MonoPLFlowNet: Permutohedral Lattice FlowNet for Real-Scale 3D Scene Flow Estimation with Monocular Images

Runfa Li(✉) and Truong Nguyen

UC San Diego, La Jolla, USA
{rul002,tqn001}@ucsd.edu

Abstract. Real-scale scene flow estimation has become increasingly important for 3D computer vision. Some works successfully estimate real-scale 3D scene flow with LiDAR. However, these ubiquitous and expensive sensors are still unlikely to be equipped widely for real application. Other works use monocular images to estimate scene flow, but their scene flow estimations are normalized with scale ambiguity, where additional depth or point cloud ground truth are required to recover the real scale. Even though they perform well in 2D, these works do not provide accurate and reliable 3D estimates. We present a deep learning architecture on permutohedral lattice - MonoPLFlowNet. Different from all previous works, our MonoPLFlowNet is the first work where only two consecutive monocular images are used as input, while both depth and 3D scene flow are estimated in real scale. Our real-scale scene flow estimation outperforms all state-of-the-art monocular-image based works recovered to real scale by ground truth, and is comparable to LiDAR approaches. As a by-product, our real-scale depth estimation is also comparable to other state-of-the-art works.

Keywords: Real-scale scene flow estimation · Monocular-image based · Permutohedral lattice · Real-scale depth estimation

1 Introduction

Scene flow are 3D vectors associating the corresponding 3D point-wise motion between consecutive frames, where scene flow can be recognized as lifting up pixel-wise 2D optical flow from the image plane to 3D space. Different to coarsely high-level motion cues such as bounding-box based tracking, 3D scene flow focuses on precisely low-level point-wise motion cues. With such advantage, scene flow can either serve for non-rigid motion as visual odometry and ego motion estimation, or rigid motion as multi-object tracking, which makes it increasingly

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-19812-0_19.

S. Avidan et al. (Eds.): ECCV 2022, LNCS 13687, pp. 322–339, 2022.
https://doi.org/10.1007/978-3-031-19812-0_19

important in motion perception/segmentation and applications in dynamic environments such as robotics, autonomous driving and human-computer interaction (Fig. 1).

Fig. 1. Results of our MonoPLFlowNet. With only two consecutive monocular images (*left*) as input, our MonoPLFlowNet estimates both the depth (*middle*) and 3D scene flow (*right*) in real scale. *Right* shows a zoom-in real-scale scene flow of the two vehicles from side view with the pseudo point cloud generating from the estimated depth map (middle), where blue points are from frame t, red and green points are blue points translated to frame $t+1$ by ground truth and estimated 3D scene flow, respectively. The objective is to align green and red points. (Color figure online)

3D scene flow has been widely studied using LiDAR point cloud [3,15,27,33,40,44,46] from two consecutive frames as input, where a few recent LiDAR works achieve very accurate performances. However, LiDARs are still too expensive to be equipped for real applications. Other sensors are also being explored for 3D scene flow estimation such as RGB-D cameras [16,20,29] and stereo cameras [2,22,30,39,42]. However, each sensor configuration also has its own limitation, such as RGB-D cameras are only reliable in the indoor environment, while stereo cameras require calibration for stereo rigs.

Since Monocular camera is ubiquitous and affordable for all real applications, it is a promising alternative to the complicated and expensive sensors. There are many works for monocular image-based scene flow estimation [8,18,19,28,37,48,50,52,54], where CNN models are designed to jointly estimate monocular depth and optical/scene flow. However, their estimations are all with scale ambiguity. This problem exists in all monocular works where they estimate normalized depth and optical/scene flow. To recover to the real scale, they require depth and scene flow ground truth, which are possible to obtain for evaluation in labeled datasets but impossible for a real application.

Our motivation for this work is to take the advantages and overcome the limitations from both LiDAR-based (real-scale, accurate but expensive) and image-based (affordable, ubiquitous but scale-ambiguous) approaches. Our key contributions are:

- We build a deep learning architecture MonoPLFlowNet, which is the first work using only two consecutive monocular images to simultaneously estimate in real scale both the depth and 3D scene flow.
- Our 3D scene flow estimation outperforms all state-of-the-art monocular image-based works even after recovering their scale ambiguities with ground truth, and is comparable to LiDAR-based approaches.

- We introduce a novel method - "Pyramid-Level 2D-3D features alignment between Cartesian Coordinate and Permutohedral Lattice Network", which bridges the gap between features in monocular images and 3D points in our application, and inspires a new way to align 2D-3D information for computer vision tasks which could benefit other applications.
- As a byproduct, our linear-additive depth estimation from MonoPLFlowNet also outperforms state-of-the-art works on monocular depth estimation.

2 Related Works

Monocular Image-Based Scene Flow Estimation: Monocular 3D scene flow estimation originates from 2D optical flow estimation [9,48]. To get real-scale 3D scene flow from 2D optical flow, real-scale 3D coordinates are required which could be derived from real-scale depth map. Recently, following the success from SFM (Structure from Motion) [25,47], many works jointly estimate monocular depth and 2D optical flow [8,28,37,50,52,54]. However, as seen in most SFM models, the real scale decays in jointly training and leads to scale ambiguity. Although [18,19] jointly estimate depth and 3D scene flow directly, they suffer from scale ambiguity, which is the biggest issue for monocular image-based approaches.

3D Point Cloud Based Scene Flow Estimation: Following PointNet [6] and PointNet++ [35], it became possible to use CNN-based models to directly process point cloud for different tasks including 3D scene flow estimation. Since directly implementing on 3D points, there is no scale ambiguity. The success of CNN-based 2D optical flow networks also boost 3D scene flow. FlowNet3D [27] builds on PointNet++ and imitates the process used in 2D optical FlowNet [9] to build a 3D correlation layer. PointPWC-net [46] imitates another 2D optical flow network PWC-net [41] to build 3D cost volume layers. The most successful works are "Permutohedral Lattice family", where BCL [21], SplatNet [40], HPLFlowNet [15], PointFlowNet [3] all belong to the family. The lattice (more details will be provided in Sect. 3.1) is a very efficient representation for processing of high-dimensional data [1], including 3D point cloud. Point cloud based methods achieve better performance than image-based methods. However, with LiDAR scanning as the input, they are still too expensive for real applications.

Our MonoPLFlowNet takes advantages and overcomes limitations from both image (affordable, ubiquitous but scale-ambiguous) and LiDAR (real-scale, accurate but expensive) based approaches by only using monocular images to jointly estimate depth and 3D scene flow in real scale.

3 MonoPLFlowNet

Our MonoPLFlowNet is an encoder-decoder based model, which only takes two consecutive monocular images as input, while both the depth and 3D scene flow are estimated in real scale. Figure 2 shows our network architecture. While sharing the same encoder, we build depth and scene flow decoders, respectively. With

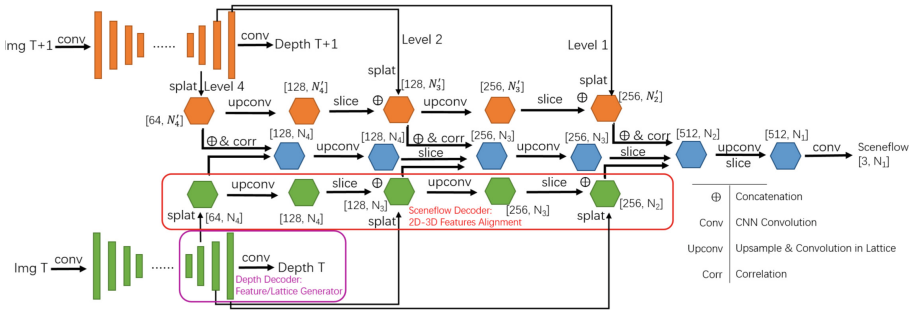

Fig. 2. MonoPLFlowNet Architecture: It shares the same encoder for two monocular images as the only input, and jointly estimates the depth and 3D scene flow in real scale by decoding separately with a *Depth decoder (purple box)* and a *Sceneflow Decoder (red box)*. Architectures of the two decoders are shown in Fig. 3 and 4, respectively. (Color figure online)

our designed mechanism, we align the 2D-3D features to boost the performance, where the real-scale 3D scene flow benefits from the depth estimation. In this section, we present the theory of permutohedral lattice, and then discuss how we design and align the two decoders.

3.1 Review of Permutohedral Lattice Filtering

High-Dimensional Gaussian Filtering: Value and position of the signal are two important components of filtering. Equation 1 shows a general form of high-dimensional Gaussian filtering:

$$\vec{v_i} = \sum_{j=1}^{n} exp^{-\frac{1}{2}(\vec{p_i}-\vec{p_j})^T \sum^{-1}(\vec{p_i}-\vec{p_j})^T} \vec{v_j} \tag{1}$$

$$\vec{v_i} = (r_i, g_i, b_i, 1),\ \vec{p_i} = (\frac{x_i}{\sigma_s}, \frac{y_i}{\sigma_s}) \tag{2}$$

$$\vec{v_i} = (r_i, g_i, b_i, 1),\ \vec{p_i} = (\frac{x_i}{\sigma_s}, \frac{y_i}{\sigma_s}, \frac{I_i}{\sigma_c}) \tag{3}$$

$$\vec{v_i} = (r_i, g_i, b_i, 1),\ \vec{p_i} = (\frac{x_i}{\sigma_s}, \frac{y_i}{\sigma_s}, \frac{r_i}{\sigma_c}, \frac{g_i}{\sigma_c}, \frac{b_i}{\sigma_c}) \tag{4}$$

where $exp^{-\frac{1}{2}(\vec{p_i}-\vec{p_j})^T \sum^{-1}(\vec{p_i}-\vec{p_j})^T}$ is a Gaussian distribution denoting the weight of the neighbor signal value $\vec{v_j}$ contributing to the target signal $\vec{v_i}$. Here $\vec{v}$ is the value vector of the signal, and $\vec{p}$ is the position vector of the signal. Equations 2, 3 and 4 denote Gaussian blur filter, gray-scale bilateral filter and color bilateral filter, respectively, where v_i and p_i are defined in Eq. 1. For these image filters, the signals are pixels, the signal values are 3D homogeneous color space, and signal positions are 2D, 3D and 5D, respectively, and the filtering processing is

on 2D image Cartesian coordinate. The dimension of the position vector can be extended to d, which is called a d-dimensional Gaussian filter. We refer the readers to [1,21] and our supplementary materials for more details.

Permutohedral Lattice Network: However, a high-dimensional Gaussian filter can also be implemented on features with n dimensions rather than on pixels with 3D color space. The filtering process can be implemented in a more efficient space, the Permutohedral Lattice rather than the classical Cartesian 2D image plane or 3D space. The d-dimensional permutohedral lattice is defined as the projection of the scaled Cartesian $(d+1)$-dimensional grid along the vector $\vec{1} = [1, 1...1]$ onto the hyperplane H_d, which is the subspace of R_{d+1} in which coordinates sum to zero:

$$B_d = \begin{bmatrix} d & -1 & ... & -1 \\ -1 & d & ... & -1 \\ ... & ... & ... & ... \\ -1 & -1 & ... & d \end{bmatrix} \tag{5}$$

Therefore it is also spanned by the projection of the standard basis for the original $(d+1)$-dimensional Cartesian space to H_d. In other words, we can project a feature at Cartesian coordinate p_{cart} to Permutohedral Lattice coordinate p_{lat} as:

$$\vec{p_{lat}} = B_d \vec{p_{cart}} \tag{6}$$

With features embedding on lattice, [1] derived a maneuver pipeline "splatting-convolution-slicing" for feature processing in lattice; [21] improved the pipeline to be learnable BCL (Bilateral Convolutional Layers). Following CNN notion, [15] improved BCL to different levels of receptive fields.

To summarize, Permutohedral Lattice Network convolves the feature values $\vec{v_i}$ based on feature positions $\vec{p_i}$. While it is straightforward to derive $\vec{v_i}$ and $\vec{p_i}$ when feature positions and lattice have the same dimension, e.g. 3D Lidar points input to 3D lattice [15,21,40] or 2D to 2D [1,45], it is challenging when dimensions are different, as in the case of our 2D image input to 3D lattice. The proposed MonoPLFlowNet is designed to overcome such a problem.

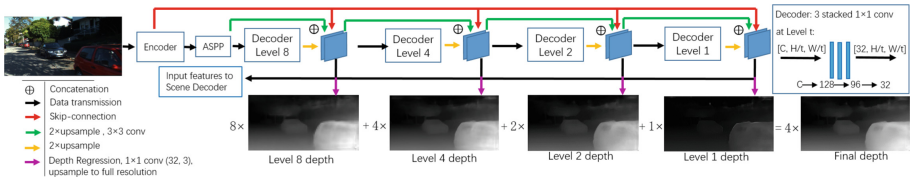

Fig. 3. Depth Decoder: Pyramid-Level Linearly-Additive Depth Estimation. Besides the accurate depth estimation, from a whole architecture view, our depth decoder also serves as the feature/lattice position generator, which is why we design it in linear-additive way.

3.2 Depth Decoder

We design our depth module as an encoder-decoder based network as shown in Fig. 3. Following the success of BTS DepthNet [24], we use their same encoder as a CNN-based feature extractor followed by the dilated convolution (ASPP) [7,49]. Our major contribution focuses on the decoder from three aspects.

Level-Based Decoder: First, while keeping a pyramid-level top-down reasoning, BTS designed "lpg" (local planar guidance) to regress depth at each level in an ambiguous scale. In our experiments, we found that "lpg" is a block where accuracy is sacrificed for efficiency. Instead, we replace the "lpg" with our "level-based decoder" as shown in Fig. 3, and improve the decoder to accommodate our sceneflow task.

Pyramid-Level Linearly-Additive Mechanism: BTS concatenates the estimated depth at each level and utilizes a final convolution to regress the final depth in a non-linear way, implying that the estimated depth at each level is not in a determined scale. Instead, we propose a "Pyramid-Level Linearly-Additive mechanism" as:

$$d_{final} = \frac{1}{4}(1 \times d_{lev1} + 2 \times d_{lev2} + 4 \times d_{lev4} + 8 \times d_{lev8}) \tag{7}$$

such that the final estimation is a linear combination over each level depth, where we force d_{lev1}, d_{lev2}, d_{lev4}, d_{lev8} to be in $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$ scale of the real-scale depth. To achieve this, we also derive a pyramid-level loss corresponding to the level of the decoder architecture to supervise the depth at each level as Eq. 11.

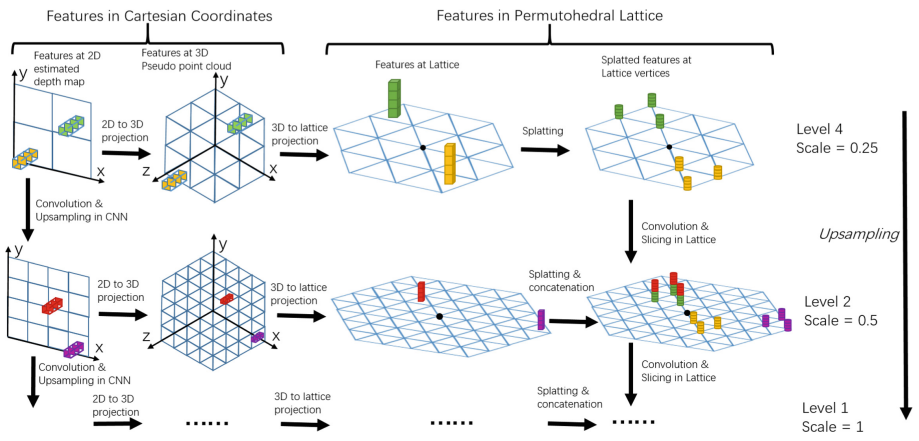

Fig. 4. 3D Scene Flow Decoder: Pyramid-Level 2D-3D features alignment between Cartesian Coordinate and Permutohedral Lattice Network. Horizontally it shows the feature embedding from Cartesian coordinate to permutohedral lattice. Vertically, it shows how the features are upsampled and concatenated for the next level.

Experiments show that with our improvement, our depth decoder outperforms the baseline BTS as well as other state-of-the-art works. More importantly, the objective is to design the depth decoder to generate feature/lattice for the scene flow decoder, which is our major contribution. More details are provided in the next section.

3.3 Scene Flow Decoder

Our scene flow decoder is designed as a two-stream pyramid-level model which decodes in parallel in two domains - 2D Cartesian coordinate and 3D permutalhedral lattice, as shown in Fig. 4. As mentioned in Sect. 3.1, feature values and positions are two key factors in filtering processing. For the feature values, we use the features decoded by level-based decoder from depth module as shown in Fig. 3, the features as the input to scene decoder (32 dimensions) are before regressing to the corresponding level depth (purple arrow), which means that the input features to the scene decoder are the high-level feature representation of the corresponding level depth. While values are trivial, positions are not straightforward. As the challenge mentioned in Sect. 3.1, the 2D feature position from our depth is not enough to project the feature values to 3D permutohedral lattice. We overcome it by projecting the estimated depth estimation into 3D space to generate real-scale pseudo point cloud. Since pseudo point cloud has real-scale 3D coordinates, we can first project the feature values from the 2D depth map to 3D point cloud in Cartesian coordinate, and then embed to the corresponding position in 3D permutohedral lattice. A 2D to 3D projection is defined as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} z/f_u & 0 & -c_u z/f_u \\ 0 & z/f_v & -c_v z/f_v \\ 0 & 0 & z \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{8}$$

where z is the estimated depth, (u, v) is the corresponding pixel coordinate in the depth map, (f_u, f_v) are horizontal and vertical camera focal lengths, (c_u, c_v) are the coordinate of camera principle point. (x, y, z) is the coordinate of the projected 3D point.

So far we have successfully derived the projection in the real scale, where the complete projection pipeline consisting of 2D to 3D (Eq. 8) and 3D to lattice (Eq. 6) is shown in Fig. 4. Due to the linear property of the projection, the proposed projection pipeline holds at different scales in the overall system. Using this property, we prove a stronger conclusion: scaling the feature depth from Cartesian coordinate leads to a same scale to the corresponding feature position in permutohedral lattice. Equation 9 summarizes the mapping where (p_x, p_y, p_z) is the permutohedral lattice coordinate of the feature corresponding to its 2D Cartisian coordinate in the depth map.

$$
\begin{aligned}
\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} &= B_d \begin{bmatrix} x \\ y \\ z \end{bmatrix} \\
&= B_d \begin{bmatrix} z/f_u & 0 & -c_u z/f_u \\ 0 & z/f_v & -c_u z/f_u \\ 0 & 0 & z \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \\
&= B_d \begin{bmatrix} \lambda z/\lambda f_u & 0 & -\lambda c_u \lambda z/\lambda f_u \\ 0 & \lambda z/\lambda f_v & -\lambda c_u \lambda z/\lambda f_u \\ 0 & 0 & \lambda z \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \\
&= B_d \begin{bmatrix} z/f_u & 0 & -c_u \lambda z/f_u \\ 0 & z/f_v & -c_u \lambda z/f_u \\ 0 & 0 & \lambda z \end{bmatrix} \begin{bmatrix} \lambda u \\ \lambda v \\ 1 \end{bmatrix} \\
&= B_d \begin{bmatrix} \lambda z/f_u & 0 & -c_u \lambda z/f_u \\ 0 & \lambda z/f_v & -c_u \lambda z/f_u \\ 0 & 0 & \lambda z \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}
\end{aligned} \tag{9}
$$

B_d is defined in Eq. 5, λ is the scale, a "prime" sign denotes scaling the original value with λ. Comparing the final result of Eq. 9 to Eq. 8, the only difference is to replace z with λz, hence only scaling the depth in Cartesian coordinate will lead to a same scale to the position in permutohedral lattice. Using the proposed "Pyramid-Level 2D-3D features alignment" mechanism, we can embed the features from 2D Cartesian coordinate to 3D Permutohetral lattice, and then implement splating-convolution-slicing and concatenate different level features directly in the permutohedral lattice network. Please see the supplementary materials for more explanation. For the basic operations in permutohedral lattice, we directly refer to [15].

We further analysis into the motivation and intuition of model design. We design the explicit "linear regression" depth decoder (Eq. 7) to produce deep features and depth at each level for scene branch, implying that the depth estimation at level 2 is at scale 1/2 to the final depth, at level 4 is 1/4 and so on, which guarantees that we can embed the features at level n from the depth map to the corresponding level n lattice (Proved by Eq. 9). However, for the implicit "non-linear regression" in BTS 3, the depth estimation at level 2 is not 1/2 but an arbitrary and ambiguous scale to the final depth and so as other levels, which means that the features cannot be safely embedded to the lattice coordinate corresponding to the Euclidean coordinate. Note that this cannot be achieved by a simple concatenation of depth and scene branch, because not only the depth but also the deep features at corresponding position in 2D map are needed for scene branch. In other words, scene branch cannot work alone without depth module, and thus we take the two modules as a whole model rather than two independent models. This is the "Pyramid-Level 2D-3D features alignment between Cartesian Coordinate and Permutohedral Lattice Network" explained in Fig. 4. Our model design also considers the meta information aggregation, monocular input (like Mono-SF [18]) contains RGB, X, Y information, point cloud input (like HPLFlowNet [15]) contains X, Y, Z information, while our real-scale depth

estimation from monocular input plus deep features extracted from the depth decoder contains information RGB, X, Y, Z. Moreover, the proposed network has advantage over Mono-SF/MonoSF-Multi [19] since our estimation is in real-scale whereas their estimation has scale ambiguity.

3.4 Loss

Depth Loss: Silog (scale-invariant log) loss is a widely-used loss [10] for depth estimation supervision defined as:

$$L_{silog}(\tilde{d}, d) = \alpha \sqrt{\frac{1}{T}\sum_i (g_i)^2 - \frac{\lambda}{T^2}(\sum_i g_i)^2} \tag{10}$$

where $g_i = \log \tilde{d}_i - \log d_i$, $\tilde{d}$ and d are estimated and ground truth depths, T is the number of valid pixels, λ and α are constants set to be 0.85 and 10. Since our depth decoder decodes a fixed-scale depth at each level, we do not directly supervise on final depth. Instead, we design a pyramid-level silog loss corresponding to our depth decoder to supervise the estimation from each level.

$$L_{depth} = \frac{1}{15}(8 \times L_1 + 4 \times L_2 + 2 \times L_4 + 1 \times L_8) \tag{11}$$

where $L_{level} = L_{silog}(\tilde{d}_{level}, d_{level}/n)$. Higher weight is assigned to low-level loss to stabilize the training process.

Scene Flow Loss: Following most LiDAR-based work, we first use a traditional End Point Error (EPE3D) loss as $L_{epe} = ||\tilde{sf} - sf||_2$, where $\tilde{sf}$ and sf are estimated and ground truth scene flows, respectively. To bring two sets of point clouds together, some self-supervised works leverage the Chamfer distance loss as:

$$L_{cham}(P, Q) = \sum_{p \in P} \min_{q \in Q} ||p - q||_2^2 + \sum_{q \in Q} \min_{p \in P} ||p - q||_2^2 \tag{12}$$

where P and Q are two sets of point clouds that optimized to be close to each other. While EPE loss supervises directly on scene flow 3D vectors, we improved canonical Chamfer loss to a forward-backward Chamfer distance loss supervising on our pseudo point cloud from depth estimates as:

$$\begin{aligned} L_{cham\_total} &= L_{cham\_f} + L_{cham\_b} \\ L_{cham\_f} &= L_{cham}(\tilde{P}_f, P_2) \\ L_{cham\_b} &= L_{cham}(\tilde{P}_b, P_1) \\ \tilde{P}_f = P_1 + \tilde{sf}_f,\ \tilde{P}_b &= P_2 + \tilde{sf}_b \end{aligned} \tag{13}$$

where P_1 and P_2 are pseudo point clouds generated from the estimated depth of two consecutive frames. $\tilde{sf}_f$ and $\tilde{sf}_b$ are estimated forward and backward scene flows.

4 Experiments

Datasets: We use Flyingthings3D [31] and KITTI [13] dataset in this work for training and evaluation. Flyingthings3D is a synthetic dataset with 19,460 pairs of images in its training split, and 3,824 pairs of images in its evaluation split. We use it for training and evaluation of both depth and scene flow estimation. For KITTI dataset, following most previous works, for depth training and evaluation, we use KITTI Eigen's [10] split which has 23,488 images of 32 scenes for training and 697 images of 29 scenes for evaluation. For scene flow evaluation, we use KITTI flow 2015 [32] split with 200 pairs of images labeled with flow ground truth. We do not train scene flow on KITTI.

4.1 Monocular Depth

We train the depth module in a fully-supervised manner using the pyramid-level silog loss derived in Eq. 11. For training simplicity, we first train the depth module free from scene flow decoder. Our model is completely trained from scratch.

Table 1. Monocular depth results comparison on KITTI Eigen's split. In the column *Output*, *D* denotes depth, *2DF* and *3DF* denote 2D optical flow and 3D scene flow. In the column *Scale*, ✓ denotes in real scale, ✗ denotes with scale ambiguity. DenseNet-121 as backbone more efficient than ResNext-101.

Method	Output	Scale	*Higher is better*			*Lower is better*			
			$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Abs Rel	Sq Rel	RMSE	RMSE *log*
Make3D [38]	D	✓	0.601	0.820	0.926	0.280	3.012	8.734	0.361
Eigen et al. [10]	D	✓	0.702	0.898	0.967	0.203	1.548	6.307	0.282
Liu et al. [26]	D	✓	0.680	0.898	0.967	0.201	1.584	6.471	0.273
LRC (CS + K) [14]	D	✓	0.861	0.949	0.976	0.114	0.898	4.935	0.206
Kuznietsov et al. [23]	D	✓	0.862	0.960	0.986	0.113	0.741	4.621	0.189
Gan et al. [12]	D	✓	0.890	0.964	0.985	0.098	0.666	3.933	0.173
DORN [11]	D	✓	0.932	0.984	0.994	0.072	0.307	2.727	0.120
Yin et al. [51]	D	✓	0.938	0.990	0.998	0.072	-	3.258	0.117
BTS (DenseNet-121) [24]	D	✓	0.951	0.993	0.998	0.063	0.256	2.850	0.100
BTS (ResNext-101) [24]	D	✓	0.956	0.993	0.998	0.059	0.245	2.756	0.096
DPT-hybrid [36]	D	✓	0.959	**0.995**	**0.999**	0.062	0.256	**2.573**	**0.092**
GeoNet [52]	D + 2DF	✗	0.793	0.931	0.973	0.155	1.296	5.857	0.233
DFNet [54]	D + 2DF	✗	0.818	0.943	0.978	0.146	1.182	5.215	0.213
CC [37]	D + 2DF	✗	0.826	0.941	0.975	0.140	1.070	5.326	0.217
GLNet [8]	D + 2DF	✗	0.841	0.948	0.980	0.135	1.070	5.230	0.210
EPC [50]	D + 2DF	✗	0.847	0.926	0.969	0.127	1.239	6.247	0.214
EPC++ [28]	D + 2DF	✗	0.841	0.946	0.979	0.127	0.936	5.008	0.209
Mono-SF [18]	D + 3DF	✗	0.851	0.950	0.978	0.125	0.978	4.877	0.208
Ours (DenseNet-121)	**D + 3DF**	✓	**0.960**	0.993	0.998	**0.059**	**0.236**	2.691	0.095

KITTI: We first train on KITTI Eigen's training split, Table 1 shows the depth comparison on KITTI Eigen's evaluation split. We classify the previous works

into two categories, joint-estimate monocular depth and flow, and single-estimate monocular depth alone. It is clearly from the table that the single estimation outperforms the joint estimation on average, and the joint estimation has scale ambiguity. With a similar design to our strongest single-estimate baseline BTS [24], our depth outperforms BTS with the same backbone DenseNet-121 [17] as well as the best BTS with ResNext-101 backbone. The joint-estimate works fail to estimate in real scale because they regress depth and scene flow together in self-supervised manner, which sacrifices the real depth. We design our model with separate decoders in a fully-supervised manner, and succeed to jointly estimate depth and scene flow in real scale, where our depth achieves a clear improvement to the strongest joint-estimate baseline Mono-SF [18]. The monocular depth evaluation metrics cannot show the difference of a normalized and real-scale depth, but real-scale depth is required for real-scale 3D scene flow estimation.

Flyingthings3D: Similarly, we train another version on Flyingthings3D from scratch and use it to train the scene flow decoder on Flyingthings3D. Since Flyingthings3D is not a typical dataset for depth training, very few previous works reported results. Because scene flow estimation is related to depth, we also evaluated two strongest baselines on Flyingthings3D as shown in Table 2. Then we use the trained depth module to train the scene flow decoder (Tables 3 and 5).

Table 2. Monocular depth results comparison on Flyingthings3D dataset. In the column *Train on*, K denotes KITTI, F denotes Flyingthings3D. In the column *Scale*, ✓ denotes in real scale, ✗ denotes with scale ambiguity.

Method	Train on	Scale	*Higher is better*			*Lower is better*			
			$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$	Abs Rel	Sq Rel	RMSE	RMSE log
Mono-SF [18]	K	✗	0.259	0.483	0.648	0.943	19.250	14.676	0.667
Mono-SF-Multi [19]	K	✗	0.273	0.492	0.650	0.931	19.072	14.566	0.666
Ours	F	✓	**0.715**	**0.934**	**0.980**	**0.188**	**1.142**	**4.400**	**0.235**

Table 3. Monocular 3D scene flow results comparison on Flyingthings3D dataset.(image based evaluation standard) In the column *Train on*, K denotes KITTI, F denotes Flyingthings3D. In the column *Scale*, ✓ denotes in real scale, ✗ denotes with scale ambiguity.

Method	Train on	Scale	EPE3D(m)	ACC3DS	ACC3DR	Outlier3D	EPE2D(px)	ACC2D
Mono-SF [18]	K	✗	1.1288	0.0525	0.1017	0.9988	58.2761	0.2362
Mono-SF-Multi [19]	K	✗	1.5864	0.0020	0.0050	0.9988	48.3099	0.3162
Ours	F	✓	**0.3915**	**0.5424**	**0.6911**	**0.8279**	**22.4226**	**0.6659**

Table 4. Monocular 3D scene flow results comparison on KITTI flow 2015 dataset.(image based evaluation standard) In the column *Train on*, K denotes KITTI, F denotes Flyingthings3D. In the column *Scale*, ✓ denotes in real scale, ✗ denotes with scale ambiguity.

Method	Train on	Scale	EPE3D(m)	ACC3DS	ACC3DR	Outlier3D	EPE2D(px)	ACC2D
[18]depth + Mono-Exp [48]	K	✗	2.7079	0.0676	0.1467	0.9982	181.0699	0.2777
Our depth + Mono-Exp [48]	K	✗	1.6673	0.0838	0.1815	0.9953	78.7245	0.2837
Mono-SF [18]	K	✗	1.1288	0.0525	0.1017	0.9988	58.2761	0.2362
Mono-SF-Multi [19]	K	✗	0.7828	0.1725	0.2548	0.9477	35.9015	0.4886
Ours	F	✓	**0.6970**	**0.2453**	**0.3692**	**0.8630**	**33.4750**	**0.4968**

Table 5. 3D scene flow results comparison with different input data form on KITTI flow 2015 and Flyingthings3D. (LiDAR based evaluation standard) Since we also compare the LiDAR approaches here, we use the strict LiDAR-based evaluation standard. In the column *Dataset*, K denotes KITTI, F denotes Flyingthings3D. All works in the table are in real scale. Since our work is the first image-based work thoroughly evaluating 3D scene flow with 3D metrics, we lack of some 3D results from previous image-based works, but it is already enough to see our Monocular-image based work is comparable to LiDAR approaches

Dataset	Method	Input	EPE3D(m)	ACC3DS	ACC3DR	Outlier3D	EPE2D(px)	ACC2D
K	LDOF [5]	RGBD	0.498	-	-	-	-	-
	OSF [32]	RGBD	0.394	-	-	-	-	-
	PRSM [43]	RGBD	0.327	-	-	-	-	-
	PRSM [43]	Stereo	0.729	-	-	-	-	-
	Ours	Mono	0.6970	0.0035	0.0255	0.9907	33.4750	0.0330
	ICP(rigid) [4]	LiDAR	0.5185	0.0669	0.1667	0.8712	27.6752	0.1056
	FGR(rigid) [53]	LiDAR	0.4835	0.1331	0.2851	0.7761	18.7464	0.2876
	CPD(non-rigid) [34]	LiDAR	0.4144	0.2058	0.4001	0.7146	27.0583	0.1980
F	FlowNet-C [9]	Depth	0.7887	0.0020	0.0149	-	-	-
	FlowNet-C [9]	RGBD	0.7836	0.0025	0.0174	-	-	-
	Ours	Mono	0.3915	0.0125	0.0816	0.9874	22.4226	0.0724
	ICP(global) [4]	LiDAR	0.5019	0.0762	0.2198	-	-	-
	ICP(rigid) [4]	LiDAR	0.4062	0.1614	0.3038	0.8796	23.2280	0.2913
	FlowNet3D-EM [27]	LiDAR	0.5807	0.0264	0.1221	-	-	-
	FlowNet3D-LM [27]	LiDAR	0.7876	0.0027	0.0183	-	-	-
	FlowNet3D [27]	LiDAR	0.1136	0.4125	0.7706	0.6016	5.9740	0.5692

4.2 Real-Scale 3D Scene Flow

Most image-based scene flow works are trained on KITTI in a self-supervised manner, which leads to the scale ambiguity. We train our scene flow decoder in a fully-supervised manner with the EPE3D and forward-backward loss proposed in Sect. 3.4, which is able to estimate real-scale depth and scene flow. While real dataset like KITTI lacks 3D scene flow label, we only train our depth decoder on synthetic dataset Flyingthings3D.

Previous image-based scene flow works mostly use $d1, d2, f1, sf1$ for evaluation, but these metrics are designed for evaluating 2D optical flow or normalized

scene flow, which themselves have the scale ambiguity. Instead, we use the metrics directly for evaluating real-scale 3D scene flow [15,27,46], which we refer as **LiDAR-based evaluation standard** (details in supplementary materials). However, since this LiDAR standard is too strict to image-based approaches, we slightly relax the LiDAR standard and use an **image-based evaluation standard**, EPE (end point error) 3D/2D are same to LiDAR standard:

- Acc3DS: the percentage of points with EPE3D < 0.3m or relative error < 0.1.
- Acc3DR: the percentage of points with EPE3D < 0.4m or relative error < 0.2.
- Outliers3D: the percentage of points with EPE3D > 0.5m or relative error > 0.3.
- Acc2D: the percentage of points whose EPE2D < 20px or relative error < 0.2.

Since we are the only monocular image-based approach estimating in real scale, to evaluate other works with our monocular approach, we need to recover other works to the real scale using the depth and scene flow ground truth (details in supplementary materials).

Flyingthings3D: Table 4 shows the monocular 3D scene flow comparison on Flyingthings3D. Even recovering [18,19] to real scale with ground truth, our result still outperforms the two strongest state-of-the-art baseline works by an overwhelming advantage. More importantly, we only need two consecutive images without any ground truth.

KITTI: We also directly evaluate scene flow on KITTI without any fine-tuning as shown in Table 4. The table also includes state-of-the-art 2D approach Mono-Expansion [48]. We first recover its direct output 2D optical flow to 3D scene flow, and then recover the scale. To recover 2D to 3D, Mono-Expansion proposed a strategy using LiDAR ground truth to expand 2D to 3D specifically for its own usage, but it is not able to extend to all works. For comparison, we recover 3D flow and scale in the same way with ground truth. In the table, the proposed approach still outperforms all state-of-the-art strong baseline works without any fine-tuning on KITTI. Since Mono-Expansion does not estimate depth, we use our depth and Mono-SF depth to help recovering 3D scene flow. Note that our scene flow decoder does not use the depth directly, but share the features and regress in parallel. In the table, by using our depth to recover Mono-Exp, it greatly outperforms by using depth from Mono-SF [18]. This comparison also shows the superiority of our depth estimation over others.

Ablation Study: We perform the ablation study for our scene flow decoder. The ablation study verifies the 2D-3D features alignment process, discussed in Sect. 3.3. As our MonoPLFlowNet architecture (Fig. 2) shows, we perform three-level 2D-3D features alignment in our full model and decode in parallel, which are level 1, 2, 4. In the ablation study, we train the model only with level1 and level1+level2, and compare to the full model trained to the same epoch. The results indicate that the performances get better with deeper levels involved, hence the concatenation of features from different levels in the lattice boost the training, which proves our 2D-3D features alignment mechanism (Table 6).

Table 6. Ablation study on our MonoPLFlowNet by changing level of the scene decoder. (image based evaluation standard) lev1 denotes only using the last level, lev1-lev2 denotes using the last two levels, full denotes using all levels. For fair comparison, we show all results after training epoch 22.

Method	EPE3D(m)	Scale	ACC3DS	ACC3DR	Outlier3D	EPE2D(px)	ACC2D
MonoPLFlowNet-lev1	0.4781	✓	0.4587	0.6146	0.8935	26.3133	0.6092
MonoPLFlowNet-lev1-lev2	0.4439	✓	0.4689	0.6333	0.8605	24.3198	0.6366
MonoPLFlowNet-full	**0.4248**	✓	**0.5099**	**0.6611**	**0.8595**	**23.7657**	**0.6456**

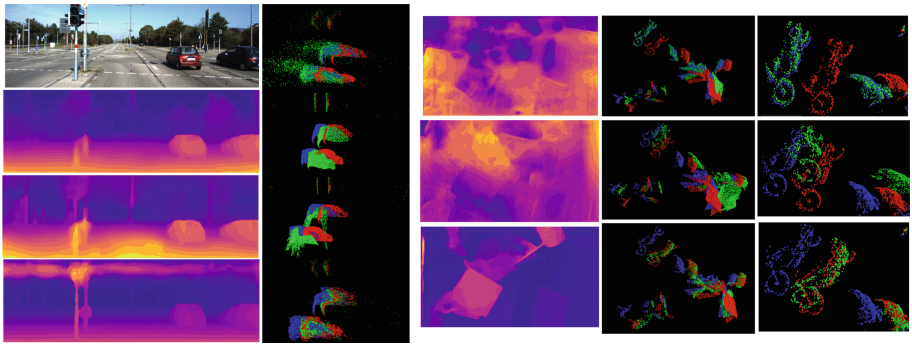

Fig. 5. Qualitative depth and real-scale 3D scene flow results of the proposed MonoPLFlowNet on KITTI and Flyingthins3D for a single pair of two consecutive frames. For KITTI (column 1&2), 1st row: 1st frame of the RGB input image, recovered scene flow of [48] by our depth. From 2nd to 4th row: depth and scene flow by Mono-sf [18], Mono-sf-multi [19] and ours. For Flyingthings3D (column3&4&5), from top to down: depth of the 1st frame, scene flow, zoom-in view scene flow by [18,19] and ours, original input RGB is shown in supplementary materials. Depth and scene flow of [18,19] are recovered to the real scale before generating point cloud.

4.3 Visual Results

We show our visual results of depth and real-scale scene flow in Fig. 5. 3D scene flow are visualized with the pseudo point cloud generating from the estimated depth map, where blue points are from 1st frame, red and green points are blue points translated to 2nd frame by ground truth and estimated 3D scene flow, respectively. The goal of the algorithm is to match the green points to the red points. Different to LiDAR-based works that have same shape of point cloud, the shapes of point cloud are different here because generating from different depth estimation. More visual results are provided in supplementary materials.

5 Conclusion

We present MonoPLFlowNet in this paper. It is the first deep learning architecture that can estimate both depth and 3D scene flow in real scale, using only

two consecutive monocular images. Our depth and 3D scene flow estimation outperforms all the-state-of-art baseline monocular based works, and is comparable to LiDAR based works. In the future, we will explore the usage of more real datasets with specifically designed self-supervised loss to further improve the performance.

References

1. Adams, A., Baek, J., Davis, M.A.: Fast high-dimensional filtering using the permutohedral lattice. Comput. Graph. Forum **29** (2010)
2. Behl, A., Jafari, O.H., Mustikovela, S.K., Alhaija, H.A., Rother, C., Geiger, A.: Bounding boxes, segmentations and object coordinates: how important is recognition for 3d scene flow estimation in autonomous driving scenarios? In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2593–2602 (2017). https://doi.org/10.1109/ICCV.2017.281
3. Behl, A., Paschalidou, D., Donné, S., Geiger, A.: PointFlowNet: learning representations for rigid motion estimation from point clouds. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7954–7963 (2019). https://doi.org/10.1109/CVPR.2019.00815
4. Besl, P., McKay, N.D.: A method for registration of 3-D shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14**(2), 239–256 (1992). https://doi.org/10.1109/34.121791
5. Brox, T., Malik, J.: Large displacement optical flow: Descriptor matching in variational motion estimation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(3), 500–513 (2011). https://doi.org/10.1109/TPAMI.2010.143
6. Charles, R.Q., Su, H., Kaichun, M., Guibas, L.J.: PointNet: deep learning on point sets for 3D classification and segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 77–85 (2017). https://doi.org/10.1109/CVPR.2017.16
7. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. **40**(4), 834–848 (2018). https://doi.org/10.1109/TPAMI.2017.2699184
8. Chen, Y., Schmid, C., Sminchisescu, C.: Self-supervised learning with geometric constraints in monocular video: connecting flow, depth, and camera. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 7062–7071 (2019). https://doi.org/10.1109/ICCV.2019.00716
9. Dosovitskiy, A., et al.: FlowNet: learning optical flow with convolutional networks. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 2758–2766 (2015). https://doi.org/10.1109/ICCV.2015.316
10. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 27. Curran Associates, Inc. (2014). https://proceedings.neurips.cc/paper/2014/file/7bccfde7714a1ebadf06c5f4cea752c1-Paper.pdf
11. Fu, H., Gong, M., Wang, C., Batmanghelich, K., Tao, D.: Deep ordinal regression network for monocular depth estimation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2002–2011 (2018). https://doi.org/10.1109/CVPR.2018.00214

12. Gan, Y., Xu, X., Sun, W., Lin, L.: Monocular depth estimation with affinity, vertical pooling, and label enhancement. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11207, pp. 232–247. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01219-9_14
13. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
14. Godard, C., Aodha, O.M., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6602–6611 (2017). https://doi.org/10.1109/CVPR.2017.699
15. Gu, X., Wang, Y., Wu, C., Lee, Y.J., Wang, P.: HPLFlowNet: hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3249–3258 (2019). https://doi.org/10.1109/CVPR.2019.00337
16. Hornácek, M., Fitzgibbon, A., Rother, C.: SphereFlow: 6 DoF scene flow from RGB-D pairs. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3526–3533 (2014). https://doi.org/10.1109/CVPR.2014.451
17. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2261–2269 (2017). https://doi.org/10.1109/CVPR.2017.243
18. Hur, J., Roth, S.: Self-supervised monocular scene flow estimation. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7394–7403 (2020). https://doi.org/10.1109/CVPR42600.2020.00742
19. Hur, J., Roth, S.: Self-supervised multi-frame monocular scene flow. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2684–2694, June 2021
20. Jaimez, M., Souiai, M., Stückler, J., Gonzalez-Jimenez, J., Cremers, D.: Motion cooperation: smooth piece-wise rigid scene flow from RGB-D images. In: 2015 International Conference on 3D Vision, pp. 64–72 (2015). https://doi.org/10.1109/3DV.2015.15
21. Jampani, V., Kiefel, M., Gehler, P.V.: Learning sparse high dimensional filters: image filtering, dense CRFs and bilateral neural networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4452–4461 (2016). https://doi.org/10.1109/CVPR.2016.482
22. Jiang, H., Sun, D., Jampani, V., Lv, Z., Learned-Miller, E., Kautz, J.: Sense: a shared encoder network for scene-flow estimation. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 3194–3203 (2019). https://doi.org/10.1109/ICCV.2019.00329
23. Kuznietsov, Y., Stückler, J., Leibe, B.: Semi-supervised deep learning for monocular depth map prediction. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2215–2223 (2017). https://doi.org/10.1109/CVPR.2017.238
24. Lee, J.H., Han, M.K., Ko, D.W., Suh, I.H.: From big to small: multi-scale local planar guidance for monocular depth estimation. arXiv preprint arXiv:1907.10326 (2019)
25. Li, R., Nguyen, T.: SM3D: simultaneous monocular mapping and 3D detection. In: 2021 IEEE International Conference on Image Processing (ICIP), pp. 3652–3656 (2021). https://doi.org/10.1109/ICIP42928.2021.9506302

26. Liu, F., Shen, C., Lin, G., Reid, I.: Learning depth from single monocular images using deep convolutional neural fields. IEEE Trans. Pattern Anal. Mach. Intell. **38**(10), 2024–2039 (2016). https://doi.org/10.1109/TPAMI.2015.2505283
27. Liu, X., Qi, C.R., Guibas, L.J.: FlowNet3D: learning scene flow in 3D point clouds. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 529–537 (2019). https://doi.org/10.1109/CVPR.2019.00062
28. Luo, C., et al.: Every pixel counts ++: joint learning of geometry and motion with 3d holistic understanding. IEEE Trans. Pattern Anal. Mach. Intell. **42**(10), 2624–2641 (2020). https://doi.org/10.1109/TPAMI.2019.2930258
29. Lv, Z., Kim, K., Troccoli, A., Sun, D., Rehg, J.M., Kautz, J.: Learning rigidity in dynamic scenes with a moving camera for 3D motion field estimation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11209, pp. 484–501. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01228-1_29
30. Ma, W.C., Wang, S., Hu, R., Xiong, Y., Urtasun, R.: Deep rigid instance scene flow. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3609–3617 (2019). https://doi.org/10.1109/CVPR.2019.00373
31. Mayer, N., et al.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4040–4048 (2016). https://doi.org/10.1109/CVPR.2016.438
32. Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3061–3070 (2015). https://doi.org/10.1109/CVPR.2015.7298925
33. Mittal, H., Okorn, B., Held, D.: Just go with the flow: self-supervised scene flow estimation. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11174–11182 (2020). https://doi.org/10.1109/CVPR42600.2020.01119
34. Myronenko, A., Song, X.: Point set registration: coherent point drift. IEEE Trans. Pattern Anal. Mach. Intell. **32**(12), 2262–2275 (2010). https://doi.org/10.1109/TPAMI.2010.46
35. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: deep hierarchical feature learning on point sets in a metric space. In: NIPS (2017)
36. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: mixing datasets for zero-shot cross-dataset transfer. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) (2020)
37. Ranjan, A., et al.: Competitive collaboration: joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12232–12241 (2019). https://doi.org/10.1109/CVPR.2019.01252
38. Saxena, A., Sun, M., Ng, A.Y.: MAKE3D: learning 3D scene structure from a single still image. IEEE Trans. Pattern Anal. Mach. Intell. **31**(5), 824–840 (2009). https://doi.org/10.1109/TPAMI.2008.132
39. Schuster, R., Wasenmuller, O., Kuschk, G., Bailer, C., Stricker, D.: SceneFlowFields: dense interpolation of sparse scene flow correspondences. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1056–1065 (2018). https://doi.org/10.1109/WACV.2018.00121
40. Su, H., et al.: SplatNet: sparse lattice networks for point cloud processing. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2530–2539 (2018). https://doi.org/10.1109/CVPR.2018.00268

41. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8934–8943 (2018). https://doi.org/10.1109/CVPR.2018.00931
42. Taniai, T., Sinha, S.N., Sato, Y.: Fast multi-frame stereo scene flow with motion segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6891–6900 (2017). https://doi.org/10.1109/CVPR.2017.729
43. Vogel, C., Schindler, K., Roth, S.: 3D scene flow estimation with a piecewise rigid scene model. Int. J. Comput. Vision **115**(1), 1–28 (2015)
44. Wang, Z., Li, S., Howard-Jenkins, H., Prisacariu, V.A., Chen, M.: Flownet3d++: geometric losses for deep scene flow estimation. In: 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 91–98 (2020). https://doi.org/10.1109/WACV45572.2020.9093302
45. Wannenwetsch, A.S., Kiefel, M., Gehler, P.V., Roth, S.: Learning task-specific generalized convolutions in the permutohedral lattice. In: Fink, G.A., Frintrop, S., Jiang, X. (eds.) DAGM GCPR 2019. LNCS, vol. 11824, pp. 345–359. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33676-9_24
46. Wu, W., Wang, Z.Y., Li, Z., Liu, W., Fuxin, L.: PointPWC-net: cost volume on point clouds for (self-)supervised scene flow estimation. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12350, pp. 88–107. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58558-7_6
47. Xu, Y., Wang, Y., Guo, L.: Unsupervised ego-motion and dense depth estimation with monocular video. In: 2018 IEEE 18th International Conference on Communication Technology (ICCT), pp. 1306–1310 (2018). https://doi.org/10.1109/ICCT.2018.8600039
48. Yang, G., Ramanan, D.: Upgrading optical flow to 3D scene flow through optical expansion. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1331–1340 (2020). https://doi.org/10.1109/CVPR42600.2020.00141
49. Yang, M., Yu, K., Zhang, C., Li, Z., Yang, K.: DenseASPP for semantic segmentation in street scenes. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3684–3692 (2018). https://doi.org/10.1109/CVPR.2018.00388
50. Yang, Z., Wang, P., Wang, Y., Xu, W., Nevatia, R.: Every pixel counts: unsupervised geometry learning with holistic 3D motion understanding (2018)
51. Yin, W., Liu, Y., Shen, C., Yan, Y.: Enforcing geometric constraints of virtual normal for depth prediction. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 5683–5692 (2019). https://doi.org/10.1109/ICCV.2019.00578
52. Yin, Z., Shi, J.: GeoNet: unsupervised learning of dense depth, optical flow and camera pose. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1983–1992 (2018). https://doi.org/10.1109/CVPR.2018.00212
53. Zhou, Q.-Y., Park, J., Koltun, V.: Fast global registration. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 766–782. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_47
54. Zou, Y., Luo, Z., Huang, J.-B.: DF-net: unsupervised joint learning of depth and flow using cross-task consistency. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11209, pp. 38–55. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01228-1_3