame odometry error on the validation images. On average our motion model drifts $0.09m$ and 0.24° every $10m$. Fig. 6 shows the detailed odometry error w.r.t. the travel distance. We note that the current result is without any pose filter, loop closure, etc. We plan to exploit this direction further in the future.

4.4. Analysis

Ablation study: To understand the effectiveness of each energy term on background and foreground objects, we evaluate our model with different energy combinations. As shown in Tab. 2, best performance is achieved for fore-

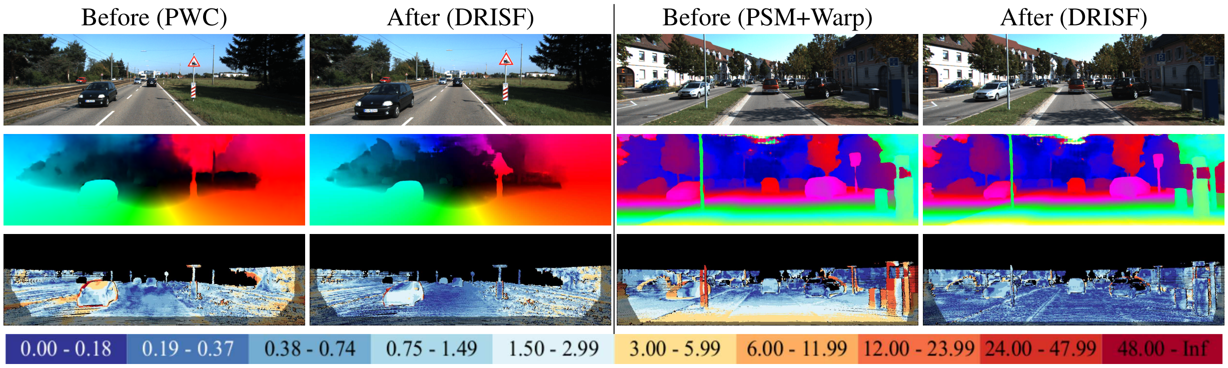

Figure 7: **Improvement over original flow/stereo:** DRISF improves the overall performance. It is especially effective on texture-less regions (*e.g*. window of the black car on the left) and occluded areas (right).

ground objects when using all energy terms, while for background the error is lowest when employing only photometric term. This can be explained by the fact that vehicles are often texture-less, and sometimes have large displacements. If we only employ photometric term, it will be very difficult to establish correspondences and handle drastic appearances changes. With the help of flow and rigid term, we can guide the motion and reduce such effect, and deal with occlusions. In contrast, background is full of discriminative textures and has relatively small motion, which is ideal for photometric term. Adding other terms may introduce extra noise and degrade the performance.

Comparison against original flow/disparity: Through exploiting the structure between visual cues and occlusion handling, our model is able to improve the performance both quantitatively (Tab. 3) and qualitatively (Fig. 7). The object motion estimation is better, the boundaries are sharper, and the occlusion error is greatly reduced, suggesting that incorporating prior knowledge, such as pixels of same instance should have same rigid motion, into the model is crucial for the task.

Potential improvement To understand the potential gain we may enjoy when improving each module, we sequentially replace the input to our solver with ground truth, one by one, and evaluate our model. Replacing D1 and flow with GT reduce the scene flow error rate by 8% and 21% respectively, while substituting GT for segmentation does not improve the results. This suggests that there are still space for flow and stereo modules to improve.

Runtime analysis We benchmark the runtime of each component in the model during inference in Tab. 4. The whole inference pipeline can be decomposed into three sequential stages: visual cues extraction, occlusion reasoning, and optimization. As modules within the same stage are independent, they can be executed in parallel. Furthermore, modern self-driving vehicles are equipped with multiple GPUs. The runtime for each stage is thus the max over all parallel modules. In practice, we exploit two Nvidia 1080Ti GPUs to extract the visual cues: one for PSM-Net, and the other for Mask R-CNN and PWC-Net. Currently, the stereo module takes more than 50% of the overall time. This is largely due to the 3D CNN cost aggregation and the stacked hourglass refinement. In the future, we plan to investigate other faster yet reliable stereo networks. The runtime of the GN solver depends highly on the number of steps we unroll and the number of points we consider. Please refer to the supp. material for detailed analysis.

Limitations: DRISF has two main limitations: first, it heavily depends on the performance of the segmentation network. If the segmentation module fails to detect a vehicle, the vehicle will be treated as background and assigned an inverse ego-car motion. In this case, the 3D motion might be completely wrong, even if the optical flow network accurately predicts its flow. In the future we plan to address this by jointly reasoning about instance segmentation and scene flow. Second, the current energy functions are highly flow centric. Only the photometric term is independent of flow. If the optical flow network completely failed, it would be difficult for the solver to recover the correct motion. One possible solution is thus adding more flow-invariant energy terms, such as instance association between adjacent frames.

5. Conclusion

In this paper we develop a novel deep structured model for 3D scene flow estimation. We focus on the self-driving scenario where the motion of the scene can be composed by estimating the 3D rigid motion of each actor. We first exploit deep learning to extract visual cues for each instance. Then we employ multiple geometry based energy functions to encode the structural geometric relationship between them. Through optimizing the energy function, we can reason the 3D motion of each traffic participant, and thus scene flow. All operations, including the Gassian-Newton solver, are done in GPU. Our method acheives state-of-the-art performance on the KITTI scene flow dataset. It outperforms all previous methods by a huge margin in both runtime and accuracy. Comparing to prior art, DRISF is 22% better while being two to three orders of magnitude faster.

References

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

[2] Tali Basha, Yael Moses, and Nahum Kiryati. Multi-view scene flow estimation: A view centered variational approach. *IJCV*, 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*, 2017. 1, 2, 4, 6

[4] Michael Black and T. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *CVIU*, 1996. 2

[5] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 1

[6] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004. 2

[7] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier analysis and applications*, 2008. 5

[8] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, 2018. 2, 3, 6

[9] Zhuoyuan Chen, Xun Sun, Liang Wang, Yinan Yu, and Chang Huang. A deep visual correspondence embedding model for stereo matching costs. In *ICCV*, 2015. 2

[10] 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 *CVPR*, 2016. 3

[11] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick Van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *arXiv*, 2015. 1, 2, 6

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 3

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 2, 3

[15] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *TPAMI*, 2008. 2

[16] William Hoff and Narendra Ahuja. Surfaces from stereo: Integrating feature matching, disparity estimation, and contour detection. *TPAMI*, 1989. 2

[17] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 1981. 2

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

[19] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *CVPR*, 2018. 2

[20] 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 *CVPR*, 2017. 2

[21] 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. 2018. 2

[22] Takeo Kanade and Masatoshi Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. In *ICRA*, 1991. 2

[23] 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. 2017. 2

[24] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *CVPR*, 2016. 1, 2

[25] 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 *ECCV*, 2016. 2, 4, 6

[26] Nikolaus Mayer, Eddy Ilg, Philip Hausser, 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*, 2016. 1, 2, 6

[27] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, 2015. 1, 2, 4, 6

[28] Michal Neoral and Jan Šochman. Object scene flow with temporal consistency. In *CVWW*, 2017. 4, 6

[29] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 2

[30] Nils Papenberg, Andrés Bruhn, Thomas Brox, Stephan Didas, and Joachim Weickert. Highly accurate optic flow computation with theoretically justified warping. *IJCV*, 2006. 2

[31] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *IJCV*, 2007. 2

[32] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, 2017. 1, 2

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 3

[34] Zhile Ren, Deqing Sun, Jan Kautz, and Erik Sudderth. Cascaded scene flow prediction using semantic segmentation. In *3DV*, 2017. 2, 4, 6

[35] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015. 2

[36] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010. 4

[37] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of cnns for optical flow estimation. *arXiv*, 2018. 2

[38] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 1, 2, 3

[39] Levi Valgaerts, Andrés Bruhn, Henning Zimmer, Joachim Weickert, Carsten Stoll, and Christian Theobalt. Joint estimation of motion, structure and geometry from stereo sequences. In *ECCV*, 2010. 2

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

[41] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *ICCV*, 2013. 1, 2

[42] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a piecewise rigid scene model. *IJCV*, 2015. 4, 6

[43] Shenlong Wang, Linjie Luo, Ning Zhang, and Jia Li. Autoscaler: Scale-attention networks for visual correspondence. *arXiv*, 2016. 2

[44] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *ECCV*, 2014. 1, 2

[45] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015. 2

[46] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *CVPR*, 2015. 1, 2

[47] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *JMLR*, 2016. 2

[48] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 2, 3