MotionNet: Joint Perception and Motion Prediction for Autonomous Driving Based on Bird's Eye View Maps

Pengxiang Wu*
Rutgers University
pw241@cs.rutgers.edu

Siheng Chen
Mitsubishi Electric Research Laboratories
schen@merl.com

Dimitris Metaxas
Rutgers University
dnm@cs.rutgers.edu

Abstract

The ability to reliably perceive the environmental states, particularly the existence of objects and their motion behavior, is crucial for autonomous driving. In this work, we propose an efficient deep model, called MotionNet*, to jointly perform perception and motion prediction from 3D point clouds. MotionNet takes a sequence of LiDAR sweeps as input and outputs a bird's eye view (BEV) map, which encodes the object category and motion information in each grid cell. The backbone of MotionNet is a novel* spatio-temporal pyramid network*, which extracts deep spatial and temporal features in a hierarchical fashion. To enforce the smoothness of predictions over both space and time, the training of MotionNet is further regularized with novel spatial and temporal consistency losses. Extensive experiments show that the proposed method overall outperforms the state-of-the-arts, including the latest scene-flow- and 3D-object-detection-based methods. This indicates the potential value of the proposed method serving as a backup to the bounding-box-based system, and providing complementary information to the motion planner in autonomous driving. Code is available at* `https://www.merl.com/research/license#MotionNet`.

1. Introduction

Determining the environmental states is critical for deploying autonomous vehicles (AVs) [11]. Accurate state information would facilitate motion planning and provide smooth user experience. The estimation of environmental state typically comprises two tasks: (1) perception, which identifies the foreground objects from the background; (2) motion prediction, which predicts the future trajectories of objects. In the past years, various methods have been developed to handle these two tasks independently or jointly, achieving remarkable progress with the aid of deep learning [22, 5]. In this work, we consider joint perception and motion prediction from a sequence of LiDAR point clouds.

Traditional approaches to the perception of environment mainly rely on the bounding box detection, which is implemented through 2D object detection based on camera data [41, 27, 20, 63], 3D object detection based on LiDAR data [64, 19, 46], or fusion-based detection [6, 24, 23]. The

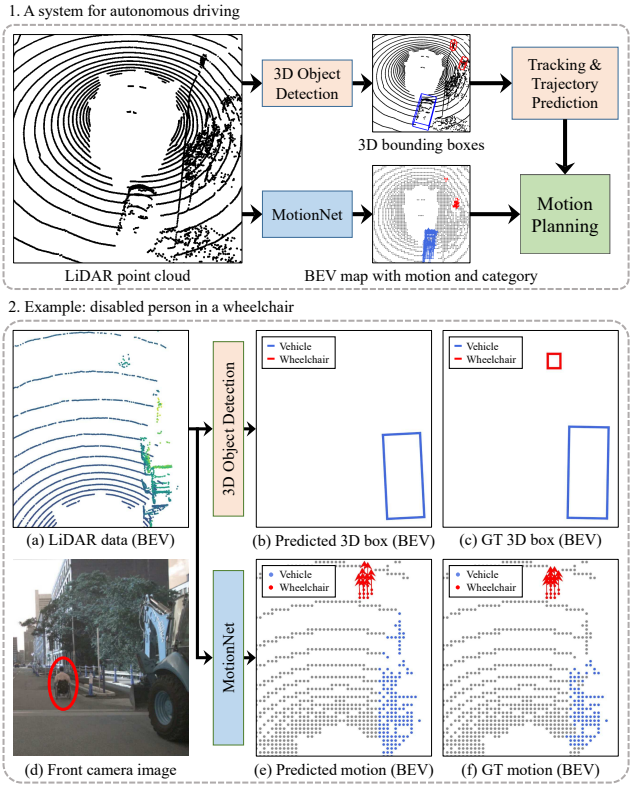


Figure 1. **Top:** *MotionNet* is a system based on bird's eye view (BEV) map, and performs perception and motion prediction jointly without using bounding boxes. It can potentially serve as a backup to the standard bounding-box-based-system and provide complementary information for motion planning. **Bottom:** During testing, given an object (e.g., disabled person on a wheelchair, as illustrated in (d)) that never appears in the training data, 3D object detection (e.g., [46]) tends to fail; see plots (b) and (c). In contrast, MotionNet is still able to perceive the object and forecast its motion; see plots (e) and (f), where the color represents the category and the arrow denotes the future displacement.

*Work done during an internship at MERL.

detected bounding boxes are then fed into an object tracker, followed by a motion predictor; see Fig. 1(1). Some recent works implement all these modules into an end-to-end framework, which directly produces bounding boxes along with future trajectories [31, 4, 59]. While being widely adopted, the above state estimation strategies tend to fail in open-set scenarios of real traffic due to the dependency on object detection. In particular, the object detectors are difficult to generalize to classes that have never been present in the training set, consequently leading to catastrophic failures for the downstream modules, as illustrated in Fig. 1(2).

One alternative direction is to represent the 3D environmental information by using an occupancy grid map (OGM) [14, 34, 44]. An OGM discretizes the 3D point cloud into equal 2D grid cells, each of which contains the belief that the corresponding space is occupied by at least one point. With this design, OGMs can be used to specify the drivable space into the future and thereby provide support for motion planning. One major weakness of OGM is the difficulty to find the correspondence between the cells across time. This makes it difficult to explicitly model the dynamics of objects. In addition, the object category information is typically discarded in OGMs, and thus it is impossible to consider category-specific constraints on the motions of traffic actors for relationship understanding.

To address these weaknesses, we represent the environmental state based on a bird's eye view (BEV) map. Similar to OGM, we discretize a point cloud around ego-vehicle into independent cells (i.e., a BEV map). The BEV map extends OGM and provides three-fold information: occupancy, motion, and category information; see Fig. 2. We encode the motion information by associating each cell with displacement vectors, which represent the positions into the future and could characterize nonlinear dynamics. In this way, we are able to determine the drivable space as well as describe the motion behavior of each individual object. The cell categories are derived from the object they belong to, and are used to facilitate the understanding of environment.

Based on a temporal sequence of such BEV maps, we propose a novel deep model for jointly reasoning about the category and motion information for each cell. We name our model *MotionNet*, with an emphasis on its ability to predict motions, even for unseen objects in the training set. MotionNet is bounding-box free, and is able to leverage motion clues for object recognition. The core of MotionNet is a novel *spatio-temporal pyramid network* (STPN). To extract the spatio-temporal features, STPN performs a series of spatio-temporal convolutions (STC) in a hierarchical fashion. Each STC relies on 2D spatial convolutions, followed by a light-weight pseudo-1D temporal convolution, yielding an efficient system. In practice, MotionNet runs at 53 Hz, making it suitable to deploy in real-time systems. The outputs of STPN are delivered to different heads for cell classification, state estimation and motion prediction, respectively; see Fig. 2. During inference, to make the predictions consistent across tasks, we regularize the predicted motions with the guide of classification results. To further enforce the smoothness of predictions over space and time, we constrain the network training with several novel spatial and temporal consistency losses, which promote more realistic motion forecast.

We evaluate our approach on the large-scale nuScenes dataset [3] and compare with different prior arts for environmental state estimation, including those based on scene flow and object detection. Experimental results demonstrate the effectiveness and superiority of our method. Our study shows the potential value of MotionNet in the real-world settings for autonomous driving: it can work collaboratively with other modules, and provide complementary perception and motion information for motion planning.

To summarize, the main contributions of our work are:

- We propose a novel model, called MotionNet, for joint perception and motion prediction based on BEV maps. MotionNet is bounding-box free and can provide complementary information for autonomous driving;
- We propose a novel spatio-temporal pyramid network to extract spatio-temporal features in a hierarchical fashion. This structure is light-weight and highly efficient, and thus is suitable for real-time deployment;
- We develop spatial and temporal consistency losses to constrain the network training, enforcing the smoothness of predictions both spatially and temporally; and
- Extensive experiments validate the effectiveness of our method, and in-depth analysis is provided to illustrate the motivations behind our design.

2. Related Work

Perception. This task aims to identify the locations and categories of objects in the surrounding environments. One typical formulation of this task is the bounding box detection. Depending on the input modality, existing works can be divided into three categories: (1) 2D object detection on images [41, 7, 27, 40, 26, 20, 63]; (2) 3D object detection on point clouds [58, 18, 57, 48, 64, 56, 19, 47, 55, 36, 46, 35], and (3) fusion-based detection [6, 24, 23]. Nevertheless, object detection relies on shape recognition and is difficult to detect objects whose categories are never present in the training set. This would cause fatal consequences in numerous real-world scenarios. In contrast to bounding boxes, the proposed BEV-map-based representation extends occupancy maps and does not rely on shape recognition. The resulting system is able to perceive salient traffic actors and provide complementary information to the motion planner.

Motion prediction. This task aims to predict the future positions of objects based on the history information. Classical methods typically formulate this task as trajectory pre-

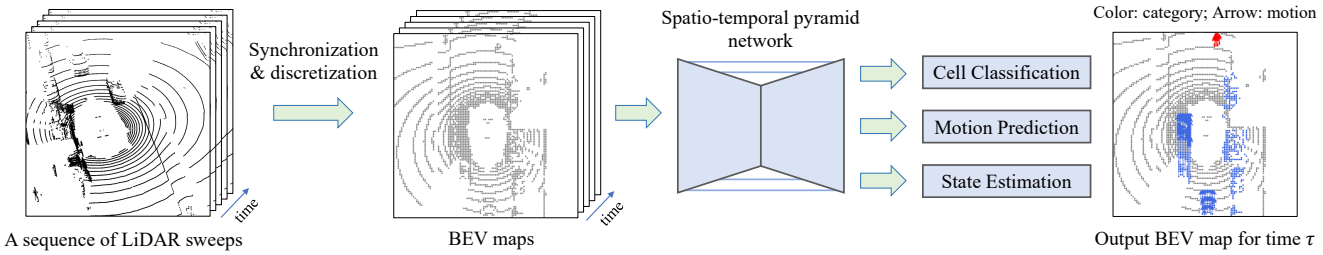


Figure 2. Overview of **MotionNet**. Given a sequence of LiDAR sweeps, we first represent the raw point clouds into BEV maps, which are essentially 2D images with multiple channels. Each pixel (cell) in a BEV map is associated with a feature vector along the height dimension. We then feed the BEV maps into the spatio-temporal pyramid network (STPN) for feature extraction. The output of STPN is finally delivered to three heads: (1) cell classification, which perceives the category of each cell, such as vehicle, pedestrian or background; (2) motion prediction, which predicts the future trajectory of each cell; (3) state estimation, which estimates the current motion status of each cell, such as static or moving. The final output is a BEV map, which includes both perception and motion prediction information.

diction, which, however, relies on accurate object detection and tracking for trajectory acquisition [1, 21, 32, 13, 8, 42, 62, 43, 60, 33]. Another direction proposes to jointly perform 3D detection, tracking and motion forecasting, and has demonstrated remarkable performance [31, 4, 59]. Still, due to the dependence on the bounding box detection, such a strategy tends to fail in the presence of unexpected objects. This weakness can be circumvented with occupancy grid map [10], particularly the multi-step dynamic OGMs [14, 34, 44], which represent the object locations and dynamics with the occupancy state of cells and their associated velocities, respectively. This representation is able to represent the drivable space and motions easily without the need for object boxes. However, due to the lack of cell correspondences between OGMs across time, it is difficult to model the nonlinear dynamic behavior of objects. This property, together with another one that OGM typically ignores object categories, make it impractical to explicitly capture the object interaction relationships. In contrast, the proposed BEV-map-based representation contains both category and motion information.

Flow estimation. Different from motion prediction, this task aims to estimate the motion from the past to current time. Depending on the input data, the motion information can be extracted from 2D optical flow [15, 30, 9, 16] or 3D scene flow [12, 28, 29]. In practice, we can exploit the estimated flow to predict future trajectories by assuming linear dynamics, as demonstrated in Sec. 4.

3. Methodology

In this section, we present MotionNet; see Fig. 2. The pipeline includes three parts: (1) data representation from raw 3D point clouds to BEV maps; (2) spatio-temporal pyramid network as the backbone; and (3) task-specific heads for grid cell classification and motion prediction.

3.1. Ego-motion compensation

Our input is a sequence of 3D point clouds, where each original point cloud frame is described by its local coordinate system. We need to synchronize all the past frames to the current one, i.e., represent all the point clouds within the current coordinate system of ego vehicle via coordinate transformation. This is critical for counteracting the ego-motion of AV and avoiding specious motion estimation. In addition, it aggregates more points for the static background while providing clues on the motions of moving objects.

3.2. BEV-map-based representation

Unlike 2D images, 3D point clouds are sparse and irregularly scattered, and thus cannot be processed directly with standard convolutions. To address this issue, we convert the point clouds into BEV maps, which are amenable to classic 2D convolutions. Specifically, we first quantize the 3D points into regular voxels. Different from [64, 56], which encode the point distribution within each voxel into high-level features through PointNet [37], we simply use a binary state as a proxy of a voxel, indicating whether the voxel is occupied by at least one point. Then we represent the 3D voxel lattice as a 2D pseudo-image, with the height dimension corresponding to image channels. Such a 2D image is virtually a BEV map, where each cell is associated with a binary vector along the vertical axis. With this representation, we can apply 2D convolutions to the BEV maps rather than the 3D convolutions for feature learning.

Compared to prior arts relying on 3D voxels [64, 56] or raw point clouds [38, 52], our approach allows employing standard 2D convolutions, which are well supported in both software and hardware levels, and therefore is extremely efficient [53]. In addition, the BEV maps keep the height information as well as the metric space, allowing the network to leverage priors on the physical extensions of objects [58].

3.3. Spatio-temporal pyramid network

As described above, the input to our model is virtually a sequence of 2D pseudo-images. To efficiently capture the spatio-temporal features, we follow the spirit of recent studies on video classification task, which suggests replacing the bulky 3D convolutions with the low-cost ones (e.g., 2D

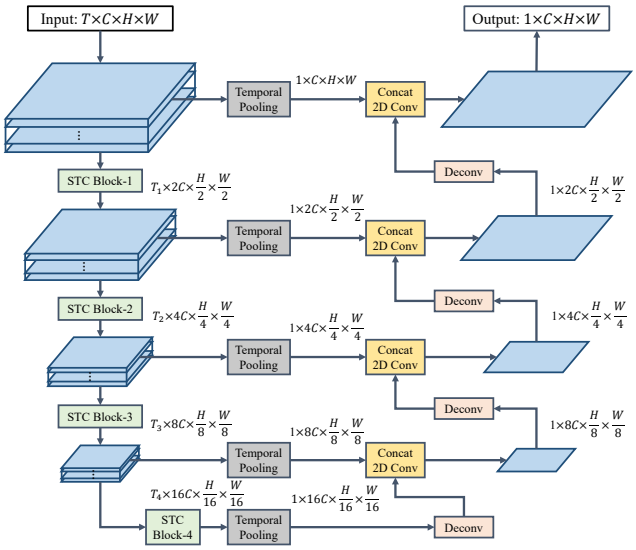
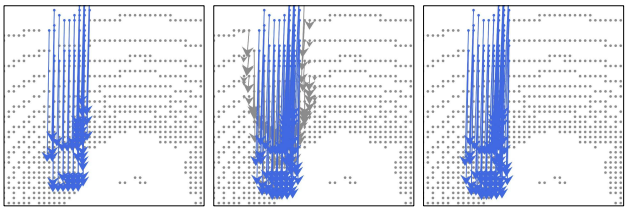


Figure 3. Spatio-temporal pyramid network. Each STC block consists of two consecutive 2D convolutions followed by one pseudo-1D convolution. The temporal pooling is applied to the temporal dimension and squeezes it to length 1. $T_1 \geq T_2 \geq T_3 \geq T_4$.

convolutions) [39, 51, 54, 50, 25]. However, unlike classical video classification task which only predicts one category label for the whole image sequence, we aim to classify each BEV lattice cell at the current time and estimate its future position. In particular, there are two issues that need to be addressed. First, when and how to aggregate the temporal features. As is indicated in [51, 54], the timing of temporal convolutions is critical for achieving good performance. Second, how to extract the multi-scale spatio-temporal features, which are known to be essential for capturing both local and global contexts in dense prediction task [61].

To address these issues, we develop a spatio-temporal pyramid network (STPN) to extract features along both the spatial and temporal dimensions in a hierarchical fashion; see Fig. 3. The basic building block of STPN is the spatio-temporal convolution (STC) block. Each STC block consists of standard 2D convolutions, followed by a degenerate 3D convolution, to capture the spatial and temporal features, respectively. The kernel size of the 3D convolution is $k \times 1 \times 1$, where k corresponds to the temporal dimension. Such a 3D filter is essentially a *pseudo-1D convolution* and thus enables a reduction of model complexity.

To promote multi-scale feature learning, STPN computes a feature hierarchy over the space and time with STC blocks. In particular, for the spatial dimension, we compute the feature maps at several scales with a scaling step of 2. Similarly, for the temporal dimension, we gradually reduce the temporal resolution after each temporal convolution, thereby extracting temporal semantics of different scales. To fuse the spatio-temporal features across different levels, we perform global temporal pooling to capture



(a) Ground-truth (b) Before suppression (c) After suppression

Figure 4. The outputs of cell-classification and state-estimation heads can be used to suppress the undesirable jitters (e.g., background may have non-zero motion). Gray: background; blue: vehicle. Arrow: motion. (Zoom in for best view.)

the salient temporal features, and deliver them to the upsampled layers of feature decoder via lateral connections. This design encourages the flow of local and global spatio-temporal contexts, which is beneficial to our dense prediction task. The overall structure of STPN only relies on 2D and pseudo-1D convolutions and thus is highly efficient.

3.4. Output heads

To generate the final outputs, we append three heads to the end of STPN: (1) cell-classification head, which essentially performs BEV map segmentation and perceives the category of each cell; (2) motion-prediction head, which forecasts the positions of cells into the future; and (3) state-estimation head, which estimates the motion status for each cell (i.e., static or moving) and provides auxiliary information for motion prediction. We implement these three heads with two-layer 2D convolutions. For the cell-classification head, the shape of output is $H \times W \times C$, where C is the number of cell categories. For motion-prediction head, it represents the predicted cell positions as $\{X^{(\tau)}\}_{\tau=t}^{t+N}$, where $X^{(\tau)} \in \mathbb{R}^{H\times W\times 2}$ denotes the positions at time τ, t is the current time and N is the number of future frames; thus its output shape is $N \times H \times W \times 2$. Note that the motion is assumed to be on the ground, which is reasonable in autonomous driving as traffic actors do not fly. For the state-estimation head, the shape of output is $H \times W$, where each element denotes the probability of being static.

The motion-prediction head can be trained with regression loss (e.g., smooth L1). However, naively regressing the future positions of cells will lead to undesirable *jitters* of static cells. For example, even though the cells are classified as background, they could still have small movements; see Fig. 4. To remedy this issue, we use the outputs from the other two heads to regularize the predicted cell trajectories. Specifically, we threshold the motions for cells that are predicted as background, i.e., set their corresponding motion estimations to zero. In addition, to deal with the static foreground objects, such as parking vehicles, we use the estimated states from the state-estimation head, and suppress the jitter effect by thresholding the motions of static cells.

Remarks. Compared to bounding-box-based methods, the above design potentially enables to better perceive the *unseen objects* beyond training set. The intuitions are: (1) the box-based methods capture objects using ROI global shape/texture information, which is different across object categories and hard to generalize from seen objects to unseen ones. In contrast, our method effectively decomposes ROIs into grid cells, and in each cell it extracts local information shared by many object categories; (2) the box-based methods involve object proposals and NMS, which might remove uncertain detections (especially for the unseen); while our method makes predictions for all occupied cells; and (3) temporal information leveraged by MotionNet provides clues on the existence of objects and their motions.

3.5. Loss function

We train the network to simultaneously minimize the losses associated with three heads. For the classification and state-estimation heads, we employ the cross-entropy loss, where each category term is assigned a different weight so as to handle the class imbalance issue. For the motion-prediction head, we adopt weighted smooth L1 loss, where the weights are determined following the same specification of classification head. However, the above losses are only able to regularize the network training globally, but do not ensure the spatial and temporal consistencies locally. To address this weakness, we introduce additional losses below.

Spatial consistency loss. Intuitively, for the cells belonging to the same rigid object, their predicted motions should be very close without much divergence. Inspired by this observation, we constrain the estimated motions locally with the following spatial consistency loss:

$$L_{\mathrm{s}} = \sum_{k} \sum_{(i,j),(i',j') \in o_k} \left\| X_{i,j}^{(\tau)} - X_{i',j'}^{(\tau)} \right\|, \qquad (1)$$

where $\|\cdot\|$ is the smooth L1 loss, o_k denotes the object instance with index k, and $X_{i,j}^{(\tau)} \in \mathbb{R}^2$ is the predicted motion at position (i,j) and time τ. Note that it is computationally expensive to exhaustively compare all pairs of $X_{i,j}^{(\tau)}$ and $X_{i',j'}^{(\tau)}$. To avoid this, we only consider a subset of pairs, each of which involves two positions adjacent in index.

Foreground temporal consistency loss. Similar to spatial consistency, we can also pose temporal constraint over the local time window. In particular, for each object, we can reasonably assume that there will be no sharp change of motions between two consecutive frames. This assumption can be achieved by minimizing the following loss:

$$L_{\mathrm{ft}} = \sum_{k} \left\| X_{o_k}^{(\tau)} - X_{o_k}^{(\tau+\Delta t)} \right\|, \qquad (2)$$

where $X_{o_k}^{(\tau)} \in \mathbb{R}^2$ denotes the overall motion of object k, which in our implementation is represented by the average motion: $X_{o_k}^{(\tau)} = \sum_{(i,j)\in o_k} X_{i,j}^{(\tau)} / M$, where M is the number of cells belonging to o_k.

Background temporal consistency loss. Note that L_{ft} mainly operates on the foreground objects, such as vehicles, and does not consider the background cells. As a compensation for this weakness, we introduce another temporal loss:

$$L_{\mathrm{bt}} = \sum_{(i,j)\in X^{(\tau)} \cap T(\widetilde{X}^{(\tau-\Delta t)})} \left\| X_{i,j}^{(\tau)} - T_{i,j}\left(\widetilde{X}^{(\tau-\Delta t)}\right) \right\|, \qquad (3)$$

where $X^{(\tau)}$ and $\widetilde{X}^{(\tau)}$ are the predictions with current time being t and $t+\Delta t$, respectively; $T \in SE(3)$ is a rigid transformation which aligns $\widetilde{X}^{(\tau-\Delta t)}$ with $X^{(\tau)}$. In practice, T could be derived from the ground-truth ego motion, or from point cloud registration algorithms (e.g., ICP [2]). Note that since $\widetilde{X}^{(\tau-\Delta t)}$ is a discrete grid, the transformed result is interpolated on the cells. After applying this transformation, $T(\widetilde{X}^{(\tau-\Delta t)})$ will be partially overlapped with $X^{(\tau)}$ on the static cells which are mainly background. By minimizing this loss, we encourage the network to produce coherent results on the overlapped regions, thereby leading to temporally smooth predictions.

To summarize, the overall loss function for the training of MotionNet is defined as:

$$L = L_{\mathrm{cls}} + L_{\mathrm{motion}} + L_{\mathrm{state}} + \alpha L_{\mathrm{s}} + \beta L_{\mathrm{ft}} + \gamma L_{\mathrm{bt}}, \quad (4)$$

where L_{cls} and L_{state} are cross-entropy losses for the cell-classification and state-estimation heads, L_{motion} is smooth L1 loss for the motion-prediction head; α, β and γ are the balancing factors. Since L involves multiple tasks, it could be minimized within multi-objective optimization framework, which enables adaptive trade-off between tasks [45].

4. Experiments

In this section, we evaluate the performance of the proposed network on the nuScenes [3] dataset. We first introduce the implementation details of MotionNet, and then compare it with previous state-of-the-art methods. We finally provide ablation studies to analyze our design choices.

Dataset. nuScenes [3] is a large-scale dataset for autonomous driving, and contains different types of sensor data with 360° coverage on the surroundings. In this work, we only utilize its LiDAR point clouds, which are captured with a frequency of 20Hz and collected from 1,000 scenes. Each scene comprises a sequence of LiDAR sweeps with a duration of $20s$. Since the original focus of nuScenes is on object detection, for each sweep it only provides annotated bounding boxes without motion information. To adapt this dataset to our task, we derive the ground-truth cell motions between two sweeps as follows: for each cell inside a bounding box, its motion is computed as $\mathcal{R}x + c_\Delta - x$, where x is the cell position, $\mathcal{R}$ is the yaw rotation with respect to the box center, and c_Δ is the displacement of box

Method	Static		Speed $\leq$ 5m/s		Speed $>$ 5m/s		Classification Accuracy (%)							Infer.
	Mean	Median	Mean	Median	Mean	Median	Bg	Vehicle	Ped.	Bike	Others	MCA	OA	Speed
Static Model	**0**	**0**	0.6111	0.0971	8.6517	8.1412	-	-	-	-	-	-	-	-
FlowNet3D (pretrain) [28]	2.0514	0	2.2058	0.3172	9.1923	8.4923	-	-	-	-	-	-	-	0.434s
FlowNet3D [28]	0.0410	0	0.8183	0.1782	8.5261	8.0230	-	-	-	-	-	-	-	0.434s
HPLFlowNet (pretrain) [12]	2.2165	1.4925	1.5477	1.1269	5.9841	4.8553	-	-	-	-	-	-	-	0.352s
HPLFlowNet [12]	0.0041	0.0002	0.4458	0.0960	4.3206	2.4881	-	-	-	-	-	-	-	0.352s
PointRCNN [46]	0.0204	0	0.5514	0.1627	3.9888	1.6252	98.4	78.7	44.1	11.9	44.0	55.4	96.0	0.201s
LSTM-Encoder-Decoder [44]	0.0358	0	0.3551	0.1044	1.5885	1.0003	93.8	91.0	73.4	17.9	71.7	69.6	92.8	0.042s
MotionNet	0.0256	0	0.2565	0.0962	1.0744	0.7332	97.3	91.1	76.2	20.6	66.1	70.3	96.1	0.019s
MotionNet + L_{s}	0.0256	0	0.2488	0.0958	1.0110	0.7001	97.5	91.3	76.2	23.7	67.6	71.2	96.3	0.019s
MotionNet + L_{ft}	0.0252	0	0.2515	0.0962	1.0360	0.7136	97.6	90.6	75.3	21.9	65.2	70.1	96.3	0.019s
MotionNet + L_{bt}	0.0240	0	0.2530	0.0960	1.0399	0.7131	97.5	91.1	74.6	25.2	68.0	71.3	96.3	0.019s
MotionNet + L_{s} + L_{ft} + L_{bt}	0.0239	0	0.2467	0.0961	1.0109	0.6994	97.6	90.7	77.2	25.8	65.1	**71.3**	**96.3**	0.019s
MotionNet + MGDA	0.0222	0	0.2366	0.0953	0.9675	0.6639	97.1	90.5	78.4	22.1	67.4	71.1	95.7	0.019s
MotionNet + $\{L\}$ + MGDA	0.0201	0	**0.2292**	**0.0952**	**0.9454**	**0.6180**	97.0	90.7	77.7	19.7	66.3	70.3	95.8	0.019s

Table 1. Performance comparison on perception and motion prediction. MotionNet is significantly faster than all the baselines and overall achieves the best performance. The proposed spatial and temporal consistency losses are able to help improve the accuracy of MotionNet.

center; for those cells outside bounding boxes, we simply set their motions to be zero. In nuScenes, the box annotations are only accessible for the training and validation sets, and therefore we only use them as our experimental data and ignore the official testing data. As a result, we have 850 scenes in total, and in the experiment we use 500 of them for training, 100 for validation and 250 for testing.

We divide each scene into short clips as the input of networks. To reduce redundancy, each clip only consists of a keyframe that corresponds to the current time, and four history sweeps that are synchronized to the keyframe. The keyframes are sampled at 2Hz for training, while for val/testing they are sampled at 1Hz to reduce the similarity between clips. The time span between each two consecutive frames in a clip is $0.2s$. For the training data, apart from the keyframe clips, we extract additional clips whose current time is $(t + 0.05)s$, where t represents the time of neighboring keyframe. These additional clips are paired with the keyframe ones to compute the temporal consistency losses. In summary, we have 17,065 clip pairs for training, 1,719 clips for validation and 4,309 clips for testing.

Implementation details. The point clouds are cropped to reside within a region defined by $[-32, 32] \times [-32, 32] \times [-3, 2]$ meters, which correspond to the XYZ ranges, respectively[1]. The resolution of each partitioned voxel is $(\Delta x, \Delta y, \Delta z) = (0.25, 0.25, 0.4)$ m. For the temporal information, we use 5 frames of synchronized point clouds, where 4 are from the past timestamps and 1 corresponds to the current time. We define 5 cell categories for the perception: background, vehicle (comprising car and bus), pedestrian, bicycle and others. The "others" category includes all the remaining foreground objects from nuScenes, and is introduced to handle the possibly unseen objects beyond training data. Note that such a setting makes the classification task fairly challenging, as the objects in the "others" category involve various shapes and some of them are similar to those from the "vehicle" category in appearance.

For MotionNet, its input is a 4D tensor of size $5 \times 13 \times 256 \times 256$. Before feeding this tensor to STPN, we firstly lift its channel size to 32 with two-layer 2D convolutions. As for STPN, we employ the spatio-temporal convolutions only in STC blocks 1 and 2, and gradually decrease the temporal resolution by unpadding the feature maps. This gives $T_1 = 5$, $T_2 = 3$, $T_3 = T_4 = 1$. As a result, STC blocks 3 and 4 degenerate to regular 2D convolutions. For the motion estimation, we predict the positions of each cell at timestamps $\{\tau\}_{\tau=t+0.05}^{t+1}$, where t is the current time. However, instead of directly regressing the motions, we predict the relative displacement between two adjacent timestamps, i.e., $\Delta d_\tau = d_{\tau+0.05} - d_\tau$, where d_τ denotes the displacement from current time t to future time τ. Therefore, during inference, the absolute displacement at timestamp τ is calculated as $d_\tau = \sum_{i=t}^{\tau-0.05} \Delta d_i$. Finally, for the training loss, we set the balancing factors as $\alpha = 15$, $\beta = 2.5$, $\gamma = 0.1$.

Evaluation criteria. For motion prediction, we evaluate the performance by dividing the cells into 3 groups, which have different speeds: static, slow ($\leq$ 5m/s), and fast ($>$ 5m/s). In each group, we compute the average L_2 distances between the estimated displacements and the ground-truth displacements. Apart from this mean value, we also report the median value. For the classification, we measure the performance with two metrics: (1) overall cell classification accuracy (OA), which is the average accuracy over all cells; (2) mean category accuracy (MCA), which is the average accuracy over all five categories. All the evaluations only involve the non-empty cells.

4.1. Comparison with state-of-the-art methods

Baselines. We compare with the following methods: (1) *Static Model*, which assumes the environment is static. (2) *FlowNet3D* [28] and *HPLFlowNet* [12], which estimate the scene flow between two point clouds. We employ these two methods by assuming linear dynamics: given flow Δd be-

[1]The nuScenes dataset adopts 32-line LiDAR. Distant objects have too few LiDAR points to do detection.

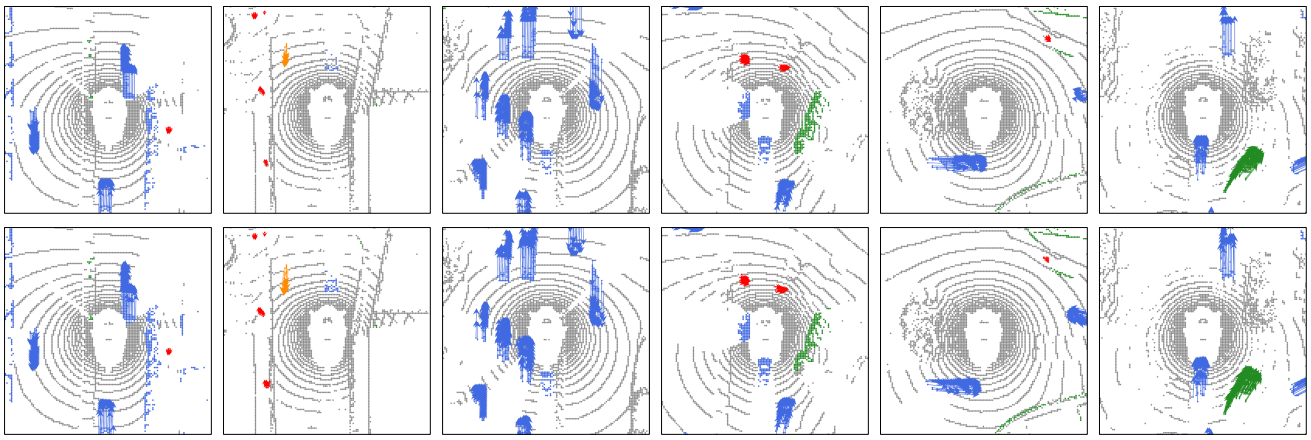


Figure 5. Qualitative results show that MotionNet produces both high-quality classification and motion prediction. Top row: ground-truth. Bottom: MotionNet predictions. Gray: background; blue: vehicle; red: pedestrian; orange: bicycle; green: others. (Zoom in for best view.)

tween two point clouds at time $t-\delta$ and t, we can predict the flow from current time t to the future time $t+n\delta$ as $n\Delta d$. The predicted flow is then projected onto BEV map for performance evaluation. (3) *PointRCNN* [46], which predicts the 3D object bounding boxes from the raw point cloud. After obtaining the bounding boxes for the sequence of point clouds, we use Kalman filter [17] to track the objects and predict their future trajectories. The trajectories are finally converted to BEV map. Note that, following [46], here we train 4 models to separately handle each object category, and the final detection results are obtained by combining the outputs from each model. (4) *LSTM-Encoder-Decoder* [44], which estimates the multi-step OGMs. We adapt this method to our task by using the same output heads with MotionNet, while preserving its backbone structure.

Results. We list the performance of different methods in Table 1, where motions are predicted 1s into the future. As can be seen, our method is significantly faster than the baselines, and outperforms them by a large margin for slow and fast cell speeds. For static case, the Static Model achieves the best result, which is not surprising. However, the Static Model is only used to demonstrate the theoretical limit and is not reasonable to deploy in reality. In Table 1 we also report the performance of FlowNet3D and HPLFlowNet which are pretrained on FlyingThings3D [28, 12] and tested on nuScenes without fine-tuning. As is shown, their performances are even inferior to that of Static Model. Although this situation can be improved by training them directly on nuScenes LiDAR data, their overall performance is still far from good: HPLFlowNet behaves similarly to Static Model while FlowNet3D is worse. Finally, in Table 1 we observe that the performance of PointRCNN is not satisfying. This is mainly due to the unstable object detection in point cloud sequence, which leads to significant failure of trajectory prediction. In contrast, our method predicts the motion more accurately and efficiently, indicating its potential value in providing complementary information to the motion planning. We show the qualitative results in Fig. 5.

Frame #	Static	Speed $\leq$ 5m/s	Speed $>$ 5m/s	MCA	OA	Infer. Speed
2	0.0270	0.2921	1.2445	69.7	95.6	**0.013s**
3	0.0264	0.2738	1.0953	69.6	95.9	0.014s
4	0.0258	0.2597	1.0804	70.2	96.0	0.017s
5	0.0256	**0.2565**	**1.0744**	**70.3**	96.1	0.019s
6	**0.0254**	0.2657	1.1220	69.7	**96.2**	0.021s
7	0.0255	0.2582	1.0779	70.0	**96.2**	0.022s

Table 2. The effects of frame number on model performance. For motion prediction, the mean errors are reported. Using frame number 5 enables a good trade-off between efficiency and accuracy.

Table 1 also demonstrates the effectiveness of spatial and temporal consistency losses. In particular, the spatial loss L_{s} benefits the prediction of moving cells, while temporal losses L_{ft} and L_{bt} facilitate the learning of static environment. Their combination further boosts the prediction performance by exploiting their respective advantages. In Table 1 we also give the results when training the network with multiple-gradient descent algorithm (MGDA) [45], which enables adaptive trade-off among the 3 prediction heads. As is shown, MGDA is able to enhance the motion prediction significantly while sacrificing the classification accuracy mildly. When equipped with spatio-temporal consistency losses, MGDA achieves the best motion predictions.

Note that Table 1 shows that the classification accuracy for the "bicycle" category is low. This is mainly due to the limited number of bicycles in the training set. In addition, the size of bicycles is small in the BEV maps, making it difficult to recognize them. This issue cannot be solved even if we increase the training weight for the "bicycle" category.

4.2. Ablation studies

Below we investigate a few design choices of MotionNet.

Number of frames. We show the effects of point cloud frame number in Table 2. As can be seen, more frames

Synch. Strategy	Static	Speed $\leq$ 5m/s	Speed $>$ 5m/s	MCA	OA
No Synch.	0.0281	0.4245	1.7317	67.1	95.2
ICP [2]	0.0279	0.4073	1.6614	67.4	95.3
GT Synch.	**0.0256**	**0.2565**	**1.0744**	**70.3**	**96.1**

Table 3. The effects of sweep synchronization. Ego-motion compensation is important for achieving good performance.

Data Rep.	Static	Speed $\leq$ 5m/s	Speed $>$ 5m/s	MCA	OA	Infer. Speed
Voxel	0.0257	0.2546	**1.0712**	69.6	**96.2**	0.107s
Pillar	0.0258	0.2612	1.0747	70.0	96.1	0.096s
BEV	0.0256	0.2565	1.0744	70.3	96.1	0.019s
$(1.0, 1.0, 0.5)\Delta$	**0.0253**	**0.2540**	1.0752	70.1	96.0	0.024s
$(1.0, 1.0, 1.5)\Delta$	**0.0253**	0.2562	1.0726	70.1	95.9	**0.014s**
$(0.5, 0.5, 0.5)\Delta$	0.0261	0.2561	1.0806	70.5	96.1	0.106s
$(0.5, 0.5, 1.0)\Delta$	0.0269	0.2545	1.0761	**71.0**	95.9	0.064s
$(0.5, 0.5, 1.5)\Delta$	0.0257	0.2547	1.0733	70.9	96.0	0.050s

Table 4. The effects of input data representation. Finer geometric details do not necessarily lead to much better performance, but would introduce extra computational costs.

would lead to improved performance at the cost of extra computation. When the frame number exceeds 5, the model accuracy saturates with small performance gain. Thus, we choose frame number 5 as an accuracy-efficiency trade-off.

Ego-motion compensation. As is shown in Table 3, sweep synchronization affects the model performance greatly. When without synchronization, the performance drops significantly compared to the one using ground-truth alignment between point clouds. This validates the importance of ego-motion compensation. From Table 3 we also see that ICP [2] is able to help undo the ego-motion to some extent, but is still inferior to using ground-truth synchronization.

Input data representations. We study the effects of different data representations on model performance. In particular, we consider replacing input BEV maps with voxels [64] or pillars [19] which contain fine geometric information. To further explore the effects of shape details, we adjust the resolution of binary voxels in our BEV maps. For example, in Table 4, $(0.5, 0.5, 0.5)\Delta = (0.5\Delta x, 0.5\Delta y, 0.5\Delta z)$ means subdividing the voxels by half, which generates $8\times$ more binary voxels for the original $\Delta x \times \Delta y$ region. To produce the final input BEV map, we reshape the subdivided voxels into a binary vector for each $\Delta x \times \Delta y$ region, thus growing the size of feature channel by $8\times$. From Table 4 we can see that, fine geometric details do not necessarily lead to improved performance for our task, but instead would introduce extra computational costs. Our BEV representation enables a good trade-off between accuracy and speed.

Spatio-temporal feature extraction. To validate our design choice, we compare our method with another two variants which aggregate spatio-temporal features at different times: (1) Early fusion, which first uses two STC blocks (without spatial downsampling) to gradually reduce the temporal resolution, and then employs STPN but discards its temporal convolutions; i.e., $T_i = 1, i \in [1, 4]$; (2) Late fusion, which also uses STPN but only employs temporal convolutions in STC blocks 3 and 4; i.e., $T_1 = T_2 = 5, T_3 = 3, T_4 = 1$. Against these two variants, we consider our method as middle fusion, which overall achieves the best accuracy (see Table 5). The reason could be that, for early fusion, there is little correlation over the frames within a temporal receptive field, especially for objects moving fast; for late fusion, it ignores too many low-level motion cues. Under the framework of middle fusion, we also investigate several other spatio-temporal convolutions, including C3D [49], S3D [54], TSM [25] and CS3D [50]. Specifically, we replace the 2D and pseudo-1D convolutions of STC with the above operations, while keeping the other network components fixed. Table 5 shows that our STC block achieves the best trade-off between accuracy and speed.

Block \ Fusion	Early	Mid	Late	Static	Speed $\leq$ 5m/s	Speed $>$ 5m/s	MCA	OA	Infer. Speed
STC	✓			0.0271	0.2596	1.1002	70.5	96.0	**0.015s**
STC		✓		**0.0256**	**0.2565**	**1.0744**	70.3	**96.1**	0.019s
STC			✓	0.0256	0.2748	1.0838	70.4	96.0	0.019s
C3D [49]		✓		0.0257	0.2624	1.0831	70.5	96.1	0.021s
S3D [54]		✓		0.0267	0.2644	1.1236	70.9	95.9	0.019s
TSM [25]		✓		0.0262	0.2651	1.1241	70.9	96.0	0.018s
CS3D [50]		✓		0.0261	0.2631	1.1787	**71.0**	96.0	0.021s

Table 5. Different strategies for spatio-temporal feature fusion. Overall, the middle fusion with STC blocks provides the best trade-off between accuracy and efficiency.

	State Head	Relative Offset	J.S. w/ Cls	J.S. w/ State	Static	Speed $\leq$ 5m/s	Speed $>$ 5m/s	MCA	OA
1		✓	✓		0.0284	0.2610	1.0957	69.8	95.0
2	✓		✓	✓	0.0264	0.2621	1.1121	70.2	95.8
3	✓	✓			0.0331	**0.2547**	**1.0601**	70.3	96.1
4	✓	✓	✓		0.0259	0.2564	1.0722	70.3	96.1
5	✓	✓		✓	0.0264	0.2554	1.0657	70.3	96.1
6	✓	✓	✓	✓	**0.0256**	0.2565	1.0744	**70.3**	**96.1**

Table 6. The effects of different training and prediction strategies. We study the following factors: (1) using auxiliary state head; (2) predicting the relative offset between adjacent timestamps, vs. directly regressing the motions at the target timestamp; (3-5) using classification and state estimation results for motion jitter suppression (J.S.). The specification of our final model is listed in (6).

Prediction strategies. Table 6 shows the effects of different training and prediction strategies. First, we see that using auxiliary state-estimation head benefits the model performance greatly. The reason could be that this additional head brings extra supervision to the network learning, as well as helps suppress the background jitters. Second, Table 6 validates the effectiveness of predicting the relative displacement between timestamps, which in practice is able to ease the training of network. Finally, we observe that both classification and state estimation results are helpful in suppressing the jitters significantly, while only sacrificing the accuracies for cells with slow and fast speeds slightly.

References

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016. 3

[2] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992. 5, 8

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 2, 5

[4] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956, 2018. 2, 3

[5] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez, and Carl Wellington. 3d point cloud processing and learning for autonomous driving. *IEEE Signal Processing Magazine, Special Issue on Autonomous Driving*, 2020. 1

[6] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017. 1, 2

[7] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 2

[8] Nachiket Deo and Mohan M Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1468–1476, 2018. 3

[9] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 3

[10] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. 3

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. 1

[12] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3254–3263, 2019. 3, 6, 7

[13] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018. 3

[14] Stefan Hoermann, Martin Bach, and Klaus Dietmayer. Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2056–2063, 2018. 2, 3

[15] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. 3

[16] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. 3

[17] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. 7

[18] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8, 2018. 2

[19] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 1, 2, 8

[20] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision*, pages 734–750, 2018. 1, 2

[21] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017. 3

[22] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal*, 1(1):1, 2014. 1

[23] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019. 1, 2

[24] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision*, pages 641–656, 2018. 1, 2

[25] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 4, 8

[26] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 2

[27] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proceedings*

of the European Conference on Computer Vision, pages 21–37, 2016. 1, 2

[28] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 529–537, 2019. 3, 6, 7

[29] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. *arXiv preprint arXiv:1910.09165*, 2019. 3

[30] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981. 3

[31] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018. 2, 3

[32] Wei-Chiu Ma, De-An Huang, Namhoon Lee, and Kris M Kitani. Forecasting interactive dynamics of pedestrians with fictitious play. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 774–782, 2017. 3

[33] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6120–6127, 2019. 3

[34] Nima Mohajerin and Mohsen Rohani. Multi-step prediction of occupancy grid maps with recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10600–10608, 2019. 2, 3

[35] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. *arXiv preprint arXiv:1904.09664*, 2019. 2

[36] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018. 2

[37] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 3

[38] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 3

[39] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017. 4

[40] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 2

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 2

[42] Nicholas Rhinehart, Kris M Kitani, and Paul Vernaza. R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 772–788, 2018. 3

[43] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1349–1358, 2019. 3

[44] Marcel Schreiber, Stefan Hoermann, and Klaus Dietmayer. Long-term occupancy grid prediction using recurrent neural networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9299–9305, 2019. 2, 3, 6, 7

[45] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pages 527–538, 2018. 5, 7

[46] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019. 1, 2, 6, 7

[47] Shaoshuai Shi, Zhe Wang, Xiaogang Wang, and Hongsheng Li. Part-aˆ 2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *arXiv preprint arXiv:1907.03670*, 2019. 2

[48] Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In *Proceedings of the European Conference on Computer Vision*, pages 197–209, 2018. 2

[49] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. 8

[50] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. *arXiv preprint arXiv:1904.02811*, 2019. 4, 8

[51] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018. 4

[52] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):146, 2019. 3

[53] Pengxiang Wu, Chao Chen, Jingru Yi, and Dimitris Metaxas. Point cloud processing via recurrent set encoding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5441–5449, 2019. 3

[54] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Pro-*

ceedings of the European Conference on Computer Vision, pages 305–321, 2018. 4, 8

[55] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018. 2

[56] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2, 3

[57] Bin Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In *Conference on Robot Learning*, pages 146–155, 2018. 2

[58] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018. 2, 3

[59] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019. 2, 3

[60] Pu Zhang, Wanli Ouyang, Pengfei Zhang, Jianru Xue, and Nanning Zheng. Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12085–12094, 2019. 3

[61] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 4

[62] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12126–12134, 2019. 3

[63] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 1, 2

[64] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 1, 2, 3, 8