End-to-end Stereo Layout Estimation

Divam Gupta

CMU-RI-TR-21-29

August 4, 2021

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To all my advisors and mentors.

Abstract

Accurate layout estimation is crucial for planning and navigation in robotics applications, such as self-driving. In this thesis, we introduce the Stereo Bird's Eye View Network (*SBEVNet*), a novel supervised end-to-end framework for estimation of bird's eye view layout from a pair of stereo images. Although our network reuses some of the building blocks from the state-of-the-art deep learning networks for disparity estimation, we show that explicit depth estimation is neither sufficient nor necessary. Instead, the learning of a good internal bird's eye view feature representation is effective for layout estimation. Specifically, we first generate a disparity feature volume using the features of the stereo images and then project it to the bird's eye view coordinates. This gives us coarse-grained information about the scene structure. We also apply inverse perspective mapping (IPM) to map the input images and their features to the bird's eye view. This gives us fine-grained texture information. Concatenating IPM features with the projected feature volume creates a rich bird's eye view representation which is useful for spatial reasoning. We use this representation to estimate the BEV semantic map. Additionally, we show that using the IPM features as a supervisory signal for stereo features can give an improvement in performance. We demonstrate our approach on three datasets: the KITTI [5] dataset, a synthetically generated dataset from the CARLA [4] simulator and a dataset collected in a forest environment. For all of these datasets, we establish state-of-the-art performance compared to baseline techniques.

Acknowledgments

I would like to thank my advisor Prof. Jeff Schneider for giving me the opportunity to work on this project. I am grateful for all the guidance, support, funding and discussions. I am also thankful to my other committee members Prof. Simon Lucey and Dinesh Reddy for taking the time to give me their inputs.

I am thankful to Trenton Tabor and Wei Pu for all the technical guidance and thoughtful discussions, Nicholas Mireles for supporting the data processing, Luis Navarro-Serment for discussions regarding autonomous driving and all other people of National Robotics Engineering Center for the support. I am also thankful to all the lab members of the Auton Lab for the discussions and inputs.

I am also grateful to my past research collaborators at Indraprastha Institute of Information Technology Delhi, Indian Institute of Technology Bombay and Microsoft Research India for the mentorship and teaching me the essential skills for academic research.

I would like to thank all my friends and classmates of Master of Science in Robotics for all the support and discussions. Lastly, I would thank my parents and family for their support and continual encouragement.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

Layout estimation is an extremely important task for navigation and planning in numerous robotics applications such as autonomous driving cars. The bird's eye view (BEV) layout is a semantic occupancy map containing per pixel class information, e.g. road, sidewalk, cars, vegetation, etc. The BEV semantic map is important for planning the path of the robot in order to prevent it from hitting objects and going to impassable locations.

In order to generate a BEV layout, we need 3D information about the scene. Sensors such as LiDAR (Light Detection And Ranging) can provide accurate point clouds. The biggest limitations of LiDAR are high cost, sparse resolution, and low scan-rates. Also, as an active sensor LiDAR is more power hungry, more susceptible to interference from other radiation sources, and can affect the scene. Cameras on the other hand, are much cheaper, passive, and capture much more information at a higher frame-rate. However, it is both hard and computationally expensive to get accurate depth and point clouds from cameras.

The classic approach for stereo layout estimation contains two steps. The first step is to generate a BEV feature map by an orthographic projection of the point cloud generated using stereo images. The second step is bird's eye view semantic segmentation using the projected point cloud from the first step. This approach is limited by the estimated point cloud accuracy because the error in it will propagate to the layout estimation step. In this thesis, we show that explicit depth estimation is actually neither sufficient nor necessary for good layout estimation. Estimating accurate depth is not sufficient because many areas in the 3D space can be occluded partially, e.g. behind a tree trunk. However, these areas can be estimated by combining spatial reasoning and geometric knowledge in bird's eye view representation. Explicitly estimating accurate depth is also not necessary because layout estimation can be done without estimating the point cloud. Point cloud coordinate accuracy is limited by the 3D to 2D BEV projection and rasterization. For these reasons, having an effective bird's eye view representation is very important.

SBEVNet is built upon recent deep stereo matching paradigm. These deep learning based methods have shown tremendous success in stereo disparity/depth estimation. Most of these models [2, 6, 9, 10, 12, 20, 22, 27] generate a 3-dimensional disparity feature volume by concatenating the left and right images shifted at different disparities, which is used to make a cost volume containing stereo matching costs for each disparity value. Given a location in the image and the disparity, we can get the position of the corresponding 3D point in the world space. Hence, every point in the feature volume and cost volume corresponds to a 3D location in the world space. The innovation in our approach comes from the observation: it is possible to directly use the feature volume for layout estimation, rather than a two step process, which uses the point cloud generated by the network. We propose *SBEVNet*, an end-to-end neural architecture that takes a pair of stereo images and outputs the bird's eye view scene layout. We first project the disparity feature volume to the BEV view, creating a 2D representation from the 3D volume. We then warp it by mapping different disparities and the image coordinates to the bird's eye view space. In order to overcome the loss of fine grained information imposed by our choice of the stereo BEV feature map, we concatenate a projection of the original images and deep features to this feature map. We generate these projected features by applying inverse perspective mapping (IPM) [14] to the input image and its features, choosing the ground as the target plane We feed this representation to a U-Net in order to estimate the BEV semantic map of the scene.

In order to perform inverse perspective mapping, we require information about the ground in the 3D world space. Hence we also consider the scenario where we perform IPM during the training time and not the inference time. Here, during the training time, we use cross modal distillation to transfer knowledge from IPM features to the stereo features.

SBEVNet is the first approach to use an end-to-end neural architecture for stereo layout estimation. We show that *SBEVNet* achieves better performance than existing approaches. *SBEVNet* outperforms all the baseline algorithms on KITTI [5] dataset, a synthetically generated dataset extracted from the CARLA simulator and a dataset collected in a forest environment. [4]. In summary, our contributions are the following:

1. We propose *SBEVNet*, an end-to-end neural architecture for layout estimation from a stereo pair of images.
2. We learn a novel representation for BEV layout estimation by fusing projected stereo feature volume and fine grained inverse perspective mapping features.
3. We evaluate *SBEVNet* and demonstrate state-of-the-art performance over other methods by a large margin on three datasets – KITTI dataset, our synthetically generated dataset using the CARLA simulator and a dataset collected in a forest environment.

Chapter 2

Background

SBEVNet is a deep learning based approach which heavily uses concepts of multi-view geometry and stereo vision. In this chapter, we provide a brief introduction of all the methods and algorithms used in this thesis.

2.1 Stereo Vision Preliminaries

In most of the stereo vision setups, it is assumed that the relative rotation and the translation between the two cameras is known. The two cameras are called the reference camera and target camera and the images produced by them are called reference image I_R and target image I_T respectively. Given a point in the 3D world space, the first step is to find the projection of it by the reference and target camera. Given the intrinsic and extrinsic parameters of both the cameras, the 3D position can be estimation via triangulation.

2.1.1 Rectified Stereo Pair

For an arbitrary camera setup, a point in the reference image can correspond to any point along the epipolar line in the target image. If the two cameras have no relative rotation and the translation is just across the x-axis, then the epipolar lines will be parallel to the x-axis in the image. A rectified stereo pair makes searching correspondences easy and also makes triangulation easy.

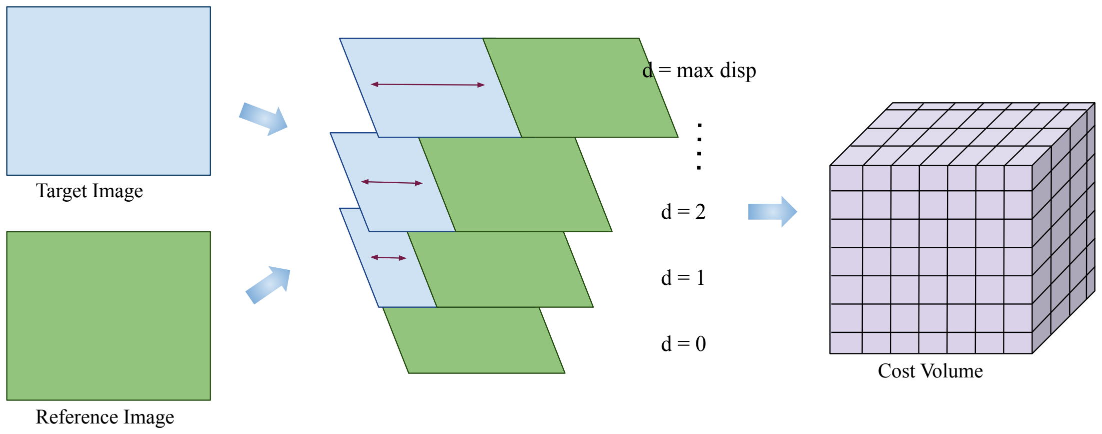


Figure 2.1: Illustration of plane sweep stereo.

Disparity

For a rectified stereo image pair, the disparity at a point is the shift in pixels of the position of the point in the reference and the target image.

3D Reconstruction from Disparity

Given the disparity d of a point u, v in the reference image, we can easily get the homogeneous 3D position in the world space using the following equation:

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -\frac{1}{T_x} & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} \tag{2.1}$$

Here c_x and c_y are the principal points of the camera, f is the focal length and T_x is the baseline length.

2.1.2 Plane Sweep Stereo

The simple way to find the disparity of a pixel is to search for the most similar pixel in the horizontal epipolar line in the other image. This can be done by storing the matching costs at all disparity levels in a tensor called cost volume. For image size $H \times W$ and maximum disparity D, the cost volume V_{cost} will be of size $H \times W \times D$. Given a disparity d and the image point u, v , the cost volume can be constructed as:

$$V[u, c, d] = \text{similarity}(I_R[u, v], I_T[u + d, v]) \tag{2.2}$$

The similarity function could be sum of squared differences or sum of absolute differences. This cost volume is then used to find the disparity score of each pixel. To account for noisy pixels and mismatched disparities, a smoothness function is applied on the cost volume which penalizes when the disparity of a pixel is very different from its neighbours.

2.1.3 Deep Learning Methods for Stereo Matching

The naive way for stereo matching using deep learning is to concatenate the reference and the target image to a single 6 channel image and apply a 2D fully convolutional network to regress on the disparity map. This is not a very good approach because the convolutional filters have to learn how to look up for other pixels at different disparities. A more efficient way for stereo matching using deep learning is to use concepts from plane sweep stereo. The following changes can me made to the classical plane sweep method to make it end-to-end trainable:

1. Rather than using the cost volume a feature volume can be created by concatenating the features of both images at all disparity levels. This gives the network flexibility to learn the similarity function on its own.
2. Rather than creating the volume using the raw pixels, the disparity volume can be created using image features extracted using a deep network.
3. The smoothness function can also be replaced by a series of 3D convolutions which can also learn some global information in the scenes.
4. To train the network we can use either a supervised loss or an unsupervised loss.

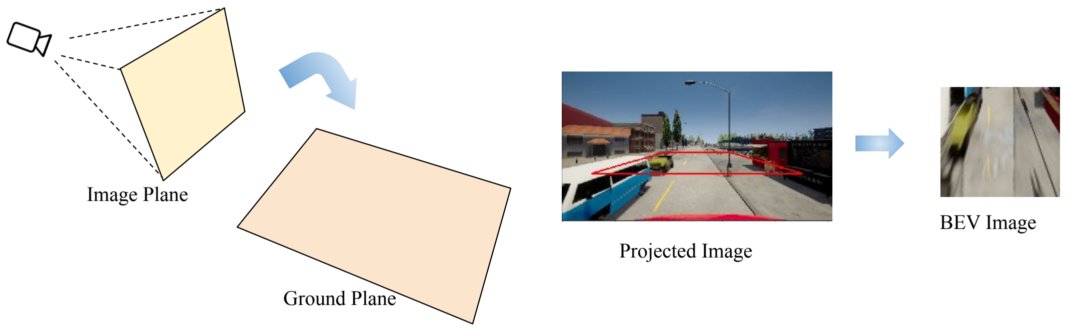


Figure 2.2: Illustration of inverse perspective mapping.

2.2 Inverse Perspective Mapping

Inverse perspective mapping (IPM) is a technique for image coordinate transformation, which can be used to take a perspective projection image and transform it in the bird's eye view, removing the perspective effect. This transformation is a homography transformation transforming the image from the road plane to the protective image plane. To perform inverse perspective mapping, we need to have the position and orientation of the ground plane with respect to the camera. The homography matrix can be computed with 4 correspondences points, which could be the corners of the BEV range on the plane.

Chapter 3

Related Work

To the best of our knowledge, there is no published work for estimating layout given a pair of stereo images. However, there are several works tackling layout estimation using a single image or doing object detection using stereo images. In this section, we review the most closely related approaches.

3.1 Monocular Layout Estimation

MonoLayout [15] uses an encoder-decoder model to estimate the bird's eye view layout using a monocular input image. They also leverage adversarial training to produce sharper estimates. MonoOccupancy [13] uses a variational encoder-decoder network to estimate the layout. Both MonoLayout and MonoOccupancy do not use any camera geometry priors to perform the task. Schulter et al. [17] uses depth estimation to project the image semantics to bird's eye view. They also use Open Street Maps data to refine the BEV images via adversarial learning. Wang et al. [23] uses [17] to estimate the parameters of the road such as lanes, sidewalks, etc. Monocular methods learn strong prior, which does not generalize well when there is a significant domain shift. Stereo methods learn weak-prior plus geometric relationship, which can generalize better.

3.2 Deep Stereo Matching

Several methods like [2, 6, 9, 10, 12, 20, 22, 27] extract the features of the stereo images and generate a 3D disparity feature volume for disparity/depth estimation. They use a 3D CNN on the feature volume to get cost volume to perform stereo matching. PSMNet [2] uses a spatial pyramid pooling module and a stacked hourglass network to further improve the performance. High-res-stereo [25] uses a hierarchical model, creating cost volumes at multiple resolutions, performing the matching incrementally from over a coarse to fine hierarchy.

3.3 Bird's Eye View Object Detection

Several approaches like Shi et al. [18], Yang et al. [24] use LiDAR to perform 3D object detection. Pseudo-lidar [21] and pseudo-LiDAR++ [26] use stereo input to first generate a 3D point cloud and then use a 3D object detection network [11, 18, 24] on top. BirdGAN [19] maps the input image to bird's eye view using adversarial learning. The closest work to our approach is DSGN [3] which constructs a depth feature volume and map it to the 3D space which is then projected to bird's eye view to perform object detection. The task of object detection is of sparse prediction, whereas layout estimation is of dense fine granularity prediction. Hence we introduced IPM to fuse low level detail with the stereo information to improve the performance of layout estimation.

Chapter 4

Approach

This section describes the detailed architecture of our proposed framework. *SBEVNet* is built upon recent deep stereo matching paradigms and follows the rules of multi-view camera geometry. An overview of the *SBEVNet* is summarized in Figure 4.1.

4.1 Problem Formulation

In this thesis, we address the problem of layout estimation from a pair of stereo images. Formally, given a reference camera image I_R and a target camera image I_T both of size $H \times W \times 3$, the camera intrinsics K, and the baseline length T_b, we aim to estimate the bird's eye view layout of the scene. In particular, we estimate the BEV semantic map of size $N_x \times N_y \times N_C$ within the rectangular range of interest area $(x_{min}, x_{max}, y_{min}, y_{max})$ in front of the camera. Here H is image height, W is image width, and 3 indicates RGB channels. N_x and N_y are the number of horizontal cells and vertical cells respectively in bird's eye view. N_C is the number of semantic classes. This BEV semantic map contains the probability distribution among all semantic classes at each cell of the layout. We assume that the input images are rectified.

4.2 Feature Extraction

The first step for *SBEVNet* is to extract features F_R and F_T of size $H' \times W' \times C$ for the reference image and the target image respectively. This is done by passing I_R and

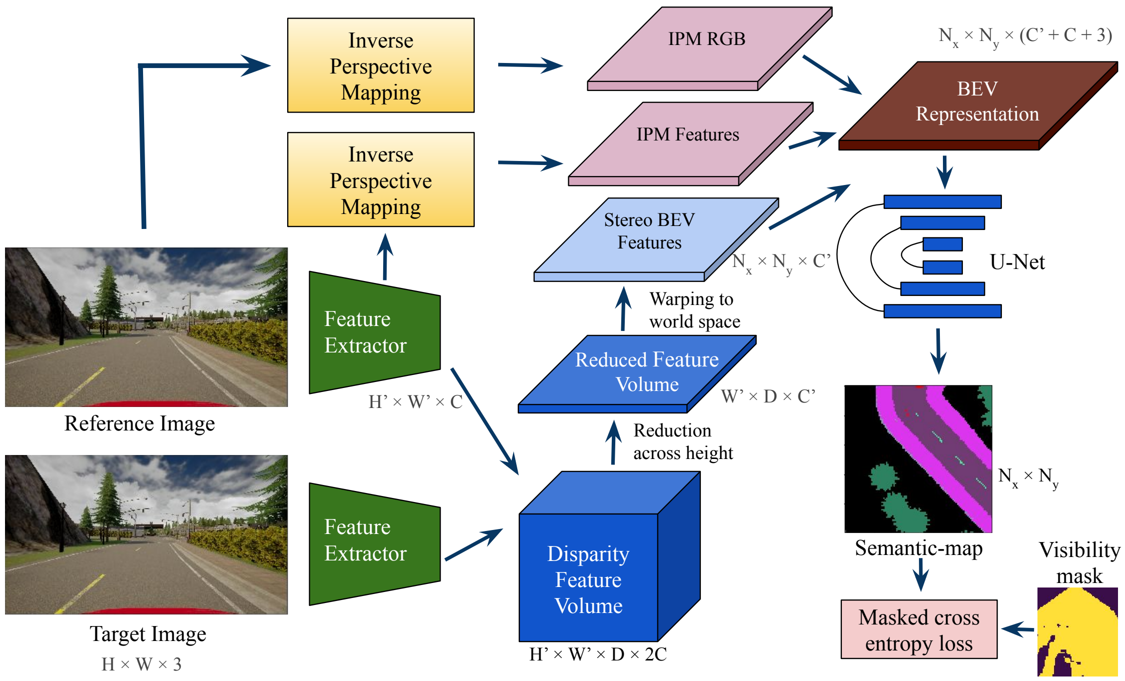


Figure 4.1: *SBEVNet* overview. We first extract the image features given the target and reference image. Using the pair of features, we create a disparity feature volume. We then reduce the disparity feature volume along with the height and warp it in the bird's eye view layout space. On a parallel branch of the network, we apply inverse perspective mapping (IPM) on the reference image and its features. We concatenate the IPM RGB, IPM feature, and the stereo BEV features. The BEV representation is then used to estimate the semantic map through a U-Net. Visibility mask is used to apply the supervised loss only at the locations in the BEV which are in the view of the front camera.

I_T through a convolutional encoder with shared weights. This produces multi-channel down-sized feature representations which are next used for building disparity feature volumes.

4.3 Disparity Feature Volume Generation

Similar to [2, 6, 9, 10, 12, 20, 22, 27] we form a disparity feature volume V by concatenating the features F_R and F_T^d, where F_T^d is F_T shifted horizontally by a disparity of d pixel, resulting in a 3D volume of size $H' \times W' \times D \times 2C$. We then pass the feature volume through a series of 3D convolution layers with skip connections to learn higher level features. This feature volume at each $d \in \{0, 1, \cdots, D-1\}$ contains a representation of the 3D world at the depth corresponding to the disparity d. Rather than using this feature volume to do disparity estimation, we project and warp it to form a bird's eye view representation in the next step.

4.4 Bird's Eye View Representation

The bird's eye view representation is composed of two parts – 1) The stereo BEV representation which is derived from the disparity feature volume, 2) The IPM BEV representation which is the result of applying inverse perspective mapping on the reference image and the features of the reference image. These two parts are concatenated to form the final bird's eye view representation.

4.4.1 Stereo BEV Representation

The disparity feature volume generated is widely used to estimate depth/disparity in the stereo image pairs. But this feature volume contains a lot of information about the 3D scene which can be used for other tasks as well. Each point in the disparity feature volume corresponds to a point in the 3D world space. We first need to map the 3D feature volume to a 2D feature map containing information of the bird's eye view. If we do max/average pooling along height dimension, a certain degree of the height information is lost quickly before being extracted for our task, which is not desirable. Considering height information a good prior for layout estimation but we don't need to recover it explicitly, we concatenate the feature volume along the height, creating a 2D image of size $W' \times D \times 2CH'$. We then use 2D convolutions to generate the reduced feature volume of size $W'' \times D'' \times C'$. This reduced feature volume does not spatially match with the bird's eye view layout. Hence, we warp the reduced feature volume, transforming it to a feature map of size $N_x \times N_y \times C'$ in the bird's eye view space. Given the disparity d, position in the image along width u, camera parameters f, c_x, c_y, and stereo baseline length T_x, we can find the coordinates in the bird's eye space x' and y' as follows:

$$x' = \frac{(u - c_x) \cdot T_x}{d} \tag{4.1}$$

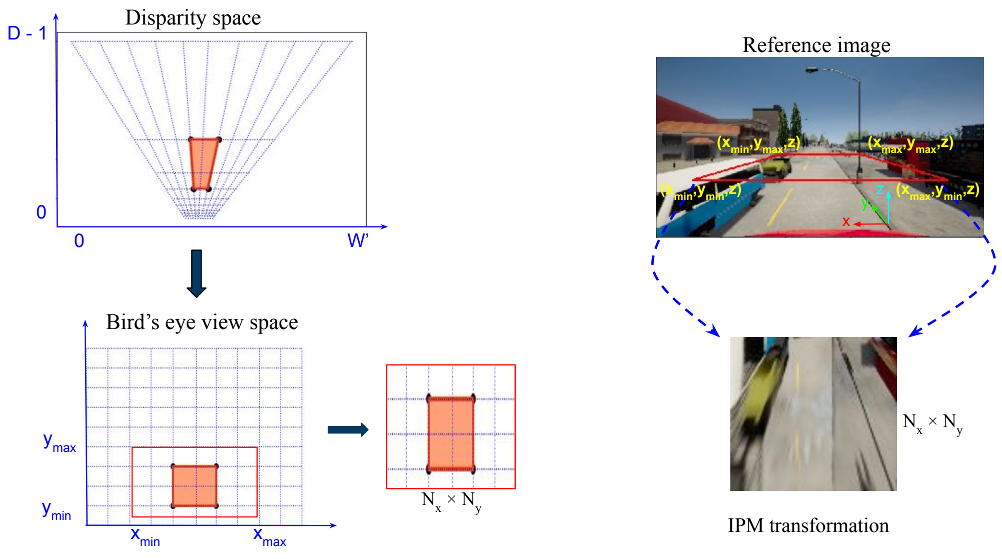


Figure 4.2: Illustration of mapping the disparity space to bird's eye view space and inverse perspective mapping. (a) The operation maps different disparities and x to the BEV space in order to match the ground truth. We also show an example layout warped to the disparity space. (b) The inverse perspective mapping operation maps pixels of the reference image to the BEV space in order to match the ground truth. The same mapping can be applied to the image features as well.

$$y' = \frac{f \cdot T_x}{d} \tag{4.2}$$

The 2D origin of the bird's eye view is co-located with the reference camera. An example visualization of the layout in the disparity volume space is shown in Figure 4.2. After mapping the coordinates to the BEV space, we map them to a grid of size $N_x \times N_y$ giving us the stereo BEV representation R_{stereo}.

4.4.2 IPM BEV Representation

The stereo BEV representation contains structural information for the bird's eye view space. Due to the refinement and reduction of the feature volume, the fine grained details are excluded by design. To circumvent that, we need to fuse the low level features to the stereo BEV features, while maintaining geometric consistency.

In order to fuse the image features to the stereo BEV features at the correct locations, we need to warp the image features to the BEV space. We apply inverse perspective mapping on the reference image and the features of the reference image to do that.

A point in the image I_R can correspond to multiple points in the 3D world space due to perspective projection, but there is a single point which also intersects with the ground plane. Let $z = ax + by + c$ be the equation of the ground plane in the world space. Given the input image coordinates (u, v) and camera parameters f,c_x, c_y , we can find the coordinates in the bird's eye space x' and y' as follows:

$$x' = \frac{cu - cc_x}{ac_x - au - bf - c_y + v} \tag{4.3}$$

$$y' = \frac{cf}{ac_x - au - bf - c_y + v} \tag{4.4}$$

This can be easily derived by combining the camera projection equation with the equation of the ground plane. For many applications, the ground is either planar or can be approximated by a plane. This is also equivalent to computing a homography H between the ground plane and the image plane of the layout and then applying the transformation. We can have the parameters of the plane a, b, and c pre-determined if the placement of the camera with respect to the ground is known, which is the case for many robotics applications. We can also determine a, b and c by using stereo depth and a semantic segmentation network for the road/ground class.

Examples of IPM on the input images is shown in Figure 4.2. We apply the inverse perspective transform on both the input image and the features of the input image to transform them to the bird's eye view space:

$$R_{\text{IPM\_feat}} = \text{IPM}(F_R) \tag{4.5}$$

$$R_{\text{IPM\_img}} = \text{IPM}(I_R) \tag{4.6}$$

They are then concatenated with the stereo BEV representation to form the combined

BEV representation:

$$R_{\text{BEV}} = [R_{\text{IPM\_feat}}; R_{\text{IPM\_img}}; R_{\text{stereo}}] \tag{4.7}$$

4.4.3 Layout Generation

We can generate the semantic map by inputting the BEV features to a semantic segmentation network. We pass the concatenated stereo BEV feature map and IPM BEV feature map to a U-Net [16] network to generate the semantic map C.

$$C = \text{U-Net}(R_{\text{BEV}}) \tag{4.8}$$

Some areas in the layout may not be in the view of the front camera, e.g. things behind a wall. That is why it is not a good idea to penalize the model for the wrong prediction for those areas. Hence, we use a visibility mask to mask the pixel-wise loss, applying it only on the pixels which are in the field of view. This mask is generated during the ground truth generation process by using ray-tracing on the point cloud to determine which are in the field of view. For a visibility mask V, V_i is 1 if the pixel i is in the view of the input image, and 0 otherwise. For the loss, we use a pixel-wise categorical cross entropy loss as follows:

$$L_r = \sum_{i \in P} V_i \cdot \text{CCE}(C_i, C_i^h) \tag{4.9}$$

where C_i^h is ground truth.

4.5 IPM for Cross Modal Distillation

There can be use-cases where we cannot do inverse perspective mapping during inference time, due to the unavailability of the ground information. Hence, we consider the case where IPM is only available during the training time. We can think of the IPM features and the stereo features as different modalities and apply cross modal distillation [7] across them, and transfer knowledge from IPM features to the stereo features. Hence, we use the IPM BEV representation as a supervisory

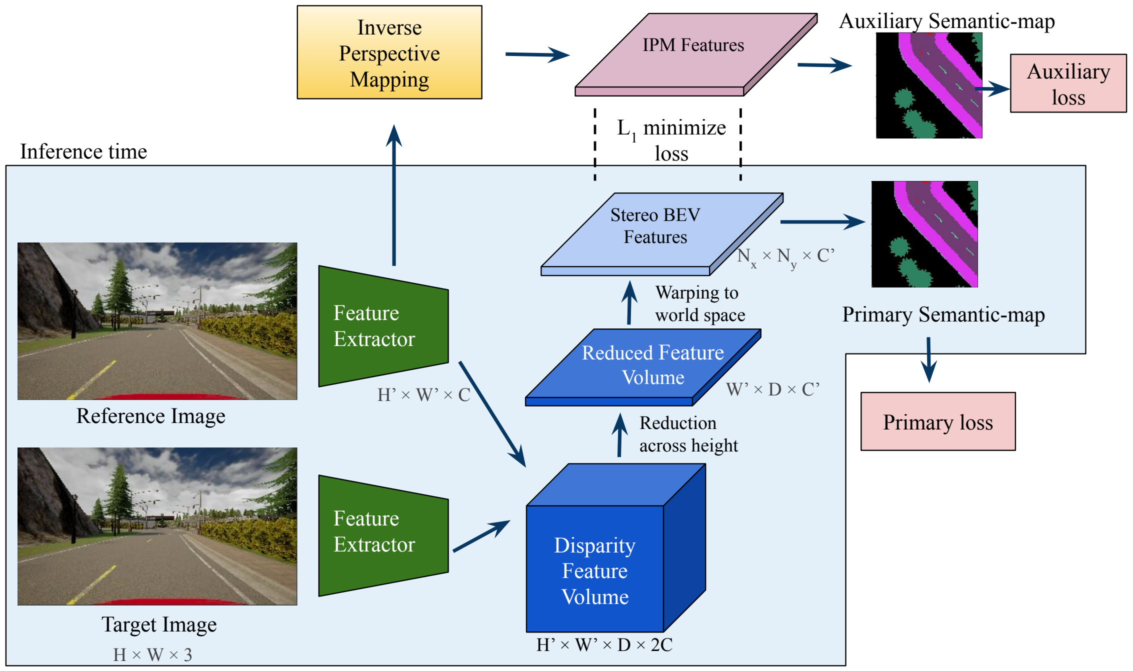


Figure 4.3: *SBEVNet*-CMD overview. We first extract the image features given the target and reference image. Using the pair of features, we create a stereo BEV representation. During training time, we apply inverse perspective mapping (IPM) on the image features which is used to predict the BEV layout separately. We minimize the L1 distance between first K channels of both the BEV feature maps. During inference time, we only use the stereo BEV representation to estimate the semantic map.

signal for the stereo BEV features. This forces the stereo branch of the model to implicitly learn the fine grained information learned by the IPM features. Rather than concatenating the IPM BEV features with the stereo features, we minimize the distance between them. We call this variant of *SBEVNet* as ***SBEVNet-CMD*** (*SBEVNet* cross modal distillation). During the training time, the IPM BEV features and the stereo features are used to generate the BEV semantic maps.

$$C^{\text{IPM}} = \text{U-Net}(R_{\text{IPM\_feat}}) \tag{4.10}$$

$$C^{\text{stereo}} = \text{U-Net}(R_{\text{stereo}}) \tag{4.11}$$

This ensures both IPM BEV features and stereo BEV features learn meaningful information. We jointly minimise the L_1 distance between first K channels of the features.

$$L_{\text{KT}} = \|R_{\text{IPM\_feat}}[:K] - R_{\text{stereo}}[:K]\|_{L_1} \tag{4.12}$$

By this, we ensure that the stereo model can learn information that is not in the IPM features. In our experiments, we found this to yield better results compared to the approach of minimizing the L_1 distance between all the channels of the features. During test time, we only use the stereo features to get the BEV layout. Our experiments show that the stereo model with cross modal distillation performs better than the stereo model without cross modal distillation.

The total loss for *SBEVNet-CMD* is the sum of supervision loss from the two feature maps and the L_1 distance minimization.

$$L_c = \sum_{i \in P} V_i \cdot \text{CCE}(C_i^{\text{IPM}}, C_i^h) + \sum_{i \in P} V_i \cdot \text{CCE}(C_i^{\text{stereo}}, C_i^h) + L_{\text{KT}} \tag{4.13}$$

4.6 Disentangling Segmentation and Disparity

In this section, we propose *SBEVNet-disentangled*, a variant of *SBEVNet* where the disparity and segmentation are explicitly disentangled. This is done by having separate disparity and segmentation branches. Both the disparity branch and the segmentation branch have separate feature extraction networks. The disparity branch learns disparity probabilities which are used to project the image features of the segmentation branch in bird's eye view. The image features in the segmentation branch are also used to learn the front view semantic segmentation, whereas a self-supervised disparity estimation loss is applied in the disparity branch.

4.6.1 Disparity Branch

In the disparity branch, we use a convolutional encoder to extract image features F_R and F_T for the reference image and the target image respectively. Like previous model, these features are used to compute a disparity feature volume V_{fea} of size

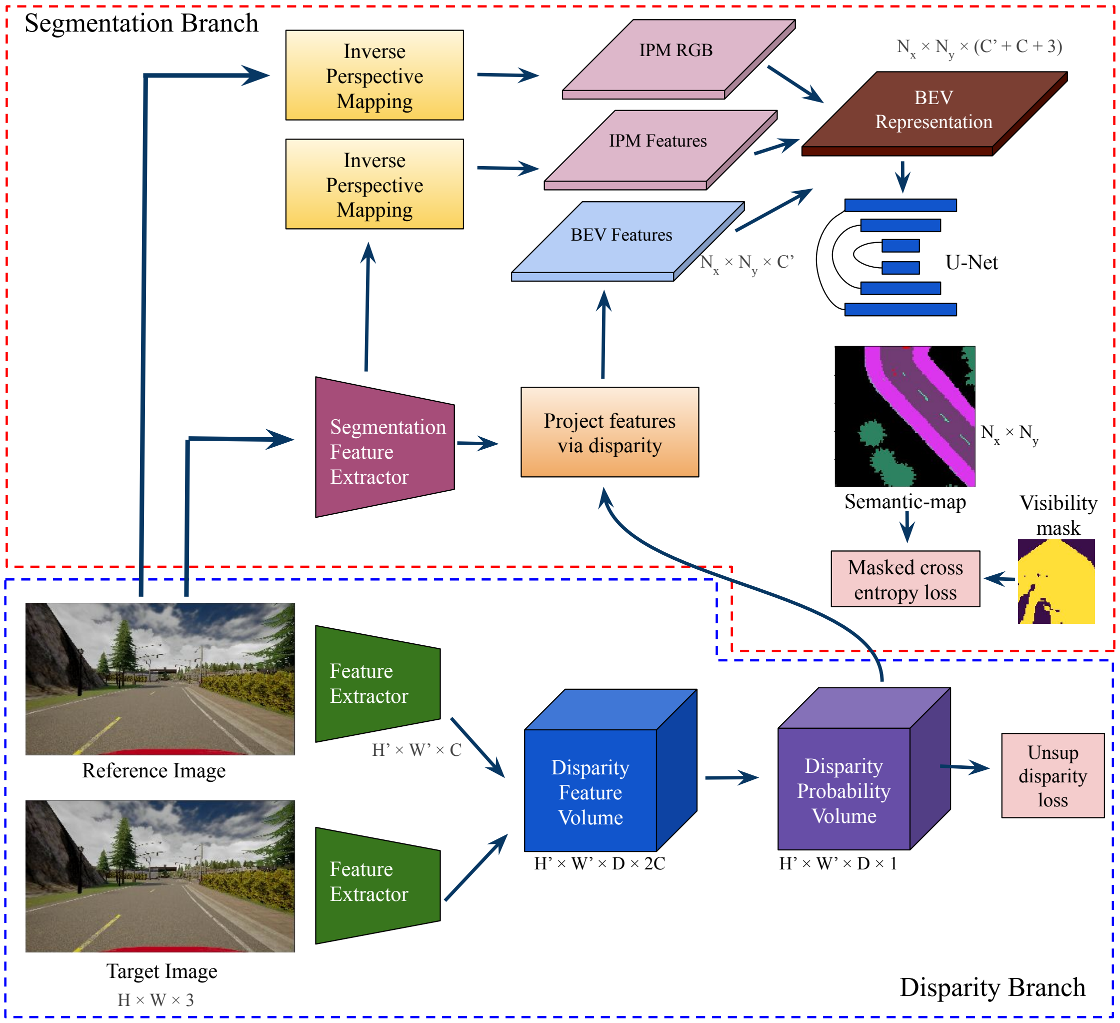

Segmentation Branch
Inverse Perspective Mapping
Inverse Perspective Mapping
IPM RGB
IPM Features
BEV Features
$N_x \times N_y \times C'$
$N_x \times N_y \times (C' + C + 3)$
BEV Representation
U-Net
Segmentation Feature Extractor
Project features via disparity
$N_x \times N_y$
Semantic-map
Visibility mask
Masked cross entropy loss
Reference Image
Feature Extractor
$H' \times W' \times C$
Disparity Feature Volume
$H' \times W' \times D \times 2C$
Disparity Probability Volume
$H' \times W' \times D \times 1$
Unsup disparity loss
Target Image
$H \times W \times 3$
Feature Extractor
Disparity Branch

Figure 4.4: *SBEVNet*-disentangled overview. In this model, the segmentation branch is separate from disparity branch. We first extract image features in both the branches using separate networks. In the disparity branch, a self-supervised disparity loss is used to compute disparity probability volume which is used to project the segmentation features in the bird's eye view. Inverse perspective mapping also applied on the reference image and the segmentation features.

$H' \times W' \times D \times 2C$. We then pass the feature volume through a series of 3D convolution layers with skip connections to learn higher level features. We then apply a 3D convolution block with softmax activation to get a disparity probability volume V_{disp} of size $H' \times W' \times D \times 1$. The softmax is applied on the 3rd dimension of V_{disp} such that for a given image pixel, the sum of disparity probabilities is equal to one. We then apply a self-supervised disparity loss to learn the disparity probabilities. In that loss function, we first warp the target image to the view of the reference image using the disparity probably volume and compute the distance with the reference image.

4.6.2 Segmentation Branch

In the segmentation branch, we extract image features F_{seg} only for the reference image. This is done by passing I_R with a convolutional encoder which is separate from the disparity branch. The image features F_{seg} are used for both front semantic segmentation and bird's eye view segmentation. The front semantic segmentation C_{front} is computed by applying some 2D convolutions on F_{seg}.

$$C_{front} = M_{\text{front}}(F_{seg}) \tag{4.14}$$

The image features F_{seg} are also projected to bird's eye view using the disparity probability volume V_{disp}. To do that, we first create a volume of the image features in the disparity space. This volume V_{seg} is made by masking the image features with disparity probabilities for all disparity levels.

$$V_{seg}[u, v, d, :] = V_{disp}[u, v, d] \cdot F_{seg}[u, v] \forall u, v, d \tag{4.15}$$

This volume is then reduced along height and warped into the bird's eye view space using an operation similar in the previous model. Just like the previous model, we also do inverse perspective mapping $R_{\text{IPM\_feat}}$ and $R_{\text{IPM\_img}}$ on I_R and F_{seg} respectively. These three BEV feature maps are concatenated and passed to a U-Net model to compute the BEV segmentation map.

Chapter 5

Datasets

We use two real world datasets and a synthetic dataset to train and evaluate our approaches. We use the CARLA [4] simulator to generate a synthetic dataset. For the real world dataset, we use the publicly available KITTI dataset and a forest dataset.

5.1 CARLA Dataset

We use the CARLA [4] simulator to generate a synthetic dataset. We use a set of standard paths in the CARLA environment and move the car along those waypoints. We place two cameras on top of the car which look in front. Both of these cameras are 0.2 meters apart and have identical rotation. We also place cameras in the air looking at ground plane to get the ground truth semantic map. To ensure that we evaluate the generalizability of the models, the training and testing are done on entirely different city models in CARLA. Town01, Town02, Town03, and Town04 are used for training, and Town05 is used for testing. To get more diversity in the images, we randomly change the cloudiness, precipitation, precipitation deposits, sun azimuth angle and sun altitude angle. The training set contains 4,000 pairs of stereo images and the testing set contains 926 pairs of stereo images. The classes we use for the semantic map are road, vegetation, car, sidewalk, and building. The bounds of the layout with respect to the camera are -19 to 19 meters in x direction and 1 to 39 meters in the y direction.

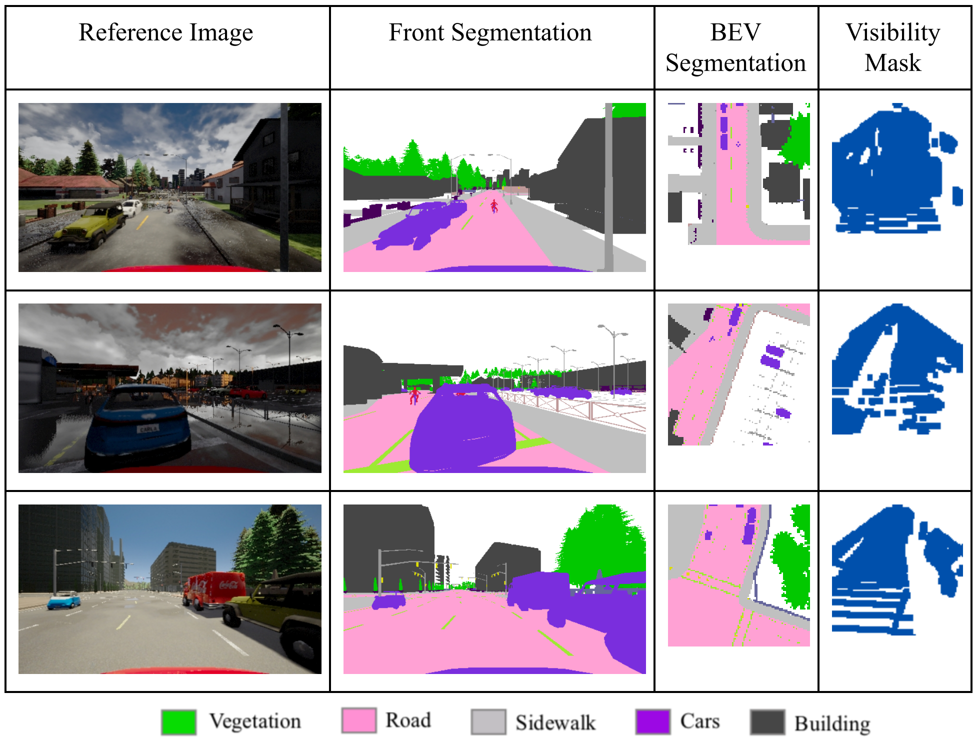


Figure 5.1: Sample frames of our dataset collected using the CARLA [4] simulator. The visibility mask is used to determine which areas in the BEV segmentation map are in the view of the front camera.

5.2 KITTI Dataset

We also evaluate *SBEVNet* on the publicly available KITTI [5] dataset. Similar to Mani et al. [15], we use the KITTI odometery subset and use the SemanticKITTI [1] dataset for labeled ground truth point clouds. LiDAR is used to determine the areas in the BEV map which are visible from the front facing camera. We use the same training/testing split as used by Mani et al. [15], where separate sequences are used for training and testing. The sequences numbered 00, 07 and 10 are used for testing and rest of them are used for training. The training set contains 3,278 stereo image pairs and the testing set contains 1,371 stereo image pairs. The classes we use for the semantic map are road, vegetation, car, sidewalk, and building. The bounds of the layout with respect to the camera are -19 to 19 meters in x direction and 5 to 43 meters in the y direction. Apart from the KITTI odometry subset, we also experiment with the KITTI object subset which only contains cars in the BEV semantic map.

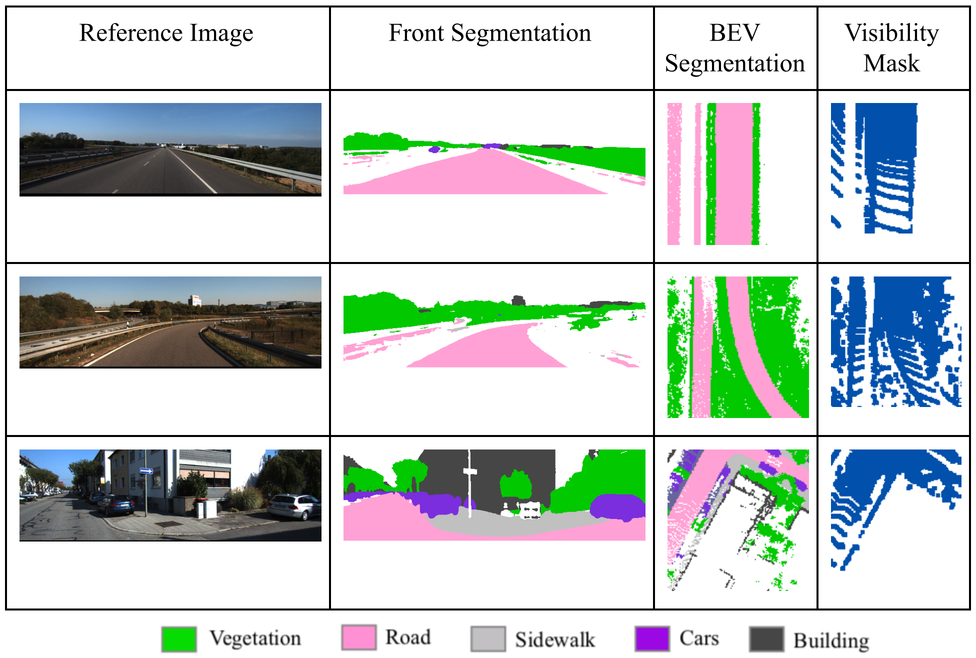


Figure 5.2: Sample frames of the KITTI [5] dataset prepared for BEV segmentation. The visibility mask is used to determine which areas in the BEV segmentation map are in the view of the front camera.

5.3 Forest Dataset

We also evaluate *SBEVNet* on a dataset collected in a forest environment. The dataset is collected by driving a utility task vehicle with a mounted stereo camera rig. The stereo setup contains four cameras vertically on top of each other. The first and the fourth cameras are grayscale near-infrared spectroscopy (NIR), the second camera is a standard RGB camera and the third camera is a thermal camera. However, we only use the two NIR gray-scale cameras for our experiments. The vehicle also contains a Velodyne's VLP-16 LiDAR sensor to capture ground truth 3D data.

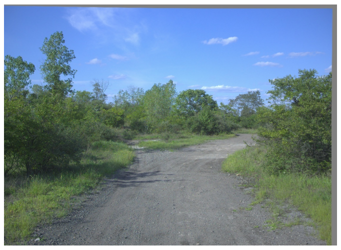
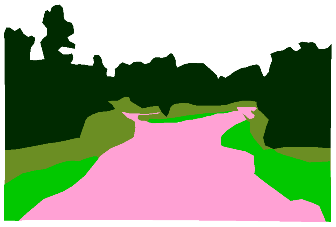
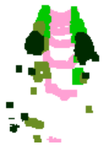
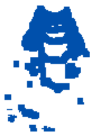
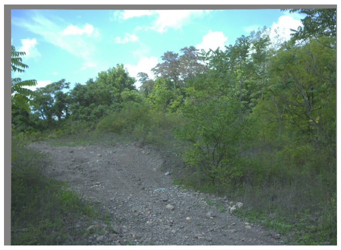
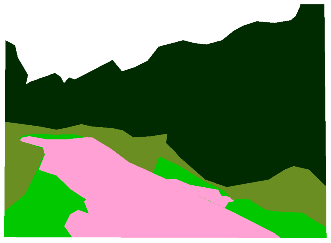
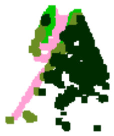
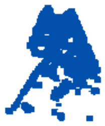
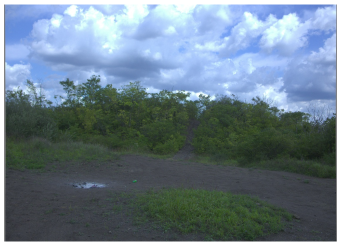
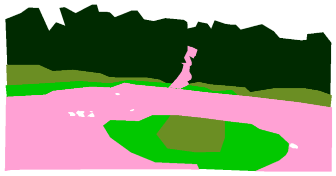
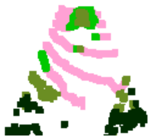
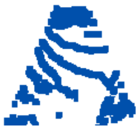
Reference Image	Front Segmentation	BEV Segmentation	Visibility Mask
			
			
			

Figure 5.3: Sample frames from our forest dataset. The BEV segmentation maps are sparse because they are generated using LiDAR data.

First the images are human annotated for standard semantic segmentation. Then we use the LiDAR scan to project the segmentation annotation in the bird's eye view. The LiDAR data is very sparse, which results in very sparse BEV maps. Hence we expand the projected points by few pixels. The visibility mask is also created using LiDAR data.

The training set contains 1,114 stereo image pairs and the testing set contains 369 stereo image pairs. The classes we use for the semantic map are trees, road, rocks, vegetation small and vegetation large. The bounds of the layout with respect to the camera are -7 to 7 meters in x direction and 1 to 15 meters in the y direction.

The main challenges for this dataset are:

1. The stereo pair images are in gray-scale rather than RGB. The lack of color information makes the semantic segmentation harder.
2. This dataset requires vertical stereo instead of horizontal stereo. Hence *SBEVNet* cannot be used as it is.
3. The ground truth BEV segmentation maps are very sparse and of low range.

Dataset	# Training Samples	# Testing Samples	Input Resolution	Classes	BEV Range
CARLA	4,000	936	512x228	road, vegetation, car, sidewalk, building	38m x 38m
KITTI	3,278	1,371	640x256	road, vegetation, car, sidewalk, building	38m x 38m
Forest	1,114	369	512x384	trees, road, rocks, vegetation small, vegetation large	14m x 14m

Table 5.1: Details of CARLA, KITTI and the forest dataset.

Chapter 6

Experiments and Results

6.1 Evaluation Metrics

As not all the regions of the ground truth layout are visible from the camera, we only consider pixels of the layout which are in the field of view. For evaluating the semantic map, we use macro averaged intersection over union (IoU) scores for the layout pixels which are in the visibility mask. We report the IoU scores for each semantic class separately.

6.2 Compared Methods

There are no previously reported quantitative results for the task of stereo layout estimation in our setting. Thus, we evaluate appropriate baselines which are prior works extended to our task.

1. **Pseudo-LiDAR** [21] **+ segmentation**: Uses Pseudo-lidar with PSMNet to generate a 3D point cloud from the input stereo images which is used to project the semantic segmentation of the front view to the bird's eye view. The PSMNet is trained separately on the respective datasets for better performance.
2. **Pseudo-LiDAR** [21] **+ BEV U-Net**: The RGB 3D point projected in the BEV aligned with the ground truth layout is used to train a U-Net segmentation network.

Method	mIoU	Road	Vegetation	Cars	Sidewalk	Building
Pseudo-LiDAR + Segmentaion	25.63	37.64	16.40	35.15	25.44	13.50
Pseudo-LiDAR + BEV U-Net	36.61	63.55	31.87	45.64	29.97	12.01
IPM + BEV U-Net	32.30	66.36	15.24	41.37	32.77	5.78
MonoLayout	22.16	52.88	9.00	16.36	23.17	9.41
MonoLayout + depth	21.85	52.94	9.95	14.07	23.02	9.31
MonoOccupancy + depth	29.49	67.96	16.56	7.66	36.35	18.91
SBEVNet only stereo	36.10	64.74	31.85	39.76	30.01	14.14
SBEVNet stereo + RGB IPM	39.77	65.01	33.20	47.88	33.24	19.53
SBEVNet stereo + features IPM	42.29	71.29	29.79	51.97	38.46	19.95
SBEVNet-CMD	40.10	69.07	32.71	45.45	35.16	18.08
SBEVNet	44.36	72.82	32.07	55.32	40.69	20.78
***SBEVNet*-disentangled**	**46.68**	**73.29**	**39.83**	**56.32**	**42.83**	**21.14**
***SBEVNet* Ensemble**	**47.92**	**75.36**	**35.33**	**60.17**	**44.25**	**24.47**

Table 6.1: Quantitative results of semantic layout estimation on the CARLA dataset.

3. **IPM + BEV U-Net**: Inverse perspective mapping is applied to the input image to project it to the BEV space which is used to train a U-Net segmentation network.

4. **MonoLayout** [15]: This baseline uses MonoLayout to generate BEV semantic map from a single image. Rather than using OpenStreetMap data for adversarial training, we used random samples from the training set itself. The comparison this baseline with stereo/depth based methods is unfair, and is provided only for reference.

5. **MonoLayout** [15] **+ depth**: The input RGB image concatenated with the depth is used as an input to the MonoLayout Model. Depth is generated using PSMNet trained on the respective datasets.

6. **MonoOccupancy** [13] **+ depth**: The input RGB image concatenated with the depth is used as an input to the MonoOccupancy Model.

We also evaluate some variations of our model to perform ablation studies. In ***SBEVNet*** **only stereo** we exclude the IPM features and only use features derived from the feature volume. To gauge the importance of IPM on RGB images and features, we also try applying IPM only on RGB images (***SBEVNet*** **stereo + RGB IPM**) and IPM only on the features of the input image (***SBEVNet*** **stereo**

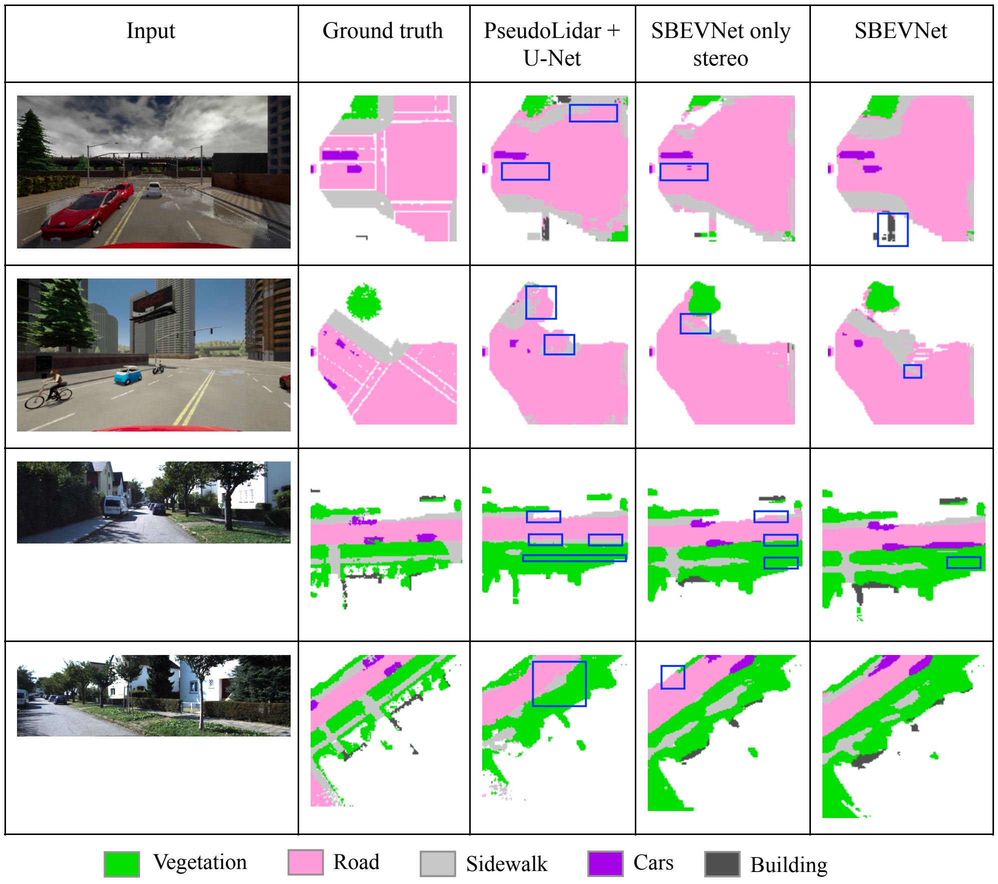


Figure 6.1: Qualitative results on the test set of the CARLA and the KITTI dataset. The major mistakes in the predictions are annotated by a blue rectangle.

+ **features IPM**). We also evaluate the cross modal distillation model ***SBEVNet*-CMD**. Finally, we evaluate our complete model (***SBEVNet***) where we use stereo features and IPM on both RGB image and its features. We also evaluate ***SBEVNet*** **Ensemble** where we take an ensemble of *SBEVNet* with the same architecture but different initialization seeds.

6.3 Network Details

SBEVNet is mainly composed of the following modules: image feature extractor, disparity volume generator, disparity volume refiner, BEV representation generator

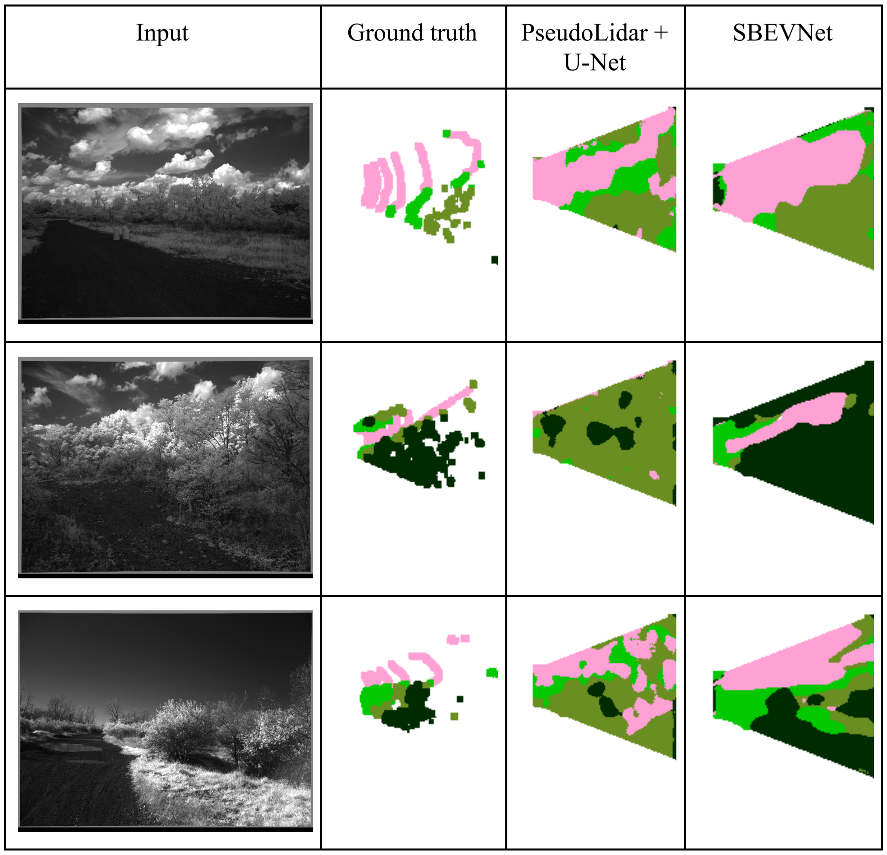


Figure 6.2: Qualitative results on the test set of the forest dataset.

and BEV segmentation head.

6.3.1 Image Feature Extractor

We use the same feature extractor which is used in the basic model of PSMNet [2]. The network architecture uses 4 residual blocks [8] and the kernel size of all convolutions is 3x3. First, three convolutions with 32 channels are applied on the images. Then 4 residual blocks of filters 32, 64, 128 and 128 respectively are applied. The dilation factor of the last residual block is 2. The stride of all convolutions is 1 except the first convolution layer and the second residual block. This results in the image features be 1/4 the resolution of the input image. Every convolution is followed by a batch normalization layer and a ReLU activation.

Method	mIoU	Road	Sidewalk	Cars	Building	Vegetation
Pseudo-LiDAR + segmentation	18.69	35.51	14.56	13.50	12.64	17.26
Pseudo-LiDAR + BEV U-Net	28.97	61.83	21.36	5.55	12.74	43.38
IPM + BEV U-Net	34.93	68.40	26.65	28.49	4.53	46.57
MonoLayout	25.19	64.36	20.53	2.43	2.59	36.05
MonoLayout + depth	21.48	55.80	16.19	1.91	3.03	30.46
MonoOccupancy + depth	29.16	70.52	22.17	7.11	5.25	40.77
SBEVNet only stereo	50.01	78.41	40.16	41.96	30.45	59.05
SBEVNet stereo + RGB IPM	49.56	78.37	39.83	42.47	28.34	58.80
SBEVNet stereo + features IPM	50.60	80.16	41.08	43.64	29.19	58.92
SBEVNet-CMD	50.73	80.59	41.67	43.16	29.13	59.37
SBEVNet	**51.36**	80.23	**41.86**	42.81	**31.35**	**59.43**
***SBEVNet*-disentangled**	48.92	**81.32**	41.80	**44.77**	18.98	57.73
***SBEVNet* Ensemble**	**53.85**	**82.22**	**45.70**	**44.97**	**34.54**	**61.83**

Table 6.2: Quantitative results of semantic layout estimation on the KITTI dataset.

Method	mIoU	Trees	Road	Veg-Small	Veg-Large	Rocks
Pseudo-LiDAR + segmentation	16.04	63.70	2.79	13.70	0.04	0.00
Pseudo-LiDAR + BEV U-Net	19.75	26.35	33.36	20.51	18.51	0.00
MonoLayout	12.31	33.32	5.64	7.70	14.87	0.00
SBEVNet	50.31	86.76	61.62	33.26	46.62	23.28
***SBEVNet*-disentangled**	**55.65**	**86.64**	**64.92**	**34.82**	**49.64**	**42.22**

Table 6.3: Quantitative results of semantic layout estimation on the forest dataset.

6.3.2 Disparity Volume Refiner

The disparity volume refining module is used to learn some additional information from the raw disparity volume generated. It is used to learn to match the features for various disparities and learn the 3D information. We use several 3D convolutions to refine the disparity feature volume which is created using concatenating features of images shifted at various disparities. The disparity volume refining module uses five residual blocks, where each block comprises of two 3D convolutions of kernel size 3x3x3 and 32 filters. Every convolution is followed by a batch normalization layer and a ReLU activation.

6.3.3 BEV Segmentation Head

The BEV segmentation head is applied on the generated BEV feature representation to generate BEV segmentation map. We first apply a U-Net [16] on the BEV feature map. The U-Net model consists of a down-sampling network and an up-sampling network. The down-sampling network consists of 3x3 kernel sized convolution followed by 2x2 max-pooling operations. We use convolutions of filter sizes 64, 128, 256, 512 and 512 respectively. The up-sampling network consists of transposed convolutions with kernel size 3x3 and stride 2. The outputs from the corresponding down-sampling convolutions are also concatenated. After the up-sampling network, we also apply three residual blocks consisting of convolutions with kernel size 3x3 and filter size 256. Finally a pixel-wise classification convolution with kernel size 3x3 and filter size same as number of classes to get the BEV segmentation map.

6.4 Implementation Details

We implemented *SBEVNet* using Pytorch. We use Adam optimizer with the initial learning rate of 0.001 and betas (0.9, 0.999) for training. We use a batch-size of 3 on a Titan X Pascal GPU. We use the same base network which is used in the basic model of PSMNet. The input image size for the CARLA dataset is 512×288, the input image size for the KITTI dataset is 640×256 and the input image size for the forest dataset is 512×384 We report the average scores of multiple runs to account for the stochasticity due to random initialization and other non-deterministic operations in the network.

6.5 Experimental results

We report the IoU scores of all the methods on the CARLA, KITTI [5] and forest dataset in Table 6.1, Table 6.2 and Table 6.3 respectively. As we can see from the tables, *SBEVNet* achieves superior performance on all the datasets. We also observe the increase in performance if we use both stereo information and inverse perspective mapping. We do not use inverse perspective mapping in the forest dataset because the ground is not planar and in some images ground is not visible. IPM yields a greater increase in performance in the CARLA [4] dataset because the ground is perfectly flat. If we use only RGB IPM along with stereo, the results are slightly worse on the KITTI dataset because the ground is not perfectly planar. We see that degradation does not persist if we also use IPM on the image features. We can also see an improvement in performance, if we take an ensemble of *SBEVNet* due to some variance in individual models. For the KITTI dataset, we see a sharp improvement over pseudo-LiDAR approaches because of inaccurate depth estimation. On the other hand, our model does not depend on explicit depth data/model. The results of MonoLayout [15] and MonoOccupancy [13] are inferior due to lack of any camera geometry priors in the network. We also show the qualitative results on the test set of CARLA [4] and KITTI [5] dataset in Figure 6.1. The qualitative results for the forest dataset are shown in Figure 6.2. We see that in certain regions *SBEVNet* gives outputs closer to the ground truth. For example, Psuedo-lidar fails to segment cars in the KITTI dataset. For the CARLA and the forest dataset, there is an improvement in results with the *SBEVNet*-disentangle model for all the classes. Whereas for the KITTI dataset, the improvement is only for the road and cars class. We also observe a drop in quality in the estimated layout as we move further from the camera.

6.5.1 Ablation Study

IPM on RGB image

For the CARLA dataset, we observe an increase of 3.67 in the mIoU score, on concatenating IPM RGB with the stereo features. We observe an increase in IoU scores for all the classes, with the biggest increase of 8.12 in the cars class. For the KITTI dataset, there is a small decrease of 0.45 in the mIoU score. This is because, the ground is not perfectly planar, hence the IPM RGB images do not exactly align with the ground truth layout.

IPM on image features

If we apply IPM on the features of the input image and concatenate it with the features from the stereo branch, we see an improvement in both the datasets. The improvements in mIoU scores are 6.19 and 0.59 for the CARLA and KITTI dataset.

The improvement is higher compared to the RGB IPM because image features contain higher level information which is transformed to the BEV space.

IPM on both RGB image and image features

We see the greatest improvement if we apply IPM on both the RGB image and the features of the RGB image. The improvements in mIoU scores are 8.26 and 1.35 for the CARLA and KITTI dataset respectively. This is because the model is able to exploit the different information present in $R_{\text{IPM\_feat}}$ and R_{stereo}.

Cross modal distillation

The performance of *SBEVNet*-CDM is in between of stereo only *SBEVNet* and full *SBEVNet*. We see an improvement of 4.00 and 0.72 in the mIoU scores on the CARLA and KITTI dataset, if we train the stereo model using cross modal distillation via IPM features. During inference, the architecture of *SBEVNet*-CMD is the same as the stereo only *SBEVNet*. This shows that CMD is able to transfer most of the IPM knowledge to the stereo branch.

Minimizing distance between first K features

We also evaluate the approach, where we try minimizing the L1 distance between all the channels of the IPM features and stereo features. We observe mIoU scores of 32.27 and 50.03 for the CARLA and KITTI dataset respectively. This is worse than the mIoU scores achieved by minimizing the distance between first K channels. This is because, if we enforce all stereo branch channels to be the same as IPM branch channel, the stereo branch is unable to learn information that is not present in the IPM features.

6.6 Further Analysis

6.6.1 3D feature volume analysis

One claim of our approach is that our model learns 3D information without any explicit depth/disparity supervision. To validate this claim, we use the learned 3D

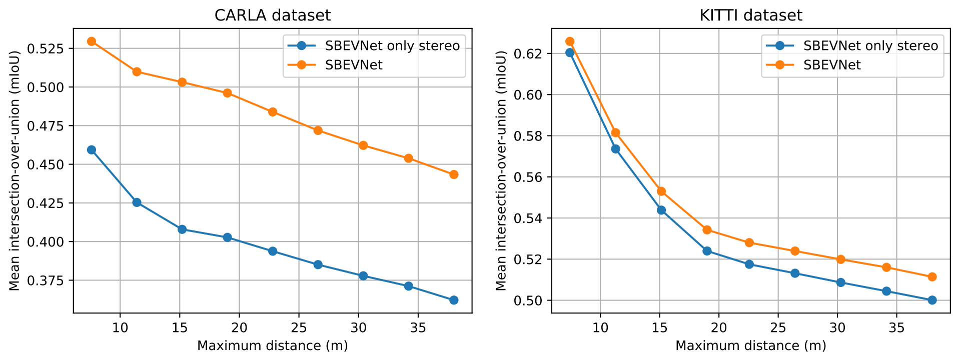


Figure 6.3: Performance as a function of maximum distance from the camera. We consider the pixels in the BEV layout which are atmost a certain distance away from the camera.

feature volume to perform disparity estimation. We freeze all the weights and add a small 3D convolution layer to perform disparity regression on the learned feature volume. We also observe that the feature volume which is trained with cross modal distillation via IPM performs better at the task of disparity estimation. For the CARLA dataset, we find that the *SBEVNet* only stereo model has a 3-pixel error of **7.92** and with cross model distillation the 3-pixel error goes down to **6.84**.

6.6.2 Distance from camera

We wish to quantify how our system performs as we move away from the camera. Hence, we plot the IoU scores for the pixels in the BEV layout which are more than a given distance from the camera (Figure 6.4) and for the pixels which are less than a given distance from the camera (Figure 6.3). For both the KITTI and the CARLA dataset, we observe that there is a drop in performance as the distance from the camera increases. We also observe that *SBEVNet* outperforms the stereo only *SBEVNet* at all distances from the camera.

6.6.3 Amount of training data

We also quantify how the performance of *SBEVNet* changes with the number of data-points in the training set, while keeping the test set same. On the CARLA dataset, with just 10% of the training data, we get mIoU score of 26.10 compared to

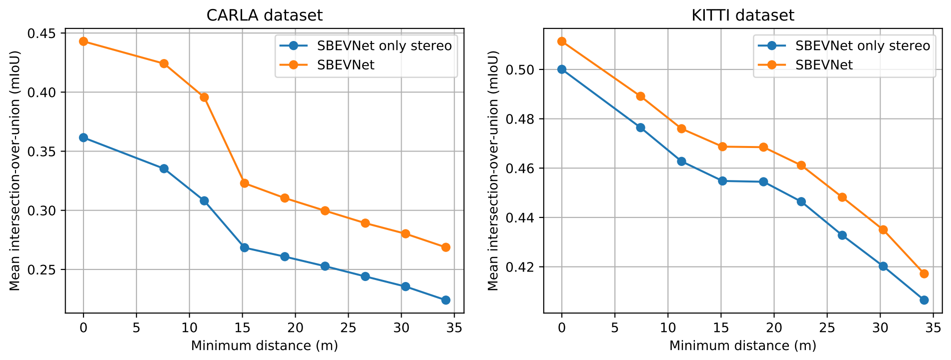


Figure 6.4: Performance as a function of minimum distance from the camera. We consider the pixels in the BEV layout which are atleast a certain distance away from the camera.

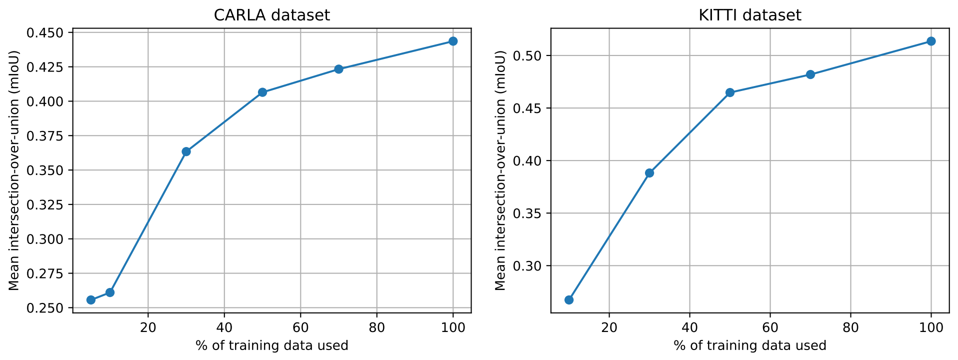


Figure 6.5: Performance of the system as a function of amount of training data used.

the mIoU score of 44.36 with all the training data. For both the datasets, performance starts to saturate when we use 100% of the training data. This shows that 3,000-4,000 training data points are sufficient for getting the optimal performance from *SBEVNet*.

6.6.4 Performance evolution during training

We also perform a qualitative analysis (Figure 6.6) of how the performance of the model changes during training. We observe, during the initial stages of training, the model learns to identify very course grained attributes such as the direction of the road. However, there is ambiguity in detailed attributes such as size and exact

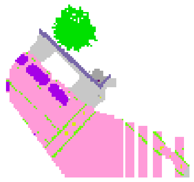
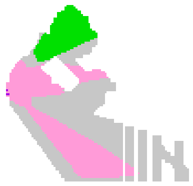
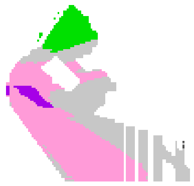
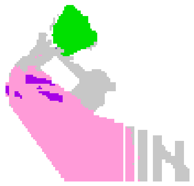
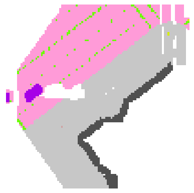
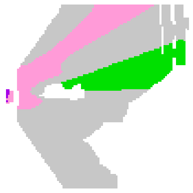
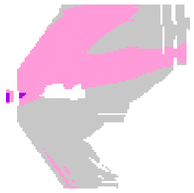
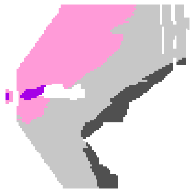
Input	Ground truth	Epoch 1	Epoch 5	Epoch 30
				
				

Figure 6.6: Evolution of predicted BEV layouts for different epochs during training

position. During the later stages of training, the model learns to identify the smaller objects and exact positioning in the BEV space.

6.6.5 Ensemble

We observe some variance in the performance of the models on training with different random seeds. For *SBEVNet* we observe a standard deviation of 2.16 and 2.46 in the mIoU scores for the KITTI and CARLA dataset respectively. Due to the diversity in outputs of the individual models [28], we see an improvement in the performance, if we take an ensemble of individual models. We observe an absolute improvement of 2.49 and 3.56 in the mIoU scores for the KITTI and CARLA dataset respectively.

6.6.6 Inference time

On NVIDIA Titan X GPU, with the batch size equal to 1, the inference time of stereo only *SBEVNet* for one input pair is 0.1307s on the average. For the full *SBEVNet* the inference time is 0.1449s on the average, which is slightly higher than the stereo only model. This inference speed is sufficient for majority of robotics applications. The majority of computation is done in processing the 3D feature volume with 3D convolutions.

Chapter 7

Conclusions

7.1 Summary

In this thesis we proposed *SBEVNet*, an end-to-end network to estimate the bird's eye view layout using a pair of stereo images. We observe improvement in the IoU scores compared with approaches that are not end-to-end or do not use geometry. We also showed that combining inverse perspective mapping with the projected disparity feature volume gives better performance. We also show that, using cross modal distillation to transfer knowledge from IPM features to the stereo features gives us an improvement in results. We also proposed a variant of *SBEVNet* which disentangles disparity and segmentation, yielding better results.

7.2 Limitations and Future Work

Our work achieves good IoU scores on several classes, but still does not perform very good on some classes such as the building class in CARLA and KITTI dataset and vegetation class in the CARLA dataset. Another limitation of our work is that the performance drops as the distance from the camera increases. This is because far away objects have a very low disparity. Hence, using a depth volume rather than a disparity volume might give us improvements. To create a larger dataset, we would look into using areal images for the forest dataset. We are also looking into using domain adaptation to transfer knowledge from simulation to real world. As a part of future work, we are also looking into other bird's eye view applications such as height map generation and object detection.

Appendix A

Appendix

A.1 Results on KITTI object dataset

We also compare our method with the published numbers on the KITTI Object dataset. We use the dataset and annotations provided by [15]. Here, there is only a single cars class. We compare our approach with published monocular approaches and 3D object detection approach on pseudo lidar with stereo input. AVOD + pseudo lidar is an object detection method which also uses the large sceneflow dataset for pre-training. More details of these baseline models can be found in [15]. Table A.1 shows the numbers on the KITTI object split which are provided by [15]. We observe that our method achieves an improvement in the mIoU scores over all the other methods. For segmentation, pixel level mAP is not a good metric as is does not consider false negatives. We still report the pixel level mAP scores for reference.

Method	mIoU	mAP (pixel level)
ENet + Pseudo lidar input(Monodepth2) PointRCNN	0.24	0.37
PointRCNN + Pseudo lidar input(Monodepth2)	0.26	0.43
MonoLayout	0.26	0.41
AVOD + Pseudo lidar input(PSMNet) (Stereo)	0.43	**0.59**
Our (Stereo)	**0.46**	**0.59**

Table A.1: Quantitative results of BEV car segmentation on the KITTI object dataset. All the results except *SBEVNet* are excerpted from [15]

Bibliography

[1] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9297–9307, 2019. 5.2

[2] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018. 1, 3.2, 4.3, 6.3.1

[3] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya 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. 3.3

[4] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017. (document), 1, 5, 5.1, 5.1, 6.5

[5] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. (document), 1, 5.2, 5.2, 6.5

[6] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li. Group-wise correlation stereo network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3273–3282, 2019. 1, 3.2, 4.3

[7] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2827–2836, 2016. 4.5

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6.3.1

[9] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–75, 2017. 1, 3.2, 4.3

[10] Sameh Khamis, Sean Fanello, Christoph Rhemann, Adarsh Kowdle, Julien Valentin, and Shahram Izadi. StereoNet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 573–590, 2018. 1, 3.2, 4.3

[11] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3D proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018. 3.3

[12] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Wei Chen, Linbo Qiao, Li Zhou, and Jianfeng Zhang. Learning for disparity estimation through feature constancy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2811–2820, 2018. 1, 3.2, 4.3

[13] Chenyang Lu, Marinus Jacobus Gerardus van de Molengraft, and Gijs Dubbelman. Monocular semantic occupancy grid mapping with convolutional variational encoder–decoder networks. *IEEE Robotics and Automation Letters*, 4(2):445–452, 2019. 3.1, 6, 6.5

[14] Hanspeter A Mallot, Heinrich H Bülthoff, JJ Little, and Stefan Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological cybernetics*, 64(3):177–185, 1991. 1

[15] Kaustubh Mani, Swapnil Daga, Shubhika Garg, Sai Shankar Narasimhan, Madhava Krishna, and Krishna Murthy Jatavallabhula. MonoLayout: Amodal scene layout from a single image. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1689–1697, 2020. (document), 3.1, 5.2, 4, 5, 6.5, A.1, A.1

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 4.4.3, 6.3.3

[17] Samuel Schulter, Menghua Zhai, Nathan Jacobs, and Manmohan Chandraker. Learning to look around objects for top-view representations of outdoor scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 787–802, 2018. 3.1

[18] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019. 3.3

[19] Siddharth Srivastava, Frederic Jurie, and Gaurav Sharma. Learning 2D to 3D lifting for object detection in 3d for autonomous vehicles. *arXiv preprint*

arXiv:1904.08494, 2019. 3.3

[20] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018. 1, 3.2, 4.3

[21] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8445–8453, 2019. 3.3, 1, 2

[22] Yan Wang, Zihang Lai, Gao Huang, Brian H Wang, Laurens Van Der Maaten, Mark Campbell, and Kilian Q Weinberger. Anytime stereo image depth estimation on mobile devices. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5893–5900. IEEE, 2019. 1, 3.2, 4.3

[23] Ziyan Wang, Buyu Liu, Samuel Schulter, and Manmohan Chandraker. A parametric top-view representation of complex road scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10325–10333, 2019. 3.1

[24] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018. 3.3

[25] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5524, 2019. 3.2

[26] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-LiDAR++: Accurate depth for 3D object detection in autonomous driving. *arXiv preprint arXiv:1906.06310*, 2019. 3.3

[27] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. GA-Net: Guided aggregation net for end-to-end stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 185–194, 2019. 1, 3.2, 4.3

[28] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms.* CRC press, 2012. 6.6.5