V2VNet: Vehicle-to-Vehicle Communication for Joint Perception and Prediction

Tsun-Hsuan Wang[1], Sivabalan Manivasagam[1,2](✉), Ming Liang[1], Bin Yang[1,2], Wenyuan Zeng[1,2], and Raquel Urtasun[1,2]

[1] UberATG, Pittsburgh, USA
[2] University of Toronto, Toronto, Canada
{tsunhsuan.wang,manivasagam,ming.liang,byang,wenyuan,urtasun}@uber.com

Abstract. In this paper, we explore the use of vehicle-to-vehicle (V2V) communication to improve the perception and motion forecasting performance of self-driving vehicles. By intelligently aggregating the information received from multiple nearby vehicles, we can observe the same scene from different viewpoints. This allows us to see through occlusions and detect actors at long range, where the observations are very sparse or non-existent. We also show that our approach of sending compressed deep feature map activations achieves high accuracy while satisfying communication bandwidth requirements.

Keywords: Autonomous driving · Object detection · Motion forecast

1 Introduction

While a world densely populated with self-driving vehicles (SDVs) might seem futuristic, these vehicles will one day soon be the norm. They will provide safer, cheaper and less congested transportation solutions for everyone, everywhere. A core component of self-driving vehicles is their ability to perceive the world. From sensor data, the SDV needs to reason about the scene in 3D, identify the other agents, and forecast how their futures might play out. These tasks are commonly referred to as perception and motion forecasting. Both strong perception and motion forecasting are critical for the SDV to plan and maneuver through traffic to get from one point to another safely.

The reliability of perception and motion forecasting algorithms has significantly improved in the past few years due to the development of neural network architectures that can reason in 3D and intelligently fuse multi-sensor data (e.g., images, LiDAR, maps) [28,29]. Motion forecasting algorithm performance has been further improved by building good multimodal distributions [4,6,12,19] that capture diverse actor behaviour and by modelling actor

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58536-5_36) contains supplementary material, which is available to authorized users.

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12347, pp. 605–621, 2020.
https://doi.org/10.1007/978-3-030-58536-5_36

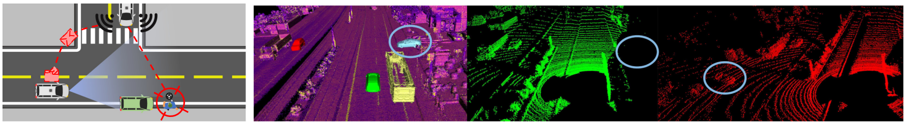

Fig. 1. *Left:* Safety critical scenario of a pedestrian coming out of occlusion. V2V communication can be leveraged to use the fact that multiple self-driving vehicles see the scene from different viewpoints, and thus see through occluders. *Right:* Example *V2VSim* Scene. Virtual scene with occluded actor (blue) and SDVs (red and green), Rendered LiDAR from each SDV in the scene. (Color figure online)

interactions [3,25,36,37]. Recently, [5,31] propose approaches that perform joint perception and motion forecasting, dubbed *perception and prediction* (P&P), further increasing the accuracy while being computationally more efficient than classical two-step pipelines.

Despite these advances, challenges remain. For example, objects that are heavily occluded or far away result in sparse observations and pose a challenge for modern computer vision systems. Failing to detect and predict the intention of these hard-to-see actors might have catastrophic consequences in safety critical situations when there are only a few miliseconds to react: imagine the SDV driving along a road and a child chasing after a soccer ball runs into the street from behind a parked car (Fig. 1, left). This situation is difficult for both SDVs and human drivers to correctly perceive and adjust for. The crux of the problem is that the SDV and the human can only see the scene from a single viewpoint.

However, SDVs could have super-human capabilities if we equip them with the ability to transmit information and utilize the information received from nearby vehicles to better perceive the world. Then the SDV could see behind the occlusion and detect the child earlier, allowing for a safer avoidance maneuver.

In this paper, we consider the *vehicle-to-vehicle* (V2V) communication setting, where each vehicle can broadcast and receive information to/from nearby vehicles (within a 70 m radius). Note that this broadcast range is realistic based on existing communication protocols [21]. We show that to achieve the best compromise of having strong perception and motion forecasting performance while also satisfying existing hardware transmission bandwidth capabilities, we should send compressed intermediate representations of the P&P neural network. Thus, we derive a novel P&P model, called *V2VNet*, which utilizes a spatially aware graph neural network (GNN) to aggregate the information received from all the nearby SDVs, allowing us to intelligently combine information from different points in time and viewpoints in the scene.

To evaluate our approach, we require a dataset where multiple self-driving vehicles are in the same local traffic scene. Unfortunately, no such dataset exists. Therefore, our second contribution is a new dataset, dubbed *V2V-Sim* (see Fig. 1, right) that mimics the setting where there are multiple SDVs driving in the area. Towards this goal, we use a high-fidelity LiDAR simulator [33], which uses a large catalog of static 3D scenes and dynamic objects built from real-world data, to

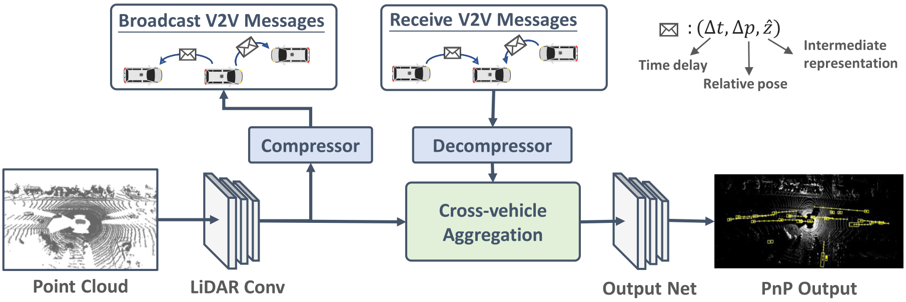

Fig. 2. Overview of V2VNet.

simulate realistic LiDAR point clouds for a given traffic scene. With this simulator, we can recreate traffic scenarios recorded from the real-world and simulate them as if a percentage of the vehicles are SDVs in the network. We show that V2VNet and other V2V methods significantly boosts performance relative to the single vehicle system, and that our compressed intermediate representations reduce bandwidth requirements without sacrificing performance. We hope this work brings attention to the potential benefits of the V2V setting for bringing safer autonomous vehicles on the road. To enable this, we plan to release this new dataset and make a challenge with a leaderboard and evaluation server.

2 Related Work

Joint Perception and Prediction: Detection and motion forecasting play a crucial role in any autonomous driving system. [3–5,25,30,31] unified 3D detection and motion forecasting for self-driving, gaining two key benefits: (1) Sharing computation of both tasks achieves efficient memory usage and fast inference time. (2) Jointly reasoning about detection and motion forecasting improves accuracy and robustness. We build upon these existing P&P models by incorporating V2V communication to share information from different SDVs, enhancing detection and motion forecasting.

Vehicle-to-Vehicle Perception: For the perception task, prior work has utilized messages encoding three types of data: raw sensor data, output detections, or metadata messages that contain vehicle information such as location, heading and speed. [34,38] associate the received V2V messages with outputs of local sensors. [8] aggregate LiDAR point clouds from other vehicles, followed by a deep network for detection. [35,44] process sensor measurements via a deep network and then generate perception outputs for cross-vehicle data sharing. In contrast, we leverage the power of deep networks by transmitting a compressed intermediate representation. Furthermore, while previous works demonstrate results on a limited number of simple and unrealistic scenarios, we showcase the effectiveness of our model on a diverse large-scale self-driving V2V dataset.

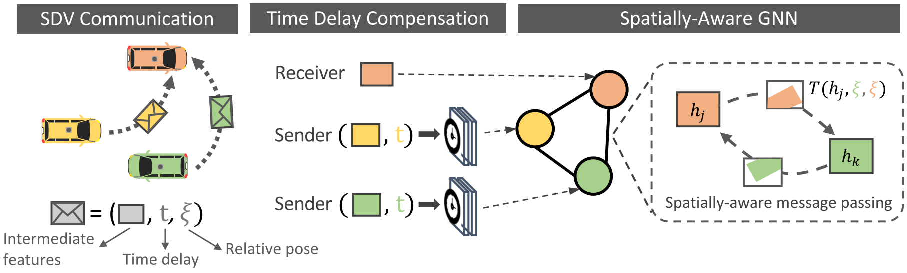

Fig. 3. After SDVs communicate messages, each receiver SDV compensates for time-delay of the received messages, and a GNN aggregates the spatial messages to compute the final intermediate representation.

Aggregation of Multiple Beliefs: In V2V setting, the receiver vehicle should collect and aggregate information from an arbitrary number of sender vehicles for downstream inference. A straightforward approach is to perform permutation invariant operations such as pooling [10, 40] over features from different vehicles. However, this strategy ignores cross-vehicle relations (spatial locations, headings, times) and fails to jointly reason about features from the sender and receiver. On the other hand, recent work on graph neural networks (GNNs) has shown success on processing graph-structured data [15, 18, 26, 46]. MPNN [17] abstract commonalities of GNNs with a message passing framework. GGNN [27] introduce a gating mechanism for node update in the propagation step. Graph-neural networks have also be effective in self-driving: [3, 25] propose a spatially-aware GNN and an interaction transformer to model the interactions between actors in self-driving scenes. [41] uses GNNs to estimate value functions of map nodes and share vehicle information for coordinated route planning. We believe GNNs are tailored for V2V communication, as each vehicle can be a node in the graph. V2VNet leverages GNNs to aggregate and combine messages from other vehicles.

Active Perception: In V2V perception, the receiving vehicle should aggregate information from different viewpoints such that its field of view is maximized, trusting more the view that can see better. Our work is related to a long line of work in active perception, which focuses on deciding what action the agent should take to better perceive the environment. Active perception has been effective in localization and mapping [13, 22], vision-based navigation [14], serving as a learning signal [20, 48], and various other robotics applications [9]. In this work, rather than actively steering SDVs to obtain better viewpoint and sending information to the others, we consider a more realistic scenario where multiple SDVs have their own routes but are currently in the same geographical area, allowing the SDVs to see better by sharing perception messages.

3 Perceiving the World by Leveraging Multiple Vehicles

In this paper, we design a novel perception and motion forecasting model that enables the self-driving vehicle to leverage the fact that several SDVs may be present in the same geographic area. Following the success of joint perception and prediction algorithms [3,5,30,31], which we call P&P, we design our approach as a joint architecture to perform both tasks, which is enhanced to incorporate information received from other vehicles. Specifically, we would like to devise our P&P model to do the following: given sensor data the SDV should (1) process this data, (2) broadcast it, (3) incorporate information received from other nearby SDVs, and then (4) generate final estimates of where all traffic participants are in the 3D space and their predicted future trajectories.

Two key questions arise in the V2V setting: (i) what information should each vehicle broadcast to retain all the important information while minimizing the transmission bandwidth required? (ii) how should each vehicle incorporate the information received from other vehicles to increase the accuracy of its perception and motion forecasting outputs? In this section we address these two questions.

3.1 Which Information Should Be Transmitted

An SDV can choose to broadcast three types of information: (i) the raw sensor data, (ii) the intermediate representations of its P&P system, or (iii) the output detections and motion forecast trajectories. While all three message types are valuable for improving performance, we would like to minimize the message sizes while maximizing P&P accuracy gains. Note that small message sizes are critical because we want to leverage cheap, low-bandwidth, decentralized communication devices. While sending raw measurements minimizes information loss, they require more bandwidth. Furthermore, the receiving vehicle would need to process all additional sensor data received, which might prevent it from meeting the real-time inference requirements. On the other hand, transmitting the outputs of the P&P system is very good in terms of bandwidth, as only a few numbers need to be broadcasted. However, we may lose valuable scene context and uncertainty information that could be very important to better fuse the information.

In this paper, we argue that sending intermediate representations of the P&P network achieves the best of both worlds. First, each vehicle processes its own sensor data and computes its intermediate feature representation. This is compressed and broadcasted to nearby SDVs. Then, each SDV's intermediate representation is updated using the received messages from other SDVs. This is further processed through additional network layers to produce the final perception and motion forecasting outputs. This approach has two advantages: (1) Intermediate representations in deep networks can be easily compressed [11,43], while retaining important information for downstream tasks. (2) It has low computation overhead, as the sensor data from other vehicles has already been pre-processed.

In the following, we first showcase how to compute the intermediate representations and how to compress them. We then show how each vehicle should incorporate the received information to increase the accuracy of its P&P outputs.

Algorithm 1. Cross-vehicle Aggregation

```
input: representation $\hat{z}_i$, relative pose $\Delta p_i$, and time delay $\Delta t_{i \to k}$ for each SDV $i$
for each vehicle $i$ do
    $h_i^{(0)} = CNN(\hat{z}_i, \Delta t_{i \to k}) \parallel \mathbf{0}$    ▷ Compensate time delay, init. node state
end for
for $l$ iterations do    ▷ Message passing
    for each vehicle $i$ do    ▷ Processed in parallel
        $m_{i \to k}^{(l)} = CNN(T(h_i^{(l)}, \xi_{i \to k}), h_k^{(l)}) \cdot M_{i \to k}$    ▷ Spatially transform message
        $h_i^{(l+1)} = ConvGRU(h_i^{(l)}, \phi_M([\forall_{j \in N(i)}, m_{j \to i}^{(l)}]))$    ▷ Node state update
    end for
end for
$z_i^{(L)} = MLP(h_i^{(L)})$    ▷ Output updated intermediate representation
```

3.2 Leveraging Multiple Vehicles

V2VNet has three main stages: (1) a convolutional network block that processes raw sensor data and creates a compressible intermediate representation, (2) a cross-vehicle aggregation stage, which aggregates information received from multiple vehicles with the vehicle's internal state (computed from its own sensor data) to compute an updated intermediate representation, (3) an output network that computes the final P&P outputs. We now describe these steps in more details. We refer the reader to Fig. 2 for our V2VNet architecture.

LiDAR Convolution Block: Following the architecture from [45], we extract features from LiDAR data and transform them into bird's-eye-view (BEV). Specifically, we voxelize the past five LiDAR point cloud sweeps into $15.6\,\text{cm}^3$ voxels, apply several convolutional layers, and output feature maps of shape $H \times W \times C$, where $H \times W$ denotes the scene range in BEV, and C is the number of feature channels. We use 3 layers of 3×3 convolution filters (with strides of 2, 1, 2) to produce a 4x downsampled spatial feature map. This is the intermediate representation that we then compress and broadcast to other nearby SDVs.

Compression: We now describe how each vehicle compresses its intermediate representations prior to transmission. We adapt Ballé *et al.*'s variational image compression algorithm [2] to compress our intermediate representations; a convolutional network learns to compress our representations with the help of a learned hyperprior. The latent representation is then quantized and encoded losslessly with very few bits via entropy encoding. Note that our compression module is differentiable and therefore trainable, allowing our approach to learn how to preserve the feature map information while minimizing bandwidth.

Cross-vehicle Aggregation: After the SDV computes its intermediate representation and transmits its compressed bitstream, it decodes the representation received from other vehicles. Specifically, we apply entropy decoding to the bit stream and apply a decoder CNN to extract the decompressed feature map. We then aggregate the received information from other vehicles to produce an

Table 1. Detection Average Precision (AP) at IoU = {0.5, 0.7}, prediction with ℓ_2 error at recall 0.9 at different timestamps, and Trajectory Collision Rate (TCR).

Method	AP@IoU ↑		ℓ_2 Error (m) ↓			TCR ↓
	0.5	0.7	1.0 s	2.0 s	3.0 s	$\tau = 0.01$
No Fusion	77.3	68.5	0.43	0.67	0.98	2.84
Output Fusion	90.8	86.3	**0.29**	**0.50**	0.80	3.00
LiDAR Fusion	92.2	88.5	**0.29**	**0.50**	0.79	2.31
V2VNet	**93.1**	**89.9**	**0.29**	**0.50**	**0.78**	**2.25**

updated intermediate representation. Our aggregation module has to handle the fact that different SDVs are located at different spatial locations and see the actors at different timestamps due to the rolling shutter of the LiDAR sensor and the different triggering per vehicle of the sensors. This is important as the intermediate feature representations are spatially aware.

Towards this goal, each vehicle uses a fully-connected graph neural network (GNN) [39] as the aggregation module, where each node in the GNN is the state representation of an SDV in the scene, including itself (see Fig. 3). Each SDV maintains its own local graph based on which SDVs are within range (i.e., 70 m). GNNs are a natural choice as they handle dynamic graph topologies, which arise in the V2V setting. GNNs are deep-learning models tailored to graph-structured data: each node maintains a state representation, and for a fix number of iterations, messages are sent between nodes and the node states are updated based on the aggregated received information using a neural network. Note that the GNN messages are different from the messages transmitted/received by the SDVs: the GNN computation is done locally by the SDV. We design our GNN to temporally warp and spatially transform the received messages to the receiver's coordinate system. We now describe the aggregation process that the receiving vehicle performs. We refer the reader to Algorithm 1 for pseudocode.

We first compensate for the time delay between the vehicles to create an initial state for each node in the graph. Specifically, for each node, we apply a convolutional neural network (CNN) that takes as input the received intermediate representation $\hat{z}_i$, the relative 6DoF pose Δp_i between the receiving and transmitting SDVs and the time delay $\Delta t_{i \rightarrow k}$ with respect to the receiving vehicle sensor time. Note that for the node representing the receiving car, $\hat{z}$ is directly its intermediate representation. The time delay is computed as the time difference between the sweep start times of each vehicle, based on universal GPS time. We then take the time-delay-compensated representation and concatenate with zeros to augment the capacity of the node state to aggregate the information received from other vehicles after propagation (line 3 in Algorithm 1).

Next we perform GNN message passing. The key insight is that because the other SDVs are in the same local area, the node representations will have overlapping fields of view. If we intelligently transform the representations and share information between nodes where the fields-of-view overlap, we can enhance the

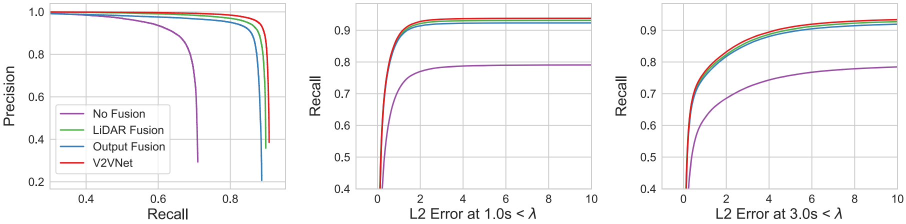

Fig. 4. Left: Detection Precision-Recall (PR) Curve at IoU = 0.7. Center/Right: Recall as a function of L_2 Error Prediction at 1.0 s and 3.0 s.

SDV's understanding of the scene and produce better output P&P. Figure 3 visually depicts our spatial aggregation module. We first apply a relative spatial transformation $\xi_{i \to k}$ to warp the intermediate state of the i-th node to send a GNN message to the k-th node. We then perform joint reasoning on the spatially-aligned feature maps of both nodes using a CNN. The final modified message is computed as in Algorithm 1 line 7, where T applies the spatial transformation and resampling of the feature state via bilinear-interpolation, and $M_{i \to k}$ masks out non-overlapping areas between the fields of view. Note that with this design, our messages maintain the spatial awareness.

We next aggregate at each node the received messages via a mask-aware permutation-invariant function ϕ_M and update the node state with a convolutional gated recurrent unit (ConvGRU) (Algorithm 1 line 8), where $j \in N(i)$ are the neighboring nodes in the network for node i and ϕ_M is the mean operator. The mask-aware accumulation operator ensures only overlapping fields-of-view are considered. In addition, the gating mechanism in the node update enables information selection for the accumulated received messages based on the current belief of the receiving SDV. After the final iteration, a multilayer perceptron outputs the updated intermediate representation (Algorithm 1 Line 11). We repeat this message propagation scheme for a fix number of iterations.

Output Network: After performing message passing, we apply a set of four Inception-like [42] convolutional blocks to capture multi-scale context efficiently, which is important for prediction. Finally, we take the feature map and exploit two network branches to output detection and motion forecasting estimates respectively. The detection output is (x, y, w, h, θ), denoting the position, size and orientation of each object. The output of the motion forecast branch is parameterized as (x_t, y_t), which denotes the object's location at future time step t. We forecast the motion of the actors for the next 3 s at 0.5 s intervals. Please see supplementary for additional architecture and implementation details.

3.3 Learning

We first pretrain the LiDAR backbone and output headers, bypassing the cross-vehicle aggregation stage. Our loss function is cross-entropy on the vehicle classification output and smooth ℓ_1 on the bounding box parameters. We apply hard-

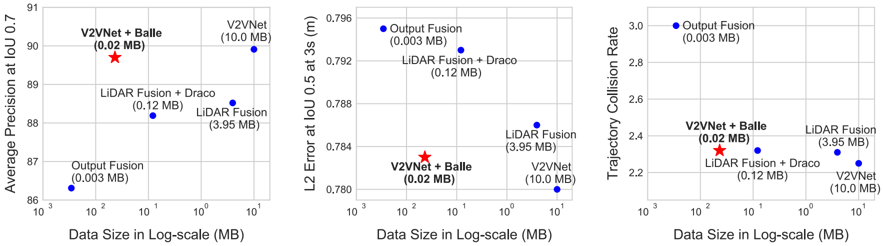

Fig. 5. Compression: Detection (AP at IoU 0.7), Prediction (ℓ_2 error at recall 0.9 at 3.0 s), and Trajectory Collision Rate ($\tau = 0.01$) performance on models with compression module.

negative mining to improve performance. We then finetune jointly the LiDAR backbone, cross-vehicle aggregation, and output header modules on our novel V2V dataset (see Sect. 4) with synchronized inputs (no time delay) using the same loss function. We do not use the temporal warping function at this stage. During training, for every example in the mini-batch, we randomly sample the number of connected vehicles uniformly on $[0, min(c, 6)]$, where c is the number of candidate vehicles available. This is to make sure V2VNet can handle arbitrary graph connectivity while also making sure the fraction of vehicles on the V2V network remains within the GPU memory constraints. Finally, the temporal warping function is trained to compensate for time delay with asynchronous inputs, where all other parts of the network are fixed. We uniformly sample time delay between 0.0 s and 0.1 s (time of one 10 Hz LiDAR sweep). We then train the compression module with the main network (backbone, aggregation, output header) fixed. We use a rate-distortion objective, which aims to maximize the bit rate in transmission while minimizing the distortion between uncompressed and decompressed data. We define the rate objective as the entropy of the transmitted code, and the distortion objective as the reconstruction loss (between the decompressed and uncompressed feature maps).

4 V2V-Sim: A Dataset for V2V Communication

No realistic dataset for V2V communication exists in the literature. Some approaches simulate the V2V setting by using different frames from KITTI [16] to emulate multiple vehicles [8,32,44]. However, this is unrealistic since sensor measurements are at different timestamps, so moving objects may be at completely different locations (e.g., a 1 s. time difference can cause 20 m change in position). Other approaches utilize a platoon strategy for data collection [7,23,35,47], where each vehicle follows behind the previous one closely. While more realistic than using KITTI, this data collection is biased: the perspectives of different vehicles are highly correlated with each other, and the data does not provide the richness of different V2V scenarios. For example, we will never see SDVs coming in the opposite direction, or SDVs turning from other lanes at intersections.

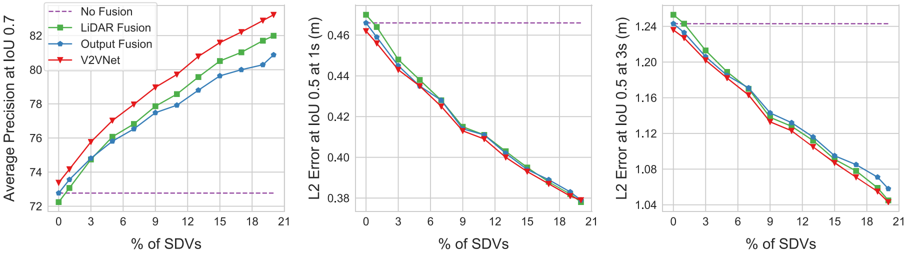

Fig. 6. Density of SDV: AP at IoU = 0.7 and ℓ_2 error at {1 s, 3 s} at highest recall rate at IoU = 0.5 wrt % of SDVs in the scene.

To address these deficiencies, we use a high-fidelity LiDAR simulator, LiDARsim [33], to generate our large-scale V2V communication dataset, which we call *V2V-Sim*. LiDARsim is a simulation system that uses a large catalog of 3D static scenes and dynamic objects that are built from real-world data collections to simulate new scenarios. Given a scenario (i.e., scene, vehicle assets and their trajectories), LiDARsim applies raycasting followed by a deep neural network to generate a realistic LIDAR point cloud for each frame in the scenario.

We leverage traffic scenarios captured in the real world ATG4D dataset [45] to generate our simulations. We recreate the snippets in LiDARsim's virtual world using the ground-truth 3D tracks provided in ATG4D. By using the same scenario layouts and agent trajectories recorded from the real world, we can replicate realistic traffic. In particular, at each timestep, we place the actor 3D-assets into the virtual scene according to the real-world labels and generate the simulated LiDAR point cloud seen from the different candidate vehicles (see Fig. 1, right). We define the candidate vehicles to be non-parked vehicles that are within the 70-m broadcast range of the vehicle that recorded the real-world snippet. We generate 5500 25 s snippets collected from multiple cities. We subsample the frames in the snippets to produce our final 46,796/4,404 frames for train/test splits for the V2V-Sim dataset. V2V-Sim has on average 10 candidate vehicles that could be in the V2V network per sample, with a maximum of 63 and a variance of 7, demonstrating the traffic diversity. The fraction of vehicles that are candidates increases linearly w.r.t broadcast range.

5 Experimental Evaluation

In this section we showcase the performance of our approach compared to other transmission and aggregation strategies as well as single vehicle P&P.

Metrics: We evaluate both detection and motion forecasting around the ego-vehicle with a range of: $x \in [-100, 100]$ m, $y \in [-40, 40]$ m. We include completely occluded objects (0 LiDAR points hit the object), making the task much more challenging and realistic than standard benchmarks. For object detection, we compute Average Precision (AP) and Precision-Recall (PR) Curve at

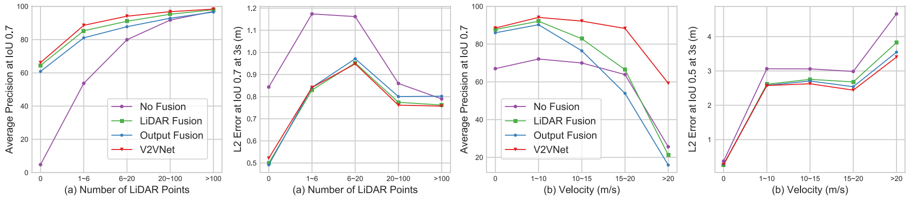

Fig. 7. Performance on objects with (fist two columns) different number of LiDAR point observation (last two columns) different velocities.

Intersection-over-Union (IoU) threshold of 0.7. For motion forecasting, we compute absolute ℓ_2 displacement error of the object center's location at future timestamps (3 s prediction horizon with 0.5 s interval) on true positive detections. We set the IoU threshold to 0.5 and recall to 0.9 (we pick the highest recall if 0.9 cannot be reached) to obtain the true positives. These values were chosen such that we retrieve most objects, which is critical for safety in self-driving. Note that most self-driving systems adopt this high recall as operating point. We also compute Trajectory Collision Rate (TCR), defined as the collision rate between the predicted trajectories of detected objects, where collision occurs when two cars overlap with each other more than a specific IoU (i.e., collision threshold τ). This metric evaluates whether the predictions are consistent with each other. We exclude the other SDVs during evaluation, as those can be trivially predicted.

Baselines: We evaluate the single vehicle setting, dubbed *No Fusion*, which consists of LiDAR backbone network and output headers only, without V2V communication. We also introduce two baselines for V2V communication: *LiDAR Fusion* and *Output Fusion*. *LiDAR Fusion* warps all received LiDAR sweeps from other vehicles to the coordinate frame of the receiver via the relative transformation between vehicles (which is known, as all SDVs are assume to be localized) and performs direct aggregation. We use the state-of-the-art LiDAR compression algorithm Draco [1] to compress *LiDAR Fusion* messages. For *Output Fusion*, each vehicle sends post-processed outputs, i.e., bounding boxes with confidence scores, and predicted future trajectories after non-maximum suppression (NMS). At the receiver end, all bounding boxes and future trajectories are first transformed to the ego-vehicle coordinate system and then aggregated across vehicles. NMS is then applied again to produce the final results.

Experimental Details: For all analysis we set the maximum number of SDVs per scene to be 7 (except for an ablation study measuring how the number of SDVs affect V2V performance in Fig. 6). All models are trained with Adam [24].

Comparison to Existing Approaches: As shown in Table 1, V2V-based models significantly outperform *No Fusion* on detection (∼20% at IoU 0.7)

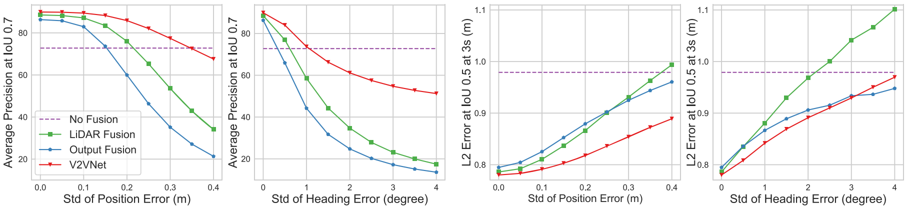

Fig. 8. Robustness on noisy vehicles' relative pose estimates.

and prediction (∼0.2 m ℓ_2 error reduction at 3 s.). *LiDAR Fusion* and *V2VNet* also show strong reduction (20% at 0.01 collision threshold) in TCR. These results demonstrate that all types of V2V communication provide substantial performance gains. Among all V2V approaches, *V2VNet* is either on-par with *LiDAR Fusion* (which has no information loss) or achieves the best performance. V2VNet's slight performance gain over *LiDAR Fusion* may come from using the GNN in the cross-vehicle aggregation stage to reason about different vehicles' feature maps more intelligently than naive aggregation. *Output Fusion*'s drop in performance for TCR is due to the large number of false positives relative to other V2V methods (see detection PR curve Fig. 4, left, at recall >0.6). Figure 4 shows the percentage of objects with an ℓ_2 error at 3 s smaller than a constant. This metric shows similar trends consistent with Table 1.

Compression: Figure 5 shows the tradeoff between transmission bandwidth and accuracy for different V2V methods with and without compression. Draco [1] achieves 33x compression for *LiDAR Fusion*, while our compressed intermediate representations achieved a 417x compression rate. Note that compression marginally affects the performance. This shows that the intermediate P&P representations are much easier to compress than LiDAR. Given the message size for one timestamp with a sensor capture rate of 10 Hz, we compute the transmission delay based on V2V communication protocol [21]. At the broadcast range 120 m, the data rate is roughly 25 Mbps. This means sending *V2VNet* messages may induce roughly 9 ms delay, which is very low.

SDV Density: We now investigate how V2V performance changes as a function of % of SDVs in the scene. To make this setting like the real world, for a given 25 s snippet, we choose a fraction of candidate vehicles in the scene to be SDVs for the whole snippet. As shown in Fig. 6, V2V performance increases linearly with the % of SDVs in both detection and prediction.

Number of LiDAR Points, Velocity: As shown in Fig. 7(a) V2V methods boost the performance on completely- and mostly-occluded objects (0 and 1–6 LiDAR points) by over 60% in AP. This is an extremely exciting result, since the main challenges of perception and motion forecasting are objects with very

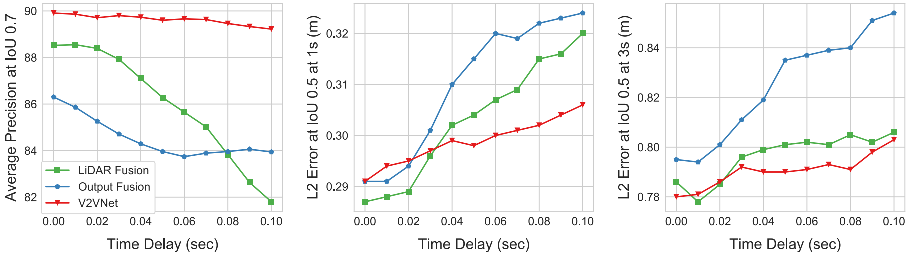

Fig. 9. Effect of time delay in data exchange.

sparse observations. Figure 7(b) shows performance on objects with different velocities. While other V2V methods drop in detection performance as object velocity increases, *V2VNet* has consistent performance gains over *No Fusion* on fast moving objects. *Output Fusion* and *LiDAR Fusion* may have deteriorated due to the rolling shutter of the moving SDV and the motion blur of moving agents during the temporal sweep of the LiDAR sensor. These effects are more severe in the V2V setting, where SDVs may be moving in opposite directions at high speeds while recording moving actors. Although not explicitly tackling such issue, *V2VNet* performs contextual and iterative reasoning on information from different vehicles, which may indirectly handle rolling shutter inconsistencies.

Imperfect Localization: We simulate inaccurate pose estimates by introducing different levels of Gaussian ($\sigma = 0.4\,\text{m}$) and von Mises ($\sigma = 4°$; $\frac{1}{\kappa} = 4.873 \times 10^{-3}$) noise to position and heading of the transmitting SDVs. As shown in Fig. 8, on both noise types, *V2VNet* outperforms *LiDAR Fusion* and *Output Fusion* in P&P performance. The only exception is *Output Fusion* ℓ_2 error with heading noise larger than 3°. We hypothesize that *Output Fusion's* performance is better at this setting due to its low-recall (fewer true positives) relative to V2VNet (0.62 vs. 0.73 at 4° noise). Fewer true positives can cause lower ℓ_2 error relative to higher recall methods. Degradation from heading noise is more severe than position noise, as subtle rotation in the ego-view will cause substantial misalignment for far-off objects; a vehicle bounding box ($5\,\text{m} \times 2\text{m}$) rotated by 1° with respect to a pivot 70 m away generates an IoU of 0.39 with the original.

Asynchronous Propagation: We simulate random time delay by delaying the messages of other vehicles at random from $\mathcal{U}(0, t)$, where $t = 0.1$. We apply a piece-wise linear velocity model (computed via finite differences) in *Output Fusion* to compensate for time delay. We do not make adjustments for *LiDAR Fusion* as it is non-trivial. As shown in Fig. 9, *V2VNet* demonstrates robustness across different time delays. *Output Fusion* does not perform well at high time delays as the piece-wise linear model used is sensitive to velocity estimates.

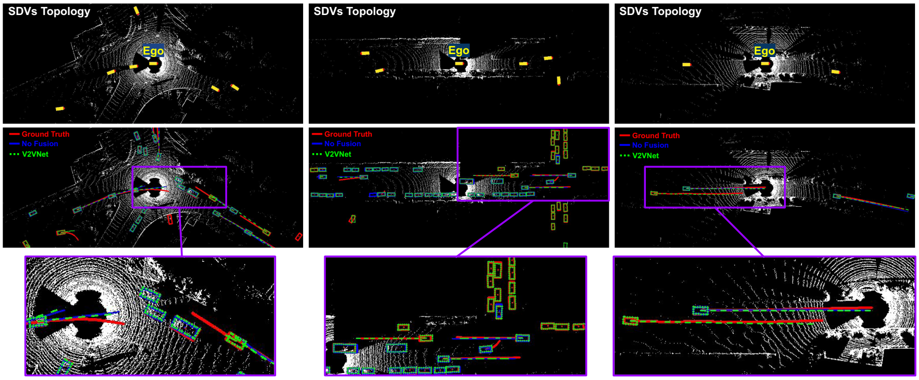

Fig. 10. V2V-Net Qualitative Examples. Left: Occluded car detected; Middle: Perception range increased; Right: Fast car detected.

Mixed Fleet: We also investigate the case that the SDV may receive different types of perception messages (i.e., sensor data, intermediate representation and P&P outputs). We analyze the setting where every SDV (other than the receiving vehicle) has 1/3 chance to broadcast each measurement type. We then perform *Sensor Fusion*, *V2VNet*, *Output Fusion* for the relevant set of messages to generate the final output. The result is in between the three V2V approaches: 88.6 AP at IoU = 0.7 for detection, 0.79 m error at 3.0 s prediction, and 2.63 TCR.

Qualitative Results: As shown in Fig. 10, V2VNet can see further and handle occlusion. For example, in Fig. 10 far right, we perceive and motion forecast a high-speed vehicle in our right lane, which can give the downstream planning system more information to better plan a safe maneuver for a lane change. *V2V-Net* also detects many more vehicles in the scene that were originally not detected by *No Fusion* (Fig. 10, middle).

6 Conclusion

In this paper, we have proposed a V2V approach for perception and prediction that transmits compressed intermediate representations of the P&P neural network, achieving the best compromise between accuracy improvements and bandwidth requirements. To demonstrate the effectiveness of our approach we have created a novel *V2V-Sim* dataset that realistically simulates the world when SDVs will be ubiquitous. We hope that our findings will inspire future work in V2V perception and motion forecasting strategies for safer self-driving cars.

Acknowledgments. We gratefully acknowledge James Tu for valuable contributions in the final paper.

References

1. Draco 3d data compression (2019). https://github.com/google/draco
2. Ballé, J., Minnen, D., Singh, S., Hwang, S.J., Johnston, N.: Variational image compression with a scale hyperprior. In: International Conference on Learning Representations (2018)
3. Casas, S., Gulino, C., Liao, R., Urtasun, R.: Spatially-aware graph neural networks for relational behavior forecasting from sensor data. arXiv (2019)
4. Casas, S., Gulino, C., Suo, S., Luo, K., Liao, R., Urtasun, R.: Implicit latent variable model for scene-consistent motion forecasting. In: ECCV (2020)
5. Casas, S., Luo, W., Urtasun, R.: Intentnet: learning to predict intention from raw sensor data. In: Conference on Robot Learning (2018)
6. Chai, Y., Sapp, B., Bansal, M., Anguelov, D.: Multipath: multiple probabilistic anchor trajectory hypotheses for behavior prediction. arXiv (2019)
7. Chen, Q., et al.: DSRC and radar object matching for cooperative driver assistance systems. In: 2015 IEEE Intelligent Vehicles Symposium (IV) (2015)
8. Chen, Q., Tang, S., Yang, Q., Fu, S.: Cooper: cooperative perception for connected autonomous vehicles based on 3D point clouds. arXiv (2019)
9. Chen, S., Li, Y., Kwok, N.M.: Active vision in robotic systems: a survey of recent developments. Int. J. Robot. Res. **30**(11), 1343–1377 (2011)
10. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3D object detection network for autonomous driving. In: CVPR (2017)
11. Choi, H., Bajic, I.V.: High efficiency compression for object detection. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2018)
12. Cui, H., et al.: Deep kinematic models for physically realistic prediction of vehicle trajectories. arXiv (2019)
13. Davison, A.J., Murray, D.W.: Simultaneous localization and map-building using active vision. PAMI (2002)
14. Davison, A.J.: Mobile robot navigation using active vision. Advances in Scientific Philosophy Essays in Honour of (1999)
15. Duvenaud, D.K., et al.: Convolutional networks on graphs for learning molecular fingerprints. In: NIPS (2015)
16. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The kitti vision benchmark suite. In: CVPR (2012)
17. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70 (2017)
18. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS (2017)
19. Jain, A., Casas, S., Liao, R., Xiong, Y., Feng, S., Segal, S., Urtasun, R.: Discrete residual flow for probabilistic pedestrian behavior prediction. arXiv (2019)
20. Jayaraman, D., Grauman, K.: Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9909, pp. 489–505. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46454-1_30
21. Kenney, J.B.: Dedicated short-range communications (DSRC) standards in the united states. Proc. IEEE **99**(7), 1162–1182 (2011)
22. Kim, A., Eustice, R.M.: Active visual slam for robotic area coverage: theory and experiment. Int. J. Robot. Res. **34**(4–5), 457–475 (2015)

23. Kim, S.W., et al.: Multivehicle cooperative driving using cooperative perception: design and experimental validation. IEEE Trans. Intell. Transp. Syst. **16**(2), 663–680 (2014)
24. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings, San Diego, CA, USA, 7–9 May 2015 (2015)
25. Li, L., Yang, B., Liang, M., Zeng, W., Ren, M., Segal, S., Urtasun, R.: End-to-end contextual perception and prediction with interaction transformer. In: IROS (2020)
26. Li, R., Tapaswi, M., Liao, R., Jia, J., Urtasun, R., Fidler, S.: Situation recognition with graph neural networks. In: ICCV (2017)
27. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.S.: Gated graph sequence neural networks. In: 4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings, San Juan, Puerto Rico, 2–4 May 2016 (2016)
28. Liang, M., Yang, B., Chen, Y., Hu, R., Urtasun, R.: Multi-task multi-sensor fusion for 3D object detection. In: CVPR (2019)
29. Liang, M., Yang, B., Wang, S., Urtasun, R.: Deep continuous fusion for multi-sensor 3D object detection. In: ECCV (2018)
30. Liang, M., Yang, B., Zeng, W., Chen, Y., Hu, R., Casas, S., Urtasun, R.: PnpNet: learning temporal instance representations for joint perception and motion forecasting. In: CVPR (2020)
31. Luo, W., Yang, B., Urtasun, R.: Fast and furious: real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net. In: CVPR (2018)
32. Maalej, Y., Sorour, S., Abdel-Rahim, A., Guizani, M.: Vanets meet autonomous vehicles: a multimodal 3D environment learning approach. In: GLOBECOM 2017–2017 IEEE Global Communications Conference (2017)
33. Manivasagam, S., et al.: Lidarsim: realistic lidar simulation by leveraging the real world. In: CVPR (2020)
34. Rauch, A., Klanner, F., Rasshofer, R., Dietmayer, K.: Car2x-based perception in a high-level fusion architecture for cooperative perception systems. In: 2012 IEEE Intelligent Vehicles Symposium (2012)
35. Rawashdeh, Z.Y., Wang, Z.: Collaborative automated driving: a machine learning-based method to enhance the accuracy of shared information. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (2018)
36. Rhinehart, N., Kitani, K.M., Vernaza, P.: R2p2: a reparameterized pushforward policy for diverse, precise generative path forecasting. In: ECCV (2018)
37. Rhinehart, N., McAllister, R., Kitani, K., Levine, S.: Precog: prediction conditioned on goals in visual multi-agent settings. arXiv (2019)
38. Rockl, M., Strang, T., Kranz, M.: V2V communications in automotive multi-sensor multi-target tracking. In: 2008 IEEE 68th Vehicular Technology Conference (2008)
39. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European Semantic Web Conference (2018)
40. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3D shape recognition. In: ICCV (2015)
41. Sykora, Q., Ren, M., Urtasun, R.: Multi-agent routing value iteration network. In: ICML 2020 (2020)
42. Szegedy, C., et al.: Going deeper with convolutions. In: CVPR (2015)
43. Wei, X., Barsan, I.A., Wang, S., Martinez, J., Urtasun, R.: Learning to localize through compressed binary maps. In: CVPR (2019)

44. Xiao, Z., Mo, Z., Jiang, K., Yang, D.: Multimedia fusion at semantic level in vehicle cooperactive perception. In: 2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW) (2018)
45. Yang, B., Luo, W., Urtasun, R.: Pixor: real-time 3D object detection from point clouds. In: CVPR (2018)
46. Yu, B., Yin, H., Zhu, Z.: Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. arXiv (2017)
47. Yuan, T., et al.: Object matching for inter-vehicle communication systems-an IMM-based track association approach with sequential multiple hypothesis test. IEEE Trans. Intell. Transp. Syst. **18**(12), 3501–3512 (2017)
48. Yun, S., Choi, J., Yoo, Y., Yun, K., Young Choi, J.: Action-decision networks for visual tracking with deep reinforcement learning. In: CVPR (2017)