# PointFlowNet: Learning Representations for Rigid Motion Estimation from Point Clouds

Aseem Behl        Despoina Paschalidou        Simon Donné        Andreas Geiger

Autonomous Vision Group, MPI for Intelligent Systems and University of Tübingen

{aseem.behl,despoina.paschalidou,simon.donne,andreas.geiger}@tue.mpg.de

## Abstract

*Despite significant progress in image-based 3D scene flow estimation, the performance of such approaches has not yet reached the fidelity required by many applications. Simultaneously, these applications are often not restricted to image-based estimation: laser scanners provide a popular alternative to traditional cameras, for example in the context of self-driving cars, as they directly yield a 3D point cloud. In this paper, we propose to estimate 3D motion from such unstructured point clouds using a deep neural network. In a single forward pass, our model jointly predicts 3D scene flow as well as the 3D bounding box and rigid body motion of objects in the scene. While the prospect of estimating 3D scene flow from unstructured point clouds is promising, it is also a challenging task. We show that the traditional global representation of rigid body motion prohibits inference by CNNs, and propose a translation equivariant representation to circumvent this problem. For training our deep network, a large dataset is required. Because of this, we augment real scans from KITTI with virtual objects, realistically modeling occlusions and simulating sensor noise. A thorough comparison with classic and learning-based techniques highlights the robustness of the proposed approach.*

## 1. Introduction

For intelligent systems such as self-driving cars, the precise understanding of their surroundings is key. Notably, in order to make predictions and decisions about the future, tasks like navigation and planning require knowledge about the 3D geometry of the environment as well as about the 3D motion of other agents in the scene.

3D scene flow is the most generic representation of this 3D motion; it associates a velocity vector with 3D motion to each measured point. Traditionally, 3D scene flow is estimated based on two consecutive image pairs of a calibrated stereo rig [17, 39, 40]. While the accuracy of scene flow methods has greatly improved over the last decade [24], image-based scene flow methods have rarely made it into robotics applications. The reasons for this are two-fold. First

of all, most leading techniques take several minutes or hours to predict 3D scene flow. Secondly, stereo-based scene flow methods suffer from a fundamental flaw, the "curse of two-view geometry": it can be shown that the depth error grows quadratically with the distance to the observer [20]. This causes problems for the baselines and object depths often found in self-driving cars, as illustrated in Fig. 1 (top).

Consequently, most modern self-driving car platforms rely on LIDAR technology for 3D geometry perception. In contrast to cameras, laser scanners provide a 360 degree field of view with just one sensor, are generally unaffected by lighting conditions, and do not suffer from the quadratic error behavior of stereo cameras. However, while LIDAR provides accurate 3D point cloud measurements, estimating the motion between two such scans is a non-trivial task. Because of the sparse and non-uniform nature of the point clouds, as well as the missing appearance information, the data association problem is complicated. Moreover, characteristic patterns produced by the scanner, such as the circular rings in Fig. 1 (bottom), move with the observer and can easily mislead local correspondence estimation algorithms.

To address these challenges, we propose *PointFlowNet*, a generic model for learning 3D scene flow from pairs of unstructured 3D point clouds. Our main contributions are:

- We present an end-to-end trainable model for joint 3D scene flow and rigid motion prediction and 3D object detection from unstructured LIDAR data, as captured from a (self-driving) car.
- We show that a global representation is not suitable for rigid motion prediction, and propose a local translation-equivariant representation to mitigate this problem.
- We augment the KITTI dataset with virtual cars, taking into account occlusions and simulating sensor noise, to provide more (realistic) training data.
- We demonstrate that our approach compares favorably to the state-of-the-art.

Our code and dataset are available at the project webpage[1].
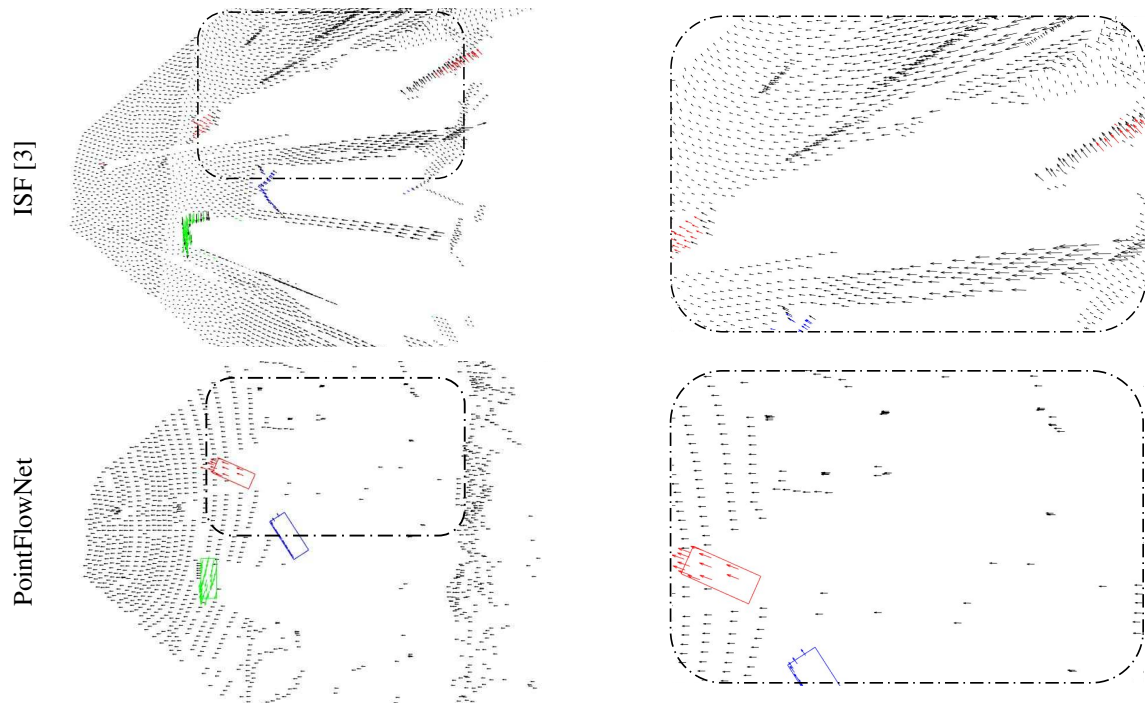
---

[1] https://github.com/aseembehl/pointflownet

Figure 1: **Motivation.** To motivate the use of LIDAR sensors in the context of autonomous driving, we provide a qualitative comparison of the state-of-the-art image-based scene flow method ISF [3] (top) to our LIDAR-based PointFlowNet (bottom) using a scene from the KITTI 2015 dataset [24]. The left column shows the output of the two methods. The right column shows a zoomed-in version of the inlet. While the image-based result suffers from the "curse of two-view geometry" (with noisy geometry, and non-uniform background movement), our LIDAR-based approach is also accurate in distant regions. Moreover, ISF relies on instance segmentation in the image space for detecting objects: depth estimation errors at the boundaries lead to objects being split into two 3D clusters (e.g., the red car). For clarity, we visualize only a subset of the points.

## 2. Related Work

In the following discussion, we first group related methods based on their expected input; we finish this section with a discussion of learning-based solutions.

**Scene Flow from Image Sequences:** The most common approach to 3D scene flow estimation is to recover correspondences between two calibrated stereo image pairs. Early approaches solve the problem using coarse-to-fine variational optimization [2, 17, 37, 39–41, 44]. As coarse-to-fine optimization often performs poorly in the presence of large displacements, slanted-plane models which decompose the scene into a collection of rigidly moving 3D patches have been proposed [22, 24, 25, 42]. The benefit of incorporating semantics has been demonstrated in [3]. While the state-of-the-art in image-based scene flow estimation has advanced significantly, its accuracy is inherently limited by the geometric properties of two-view geometry as previously mentioned and illustrated in Figure 1.

**Scene Flow from RGB-D Sequences:** When per-pixel depth information is available, two consecutive RGB-D frames are sufficient for estimating 3D scene flow. Initially,

the image-based variational scene flow approach was extended to RGB-D inputs [15, 30, 45]. Franke et al. [11] instead proposed to track KLT feature correspondences using a set of Kalman filters. Exploiting PatchMatch optimization on spherical 3D patches, Hornacek et al. [16] recover a dense field of 3D rigid body motions. However, while structured light scanning techniques (e.g., Kinect) are able to capture indoor environments, dense RGB-D sequences are hard to acquire in outdoor scenarios like ours. Furthermore, structured light sensors suffer from the same depth error characteristics as stereo techniques.

**Scene Flow from 3D Point Clouds:** In the robotics community, motion estimation from 3D point clouds has so far been addressed primarily with classical techniques. Several works [6, 34, 36] extend occupancy maps to dynamic scenes by representing moving objects via particles which are updated using particle filters [6, 34] or EM [36]. Others tackle the problem as 3D detection and tracking using mean shift [1], RANSAC [7], ICP [26], CRFs [38] or Bayesian networks [14]. In contrast, Dewan et al. [8] propose a 3D scene flow approach where local SHOT descriptors [35] are associated via a CRF that incorporates local smoothness and

rigidity assumptions. While impressive results have been achieved, all the aforementioned approaches require significant engineering and manual model specification. In addition, local shape representations such as SHOT [35] often fail in the presence of noisy or ambiguous inputs. In contrast, we address the scene flow problem using a generic end-to-end trainable model which is able to learn local and global statistical relationships directly from data. Accordingly, our experiments show that our model compares favorably to the aforementioned classical approaches.

**Learning-based Solutions:** While several learning-based approaches for stereo [19, 21, 46] and optical flow [9, 18, 33] have been proposed in literature, there is little prior work on learning scene flow estimation. A notable exception is SceneFlowNet [23], which concatenates features from FlowNet [9] and DispNet [23] for *image-based* scene flow estimation. In contrast, this paper proposes a novel end-to-end trainable approach for scene flow estimation from unstructured *3D point clouds*. More recently, Wang et al. [43] proposed a novel continuous convolution operation and applied it to 3D segmentation and scene flow. However, they do not consider rigid motion estimation which is the main focus of this work.

## 3. Method

We start by formally defining our problem. Let $\mathbf{P}_t \in \mathbb{R}^{N \times 3}$ and $\mathbf{P}_{t+1} \in \mathbb{R}^{M \times 3}$ denote the input 3D point clouds at frames $t$ and $t + 1$, respectively. Our goal is to estimate

- the 3D scene flow $\mathbf{v}_i \in \mathbb{R}^3$ and the 3D rigid motion $\mathbf{R}_i \in \mathbb{R}^{3 \times 3}$, $\mathbf{t}_i \in \mathbb{R}^3$ at each of the $N$ points in the reference point cloud at frame $t$, and

- the location, orientation, size and rigid motion of every moving object in the scene (in our experiments, we focus solely on cars).

The overall network architecture of our approach is illustrated in Figure 2. The network comprises five main components: (1) feature encoding layers, (2) context encoding layers (3) scene flow estimation, ego-motion estimation and 3D object detection layers, (4) rigid motion estimation layers and (5) object motion decoder. In the following, we provide a detailed description for each of these components as well as the loss functions.

### 3.1. Feature Encoder

The feature encoding layers take a raw point cloud as input, partition the space into voxels, and describe each voxel with a feature vector. The simplest form of aggregation is binarization, where any voxel containing at least one point is set to 1 and all others are zero. However, better results can be achieved by aggregating high-order statistics over the

voxel [5, 27–29, 32, 48]. In this paper, we leverage the feature encoding recently proposed by Zhou et al. [48], which has demonstrated state-of-the-art results for 3D object detection from point clouds.

We briefly summarize this encoding, but refer the reader to [48] for more details. We subdivide the 3D space of each input point cloud into equally spaced voxels and group points according to the voxel they reside in. To reduce bias with respect to LIDAR point density, a fixed number of T points is randomly sampled for all voxels containing more than T points. Each voxel is processed with a stack of Voxel Feature Encoding (VFE) layers to capture local and global geometric properties of its contained points. As more than $90\%$ of the voxels in LIDAR scans tend to be empty, we only process non-empty voxels and store the results in a sparse 4D tensor.

We remark that alternative representations, e.g., those that directly encode the raw point cloud [13, 43], could be a viable alternative to voxel representations. However, as the representation is not the main focus of this paper, we will leave such an investigation to future work.

### 3.2. Context Encoder

As objects in a street scene are restricted to the ground plane, we only estimate objects and motions on this plane: we assume that 3D objects cannot be located on top of each other and that 3D scene points directly above each other undergo the same 3D motion. This is a valid assumption for our autonomous driving scenario, and greatly improves memory efficiency. Following [48], the first part of the context encoder vertically downsamples the voxel feature map by using three 3D convolutions with vertical stride 2. The resulting 3D feature map is reshaped by stacking the remaining height slices as feature maps to yield a 2D feature map. The resulting 2D feature map is provided to three blocks of 2D convolutional layers. The first layer of each block downsamples the feature map via a convolution with stride 2, followed by a series of convolution layers with stride 1.

### 3.3. 3D Detection, Ego-motion and 3D Scene Flow

Next, the network splits up in three branches for respectively ego-motion estimation, 3D object detection and 3D scene flow estimation. As there is only one observer, the ego-motion branch further downsamples the feature map by interleaving convolutional layers with strided convolutional layers and finally using a fully connected layer to regress a 3D ego-motion (movement in the ground-plane and rotation around the vertical). For the other two tasks, we upsample the output of the various blocks using up-convolutions: to half the original resolution for 3D object detection, and to the full resolution for 3D scene flow estimation. The resulting features are stacked and mapped to the training targets with one 2D convolutional layer each. We regress a 3D vector per
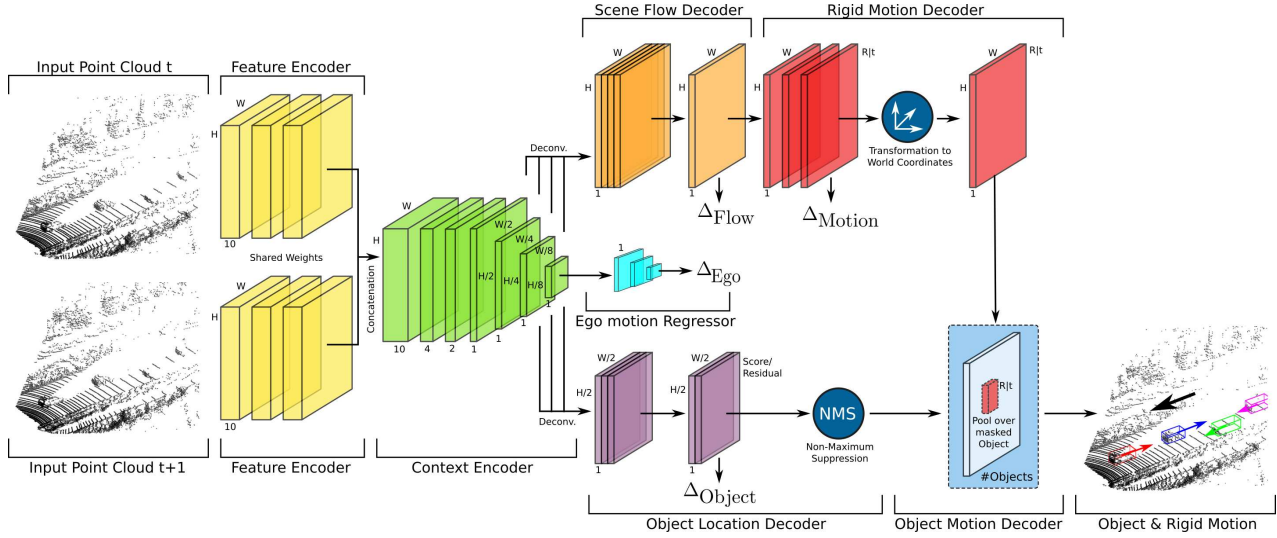
Figure 2: **Network Architecture.** The **feature encoder** takes a raw LIDAR point cloud as input, groups the points into $W \times H \times 10$ voxels, and outputs 128D feature maps (for clarity, the size of the feature maps is not shown in the figure) which are concatenated and passed to the **context encoder**. The context encoder learns a global representation by interleaving convolution with strided convolution layers and "flattening" the third dimension (height above ground), i.e., we assume that 3D objects cannot be located on top of each other and that 3D scene points that project to the same location in the ground plane undergo the same 3D motion. Feature maps at different resolutions are upsampled, stacked and fed into the decoding branches. 3D scene flow is computed for every input voxel in the **scene flow decoder** and the result is passed to the **rigid motion decoder**, which infers a rigid body transformation for every point. In parallel, the **ego-motion regressor**, further downsamples the feature map by interleaving convolutional layers with strided convolutional layers and a fully connected layer at the end to regress rigid motion for the ego vehicle. In addition, the **object decoder** predicts the location and size (i.e., 3D bounding box) of objects in the scene. Finally, the **object motion decoder** takes the point-wise rigid body motions as input and predicts the object rigid motions by pooling the rigid motion field over the detected 3D objects.

voxel for the scene flow, and follow [48] for the object detections: regressing likelihoods for a set of proposal bounding boxes and regressing the residuals (translation, rotation and size) between the positive proposal boxes and corresponding ground truth boxes. A proposal bounding box is called positive if it has the highest Intersection over Union (IoU, in the ground plane) with a ground truth detection, or if its IoU with any ground truth box is larger than 0.6, as in [48].

### 3.4. Rigid Motion Decoder

We now wish to infer per-pixel and per-object rigid body motions from the previously estimated 3D scene flow. For a single point in isolation, there are infinitely many rigid body motions that explain a given 3D scene flow: this ambiguity can be resolved by considering the local neighborhood.

It is unfortunately impossible to use a convolutional neural network to regress rigid body motions that are represented in global world coordinates, as the conversion between scene flow and global rigid body motion depends on the location in the scene: while convolutional layers are translation equivariant, the mapping to be learned is not. Identical regions of flow lead to different global rigid body motions, depending on the location in the volume, and a fully convolutional net-

work cannot model this. In the following, we first prove that the rigid motion in the world coordinate system is not translation equivariant. Subsequently, we introduce our proposed rigid motion representation in local coordinates and show it to be translation equivariant and therefore amenable to fully convolutional inference.

Let us assume a point $\mathbf{p}$ in world coordinate system $W$ and let $A$ denote a local coordinate system with origin $\mathbf{o}_A$ as illustrated in Fig. 3a. A scene flow vector $\mathbf{v}$ is explained by rigid body motion $(\mathbf{R}_A, \mathbf{t}_A)$, represented in local coordinate system $A$ with origin $\mathbf{o}_A$, if and only if:

$$\mathbf{v} = [\mathbf{R}_A (\mathbf{p} - \mathbf{o}_A) + \mathbf{t}_A)] - (\mathbf{p} - \mathbf{o}_A) \qquad (1)$$

Now assume a second world location $\mathbf{q}$, also with scene flow $\mathbf{v}$ as in Fig. 3a. Let $B$ denote a second local coordinate system with origin $\mathbf{o}_B$ such that $\mathbf{p}$ and $\mathbf{q}$ have the same local coordinates in their respective coordinate system, i.e., $\mathbf{p} - \mathbf{o}_A = \mathbf{q} - \mathbf{o}_B$. We now prove the following two claims:

1. There exists no rigid body motion $\mathbf{R}_W, \mathbf{t}_W$ represented in world coordinate system $W$ that explains the scene flow $\mathbf{v}$ for both $\mathbf{p}$ and $\mathbf{q}$, unless $\mathbf{R}_W = \mathbf{I}$.

2. Any rigid body motion $(\mathbf{R}_A, \mathbf{t}_A)$ explaining scene flow $\mathbf{v}$ for $\mathbf{p}$ in system $A$ also does so for $\mathbf{q}$ in system $B$.
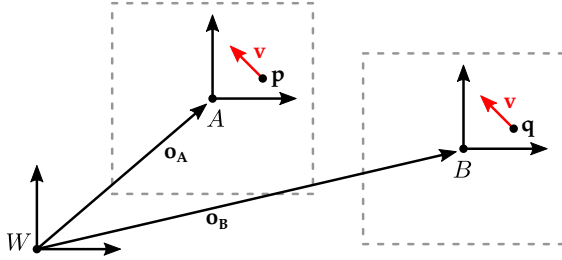
(a) Local (A,B) and World Coordinate System (W)
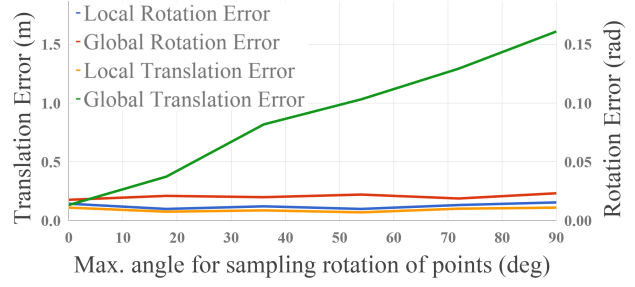


(b) Quantitative Comparison

Figure 3: **Rigid Motion Estimation.** In (a), indices $A$ and $B$ denote the coordinate system of points $\mathbf{p}$ and $\mathbf{q}$ at origin $\mathbf{o}_A$ and $\mathbf{o}_B$, respectively. The same scene flow $\mathbf{v}$ can locally be explained with the same rigid body motion $(\mathbf{R}_L, \mathbf{t}_L)$, but requires different translations $\mathbf{t}_W^{\mathbf{p}} \neq \mathbf{t}_W^{\mathbf{q}}$ in the global coordinate system. A simple example (b) provides empirical evidence that translation cannot be learned in global coordinates with a CNN. Using global coordinates, the translation error increases significantly with the magnitude of rotation (green). There is no such increase in error when using local coordinates (orange).

Towards this goal, we introduce the notation $(\mathbf{R}_W^{\mathbf{p}}, \mathbf{t}_W^{\mathbf{p}})$ to indicate rigid motion in world coordinates $W$ induced by $\mathbf{v_p}$.

**Claim 1**

$$\forall \mathbf{p}, \mathbf{q} \in \mathbb{R}^3, \mathbf{p} - \mathbf{o}_A = \mathbf{q} - \mathbf{o}_B, \mathbf{o}_A \neq \mathbf{o}_B :$$
$$\mathbf{v_p} = \mathbf{v_q} \implies \mathbf{R}_W^{\mathbf{p}} \neq \mathbf{R}_W^{\mathbf{q}} \quad or$$
$$\mathbf{t}_W^{\mathbf{p}} \neq \mathbf{t}_W^{\mathbf{q}} \quad or \quad (2)$$
$$\mathbf{R}_W^{\mathbf{p}} = \mathbf{R}_W^{\mathbf{q}} = \mathbf{I}$$

**Proof of Claim 1** *From $\mathbf{v_p} = \mathbf{v_q}$ we get*

$$\mathbf{R}_W^{\mathbf{p}} \mathbf{p} + \mathbf{t}_W^{\mathbf{p}} - \mathbf{p} = \mathbf{R}_W^{\mathbf{q}} \mathbf{q} + \mathbf{t}_W^{\mathbf{q}} - \mathbf{q}$$
$$\mathbf{R}_W^{\mathbf{p}} \mathbf{p} + \mathbf{t}_W^{\mathbf{p}} = \mathbf{R}_W^{\mathbf{q}} (\mathbf{p} - \Delta\mathbf{o}) + \mathbf{t}_W^{\mathbf{q}} + \Delta\mathbf{o}$$
$$(\mathbf{R}_W^{\mathbf{p}} - \mathbf{R}_W^{\mathbf{q}}) \mathbf{p} = (\mathbf{I} - \mathbf{R}_W^{\mathbf{q}}) \Delta\mathbf{o} + (\mathbf{t}_W^{\mathbf{q}} - \mathbf{t}_W^{\mathbf{p}})$$

*where $\Delta\mathbf{o} = \mathbf{o}_A - \mathbf{o}_B$. Now, we assume that $\mathbf{R}_W^{\mathbf{p}} = \mathbf{R}_W^{\mathbf{q}}$ and that $\mathbf{t}_W^{\mathbf{p}} = \mathbf{t}_W^{\mathbf{q}}$ (in all other cases the claim is already fulfilled). In this case, we have $\Delta\mathbf{o} = \mathbf{R}_W^{\mathbf{p}} \Delta\mathbf{o}$. However, any rotation matrix representing a non-zero rotation has no real eigenvectors. Hence, as $\mathbf{o}_A \neq \mathbf{o}_B$, this equality can only be fulfilled if $\mathbf{R}_W^{\mathbf{p}}$ is the identity matrix.* ∎

**Claim 2**

$$\forall \mathbf{p}, \mathbf{q} \in \mathbb{R}^3, \mathbf{p} - \mathbf{o}_A = \mathbf{q} - \mathbf{o}_B, \mathbf{o}_A \neq \mathbf{o}_B :$$
$$\mathbf{v} = \mathbf{R}(\mathbf{p} - \mathbf{o}_A) + \mathbf{t} + (\mathbf{p} - \mathbf{o}_A) \quad (3)$$
$$\implies \mathbf{v} = \mathbf{R}(\mathbf{q} - \mathbf{o}_B) + \mathbf{t} + (\mathbf{q} - \mathbf{o}_B)$$

**Proof of Claim 2** *Trivially from $\mathbf{p} - \mathbf{o}_A = \mathbf{q} - \mathbf{o}_B$.* ∎

The first proof shows the non-stationarity of rigid body motions represented in global coordinates, while the second proof shows that the rigid motion represented in local coordinates is stationary and can therefore be learned by a translation equivariant convolutional neural network.

We provide a simple synthetic experiment in Figure 3 to empirically confirm this analysis. Towards this goal, we warp a grid of $10 \times 10$ points by random rigid motions, and then try to infer these rigid motions from the resulting scene flow: as expected, the estimation is only successful using local coordinates. Note that a change of reference system only affects the translation component while the rotation component remains unaffected. Motivated by the preceding analysis, we task our CNN to predict rigid motion in local coordinates, followed by a deterministic layer which transforms local coordinates into global coordinates as follows:

$$\mathbf{R}_L = \mathbf{R}_W \qquad \mathbf{t}_L = (\mathbf{R}_W - \mathbf{I})\mathbf{o}_L + \mathbf{t}_W$$
$$\mathbf{R}_W = \mathbf{R}_L \qquad \mathbf{t}_W = (\mathbf{I} - \mathbf{R}_W)\mathbf{o}_L + \mathbf{t}_L \quad (4)$$

In our case, the origin of the world coordinate system $W$ coincides with the LIDAR scanner and the origin of the local coordinate systems is located at the center of each voxel.

### 3.5. Object Motion Decoder

Finally, we combine the results of 3D object detection and rigid motion estimation into a single rigid motion for each detected object. We first apply non-maximum-suppression (NMS) using detection threshold $\tau$, yielding a set of 3D bounding boxes. To estimate the rigid body motion of each detection, we pool the predicted rigid body motions over the corresponding voxels (i.e., the voxels in the bounding box of the detection) by computing the median translation and rotation. Note that this is only possible as the rigid body motions have been converted back into world coordinates.

### 3.6. Loss Functions

This section describes the loss functions used by our approach. While it seems desirable to define a rigid motion loss directly at object level, this is complicated by the need for differentiation through the non-maximum-suppression step and the difficulty associating to ground truth objects.

Furthermore, balancing the influence of an object loss across voxels is much more complex than applying all loss functions directly at the voxel level. We therefore use auxiliary voxel-level loss functions. Our loss comprises four parts:

$$\mathcal{L} = \alpha\mathcal{L}_{flow} + \beta\mathcal{L}_{rigmo} + \gamma\mathcal{L}_{ego} + \mathcal{L}_{det} \tag{5}$$

Here, $\alpha, \beta, \gamma$ are positive constants for balancing the relative importance of the task specific loss functions. We now describe the task-specific loss functions in more detail.

**Scene Flow Loss:** The scene flow loss is defined as the average $\ell_1$ distance between the predicted scene flow and the true scene flow at every voxel

$$\mathcal{L}_{flow} = \frac{1}{K} \sum_j \left\| \mathbf{v}_j - \mathbf{v}_j^* \right\|_1 \tag{6}$$

where $\mathbf{v}_j \in \mathbb{R}^3$ and $\mathbf{v}_j^* \in \mathbb{R}^3$ denote the regression estimate and ground truth scene flow at voxel $j$, and $K$ is the number of non-empty voxels.

**Rigid Motion Loss:** The rigid motion loss is defined as the average $\ell_1$ error between the predicted translation $\mathbf{t}_j \in \mathbb{R}^2$ and its ground truth $\mathbf{t}_j^* \in \mathbb{R}^2$ in the local coordinate system and the average $\ell_1$ error between the predicted rotation $\theta_j$ around the Z-axis and its ground truth $\theta_j^*$ at every voxel $j$.

$$\mathcal{L}_{rigmo} = \frac{1}{K} \sum_j \left\| \mathbf{t}_j - \mathbf{t}_j^* \right\|_1 + \lambda \left\| \theta_j - \theta_j^* \right\|_1 \tag{7}$$

where $\lambda$ is a positive constant to balance the relative importance of the two terms. The conversion from world coordinates to local coordinates is given by (see also Eq. 4)

$$\mathbf{R}_L = \mathbf{R}_W(\theta_j) \qquad \mathbf{t}_L = (\mathbf{R}_W(\theta_j) - \mathbf{I})\,\mathbf{p}_j + \mathbf{t}_W \tag{8}$$

where $\mathbf{p}_j \in \mathbb{R}^2$ specifies the position of voxel $j$ in the XY-plane in world coordinates and $\mathbf{R}_W(\theta_j)$ is the rotation matrix corresponding to rotation $\theta_j$ around the Z-axis.

**Ego-motion Loss:** Similarly, the ego-motion loss is defined as the $\ell_1$ distance between the predicted background translation $\mathbf{t}_{BG} \in \mathbb{R}^2$ and its ground truth $\mathbf{t}_{BG}^* \in \mathbb{R}^2$ and the predicted rotation $\theta_{BG}$ and its ground truth $\theta_{BG}^*$:

$$\mathcal{L}_{ego} = \|\mathbf{t}_{BG} - \mathbf{t}_{BG}^*\|_1 + \lambda\|\theta_{BG} - \theta_{BG}^*\|_1 \tag{9}$$

**Detection Loss:** Following [48], we define the detection loss as follows:

$$\mathcal{L}_{det} = \frac{1}{M_{pos}} \sum_k \mathcal{L}_{cls}(p_k^{pos}, 1) + \mathcal{L}_{reg}(\mathbf{r}_k, \mathbf{r}_k^*) \\ + \frac{1}{M_{neg}} \sum_l \mathcal{L}_{cls}(p_l^{neg}, 0) \tag{10}$$
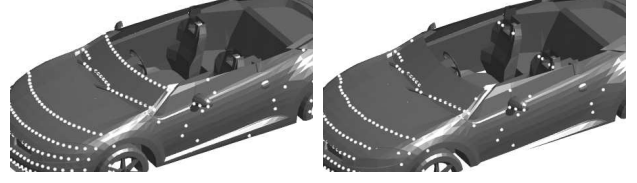


Figure 4: **Augmentation.** Simulating LIDAR measurements based on 3D meshes would result in measurements at transparant surfaces such as windows (**left**), wheres a real LIDAR scanner measures interior points instead. Our simulation replicates the behavior of LIDAR scanners by taking into account model transparency and learning the noise model from real KITTI scans (**right**).

where $p_k^{pos}$ and $p_l^{neg}$ represent the softmax output for positive proposal boxes $a_k^{pos}$ and negative proposal boxes $a_l^{neg}$, respectively. $\mathbf{r}_k \in \mathbb{R}^7$ and $\mathbf{r}_k^* \in \mathbb{R}^7$ denote the regression estimates and ground truth residual vectors (translation, rotation and size) for the positive proposal box $k$, respectively. $M_{pos}$ and $M_{neg}$ represent the number of positive and negative proposal boxes. $\mathcal{L}_{cls}$ denotes the binary cross entropy loss, while $\mathcal{L}_{reg}$ represents the smooth $\ell_1$ distance function. We refer to [48] for further details.

## 4. Experimental Evaluation

We now evaluate the performance of our method on the KITTI object detection dataset [12] as well as an extended version, which we have augmented by simulating virtual objects in each scene.

### 4.1. Datasets

**KITTI:** For evaluating our approach, we use 61 sequences of the training set in the KITTI object detection dataset [12], containing a total of 20k frames. As there is no pointcloud-based scene flow benchmark in KITTI, we perform our experiments on the original training set. Towards this goal, we split the original training set into 70% train, 10% validation, 20% test sequences, making sure that frames from the same sequence are not used in different splits.

**Augmented KITTI:** However, the official KITTI object detection datasets lacks cars with a diverse range of motions. To generate more salient training example, we generate a realistic mixed reality LiDAR dataset exploiting a set of high quality 3D CAD models of cars [10] by taking the characteristics of real LIDAR scans into account.

We discuss our workflow here. We start by fitting the ground plane using RANSAC 3D plane fitting; this allows us to detect obstacles and hence the drivable region. In a second step, we randomly place virtual cars in the drivable region, and simulate a new LIDAR scan that includes these virtual cars. Our simulator uses a noise model learned from the real KITTI scanner by fitting a Gaussian distribution conditioned

| Eval. Dataset | Training Dataset | Scene Flow (m) | | | Object Motion | | Ego-motion | |
|---|---|---|---|---|---|---|---|---|
| | | FG | BG | All | Rot.(rad) | Tr.(m) | Rot.(rad) | Tr.(m) |
| K | K | 0.23 | **0.14** | **0.14** | **0.004** | 0.30 | **0.004** | **0.09** |
| K | K+AK | **0.18** | **0.14** | **0.14** | **0.004** | **0.29** | **0.004** | **0.09** |
| K+AK | K | 0.58 | **0.14** | 0.18 | **0.010** | 0.57 | **0.004** | 0.14 |
| K+AK | K+AK | **0.28** | **0.14** | **0.16** | **0.011** | **0.48** | **0.004** | **0.12** |

Table 1: **Ablation Study** on our KITTI and Augmented KITTI validation datasets, abbreviated with K and AK, respectively.

| Eval. Dataset | Method | Scene Flow (m) | | | Object Motion | | Ego-motion | |
|---|---|---|---|---|---|---|---|---|
| | | FG | BG | All | Rot.(rad) | Tr.(m) | Rot.(rad) | Tr.(m) |
| K | ICP+Det. | 0.56 | 0.43 | 0.44 | 0.22 | 6.27 | **0.004** | 0.44 |
| K | 3DMatch+Det. | 0.89 | 0.70 | 0.71 | 0.021 | 1.80 | **0.004** | 0.68 |
| K | FPFH+Det. | 3.83 | 4.24 | 4.21 | 0.299 | 14.23 | 0.135 | 4.27 |
| K | Dewan et al.+Det. | 0.55 | 0.41 | 0.41 | 0.008 | 0.55 | 0.006 | 0.39 |
| K | Ours | **0.29** | **0.15** | **0.16** | **0.004** | **0.19** | 0.005 | **0.12** |
| K+AK | ICP+Det. | 0.74 | 0.48 | 0.50 | 0.226 | 6.30 | **0.005** | 0.49 |
| K+AK | 3DMatch+Det. | 1.14 | 0.77 | 0.80 | 0.027 | 1.76 | **0.004** | 0.76 |
| K+AK | FPFH+Det. | 4.00 | 4.39 | 4.36 | 0.311 | 13.54 | 0.122 | 4.30 |
| K+AK | Dewan et al.+Det. | 0.60 | 0.52 | 0.52 | 0.014 | 0.75 | 0.006 | 0.46 |
| K+AK | Ours | **0.34** | **0.18** | **0.20** | **0.011** | **0.50** | **0.005** | **0.15** |

Table 2: **Comparison to Baselines** on test sets of KITTI and Augmented KITTI, abbreviated with K and AK, respectively.

on the horizontal and vertical angle of the rays, based on KITTI LIDAR scans. Our simulator also produces missing estimates at transparent surfaces by ignoring them with a probability equal to their transparency value provided by the CAD models, as illustrated in Figure 4. Additionally, we remove points in the original scan which become occluded by the augmented car by tracing a ray between each point and the LIDAR, and removing those points whose ray intersects with the car mesh. Finally, we sample the augmented car's rigid motion using a simple approximation of the Ackermann steering geometry, place the car at the corresponding location in the next frame, and repeat the LIDAR simulation. We generate 20k such frames with 1 to 3 augmented moving cars per scene. We split the sequences into 70% train, 10% validation, 20% test similar to our split of the original KITTI dataset.

### 4.2. Baseline Methods

We compare our method to four baselines: a point cloud-based method using a CRF [8], two point-matching methods, and an Iterative Closest Point [4] (ICP) baseline.

**Dewan et al.** [8] estimate per-point rigid motion. To arrive at object-level motion and ego-motion, we pool the estimates over our object detections and over the background. As they only estimate valid scene flow for a subset of the points, we evaluate [8] only on those estimates and the comparison is therefore inherently biased in their favor.

**Method Matching 3D Descriptors** yield a scene flow estimate for each point in the reference point cloud by finding correspondences of 3D features in two timesteps. We evaluate two different descriptors: 3D Match [47], a learnable 3D descriptor trained on KITTI and Fast Point Feature Histogram features (FPFH) [31]. Based on the per-point scene flow, we fit rigid body motions to each of the objects and to the background, again using the object detections from our pipeline for a fair comparison.

**Iterative Closest Point (ICP)** [4] outputs a transformation relating two point clouds to each other using an SVD-based point-to-point algorithm. We estimate object rigid motions by fitting the points of each detected 3D object in the first point cloud to the entire second point cloud.

**Evaluation Metrics:** We quantify performance using several metrics applied to both the detected objects and the background. To quantify the accuracy of the estimates independently from the detection accuracy, we only evaluate object motion on true positive detections.

- For 3D scene flow, we use the average endpoint error between the prediction and the ground truth.

- Similarly, we list the average rotation and translation error averaged over all of the detected objects, and averaged over all scenes for the observer's ego-motion.

(a) Ground Truth          (b) Our result

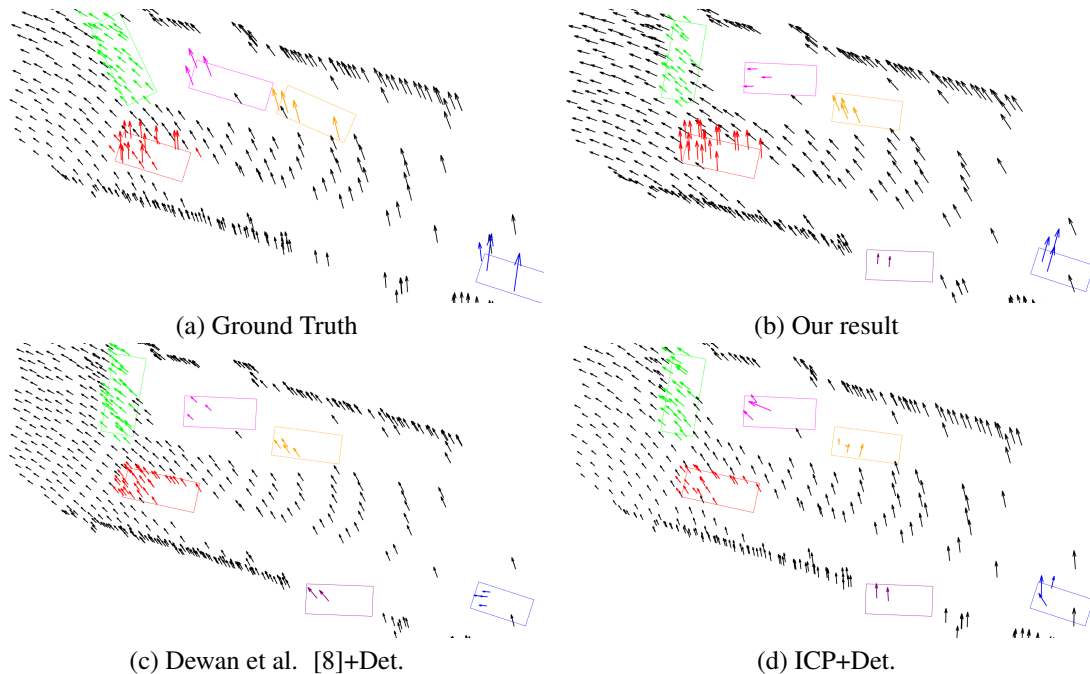(c) Dewan et al. [8]+Det.          (d) ICP+Det.

Figure 5: **Qualitative Comparison** of our method with the best performing baseline methods on an example from the Augmented KITTI. For clarity, we visualize only a subset of the points. Additional results can be found in the supplementary.

## 4.3. Experimental Results

**The importance of simulated augmentation:** To quantify the value of our proposed LIDAR simulator for realistic augmentation with extra cars, we compare the performance of our method trained on the original KITTI object detection dataset with our method trained on both KITTI and Augmented KITTI. Table 1 shows the results of this study. Our analysis shows that training using a combination of KITTI and augmented KITTI leads to significant performance gains, especially when evaluating on the more diverse vehicle motions in the validation set of Augmented KITTI.

**Direct scene flow vs. object motion:** We have also evaluated the difference between estimating scene flow directly and calculating it from either dense or object-level rigid motion estimates. While scene flow computed from rigid motion estimates was qualitatively smoother, there was no significant difference in overall accuracy.

**Comparison with the baselines:** Table 2 summarizes the complete performance comparison on the KITTI test set. Note that the comparison with Dewan et al. [8] is biased in their favor, as mentioned earlier, as we only evaluate their accuracy on the points they consider accurate. Regardless, our method outperforms all baselines. Additionally, we observe that the ICP-based method exhibits large errors for object motions. This is because of objects with few points: ICP often performs very poorly on these, but while their impact on the dense evaluation is small they constitute a relatively

larger fraction of the object-based evaluation. Visual examination (Fig. 5) shows that the baseline methods predict a reasonable estimate for the background motion, but fail to estimate motion for dynamic objects; in contrast, our method is able to estimate these motions correctly. This further reinforces the importance of training our method on scenes with many augmented cars and challenging and diverse motions.

Regarding execution time, our method requires 0.5 seconds to process one point cloud pair. In comparison, Dewan et al. (4 seconds) and the 3D Match- and FPFH-based approaches (100 and 300 seconds, respectively) require significantly longer, while the ICP solution also takes 0.5 seconds but performs considerably worse.

## 5. Conclusion

In this paper, we have proposed a learning-based solution for estimating scene flow and rigid body motion from unstructured point clouds. Our model simultaneously detects objects in the point clouds, estimates dense scene flow and rigid motion for all points in the cloud, and estimates object rigid motion for all detected objects as well as the observer. We have shown that a global rigid motion representation is not amenable to fully convolutional estimation, and propose to use a local representation. Our approach outperforms all evaluated baselines, yielding more accurate object motions in less time.

## 6. Acknowledgements

# References

[1] A. Asvadi, P. Girao, P. Peixoto, and U. Nunes. 3d object tracking using RGB and LIDAR data. In *Proc. IEEE Conf. on Intelligent Transportation Systems (ITSC)*, 2016.

[2] T. Basha, Y. Moses, and N. Kiryati. Multi-view scene flow estimation: A view centered variational approach. *International Journal of Computer Vision (IJCV)*, 101(1):6–21, 2013.

[3] A. Behl, O. H. Jafari, S. K. Mustikovela, H. A. Alhaija, C. Rother, and A. Geiger. Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios? In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.

[4] P. Besl and H. McKay. A method for registration of 3d shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 14:239–256, 1992.

[5] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] R. Danescu, F. Oniga, and S. Nedevschi. Modeling and tracking the driving environment with a particle-based occupancy grid. *IEEE Trans. on Intelligent Transportation Systems (TITS)*, 12(4):1331–1342, 2011.

[7] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard. Motion-based detection and tracking in 3d lidar scans. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2016.

[8] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard. Rigid scene flow for 3d lidar scans. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[9] A. Dosovitskiy, P. Fischer, E. Ilg, P. Haeusser, C. Hazirbas, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2015.

[10] S. Fidler, S. Dickinson, and R. Urtasun. 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *Advances in Neural Information Processing Systems (NIPS)*, December 2012.

[11] U. Franke, C. Rabe, H. Badino, and S. Gehrig. 6D-Vision: fusion of stereo and motion for robust environment perception. In *Proc. of the DAGM Symposium on Pattern Recognition (DAGM)*, 2005.

[12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[13] F. Groh, P. Wieschollek, and H. P. A. Lensch. Flex-convolution (million-scale point-cloud learning beyond grid-worlds). In *Proc. of the Asian Conf. on Computer Vision (ACCV)*, Dezember 2018.

[14] D. Held, J. Levinson, S. Thrun, and S. Savarese. Robust real-time tracking combining 3d shape, color, and motion. *International Journal of Robotics Research (IJRR)*, 35(1-3):30–49, 2016.

[15] E. Herbst, X. Ren, and D. Fox. RGB-D flow: Dense 3D motion estimation using color and depth. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2013.

[16] M. Hornacek, A. Fitzgibbon, and C. Rother. SphereFlow: 6 DoF scene flow from RGB-D pairs. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[17] F. Huguet and F. Devernay. A variational method for scene flow estimation from stereo sequences. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2007.

[18] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] A. Kendall, H. Martirosyan, S. Dasgupta, and P. Henry. End-to-end learning of geometry and context for deep stereo regression. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.

[20] P. Lenz, J. Ziegler, A. Geiger, and M. Roser. Sparse scene flow segmentation for moving object detection in urban environments. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2011.

[21] Z. Liang, Y. Feng, Y. Guo, H. Liu, L. Qiao, W. Chen, L. Zhou, and J. Zhang. Learning deep correspondence through prior and posterior feature constancy. *arXiv.org*, 1712.01039, 2017.

[22] Z. Lv, C. Beall, P. Alcantarilla, F. Li, Z. Kira, and F. Dellaert. A continuous optimization approach for efficient and accurate scene flow. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[23] N. Mayer, E. Ilg, P. Haeusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[24] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[25] M. Menze, C. Heipke, and A. Geiger. Joint 3d estimation of vehicles and scene flow. In *Proc. of the ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[26] F. Moosmann and C. Stiller. Joint self-localization and tracking of generic objects in 3d range data. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2013.

[27] P. Purkait, C. Zhao, and C. Zach. Spp-net: Deep absolute pose regression with synthetic views. *arXiv.org*, 1712.03452, 2017.

[28] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[29] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[30] J. Quiroga, T. Brox, F. Devernay, and J. L. Crowley. Dense semi-rigid scene flow estimation from RGB-D images. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2014.

[31] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3d registration. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2009.

[32] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M. Yang, and J. Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[33] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[34] G. Tanzmeister, J. Thomas, D. Wollherr, and M. Buss. Grid-based mapping and tracking in dynamic environments using a uniform evidential environment representation. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2014.

[35] F. Tombari, S. Salti, and L. di Stefano. Unique signatures of histograms for local surface description. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2010.

[36] A. K. Ushani, R. W. Wolcott, J. M. Walls, and R. M. Eustice. A learning approach for real-time temporal scene flow estimation from LIDAR data. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2017.

[37] L. Valgaerts, A. Bruhn, H. Zimmer, J. Weickert, C. Stoll, and C. Theobalt. Joint estimation of motion, structure and geometry from stereo sequences. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2010.

[38] J. van de Ven, F. Ramos, and G. D. Tipaldi. An integrated probabilistic model for scan-matching, moving object detection and motion estimation. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2010.

[39] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1999.

[40] S. Vedula, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 27(3):475–480, 2005.

[41] C. Vogel, K. Schindler, and S. Roth. 3D scene flow estimation with a rigid motion prior. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2011.

[42] C. Vogel, K. Schindler, and S. Roth. 3d scene flow estimation with a piecewise rigid scene model. *International Journal of Computer Vision (IJCV)*, 115(1):1–28, 2015.

[43] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun. Deep parametric continuous convolutional neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[44] A. Wedel, T. Brox, T. Vaudrey, C. Rabe, U. Franke, and D. Cremers. Stereoscopic scene flow computation for 3D motion understanding. *International Journal of Computer Vision (IJCV)*, 95(1):29–51, 2011.

[45] A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, and D. Cremers. Efficient dense scene flow from sparse or dense stereo data. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2008.

[46] J. Žbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research (JMLR)*, 17(65):1–32, 2016.

[47] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[48] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.