ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

RSL Robotic Systems Lab
Prof. Dr. Marco Hutter

Master Thesis

Efficient Multimodal Environment Representation for Robot Navigation

Spring Term 2024

Supervised by:
Dr. Vaishakh Patil
Fan Yang
Pascal Roth

Author:
Sapar Charyyev

Contents

Abstract

Autonomous robots are heavily dependent on perception modules to interpret their environment in real time, enabling safe navigation for tasks such as driving, search and rescue, and food delivery. These modules utilize data from various sensors such as cameras and LiDAR to generate semantic and geometric representations of the environment. Although cameras provide dense visual information, they lack depth and LiDAR provides accurate depth but suffers from sparsity. Multimodal perception, which combines both sensors, overcomes these limitations, but poses challenges such as computational inefficiency and the reliance on large labeled datasets.

In this thesis, we propose a novel, efficient multimodal architecture that leverages range image representation for LiDAR point clouds and triplane representation for unified 3D scene understanding. Our model encodes camera and range images through lightweight 2D encoders, utilizing an interaction module to exchange depth and semantic information. The fused modality-specific features are then mapped to a triplane representation using a simple yet effective self-attention based fusion module.

To address the data inefficiency of multimodal models, we introduce a novel self-supervised pretraining framework. Our approach combines a surface reconstruction task to learn geometrically meaningful features and a multi-positive contrastive learning method for semantically meaningful features, using the Segment Anything Model to create robust multimodal supervision.

We evaluated our model on the semantic occupancy prediction task, achieving competitive performance among multimodal methods while being five times faster. This work demonstrates that our proposed architecture and pretraining method are effective for dense 3D perception in autonomous systems. The code is available here.

Chapter 1

Introduction

Autonomous robots are being utilized in many tasks such as driving, search and rescue and food delivery. One of the core parts of these is the perception module; robots need to be able to understand their environment in real time to navigate safely. Perception module relies on one or multiple sensors such as camera and lidar. This module takes input from these sensors and returns geometric and semantic information of the environment that robot can use to navigate.

Cameras are relatively cheap sensors that give dense visual information. That is why it is widely used in robotics to solve many perception tasks such as object detection and semantic segmentation. Task of object detection is to localize and classify multiple objects in an image [1, 2, 3] while segmentation is concerned with grouping pixels to one of several classes [4, 5]. Semantic segmentation is more general task than object detection because it requires to produce accurate boundaries of objects and some things such as roads cannot be described by a bounding box.

However, these tasks produce results in 2D pixel space, while robots operate in 3D space. Thus, it is essential to know the 3D location of objects for safe navigation. Some works use single camera to do 3D tasks such as monocular 3D object detection and depth estimation. Goal of monocular 3D object detection is predict 3D pose of objects using single camera. Some works [6, 7] extend 2D detection by adding extra component to predict 3D information, while some works [8, 9] use 3D priors. Monocular depth estimation models try to predict per pixel depths of an image. Main challenge of monocular depth estimation is scale ambiguity due to lack of 3D information. Many works [10, 11] regress per pixel depth in an end-to-end manner with idea that ambiguity will be resolved by observing more data. Depths values can then be combined with semantic classes to infer 3D position and semantics of objects in a scene.

In many scenarios, robots need to perceive their surrounding from multiple angles for safe navigation which is not possible with single camera. One common way to achieve this is to strategically equip robots with multiple cameras to perceive the entire surrounding. This poses new challenges for perception as applying models from monocular setting directly to multi-view does not yield best results. For instance, monocular 3D object detection model can be applied to each image separately and the results are combined with post processing techniques. However, using post processing to combine results prevents the model from learning how to fuse information from multiple cameras in the best way [12]. Thus, state-of-the-art works project the 2D image features to the common representation such as birds-eye-view [12, 13, 14], volumetric [15, 16, 17] and triplane [18].

Alternatively, LiDAR sensor can be used to perceive the surrounding. Compared to the camera, LiDAR provides reliable depth information that can be used for 3D perception tasks such as 3D object detection [19, 20, 21, 22] and pointcloud

segmentation [23, 24, 25]. Since pointcloud is an unordered set, it is necessary to represent the points in some format before it can be processed by neural networks. One line of work [26, 27] processes unordered points directly to achieve the tasks point set classification and segmentation. However, these models need to be permutation invariant to process unordered sets which limits the architecture that can be used. Furthermore, this methods are very slow and not suitable for autonomous navigation [19]. Most common representation of pointcloud is to divide the scene into equal sized 3D voxels and each voxel will be assigned a vector representing statistics of points that belong to that voxel [28, 19, 20, 22, 29, 30]. However this representation is memory inefficient and requires complex 3D convolutions[19] or transformers [31] to process. Another representation of pointcloud is range image, which is compact 2D represenation. This is memory friendly, and can be processed by standard 2D convolutions. However, this is not much utilized because voxel representation achieves better accuracy [32, 33].

Even though both lidar and multi-view cameras can be used for 3D tasks, both of these modalities have their shortcomings. Camera gives dense visual cues, however they lack depth information. Lidar gives depth information but it is sparse and it has variable point density. Using both modalities together helps alleviate these shortcomings; images lack depth information which can be obtained by lidar pointcloud and lidar pointcloud represents objects with sparse points while image represents them densely. So, several works use both of these modalities for 3D perception to achieve better results. Most methods project modality specific features to unified representation for fusion. [34, 35] project both LiDAR and camera features into birds-eye-view representation. [29, 30] projects both modality features into volumetric representation. However, BEV representation cannot handle complex 3D structures and volumetric representation is memory and computation inefficient. Recently, triplane representation [18, 36] became popular alternative representation that uses three planes to represent the scene, one plane for each view. It combines best of both representations, unlike volumetric representation it is memory friendly and unlike BEV, it can represent complex 3D structures. However, this representation has not been utilized in multi-modal setting, it has been used with cameras only [18] or lidar only [36].

One of the main challenges of multimodal models that prevents it from being utilized in autonomous robots is that they are very inefficient. The reason for inefficiency lies on the choice of representation of pointcloud and 3D scene. For this, we propose novel efficient architecture that uses range image representation for pointcloud and triplane for unified scene representation. This model takes camera images and range image as input and encodes them with simple 2D encoders. There is an interaction module between these modalities, it augments image features with depth information and range image with semantic features. These modality specific features are then mapped to triplane with simple fusion module.

Another challenge of multimodal perception models is that they are data hungry; they rely on extensive labeled datasets. Self-supervised pretraining has shown promise in reducing the reliance of models on labeled datasets. It utilizes vast amount of unlabeled data to train the model on pretext task. The pretext task can be anything that supervises the model to learn useful features for downstream task. The weights learned during pretraining are then used as starting point for downstream tasks.

Pretext tasks for pretraining generally depend on input modality. Methods based on image input can be categorized into two main categories, contrastive methods and masked autoencoding methods. Contrastive methods [37, 38] augment images and pull together representation of same images together and push apart representation of other images. Masked Autoencoders [39] work by masking some part of the image and reconstructing it as a pretext task. These methods

have also been extended to lidar modality. [28] converts pointcloud into voxels and reconstructs some properties of points lying in masked voxels. [40] creates two views of the same scene and supervises it with contrastive loss.

Multimodal pretraining methods are mostly inspired by their single modal counterparts. [29] voxelizes the pointcloud and masks some voxels and some image patches. These masked inputs are processed with respective encoders and features of each modality are projected into common volumetric representation for fusion. The unified representation with fused features are then used to reconstruct both masked image patches and voxels. However, reconstructing points overfits to the lidar configuration and how to best project features from unified representation to image planes is unclear. Similarly, [30] masks some voxels and image patches and projects the extracted features to unified volumetric representation. They use neural rendering to render multi-view images as a supervision. This does not use lidar for supervision and neural rendering is very computationally expensive.

Multimodal setting is inherently different than single modal setting. Features for single modal models are always in same modality, however multimodal models need to project the features into common representation. That's why single modal pretraining tasks may not learn good 3D representation. In this work, instead of relying on input reconstruction tasks, we propose to pretrain the model to explicitly learn semantically and geometrically meaningful features. To learn geometrically meaningful features, we adapt surface reconstruction task from [41]. We use visibility information of lidar to construct points that are empty and that belong to surface. This points are then used as supervision for the model. To learn semantically meaningful features, we propose novel method utilizing Segment Anything Model [42] and multi-positive contrastive loss [43]. We first segment all the surrounding images with SAM [42] and project each lidar points to each image and group the points that belong to the same segment. This group information is used to supervise the model with multi-positive contrastive loss [43].

We demonstrate the effectiveness of our architecture and pretraining method on semantic occupancy prediction task [16]. Our pretraining method achieves better result than input reconstruction method and our model achieves better results than camera only methods. Our model is competitive with multimodal state-of-the-art methods for occupancy prediction while being 5 times faster.

The contributions of this thesis are as follows:

- We are the first to use range image representation of pointcloud for dense 3D perception.
- We are the first to use triplane representation for multimodal perception.
- We propose very simple module based on self-attention to lift modality specific features to triplane representation.
- We propose novel self-supervised pretraining method to learn semantically meaninful features.
- We adapt surface reconstruction task [41] from single modality to multimodal setting.

Chapter 2

Related Work

2.1 Pointcloud Representation

Point representation. Lidar produces geometry of a scene as unordered set of points. Point based methods pioneered by [26, 27] process the unordered pointcloud directly and extracts features for each point. These methods satisfy the permutation invariance property that is necessary to process unordered sets. However this limits the choice for architecture and previous works on CNNs [44, 45, 46] and transformers [47, 5, 48] cannot be utilized. Furthermore, this representation is more computationally demanding than processing pointcloud with other methods such as CNNs [49, 19]. Finally, this point representation produces a feature for each point and there is no straightforward way map it to dense representation.
Volumetric representation. Another common way to represent pointcloud is to divide the scene into uniform 3D voxels and represent each voxel with the features of points belonging to that voxel [19, 22, 21, 20]. Methods that use this representation achieve good results in multiple tasks. However, processing voxel representation requires 3D convolutions or transformers that grow cubically in memory and computation requirement.
Range image representation. Pointclouds are inherently 2.5 dimensional, only points belonging to the surface of an object facing lidar return points. Thus, it is possible to represent the pointcloud in 2D by projecting points to spherical coordinates [33]. This representation is under utilized in community because of information loss from 3D-2D projection [36] and objects are not distance invariant after projection [32]. We choose range images as our pointcloud representation due to its compactness, and show that it can be mapped to dense representation of 3D scene.

2.2 3D Scene Representation

Volumetric representation. Some works represent 3D scene by splitting it into voxels and assigning a vector that describe the voxel [29, 30]. Methods based on voxel representation perform very well on several tasks, however this comes with computation and memory burden. This representation requires complex 2D to 3D mapping for image modalities [12, 34], and inefficient volumetric representation of pointcloud.
Birds-eye-view (BEV) representation. Vast number of voxels pose challenges to the computation efficiency of models. So, in self-driving cars people utilize birds-eye-view (bev) [34] representation that collapses the height dimension of voxel representation. This works well for self-driving as the road is mostly flat. This methods perform well in 3D object detection and bev segmentation tasks, however this rep-

resentation is not adequate for offroad navigation that might include objects with complex 3D structure.

Triplane representation. For offroad navigation, we need to be able to represent detailed height information. Recently, triplane [18] representation became popular in view reconstruction [50, 51] and autonomous driving [18, 36] for its efficiency. Triplane representation is compromise between bev and voxel representation, where instead of using one plane to represent scene as in bev, it uses three planes to represent a scene. Each plane represents one view of the scene and thus it can capture cetailed height information. Unlike voxel representation, this only requires $O(N^2)$ memory. We choose this as our representation for its efficiency.

2.3 Projections

2D-3D lifting. For common representation, it is necessary to lift 2D features into unified representation. [12] predicts discrete depth probability for each pixel and lifts these features to 3D with linear combination of these predicted probabilities. [52] projects 3D points to 2D image plane and uses cross attention to combine neighboring features. [53] claims cross attention is not necessary, and projects 3D points to 2D and assigns feature of that pixel.

Volumetric-BEV-Triplane. Works that use BEV or triplane representation first project features to volumetric representation. [53] reshapes height dimension and feature channels to a single channel of volumetric representation to obtain BEV representation. [52] projects volumetric features to BEV by taking average over height dimension. [18] similarly obtains features for uniformly sampled 3D points, and uses cross-view hybrid attention to project the features to triplane. [36] first obtains volumetric representation of pointcloud and projects them to triplane with max pooling.

Even though some of these methods use efficient representation of the scene such as BEV or triplane, they require lifting of 2D image features to volumetric representation which is both slow and memory demanding. Our method uses simple efficient self-attention module to map 2D features to triplane without explicitly projecting features to volumetric representation first.

2.4 Unimodal Pretraining

Self-Supervised learning is the task of training the model with pretext task using unlabeled data. The pretext task can be anything as long as it encourages model to learn useful features.

Image representation learning. The most common self-supervised learning methods for image modality are based on Masked Autoencoders (MAE) [39] and contrastive learning. MAEs work by dividing the image into patches and masking some of the patches. The pretext task is then to reconstruct masked patches of the input. Contrastive learning aims to learn visual features by pulling similar data points together and pushing dissimilar data points apart. SimCLR [37] works by augmenting images and pulling together augments of same image while pushing away other images in a minibatch. MoCo [38] further creates queue of previous representation for more consistent representation. These works require negative samples and how you choose negative samples affect the performance. BYOL [54] removes the need for negative samples by deploying two networks. Two views of same image are created and these networks are required to predict same representation for these images. One of the networks is updated by exponential moving average to deal with collapse.

Pointcloud Representation Learning. There has been several progress on self-supervised learning from pointcloud. [28] adapts MAE for voxel based representation, which works by masking some voxels and supervising the model to predict positions of points, number of points in each masked voxel and whether the voxel is empty. [55] applies similar approach, but instead of supervising with lidar specific task, it supervises the model with geometric features such as centroid, normal and curvature of points. [40] creates two views of the same scene and supervises it with contrastive loss. [41] adapts surface reconstruction as a pretext task. It creates target values in the form of empty and non-empty points utilizing properties of lidar sensor.

2.5 Multimodal Pretraining

Several methods extend single modal pretraining to multimodal setting. [56] adapts the MAE to pointcloud and image modality in a direct way. They mask some input points and image patches before feeding them through encoders. Encoded features of different modalities are concatenated and fed through the shared encoder for information exchange. One decoders is adapted to reconstruct each modality. This method assumes that there is a single image and pointcloud is generated from depth sensor which is not suitable for autonomous navigation. [29] supervises the model with voxel MAE[28] and MAE [39] simultaneously. This method first projects modality specific features into unified volumetric representation. Unified features are then projected into each modality for reconstruction. Similarly [30] projects features to unified volumetric representation and supervises the model with multi-view reconstruction with neural rendering. These methods utilize input reconstruction task for pretrainig, which may not learn useful 3D features. We propose tasks that specifically supervise the model to learn geometrically and semantically meanginful features of the scene.

2.6 Semantic Occupancy Prediction

3D semantic occupancy prediction gained increasing attention over the years because of its flexibility in representing objects in detail. Existing works predict semantic occupancy using multi-view cameras, lidar only, and multi-view cameras and lidar combined.

Cameras. Among works that use cameras only, [16] extracts multi-scale features from images and maps them to 3D voxel space using cross attention. 3D representation is then progressively upsampled by 3D convolutions. [18] extracts image features with 2D backbone and lifts them into triplane with cross attention. The decoder is multi-layer perceptron that takes feature of center of voxel as input and predicts semantic class. [17] trains semantic occupancy using only 2D labels. More specifically, they extract image features and map them to unified 3D representation. Then they employ NeRF style rendering as supervision.

Lidar. Among lidar based methods, [36] projects point features into triplane and process each plane with same 2D encoder and FPN.

Lidar and cameras. State-of-the-art results in semantic occupancy prediction are achieved using both cameras and lidar. [57] extracts multi-scale image features and maps them to BEV and 3D voxel space. The features are then fused with 3D lidar features. [58] also extracts image features and maps them to 3D voxel space and fuses with lidar features.

Chapter 3

Proposed Method

Our goal is to have efficient multimodal environment representation that achieves good results on 3D tasks. For efficiency, we propose novel architecture that takes range image representation of pointcloud and camera images as input. They are encoded with their respective encoders and this encoded representation are fed to the fusion module that directly predicts three planes to be used as triplane representation. To improve accuracy, we propose multimodal pretraining method that explicitly supervises the unified features to be geometrically and semantically meaningful. To learn geometry of a scene, we adapt surface reconstruction task from [41] as our pretext task. For semantics, we propose a novel method that utilize Segment Anything Model (SAM) [42] to cluster the lidar points and supervise model with multi-positive contrastive loss [43]. We describe these in detail in the following sections starting from efficient architecture.

3.1 Architecture

Our multimodal architecture takes images and range image as input and encodes them with their respective encoders, and each modality features are augmented with complementary information. These features are then fed to the fusion module that predicts triplane as shown in figure 3.1.

3.1.1 Range Image Representation

Lidar only returns sparse points that lie on the surface of objects. So, is not necessary to to use entire 3D scene to represent the pointcloud. It can be represented in the form of range image, which works by projecting lidar points into spherical coordinates and representing them in 2D where height dimension is vertical angle, and width dimension is horizontal angle. The values of pixels will contain information belonging to that point. In our case, the pixel values are just distance from lidar to the point. More specifically, for each point $p = (x, y, z)$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}[1 + \arctan(x, y)\pi^{-1}]w \\ [1 - ((\arcsin z\rho^{-1} + |f_{\text{down}}|)f^{-1})h) \end{pmatrix} \tag{3.1}$$

where h, w are height and width of range image, $\rho = \sqrt{x^2 + y^2 + z^2}$, $f = f_{\text{up}} + f_{\text{down}}$ is vertical field of view of lidar. u, v are pixel coordinates of range image. So, range image is collection of triplets (u, v, ρ). In general, position of robot will change before the lidar can finish one sweep and this will lead to several points belonging to same pixel in range image. In this case, we take the closest point to the lidar.

3.1.2 Encoder

Range image from lidar and images from surronding view cameras are processed by specific encoders. To encode images, we adapt Convnext [44] with some modifications. After feeding images through two blocks of Convnext, there is two way information exchange between range image and image features. The goal of this exchange is to provide the information each modality is lacking. Image features lack depth information, so we add positional encoding of lidar points that map to that feature. On the other hand, lidar points lack semantic information and we augment range image with semantic information by concatenating image features with range image. More specifically, let $\mathbf{f}_2$ be the feature map of an image after passing through second layer of Convnext. For each point $p = (x, y, z)$, we obtain its coordinates in range image (u, v) as described in equation 3.1. We can further use intrinsic and extrinsic matrices of camera to project p into feature space, which we denote with (u', v'). Then we augment depth information to image feature by

$$\mathbf{f}_2'[u', v'] = \mathbf{f}_2[u', v'] + \text{MLP}(x, y, z)$$

and we augment semantic information to range image by

$$R'[u, v] = \text{concat}(R[u, v], \mathbf{f}_2[u', v'])$$

Then modified image features $\mathbf{f}_2'$ is fed through the rest of the layers. To process the modified range image, we modify Convnext model to make it more suitable to process range images. Convnext was designed to process images which have more or less similar height and width dimension. On the other hand, width dimension of range image is significantly bigger than its height; for example range image we obtain from Nuscenes dataset [59] is 32×1024. Inspired by [23], we modify Convnext so that it only downsamples features in width dimension. This helps with retaining information in height dimension, and we do not need to worry about receptive field because height dimension of range image is generally small. Modified range image R' is processed by this modified encoder to generate range image features.

3.1.3 Fusion

Inspired by [50, 51], we directly predict features in the shape $(96, 128, 128)$ and reshape it in feature dimension $(3, 32, 128, 128)$ to obtain three planes. For that, features from both range image and camera images are processed by single Mix Vision Transformer [5] layer. Mix Vision Transformer is an encoder that is

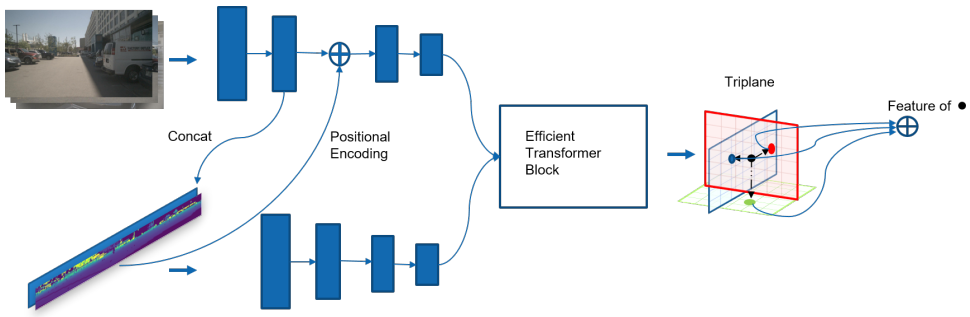


Figure 3.1: **Proposed Architecture.** Takes multi-view images and range image as input. They are processed by respective encoders with interaction among modalities. Final features are fed to fusion module that projects them to triplane.

specifically designed to be efficient. This layer has an overlapping patch merging module that takes image as input and converts them into patches. These patches are processed by efficient self-attention block, which uses sequence reduction technique to reduce the number of sequences. The processed sequences are then fed through Mix-FFN that combines MLP with convolutions to remove the necessity for positional encoding. These are then mapped to triplane with pixel shuffle [60] and convolutions.

More specifically, we add position embedding [61] to range image features of the shape $(C, 32, 32)$ and image features of the shape $(6, C, 16, 32)$. These features are then concatenated in spatial dimension to obtain features of shape $(C, 64, 64)$. These features are fed to the Mixed Vision Transfromer layer and upsampled using pixel shuffle to obtain features of shape $(256, 128, 128)$ which is then processed by convolutions to obtain features of shape $(96, 128, 128)$.

3.1.4 Feature Sampling

Triplane representation is analogous to having one plane to represent each view. Since we did not explicitly define geometry of triplane, how we sample features will define what features each plane will learn. In this work, we treat triplane as three axis-aligned orthogonal planes with center conciding with the center of lidar. Then, for any point $p = (x, y, z)$ in 3D space, we can obtain its feature by projecting it to each plane

$$p_x = \lfloor \frac{x - r_x}{v_x} \rfloor$$

$$p_y = \lfloor \frac{y - r_y}{v_y} \rfloor$$

$$p_z = \lfloor \frac{z - r_z}{v_z} \rfloor$$

where (r_x, r_y, r_z) is lower bound of 3D scene, (v_x, v_y, v_z) is plane resolution. Then we sample features from each plane

$$\mathbf{t}_{xy} = S(T_{xy}, (p_x, p_y))$$

$$\mathbf{t}_{yz} = S(T_{yz}, (p_y, p_z))$$

$$\mathbf{t}_{xz} = S(T_{xz}, (p_x, p_z))$$

where (T_{xy}, T_{yz}, T_{xz}) are three planes of triplane and S is sampling method: in this case it is bilinear sampling. Finally, the feature of (x, y, z) is

$$\mathbf{f}_{xyz} = \mathbf{t}_{xy} + \mathbf{t}_{yz} + \mathbf{t}_{xz} \tag{3.2}$$

3.2 Geometric Supervision

For geometric supervision, we adapt surface reconstruction task from ALSO [41]. Unlike other methods that learn sensor patterns [29, 28], this method learns actual surface of environment with sparse supervision. This method is not specific for any lidar which makes it desirable pretraining task.

3.2.1 Query Generation

First step is to generate target values that will be used to supervise the model, we call these query points. Queries are in the form of 3D points with binary labels: empty or surface. To generate these points, we utilize properties of lidar sensor. Let c be lidar location, and p be any point in pointcloud. Then, any point on the segment $[c, p]$ does not belong to a surface. We create two query points from that segment with label 0, one is randomly chosen point on the segment $[c, p]$ and the other is $p - \delta_p$, where $\delta_p = \delta u$ with $u = (c - p)/||c - p||$. For query points with value 1, we make an assumption that $p + \delta_p$ belongs to a surface for small δ and randomly sample a point on the segment $[p, p + \delta_p]$ as a target with value 1.

3.2.2 Surface Reconstruction

We then use these targets to supervise the model. One straightforward way to supervise the model would be to sample the features at the query's location from triplane and train it as binary classification task. Even though this is reasonable task to learn geometrically meaningful features of the scene, ALSO [41] goes one step further and uses feature of single point to predict occupancy of all points around some radius of that point. This will encourage features to be similar of nearby points adding semantic meaning to the features.

For this, we need to first define which points we want to supervise, and we call these points support points. This is necessary because triplane can represent any point in 3D space and there are infinitely many options. Support points can be any points that are important for the downstream tasks. For example, we can randomly sample points in a grid for object detection and we can use lidar points for pointcloud segmentation. For our case, we randomly sample some points from pointcloud as support points because pointcloud represents surface of objects which are important for any task. Then, for any support point s, we sample its feature $\mathbf{t}_s$ from triplane using equation 3.2. Then for each query q that lies within r radius of s, i.e with $||q - s|| < r$, we concatenate $\mathbf{t}_s$ with $q - s$ and feed it to the decoder that consist of MLP and sigmoid activation. This produces prediction $\hat{q}_{q|s}$ in $[0, 1]$ and is supervised with the label of q with cross entropy loss. Doing this for all support points, we get the following loss:

$$\mathcal{L}_{occ} = \frac{-1}{|S|} \sum_{s \in S} \frac{1}{|Q_s|} \sum_{q \in Q_s} o_q \log(\hat{q}_{q|s}) + (1 - o_q) \log(1 - \hat{q}_{q|s})$$

where $|S|$ is the number of support points and $|Q_s|$ is the number of query points for point s and o_q is the label of query point q.

3.3 Semantic Supervision

To learn semantically meaningful features of the environment, we propose novel pretraining method that utilize Segment Anything Model [42] to group points belonging to the same object in pointcloud together and supervise the model with multi-positive contrastive loss [43]. This process is described in figure 3.2.

3.3.1 Image Mask Generation

Segment Anything Model (SAM) [42] is a foundation model for segmentation. Inspired by recent success in NLP, SAM utilizes prompts to be able to segment any object. Prompts can be anything that tells the model what to segment, in this specific case the authors trained SAM with three types of prompts; points,

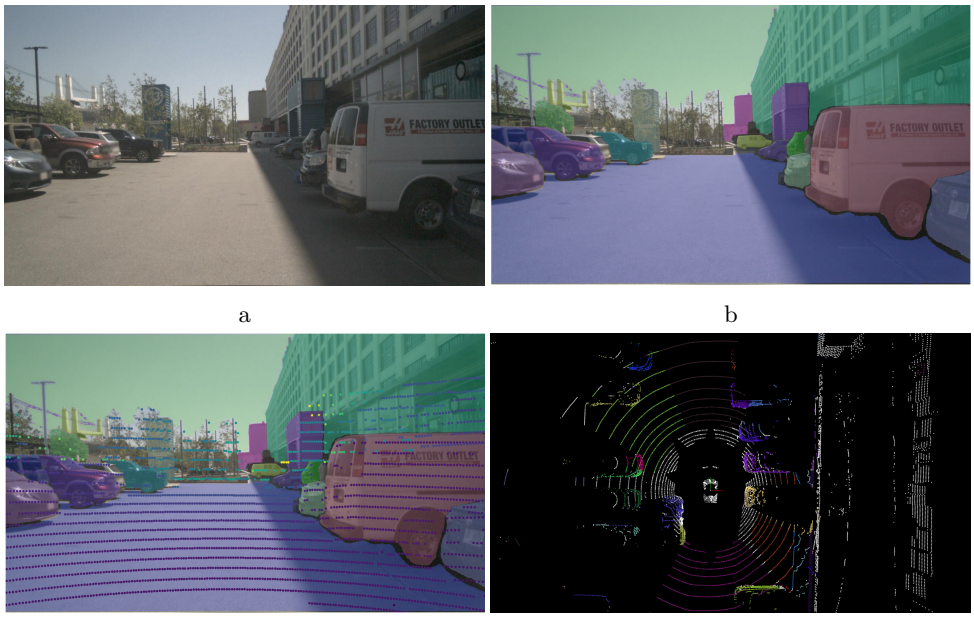


Figure 3.2: **Pointcloud clustering.** Original image a) is segmented by SAM b). Pointcloud is projected to image c) and points that projected to same segment are grouped together d).

bounding boxes and text. The model was trained with around one billion object masks and it has been demonstrated that it can even segment the objects that is not seen during training.

Automatic mask generation is the process of segmenting entire image using SAM. In this process, point prompts are sampled uniformly over the image and masks for each of these point prompts are generated. There will be a lot of overlap between generated masks and postprocessing techniques such as non-maximum suppression are used to remove duplicates and obtain clean masks.

Directly applying this technique to images from cameras on the robot gives very detailed segments, i.e. every window of a car is segmented as different object as shown in figure 3.3. The reason for this is that SAM predicts three masks for each prompt. This was done to resolve ambiguity, if point prompt lies on a window of a car it is ambiguous whether it means we want to segment that window or whether we want to segment entire car. However, we do not want that detailed segments for our representation. We are more interested in learning semantic class of an object, i.e. we want to have one segment for entire car rather than having different segments for different parts. To achieve this, we modify the automatic mask generation process of SAM. SAM predicts three masks for each point prompt, and biggest of these mask is the mask for the entire object that we are interested in. So, rather than using all three masks for each prompt, we use only the largest one and keep everything same in automatic mask generation process. With this small modification, we get masks that we desire.

3.3.2 Point Clustering

We apply image mask generation from previous section to every image of multi-view cameras. Then we project lidar points to each image, which can be done using intrinsic and extrinsic matrices of cameras. We then group the points together that

is projected to same segment in an image. We need to do this entire process once and save group information for each point.

3.3.3 Loss Function

Segment Anything Model only gives group information, it does not tell what is the semantic class of that object. So, after point clustering process, we will only have information regarding which points form a group together. One way to supervise the model with group information is to use contrastive loss, which works by pulling together the representation of similar points and pushing apart the representation of dissimilar points. Since we have multiple points that belong to same object, we adapt multi-positive contrastive loss [43] to supervise the model to predict similar representations for points belonging to same cluster. Multi-positive contrastive loss is modified counterpart of the single positive contrastive loss to deal with having multiple positive objects. Specifically, we choose an anchor point from each cluster randomly. Let $a = \{a_1, a_2, ..., a_K\}$ be the triplane features of anchor points and let $b = \{b_1, b_2, ..., b_N\}$ be the features of rest of the points that belong to a cluster. We first compute contrastive categorical distribution q_i that describe how likely a_i is to mach each b.

$$q_{ij} = \frac{\exp\left(a_i \cdot b_j / \tau\right)}{\sum_{n=1}^{N} \exp\left(a_i \cdot b_n / \tau\right)}$$

where $\tau \in \mathcal{R}_+$ is temperature parameter and a and b are all ℓ_2 normalized. And the ground truth categorical distribution p is

$$p_{ij} = \frac{\mathbb{1}_{\text{match}(a_i, b_j)}}{\sum_{n=1}^{N} \mathbb{1}_{\text{match}(a_i, b_n)}}$$

where $\mathbb{1}_{\text{match}(\cdot,\cdot)}$ is an indicator function that indicates whether anchor and candidate belong to same cluster. Then the multi-positive contrastive loss is defined as

$$\mathcal{L}_{cl}(a, b) = -\sum_{i=1}^{K} \sum_{j=1}^{N} p_{ij} \log(q_{ij}) \tag{3.3}$$

In a lot of the cases, one camera sees one part of an object and another camera sees the other part. In these cases, same object will have multiple clusters and directly using loss function 3.3 will give incorrect supervision. So, we apply the loss function to points in each camera view separately and add them. Let $C_1, C_2, ..., C_M$ be the groups of points visible in each camera C_i where M is the number of cameras. Then the final loss for learning semantic features is given as

$$\mathcal{L}_{seg} = \sum_{m=1}^{M} \mathcal{L}_{cl}(M_i)$$

a b c

Figure 3.3: **SAM automatic mask generation.** a) Original image. b) SAM automatic mask generation. c) Our modified automatic mask generation.

The final loss for pretrainig is sum of these two losses:

$$\mathcal{L} = \mathcal{L}_{occ} + \mathcal{L}_{seg}$$

Chapter 4

Experimental Setup

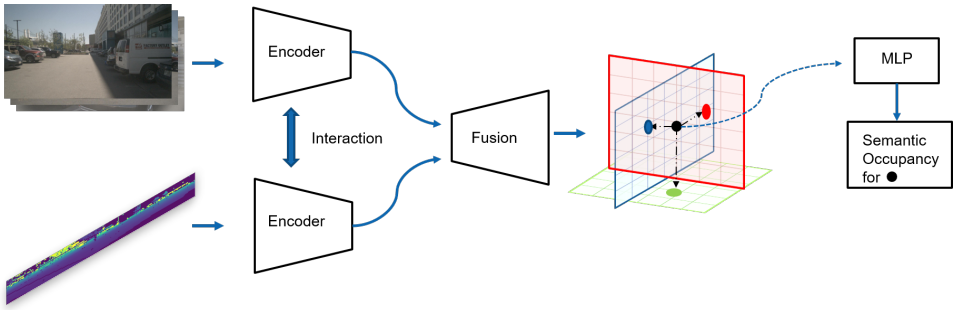


Figure 4.1: **Semantic Occupancy Prediction.** We add simple MLP decoder to extend our architecture for semantic occupancy prediction. We sample feature from triplane for center of every voxel and predict semantic class.

4.1 3D Semantic Occupancy Prediction

4.1.1 Task Description

We demonstrate the effectiveness of our method on 3D semantic occupancy prediction task. In this task, the 3D scene is divided into equal sized voxels and the model is required to predict semantic class of the voxel and whether the voxel is empty. The choice of this task is motivated by several reasons: to do well in this task, our representation needs to be dense, and the features of the represenation need to be geometrically and semantically meaningful. Furthermore, most 3D tasks are subtasks of semantic occupancy prediction: it predicts accurate bounds of objects so it is more challenging than 3D object detection and pointcloud segmentation is sparse version of this.

4.1.2 Dataset and Metrics

We use NuScenes[59] for both pretraining and finetuning our models. This dataset encompasses 1000 sequences, 700 for training and 300 for validation and testing. It is equipped with multiple sensors, and we use all of 6 cameras and the lidar for our model.

We train our models using labels from [16]. These labels are dense, and since acquiring dense labels is very expensive, they are generated automatically with Nuscenes dataset[59]. They first combine multiple lidar sweeps using ego motion to get dense pointcloud. However, some objects in the environment are also moving, thus they stitch dynamic objects and static scenes separately. Then they assign for each voxel semantic class based on majority of point lying in that voxel. To further fill the holes, they use Poisson Reconstruction. This results in a dense labels with voxel size 0.5 meters.

For simplicity, we make several modifications to the dataset. First, we limit the range of points to $[-25, 25]$ in x, y plane and $[-5, 3]$ in z plane. Furthermore, we combine some of the classes together and ignore uncommon classes. In the end, we end up with 4 classes which are "vehicle", "drivable surface", "other surface" and "vegetation".

We use two metrics to measure the performance, mIoU and IoU, suggested in [16]. For IoU, we compute intersection over union of voxels by ignoring the semantic classes and mIoU takes into account the semantic classes.

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}$$

$$\text{mIoU} = \frac{1}{C} \sum_{i=1}^{C} \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i + \text{FN}_i}$$

where TP, FP, FN indicate number of true positive, false positive and false negative and C is the number of semantic classes.

4.1.3 Decoder and Loss

We can extend our proposed architecture for semantic occupancy prediction with a simple MLP decoder as shown in figure 4.1. For center of each voxel, we sample its features from triplane using equation 3.2 and it through MLP decoder and apply softmax to get probability for each class.

The model is supervised by cross entropy loss

$$\mathcal{L} = \frac{-1}{|V|} \sum_{v \in V} \sum_{c=1}^{C} y_{v,c} \log(p_{v,c})$$

where C is the number of classes, V is the set of all voxels, $y_{v,c}$ is the binary indicator which is 1 if class label c is correct observation for voxel v, and $p_{v,c}$ is the predicted probability that voxel v of class c.

4.2 Implementation Details

4.2.1 Pretraining

We use Convnext-Tiny[44] for both range image and image encoders. Each image is reshaped to spatial dimensions $(256, 512)$, and range image is of spatial dimension $(32, 1024)$. Position encoder is MLP with one hidden layer that takes (x, y, z) coordinates of point as input and outputs a feature of dimension 192, which is same as feature channels after second layer of Convnext. Both encoders output features of size 768, and processed by 2 self-attention layers in fusion module. These features are upsampled by 2 with pixelshuffle operation and processed by convolutions to obtain triplane features of shape $(3, 32, 128, 128)$, where each triplane is of

128 spatial dimension and 32 channel dimension. The plane resolution of triplane is $v_x = 0.4, v_y = 0.4, v_z = 0.1$.

For surface reconstruction, we set $\delta = 0.1$, meaning that most objects in the scene are more than 10cm in width. We randomly sample $|S| = 2048$ points from lidar pointcloud as support points and use 1 meter radius to supervise each support point.

We pretrain each model for 40 epochs with 6 GPUs which takes around 2 days to train. We use Adam optimizer with 0.001 weight decay and cosine annealing learning rate with initial rate $2.5e^{-4}$

4.2.2 Finetuning

Parameters of encoders are same as in pretraining, and the decoder is an MLP implemented with 3 1×1 3D convolutions. All models are trained for 50 epochs with 6 GPUs using all training data which takes around 2 days to train. Optimizers and learning rates are same as in pretraining.

4.3 Masked Autoencoders

4.3.1 Pretext Task

We implement popular pretraining method based on Masked Autoencoders for comparison. Recently, [29] obtained impressive results by applying both voxel based and image based autoencoding tasks simultaneously. We compare our method to that with some modifications. First, our pointcloud representation is not in the form of voxel but a range image. So, we reconstruct range image instead of voxels. Since we use only distance as value in range image, the task is to predict the distance value for each pixel.

Second, we set masking ratio in both modalities to 0. The reason for that is our network has a lot of projections and triplane representation is compact, hence just reconstructing input without masking is a challenging task in itself. This essentially means that our model is an Autoencoder. Autoencoders [62] have a bottleneck in the middle and reconstructing the target forces the model to learn useful features. Our triplane representation is very compact representation of 3D space hence it acts as a bottleneck.

4.3.2 3D-2D projection

We encode input into triplane using our proposed method. For range image and image reconstruction, we need to project the triplane features to their respective modalities. Intuitively, we sample features of lidar points from triplane using equation 3.2, and for each lidar point we calculate its pixel location in range image using equation 3.1 and assign triplane feature to that pixel location. Projecting triplane features into image is not straightforward because we do not know the depth value of pixels. [29] projects center of non-empty voxels to image plane using intrinsic and extrinsic matrices and assign feature of voxels to the projected pixels. Similar to that, we also project lidar points to image planes and assign the feature of projected points to the pixels. In the end, we get sparse features that are projected from triplane into image planes.

4.3.3 Decoders

These reprojected features are processed by mixed vision transformer[5] to reconstruct the inputs. Features are first converted to patches using overlapping patch

merging, where patch size for image is 4×4 and range image is 1×4. The overlapping patch merging serves two purposes: since projected features to image planes are sparse, it reduces the number of non-zero patches, and it reduces computation. Patches are processed by efficient self-attention and MixFFN layers. Reconstruction is supervised by ℓ_2 loss on all pixels.

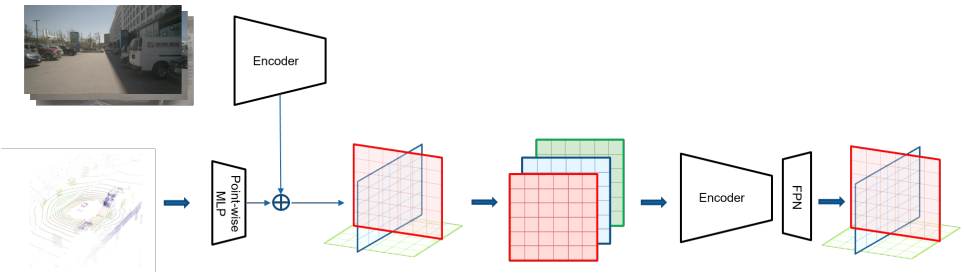


Figure 4.2: **Point Triplane.** Image features are extracted with 2D encoder and projected to pointcloud. These features are added to point features extracted with point-wise MLP. The fused features are projected to triplane and processed by encoder and FPN.

4.4 Point Triplane

Our architecture is efficient because it utilizes triplane as a common representation and everything can be processed by 2D networks. Alternative to ours, [36] also uses triplane representation and processes everything by 2D networks. They first extract point features using MLP and projects those features to triplane and encodes triplane with 2D backbones. This works only for lidar modality, and we extend this idea to handle multiple modalities for comparison. See figure [36]

4.4.1 Fusion

We extract multi-view image features using Convnext[44] encoder. We then project each lidar point into image planes and sample features using bilinear interpolation. Furthermore, we extract pointwise features using MLP. This MLP takes (x, y, z, i) as input and outputs a feature where (x, y, z) is the position of point and i is intensity. Features from images and points are added together as fused features.

4.4.2 Triplane Encoding

After the fusion step, each point has a feature which contains information from both camera and lidar modalities. We then split the 3D scene into voxels and perform max pooling to the features of points falling within the same voxel. In order to obtain triplane features, we apply spatial pooing to the voxel features along the three axes. Directly applying max pooling along an axis would result in a loss of information, thus apply group pooling as in [36]. This operation first divides the pooling axis into K groups and and perform max pooling in each group separately. Then the features of K groups are concatenated and processed with an MLP. After the pooling, each of the planes are process with same 2D network, Convnext, and feature pyramid network[63] to get final triplane features.

Chapter 5

Results

5.1 Comparison with Other Pretraining Methods

We first compare our method to other pretraining methods on the semantic occupancy prediction task. Table 5.1 shows the result of training from sracth, input reconstruction as in section 4.3, and our proposed method. We observe that pretraining is essential, as it increases mIoU by 16.9%, and using our proposed method for pretraining achieves highest score in every category.

	IoU	**mIoU**	**vehicle**	**drivable surface**	**other surface**	**vegetation**
No Pretraining	37.2	25.5	10.2	43.2	27.5	21.0
Input Reconstruction	46.9	39.2	32.2	52.1	41.4	31.2
Surface + CL	**48.8**	**42.4**	**38.4**	**54.4**	**44.6**	**32.5**

Table 5.1: **Pretraining methods on semantic occupancy prediction.**

5.2 Comparison with State-of-the-Art Methods

We compare our result to state-of-the-art methods OCCFusion [57] and SurroundOCC [16]. OCCFusion uses both lidar and camera while SurroundOCC uses only multi-view images and both of these methods are optimized for semantic occupancy task. These methods were both trained using dataset from [16], with range $[-50, 50]$ for x, y and $[-5, 3]$ for z axis. Furthermore, they were trained to classify all classes in NuScenes pointcloud segmentation. To adapt these methods to our setting, we make prediction for the full range and ignore the predictions outside $[-25, 25]$ for x, y and $[-5, 3]$ for z. We also combine the classes to get the 5 classes that we use. Table 5.2 shows the results of these methods after the modifications. We observe that our method outperforms camera only SurroundOCC in all categories. Additionally, we outperform OCCFusion on both surface classes. However, OCCFusion achieves 5% better mIoU. It should be noted here that OCCFusion uses multiple sweeps while our method only use single, and it was specifically trained to achieve best performance on semantic occupancy by combining three losses: focal loss [64], Lovasz-Softmax loss [65] and scene-class affinity loss [66] while we only use cross entropy loss.

5.3 Comparison with Point Triplane

We compare our method to another efficient multi-modal model that utilizes triplane for scene representation as we described in section 4.4. We pretrain both

	IoU	mIoU	vehicle	drivable surface	other surface	vegetation
OCCFusion	**52.3**	**47.3**	**45.4**	50.7	42.7	**50.8**
SurroundOCC	37.3	37.6	38.0	46.3	36.9	29.1
Ours	48.8	42.4	38.4	**54.4**	**44.6**	32.5

Table 5.2: **State-of-the-art methods on semantic occupancy prediction**

methods with our proposed pretraining method; surface reconstruction and multi-positive contrastive loss. We can see from table 5.3 that our method outperforms point triplane in all categories significantly.

	IoU	mIoU	vehicle	drivable surface	other surface	vegetation
Point Triplane	40.5	29.4	14.1	46.6	30.9	26.2
Ours	**48.8**	**42.4**	**38.4**	**54.4**	**44.6**	**32.5**

Table 5.3: **Point triplane:** Both methods were pretrained with our proposed pretraining method.

5.4 Learned Representation

We want to see how meaningful the learned representations are with each pretraining method. To do that, we freeze everything except the decoder during finetuning for semantic occupancy prediction. Since our decoder has access to only features of a single point, it will need to rely only on learned representation to do semantic occupancy prediction. This will tells us how semantically and geometrically meaningful these representations are. Table 5.4 shows the results of input reconstruction and our proposed pretraining method. We can see from there that our proposed method learns more semantically and geometrically meaningful features.

	IoU	mIoU	vehicle	drivable surface	other surface	vegetation
Input Reconstruction	28.5	13.8	8.70	32.2	13.9	0.60
Surface + CL	**30.2**	**18.2**	**9.60**	**35.7**	**22.9**	**4.80**

Table 5.4: **Learned triplane representations**: Everything except decoder is frozen.

5.5 Efficiency Comparison

Our goal is not to outperform state-of-the-art methods on semantic occupancy prediction, rather it is achieving efficient representation while having competitive accuracy. We have seen that our method only performs 5% worse that OCCFusion, but our method is much more memory and compute efficient. Figure 5.1 compares our method to other methods in terms of efficiency. Left figure compares GFLOps versus mIoU and we can see that ours use around 3 times less computation than other methods. On the right, we have plot of inference speed versus memory on RTX 3090 GPU. It shows that our method uses almost half memory of other methods while being 4 times faster. Indeed our method can achieve real time performance while none of the other methods can.

5.6 Visualization

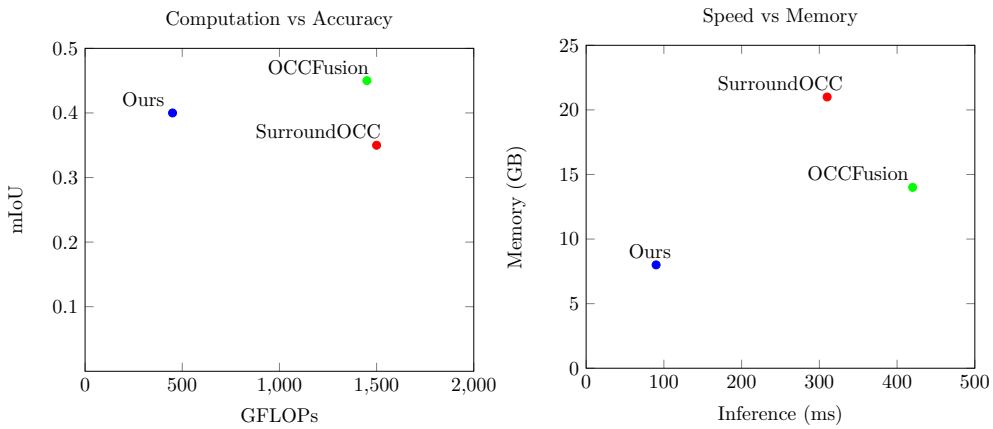


Figure 5.1: **Efficiency and Accuracy**

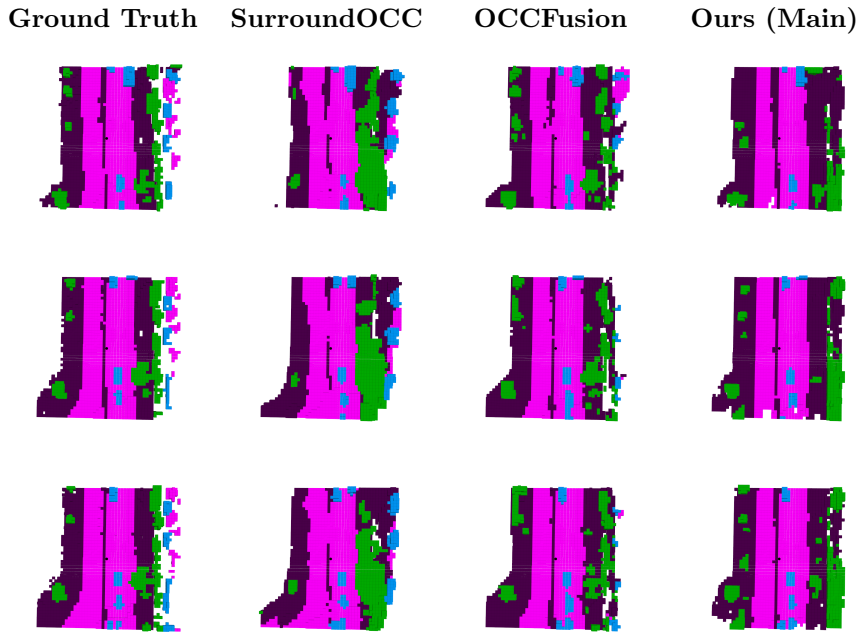


Table 5.5: **Visual comparison to SOTA.** Our method separates nearby objects from each other better, which is due to triplane having subpixel accuracy.

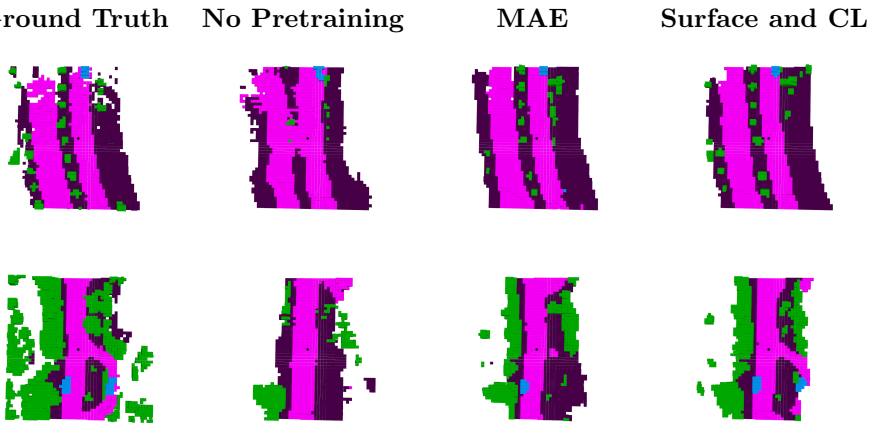


Table 5.6: **Visual comparison to other pretraining methods.** Our proposed method based on surface reconstruction and contrastive loss detects objects that other methods miss.

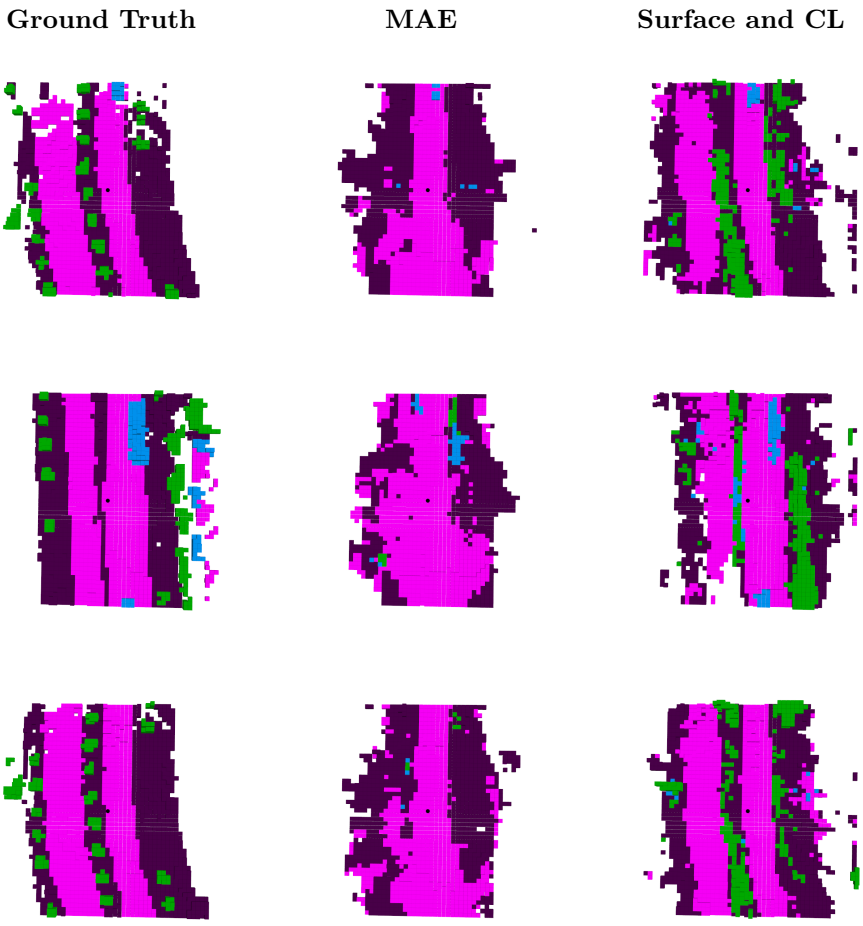


Table 5.7: **Visual comparison of learned features.** We freeze the representation and train only decoder on semantic occupancy prediction task. Our proposed method learns more semantically and geometrically meaningful representations.

Chapter 6

Discussion

6.1 Necessity of Pretraining

Our method is very efficient, we can achieve multi-modal semantic occupancy prediction in 89 milliseconds on RTX 3090 GPU. However, there is a trade-off between efficiency and accuracy. From table 5.1, we see that pretraining is essential for our method; even input reconstruction increases mIoU by 14%. The reason for that stems from the fact that we use learnable module to map modality specific features to triplane. It seems that if we pretrain the model, it learns how to do this projection and hence improves the performance significantly on downstream task.

6.2 Efficiency with Increasing Size of Scene

One desirable property of our architecture is that it scales very efficiently with increasing size of the scene. One of the reasons is range image representation of pointcloud, it can represent all points with fixed size image. Another reason is that our fusion module only needs a few extra upsampling blocks to increase size of the scene. Concretely, figure 6.1 shows how much extra memory and inference time is required to increase the size of scene from 25 meters to 50 meters. We see that our method only takes 5 milliseconds extra time and 0.25 GB extra memory, while OCCFusion requires extra 200 milliseconds and 3.5GB memory.

6.3 Arbitrary Resolution During Test Time

Another desirable property of our model is that it can make predictions for arbitrary resolution during test time. This is because triplane can represent sub-pixel information, and during both pretraining and finetuning we supervise the model with sub-pixel targets. We only use center of voxel to make predictions, hence we can have have arbitrary voxel resolution during inference. Figure 6.2 shows the result of model trained with 0.5 meters voxel size and predicted for 0.1 meters voxel size. As we see higher resolution predictions give better shape of objects and it can also separate nearby objects that are not separated in ground truth.

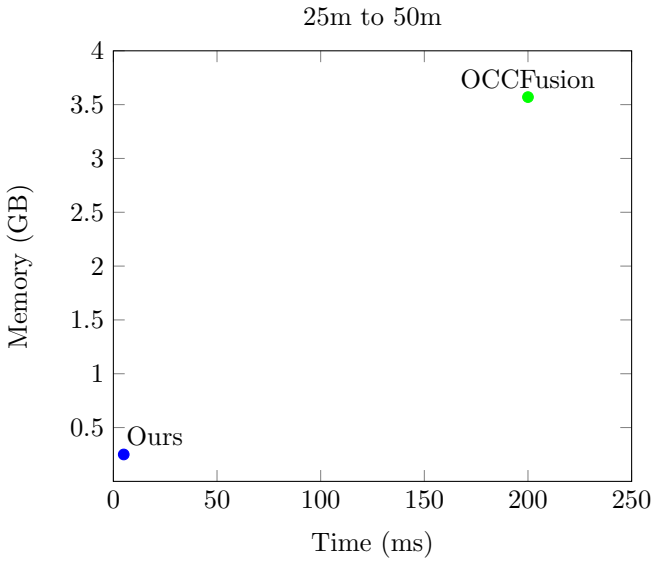
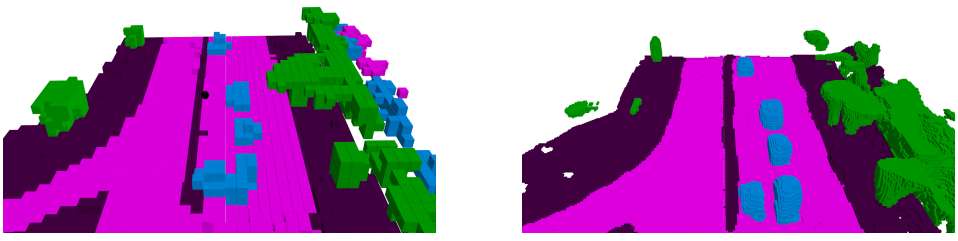


Figure 6.1: **Efficiency with respect to scene size.** This shows how much extra time and memory it takes to increase scene size from 25m to 50m.



a b

Figure 6.2: **Better resolution during test time.** We can make predictions with arbitrary resolution during test time. a) Ground truth, 0.5m resolution. b) Prediction, 0.1m resolution.

Chapter 7

Conclusion

In this thesis, we addressed key challenges in multimodal perception for autonomous robots, particularly inefficiency in 3D scene representation and the reliance on extensive labeled datasets. We proposed a novel architecture that combines the strengths of camera and LiDAR data using range image representation for point clouds and triplane representation for unified scene understanding. By designing lightweight 2D encoders and an interaction module for cross-modal feature enhancement, we achieved a computationally efficient and effective fusion of multimodal data.

Additionally, we introduced a novel self-supervised pretraining approach to reduce dependency on labeled datasets. By combining a surface reconstruction task with a multi-positive contrastive learning method, we enabled the model to learn geometrically and semantically meaningful features. Our method effectively leverages unlabeled data, using the Segment Anything Model to enrich multimodal supervision.

We evaluated our architecture and pretraining framework on the semantic occupancy prediction task, demonstrating competitive performance with state-of-the-art multimodal methods while significantly improving efficiency. Our approach achieves inference five times faster than existing methods, making it suitable for real-time applications in autonomous systems.

In general, this work highlights the potential of efficient multimodal architectures and self-supervised pretraining to advance dense 3D perception for robotics. Future research could explore extending the proposed approach to other tasks, such as motion prediction or dynamic scene understanding, and further optimizing the fusion mechanisms for scalability in large-scale environments.

Bibliography

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, *SSD: Single Shot MultiBox Detector*. Springer International Publishing, 2016, p. 21–37.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[5] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," 2021.

[6] A. Simonelli, S. R. R. Bulò, L. Porzi, M. López-Antequera, and P. Kontschieder, "Disentangling monocular 3d object detection," 2019.

[7] F. Manhardt, W. Kehl, and A. Gaidon, "Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape," 2019.

[8] M. Ding, Y. Huo, H. Yi, Z. Wang, J. Shi, Z. Lu, and P. Luo, "Learning depth-guided convolutions for monocular 3d object detection," 2019.

[9] G. Brazil and X. Liu, "M3d-rpn: Monocular 3d region proposal network for object detection," 2019.

[10] S. Farooq Bhat, I. Alhashim, and P. Wonka, "Adabins: Depth estimation using adaptive bins," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2021.

[11] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," 2016.

[12] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d," 2020.

[13] S. Sirko-Galouchenko, A. Boulch, S. Gidaris, A. Bursuc, A. Vobecky, P. Pérez, and R. Marlet, "Occfeat: Self-supervised occupancy feature prediction for pretraining bev segmentation networks," 2024.

[14] Y. Li, Q. Han, M. Yu, Y. Jiang, C. Yeo, Y. Li, Z. Huang, N. Liu, H. Chen, and X. Wu, "Towards efficient 3d object detection in bird's-eye-view space for autonomous driving: A convolutional-only approach," 2024.

[15] D. Rukhovich, A. Vorontsova, and A. Konushin, "Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection," 2021.

[16] Y. Wei, L. Zhao, W. Zheng, Z. Zhu, J. Zhou, and J. Lu, "Surroundocc: Multi-camera 3d occupancy prediction for autonomous driving," *arXiv preprint arXiv:2303.09551*, 2023.

[17] M. Pan, J. Liu, R. Zhang, P. Huang, X. Li, B. Wang, H. Xie, L. Liu, and S. Zhang, "Renderocc: Vision-centric 3d occupancy prediction with 2d rendering supervision," 2024.

[18] Y. Huang, W. Zheng, Y. Zhang, J. Zhou, and J. Lu, "Tri-perspective view for vision-based 3d semantic occupancy prediction," 2023.

[19] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia, "Voxelnext: Fully sparse voxelnet for 3d object detection and tracking," 2023.

[20] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," 2017.

[21] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang, "Structure aware single-stage 3d object detection from point cloud," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[22] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu, "Voxel transformer for 3d object detection," 2021.

[23] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet++: Fast and Accurate LiDAR Semantic Segmentation," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[24] E. Li, S. Casas, and R. Urtasun, "Memoryseg: Online lidar semantic segmentation with a latent memory," in *ICCV*, 2023.

[25] A. Athar, E. Li, S. Casas, and R. Urtasun, "4d-former: Multimodal 4d panoptic segmentation," in *Proceedings of the 2023 Conference on Robot Learning*, 2023.

[26] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," 2017.

[27] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," 2017.

[28] G. Hess, J. Jaxing, E. Svensson, D. Hagerman, C. Petersson, and L. Svensson, "Masked autoencoder for self-supervised pre-training on lidar point clouds," in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*. IEEE, Jan. 2023.

[29] J. Zou, T. Huang, G. Yang, Z. Guo, T. Luo, C.-M. Feng, and W. Zuo, "Unim^{2}ae: Multi-modal masked autoencoders with unified 3d representation for 3d perception in autonomous driving," 2024.

[30] H. Yang, S. Zhang, D. Huang, X. Wu, H. Zhu, T. He, S. Tang, H. Zhao, Q. Qiu, B. Lin, X. He, and W. Ouyang, "Unipad: A universal pre-training paradigm for autonomous driving," 2024.

[31] L. Fan, Z. Pang, T. Zhang, Y.-X. Wang, H. Zhao, F. Wang, N. Wang, and Z. Zhang, "Embracing single stride 3d object detector with sparse transformer," 2021.

[32] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, and S. Pu, "Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation," 2021.

[33] L. Fan, X. Xiong, F. Wang, N. Wang, and Z. Zhang, "Rangedet:in defense of range view for lidar-based 3d object detection," 2021.

[34] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus, and S. Han, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," 2024.

[35] T. Liang, H. Xie, K. Yu, Z. Xia, Z. Lin, Y. Wang, T. Tang, B. Wang, and Z. Tang, "Bevfusion: A simple and robust lidar-camera fusion framework," 2022.

[36] S. Zuo, W. Zheng, Y. Huang, J. Zhou, and J. Lu, "Pointocc: Cylindrical tri-perspective view for point-based 3d semantic occupancy prediction," 2023.

[37] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020.

[38] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," 2020.

[39] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," 2021.

[40] S. Xie, J. Gu, D. Guo, C. R. Qi, L. J. Guibas, and O. Litany, "Pointcontrast: Unsupervised pre-training for 3d point cloud understanding," 2020.

[41] A. Boulch, C. Sautier, B. Michele, G. Puy, and R. Marlet, "Also: Automotive lidar self-supervision by occupancy estimation," 2023.

[42] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv:2304.02643*, 2023.

[43] Y. Tian, L. Fan, P. Isola, H. Chang, and D. Krishnan, "Stablerep: Synthetic images from text-to-image models make strong visual representation learners," 2023.

[44] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," 2022.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[47] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[48] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021.

[49] 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," 2021.

[50] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. D. Mello, O. Gallo, L. Guibas, J. Tremblay, S. Khamis, T. Karras, and G. Wetzstein, "Efficient geometry-aware 3d generative adversarial networks," 2022.

[51] A. Trevithick, M. Chan, M. Stengel, E. R. Chan, C. Liu, Z. Yu, S. Khamis, M. Chandraker, R. Ramamoorthi, and K. Nagano, "Real-time radiance fields for single-image portrait view synthesis," 2023.

[52] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Q. Yu, and J. Dai, "Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," 2022.

[53] A. W. Harley, Z. Fang, J. Li, R. Ambrus, and K. Fragkiadaki, "Simple-bev: What really matters for multi-sensor bev perception?" 2022.

[54] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent: A new approach to self-supervised learning," 2020.

[55] X. Tian, H. Ran, Y. Wang, and H. Zhao, "Geomae: Masked geometric target prediction for self-supervised point cloud pre-training," 2023.

[56] A. Chen, K. Zhang, R. Zhang, Z. Wang, Y. Lu, Y. Guo, and S. Zhang, "Pimae: Point cloud and image interactive masked autoencoders for 3d object detection," 2023.

[57] Z. Ming, J. S. Berrio, M. Shan, and S. Worrall, "Occfusion: Multi-sensor fusion framework for 3d semantic occupancy prediction," *IEEE Transactions on Intelligent Vehicles*, 2024.

[58] X. Wang, Z. Zhu, W. Xu, Y. Zhang, Y. Wei, X. Chi, Y. Ye, D. Du, J. Lu, and X. Wang, "Openoccupancy: A large scale benchmark for surrounding semantic occupancy perception," 2023.

[59] 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," 2020.

[60] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," 2016.

[61] X. Chen, S. Xie, and K. He, "An empirical study of training self-supervised vision transformers," 2021.

[62] G. G. Pihlgren, F. Sandin, and M. Liwicki, "Pretraining image encoders without reconstruction via feature prediction loss," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, Jan. 2021.

[63] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2017.

[64] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018.

[65] M. Berman, A. R. Triki, and M. B. Blaschko, "The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks," 2018.

[66] A.-Q. Cao and R. de Charette, "Monoscene: Monocular 3d semantic scene completion," 2022.