Depth-Supervised Lift-Splat-Shoot for Birds-Eye-View Outdoor Road Scene Segmentation

Rolanda Hutson Sergi Widjaja Yusyuan Du
Carnegie Mellon University

1 Motivation

Perception is a critical component of autonomous driving systems. By improving the ability to extract semantic information using sensor data, safer and more reliable autonomous driving systems. In more recent years the pursuit of safe and efficient autonomous driving systems has intensified, highlighting the necessity of accurate perception models in navigating complex environments. This includes leveraging sensors, such as LiDAR and RADAR, that are essential for robust scene understanding and decision-making in autonomous vehicles. This project stems from the significant advancements made by Lift, Splat, Shoot[4] which uses multi-view camera data. By building upon this, our project aims to enhance the scene understanding by integrating depth information from additional sensors like LiDAR and RADAR. Augmenting the original method (α vector) used for depth reconstruction and incorporating a neural network architecture, we seek to improve the accuracy and robustness of bird's eye-view scene segmentation. This project is a step toward advancements in vehicle perception systems, enabling safer transportation methods.

2 Prior Work

2.1 Overview

Most planning and robotic decision-making in contemporary autonomy stacks operate in the bird's eye-view space. Traditionally, however, image-based perception modules perceive semantic features in the image space and project them back into the euclidean or bird's eye-view space for downstream processing. This paradigm typically suffers from undesirable artifacts, such as errors that are introduced from imperfect sensor calibration. It was only until the introduction of state-of-the-art sensors such as LIDARs and RADARs that the work started to shift towards the direction of explicit reasoning in 3D [2] [1].

Image-based methods somewhat lag behind in this effort towards explicit reasoning due to the nature of cameras, which does not explicitly produce semantic understanding in 3D space. One prominent work has tried to circumvent this by producing depth artifacts on the image space, which are converted into a more workable point-cloud representation [6] to be fed into point-cloud perception modules. Despite its promising performance, the training process is designed as a two-stage approach, lacking end-to-end differentiation from the original image data all the way to the output space.

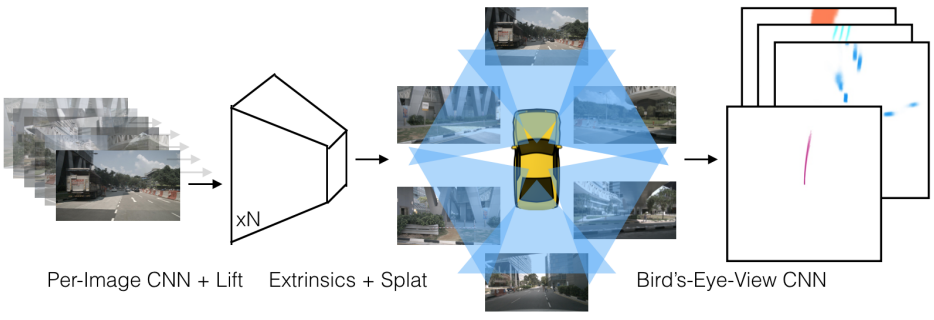

Figure 1: Lift-Splat-Shoot Network Architecture.

Our work will primarily be based on the seminal work of Lift-Splat-Shoot (LSS) [4], a method to encode image features to the bird's eye-view feature space. It is proposed as an image-only method that can be embedded into any existing learning-based perception pipeline for autonomous driving. It is one among few works [5] [3] that attempts to reason explicitly in the BEV space based on camera images. We aim to tackle the BEV segmentation task with general metrics such as class-wise IOU and mIOU.

3 Key Ideas and Proposed Extension

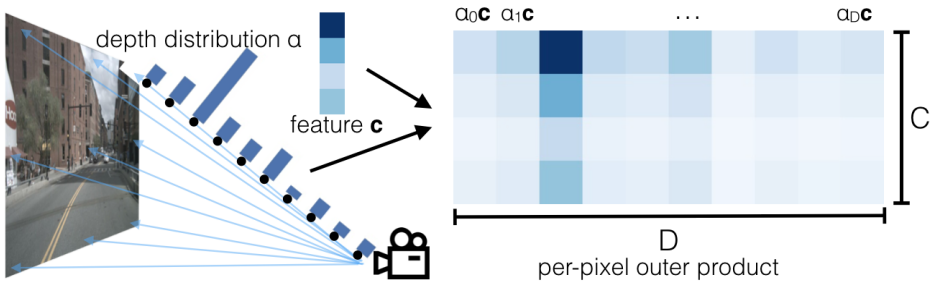

Figure 2: Soft assignment of context vector present in the original work of LSS, which we aim to augment.

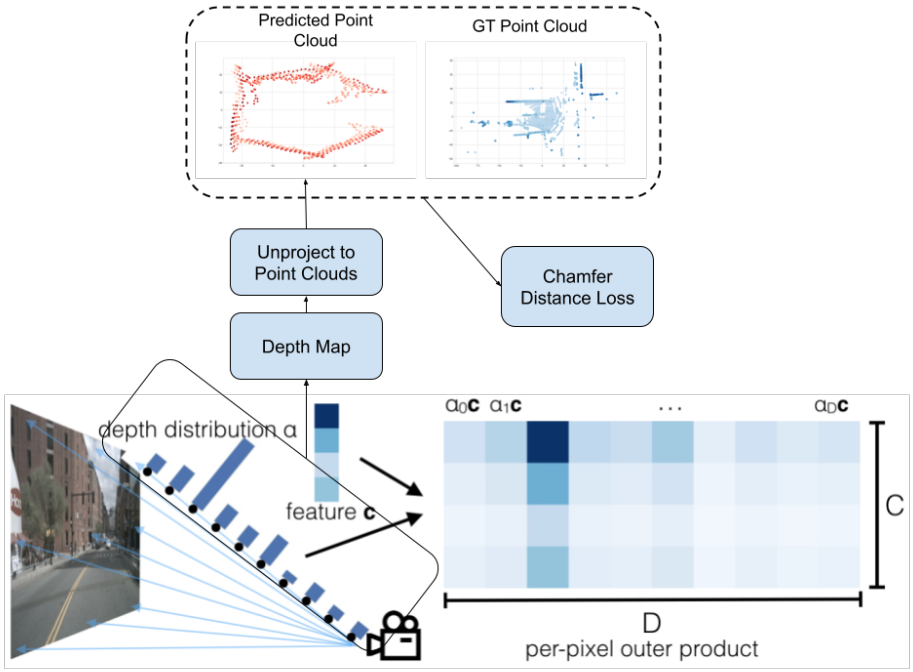

Figure 3: The depth regularization pipeline, we unproject the depth maps that are implicitly obtained from the α depth distributions. The unprojected point clouds are then compared with the ground truth readings.

A shared insight across different works in this image-based BEV reasoning is the idea of pixel features as ray features. After pre-processing with a CNN backbone, a context vector is produced for every pixel that can be assigned to a point along a ray from the camera center. Preceding work of LSS [5] assigns the context vector for every point along the ray. This is a relatively naive assumption, where every point along a ray shares the same feature which doesn't align with the physical reality of a scene. To mitigate this, one key idea used in LSS is the discretization of depth values followed by predicting a depth distribution without explicit supervision. This depth distribution is then used to assign the context feature to a correct point along the pixel ray.

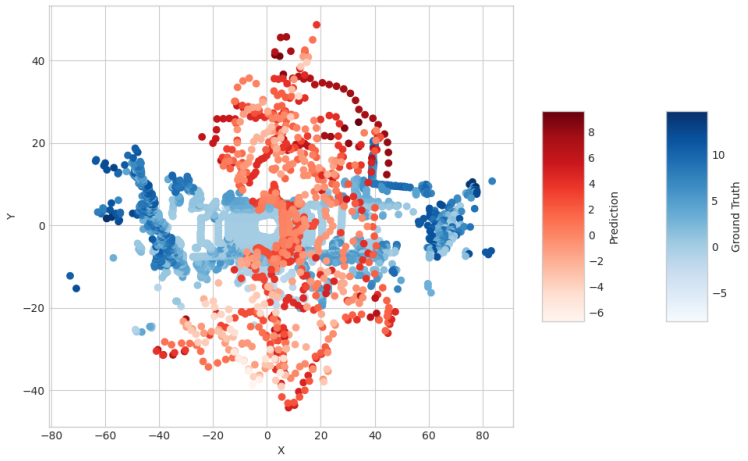

Figure 4: Mismatch between the unprojected depth values that constitute the feature assignment and the ground truth point cloud readings.

More specifically, an α vector is predicted for every pixel ray that dictates the likelihood of a pixel value equating to an arbitrary depth. A soft assignment of the context feature is then applied based on this predicted depth distribution for a more targeted feature assignment in the BEV space. Note that up to this stage, *no explicit supervision* of depth values are applied to this part of the network module.

Figure 4 illustrates our point. We trained a model with the baseline settings (i.e. without depth regularization) and we discovered how the feature assignment fails to match the physical reality captured by the LIDAR readings. More particularly, the pipeline mistakenly assigns features towards empty regions on the road. We postulate that more features should be assigned to the more physically salient regions on the road (road regions that span horizontally). By doing so, the model should be able to discriminate road objects better.

We aim to extend this process of context feature assignment by adding more explicit supervision to the depth distribution histogram. More particularly, we calculate the depth implicitly perceived by the network by calculating the sum-product of the depth bin values and its α weights. We then unproject the depth map to obtain a point cloud that we can compare with the ground truth point cloud readings collected from other sensor redundancies such as LIDAR and/or RADAR. Chamfer distance is used as a loss function to compare the two point cloud instances to regularize alpha values.

The Chamfer distance between two point sets A and B is a measure of similarity between the sets. It is defined as:

$$L_{\text{Chamfer}}(A, B) = \sum_{a \in A} \min_{b \in B} \|a - b\|^2 + \sum_{b \in B} \min_{a \in A} \|b - a\|^2$$

where:

- A and B are two sets of points in a metric space,
- $\|a - b\|^2$ is the squared Euclidean distance between points a and b.

Mathematically, we augment our model's performance by incorporating a Chamfer loss term with the standard segmentation loss controlled by a parameter λ. This addition allows for a more nuanced optimization through gradient descent, finely balancing both loss contributions.

$$L_{\text{total}} = L_{\text{segmentation}} + \lambda L_{\text{Chamfer}}$$

Intuitively, by learning to produce sensible α weight values that correspond better to the ground truth readings, the network will eventually learn to better assign the context vector to its supposed location in the BEV space. Note that despite the usage of sensor redundancies as supervision, the network only utilizes images coming from cameras during test time. Therefore, we remain consistent with the original spirit of the work, which is to explore the possibilities of camera-based perception.

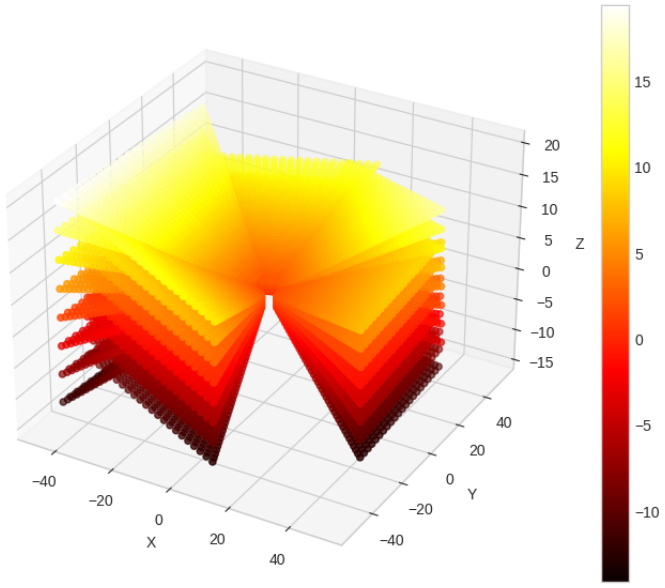

Figure 5: Sparse frustum as obtained from the Lift-Splat-Shoot model's internal representation which forms our pseudo point clouds containing our image features.

We experimented with different ways of applying depth regularization, and we found that unprojecting the depth and using the regularization in the 3D space is a straightforward and intuitive option. The reason is that the frustum discretization downsamples the image pixels into a relatively low-dimensional pseudo image. Given this downsampling operation, the physical size of the depth pixels obtained from the frustum would be inflated. Thus, applying regularization in the image space would be difficult because the task necessitates finding the correct depth pixel and/or aggregating depth pixels from the ground truth sensor redundancies that correspond to a predicted depth pixel.

Figure 5 illustrates our point. In this 3D plot, the default model's hyperparameter set produces a frustum representation that is relatively sparse. Notably, it divides the image into 8 rows cells which produces a visibly streaky visualization. While upsampling is indeed an option for more robust predictions, this will inclur additional memory overhead which is undesirable for fast predictions in autonomous systems.

Unprojecting the pixels based on the predicted depth values, on the other hand, results in a more natural problem statement where we are trying to compare the distance between two point clouds. Chamfer distance, in this case becomes an appropriate choice of loss function due to its relative flexibility.

4 Experiments

4.1 Overview

Applying the Chamfer loss in the 3D space comes with its own set of design parameters. As mentioned in Section 3. The chamfer distance loss calculates for every point in the reconstructed (reference) point set and finds the minimum distance to the reference (reconstructed) point set. By tuning the loss in a certain way, one can produce the right tradeoff between precision and recall.

Figure 6: Balancing precision and recall in depth regularization. *Top:* Maximizing recall (coverage) results in a higher degree of exploration for the feature assignment in the 3D space *Bottom:*: Maximizing precision resulted in a lower coverage of the feature assignment.

- Maximize precision: What fraction of the points in the reconstructed point cloud are close to any point in the reference point cloud? High precision means that most points in the reconstructed point cloud represent the shape well. To maximize precision, the chamfer loss term becomes

$$L_{\text{Chamfer}}(P_{recons} \rightarrow P_{gt}) = \sum_{p_{recons} \in P_{recons}} \min_{p_{gt} \in P_{gt}} \|p_{recons} - p_{gt}\|^2$$

- Maximize recall: What fraction of points in the reference point cloud have a close point in the reconstructed point cloud? High recall means that most points in the reference point cloud are well represented in reconstructed point cloud. To maximize recall, the chamfer loss term becomes

$$L_{\text{Chamfer}}(P_{gt} \rightarrow P_{recons}) = \sum_{p_{gt} \in P_{gt}} \min_{p_{recons} \in P_{recons}} \|p_{recons} - p_{gt}\|^2$$

- Balancing both precision and recall: The loss function will simply be the weighted summation of both of the above terms.

We experimented with different ways to apply the Chamfer loss. We tried a Chamfer loss that only attempted to maximize precision and recall individually, which unfortunately led to sub-optimal results. Figure 6 depicts the balance

we should strive for when tuning the chamfer loss for a precise yet comprehensive feature assignment coverage. Thus, we will be constraining ourselves to use only the Chamfer loss variant that maximizes *both* precision and recall (bidirectional).

4.2 Results

Setting	IOU	Chamfer Loss
Baseline ($\lambda = 0$)	32.66	58.61
Lower weight ($\lambda = 5e-3$)	30.66	22.17
Higher weight ($\lambda = 1e-2$)	22.22	22.54

Table 1: Loss settings and their performance outcomes.

As described, we modulated the effect of the chamfer distance term on the total loss function with the parameter λ. We applied the same hyperparameters across different experiments and only modified the λ values across experiments. As shown in Table 1, adjusting the weight parameter upwards λ influences both the IOU and Chamfer Loss values fairly negatively.

A potential reason on why this is the case is the λ values that are too high. Particularly, we see how applying a higher loss weight associated with the point cloud chamfer loss does not necessarily result in a lower chamfer loss. Had we chosen a loss weight value that are significantly lower in magnitude, there's a possibility that we might have found favorable results.

Furthermore, the task in which the network is optimized for, is the task of vehicle segmentation. We postulate that the task is slightly too simple for any form of regularization to result in any positive effect. There might be some positive results on applying this method for other more fine-grained task. Consider the case if we are operating in a multi-class setting where the network would encounter a higher variety of objects in their dataset. More objects can be placed in less predictable location as opposed to the vehicle agents which are fairly likely to appear in road bodies. The depth regularization mechanism might be of use in this setting.

5 Conclusion

Overall, despite the negative results, we have strong reasons on why our proposed method didn't pan out as expected. The theoretical grounding and the intuitive nature of the method provided a strong motivation for the experiments. After tying up some loose-ends in the implementation details, we are confident that this can provide some baseline and insights on using LIDAR as priors to build camera-only ML-based perception systems.

There are a number of things that we can do to augment the merits of our project. To recap, we found there are multiple reasons on why our method didn't work as expected. One reason is the fact that we are overweighting the contribution of the chamfer loss term, another potential reason pertains to the complexity of the task on hand. We want to mitigate these potential reasons and validate our assumptions in future iterations of this project

Furthermore, we'd like to supplement our analysis of depth regularization by incorporating the sparser RADAR data into the project pipeline and comparing it to the denser LIDAR data. We also want to see how superimposing LIDAR and RADAR readings compare with our current implementation.

Lastly, we'd like to explore the potential of eliminating the need to use any form of ground truth collected from sensor redundancies. We will shift our attention towards incorporating pre-trained depth-estimation modules, which process the image features to produce reasonably accurate depth predictions, which will end up in our depth regularization pipeline as a pseudo-ground truth.

References

[1] Jason Ku et al. *Joint 3D Proposal Generation and Object Detection from View Aggregation*. 2018. arXiv: 1712.02294 [cs.CV] (cit. on p.).

[2] Alex H. Lang et al. “PointPillars: Fast Encoders for Object Detection From Point Clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019 (cit. on p.).

[3] Kaustubh Mani et al. “MonoLayout: Amodal scene layout from a single image”. In: *The IEEE Winter Conference on Applications of Computer Vision*. 2020, pp. 1689–1697 (cit. on p.).

[4] Jonah Philion and Sanja Fidler. “Lift, Splat, Shoot: Encoding Images From Arbitrary Camera Rigs by Implicitly Unprojecting to 3D”. In: *Proceedings of the European Conference on Computer Vision*. 2020 (cit. on p.).

[5] Thomas Roddick and Roberto Cipolla. *Predicting Semantic Map Representations from Images using Pyramid Occupancy Networks*. 2020. arXiv: 2003.13402 [cs.CV] (cit. on p.).

[6] Yan Wang et al. *Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving*. 2020. arXiv: 1812.07179 [cs.CV] (cit. on p.).