Scalable Primitives for Generalized Sensor Fusion in Autonomous Vehicles

Sammy Sidhu, **Linda Wang**, **Tayyab Naseer**, **Ashish Malhotra**, **Jay Chia**, **Aayush Ahuja**,

Ella Rasmussen, **Qiangui Huang**, and **Ray Gao**

Woven Planet Level 5

Abstract

In autonomous driving, there has been an explosion in the use of deep neural networks for perception, prediction and planning tasks. As autonomous vehicles (AVs) move closer to production, multi-modal sensor inputs and heterogeneous vehicle fleets with different sets of sensor platforms are becoming increasingly common in the industry. However, neural network architectures typically target specific sensor platforms and are not robust to changes in input, making the problem of scaling and model deployment particularly difficult. Furthermore, most players still treat the problem of optimizing software and hardware as entirely independent problems. We propose a new end to end architecture, Generalized Sensor Fusion (GSF), which is designed in such a way that both sensor inputs and target tasks are modular and modifiable. This enables AV system designers to easily experiment with different sensor configurations and methods and opens up the ability to deploy on heterogeneous fleets using the same models that are shared across a large engineering organization. Using this system, we report experimental results where we demonstrate near-parity of an expensive high-density (HD) LiDAR sensor with a cheap low-density (LD) LiDAR plus camera setup in the 3D object detection task. This paves the way for the industry to jointly design hardware and software architectures as well as large fleets with heterogeneous configurations.

1 Introduction

In a very short amount of time, Deep Neural Networks (DNNs) have become the dominant approach for solving various problems in the computer vision domain [14, 16]. This is especially true for the Autonomous Vehicle (AV) space where we see that the winning approach for nearly every benchmark dataset is Deep Learning based [7, 12, 29]. These datasets are composed of a variety of sensors such as LiDAR, RaDAR, and cameras and target a variety of tasks required for autonomous driving such as object detection [7, 12, 29], object tracking [2], semantic segmentation [2] and motion prediction [10, 29]. As a product of these public datasets, we have seen an explosion of DNN model development in literature (discussed in section 2), where we are seeing the bar being raised year over year on these tasks. However we often see that these cutting edge models usually only operate on one modality and one task such as in [21, 26] or that they perform multi-modal sensor fusion in a way that the sensor inputs are entangled and non-modular[3, 18, 22, 27, 30].

For many AV companies, the ability to scale a unified model architecture across sensor configurations, platforms, and engineering teams can outweigh the benefits of deploying an over-optimized one for a single specific platform. Ideally, we want our model architecture to be designed in a way where both sensor inputs and target tasks are modular and easily switchable. This many-to-many relationship allows models to be trained with one or multiple inputs such as images, LiDAR, or RaDAR to produce one or more of the aforementioned tasks. To achieve this, we introduce a Generalized Sensor

Machine Learning for Autonomous Driving Workshop at the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

Fusion (GSF) meta-architecture in section 3 that we use for all our experiments. Using the GSF meta-architecture, we demonstrate 3D object detection results leveraging various methods in the form of encoders such as a LiDAR PointNet voxelizer, Stereo Cost Volume and a multiview image uplifting method that we discuss in section 4.

Using the methods we implement using the GSF framework, we run ablations with various sensor configurations both for front facing and surround detection. With these results, we draw several important insights on how to design better sensor configurations and created a meta-architecture with powerful abstractions that is used by a large engineering team and can be deployed on heterogeneous sensor platforms.

2 Related Work

3D object detection using multiple sensor configurations has gained tremendous attention in recent years. We categorize relevant research in the following areas and build upon some of the working principles proposed in these approaches.

Monocular and Stereo Detection

Monocular and Stereo 3D detection methods can be broadly categorized into three domains: Direct, Voxel grid-based and Pseudo-LiDAR based.

Direct: Direct methods directly estimate 3D bounding boxes from images without generating an intermediate 3D feature representation. As there is no explicit depth-based localization of features in 3D space, these approaches geometrically reason for 3D objects by associating object keypoints in image plane. M3D-RPN[1] proposed to use depth-aware convolutions using non-shared kernels for each sub region. This enables the approach to learn location specific features that are correlated in 3D space. Stereo R-CNN [17] formulates 3D detection into multiple components explicitly to resolve several constraints like depth and matching keypoints. This approach suffers from generalization to other agent types and occlusions in image space.

Voxel Grid-based: Grid-based methods generally voxelize the world observed by the cameras into 3D grid cells and populate these cells with image features. These features are then further processed to generate 2D BEV features for efficient downstream processing. Such methods provide rich geometric reasoning to estimate 3D objects. Monocular methods like [25, 28] project image features to 3D voxels to reason downstream 3D object detection task in 3D space. Such monocular methods lack the ability of localizing features in a frustum due to depth ambiguity. CaDDN [24] copes with this problem by estimating depth distribution for each pixel before projecting the features to 3D. Similarly for stereo, DSGN [4] builds a plane sweep volume (PSV) distributed using stereo image features to estimate per pixel depth as an auxiliary task in addition to 3D object detection. It warps image-centric PSV features to 3D geometric space to perform 3D object detection in a end-to-end manner. PLUME [32] avoids building the PSV and instead estimates occupancy of 3D voxel grid as the auxiliary task. This makes the model memory and compute efficient as it does not build an explicit 3D cost volume for depth estimation. Our stereo methods in section 7 draws inspiration from both DSGN and PLUME.

Pseudo-LiDAR based: The pioneer work in this area [31, 33] generates depth information from single or stereo images and leverage existing 3D LiDAR detectors for 3D object detection. These approaches are not learned in end-to-end manner and train both the depth estimation and object detection branches separately. Such approaches suffer from learning intermediate features and depth maps that should be rather object centric than pixel centric. [23] transforms this approach to a end-to-end trainable model, so that both detection and depth losses are optimized jointly. This shows better performance than disjoint Pseudo-LiDAR methods. Ma [20] combines RGB features with the Pseudo-LiDAR pipeline by introducing a particular component that maps 2D image data to 3D point cloud.

Multiview Monocular Detection

Multiview monocular-based methods take multiple image inputs that can be stitched together to form a birds-eye-view (BEV) representation of the scene around the ego. Multiview 3D detection methods first extract features from the images, then lifts the features to a 3-dimensional frame, most often a voxel representation, that can be shared across all cameras. The challenge of projecting from 2D to 3D is that depth is required, however, the depth associated with each pixel is ambiguous. To overcome this challenge, [21] generates possible depth distributions for each pixel, which makes it able to place the context in the nearest pillar, rather than projecting along the entire ray. [26] projects

the 2D image features into an aggregated 3D voxel representation using a pinhole camera model. The 3D representations are then processed by a standard CNN to predict BEV bounding boxes.

LiDAR and Vision Fusion Detection

Multimodal fusion between LiDAR point clouds and images aims to combine the strengths of each modality. However, an end-to-end trainable model with both camera and LiDAR sensor modalities is a non-trivial task since images represent a dense 2D projection of the scene, while LiDAR captures a sparse 3D structure. For many methods, feature maps are first learned separately for each modality [3, 15, 18, 27]. After features are learned from LiDAR and images, [3] then uses the BEV map to generate 3D object proposals to fuse image and point cloud features through ROI-pooling. Similarly, [15] uses a regional proposal network on BEV and image feature maps to generate region proposals for each modality. The proposals are then passed to a detection network to fuse the modalities. Both of these approaches perform late fusion, which inhibits the network from learning interactions of the two modalities at earlier stages. To learn the interactions at earlier stages, [27] fuses LiDAR points and image features by appending each 3D point with the corresponding image feature. In order to reduce the number of image features dropped, [27] appends the image features after the point cloud has been voxelized. [18] uses a continuous fusion layer in order to bridge multiple intermediate feature layers and perform multi-sensor fusion at different scales. However, these architectures are LiDAR dependent and are not easily interchangeable between modalities.

3 Generalized Sensor Fusion

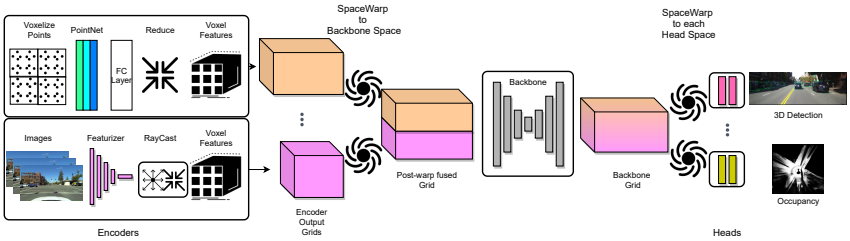

Figure 1: Overview of the Generalized Sensor Fusion (GSF) meta-architecture that allows for a modular set of both sensors and tasks.

To experiment with various sensors, methods and tasks in a modular fashion, we introduce a set of primitives called Generalized Sensor Fusion (GSF) that allow for a scalable meta-architecture. An overview can be seen in Figure 1.

- a **Space** defines the coordinate system of an abstract space. An example of this is a Cartesian Space that is defined by voxel range and resolution.
- a **Pose** defines the origin and coordinate basis change relative to a base pose. This is useful when dealing with multiple sensors in different locations in the same time-step or dealing with sensors across time-steps accounting for ego motion.
- a **Grid** is an object that abstracts a 5 dimensional feature tensor (N, C, Z, X, Y) and attaches it to both a **Space** and **Pose** that defines its coordinate system.
- **SpaceWarp** allows for the transformation of a source **Grid** into another **Space** and **Pose**, yielding a new Grid in the target Space and Pose. This allows for parts of the network to operate in different Spaces and the ability to convert one to another in an efficient and differentiable way. SpaceWarp operates by first querying the target Space object for 3D Grid sample points that we would need to transform the source grid. Then we correct for Pose and coordinate basis changes, which can also be chained if there are multiple intermediate Spaces between the source and target Space. Finally, we perform a 3D tri-linear interpolation using the corrected sample points to warp the source Grid to the target Space.
- **Encoders** are responsible for transforming source sensor inputs, such as camera images or LiDAR point clouds, into featurized **Grids**. An example of converting camera images into a Cartesian Grid is described in section 4. For a LiDAR point cloud input, an example encoder may be a PointNet sub-network [34] that yields a Grid composed of PointNet features for every 3D voxel. One thing to note is that a model may have multiple encoders each running in their own Space and Pose. Additionally, Encoders may also produce auxiliary outputs and losses that are used in downstream parts of the network.

- **Heads** are responsible for taking in a **Grid** and producing outputs for a Task such as object detection or segmentation. Head also can be parameterized by the **Space** they operate in and will **SpaceWarp** the input Grid to the target space as needed. This allows various Heads in the network to produce predictions in their own Space without modification to the **Encoders** or **Backbone**. An example is if the detection and segmentation tasks operate in different voxel cell sizes compared with each other and the Backbone.
- The **Backbone** is a shared sub-network that ingests all **Encoder** outputs and feeds all the downstream **Heads**. The input for the Backbone is a sequence of **Grids** that each get **SpaceWarped** to common **Space** defined by the Backbone. Then these warped Grids are fused and fed into a sub-network that is user defined. Finally the Backbone yields a Grid that may be in a different Space that is fed to each of the heads.

The net result allows us to implement encoders and tasks in a self contained manner. For encoders, we show a PointNet and a vision uplifting example in Figure 1 and for tasks, we show a detection and occupancy task example.

4 Raycasting Vision uplift method

Image features are inherently 2D, however, in section 3, all sensor encoders are required to have the same output space. To uplift the image features to a 3D Cartesian Space, we use a method called Raycasting. The Raycasting operation is parameterized by the target range and voxel resolution and during computation will take in a set of image feature maps from the same frame as well as their corresponding metadata such as pose and the projection matrix. The Raycast operation is comprised of two phases, the first is the project and gather phase and the second is the reduction phase. A diagram of this method is shown in Figure 2.

In the project and gather phase, we first project the centroid of each of the target voxels onto each of the inputted cameras that has that voxel in its field of view. The result of this projection is a tensor that stores the $(voxel\ id, camera\ index, u, v)$ of each valid camera voxel pairs. The second and final step of this phase is to gather the features from the image sub-network for each of the matched valid camera voxel pairs using the previously computed image coordinates; this produces a tensor that is size $(valid\ voxel\ pairs, image\ feature\ channels)$. We also store the corresponding $voxel\ id$ for each of these pairs for the reduction phase. Both of these operations are vectorized and are fast to compute on GPU without having to rely on any custom kernels. Another thing to note here is that we can have anywhere from zero to the number of cameras occurrence K for a $voxel\ id$. A $voxel\ id$ will be absent from our feature tensor in the case that it doesn't fall on any of the inputted cameras, and we may have K occurrences in the case that it falls on every camera.

In the reduction phase, we use the gathered feature tensor and corresponding $voxel\ id$s and perform a scatter reduce, which results in a tensor that is size $(num\ voxels, image\ feature\ channels)$ and has the accumulated feature vectors. For absent $voxel\ id$s, the result of the scatter for that row will be a zero vector. Next the reduced voxels are reshaped to transform it into a 3D grid. Finally we concatenate the voxel location in Cartesian coordinates as additional channels to the final grid. This ensures voxels that only fall on a single camera and share a frustum with another voxel don't produce the same result, which could be harmful for a task like object detection.

This method for uplifting differs from the one presented in [21] due to the fact that we do not encode our image features as a depth volume prior to projection and that we project the voxels onto the image plane rather than shooting out the image space depth cells. This actually yields a much higher density in the voxel grid since every voxel that is in a field of view of a camera will have image features. This method also differs from [32] since Raycast runs in a sparse fashion and allows for a generic camera setup with or without camera overlap rather than being constrained to stereo.

5 Datasets

In order to evaluate the performance of our proposed method and perform ablations on sensors, we benchmark it on 2 datasets:

- the Panoramic with LiDAR dataset: each data sample consists of a LiDAR point cloud and multiple monocular cameras covering 360° of the surroundings.
- the Forward Stereo with LiDAR Dataset (FSLD): each data sample consists of a LiDAR point cloud and a front-facing stereo camera image pair.

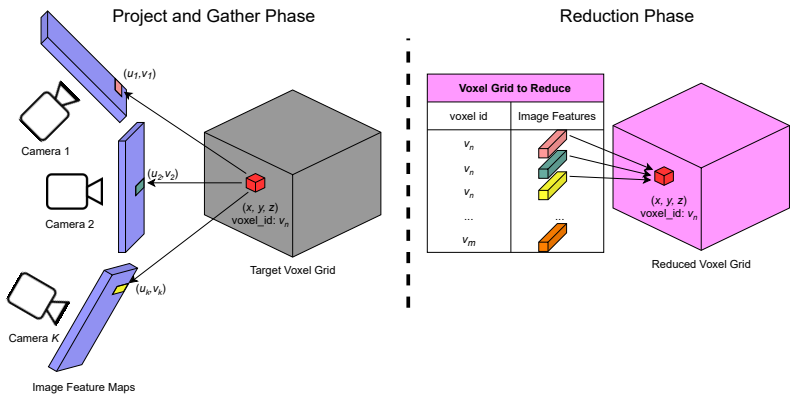

Figure 2: Overview of the Raycasting operation that maps a set of image features to a 3D grid.

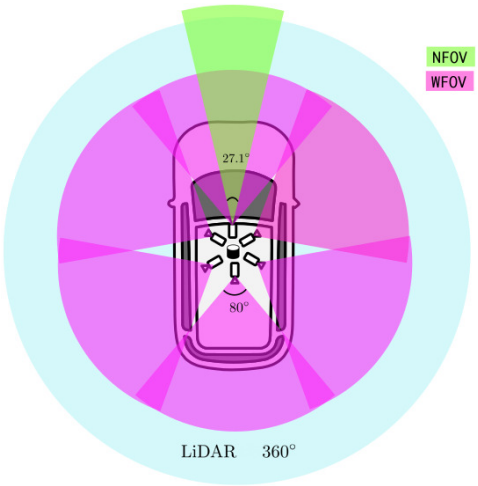

Figure 3: This figure illustrates the AV setup for the dataset described in section 5.1, where we have 6 wide field of view (WFOV) cameras and 1 narrow (NFOV) in addition to surround LiDAR.

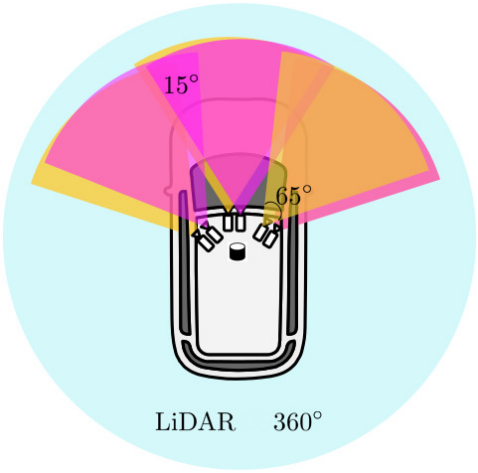

Figure 4: This figure illustrates our multi-stereo setup with 3 pairs. However we focus only on the forward facing pair.

5.1 Panoramic with LiDAR Dataset

The dataset consists of 41,525 training samples and 8,998 test samples, with a total of 860,075 car annotations. The data is collected in areas spanning across San Francisco Bay Area. Each data sample includes a LiDAR point cloud and 7 monocular cameras covering 360° of the surrounding. The cameras have a resolution of 1080×1920 pixels but we feed the models using this dataset half-res images. Figure 3 presents the details of the sensor layout.

To assess the model performance on LiDAR, our experiments compare high and low density LiDAR configurations, which are defined below. The low density configuration is similar to LiDARs we might see on high volume consumer vehicle for ADAS soon. The low density configuration is generated by sub-sampling data from the full density panoramic dataset LiDAR via vertical beam sub-sampling, azimuth sub-sampling and probability of detection mapping.

- High density (HD): Single roof LiDAR with 64 beams.
- Low density (LD): Single roof LiDAR with 13 beams and azimuth subsampling of 10.

The following table details specifications for each sensor configuration, including angular resolution, number of LiDAR beams, and maximum number of points per frame.

Config	angular step (°)	number beams	max points/scan	median points/object
HD	0.2	64	113,400	117
LD	2	13	2,340	1

Table 1: LiDAR Dataset Configurations. Median points per object is median number of points that are contained in annotated objects.

Valid Objects

Our training data contains annotations for objects across occlusion as well. As such, many objects are not visible in the camera frame. This occlusion could occur from all sorts of objects such as other cars or static obstacles such as bushes. To prevent penalizing the model during training and test for not detecting these objects that are not visible, we derive a set of valid objects using occlusion reasoning. This is done by constructing a range image of the LiDAR point cloud and deriving the percentage of the annotation box visible from the camera frame. Only boxes above a certain threshold are actually used. For objects below the visibility threshold, they do not contribute any loss during training and are ignored during metric calculation.

5.2 Forward Stereo with LiDAR Dataset

The goal of the Forward Stereo with LiDAR (FSLD) dataset is to evaluate the impact of multi-view stereo. The sensor layout, shown in Figure 4 is similar to the panoramic layout except that there are 3 forward/side stereo cameras pairs rather than 7 monocular cameras. Although, we have 3 camera pairs, we only focus on the forward facing one for our ablations. Front stereo consists 25000 training images and 3750 test frames. Each image has a 65° horizontal field of view and a resolution of 720×1536 pixels. A single stereo camera has a baseline of 35.6 cm as compared to 54 cm in KITTI [7] sensor setup. Although wider baseline cameras enable higher quality depth estimations at far ranges, these are mechanically less viable due to physical constraints in multi-view stereo setups on the cars.

6 Panoramic Experiments and Results

Using our modular GSF meta-architecture, we evaluate detection performance on a variety of configurations of vision and LiDAR. For models with LiDAR, we use a 2 layer PointNet encoder, as shown in Figure 1, that creates a learnable embedding for each voxel. All LiDAR models use the same encoder in the range (X, Y, Z) of $[-51.2m, -51.2m, -2m]$ to $[+51.2m, +51.m, 12m]$ with a voxel cell size of $[0.32m, 0.32m, 14m]$.

For models with vision, we use our Raycast method that is fed image features from a ResNet-18 [9] hour-glass model. The Raycaster then uplifts the image features to the same range as the LiDAR encoder but at voxel cell size of $[0.4m, 0.4m, 1m]$, yielding voxels in 3D rather than pillars.

For all models, we use a 3 stage ResNet style hour-glass shared backbone that SpaceWarps each of the encoder outputs to the common space which is the same as the LiDAR encoder space. All models also use the AFDet [6] head for 3D object detection. We do not use any augmentations during train or test time for these experiments. This is an area for future improvements.

Sensor Configuration Ablation

The three LiDAR profiles that we sweep over are High Density (HD), Low Density (LD), and None. In addition, we evaluate the effects of using vision in combination with LiDAR. The vision models here take in 6 wide field of view (WFOV) camera images for each frame providing full $360°$ panoramic coverage as seen in Figure 3. We also determine if having LiDAR as supervision during training only in the form of an occupancy grid task improves performance for vision only models at test time.

In Table 5, we can view the results of our various experiments with LiDAR and Vision. We establish baselines for single sensor models for LD LiDAR, HD LiDAR and panoramic vision. As expected, we see that with higher density LiDAR, our AP for LiDAR only models improves massively (+28 AP). We also see that the inclusion of vision to the model can substantially improve performance (+26AP for LD) but becomes less significant for the *HD LiDAR* model (+5AP). Next, we observe that the addition of LD LiDAR to the vision only model can improve performance significantly (+26AP). Finally, we can conclude that single modality models perform much worse compared to

LiDAR	Vision	AP@0.5	Latency
HD		93.4	16.6
	X	**98.3**	49.9
LD		64.6	**16.0**
	X	91.3	49.8
None	X	65.0	51.0
HD*	X	66.9	53.5

Figure 5: Results for sensor configuration ablations. We report the Bird's Eye View (BEV) Average Precision at 0.5 IoU as well as the inference latency in milliseconds on a NVIDIA V100. All of these models are trained/evaluated the same datasets but have different simulated configurations as described in 5.1. *HD** signifies LiDAR was only used during train time.

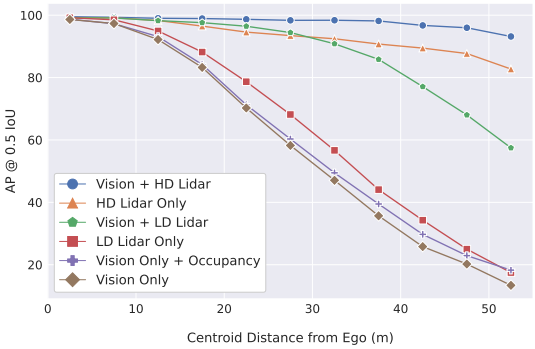

Figure 6: AP vs Distance: We measure Car AP@0.5 BEV IoU (for non-heavily-occluded objects) as a function of distance of objects from the ego in 5m radial buckets.

Run	AP @ IoU=0.5	Mean DE (cm)
Mono	65.0	62
Multiview (MV)	69.0	54

Table 2: Mono Raycasting (Mono) vs. MultiView Raycasting (MV). The metrics are computed only on objects in the overlapping region of the both camera frustums within a radius of 50m from the ego vehicle. Both AP and Mean DE (Mean depth error) see improvements in the MV case. Mean DE is absolute mean of the ground truth and prediction depth of matched objects only.

their multi-modal siblings. We also see that the *LD + vision* model performs nearly at the level of a model that only uses an expensive HD LiDAR at a fraction of the cost.

Multiview

In this experiment, we evaluate the effects of overlapping cameras by comparing the performance of the Raycast method on Multi-View (non-stereo) cameras vs. a monocular camera. In the panoramic dataset, we have input images from both WFOV cameras (FOV: 80°, FL: 3mm) and a NFOV camera (FOV: 27°, FL: 12mm). Somethings to note are that the baseline between the forward facing cameras is small and the NFOV camera frustum is completely contained in the WFOV camera (see Figure 3).

Since the NFOV camera covers less area, the results presented in Table 2 are filtered to objects that are visible to both cameras. Based on the results, multiview shows an improvement in both AP (+4AP) and localization errors (-8 cm mean DE) of matched true positive objects.

Occupancy Prediction

In addition to evaluating overlapping cameras, we also evaluate the performance gain from including an auxiliary occupancy prediction task during model training for vision only. Recent literature in multitask learning has demonstrated the ability of leveraging information learned from one task to another. By properly balancing losses from each task, research has shown that multitask models can outperform separately trained models [5, 11, 19].

The auxiliary task produces a 3D voxel grid, where each voxel's value represents the probability of the voxel being occupied. Ground truth for this task is obtained by voxelizing the point cloud obtained from our high-density LiDAR data, and labelling all voxels with at least one point as occupied. The task is performed by the occupancy prediction head, which consists of two 2D convolution layers. The last layer outputs Z channels that represent the probability of that voxel being occupied. We observe improvements to detection performance at further ranges as seen in Figure 6.

Training Details

All models in this section are trained for 60 epochs on the panoramic dataset using the Adam optimizer [13] and the OneCycle Learning Rate policy. We use 64 NVIDIA V100 16GB to train each model with a batch size of 6 frames or 36 images per GPU.

7 Stereo Experiments and Results

For our experiments on the stereo dataset, we evaluate 3 primary approaches as GSF encoders. All of the approaches share the same ResNet style image featurizer, backbone and detection head but differ in how the image is uplifted.

- **Image-Centric stereo Cost Volume (CV)**: For every stereo pair, we correlate the left and warped right features to construct 3D camera frustum volume, similar to the plane sweep volume in [4]. We then warp this cost volume to a Cartesian Grid that runs in the same GSF pipeline as the models in prior section. We also evaluate two methods to add an auxiliary loss for depth estimation. The first is a direct per-pixel depth estimation that uses LiDAR points projected onto the image as supervision. The second is an unsupervised task that minimizes the image reconstruction error similar to the work in [8].
- **Raycast multiview with stereo cameras**: Next we use the Raycast method that we discussed in section 4, but use it on the largely overlapping forward facing stereo cameras. Additionally, we evaluate the use of the occupancy auxiliary task that was discussed in 6.
- **Raycast multiview with a monocular camera**: Finally, try the same approach as above but provide the Raycaster a single image. We achieve this by simply dropping the right image of our stereo camera pair.

model	AP @ IoU=0.5
Image-centric CV Supervised	**83.6**
Image-centric CV Unsupervised	83.1
Raycast Stereo w Occupancy	76.5
Raycast Stereo w/o Occupancy	70.5
Raycast Monocular w/ Occupancy	56.5

Table 3: Image centric stereo constraints provide richer object depth information that improves 3D agent detection. The auxiliary task of unsupervised depth estimation achieves similar accuracy to the one supervised with LiDAR depth signal. Occupancy prediction significantly helps the detection task for multi-view stereo setup.

From our results for the ablation in Table 3, we can draw some observations about the Image-centric models. The first is that the Image-centric Cost volume approach on stereo cameras performs best, beating out the Raycast approach by (+7AP). However this could be due to the difference in auxiliary depth estimation between the two approaches (per-pixel vs occupancy). The next thing we see that that both methods of auxiliary depth estimation for Image-centric CV perform nearly the same, allowing us to provide the additional loss without LiDAR.

If we take a look at the Raycast results, we notice two major points. The first being that the occupancy prediction has a major effect on the stereo model, yielding a +6AP improvement. The second being that having major camera overlap also has large effect on performance. We see a +20AP difference between the monocular and stereo setup on the same data. However we believe that explicit stereo may not be needed for this type of improvement, rather just a strong multiview setup.

Training details
We use the Adam Optimizer for 40 epochs with a base learning rate of 1.25e-4 and use OneCycle learning rate scheduler with batch size 64. We use encoder range of $[-36m, 0m, -1m]$ to $[+36m, +60m, +5m]$ with a voxel resolution of $[0.3m, 0.3m, 0.5m]$. Similar to our panoramic experiments, our stereo models also use the Anchor-Free Detection Head (AFDet) [6] for 3D object detection task and we do not perform any data augmentations during training or test time.

8 Model Analysis

Here we investigate our models from section 6 in more detail. All metrics were generated on our validation set. For every prediction and ground truth object during evaluation we log statistics such as: object size, distance, number of enclosed LiDAR points and per-camera visibility.

We find that the effects of vision are more prominent at lower LiDAR densities. In Figure 6, we see that *Vision + LD* significantly out performs *Vision only* and *LD only*. *Vision + LD* AP performance is even comparable to *HD only* at short ranges (under 35m). At longer ranges, *Vision + LD* sharply degrades, which is consistent with the fact that the panoramic monocular cameras used in this study

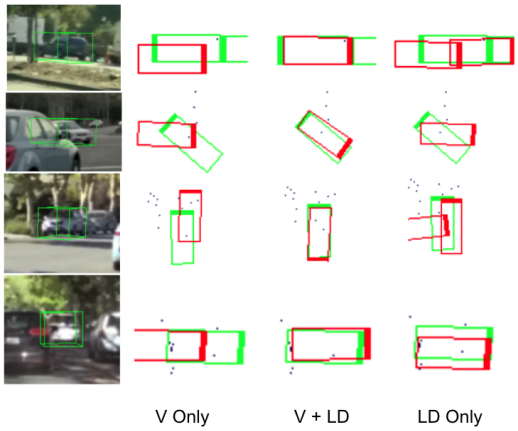

Figure 7: Examples where a multi-modal model outperforms single modality models. Groundtruth boxes are green and predicted boxes are red. We see identical detection outputs from a *Vision only* model (left) and a *Vision + LD* model (middle) and a *LD only* model. Isolating objects that have 1 LiDAR point, we see improvements in the fused model where single modality models have failed.

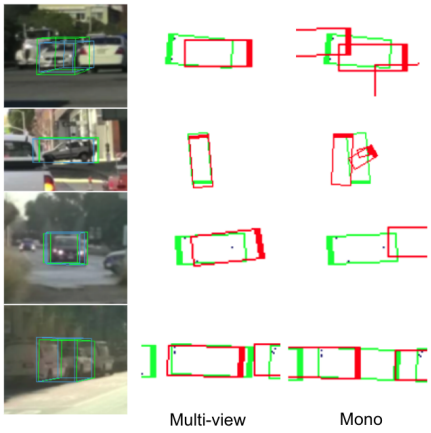

Figure 8: Some examples of where multi-view improves over monocular baseline; we observe that multiview improves localization error and reduces false positive detections. Grouthtruth boxes are in green and predicted boxes are in red. Here we use the LD dataset but the LiDAR encoder is disabled and is not used in training or evaluation.

are fairly short ranged and designed for a wide-surround view. Although the addition of multi-view vision signal to high density LiDAR has less of an effect, the AP performance still outperforms HD only.

Low Density LiDAR Regime

We can take a closer look at the synergy of vision and LiDAR in the low-density (LD) regime. We see in Table 1 that the LD configuration has a median of 1 point per object. We hypothesize that even a few LiDAR points significantly improves depth localization for vision. This is evident in Figure 6 where the performance of *Vision only* matches both *LD* and *Vision + LD* at near distances but quickly diverges. As we add more LiDAR points, we achieve longer range detections. We see more evidence of this looking at individual examples where ground truth objects only contain only a single LiDAR point Figure 7.

Effects of multiview

In section 6, we observed that adding an additional view of a scene to Raycasting improves the localization of detections, similar to the effect of adding a few LiDAR points. In Figure 8, we present a subsample of cases where multiview improves the monocular camera baseline. We observe that localization improves in true positive (TP) cases and reduces the number of false positive (FP) boxes around the objects. This is likely attributed to intersection of rays in voxel space, which helps to disambiguate the object location.

9 Conclusion

In this work, we present a modular meta-architecture that allows us to switch between different encoders and modalities to build a generic perception model that enables experimentation with different sensor configurations and deployment across a heterogeneous vehicle fleet. This paradigm allows for a team of model developers that are targeting different AV platforms to easily collaborate on the same model by simply changing parameters. By adding and removing different sensor modalities, we isolate the effects of each sensor. We find that the early fusion of vision and LiDAR is more important at the lower LiDAR densities, and that our low LiDAR density fused model can outperform more expensive sensors in certain cases. Furthermore, we find that having a few LiDAR points on objects significantly increases a vision model's ability to localize objects. We see a similar effect with overlapping cameras in both multiview and stereo setups. Finally, we explore the use of auxiliary depth estimation tasks and find that it improves our vision only models.

References

[1] G. Brazil and X. Liu. M3d-rpn: Monocular 3d region proposal network for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, Seoul, South Korea, 2019.

[2] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[3] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6526–6534, 2017.

[4] Y. Chen, S. Liu, X. Shen, and J. Jia. Dsgn: Deep stereo geometry network for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12536–12545, 2020.

[5] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, 2018.

[6] R. Ge, Z. Ding, Y. Hu, Y. Wang, S. Chen, L. Huang, and Y. Li. Afdet: Anchor free one stage 3d object detection, 2020.

[7] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow. Digging into self-supervised monocular depth prediction. October 2019.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[10] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska. One thousand and one hours: Self-driving motion prediction dataset, 06 2020.

[11] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, 2018.

[12] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft level 5 perception dataset 2020. `https://level5.lyft.com/dataset/`, 2019.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014. URL `http://arxiv.org/abs/1412.6980`. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[15] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3d proposal generation and object detection from view aggregation. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8, 2018.

[16] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539. URL `https://doi.org/10.1038/nature14539`.

[17] P. Li, X. Chen, and S. Shen. Stereo r-cnn based 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7644–7652, 2019.

[18] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, 2018.

[19] S. Liu, E. Johns, and A. Davison. End-to-end multi-task learning with attention. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1871–1880, 2019.

[20] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan. Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6851–6860, 2019.

[21] J. Philion and S. Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 194–210, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58568-6.

[22] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017.

[23] R. Qian, D. Garg, Y. Wang, Y. You, S. Belongie, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao. End-to-end pseudo-lidar for image-based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5881–5890, 2020.

[24] C. Reading, A. Harakeh, J. Chae, and S. L. Waslander. Categorical depth distribution network for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8555–8564, 2021.

[25] T. Roddick, A. Kendall, and R. Cipolla. Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*, 2018.

[26] D. Rukhovich, A. Vorontsova, and A. Konushin. Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection. *CoRR*, abs/2106.01178, 2021. URL `https://arxiv.org/abs/2106.01178`.

[27] V. Sindagi, Y. Zhou, and O. Tuzel. Mvx-net: Multimodal voxelnet for 3d object detection. *2019 International Conference on Robotics and Automation (ICRA)*, pages 7276–7282, 2019.

[28] S. Srivastava, F. Jurie, and G. Sharma. Learning 2d to 3d lifting for object detection in 3d for autonomous vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4504–4511. IEEE, 2019.

[29] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. 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.

[30] P. Tomasello, S. Sidhu, A. Shen, M. W. Moskewicz, N. Redmon, G. Joshi, R. Phadte, P. Jain, and F. Iandola. Dscnet: Replicating lidar point clouds with deep sensor cloning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1317–1325, 2019. doi: 10.1109/CVPRW.2019.00171.

[31] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8445–8453, 2019.

[32] Y. Wang, B. Yang, R. Hu, M. Liang, and R. Urtasun. Plume: Efficient 3d object detection from stereo images. *arXiv preprint arXiv:2101.06594*, 2021.

[33] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. *arXiv preprint arXiv:1906.06310*, 2019.

[34] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4490–4499. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00472. URL `http://openaccess.thecvf.com/content_cvpr_2018/html/Zhou_VoxelNet_End-to-End_Learning_CVPR_2018_paper.html`.