

4D-Net for Learned Multi-Modal Alignment

AJ Piergiovanni
Google Research

Vincent Casser
Waymo LLC

Michael S. Ryoo
Robotics at Google

Anelia Angelova
Google Research

Abstract

We present *4D-Net*, a 3D object detection approach, which utilizes 3D Point Cloud and RGB sensing information, both in time. We are able to incorporate the 4D information by performing a novel dynamic connection learning across various feature representations and levels of abstraction, as well as by observing geometric constraints. Our approach outperforms the state-of-the-art and strong baselines on the Waymo Open Dataset. *4D-Net* is better able to use motion cues and dense image information to detect distant objects more successfully. We will open source the code.

1. Introduction

Scene understanding is a long-standing research topic in computer vision. It is especially important to the autonomous driving domain, where a central point of interest is detecting pedestrians, vehicles, obstacles and potential hazards in the environment. While it was traditionally undertaken from a still 2D image, 3D sensing is widely available, and most modern vehicle platforms are equipped with both 3D LiDAR sensors and multiple cameras producing 3D Point Clouds (PC) and RGB frames. Furthermore, autonomous vehicles obtain this information *in time*. Since all sensors are grounded spatially, their data collectively, when looked at in time, can be seen as a 4-dimensional entity. Reasoning across these sensors and time clearly offers opportunities to obtain a more accurate and holistic understanding, instead of the traditional scene understanding from a single 2D still-image or a single 3D Point Cloud.

While all this 4D sensor data is readily available on-board, very few approaches have utilized it. For example, the majority of methods targeting 3D object detection use a single 3D point cloud as an input [17], with numerous approaches proposed [25, 53, 37, 32, 40, 41, 56, 58, 57]. Only more recently has point cloud information been considered in time, with approaches typically accumulating several point clouds over a short time horizon [21, 20, 56, 32].

Furthermore, the sensors have complementary character-

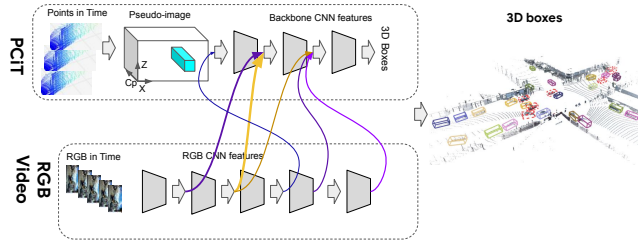


Figure 1. 4D-Net effectively combines 3D sensing in time (PCiT) with RGB data also streamed in time, learning the connections between different sensors and their feature representations.

istics. The point cloud data alone may sometimes be insufficient, e.g., at far ranges where an object only reflects a handful of points, or for very small objects. More information is undoubtedly contained in the RGB data, especially when combined with the 3D Point Cloud inputs. Yet, relatively few works attempted to combine these modalities [35, 51, 19, 24]. Notably, only 2 of the 26 submissions to the Waymo Open Dataset 3D detection challenge operated on both modalities [47]. No methods have attempted combining them when both are streamed in time. The questions of how to align these very different sensor modalities most effectively, as well as how to do so efficiently, have been major roadblocks.

To address these challenges, we propose *4D-Net*, which combines Point Cloud information together with RGB camera data, both in time, in an efficient and learnable manner. We propose a novel learning technique for fusing information in 4D from both sensors, respectively building and learning connections between feature representations from different modalities and levels of abstraction (Figure 1). Using our method, each modality is processed with a suitable architecture producing rich features, which are then aligned and fused at different levels by dynamic connection learning (Figure 2). We show that this is an effective and efficient way of processing 4D information from multiple sensors. 4D-Nets provide unique opportunities as they naturally learn to establish relations between these sensors' features, combining information at various learning stages. This is in contrast to previous late fusion work, which fuse

already mature features that may have lost spatial information, crucial to detecting objects in 3D.

Our results are evaluated on the Waymo Open Dataset [48], a challenging Autonomous Driving dataset and popular 3D detection benchmark. 4D-Net outperforms the state-of-the-art and is competitive in runtime. Importantly, being able to incorporate dense spatial information and information in time improves detection at far ranges and for small and hard to see objects. We present several insights into the respective significance of the different sensors and time horizons, and runtime/accuracy trade-offs.

Our contributions are: (1) the first 4D-Net for object detection which spans the 4-Dimensions, incorporating both point clouds and images in time, (2) a novel learning method which learns to fuse multiple modalities in 4D, (3) a simple and effective sampling technique for 3D Point Clouds in time, (4) a new state-of-the-art for 3D detection on the Waymo Open Dataset and a detailed analysis for unlocking further performance gains.

2. Related Work

Object Detection from RGB. The earliest detection approaches in the context of autonomous driving were primarily focused on camera-based object detection, often drawing heavily from the extensive body of 2D vision work [50, 14, 4, 14, 3, 7, 33, 9, 45, 17, 13], with some more advanced works using deep learning features [2, 12, 44]. Detection with temporal features, i.e., integrating features across several neighboring frames [16, 54], and leveraging kinematic motion to improve detection consistency across time [6] have also been applied. However, looking at the images as videos and processing them with video CNNs is not common.

Object Detection from Point Cloud. Many approaches found it effective to apply well-established 2D detectors on a top-down (BEV) projection of the point cloud (AVOD [24], PIXOR [58], Complex-YOLO [42], HD-NET [57]). This input representation can be advantageous because it makes it easier to separate objects, and object sizes remain constant across different ranges. However, it results in loss of occlusion information, does not effectively exploit the full 3D geometric information, and is inherently sparse. Certain techniques can be used to alleviate these drawbacks, e.g., learning a pseudo-image projection [25].

An alternative to top-down representations is operating directly on the range image, projecting the PC into perspective view. This representation is inherently dense, simplifies occlusion reasoning and has been used in various works (e.g. LaserNet [31], VeloFCN [27], [5]). Another line of work tries to exploit the complementary nature of both by jointly operating on multiple views [60, 53].

Instead of relying on view projections, some methods directly operate on the 3D voxelized point cloud [61, 26, 52].

Voxel resolution can greatly affect performance and is typically limited by computational constraints. To reduce compute, some rely on applying sparse 3D convolutions, such as Vote3Deep [15], Second [56] or PVRCNN [40]. Dynamic voxelization has been proposed in [60].

Other methods operate directly on the raw point cloud data, e.g. SPLATNet [46], StarNet [32] or PointRCNN [41].

Point Clouds in Time (PCiT). Integrating information from multiple point clouds has been proposed recently. StarNet [32] does not explicitly operate in time, but can use high-confidence predictions on previous frames as “temporal context” to seed object center sampling in the following frames. [21] extract features on individual frames and accumulate information in an LSTM over 4 frames for detection. [11] apply 4D ConvNets for spatial-temporal reasoning for AR/VR applications. [56, 20, 59] combine multiple point clouds in time by concatenating them and adding a channel representing their relative timestamps.

Point Clouds and RGB fusion. Acknowledging the merits of sensor fusion, researchers have attempted to combine LiDAR and camera sensing to improve performance [35, 10, 28]. Frustum PointNet [36] first performs image-based 2D detection, and then extrapolates the detection into a 3D frustum based on LiDAR data. Alternatively, one can project the point cloud into the camera view - in its simplest form creating RGB-D input, although alternative depth representations may be used [19, 18]. However, this provides limited scalability as each detection inference would only cover a very limited field of view. Conversely, the point cloud input can be enriched by adding color or semantic features [51, 43, 24]. This, however, comes at the expense of losing spatial density - one of the primary advantages of camera sensors. Several methods have instead applied modality-specific feature extractors, which are then fused downstream [24, 29, 10, 55, 28]. None of the above-mentioned approaches operate on either modality in time.

3. 4D-Net

4D-Net proposes an approach to utilize and fuse multi-sensor information, learning the feature representation from these sensors and their mutual combinations. An overview of the approach is shown in Figure 2. In 4D-Net, we consider the point clouds in time (i.e., a sequence of point clouds) and the RGB information, also in time (a sequence of images). We first describe how to handle the raw 3D point clouds and RGB input information streaming in time (Section 3.1) and then describe our main 4D-Net architectures which learn to combine information from both sensor modalities and across dimensions (Section 3.2). We further offer a multi-stream variant of the 4D-Net which achieves further performance improvements (Section 3.3).

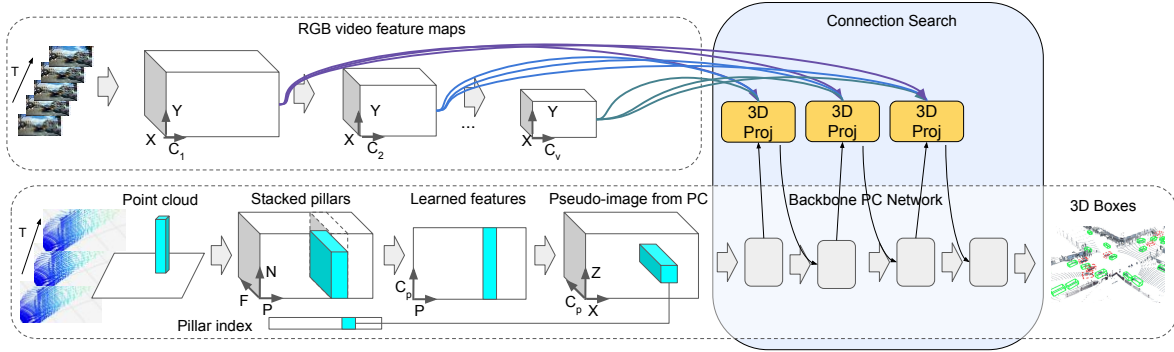


Figure 2. 4D-Net Overview. RGB frames and Point Clouds in time are processed producing features, abstracting some dimensions. A connection search learns where and how to fuse the modalities’ features together. We use 3D projection to align the PC and RGB features.

3.1. 3D Processing and Processing Data in Time

3.1.1 3D Processing

Our approach uses a learnable pre-processor for the point cloud data; it is applied to the 3D points and their features from the LiDAR response to create output features. We chose to use PointPillars [25] to generate these features, but other 3D point ‘featurising’ approaches can be used. PointPillars converts a point cloud into a pseudo-image, which can then be processed by a standard 2D CNN. For clarity, in the derivations below, we will be using a 3D $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$, coordinate system, where the \mathcal{Z} direction is forward (aligned with the car driving), \mathcal{Y} is vertical pointing up and \mathcal{X} is horizontal, i.e., we use a left-hand coordinate system (this is the default system used in the Waymo Open Dataset).

Given a point cloud $\mathcal{P} = \{p\}$ where p is a 3D (x, y, z) point and associated F -dimensional feature vector (e.g., intensity, elongation), the pseudo-image is created as follows. Each point is processed by a linear layer, batch norm and ReLU, to obtain a featurized set of 3D points. The points are grouped into a set of pillars in the \mathcal{X}, \mathcal{Z} plane based on their 3D location and distances between the points. This gives a point cloud representation with shape (P, N, F) , where P is the number of pillars, and N is the maximum number of points per pillar. Each of the P pillars is associated with a x_0, y_0, z_0 location that is the pillar center. The idea is then to further ‘featurize’ information in this (P, N, F) representation and then, using the original coordinates, to ‘distribute’ back the features along the \mathcal{X}, \mathcal{Z} plane and produce a pseudo-image [25], say of size (X, Z, C_P) . Specifically, from (P, N, F) , a feature of size (P, C_P) is obtained via learnable layers and pooling, to then get (X, Z, C_P) . In effect, PointPillar produces a (X, Z, C_P) feature representation from a (X, Y, Z, F) input for a single PC.

3.1.2 Point Clouds in Time

Point clouds and the subsequent feature creation (e.g., as in Section 3.1.1) are computationally expensive and memory intensive operations. Given a sequence of T point clouds,

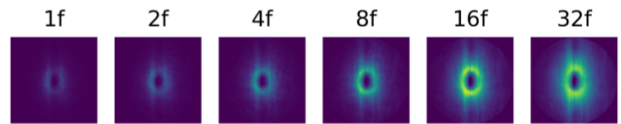


Figure 3. The average point density per voxel illustrates how long-term temporal aggregation, combined with our subsampling strategy, leads to an increased point density, especially at far ranges.

creating T PointPillar ‘pseudo-images’ and then using a 2D or 3D CNN to process all those frames would be prohibitively expensive [59], limiting its usefulness. Previous work [21] explored using sparse convolutions with LSTMs to handle point clouds in time, where a compressed feature representation is fed recursively to next frame representations. Instead, we take a simpler approach similar to [8], which however preserves the original feature representation per 3D point, together with a sense of time.

First, the original feature representation is directly merged in the 3D point cloud, together with a feature to indicate its timestamp. Specifically, we use the vehicle pose to remove the effect of ego-motion and align the point clouds. Next, we add a time indicator t to the feature of each point: $p = [x, y, z, t]$. Then, the PointPillar pseudo-image representation is created as before, which also results in a denser representation. While dynamic motion will obviously create a ghost/halo effect, it can in fact be a very useful signal for learning, and be resolved by the time information. In some circumstances, it is only through motion that distant or poorly discernible objects can be detected.

Point Cloud Subsampling. Importantly, when accumulating points from multiple point clouds, the voxelization step converts all the points into a *fixed-sized* representation based on the grid cell size. This results in a tensor with a fixed size that is padded to N , the maximum number of points. Thus the amount of subsequent compute remains the same, regardless of the number of PCs. If the points exceed N , only N points are randomly sampled ($N=128$ throughout). This has the effect of subsampling the accumulated

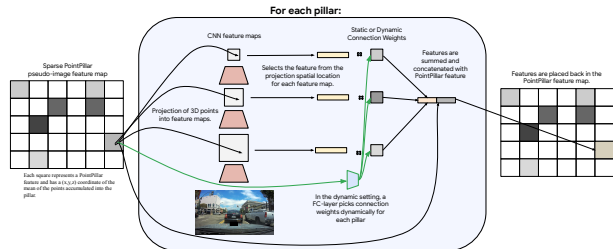


Figure 4. **Connection architecture search:** Each PointPillar feature is projected into the CNN feature map space based on the 3D coordinate of each pillar and the given camera matrices. The feature from the 2D location is extracted from each feature map. Learned static connection weights (grey boxes) then combine these and concatenate them with the pillar feature to create the pillar feature map used as input to the next layer. **Dynamic connections**, shown in green, are produced for each value which are controlled by the features from the point clouds, thus they determine how to fuse in the rest of the features generated by the model at various levels of abstractions. Please see Section 3.2.2.

point cloud, but proportionally more points will be sampled in sparser areas than in dense ones. By densifying the point cloud in sparse regions and sparsifying it in dense regions, we distribute compute more efficiently and provide more signal for long-range detection by increased *point density at far ranges* (e.g. see point density for 16 PCs in Figure 3). We find this representation to be very effective, resulting in significant improvements over using a single point cloud (as seen later in ablation Table 5).

3.1.3 RGB and RGB in Time

While point clouds have become the predominant input modality for 3D tasks, RGB information is very valuable, especially at larger distances (e.g., 30+ or 50+ meters) where objects garner fewer points.

Furthermore, images in time are also highly informative, and complementary to both a still image and PCiT. In fact, for challenging detection cases, motion can be a very powerful clue. While motion can be captured in 3D, a purely PC-based method might miss such signals simply because of the sensing sparsity.

RGB frames, unlike PCs, represent a dense feature containing color pixel information for everything in view. Here we take RGB frames as input and use video CNNs to produce the RGB feature maps. In the video settings, we take T previous frames as input and predict the object bounding boxes in the final frame, same as for point clouds in time. We process a sequence of RGB frames as a video input. Since runtime is of the essence, we use efficient video representations, Tiny Video Networks [34] for processing. More specifically, by a series of layers, some of which working in the spatial dimensions, some temporal, a set of feature representations in the spatial dimension will be learned. As a result, the feature with shape (X, Y, C_V) is produced, abstracting away the time coordinate in input $(X, Y, T, 3)$.

3.2. 4D-Net: Fusing RGB in Time and PC in Time

To combine RGB information into the 3D Point Cloud PointPillar representation (both in time), there are two major considerations: 1) the two sensors need to be geometrically and spatially aligned and 2) the fusion mechanisms of the features produced from these modalities should ideally be learned from the data.

Our 4D-Net entails both projection fusion mechanisms and connectivity search to learn where and how to fuse features (Figure 2). Of note is that both representations have abstracted away some ‘dimensions’ in their features but still contain complementary information: the RGB representation has (X, Y, C_V) , whereas the PCiT has (X, Z, C_P) . Since our end goal is 3D object detection, we chose to fuse from RGB into the point cloud stream, but we note that these approaches could be used to fuse in the other direction as well. Section 3.2.1 and Section 3.2.2, provide details.

3.2.1 3D Projection

To fuse the RGB into the point cloud, we need to (approximately) align the 3D points with 2D image points. To do this, we assume we have calibrated and synchronized sensors and can therefore define accurate projections. Note that the Waymo Open Dataset provides all calibration and synchronized LiDAR and camera data.

The PointPillar pseudo-image M has shape (X, Z, C_P) and is passed through a backbone network with a ResNet-like structure. After each residual block, i , the network feature map M_i has shape (X_i^M, Z_i^M, C_i^M) , where each location (in X and Z) corresponds to a pillar. Each pillar p also has an (x_0, y_0, z_0) coordinate representing its center based on the accumulated 3D points. This provides a 3D coordinate for each of the non-empty feature map locations.

The RGB network also uses a backbone to process the video input. Let us assume that after each block, the network produces a feature map R_i with shape (X_i^R, Y_i^R, C_i^R) , which is a standard image CNN feature map.

Using projections, we can combine the RGB and point cloud data. Specifically, given a 4×4 homogeneous extrinsic camera matrix E (i.e., the camera location and orientation in the world) and a 4×4 homogeneous intrinsic camera matrix K (E and K are part of the dataset), we can project a 3D point $p = (x, y, z, 1)$ to a 2D point $q = K \cdot (E \cdot p)$. For each point pillar location, we obtain the 2D point q , which provides an RGB feature for that point as $R_i[q_x, q_y]$ ¹. This is concatenated to the pillar’s feature, e.g.,

$$M_i[p_x, p_y] = [M_i[p_x, p_y] \parallel R_i[K \cdot (E \cdot p)]] p \in \mathcal{P} \quad (1)$$

Note that LiDAR data typically covers a full 360 degree surround view, while individual cameras typically have a

¹We tried a spatial crop around the point, but found it to be computationally expensive. The CNN’s receptive field also provides spatial context.

quite limited horizontal field of view. To account for this, we only obtain RGB features for points which are captured by one of the cameras. For points outside of the image view, we concatenate a vector of zeros. This approach is easily applied to settings with multiple RGB cameras covering different viewpoints – the same RGB CNN is applied to each view, then the projection is done per-camera, and added together before concatenation.

3.2.2 Connection Architecture Search in 4D

While the above projection will align the two sensors geometrically, it is not immediately obvious what information should be extracted from each and how the sensor features interact for the main task of object detection.

To that end we propose to learn the connections and fusion of these via a light-weight differentiable one-shot architecture search. One-shot differentiable architecture search has been used for strengthening the learned features for image understanding [30] and for video [38, 39]. In our case, we are utilizing it for relating information in 4D, i.e., in 3D and in time, and also connecting related features across different sensing modalities (the RGB-in-time and point cloud-in-time streams). Of note is that we learn the combination of feature representations at various levels of abstraction for both sensors (Figure 2).

In Figure 4, we illustrate how this architecture search works. Given a set of RGB feature maps, $\{R_i | i \in [0, 1, \dots, B]\}$ (B being the total number of blocks/feature maps in the RGB network), we can compute the projection of each pillar into the 2D space and obtain a feature vector. This produces a set of feature vectors $\mathcal{F} = \{f_i | i \in [0, 1, \dots, B]\}$. We then have a learned weight w , which is a B -dimensional vector. We apply softmax and then compute $\sum w \times \mathcal{F}$ to obtain the final feature vector. w learns the connection weights, i.e. which RGB layer to fuse into the PointPillars layer. This is done after each block in the PointPillars network, allowing many connections to be learned (Figure 2).

Dynamic Connections. The above-mentioned mechanism is very powerful as it allows to learn the relations between different levels of feature abstraction and different sources of features (e.g., RGB, PC) (Figure 4). Furthermore, as shown in later sections (Section 3.3) it allows for combining multiple computational towers seamlessly without any additional changes.

However, in the autonomous driving domain it is especially important to reliably detect objects at highly variable distances, with modern LiDAR sensors reaching several hundreds of meters of range [1]. This implies that further-away objects will appear smaller in the images and the most valuable features for detecting them will be in earlier layers, compared to close-by objects. Based on this observation, we modified the connections to be *dynamic*, inspired

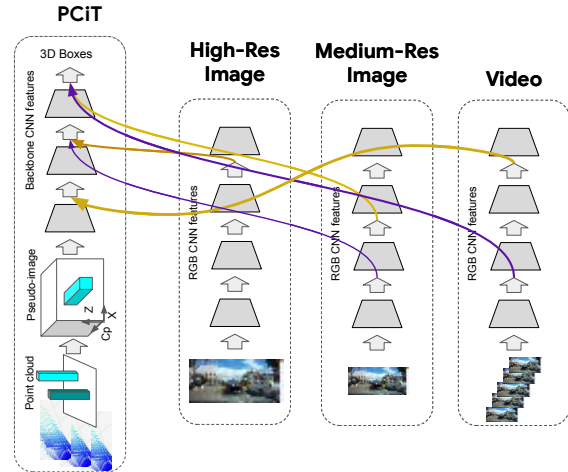


Figure 5. Illustration of a multi-stream 4D-Net. It takes point clouds (in time) and still-image and video as input, computes features for them, learns connection weights between the streams.

by self-attention mechanisms. Specifically, instead of w being a learned weight, we replace w with a linear layer with B outputs, ω , which is applied to the PointPillar feature $M_i[p_x, p_y]$ and generates weights over the B RGB feature maps. ω is followed by a softmax activation function. This allows the network to dynamically select which RGB block to fuse information from, e.g. taking a higher resolution feature from an early layer or a low resolution feature from a later layer (Figure 4). Since this is done for each pillar individually, the network can learn how and where to select these features based on the input.

3.3. Multi-Stream 4D-Net

Multiple RGB streams. Building on the dynamic connection learning, we propose a Multi-Stream (MS) version of 4D-Net. While the 4D-Net itself already learns to combine the information from two streams – the sparse 3D PCiT and camera input – we can have more than one RGB input stream (Figure 5). One advantage of the proposed (dynamic) connection learning between features of different modalities is that it is applicable to many input feature sources and is agnostic to where they originate from. For example, we can add a separate tower for processing high-resolution still images, or an additional video tower using a different backbone or a different temporal resolution. This enables a more rich set of motion features to be learned and surfaced for combination with the PC features. Note that all these are combined with the PC (in time) features in the same dynamic fusion proposed above, thus allowing the PCiT stream to dynamically select the RGB features to fuse from all streams. Similarly, it is also possible to introduce additional PC streams.

Multiple Resolutions. Empirically, we observe that adding an RGB stream benefits recognition of far away ob-

Method	AP L1	AP L2	AP 30m	AP 30-50m	AP 50m+	Runtime
StarNet [32]	53.7	-	-	-	-	-
LaserNet [31]	52.1	-	70.9	52.9	29.6	64ms
PointPillars [25], from [21]	57.2	-	-	-	-	-
MVF [60]	62.9	-	86.3	60.0	36.0	-
Huang et al [21] (4 PCs)	63.6	-	-	-	-	-
PillarMultiView [53]	69.8	-	88.5	66.5	42.9	67ms
PVRCNN [40]	70.3	65.4	91.9	69.2	42.2	300ms
4D-Net (Ours)	73.6	70.6	80.7	74.3	56.8	142 ms (net) + 102 ms (16f voxel)
4D-Net (Ours with Multi-Stream)	74.5	71.2	80.9	74.7	57.6	203 ms (net) + 102 ms (16f voxel)

Table 1. Waymo Open Dataset [47]. 3D detection AP on vehicles @ 0.7 IoU on the validation set. For 4D-Net we report the runtime of the network and of the pre-processing voxelization step for the point clouds in time.



Figure 6. 4D-Net predictions on a scene in the Waymo Open Dataset [47]. Individual instances are shown in different colors. Red boxes indicate errors (dashed lines: FN, solid lines: FP). In this example, all predictions are matching the ground-truth except for a false-negative on the right side (front camera). Note that any misalignments in the camera view are due to projection, not by inaccuracies in the predictions.

jects the most. Distant objects appear smaller than close objects, suggesting that using higher resolution images will further improve recognition. Additionally, adding RGB inputs at two or more different resolutions will increase the diversity of features available for connection learning.

Thus in the multi-stream setting, we combine inputs at different resolutions (see Figure 5 for a schematic). Our main Multi-Stream 4D-Net uses 1) one single still image tower at high image resolution (312x312), 2) one video image tower at lower resolution (192x192) with 16 frames and 3) the PCiT which has aggregated 16 point clouds. The original 4D-Net has the latter two streams only but uses 12 RGB frames. More streams and more resolutions are explored in the ablations, Section 4.2. Similar to our main 4D-Net, we use a lightweight Tiny Video Network [34], so that the multi-stream 4D-Net is efficient at inference.

Implementation Details. We train the model to minimize a standard cross-entropy loss for classification and a L_2 regression loss for the residuals between the anchor boxes and

the GT boxes. $N = 128$ throughout the paper and we use a $224 \times 224 \times 1$ output grid. Further implementation and experimental details are included in the sup. material.

4. Experiments

We conduct experiments on the Waymo Open Dataset [48], following the standard evaluation protocol and evaluation script provided. We report results on vehicles at 0.7 IoU overlap, which is the same setting as in previous work. We conduct ablation studies which demonstrate the various benefits of the proposed approach, and provide analysis with respect to different model and input configurations. Some of the ablation experiments show additional opportunities for improving performance which are not included in the main results, e.g., increasing the number of RGB frames or input resolutions or using stronger backbones.

4.1. Waymo Open Dataset results

Table 1 shows the results of 4D-Net in comparison to the recent state-of-the-art (SOTA) approaches. As seen, it outperforms the previous SOTA, e.g. by **3.3 and 4.2 AP**,

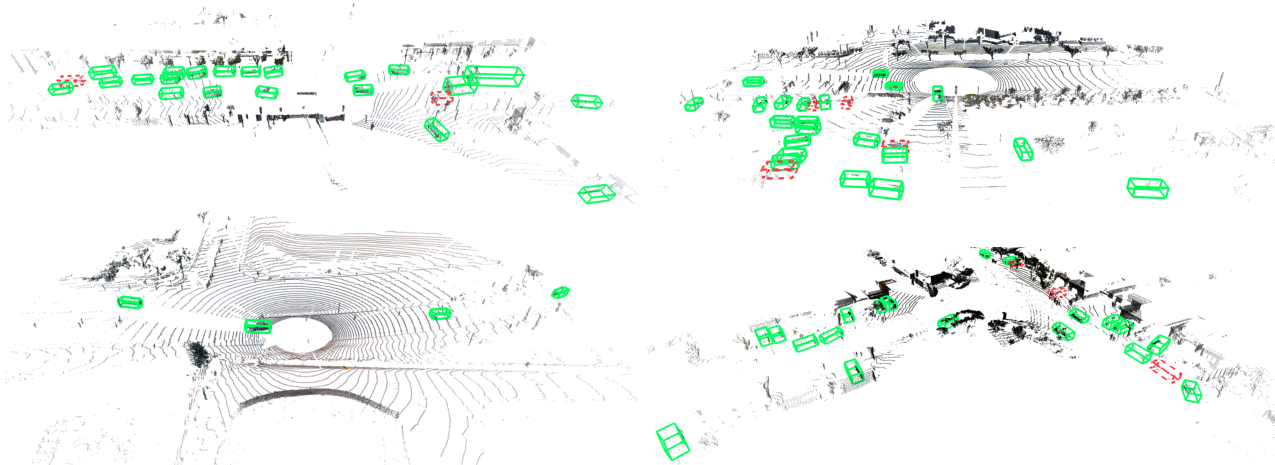


Figure 7. Example results on the Waymo Open Dataset. Green boxes are correct detections, and red boxes indicate errors (dashed lines: FN, solid lines: FP). We observe that the model is more inclined to produce false-negatives, rather than false-positives.

and more importantly, it significantly outperforms when detecting objects at far distances by **14.6 and 15.4 AP**. We also report inference runtimes, split into time to run the 4D-Net (net) with PC accumulated in time and RGB streams, and the time for point cloud pre-processing (voxelization). We observe very competitive runtimes, despite processing much more information than other methods. Qualitative results are shown in Figure 6 and Figure 7.

4.2. Ablation Studies

This section presents the ablation studies. We make best efforts to isolate confounding effects and test components individually, e.g., by removing multiple PCs, or RGB.

Main Contributions and Fusion Approaches. Table 2 shows the main ablation experiments, investigating key components of the 4D-Net. Starting from the main 4D-Net approach (first line), in the top lines we evaluate the approach when the key contributions are removed individually or jointly. As seen, dynamic learning on top of 3D projections is the most beneficial and both are important. At the bottom of the table, we show performance of the approach with different modalities enabled or disabled, for direct comparison. We notice interesting phenomena: using multiple PC in time is definitely helpful, but a single RGB image can boost up the performance of both a single PC and multiple PCs much more significantly, with the proposed projection and connection learning. Similarly, RGB in time can boost a single PC variant significantly, too. Thus a combination of the two sensors, at least one of which is in time is important. The best result comes from multiple RGBs and PC in time, i.e., spanning all 4 dimensions. The supp. material has details of the baselines used in lieu of the proposed components.

Multi-Stream 4D-Net Variants. Table 3 shows the results of using a various number of additional input streams

(Figure 5). As seen, various interesting combinations can be created and learned successfully. Multi-stream models can also afford to reduce the resolution of the video stream inputs and obtain equally powerful models. More multi-stream variants can be explored.

RGB Resolution and Video. In this section we explore the impact of RGB resolution (Table 4²). We observe potential improvements to the 4D-Net that were not used in the main method and can be leveraged in future work. Some improvements are gained at the expense of runtime as seen to the right of the table. As expected, better image resolution helps, particularly for detection of far-away objects, e.g., almost 5 and 12 AP improvements for objects beyond 50m when resolution is increased to 512 from 224 and 192. Another interesting observation is that RGB video helps a lot. For example, increasing resolution from 224 to 512 improves still-image performance by 1% for close objects, but keeping the same 224x224 resolution for a video input gets even higher performance, an improvement by 2.6%. Naturally, higher resolution videos than the ones shown will improve both metrics at additional latency cost.

Leveraging more powerful video methods is also possible, although not particularly worthwhile, as they gain only a little in accuracy, but at high computational cost (see Table 4, bottom). Specifically, we compare two of the most popular video models: the 3D-ResNet [49, 23] and AssembleNet [38]. As seen, they provide more accurate results, especially the powerful AssembleNet, but are very slow.

Point Clouds in Time. Table 5 shows the effect of using point clouds in time. As expected, multiple point clouds improve performance notably. Using 16 PCs, about 1.6 seconds of history, seems to be optimal and also matches observation that density saturates around 16 frames (Figure 3).

²The AP is lower than the main 4D-Net as we use a single PC and no dynamic connections, to show the effects of image or video models.

Method	Components			Modalities		AP L1	AP L2	AP 30m	AP 30-50m	AP 50m+
	Proj	Conn	Dyn	PC	RGB					
4D-Net	✓	✓	✓	PC+T	RGB+T	73.6	70.6	80.7	74.3	56.8
	✓	✓		PC+T	RGB+T	73.1	70.1	80.5	73.9	56.2
	✓			PC+T	RGB+T	72.6	69.7	79.6	72.8	54.6
				PC+T	RGB+T	62.5	58.9	70.1	57.8	42.5
	✓	✓		PC+T	RGB	72.6	69.7	79.8	73.6	55.8
	✓	✓		PC	RGB+T	65.4	63.9	77.5	67.4	48.5
	✓	✓		PC	RGB	64.3	63.0	74.9	65.1	47.2
	✓			PC	RGB	62.5	61.5	71.5	61.5	41.0
		✓		PC	RGB	56.7	53.6	66.2	52.5	37.5
				PC+T		60.5	55.7	68.4	57.6	38.1
				PC		55.7	52.8	65.0	51.3	35.4

Table 2. Ablation results for the 4D-Net. From the full 4D-Net variant, components are removed one at a time, to demonstrate their effect. Proj is the proposed projection method, Conn is the connectivity search and Dyn is the dynamic connection method. We also ablate with using single PC or single RGB input. PC+T denotes Point Clouds in Time, RGB+T is RGB frames in time. 16 PCiT and 16 RGB of 224x224 size are used (except the top two with 12 RGB of 192x192). This is a single 4D-Net. See Table 3 for multi-stream 4D-Nets.

Method	AP	30m	30-50m	50m+	Runtime
4D-Net (192x192 12f video)	73.6	80.7	74.3	56.8	142 ms
4D-Net MS (192x192 16f video + 312x312 image)	74.5	80.9	74.7	57.6	203 ms
4D-Net MS-1 (192x192 16f video + 224x224 image)	73.4	81.2	72.5	56.5	162 ms
4D-Net MS-2 (224x224 16f video + 224x224 image)	73.8	80.5	73.7	56.9	171 ms
4D-Net MS-3 (128x128 16f video + 192x192 image + 312x312 image)	74.2	81.5	72.9	57.8	225 ms

Table 3. Ablation results for Multi-Streams (MS) models. AP shown. The top portion shows the main 4D-Net and the Multi-Stream version from Table 1. MS-1 and MS-2 include an additional image stream but at different resolutions. MS-3 has two additional image streams. It shows one can significantly reduce the input video resolution achieving top results. All video models use 16 frames except 4D-Net which has 12. Voxel pre-processing is not included in runtime as in Table 1.

Image resolutions	AP	30m	30-50m	50m+	Time
192x192 1 fr.	60.8	73.6	60.7	40.4	82
224x224 1 fr.	64.3	74.9	65.1	47.2	97
312x312 1 fr.	67.3	75.7	66.4	49.5	142
512x512 1 fr.	68.2	75.9	67.5	52.4	297
192x192 12-fr.	64.2	75.2	65.3	46.2	109
224x224 16-fr.	65.4	77.5	67.4	48.5	115
224x224 3DRes	66.4	77.8	68.6	49.5	254
224x224 Assm	66.8	79.1	69.2	50.7	502

Table 4. Ablations for input image resolutions for a single frame RGB tower and a single point cloud. AP (L1) shown. For comparison, models with 12-frame and 16-frame video input are given, as well as, stronger but much slower methods 3DResNet [49] and AssembleNet [38], both with 32 frames. All are with a single point cloud, which reduces compute, as well. Time is in ms.

5. Conclusions and Future Work

We present 4D-Net, which proposes a new approach to combine underutilized RGB streams with Point-Cloud-in-time information. We demonstrate improved state-of-the-art performance and competitive inference runtimes, despite

Number of PC	AP	AP 30m	AP 30-50m	AP 50m+
1 PC	55.7	65.0	51.3	35.4
2 PC	56.3	66.1	52.5	36.4
4 PC	57.8	66.9	54.6	36.7
8 PC	59.4	67.6	56.4	37.8
16 PC	60.5	68.4	57.6	38.1
32 PC	60.3	67.4	56.4	38.4

Table 5. Ablations for Point Clouds (PC) in time. No RGB inputs are used. One PC (top row) is effectively the PointPillar model.

using 4D sensing and both modalities in time. Without loss of generality, the same approach can be extended to other streams of RGB images, e.g., the side cameras providing critical information for highly occluded objects, or to diverse learnable feature representations for PC or images, or to other sensors. While this work is demonstrated for the challenging problem of aligning different sensors for autonomous driving which span the 4D, the proposed approach can be used for various related modalities which capture different aspects of the same domain: aligning audio and video data or text and imagery.

References

- [1] Introducing the 5th-generation Waymo Driver: Informed by experience, designed for scale, engineered to tackle more environments. In <https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html>, 2020. 5
- [2] Anelia Angelova, Alex Krizhevsky, Vincent Vanhoucke, Abhijit Ogale, and Dave Ferguson. Real-time pedestrian detection with deep network cascade. In *British Machine Vision Conference*, 2015. 2
- [3] Mathias M. Timofte R. Van Gool L. Benenson, R. Pedestrian detection at 100 frames per second. In *CVPR*, 2012. 2
- [4] Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. Ten years of pedestrian detection, what have we learned? In *ECCV Workshop on Computer Vision for Road Scene Understanding Autonomous Driving*, 2014. 2
- [5] Alex Bewley, Pei Sun, Thomas Mensink, Dragomir Anguelov, and Cristian Sminchisescu. Range conditioned dilated convolutions for scale invariant 3d object detection. *CoRL*, 2020. 2
- [6] Garrick Brazil, Gerard Pons-Moll, Xiaoming Liu, and Bernt Schiele. Kinematic 3d object detection in monocular video. In *European Conference on Computer Vision*, pages 135–152. Springer, 2020. 2
- [7] Garrick Brazil, Xi Yin, and Xiaoming Liu. Illuminating pedestrians via simultaneous detection and segmentation. In *ICCV*, 2017. 2
- [8] 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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 3
- [9] Zhaowei Cai, Quanfu Fan, Rogerio S. Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, 2016. 2
- [10] 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. 2
- [11] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 2
- [12] Xuangeng Chu, Anlin Zheng, Xiangyu Zhang, and Jian Sun. Detection in crowded scenes: One proposal, multiple predictions. In *CVPR*, 2020. 2
- [13] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2
- [14] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *CVPR*, 2009. 2
- [15] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1355–1361. IEEE, 2017. 2
- [16] Markus Enzweiler and Darius M. Gavrilă. A multilevel mixture-of experts framework for pedestrian classification. In *IEEE Transactions on Image Processing*, 2011. 2
- [17] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 1, 2
- [18] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 2
- [19] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European conference on computer vision*, pages 345–360. Springer, 2014. 1, 2
- [20] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11001–11009, 2020. 1, 2
- [21] Rui Huang, Wanyue Zhang, Abhijit Kundu, Caroline Pantofaru, David A Ross, Thomas Funkhouser, and Alireza Fathi. An LSTM approach to temporal 3d object detection in lidar point clouds. In *European Conference on Computer Vision*, 2020. 1, 2, 3, 6
- [22] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *arXiv preprint arXiv:1506.02025*, 2015. 12
- [23] Yutaka Satoh Kensho Hara, Hirokatsu Kataoka. Learning spatio-temporal features with 3d residual networks for action recognition. In *ICCV ChaLearn Looking at People Workshop*, 2017. 7
- [24] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018. 1, 2
- [25] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, and Oscar Beijbom Jiong Yang. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 1, 2, 3, 6
- [26] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. In *IROS*, 2017. 2
- [27] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. *Proceedings of Robotics: Science and Systems (RSS)*, 2016. 2
- [28] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, 2019. 2
- [29] 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 (ECCV)*, pages 641–656, 2018. 2

- [30] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019. 5
- [31] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *CVPR*, 2019. 2, 6
- [32] Jiquan Ngiam, Benjamin Caine, Wei Han, Brandon Yang, Yuning Chai, Pei Sun, Yin Zhou, Xi Yi, Ouais Alsharif, Patrick Nguyen, Zhifeng Chen, Jonathon Shlens, and Vijay Vasudevan. Starnet: Targeted computation for object detection in point clouds. In *CoRR:1908.11069*, 2019. 1, 2, 6
- [33] Wanli Ouyang and Xiaogang Wang. A discriminative deep model for pedestrian detection with occlusion handling. In *CVPR*, 2012. 2
- [34] AJ Piergiovanni, Anelia Angelova, and Michael S Ryoo. Tiny video networks: Architecture search for efficient video models. In *ICML Workshop on Automated Machine Learning (AutoML)*, 2020. 4, 6
- [35] Cristiano Premevida, Joao Carreira, Jorge Batista, and Urbano Nunes. Pedestrian detection combining rgb and dense lidar data. In *IROS*, 2014. 1, 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. *CVPR*, 2017. 1
- [38] Michael S. Ryoo, AJ Piergiovanni, , Mingxing Tan, and Anelia Angelova. AssembleNet: Searching for multi-stream neural connectivity in video architectures. In *ICLR*, 2020. 5, 7, 8
- [39] Michael S. Ryoo, AJ Piergiovanni, Juhana Kangaspona, and Anelia Angelova. AssembleNet++: Assembling modality representations via attention connections. In *European Conference on Computer Vision*, 2020. 5
- [40] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, and Hongsheng Li Xiaogang Wang. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *CVPR*, 2020. 1, 2, 6
- [41] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA*, pages 16–20, 2019. 1, 2
- [42] Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross. Complex-yolo: Real-time 3d object detection on point clouds. In *CVPR*, 2018. 2
- [43] Shuran Song and Jianxiong Xiao. Sliding shapes for 3d object detection in depth images. In *European conference on computer vision*, pages 634–651. Springer, 2014. 2
- [44] Xiaolin Song, Kaili Zhao, Wen-Sheng Chu, Honggang Zhang, and Jun Guo. Progressive refinement network for occluded pedestrian detection. In *European Conference on Computer Vision*, 2020. 2
- [45] Russell Stewart, Mykhaylo Andriluka, and Andrew Y. Ng. End-to-end people detection in crowded scenes. In *CVPR*, 2016. 2
- [46] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *CVPR*, 2018. 2
- [47] Pei Sun, Henrik Kretschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 1, 6, 13
- [48] Pei Sun, Henrik Kretschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Etinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 2, 6
- [49] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, pages 6450–6459, 2018. 7, 8
- [50] Paul Viola, Michael Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. In *CVPR*, 2003. 2
- [51] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4604–4612, 2020. 1, 2
- [52] Dominic Wang and Ingmar Posner. Voting for voting in online point cloud object detection. In *In Proceedings of Robotics: Science and Systems*, 2015. 2
- [53] Yue Wang, Alireza Fathi, Abhijit Kundu, David Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar based object detection for autonomous driving. In *European Conference on Computer Vision*, 2020. 1, 2, 6
- [54] Jialian Wu, Chunluan Zhou, Ming Yang, Qian Zhang, Yuan Li, and Junsong Yuan. Temporal-context enhanced detection of heavily occluded pedestrians. In *CVPR*, 2020. 2
- [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. 1, 2
- [57] Bin Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In *Conference on Robot Learning (CoRL)*, 2018. 1, 2
- [58] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018. 1, 2
- [59] Zhishuai Zhang, Jiyang Gao, Junhua Mao, Yukai Liu, Dragomir Anguelov, and Congcong Li. Stinet: Spatio-temporal-interactive network for pedestrian detection and trajectory prediction. In *Proceedings of the IEEE/CVF Con-*

ference on Computer Vision and Pattern Recognition, pages 11346–11355, 2020. [2](#), [3](#)

- [60] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *Conference on Robot Learning (CoRL)*, 2019. [2](#), [6](#)
- [61] 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. [2](#)

Block	Conv Size	Channels	Repeat	Output Size
Input	-	-	-	$16 \times 224 \times 224$
Block 1	$1 \times 3 \times 3 + 3 \times 1 \times 1$ conv	32	1	$8 \times 112 \times 112$
Block 2	$1 \times 3 \times 3 + 3 \times 1 \times 1$ conv	64	1	$4 \times 56 \times 56$
Block 3	$1 \times 3 \times 3$ conv	128	4	$2 \times 28 \times 28$
Block 4	$1 \times 3 \times 3$ conv	256	4	$2 \times 14 \times 14$

Table 6. RGB Video Network structure used in 4D-Net. Note that the sizes are shown assuming 16 frames at 224×224 input size. For networks that used smaller inputs, the output sizes are each step would be smaller, following the same scaling. Average pooling was used after the convolution to reduce the spatial size. The first two blocks apply both spatial and temporal convolutions in that order.

6. Implementation Details

The models were implemented in TensorFlow. We trained for 120 000 iterations using a batch size of 256, split across 8 devices. The learning rate was set to 0.0015 using a linear warmup for 6 000 steps followed by a cosine decay schedule.

The anchor boxes had size of $[4.7, 2.1, 1.7]$, and 2 rotations (0 and 45 degrees) used at each feature map location. For the PointPillar pseudo-image creation, we used a grid size of $(224, 224, 1)$, an x-range of $(-74.88, 74.88)$ and the same for y-range. The z-range was $(-5, 5)$. The max number of points per cell, $N = 128$. We used 10,000 pillars.

We applied data augmentation to the point clouds (random 3D rotations and flips). The camera matrices were also updated based on the augmentations so the projections would still apply. No augmentation was used on the RGB streams.

The PointPillars network used a feature dim of 64 for the input points. The created pseudo-image had 224×224 shape. This was followed by 3 convolutional blocks with 4, 6, and 6 repeats. Each block consisted of a convolution, batch norm and ReLU activation. This was followed by 3 deconvolutional layers which generate the predictions.

The RGB single frame network is a standard ResNet-18. The video network is based on TinyVideoNetworks. Specifically, we use a network that consists of 6 Residual blocks, the structure is outlined in Table 6. Note that the first two blocks apply both spatial and temporal convolutions to the data (in that order).

We trained on the Waymo Open Dataset, which consists of 1 950 segments that are each 20 seconds long (about 200 frames), a total of 390 000 frames. The LiDAR data is processed into point clouds grouped by timestamp and aligned with the RGB frames. In most experiments, we take sequences of 16 frames as input and predict 3D boxes for the last frame. We evaluate using the provided Waymo metrics library. Before NMS, we filter out boxes with probability less than 0.4 and boxes larger than 30m in length and 5m in width and boxes smaller than 0.5m in length and width.

Method	AP L1	AP L2	AP 30m	AP 30-50m	AP 50m+
Base PointPillars	55.7	52.8	65.0	51.3	35.4
RGB	56.7	53.6	66.2	52.5	37.5
Spatial Avg	58.9	57.6	69.7	56.1	39.6
Spatial Transformer	59.5	58.2	70.0	54.7	43.1
Projection	64.3	63.0	74.9	65.1	47.2

Table 7. Comparison of different spatial fusion methods.

7. Additional experimental results

Fusion Methods In the main paper, we focused on projection as the main method to fuse RGB and point cloud data. Here, we also compare to several other methods. The results are shown in Table 7.

Basic RGB. Here we flatten the RGB image feature output into a 1-D tensor, and concatenate it to the point cloud feature. This loses all spatial information and essentially puts the entire image into each point.

Spatial Avg Pooling. As another baseline, we apply average spatial pooling to the image-based features R_i , obtaining a F_i^R -dimensional feature vector. We then concatenate this to each feature in the PC-based features M_i , resulting in a $(X_i^M, Z_i^M, F_i^M + F_i^R)$ feature map. This is then passed through the remaining CNN for classification. This provides the point cloud stream with some RGB information, but it has no spatial information.

Spatial Transformer. We also tried using a spatial transformer [22] to crop regions around each projected point to append to the PC feature. However, despite small improvements, we found this to be extremely slow due to taking many spatial crops with the transformer.

These baselines are compared with the proposed *Projection* method which is in Table 2 of the main paper. The experiments are conducted in the same conditions as the Projection method in the main paper. The *Basic PointPillars* (also in Table 2 of the main paper) does not have an RGB input and is included for reference only.

8. Additional Visualizations

In Figure 8 we show more visualizations of the predictions of the 4D-Net on the Waymo Open Dataset.

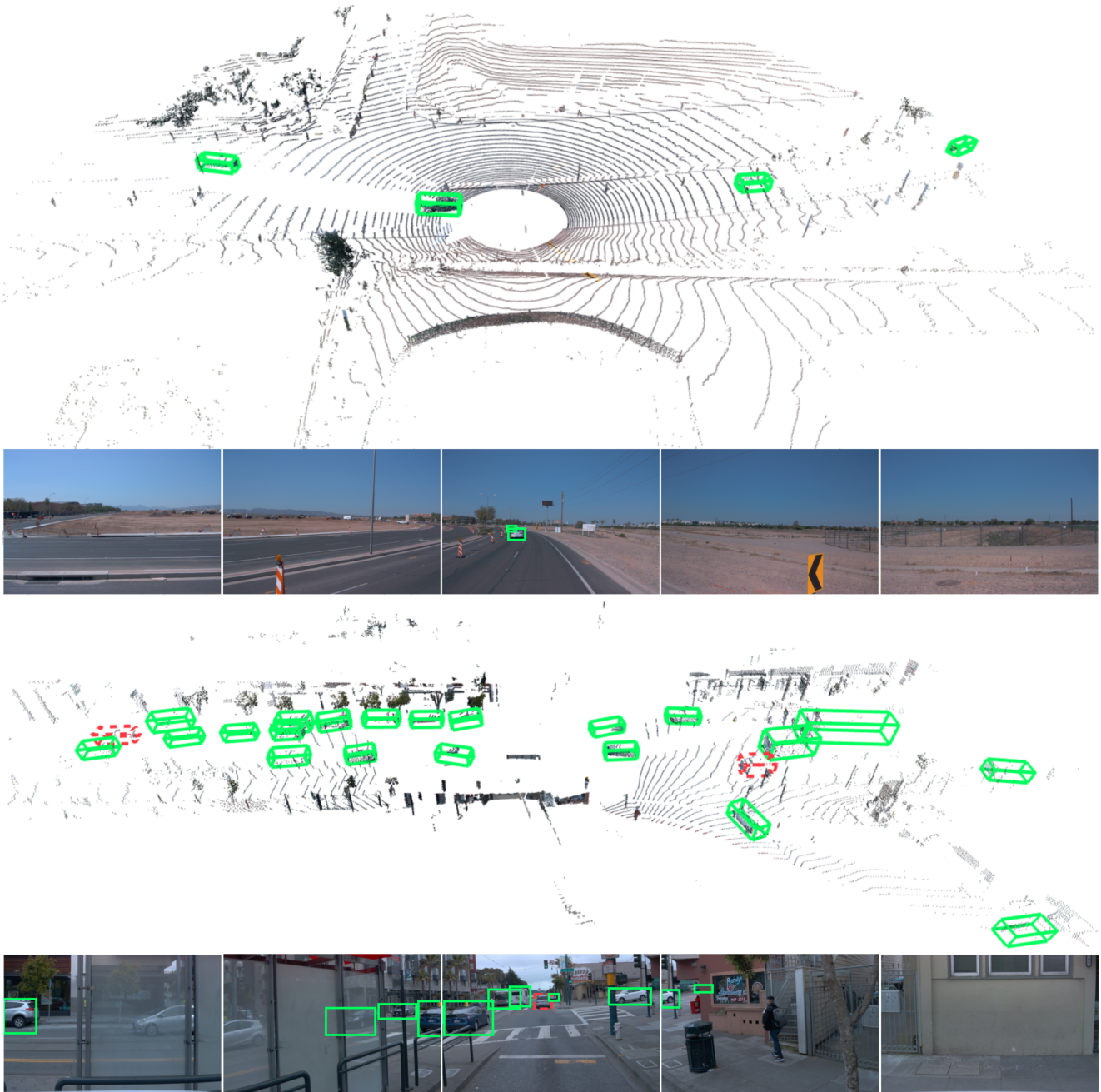


Figure 8. 4D-Net predictions on a scene in the Waymo Open Dataset [47]. Individual instances are shown in green (here) or in different colors in the figures below. Red boxes indicate errors (dashed lines: FN, solid lines: FP). The front camera (central image) is the only one used in our work presently, the others are included for visualization purposes. Note that any misalignments in the camera view are due to projection, not by inaccuracies in the predictions.

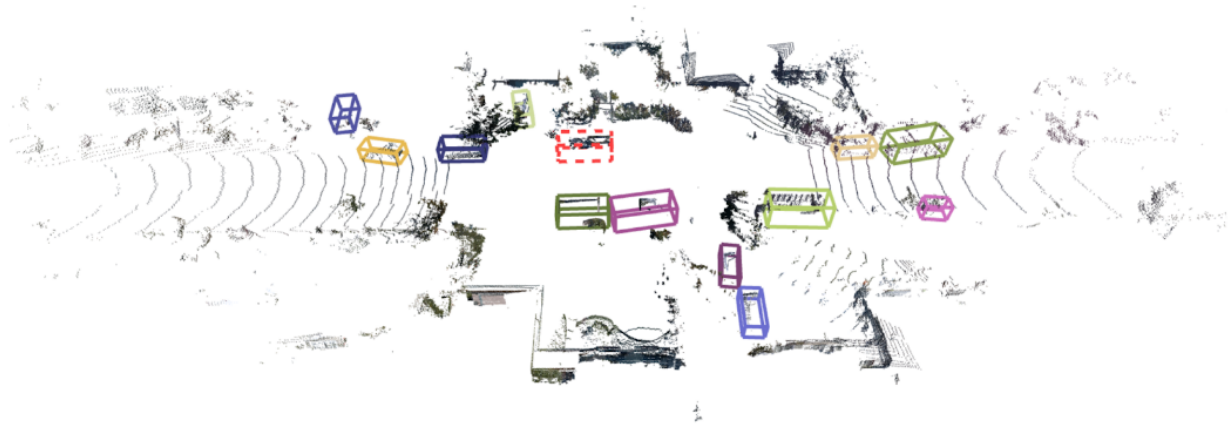
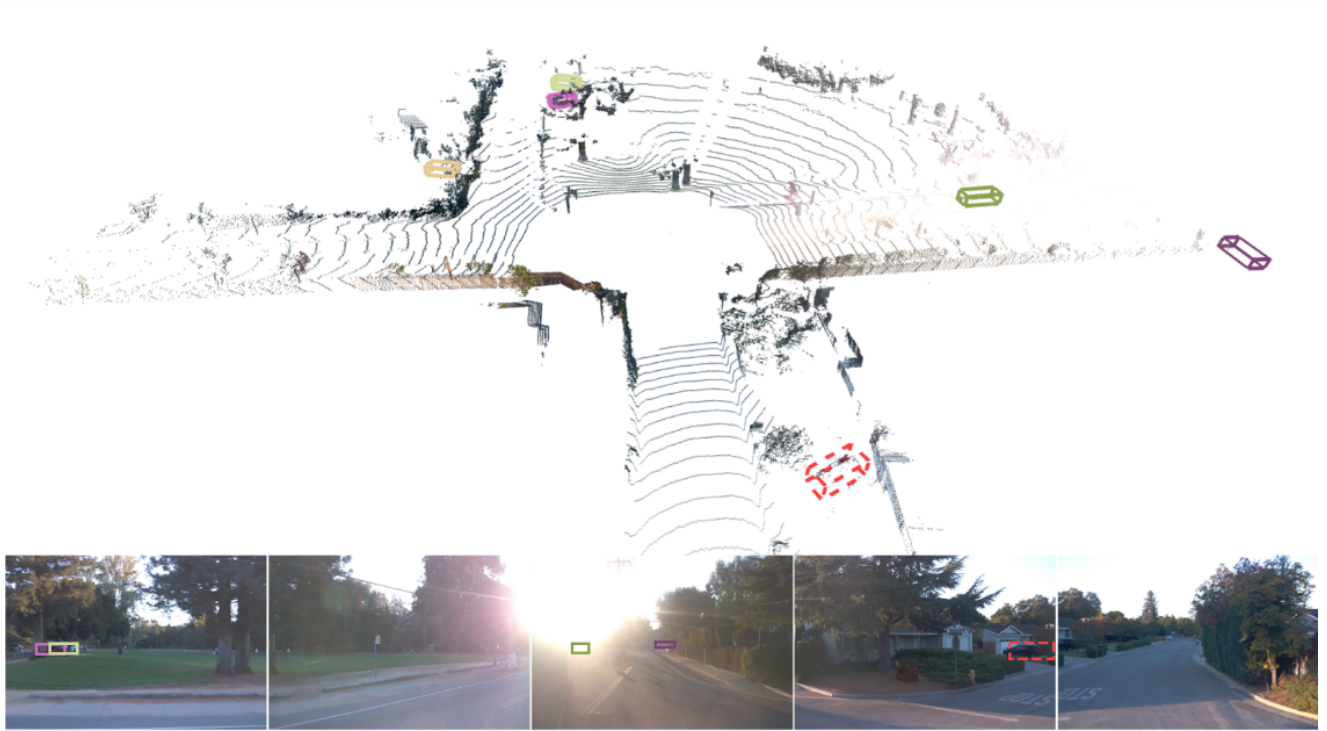




Figure 9. Failure cases examples: In these two challenging cases, the central image (stream) is not sufficient and a vehicle is missed.