Incremental Perception on Real Time 3D Data

Arup Kumar Sarker
University of Virginia
USA

Felix Xiaozhu Lin
University of Virginia
USA

ABSTRACT

Perception based on 3D data in autonomous vehicles (AV) pursues high accuracy at a high computational cost. It delays the downstream modules in an AV pipeline – planning, prediction, and control. This prolongs the AV's time-to-decision and ultimately hurts the vehicle's maneuver and safety. Towards efficient perception, our insight is that the perception module should attend to 3D pixels that are more relevant to the downstream modules. Accordingly, we propose *incremental perception*: each 3D frame is processed in progressive iterations; early iterations emit inexact perception results to the downstream modules; later iterations incorporate feedback from the downstream modules and accordingly refine a 3D frame's most relevant portions. Our early results show reduction in AV's time-to-decision and therefore safety improvement.

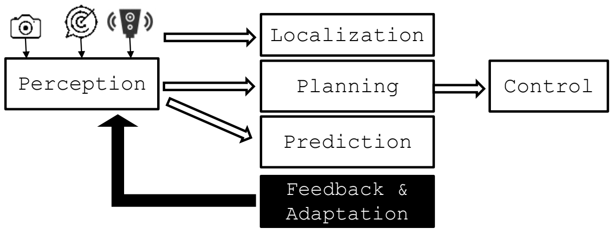

Figure 1: An overview of the AV pipeline and our proposed addition (in black)

CCS CONCEPTS

• **Computer systems organization → Embedded software**.

ACM Reference Format:
Arup Kumar Sarker and Felix Xiaozhu Lin. 2022. Incremental Perception on Real Time 3D Data. In *The 23rd International Workshop on Mobile Computing Systems and Applications (HotMobile '22), March 9–10, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3508396.3512875

1 INTRODUCTION

3D vision based on point clouds has core use cases in AVs [25]. A key challenge is a tension between high computational cost and the need for low time-to-decision. Point clouds are originated in LiDAR and fed to a perception module; the perception results, e.g. object bounding boxes and labels, are further fed to downstream modules including prediction, planning, and control. The short delay between data ingestion and control stimulus is crucial to a vehicle's maneuver. Prior work suggests that the delay should be less than 100 ms for safety and passenger comfort [1, 7].

While much prior work focuses on 3D vision accuracy and efficiency [3, 18, 23, 24, 27], the current perception still pays *uniform* attention over an entire 3D point cloud, resulting in much resource waste. For instance, planning shows different sensitivities to different neighborhoods of a point cloud, requiring more accurate object detection on the trajectory of the ego vehicle than on the sidewalk. Furthermore, the sensitivities also vary across *types* of perception errors, depending on the context. For instance, the planner may care more about low spatial errors for an obstacle in front of the ego vehicle, than whether the obstacle is a sedan or a pickup truck. This resembles how humans drive: to keep our attention on relevant objects and their relevant details.

Motivated by this observation, we advocate for co-designing AV's perception and the downstream modules such as planning and prediction. As shown in Figure 1, the idea is to add a feedback path from the downstream to the perception and guide the perception with the feedback. This enables two key optimizations: (1) early exit: the perception eschews processing all 3D points in full; (2) targeted processing: the perception chooses algorithmic building blocks that best cater to the need of downstream modules. As a result, the perception can reduce its processing delay and achieve higher efficiency. To realize the idea, we propose a mechanism called *incremental perception*, where the perception module processes each point cloud frame in multiple iterations. The initial iteration quickly produces inexact results, e.g., rough 3D bounding boxes. Based on the inexact results and the associated confidence, downstream modules either make decisions with low delays (e.g., change the maneuver class so that the ego vehicle can decelerate early) or request the perception to refine the results with additional iterations (e.g., to get more accurate coordinates of certain bounding boxes). Essentially, the downstream steers the perception module's attention to the most relevant details of the 3D scene.

Incremental perception raises multiple challenges to the existing AV stack, which we discuss as open questions. (1) The 3D vision algorithms shall expose tradeoffs of specific error types; they should support incremental computation with results reuse across iterations. (2) The planning module should act on inexact perception when it is appropriate and provide feedback to the perception for refinement. (3) A vision job manager shall dynamically create and configure predefined and on-demand vision jobs, and dispatch them to respective point cloud frames. (4) The OS should schedule a job graph with circular dependency.

This paper makes the following contributions:

- We identify the lack of targeted attention as a major inefficiency in AV's perception. In response, we advocate for guiding the perception with feedback from downstream modules, notably planning and prediction.
- We propose the mechanisms of incremental perception and priority object vectors, which allow an AV system to act upon perception results with low delays and refine perception on demand.
- We identify open research questions in 3D vision algorithms, planning, and AV job management.

2 BACKGROUND

An AV pipeline comprises major modules as shown in Figure 1. The **perception** module captures input from multiple sensors: radars, 2D cameras, LiDAR, inertial measurement units (IMUs), and global navigation satellite system (GNSS) receivers. The perception fuses multi-modal sensor data to detect and track nearby objects. Consuming the perception results, the **localization** module identifies the current position of ego vehicle with predefined HD maps; the **prediction** module predicts the intentions and actions of nearby objects. Based on the information, the **planning** module generates a navigation plan for the next several seconds. According to the plan, the **control** module takes driving policies into account and outputs control commands to the actuators. Often, the control module sends a new command every tens of milliseconds. If there is no update (e.g. the perception is busy processing a 3D frame), the actuator will continue executing the most recent command. Therefore, to timely respond to driving events, an AV system must have low time-to-decision.

3 A CASE FOR INCREMENTAL PERCEPTION

3.1 The cost of 3D vision

3D perception is expensive. Real-time processing of point clouds requires substantial computing resources. Popular object detection algorithms such as PV-RCNN [18] are reported to take over 100 ms per frame on a modern GPU such as Titan RTX; such a delay is already over the ideal time-to-decision which is often tens of ms. In real-world deployment, 3D perception is likely to take even longer. (1) The GPU is likely to face energy and thermal constraints. (2) The GPU is shared by multiple AV modules, e.g. prediction based on deep learning. (3) Recent AV systems incorporate more LiDARs per vehicle, e.g. 4–8 in recent products [13, 17], which amplify the rate of 3D data. To process points clouds from these LiDARs in real time, an AV would need multiple GPUs which can consume more than one kilowatt. Although GPUs are getting faster over years, their scaling rate can hardly catch up with the scaling of algorithm complexity and LiDARs.

Accuracy is not the only goal Most existing research optimizes AV modules in isolation. In particular, most 3D vision works focus on accuracy or efficiency within the scope of a vision algorithm. The perception accuracy, measured as mAP or IoU, is often evaluated against pre-defined thresholds, e.g. IoU >= 0.7 for successful car detection and IoU >= 0.5 for pedestrian detection. The metrics can fail to capture the result relevance to AV driving [15].

Pt %	Accuracy (IoU=0.7)			Latency
	Easy	Medium	Hard	
100%	98.2	89.6	89.0	127ms
80%	96.5	89.43	88.1	115ms
75%	94.8	86.8	80.2	105ms
70%	90.4	79.7	77.7	101ms
60%	80.5	69.6	67.6	99ms
55%	71.2	60.7	57.6	98ms

Table 1: Performance comparison of 3D object detection with partial processing of point clouds. Model: PointRCNN. Dataset: KITTI. Object: cars. GPU: Titan XP

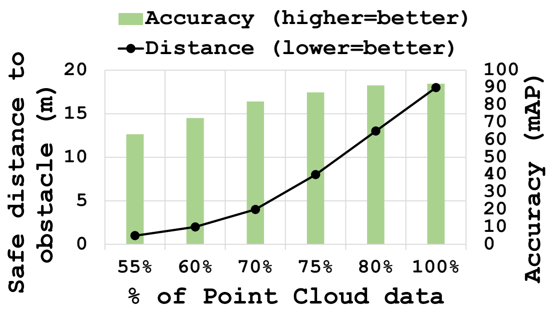

Figure 2: Processing more 3D points increases object detection accuracy (right y-axis) while reducing safe distance (left y-axis) due to longer processing delays. Object: cars. mAP: IoU>=0.7. Vehicle speed 22m/s

For this reason, inexact yet timely object detection can be more preferable than exact yet slow detection, as the former can lead to safer vehicle maneuver [9]. For instance, for the planning module to avoid colliding with an object that suddenly appears, low spatial error matters while low classification error (is the object a pedestrian or a biker?) matters less.

3.2 Benefits of incremental perception

Incremental perception executes a 3D vision pipeline in multiple refinement iterations, where each iteration consumes a part of the point cloud and/or executes a part of the vision pipeline. As a result, the AV's time-to-decision decreases; perception is more targeted and thus enjoys higher efficiency.

Table 1 shows motivational evidence for incremental perception. It shows our experiment of applying partial processing in the perception module based on PointRCNN [19]. During the training phase, we only require the regression head to regress 3D bounding box positions from the foreground points. We measure that processing a 3D frame in full (16K points) will take 127 ms. We test the model with sampling 100 percent, 80 percent, 75 percent, and 70 percent of points in separate runs. Our results show that partial processing of point clouds offers a variety of trade-offs between latency and accuracy. For instance, with 80% points, mAP for detecting a car is 91.3 (across all test cases), and the average latency per image is 115 ms; processing 75% and 70% of points reduced latency to 105 and 101 milliseconds, respectively.

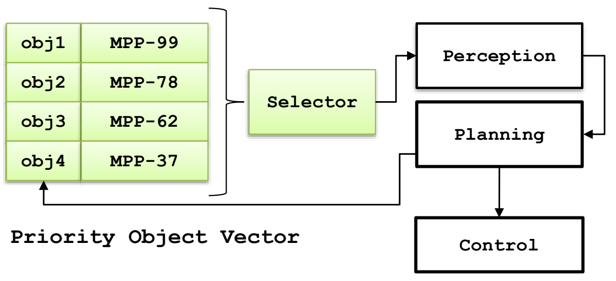

Figure 3: Proposed feedback flow with Priority Object Vector for the perception refinement request

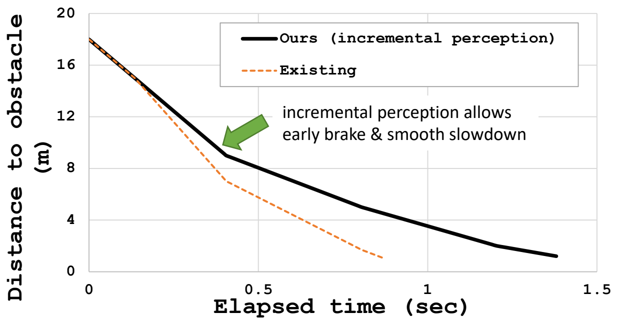

Figure 4: Example execution of the proposed incremental perception as compared to the existing one-shot perception, showing that the former results in safer, more comfortable maneuver. At time = 0, AV ingests a point cloud frame which contains an obstacle.

Based on the experiment, Figure 2 further shows the impact on driving. As the perception module processes a larger fraction of 3D points per frame, the object detection accuracy increases as expected. On the other hand, however, the safe distance also increases, meaning that the vehicle will travel a much longer distance before it fully stops and avoids collision with the detected object. A range of viable middleground exists: for instance, the perception may process 70% – 100% of 3D points, giving the control module early opportunities to start deceleration.

3.3 Need for downstream feedback

Not all objects or regions in a 3D frame have the same level of significance to the downstream modules. This has already been recognized by existing work and used to prioritize objects or regions in a frame. However, prior work exploits only static and coarse-grained knowledge, such as important object classes [12] or ground segmentation [14]; it lacks crucial contexts of planning/prediction, e.g., whether an obstacle is on the trajectory of an ego-vehicle, what are the predicted trajectories of other vehicles, etc. As a result, prior work often has hard-coded policies built in and loses flexibility.

3.4 Proposed mechanism

Our idea is to execute perceptional incrementally, with downstream tasks guiding the "attention" of perception. As a preliminary design, we introduce an adaptation module shown in black in Figure 1. The module takes feedback from the downstream and creates a priority object vector (POV). POV is continuously generated, reflecting how likely the detected objects (which have inexact features) influence the outcome of downstream modules. For example, vehicles traveling on the road will have higher priorities in the POV than those parked on the roadside. Similarly, pedestrians on a crosswalk will have higher priorities than those on sidewalks. With POV, the subsequent perception iterations tolerate errors on selective features such as bounding box coordinates or object classes while dedicating more resources to other features of objects of interest, e.g. vehicle turn signals or human ages.

We next describe two use cases of incremental perception.

Use case 1: perception guided by motion planning In AV, replanning occurs frequently in order to take into account newly detected objects and update the ego vehicle trajectories. The trajectories will exhibit different sensitivities to the objects: as these objects have different motion models and uncertainty, their trajectories may intersect with the ego trajectory in the future with different probabilities. Objects with higher probabilities of interaction should be highlighted in the POV and thus receive higher attention from the perception.

Use case 2: perception guided by risk estimation The surrounding objects pose different levels of risks to the ego vehicle. The risk is contributed by the motion patterns of nearby vehicles or pedestrians; it also depends on the context including the road condition, the traffic flow, and the lighting. The downstream modules can quantify the risk based on their knowledge and associate estimated risks with detected objects or frame regions. Based on the risk feedback, the perception may refine the bounding boxes of high-risk objects in order to estimate their velocities more accurately; it may also refine the class labels of objects in order to better model their future motion. To do so, the perception may either sample more 3D points from the neighborhoods of these objects, or add extra processing stages for these objects.

Example execution Figure 4 showcases our proposed mechanism compared to the existing one. With our incremental perception, the AV perception detects an obstacle early on (despite of inaccurate obstacle coordinates), which prompts the planning/control to decelerate the vehicle (t=0.25s). Subsequent perception iterations further refine the detection results, allowing replanning with more accurate deceleration (t=0.4s and later) until full stop. Overall, incremental perception reduces time-to-decision, allows early brake, and results in smoother vehicle slowdown. By contrast, the one-shot, non-incremental perception incurs a much longer delay in processing one 3D frame. As a result, the planning becomes aware of the obstacle much later in time (t=0.4s) and is left less time to react to the object. The planning module has to exert a higher deceleration, which is reflected as a steeper curve in Figure 4. Such maneuver increases vehicle jerk and passenger discomfort; it may even result in a collision if the obstacle's initial distance is too close.

4 RESEARCH QUESTIONS

Our proposal raises multiple challenges to AV systems.

4.1 3D vision algorithms

Vision algorithms need the following new capabilities.

• Targeted tradeoffs. While accuracy/delay tradeoffs are well studied in 3D vision, we believe they should also allow control of *types* of errors, such as bounding box errors or object categorization failures. It enables the perception to precisely adapt to the feedback from the planning and control modules.

• Incremental by construction. As a point cloud frame is processed in multiple iterations, it is beneficial for later iterations to reuse results of earlier iterations and minimize re-computation across iterations.

We next analyze how existing 3D vision algorithms can be transformed to meet the above requirements. We study PV-RCNN [18], a popular 3D object detector of point clouds. In a nutshell, PV-RCNN detects objects in two steps: (1) it first voxelizes a raw point cloud, encodes voxels with 3D sparse convolution layers, and generates 3D object proposals with object detection on a 2D bird view; (2) it then refines the 3D proposals by summarizing voxel features as vital points and aggregating keypoint to RoI grids.

We bake *targeted trade-offs* into the algorithm by tuning its hyperparameters based on their cognitive motivations. (1) To trade off the classification accuracy, we vary the object detector in step 1 by adjusting the detector's neural network depth and internal feature dimensions. It does not affect the bounding box accuracy in step 2 because it only affects PV-RCNN's step 1. (2) To trade off the bounding box accuracy, we vary PV-RCNN's keypoint sampling strategies in step 2, e.g., choosing between cheap random sampling or expensive farthest point sampling and changing the spatial density of the sampled keypoints. Doing so will not affect object classification in step 1.

To make PV-RCNN *incremental*, we decompose it along two dimensions.

• By algorithm steps. As soon as step 1 generates rough 3D box proposals with classifications, the algorithm emits the results to the downstream modules. For each proposal, the vision algorithm also emits estimated confidence. Only for the 3D boxes on which the downstream modules show interest (e.g., those intersecting with the ego vehicle's trajectory), the vision algorithm invokes step 2 to refine bounding boxes with proposal-specific features, which requires more expensive keypoint feature extraction and RoI grid pooling.

• By data sampling. As shown in Figure 2, the perception module runs the first iteration with randomly sampled points, e.g., 50% of a point cloud frame. On the sparse samples, the module runs object detection and/or segmentation and reveals inexact results to the planner, allowing it to start planning. As feedback, the planner requests additional processing over key segments and priority objects, for which the perception runs extra passes over additional 3D points sampled around the interesting segments or objects, creating refined patches in the point cloud.

The two decompositions are not mutually exclusive and can be integrated. It is worth noting that our current design does not require changing the set of neural networks used in the original PV-RCNN design. Furthermore, it may be possible to train a series of nested neural networks [6] for data sampling decomposition: each neural network runs in one iteration on a down-sampled point cloud version.

Overhead analysis The aforementioned decompositions may introduce additional or redundant computation to the perception. The key to reducing the overhead is computation reuse across iterations. For the decomposition by algorithm steps, computation reuse can be as trivial as caching results of prior steps; the overhead is therefore insignificant. For decomposition by data, computation reuse is more challenging. Because the popular building blocks of 3D vision algorithms, e.g., PointNet [3], have extensive nonlinear computations interleaved with linear layers, partial results of nonlinear computations cannot be trivially combined. Although recent works on graph neural networks proposed to reorder linear/nonlinear layers or even remove nonlinearity [22], their efficacy on 3D vision is yet to be seen.

4.2 Planning with inexact perception

Today's planning module assumes that the perception input is exact and final. With incremental perception, the planning module should work with inexact perception that is being refined and take into account the uncertainty associated with perception.

The first challenge is how to ensure that the generated vehicle maneuver is safe. To this end, we consider two planning strategies. First, the planning can use the inexact perception to support a high-level plan, e.g. maneuver class such as lane-keeping or changing. It still generates low-level plans, e.g. trajectories, with the exact perception results on the previous 3D frame. Second, the planner computes multiple alternative plans (i.e., multimodal planning [21]), assesses their risks, and starts executing one with sufficiently low risk. For instance, the AV may switch from the current lane that may have an obstacle to an adjacent lane that is certainly empty. Meanwhile, the planning module submits a refinement request to the perception module: $\mathcal{R} =< S, t, p >$, where S can be an object ID; t is the time budget for refining the perception, decided based on the velocity of the ego vehicle and other objects; p is the dimension of refinement, e.g., spatial error, classification, properties, etc. Upon the refinement arrives, the planning may continue the current plan or switch to an alternative one.

The second challenge is to ensure that the planning and control have sufficient knowledge for guiding the upstream perception. The knowledge can be conveyed in multiple ways. First, it comes from the perception. For each detected object, the perception indicates an estimated accuracy *improvement* α and time cost t. For instance, for a bounding box of pedestrian it can specify α=0.2 (mAP) and t=100 ms. Based on the possible replanning implication of the improved accuracy (e.g. will the vehicle need a new maneuver class or just adjust the trajectory?) and the ego vehicle's state after the estimated delay (100 ms), the planning module can decide whether to request refinement. Second, the knowledge comes from developers, who can specify pre-defined rules, e.g. pedestrians with small bounding boxes should receive coordinate refinement with high priority, as these bounding boxes may correspond to children. Third, the planner may learn such knowledge from historic driving traces through training. As a result, the planner can predict what objects (based on their partial, inexact features) are likely to have the highest impact on the planning outcome.

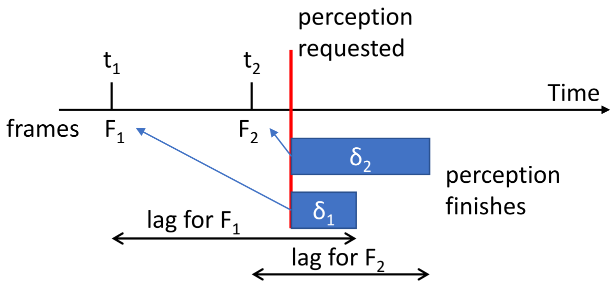

Figure 5: Job manager selects frames $F_{1,2}$

4.3 Vision job manager

A job manager creates vision jobs and configures them: dispatching jobs to input data frames and setting job hyperparameters, e.g., the depth of neural networks. In response to incremental perception, the manager manages two types of jobs:

- Predefined: these periodic jobs are scheduled on new point clouds with aggressive sampling. Their purpose is to maintain a rough, continuous perception of the surrounding environment.
- On-demand: these jobs are created based on refinement requests from downstream modules. The purpose is to increase perception accuracy on certain dimensions.

Figure 5 shows an example decision to be made by the job manager, given that the planner requests a perception update. (1) The job manager may choose to create a vision job for refining objects on a previous frame F_1 sampled at t_1, which has been roughly processed with sparse 3D point samples. By refining the partial results, the vision job incurs a short delay (δ_1) before the planner can replan. However, F_1 lags behind the physical world; the lag introduces uncertainty in object location, which the planner must take into account. (2) Alternatively, the manager may create a vision job on a more recent frame F_2 sampled at t_2. Because F_2 is fresh and unprocessed, the vision job incurs a longer delay δ_2; but the detected objects will reflect a more updated physical world. The result imposes less uncertainty on the planner.

4.4 Job scheduling

To schedule CPU, GPU, and IO jobs, existing schedulers for AV often assume a DAG task graph, where jobs have no circular dependencies. By doing so, it allows simple schedule policies such as earliest deadline first in Autoware.

Our incremental decision raises new challenges: the job manager will create jobs dynamically per the downstream feedback; the outcome of some jobs may entail more jobs, resulting in circular dependencies. To schedule jobs, a new scheduler may order all jobs by two dimensions: frame timestamps and refinement levels. With lightweight reinforcement learning, the scheduler can predict the chances of a job spawning new jobs; it can then unroll the task graph for the next scheduling horizon (often several seconds) and allocate time to jobs accordingly.

5 RELATED WORKS

Deep Learning with 3D data: Qi et al. introduced PointNet, a popular representation of 3D data [4]. It introduces a neural network model for point cloud data with multi-layer perception, pooling, and fully connected layers. To extend PointNet, they incorporated hierarchical feature learning into the network design where the centroid points are used by random FPS in the aggregation steps [3].

Some works focused on reducing latency by rasterizing a point cloud and converting points into a voxel grid [5]. Because of the low resolution of the grid, multiple points are consolidated into a single cell during the voxelization process, which loses information.

Point cloud sampling: There have been rich results on sampling point clouds for analytics. Popular packages such as PCL implements standard sampling algorithms such as VoxelGrid filter and Poisson sampling. For semantic segmentation, RandLA-Net [10] downsamples large point clouds using cheap random sampling, and makes up the possible accuracy loss with progress local feature aggregation.

Algorithm-hardware co-optimization: Another line of related work is to co-design AV pipelines with the underlying hardware. To this end, Mesorasi [8] modifies the vision algorithm to realize delayed-aggregation, a new technique for hiding long delays and reducing redundancies. They further implement delay-aggregation on neural accelerators. Tigris [23] speeds up KD trees, a key bottleneck in point cloud registration.

Co-design of AV modules: Related to our work, a small number of works optimize across modules of an AV pipeline. Fang [7] proposes to adapt motion planning algorithms to meet the target delays allocated by the whole AV system, for which it builds on projects ERDOS and Pylot [9]. Piazzoni et al. [16] models the perception's errors with regard to the AV's decision-making, which motivates our work. There has been efforts on unifying multiple AV modules with one learnable neural network [2, 26]; while effectiveness is shown, such approaches would have to forgo mature algorithms of 3D vision, planning, and control.

AV software/hardware stack: Liu et al. overviews the stack and describes the top challenges [11]. Sung et al. addresses scheduling inefficiency in the stack and shows that a modern stack can run on a single Jetson AGX Xavier [20].

6 CONCLUSIONS

This paper identifies a key inefficiency in the modern AV pipeline: the computation of perception is one-shot and does not take into account the relevance of 3D points, scene regions, and objects. To this end, we propose a feedback path from the planning and control modules to the perception module, an approach dubbed incremental perception. At run time, the perception module processes each 3D frame in multiple iterations, of which early iterations emit fast, inexact results and later iterations emit more accurate results for selective 3D points and objects. In this process, the perception continuously incorporates the feedback from planning and control, therefore steering its attention to the most relevant scene details. We present an initial design, show early evidence of the approach's benefit, and identify open questions. Incremental perception urges a rethink of the AV software stack design.

ACKNOWLEDGEMENT

The authors were supported in part by NSF awards #1846102, #1919197, #2106893, and the Virginia's Commonwealth Cyber Initiative. We thank our shepherd, Dr. Yingying Chen, and the anonymous reviewers for their insightful suggestions.

REFERENCES

[1] P. H. E. Becker, J. M. Arnau, and A. GonzÃąlez. Demystifying power and performance bottlenecks in autonomous driving systems. In *2020 IEEE International Symposium on Workload Characterization (IISWC)*, pages 205–215, 2020.

[2] S. Casas, A. Sadat, and R. Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14403–14412, 2021.

[3] H. S. Charles R Qi, Li Yi and L. J. Guibas. Point-net++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*. NIPS, NIPS, 2017.

[4] K. M. Charles Ruizhongtai Qi, Hao Su and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Conference on Computer Vision and Pattern Recognition*. CVPR, CVPR, 2017.

[5] D. X. Christopher Bongsoo Choy, K. C. JunYoung Gwak, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European Conference on Computer Vision*. ECCV, ECCV, 2016.

[6] B. Fang, X. Zeng, and M. Zhang. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 115–127, 2018.

[7] E. Fang. Dynamic deadlines in motion planning for autonomous driving systems. 2020.

[8] Y. Feng, B. Tian, T. Xu, P. Whatmough, and Y. Zhu. Mesorasi: Architecture support for point cloud analytics via delayed-aggregation. In *Proceedings of the 53th International Symposium on Microarchitecture*. ACM, ACM, 2020.

[9] I. Gog, S. Kalra, P. Schafhalter, M. A. Wright, J. E. Gonzalez, and I. Stoica. Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles. *arXiv preprint arXiv:2104.07830*, 2021.

[10] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[11] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi. Computing systems for autonomous driving: State of the art and challenges. *IEEE Internet of Things Journal*, 8(8):6469–6486, 2020.

[12] X. Liu, Y. Han, S. Bai, Y. Ge, T. Wang, X. Han, S. Li, J. You, and J. Lu. Importance-aware semantic segmentation in self-driving with discrete wasserstein training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11629–11636, 2020.

[13] G. Motors. Self-driving safety report. 2018.

[14] P. Narksri, E. Takeuchi, Y. Ninomiya, Y. Morales, N. Akai, and N. Kawaguchi. A slope-robust cascaded ground segmentation in 3d point cloud for autonomous vehicles. In *2018 21st International Conference on intelligent transportation systems (ITSC)*, pages 497–504. IEEE, 2018.

[15] A. Piazzoni, J. Cherian, M. Slavik, and J. Dauwels. Modeling perception errors towards robust decision making in autonomous vehicles. In *IJCAI*, pages 3494–3500, 2020.

[16] A. Piazzoni, J. Cherian, M. Slavik, and J. Dauwels. Modeling perception errors towards robust decision making in autonomous vehicles. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3494–3500, 2021.

[17] QCraft. Qcraft robobus.

[18] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[19] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[20] H.-H. Sung, Y. Xu, J. Guan, W. Niu, S. Liu, B. Ren, Y. Wang, and X. Shen. Enabling level-4 autonomous driving on a single 1k off-the-shelf card. *arXiv preprint arXiv:2110.06373*, 2021.

[21] C. Tang and R. R. Salakhutdinov. Multiple futures prediction. *Advances in Neural Information Processing Systems*, 32:15424–15434, 2019.

[22] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[23] T. Xu, B. Tian, and Y. Zhu. Tigris: Architecture and algorithms for 3d perception in point clouds. In *Proceedings of the 52th International Symposium on Microarchitecture*. ACM, ACM, 2019.

[24] M. S. Yangyan Li, Rui Bu, X. D. Wei Wu, and B. Chen. Pointcnn: Convolution on x-transformed points. In *32rd Conference on Neural Information Processing Systems (NeurIPS 2018)*. NeurIPS, NeurIPS, 2018.

[25] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.

[26] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019.

[27] Y. L. Zhijian Liu, Haotian Tang and S. Han. Point-voxel cnn for efficient 3d deep learning. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada*. NeurIPS, NeurIPS, 2019.