7-7-2021

Monocular 3D Object Detection via Ego View-to-Bird's Eye View Translation

Atharva Arun Tembe
at2216@rit.edu

Monocular 3D Object Detection via Ego View-to-Bird's Eye View Translation

Atharva Arun Tembe

Monocular 3D Object Detection via Ego View-to-Bird's Eye View Translation

Atharva Arun Tembe

July 07, 2021

A Thesis Submitted
in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in
Computer Engineering

RIT Kate Gleason College of Engineering

Department of Computer Engineering

Monocular 3D Object Detection via Ego View-to-Bird's Eye View Translation

Atharva Arun Tembe

Committee Approval:

Dr. Guoyu Lu | *Advisor* Date
Chester F. Carlson Center for Imaging Science

Dr. Andres Kwasinski Date
Department of Computer Engineering

Dr. Alexander Loui Date
Department of Computer Engineering

Acknowledgement

I would like to take this opportunity to thank my advisor Dr. Guoyu Lu for his support and guidance in academics. I am thankful to Dr. Andres Kwasinski and Dr. Alexander Loui for being on my thesis committee. I am grateful for the members of the Intelligent Vision and Sensing Lab who were all so welcoming and helpful. I would also like to thank my family and friends for their love and support during this challenging endeavor.

Abstract

The advanced development in autonomous agents like self-driving cars can be attributed to computer vision, a branch of artificial intelligence that enables software to understand the content of image and video. These autonomous agents require a three-dimensional modelling of its surrounding in order to operate reliably in the real-world. Despite the significant progress of 2D object detectors, they have a critical limitation in location sensitive applications as they do not provide accurate physical information of objects in 3D space. 3D object detection is a promising topic that can provide relevant solutions which could improve existing 2D based applications. Due to the advancements in deep learning methods and relevant datasets, the task of 3D scene understanding has evolved greatly in the past few years. 3D object detection and localization are crucial in autonomous driving tasks such as obstacle avoidance, path planning and motion control. Traditionally, there have been successful methods towards 3D object detection but they rely on highly expensive 3D LiDAR sensors for accurate depth information. On the other hand, 3D object detection from single monocular images is inexpensive but lacks in accuracy. The primary reason for such a disparity in performance is that the monocular image-based methods attempt at inferring 3D information from 2D images. In this work, we try to bridge the performance gap observed in single image input by introducing different mapping strategies between the 2D image data and its corresponding 3D representation and use it to perform object detection in 3D. The performance of the proposed method is evaluated on the popular KITTI 3D object detection benchmark dataset.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

An autonomous intelligent agent requires an accurate perception of its environment in order to operate reliably. A typical framework of an autonomous system consists of environment perception, self-localization, mapping, path planning and control. The perception system of an autonomous vehicle (AV) converts sensory information into semantic data such as to identify road agents like vehicles, pedestrians, cyclists etc. Object detection constitutes an important function of the perception system. There has been a lot of work in object detection with most of them using 2D detection methods. The 2D object detection has shown great progress however, the task of 3D object detection (3DOD) has been a difficult challenge. The 2D methods detect objects on the image plane lacking depth information of the scene which is important for tasks such as object perception and localization. The 3DOD on the other hand introduces a third dimension revealing the depth information which further helps in determining the object size and position in 3D space.

The 3DOD methods use a wide variety of artificial sensors for the task of perception. The most commonly used are passive sensors such as monocular, stereo cameras and active sensors such as LiDAR. The monocular cameras are readily available and inexpensive sensors. They provide scene information in the form of pixel intensities revealing the texture and shape properties. One disadvantage of monocular camera is that they lack depth information which is vital for object size and position inference. A stereo camera configuration could be used to recover the depth channel information using more

computationally expensive procedures. The LiDAR sensors emit laser beams and calculate the time difference to reach the obstacle and back. With the help of the time difference, the distance and position of the obstacle is determined and stored in a 3D form called as a point cloud. This point cloud representation is sparse with samples not uniformly distributed. Although LiDAR sensors provide precise depth measurements, they are expensive and constitute a large equipment.

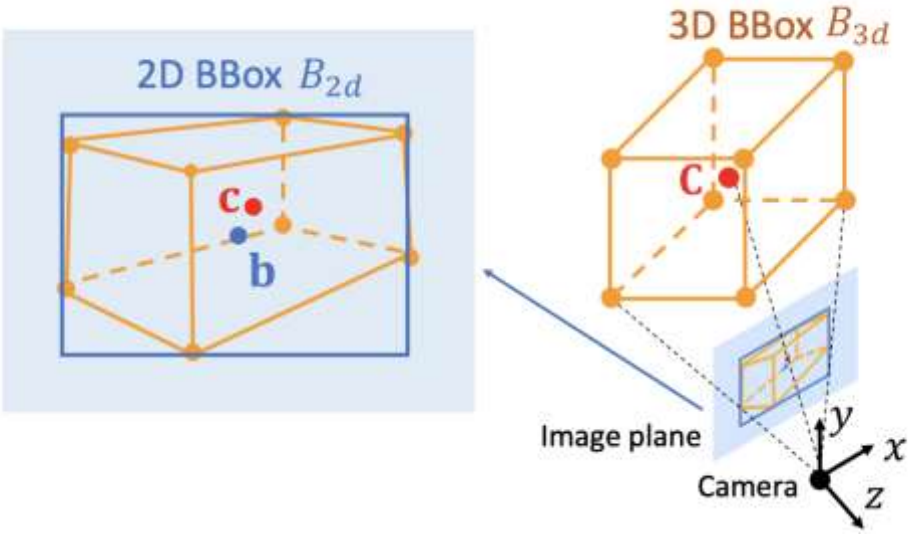

Figure 1.1: Basic difference between a 2D and a 3D bounding box [23]

Due to the availability of fully annotated and synchronized 3D LiDAR-camera datasets such as KITTI [15], nuScenes [17] and Waymo Open Dataset [16], the research for 3DOD has shifted more towards data driven deep learning techniques. The 3DOD methods could be divided into two categories- monocular image and point cloud-based methods. The monocular image-based approach uses only a single RGB image to predict a 3D bounding box. In monocular images, there is no availability of depth information. Hence, most approaches first predict 2D bounding boxes and then infer 3D bounding boxes using geometrical constraints [2], neural networks [1] or 3D model matching [3], [4] methods.

Most of the monocular methods only predict objects from a front facing camera ignoring those on the side and the rear. The main drawback of the monocular methods is they lack depth channel, limiting the object detection accuracy especially for faraway or occluded objects. Wang et al. [39] in their work try to address this drawback by using a Convolutional Neural Network (CNN) [5] architecture to predict the dense depth maps of the monocular images prior to regressing the 3D object bounding box.

The point cloud-based methods for 3DOD could be divided into two subcategories- projection and volumetric convolution methods. The projection methods transform the 3D point cloud representations into 2D images via plane [6], spherical [7] or bird eye view (BEV) [8], [9], [10] projections. A standard 2D object detection model is used to process the 2D projection images and a 3D bounding box is predicted using position and dimension regression. The volumetric methods assume the objects to be present in a fixed size voxels or 3D grids [11], [12] and transform the raw point clouds into volumetric structures. As the data is in 3D, the volumetric methods require computationally expensive 3D convolutions. The projection category among the point cloud methods is more popular due to its propinquity to image object detection models. Also, it offers better trade-off between time complexity and object detection performance.

The point clouds do not provide any texture information while monocular images do not capture depth information. In order to improve the performance, some methods [13], [14] use fusion of point clouds and image modalities using different fusion schemes. Due to the combination of different sensor modalities, these methods achieve state of the art results. They exploit accurate depth information obtained from sparse LiDAR and texture information from monocular images to enhance the 3DOD performance. Although, the 3D sensor-based methods show promising results, they are expensive, significantly increasing the overall manufacturing cost. Hence, a need to develop accurate 3DOD detection methods that rely only on inexpensive sensors such as a single monocular camera is highly practical.

1.2 Motivation

Given an image, the task of an object detector involves localization of all the object instances in the scene and determining the bounding boxes of a certain class. Object detection is widely used in many applications including robotics, augmented reality and autonomous driving. Recently, due to the advancements in deep learning methods and relevant datasets, 2D object detection, classification and semantic segmentation solutions have greatly improved. Despite the significant progress achieved by 2D object detection methods, the 3D understanding of the real-world objects is still an open problem. The 3D based systems are more complicated as compared to the 2D systems mainly due to the convoluted representation of 3D data. The 3D data of the scene is represented using point clouds, volumes, meshes as compared to pixel grid representation of 2D images. The 2D detection methods have a limitation in applications such as autonomous driving and robot manipulation which are location sensitive and require physically accurate information of objects in 3D space.

Successful modern 3DOD methods heavily rely on highly expensive 3D sensors such as LiDAR and depth cameras which provide 3D information of the entire scene. Major disadvantages of 3D sensor-based methods are- (1) calibration process required for LiDAR-camera synchronization; (2) expensive LiDAR equipment; (3) limited working range of depth cameras. On the contrary, a single camera is much cheaper and can capture the scene up to 80 meters. Although, there has been some research on monocular 3DOD [1], [18], [19], the performance of monocular methods is drastically low as compared to the LiDAR based methods. This makes monocular 3DOD an unsolved problem with a scope of improvement especially in terms of accuracy.

There are state of the art 3DOD methods [8], [10], [25] which project the 3D point cloud data into its corresponding 2D BEV image representation as an intermediate step before feeding them to an object detection network. The accuracy of these methods depends on

the quality of the generated BEV images. Generative Adversarial Networks (GANs) have become very popular recently in image-to-image translation tasks. Hence, by leveraging GANs, a monocular RGB image to BEV image translation would be possible. The quality of training data affects the final performance of a GAN network. This thesis aims at developing better training strategies for the GAN network for a better BEV image inference given a single monocular image. By enhancing the quality of the generated BEV images, the 3DOD accuracy would be boosted. The performance of our method is evaluated using KITTI 3D object detection benchmark dataset [15].

1.3 Contributions

The principal contributions of this thesis research are outlined as below:

- Examine 3D object detection methods that predict bounding boxes based on point cloud BEV image projection of the monocular RGB image.
- Introduce a novel method to predict a BEV image given a monocular RGB image as input using a GAN based network.
- Compare the effect of using dense versus sparse BEV image projection on the performance of a GAN based network
- Compare and evaluate the effect of using dense versus sparse BEV image projection on the 3DOD performance.
- Analyze and compare the proposed method with other monocular image-based 3D object detection methods.

1.4 Document Structure

Chapter 2 discusses the related work on 3DOD along with a background on Convolutional Neural Networks, Generative Adversarial Networks and an object detection network. It also provides an overview of different sensor modalities used in 3DOD such as LiDAR

and its data representation like Pseudo-LiDAR. Chapter 3 will discuss the methodology, that includes the RGB image-to-BEV translation network along with the training data preprocessing and the 3D object detection network used for the experimentation. Chapter 4 outlines the training dataset, the hyperparameters used along with the implementation details. Chapter 5 will analyze the qualitative and quantitative results from experimentations on the proposed 3DOD method. Chapter 6 will summarize with a conclusion giving directions for future work.

Chapter 2

Background

2.1 Convolutional Neural Networks

The Convolutional Neural Networks (CNN) are deep learning algorithms that take an image as an input and learn appropriate features of the image using different operations for the required learning task. The CNN architecture is inspired by the connectivity pattern of the neurons in the human brain. It consists of a convolutional layer, pooling layer, activation layer and a fully connected layer. The role of convolutional layer is to extract the high level and low-level features such as edges, color, gradient from the image. The element responsible for the convolutional operation is called as the filter. There are multiple filters which slide with a predefined stride and apply convolution on every region of the image.

The pooling layer is responsible for reducing the spatial size of the convolved features of the input image. This dimensionality reduction helps in reducing the computational power required to process the image data. The pooling operation also helps in training of the model by extracting dominant features which are rotational and positional invariant. There are two types of popularly used pooling methods namely max pooling and average pooling. The max pooling returns the maximum value while the average pooling returns the average value of the image region covered by the filter. The activation function is applied at the end of the convolutional layer. It is used to output a non-linear transformation of the input signal. The output of the last convolutional or pooling layer is flattened and

fed to a fully connected layer. The fully connected layer does a non-linear combination of all the features and transforms them into an n-dimensional representation.

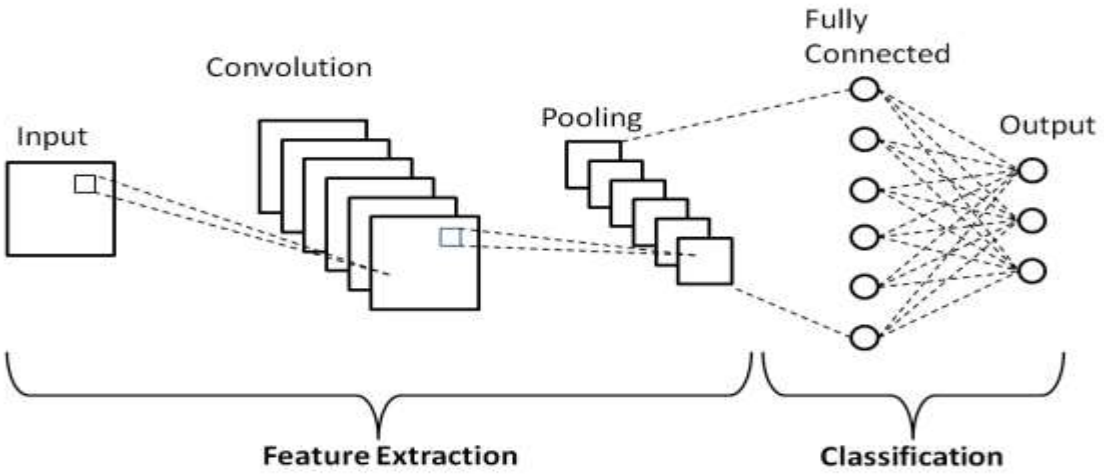

Figure 2.1: Sample architecture of a Convolutional Neural Network [21]

2.1.1 Generative Adversarial Networks

Generative Adversarial Networks or GANs, are generative models based on deep learning methods such as CNN. They were first introduced by GoodFellow et. al. [22] in 2014. Generative modelling is an unsupervised learning task which involves learning the patterns in the input data in such a way that the model could generate new samples that could plausibly represent those drawn from the original dataset. The GAN model architecture consists of two CNNs namely a generator model and a discriminator model. The role of the generator model is to generate new plausible examples from the problem domain while the discriminator is responsible to classify whether the generated examples are real or fake.

The generator model takes in a random vector of a fixed length as input and generates a sample in the domain. The discriminator takes a sample from the domain created by the generator and outputs a binary class label as real or fake. Both the generator and the

discriminator models are trained together in a zero-sum game or in an adversarial way. The discriminator is updated in order to get better at discriminating between real and fake samples while the generator is updated based on how well its generated samples fool the discriminator. Some compelling application of GANs include image-to-image translation, generating super resolution images, generating realistic faces even though the face does not belong to any real person.

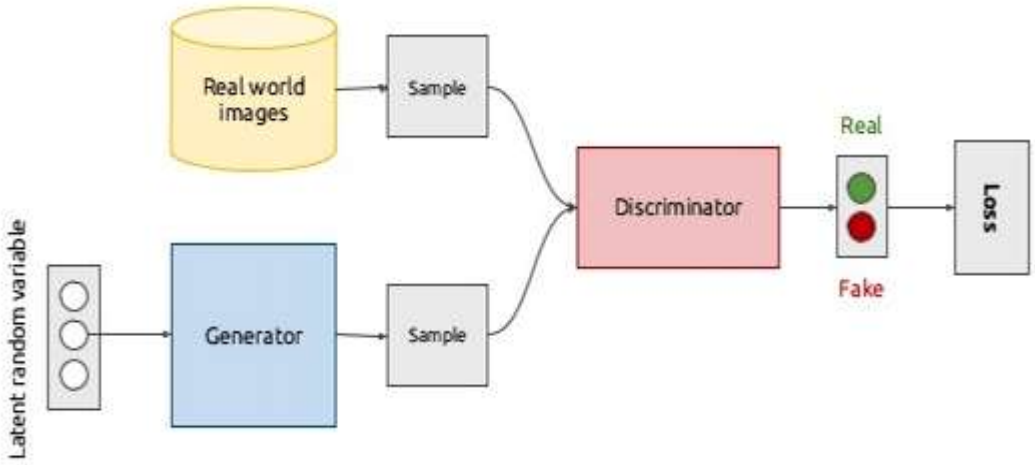

Figure 2.2: Sample architecture of a Generative Adversarial Network [24]

2.1.2 CenterNet: Objects as Points

There are two types of approaches used to regress bounding boxes around an object in anchor free object detection namely the center-based approach and the keypoint based approach. The center-based approach [28], [29] uses the object center or any object point to define positive and negative samples. From the predicted positive samples, the distance to the four bounding box coordinates is regressed. On the contrary, the keypoint based approach [26], [27] first predicts the predefined key points from the network which are

then used for the task of object detection. Objects as Points [30] also called as CenterNet is an object detection method which follows the keypoint based approach. In this, the center of the box is considered as an object as well as a keypoint and from the predicted box center, the object bounding box features are estimated.

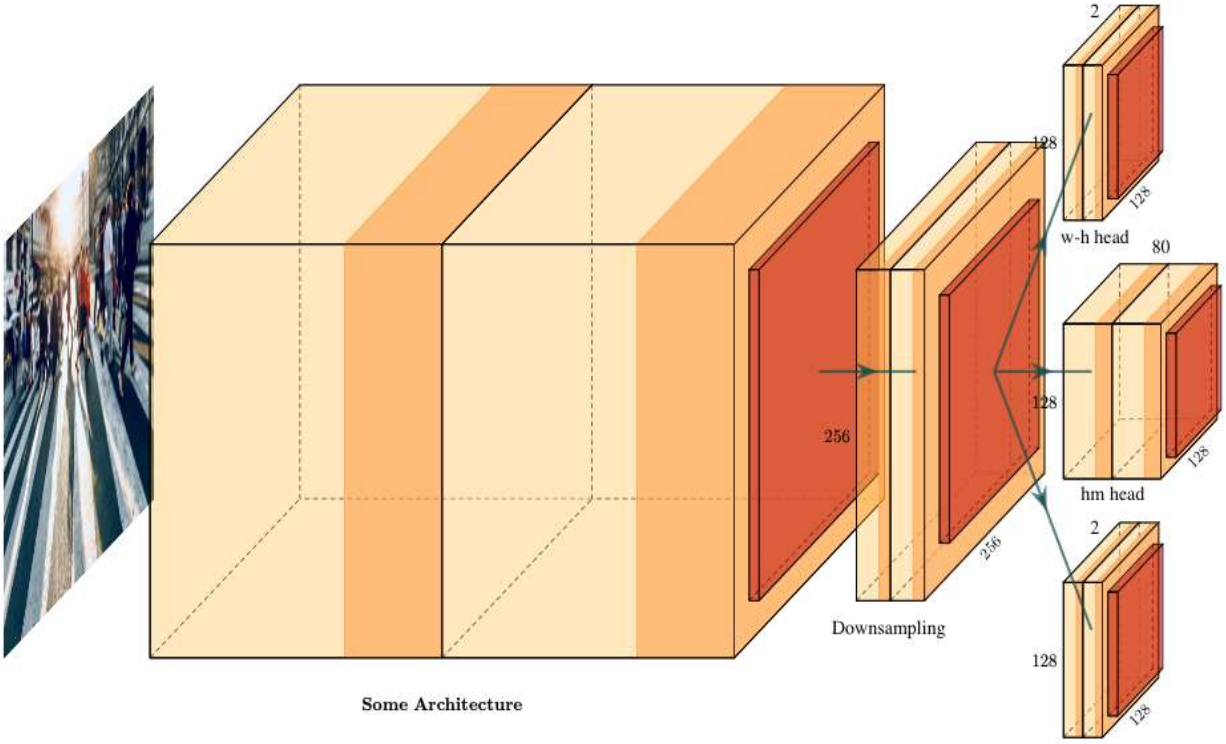

Figure 2.3: CenterNet model workflow. [33]

Figure 2.3 proposes the workflow of the method. The input image is passed through a CNN feature extractor such as ResNet [31] or a Deep layer Aggregation Network [32] to deduce feature maps which are further down sampled using a predefined stride. For every forward pass through the network, three heads are predicted namely the heatmap head, dimension head and the offset head. The heatmap head is used for the estimation of the box center (keypoint) for a particular class. This is achieved with the help of a heatmap variant focal loss. The heatmap head keypoint connects the dimension and the offset head for bounding box prediction. The dimension head predicts the dimension of the bounding box

viz. height and width. The box dimensions are regressed using a standard L1-norm loss of the ground truth and the predicted width and height. The offset head is used to recover any discretization error introduced due to the down sampling of input features. It predicts the offset observed between the box center coordinates in a high-resolution image and that predicted in a low-resolution feature map. A standard L1-norm loss function is used to solve for the predicted and ground truth offset values.

2.1.3 PSMNet

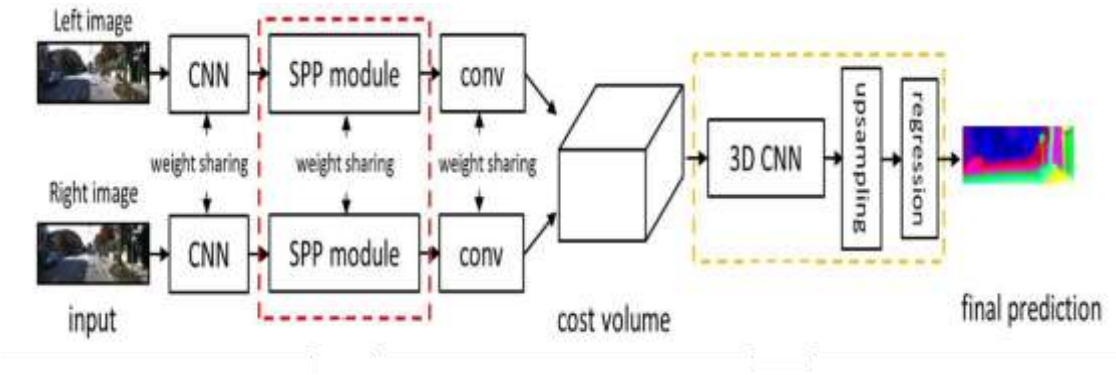

Figure 2.4: Architecture overview of PSMNet [43]

Chang et al. propose a pyramid stereo matching network (PSMNet) [43] for the task of depth estimation from stereo images. Figure 2.4 denotes the PSMNet architecture. The network takes left and right stereo images as input and predicts feature maps using a CNN based module. The feature maps are then provided to a spatial pyramid pooling (SPP) module which extracts hierarchical context information and learns the relationship between different object features and their subregions. The SPP module involves and concatenates different levels of features to achieve stereo matching. The SPP feature maps generated from the left and right images are then connected across each disparity level resulting in a

4D cost volume (height × width × disparity × feature size).

The 3D CNN module facilitates the cost volume regularization using a stacked hourglass (encoder-decoder) based architecture. It has three main hourglass networks having three outputs each of which generates a disparity map. All the hourglass networks are trained using a smooth L1 loss and the total loss is calculated as the weighted summation of all the three losses. The final disparity map is the output from the last network.

2.2 LiDAR and its Data Representation

LiDAR or light detection and ranging is a remote sensing method which collects measurements used to create 3D models of the objects and their surrounding environment. The LiDAR uses light energy to gauge the spatial relationships and shapes of the objects by measuring the time taken by the signal to bounce off an object and return to the scanner. The data provided by the LiDAR is rich in-depth information but lacks in texture and color information.

Point clouds are dynamic information storage technology which can handle and manipulate large spatial data for varying downstream tasks. Once the LiDAR readings are measured and processed, they are stored in the form of 3D point clouds. The 3D point clouds are a large collection of 3D coordinates, which include x, y, z along with additional attributes such as reflectance values. Unlike image data representation, the 3D point cloud is sparse and unbounded.

2.2.1 Pseudo-LiDAR

Depth estimation is the task of estimating the scene depth using monocular or stereo images. In this, the per pixel depth value of the image is derived using stereo vision, geometry or deep learning methods. Wang et. al. [39] introduced the concept of Pseudo-

LiDAR, where the pixelwise image depth information is lifted to a 3D point cloud representation, mimicking the original 3D LiDAR point clouds.

Given a disparity map Y of stereo images, the depth map D is derived using the transform as shown in (1).

$$D(u,v) = \frac{fu * b}{Y(u,v)} \tag{1}$$

where, fu is the horizontal focal length and b is the baseline (horizontal offset) of the pair of cameras. Using depth information D, the 3D coordinates (x, y, z) of every image pixel (u, v) is derived in camera coordinates using the following transforms in (2), (3), and (4).

$$z = D(u,v) \tag{2}$$

$$x = \frac{(u - cu)^{*}z}{fu} \tag{3}$$

$$y = \frac{(v - cv)^{*}z}{fv} \tag{4}$$

where, fv represents the vertical focal length and (cu, cv) is the pixel location corresponding to the camera centre. In addition to depth, the original LiDAR also provides the reflectance value. As there is no such information in Pseudo-LiDAR, the reflectance value is set to 1.0 for every pseudo-LiDAR points.

Pseudo-LiDAR++ [40] is an extension of Pseudo-LiDAR where the scene depth is updated by correcting the predicted depth values with a low-cost yet accurate sparse (4 beams) LiDAR. It proposes a graph-based depth correction (GDC) algorithm that effectively combines the stereo depth and the sparse LiDAR measurements to get a corrected dense depth map. The GDC uses a weighted K-nearest-neighbor (KNN) approach to connect and correct each 3D point with the value of its nearest neighbors. To construct the KNN graph, the weight matrix is chosen as coefficients that reconstruct the depth of any point from the corrected depth to its neighbors. For the points which have corresponding LiDAR measurements, the depth values are updated to the ground truth values. The remaining depth points are updated based on the learned weight matrix and the ground truth depth points. Eventually, all the depth points are corrected as the depth correction propagates across the entire graph.

2.3 3D Object Detection Methods

Accurate and robust object detection in 3D plays an indispensable role in robotics and computer vision. Recently, due to the emergence of CNNs and large annotated image datasets, object detection research has progressively moved towards feature learning approaches thereby producing robust representation of the objects. The 3DOD methods could be broadly divided into three categories namely monocular image, 3D point cloud and fusion based methods.

Since, monocular image methods do not have any depth information, most approaches first detect 2D candidates and then predict a 3D bounding box over the object using geometric constraints [2] or CNNs [1]. Mono3D [1] introduces a region proposal algorithm based on semantics, context, shape features and location priors. These features are computed and scored for any given proposal using an energy model. The proposals are generated using an exhaustive search in the 3D space and further filtration using non-

maxima suppression techniques. Mono3D is an extension to the authors previous work 3DOP [34] which uses depth images to generate proposals in a similar framework. Deep MANTA [4] in their work use a many task network to predict vehicle position, localization and shape using only monocular images. The vehicle shape consists of a set of keypoints which characterize the 3D dimensional vertices of the vehicle. Initially, a 2D bounding box is regressed using a two-level region proposal network. Using the inferred 2D shape, the 3D object pose is predicted using 3D model matching techniques. Most monocular methods rely on first learning region proposals and then 3D model matching and reprojection in the second stage for 3DOD.

3D sensors such as LiDAR are naturally represented as 3D point clouds. Many 3D point cloud methods first project the 3D points into 2D images using plane [6], spherical [7] or BEV [8], [9], [10] projections before feeding them to a standard 2D object detection model. The availability of good datasets and state of the art architectures for 2D images have made the projection-based methods more popular. Li et. al. [35] uses a cylindrical projection of the 3D point clouds along with CNNs to predict 3D bounding boxes. The projection image has channels encoding the points height and distance from the 3D sensor. This image is fed as an input to a fully convolutional network which first down-samples and then uses transposed convolutional layers to up-sample the map to predict bounding boxes. Since there are many bounding box predictions, a non-max suppression is used to filter the overlapping predictions based on score and distance.

BirdNet [8] uses a BEV image projection to generate the 3D proposals. It encodes the BEV image as height, intensity and density channels. They also propose a novel encoding that normalizes the density channel based on the type of LiDAR being used. The density normalization creates a uniform representation making the detection model generalize to LiDAR sensors with different specifications and number of beams. ComplexYOLO [10] also uses a BEV projection with three channels encoding the minimum, median and maximum height values of the points associated with a pixel. It introduces real time

operation capability using a YOLO [36] based architecture with extensions to predict the extra dimension and orientation angle. Its architecture is categorized as a single shot detector allowing ComplexYOLO to achieve a runtime of 50 fps, five times more efficient than comparable methods. BirdGAN [41] uses a GAN based approach towards generating a sparse BEV map given a monocular image as input. The method uses a mapping mechanism between an RGB image and its corresponding BEV projection to train a GAN based network. The network further uses existing 3DOD methods which utilize BEV projection for its bounding box estimation. BirdGAN requires only a single RGB image as input at test time and achieves high performance for nearby objects. But, one major disadvantage of their method is that they restrict the frontal distance by clipping the generated sparse BEV map to 25 meters, severely limiting the 3D detection to only objects which are close.

The 3D point clouds consist of a variable number of sparse 3D points distributed over the space. Hence, it is not possible to incorporate their structure to standard feed forward CNNs which assume fixed input sizes. PointNet [37] introduced a method to handle these irregularities by using raw points as inputs to their network. The network takes segmented 3D point clouds as input to perform object classification and part-segmentation. It performs pointwise transformation and feature extraction using fully connected and max pooling layers. PointNet++ [38] is an extension to PointNet and outperforms it by encoding more complex features at every layer of its model architecture.

MV3D [13] introduces different fusion strategies of camera and LiDAR modalities in order to improve the 3DOD performance. They propose three types of fusion strategies namely early fusion, late fusion and deep fusion. In early fusion, the modalities are combined initially creating a new representation which is dependent on all the participating modalities. The late fusion scheme processes the modalities separately and independently until the last process, where fusion occurs. In deep fusion scheme, the modalities are mixed hierarchically in CNN layers such that their features interact over the layers to produce

more general fusion scheme. MV3D combines the RGB image of the camera along with the BEV and front view projection of the LiDAR points for its prediction. The authors conclude that the deep fusion scheme obtains the best performance due to hierarchical feature aggregation from different modalities. AVOD [14] uses a similar input representation as MV3D except that only BEV projection and camera RGB image are used. They propose an early fusion scheme to merge the feature maps obtained from both the modalities to achieve a high proposal recall from their region proposal network.

Wang et. al. [39] introduced the concept of Pseudo-LiDAR, where initially the depth of a monocular image is estimated and later is converted into 3D dense point clouds mimicking the original LiDAR point cloud representation. The Pseudo-LiDAR representation is used instead with any 3D point cloud-based method to achieve comparable results. Some methods [19] often treat image depth as a fourth channel, Pseudo-LiDAR uses it to lift 2D image pixel to its 3D coordinates. Pseudo-LiDAR++ [40] is an extension to Pseudo-LiDAR in which the generated pseudo-LiDAR is corrected using sparse but accurate measurements from a low-cost LiDAR.

Chapter 3

Methodology

3.1 Proposed Method Overview

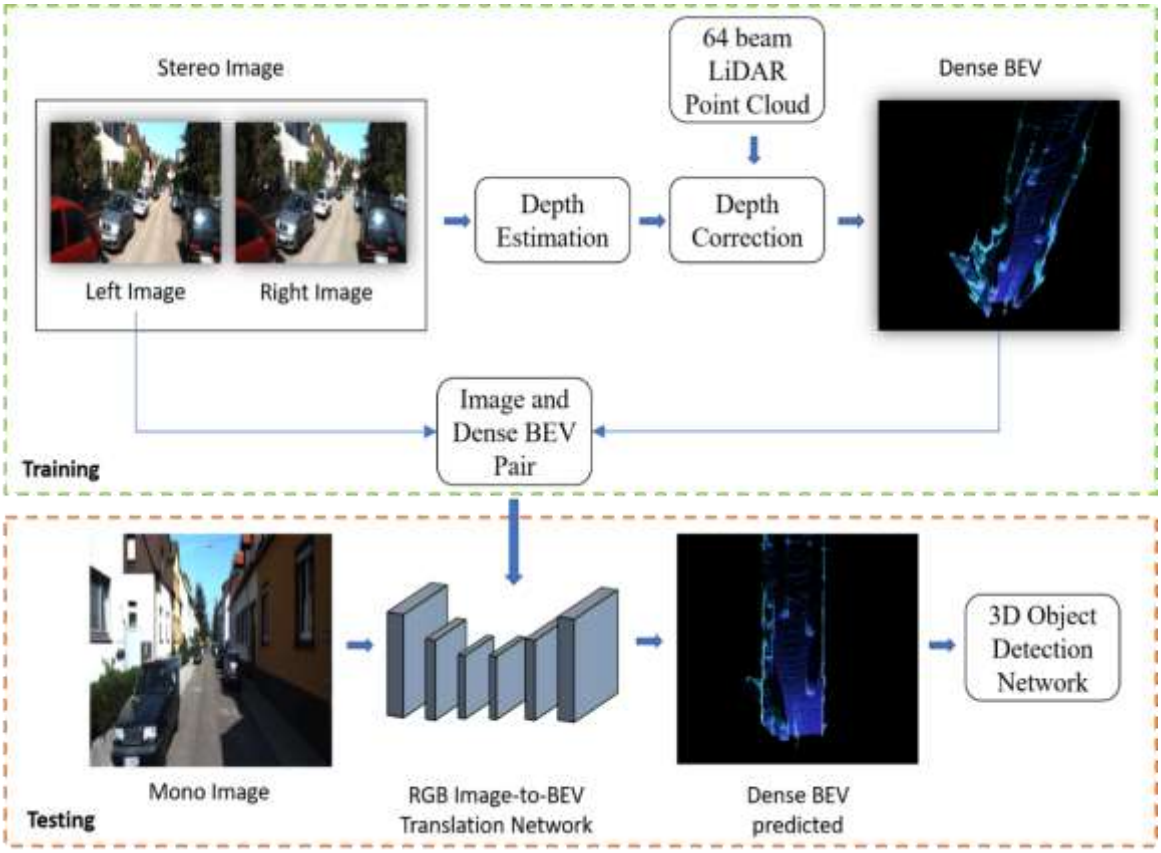

Figure 3.1: Overview of the proposed method. The training process of the network is enclosed in the green dash line box. The testing process is enclosed by the orange dash line box.

The proposed method could be divided into training and testing phases. For the training, a depth estimation network is used to predict the depth of a given pair of stereo RGB images. The predicted depth map is then corrected using a depth correction algorithm. The depth correction algorithm takes the predicted depth maps and with the help of the ground truth depth values obtained from a sparse 64-beam LiDAR corrects them. The corrected dense depth map is then lifted to its 3D point cloud representation using the pseudo-LiDAR method. This corrected dense 3D point cloud is then projected to a dense BEV image. The input RGB image and its corresponding dense BEV image consists of the image pair used to train the Image-to-BEV translation network.

During testing, the Image-to-BEV translation network takes a single image as input and predicts its corresponding dense BEV projection. The predicted dense BEV image is then processed by a 3D object detection network which regresses the 3D bounding box parameters. The 3D object detection network consists of a 2D object detector architecture with additional object dimension and position regression. Figure 3.1 represents the overview of our proposed method.

3.2 3D Point Cloud to BEV Image Projection

The BEV image projection is obtained from a 3D point cloud representation and encodes three channels namely height, intensity and density. The LiDAR sensor scans the region of interest using laser beams and collects the 3D coordinates (x, y, z) and reflectivity (r) values for every laser pulse. The collection of the 3D coordinates and reflectivity values are stored in the form of 3D point clouds. The 3D point cloud representation is projected to a 2D BEV image of size N x N x 3. The 3D point clouds covering an area of $x \, \varepsilon \, [0, 50\text{m}]$, $y \, \varepsilon \, [-25\text{m}, 25\text{m}]$ and $z \, \varepsilon \, [-2\text{m}, 1.23\text{m}]$ in front of the LiDAR sensor are discretized and converted into 2D BEV images with a resolution $\frac{50}{\text{N}}$ meters per pixel. The z value range is chosen considering the LiDAR z position to be 1.73m [44] above the ground level such

that it can cover a maximum height of 3m above the ground. The three channels of the BEV image encode the height, intensity and density values [8].

The height channel represents the maximum height value of the point clouds associated by the pixel, and is limited to 3 meters above the ground. The intensity channel encodes the mean reflectance intensity value of the points associated by the pixel. The last which is the density channel encodes the density of all the points associated by a pixel. It is computed by dividing the number of points present in a pixel with the maximum possible number of points. More the number of points associated by a pixel, higher is the density value of that particular pixel. Figure 3.2 shows an RGB image and its corresponding BEV image of size 256 x 256 encoding the height, intensity and density channels. The BEV image is discretized and projected from the 64-beam LiDAR sparse 3D point clouds by considering an area of 50m x 50m in front with a resolution of about 0.19 meters per pixel and a height of 3 meters above the ground.

The task of translating the street view RGB image to its corresponding BEV projection is achieved using a GAN based architecture. The GAN based architectures are a popular choice in many image-to-image translation applications. The performance of a GAN is highly dependent on the quality of the training data. Hence, the BEV images used for training the GAN are first pre-processed, to remove the noise and include the relevant information, prior to training.

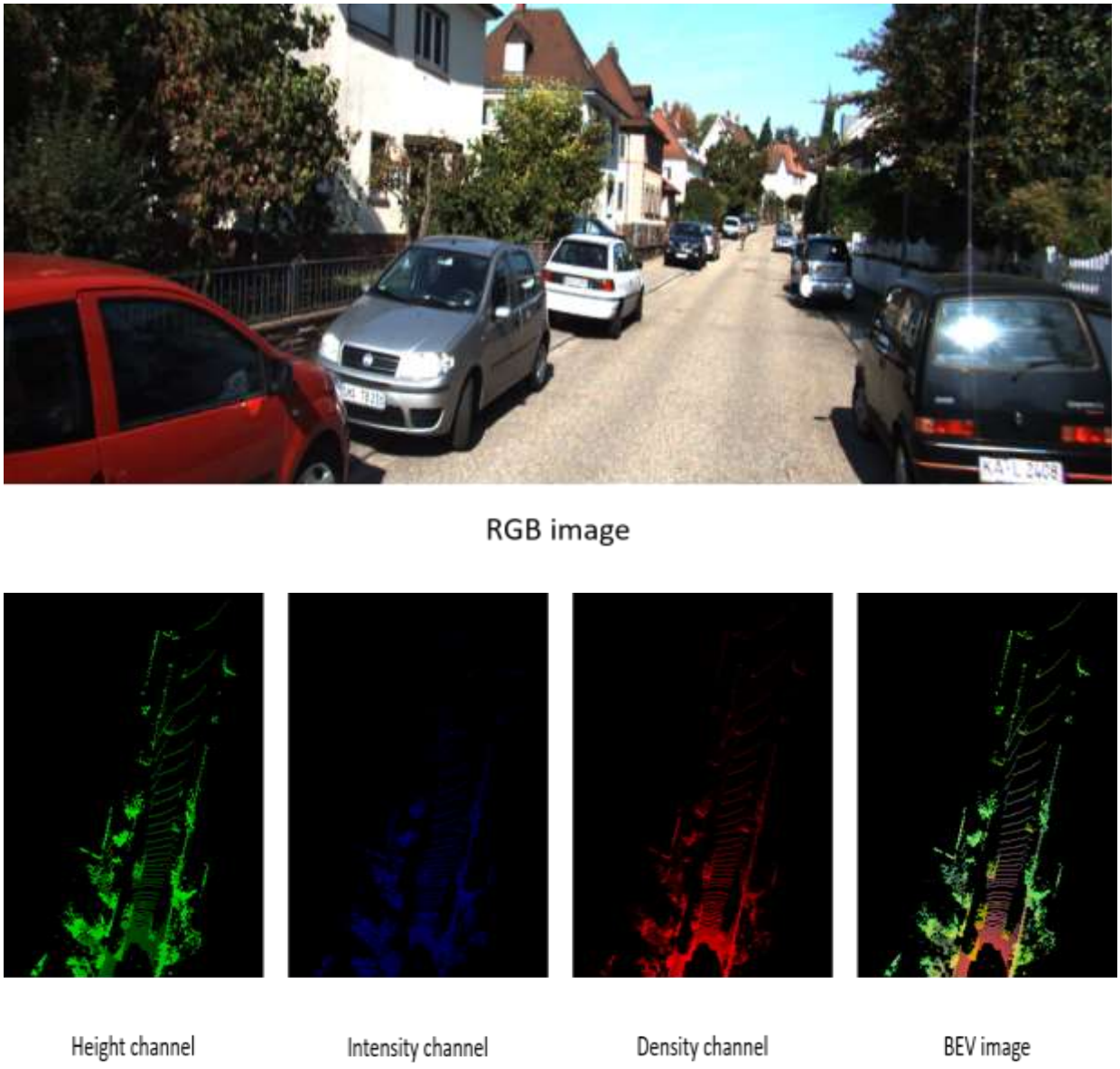

Figure 3.2: Street view RGB image (top) and corresponding BEV image features (below)

3.3 Dense BEV Image Generation

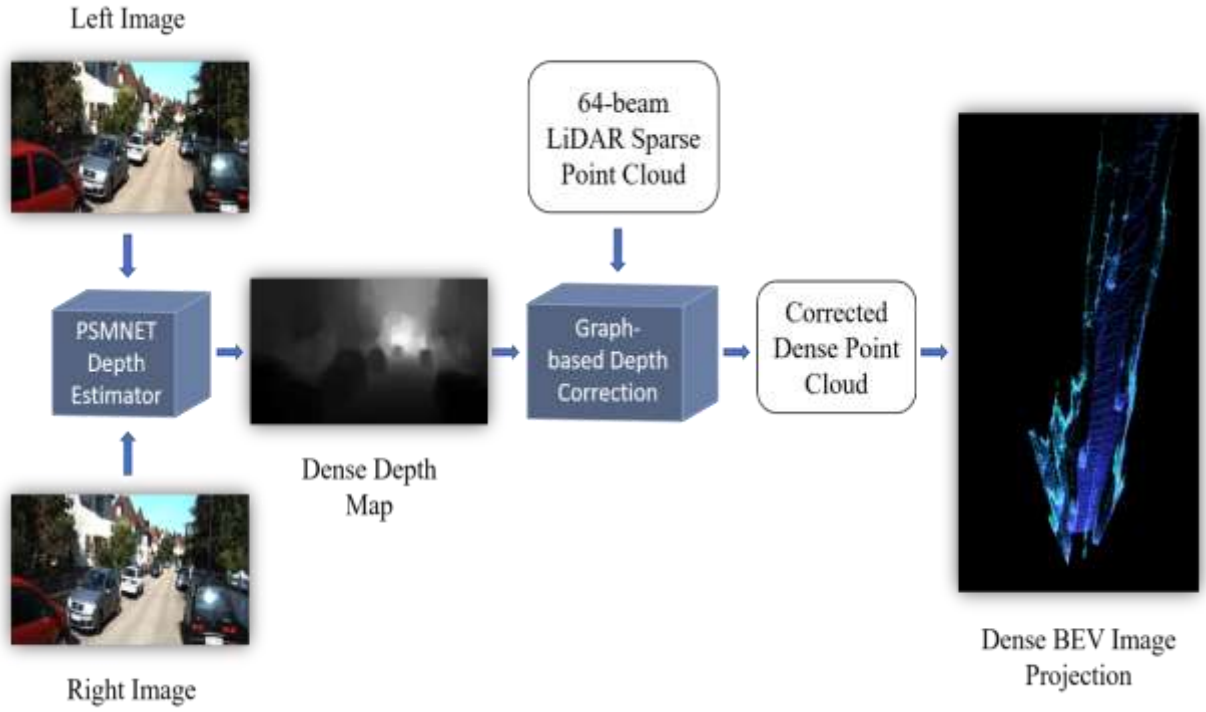

Figure 3.3: Workflow of the generation of dense BEV images prior to training. The newly generated dense BEV images are used as ground truth for training the RGB image-to-BEV translation network.

A 64-beam LiDAR produces sparsely distributed 3D point clouds. The generated points have accurate location coordinates, however, due to the sparse distribution some important details present in between two beams are lost. The BEV projection of a sparse 3D point cloud generates a sparse image with minimum information density in the object regions. On the other hand, by projecting a dense depth map of the image to a pseudo-LiDAR 3D point cloud representation, a dense BEV image is obtained. However, the pseudo-LiDAR point cloud accuracy highly depends on the accuracy of the depth prediction network.

For the task of generating ground truth BEV images, a stereo-pair of RGB images are first passed through a stereo depth estimation network PSMNet [43]. The PSMNet stands for pyramid stereo matching and consists of a Siamese network for stereo disparity estimation followed by 3D convolutions for outlier refinement. The estimated depth map consists of the per pixel depth value of the RGB image. By utilizing the depth values, the 2D image (u, v) is lifted to its 3D coordinates (x, y, z) using the pseudo-LiDAR method. The newly transformed 3D point clouds have dense representation with size equal to the image dimensions. However, the point cloud values are not accurate due to the error introduced by the stereo depth estimation (PSMNet) network.

The predicted dense depth map values are corrected using the graph-based depth correction (GDC) [40] algorithm. The GDC algorithm takes the predicted dense depth map and the sparse (64-beam) LiDAR ground truth depth map as input and uses a weighted KNN approach to correct the predicted dense depth values. In the GDC algorithm, initially every point of the predicted dense depth is connected to its K nearest neighbor using the KNN method. An edge weight matrix is then computed between every depth point and its neighbors by solving a linear matrix equation. The weight matrix is computed such that it could be used to reconstruct the depth of any point from the depth of its K neighbors. The depth points which have their corresponding LiDAR values are replaced by their ground truth depth values. Then by utilizing the pre-computed weight matrix and the ground truth replaced depth points, the entire dense depth map is corrected. By using the pseudo-LiDAR method, the corrected dense depth map is projected to its 3D coordinates to get a corrected dense 3D point cloud representation. A dense BEV image encoding the height, intensity and density channels is projected from the newly corrected dense point clouds. Figure 3.4 denotes a sparse and a dense BEV projection of a sample image from the KITTI dataset [15]. The sparse BEV is projected from a 64-beam LiDAR based sparse 3D point cloud while the dense BEV is obtained from the GDC corrected depth estimated 3D point cloud. The newly generated dense BEV images are used as ground truth for training the RGB

image-to-BEV translation GAN based network.

3.4 Image-to-BEV Translation Network

The Image-to-BEV translation Network is based on the Pix2Pix [20] architecture. Pix2Pix is a GAN model designed for the task of image-to-image translation. The conditional GAN (cGAN) is an extension towards the GAN architecture that allows an image of a particular class to be generated. Our Image-to-BEV translation Network is an implementation of cGAN where the image generation is conditional on a given input image. In this, the generator model takes input an image and generates a translated version of that image. The discriminator model is given as input a pair of an image and its generator generated image or a translated real image and must determine whether the paired image is real or fake. The generator model is trained to fool the discriminator by generating plausible sample images.

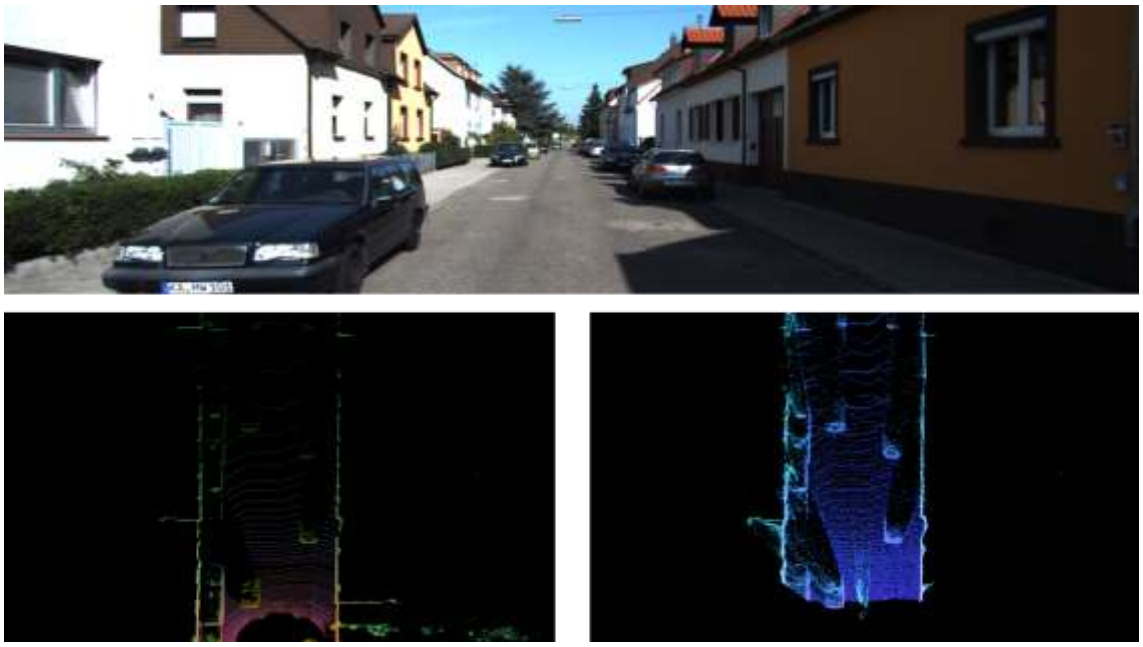

Figure 3.4: Sparse BEV (left) and dense BEV (right) image projection of a sample image (top) from the KITTI dataset.

The generator and discriminator models of the architecture use standard convolutional-batch normalization-ReLU blocks of layers. Unlike a traditional GAN architecture, the generator model does not take a random point from latent space as input. Instead, it takes an image as input with the source of randomness coming from the use of dropout layers applied at several layers of the generator. The generator comprises of a U-net [54] style model architecture which involves an encoder down sampling the input image for a few layers until a bottleneck layer and then a decoder up sampling it to get the desired output size. In order to make the most of any low-level information shared between the input and the output layers, skip connections are added between layers of the same size of the encoder and the decoder. Figure 3.5 depicts the U-net generator model. In this, the first layer of the encoder is merged with the last layer of the decoder and both have the same sized feature maps. This is repeated for every encoder layer and its corresponding decoder layer, forming a U-shaped model.

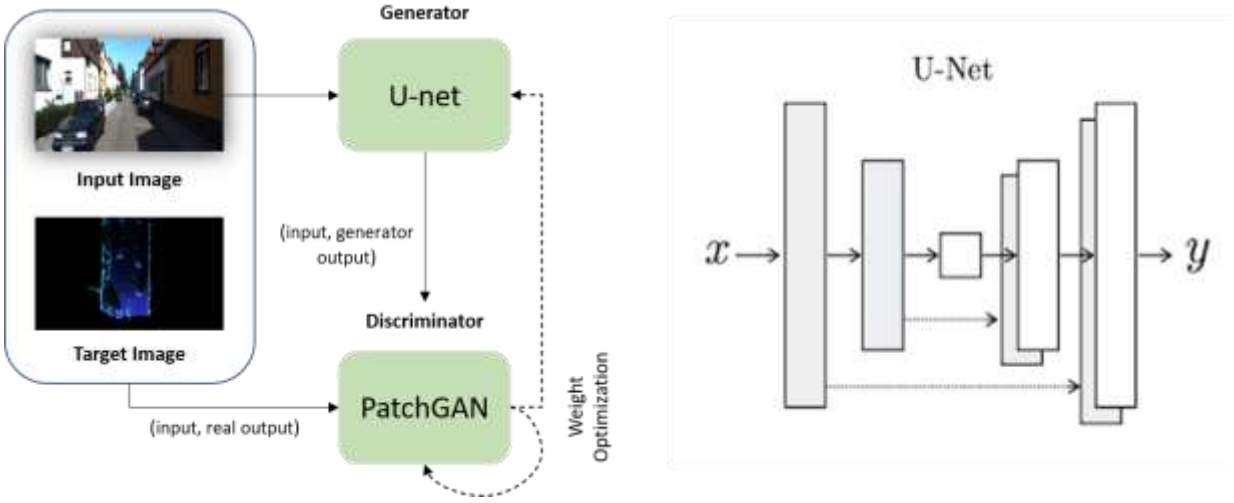

Figure 3.5: Framework of the image-to-BEV translation network (left) and U-net generator model having skip connections (right).

A traditional GAN discriminator model uses a CNN to classify the images, our image-to-BEV translation discriminator uses a PatchGAN model. The PatchGAN is a CNN designed to classify patches of the input image as real or fake instead of the entire image. The output of the PatchGAN model is a single feature map of predictions which can be averaged to get a single score. From Pix2Pix, it is observed that a patch size of 70 x 70 is found to be the most effective for the image-to-image translation. Hence, our translation model uses a patch size of 70 x 70. The PatchGAN discriminator model is trained to minimize the negative likelihood of identifying real and fake images, conditioned on the source image. The discriminator loss is halved as the discriminator training is too fast as compared to the generator. The generator model is trained using both the L1 loss and the adversarial loss. The L1 loss regularizes the generator to obtain reasonable images which are translated from the source image. The adversarial loss controls whether the generated images are plausible in the target domain. Both the L1 loss and the adversarial loss are combined into a composite cost function such as the contribution of the L1 loss to the adversarial loss is controlled by a hyperparameter lambda ($\mathcal{L}$).

$$\text{Generator Loss} = \text{Adversarial Loss} + \mathcal{L} * \text{L1 Loss}$$

The image-to-BEV translation model must be trained on image datasets that constitute the input images and their translated target images. Hence, the RGB images along with their corresponding generated sparse/dense BEV images comprises the dataset for training the image-to-BEV translation network.

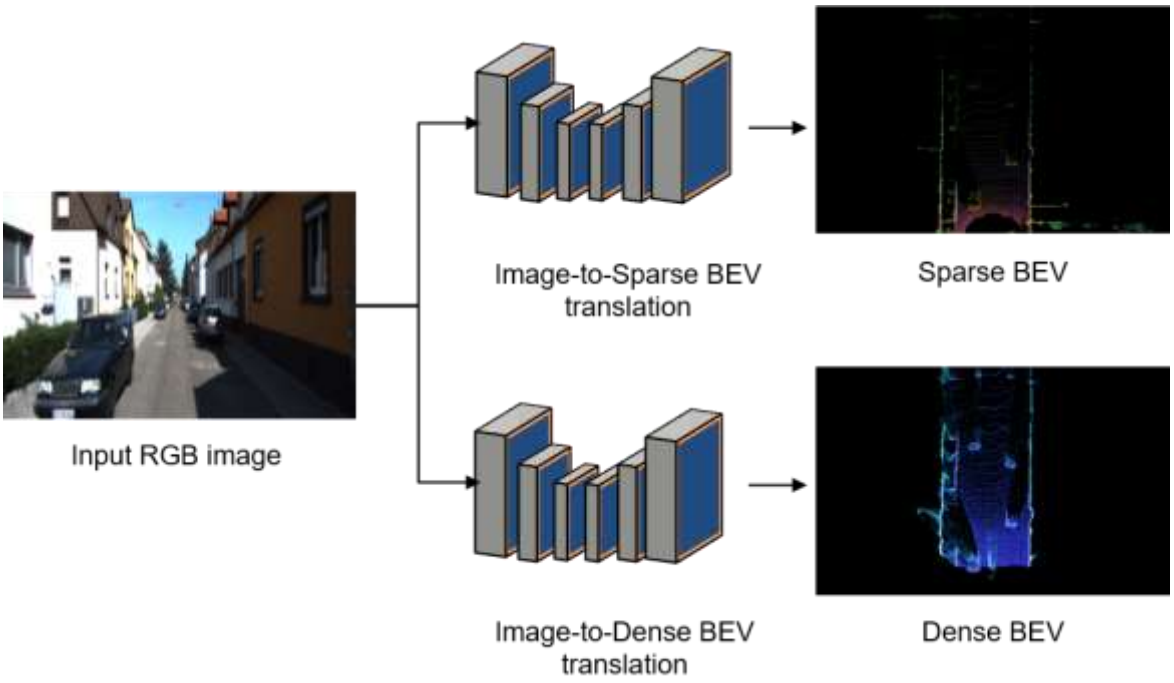

Figure 3.6: The image-to-BEV translation network predicts a sparse or a dense BEV representation of the given input street view RGB image.

3.5 3D Object Detection

We extend the CenterNet [30] architecture with a BEV image as input and use it as our 3DOD framework. The model uses a keypoint based approach and considers the center of the object as the keypoint for connecting all the bounding box features. Figure 3.7 denotes the overall architecture of the 3DOD pipeline. The RGB image-to-BEV translation model takes a single RGB image as input and predicts its BEV representation. The BEV image is further encoded using a variant of ResNet [31] called as ResNet-50 based feature extractor. Similar to CenterNet, the ResNet-50 feature extractor is augmented with three up convolutional layers each having 256, 128, 64 channels respectively. It takes the BEV image of size 256 x 256 as input and down samples it by a stride (S) factor 4 to produce feature maps of size 64 x 64.

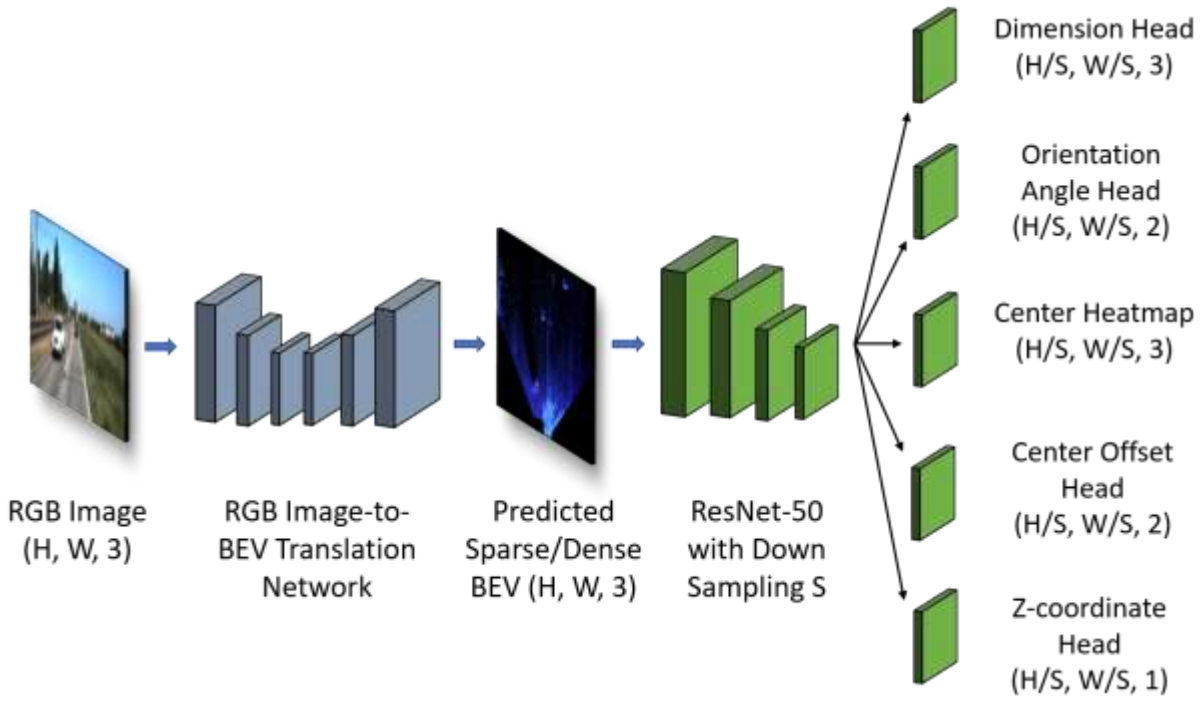

Figure 3.7: 3D object detection framework.

In order to predict the 3D bounding box constrains, for every forward pass through the feature extractor, five heads (keypoints) are regressed. They constitute the object center heatmap, center offset, dimension, orientation and the z-coordinate head. The center heatmap is used for the estimation of the object center for each class and has dimensions 64 x 64 x 3 (for classes = 3). To generate the ground truth center heatmaps for training and loss propagation, the object center is first converted to its low-resolution equivalent by dividing it by a stride 4 and then splat using gaussian kernels. If two gaussian kernels of the same class are overlapping, an element wise maximum is taken to decide the target class. All the values corresponding to the object center and its related 3D box features are first converted from camera to LiDAR coordinates in order to project them on BEV images. For the inference, a 3 x 3 max pooling operation is applied on the predicted center heatmaps

and only the top predictions whose confidence score is greater than 0.2 are kept.

The center heatmap keypoint connects all the feature heads viz. center offset, dimension, orientation and the z-coordinate for an object belonging to a particular class. Due to the down sampling of the input image by stride 4, a discretization error is introduced at the object center coordinates. The center offset head is used to recover the discretization error of the predicted object centers. The keypoint coordinates are very position sensitive in keypoint estimation problems as all the bounding box features are related to it. Hence, the center offset head is added to adjust for any quantization errors and get more accurate results. The dimension of the center offset head is 64 x 64 x 2.

The 3D bounding box dimensions such as the height, width and the length are regressed from the dimension head. The size of the dimension head is 64 x 64 x 3. The orientation head constitutes the sine and cosine value prediction of the yaw angle displacement of the object and has dimensions 64 x 64 x 2. The depth of the object or the z-coordinate of the center of the object is regressed using the z-coordinate head. The size of which is given as 64 x 64 x 1. The object center heatmap uses a heatmap variant focal loss while the bounding box heads are regressed using a standard smooth L1 loss for the stabilized training of the network.

3.5.1 Heatmap Variant Focal Loss

The center keypoint heatmap is trained and propagated through the network using a variant of the focal loss [45] called as the heatmap variant focal loss. Focal loss helps to address the issue of the class imbalance problem. This issue is observed when information related to one class in the dataset overrepresents the other classes. The single stage object detectors produce a large amount of candidate targets out of which very few belong to the positive samples with rest being background targets. This introduces an imbalance during training with the loss of the negative samples overpowering that of the positive samples. The focal

loss addresses this issue and is given by the equation (5).

$$\mathrm{FL}(\mathrm{p}) = \begin{cases} -\alpha(1-\mathrm{p})^{\gamma}\log(\mathrm{p})\,, \text{if } \mathrm{y}=1 \\ -(1-\alpha)\mathrm{p}^{\gamma}\log(1-\mathrm{p})\,, \quad \text{otherwise} \end{cases} \tag{5}$$

where p is the class probability of the anchor, α and γ are the hyperparameters with $\mathrm{y}=1$ representing a positive sample and $\mathrm{y}=0$ representing a negative sample. The effect of loss due to the negative samples as compared to the positive samples is balanced by adjusting the terms $(1-\alpha)$ and α. Greater the value of γ, more is the importance given to misclassified samples and hence less loss will be propagated by the well-classified samples.

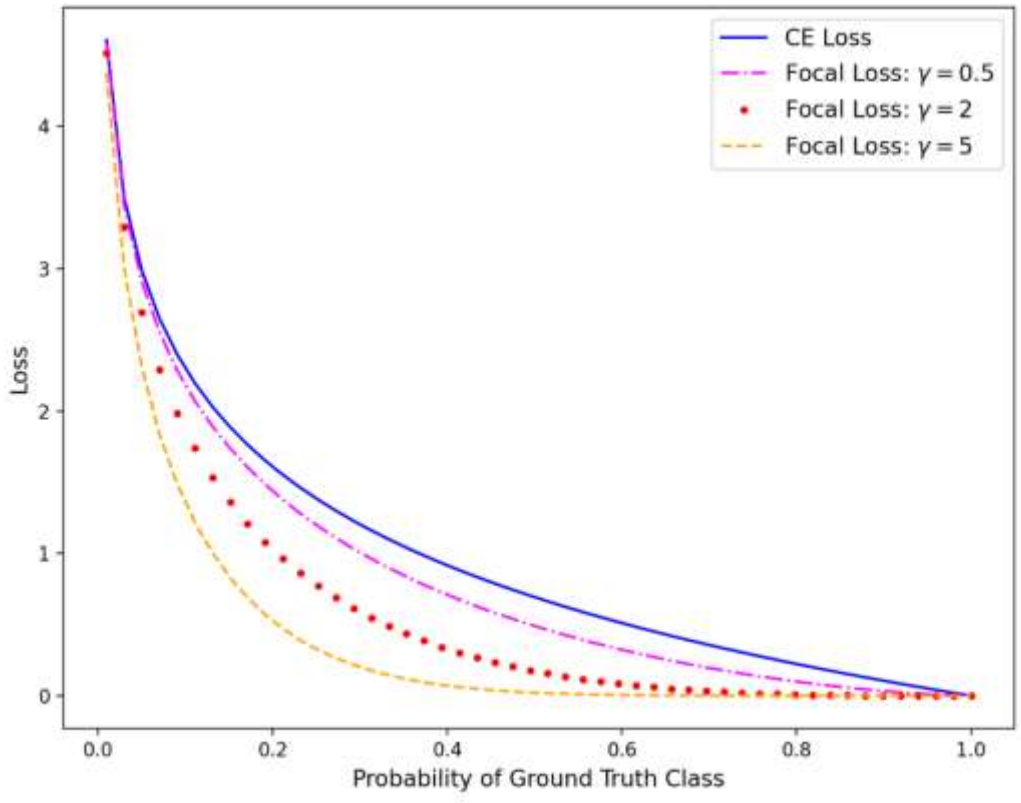

Figure 3.8: Effect of different γ value on the focal and the cross-entropy (CE) loss [47]

The heatmap variant focal loss helps to increase the number of positive samples and decrease the imbalance by considering the center heatmap values generated by the gaussian kernels. It is represented by the equation (6).

$$\mathrm{Lk} = \frac{-1}{\mathrm{N}} \sum_{\mathrm{xyc}} \begin{cases} (1-\widehat{\mathrm{Y}}_{\mathrm{xyc}})^{\alpha} \log\left(\widehat{\mathrm{Y}}_{\mathrm{xyc}}\right), & \text{if } \mathrm{Y}_{\mathrm{xyc}} = 1 \\ \left(1-\mathrm{Y}_{\mathrm{xyc}}\right)^{\beta} (\widehat{\mathrm{Y}}_{\mathrm{xyc}})^{\alpha} \log\left(1-\widehat{\mathrm{Y}}_{\mathrm{xyc}}\right), & \text{otherwise} \end{cases} \quad (6)$$

where α and β are the hyperparameters and are set to values 2 and 4 respectively. When $\mathrm{Y}_{\mathrm{xyc}} = 1$ and the predicted $\widehat{\mathrm{Y}}_{\mathrm{xyc}}$ is close to 1, it considers a well classified sample and by the logic of focal loss the propagated loss is decreased. The same logic follows for misclassified samples when $\mathrm{Y}_{\mathrm{xyc}} = 1$ and the propagated loss slope is increased by value α. When $\mathrm{Y}_{\mathrm{xyc}}! = 1$ and $\widehat{\mathrm{Y}}_{\mathrm{xyc}}$ is close to 0, then $(\widehat{\mathrm{Y}}_{\mathrm{xyc}})^{\alpha}$ will make the overall loss towards 0 and less loss will be propagated. The center heatmap ground truths are gaussian kernels and hence there is no sudden drop in values close to $\mathrm{Y}_{\mathrm{xyc}} = 1$. The values inside the gaussian outputs are considered candidate positives. For example, if $\widehat{\mathrm{Y}}_{\mathrm{xyc}}$ is not near to 0 and has a value close to 1 but in the vicinity of the peak of the ground truth heatmap, there will be less loss propagated, even in this condition of misclassification, due to term $\left(1-\mathrm{Y}_{\mathrm{xyc}}\right)^{\beta}$. This is because $\mathrm{Y}_{\mathrm{xyc}}$ will be close to 1 in the region near the center peaks of the heatmap.

3.5.2 Smooth L1 Loss

The 3D bounding box features such as the box dimension, orientation and the z-coordinate value is regressed using a smooth L1 loss. It can be observed form Figure 3.9 that the smooth L1 loss grows linearly with error rather than squarely as observed in L1 loss.

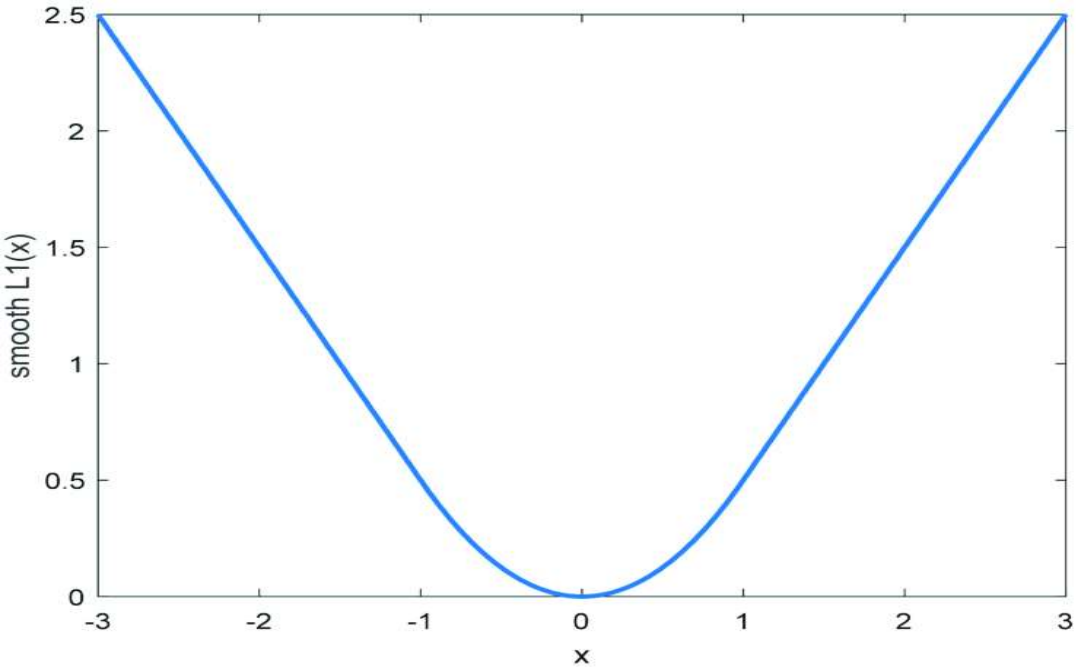

Figure 3.9: Smooth L1 loss [46]

Smooth L1 loss combines the advantages of both the L1 and the L2 loss with steady gradients for large error values and less oscillations to updates when error value is small. The smooth L1 loss is less sensitive to outliers than L2 loss. It is given by the equation in (7).

$$L_{1;\text{smooth}} = \begin{cases} |x| \text{ if } |x| > \alpha \\ \frac{1}{\alpha} x^2 \text{ if } |x| \leq \alpha \end{cases} \tag{7}$$

where α is a hyperparameter and is set to a value 1. The smooth L1 loss behaves as L2 when the absolute value of the error is close to zero and like L1 when it is very high.

Chapter 4

Implementation

4.1 Datasets

4.1.1 KITTI Dataset

The KITTI dataset [15] contains a suite of vision tasks such as 3D object detection, stereo, optical flow, visual odometry, etc. built using an autonomous driving platform. It consists of 6 hours of real-world road traffic scenarios recorded using different sensors. The KITTI 3D object detection benchmark dataset has about 200k 3D object annotations for image data captured from four HD cameras as well as their corresponding point cloud data from a Velodyne 64-beam LiDAR. All the cameras, sensors and localization systems are calibrated and synchronized. Each image has a maximum of 15 cars and 30 pedestrians.

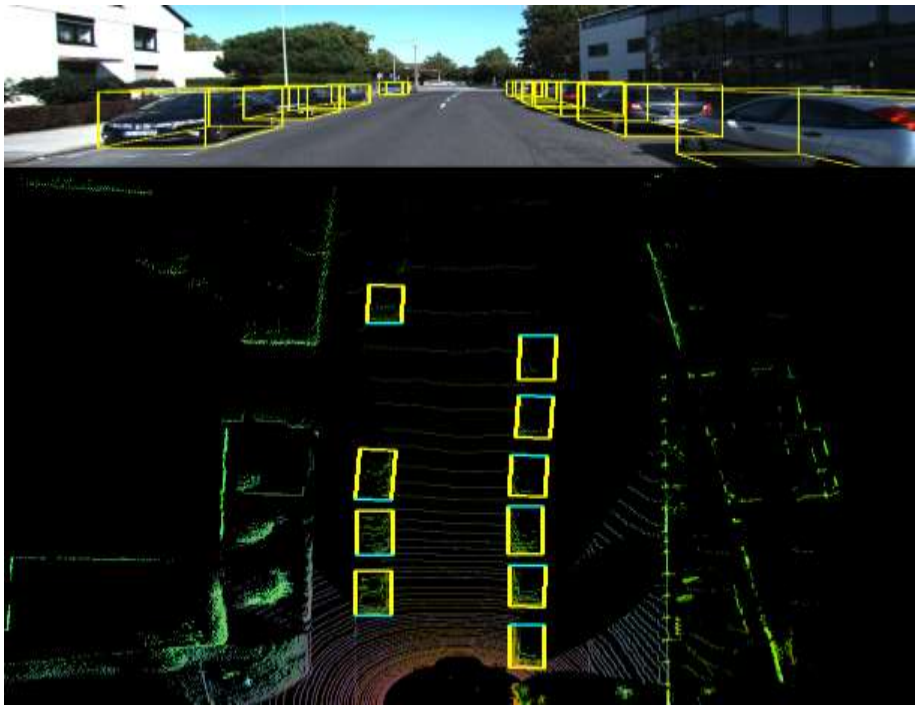

Figure 4.1: Sample image and 3D bounding boxes of the KITTI dataset

For the 3D object detection, the data is split in 7481 training samples and 7581 testing samples. However, there are no ground truth labels available for the test split. Hence, in this the train test split followed by BirdNet [8] and other similar works is used. The training set is further split into a new training and validation set of 3712 and 3769 samples respectively. The evaluation is done on the validation set. Each annotation label is marked as easy, moderate and hard depending on the object occlusion, bounding box height and truncation. The annotations are provided in camera reference frame. Every object in the image is associated with a class and has 2D, 3D bounding box coordinates and yaw angle of orientation. Mainly there are 3 dominant object classes namely car, pedestrian and cyclists. Figure 4.1 shows a sample image from the dataset with 3D bounding boxes.

4.1.2 Scene Flow Dataset

The Scene Flow dataset [48] is a synthetic dataset suite that consists of more than 39k stereo frames rendered from various sequences. It consists of three subsets namely FlyingThings3D, Monkaa and Driving. The Driving data includes street scenes from the viewpoint of the driving car. It consists of synthetically rendered car models, tree models and street lights representing a street view. Each of the subsets has data including RGB stereo images, object level segmentation, optical flow maps, disparity maps, disparity change maps along with the extrinsic and intrinsic camera data for each view of the stereo frame.

Figure 4.2: Sample stereo images from Scene Flow driving dataset [18]

4.2 Evaluation Metric

Intersection of Union (IoU) is a measure of the overlap between two boundaries of an object. It is used to calculate how much the predicted bounding box overlaps with the ground truth box. A predicted box is considered a true positive (TP) if its IoU exceeds a predefined threshold for a particular class category. Similarly, if the IoU is less than the threshold, the predicted box is considered a false positive (FP). A false negative (FN) occurs when the detector fails to predict an object that is present in the view. Figure 4.3 visualizes the IoU calculation for the object detection.

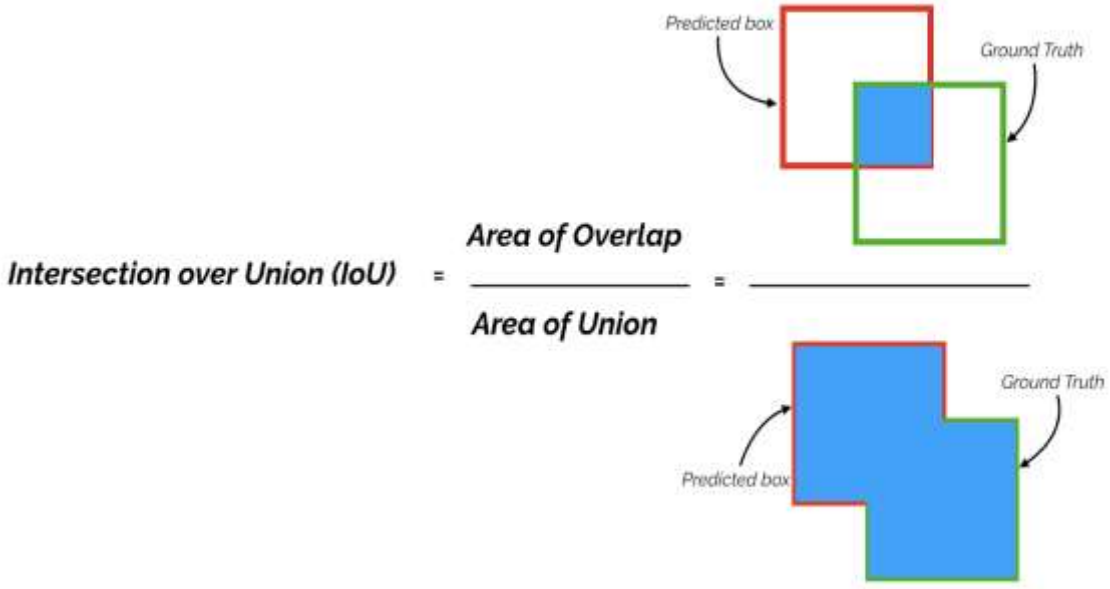

Figure 4.3: Visualization of the Intersection over Union (IoU) [49]

Average precision (AP) is a popular metric used to evaluate the accuracy of object detectors. Precision measures how well the network is able to predict accurately. It is given by the ratio of TP and the total number of predicted positives.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{8}$$

Similarly, recall or sensitivity measures how efficient the network is to predict all the true positives. It is given by the ratio of the TP and the total ground truth positives.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{9}$$

The AP is the weighted sum of precisions at each threshold where the weight is the increase in recall. When the network has a low precision but a high recall value, then it classifies most of the positive samples correctly but has a lot of false positives. On the other hand, if the network has a high precision but low recall, it accurately classifies a sample as positive but may classify only a few positive samples. A precision-recall curve shows a trade-off between precision and recall values for different thresholds. The AP is a way of finding the area under the precision-recall curve. The AP is calculated according to the equation (10).

$$\text{AP} = \sum_{\text{i}=0}^{\text{i}=\text{t}-1} [\text{Recall(i)} - \text{Recall(i} + 1)] * \text{Precision(i)} \tag{10}$$

where t is the number of thresholds. In most cases the AP is calculated for each class and then averaged to get the mean average precision (mAP). In this work, the AP is calculated for the easy, moderate and hard object labels and then averaged to get the mAP values.

4.3 Implementation

4.3.1 RGB Image-to-BEV Translation

The RGB image-to-BEV translation GAN based network is trained end to end using the paired RGB image and its corresponding dense and sparse BEV representations. The dense BEV images are generated from the KITTI [15] 3D object detection training set consisting of 3712 samples. Initially, a PSMNet [43] disparity estimation model which is pretrained on the Scene Flow dataset and finetuned on 200 training pairs of the KITTI [15] stereo dataset is used to predict dense disparity maps of the training samples. The PSMNet model is trained for 300 epochs with the same strategy as done in Pseudo-LiDAR [39]. The estimated dense depths of the training samples are first converted into dense 3D point cloud representations and then corrected by the GDC [40] algorithm using the 64-beam LiDAR point clouds.

The corrected dense 3D point clouds are then projected to 2D BEV images with the configuration as shown in Table 4.1. The RGB image-to-BEV translation network is trained in a similar strategy as done in the original Pix2Pix [20] paper. A 70 x 70 PatchGAN discriminator model is implemented with a binary cross entropy loss and optimized using the Adam optimizer. The generator model is updated using a weighted sum of the adversarial loss and the L1 loss. The weight hyperparameter lambda ($\mathcal{L}$) is set to 100 in favor of the L1 loss. The network is first trained on the RGB and dense BEV image pairs and then on the RGB and sparse BEV image pairs for 150 epochs with a batch size value set to 1.

Table 4.1: Configuration of point cloud to BEV image projection

	Configuration	Value
Point cloud range (meters)	Xmin	0
	Xmax	50
	Ymin	-25
	Ymax	25
	Zmin	-2
	Zmax	1.23
BEV dimensions	Height	256
	Width	256
BEV resolution (meters)	$\frac{\text{Xmax} - \text{Xmin}}{\text{Height}}$	0.19

4.3.2 3D Object Detection

All the 3DOD training dataset labels are converted to BEV image coordinates using the BEV image dimensions and resolution. The down-sampling layers of the ResNet-50 backbone network are initialized with an ImageNet dataset pre-trained model and the up-sampling layers are randomly initialized. The network uses a heatmap variant focal loss and a standard smooth L1 loss as discussed in section 3.2 for the training. The network is trained on the image-to-BEV predicted sparse/dense BEV images of the 3D object

detection training set. The network is trained for 120 epochs with a learning rate of 0.001 and a batch size of 16. At inference, a 3 x 3 max pooling operation is applied on the predicted object center heatmap and only the top predictions with the confidence scores greater than 0.2 are kept while the rest are discarded. The implementation and training of the network is done using the Pytorch framework.

Chapter 5

Results and Analysis

5.1 Results

The RGB image-to-BEV translation model is trained on the KITTI 3D object detection training split and then evaluated on the validation images. Figure 5.1 and Figure 5.2 presents the qualitative results of the translation model on some images from the validation split when trained on two different types of training data. In the first case, the training data includes the RGB and dense BEV image pairs. For the second, the training consists of the RGB and the sparse BEV projection as ground truth data.

In sparse BEV representations, as the distance of objects in the image increases the BEV image becomes extremely sparse making a GAN based model difficult to train. Hence, for the experimentation, the sparse BEV image data corresponding to points which are more than 20 meters are removed. This modified sparse BEV (20m) images along with their corresponding RGB images are used to train the translation model. Figure 5.1 denotes the results reported when trained on dense BEV images. The dense BEV images are generated by considering objects which are 50m in the front view. Figure 5.2 denotes the results when the model is trained on sparse BEV images (20m).

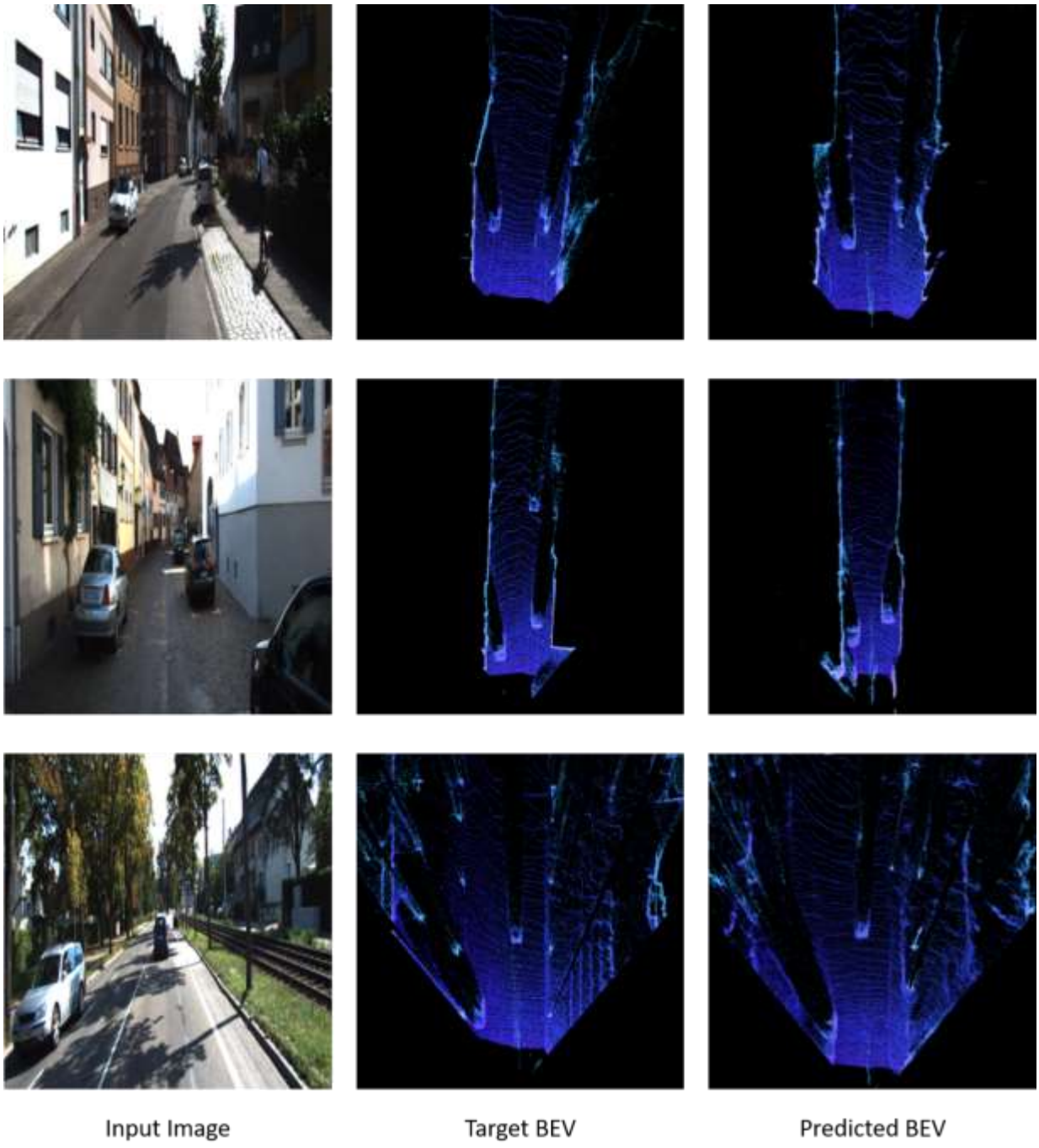

Figure 5.1: Qualitative validation set results of the image-to-BEV translation model when trained on dense BEV (50m) images.

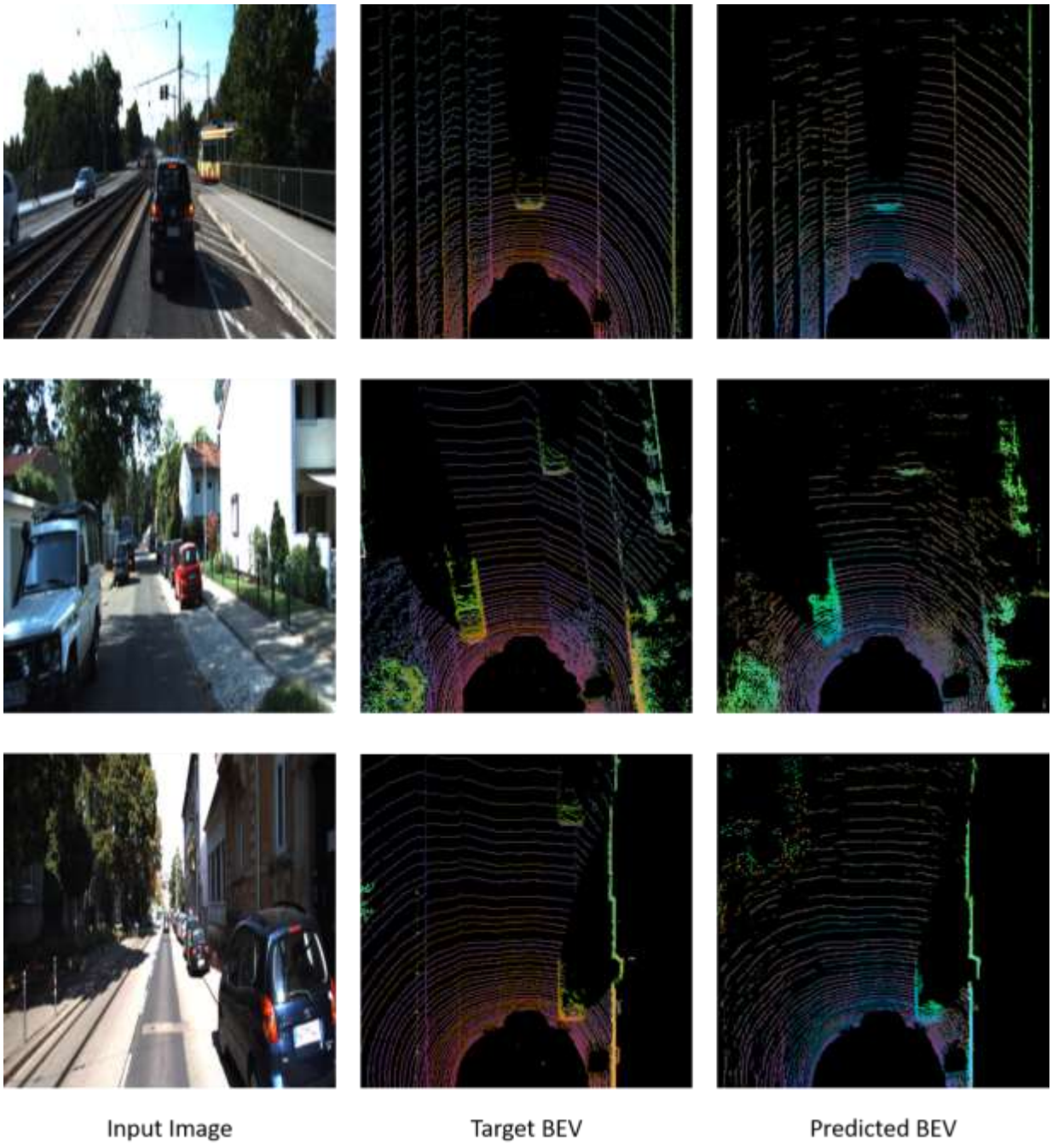

Figure 5.2: Qualitative validation set results of the image-to-BEV translation model when trained on sparse BEV (20m) images. The sparse BEV images are clipped to objects at a distance of 20m in the front view.

Looking at the results, it is evident that the translation model generates better images with object details when trained on dense BEV in comparison to the sparse BEV. This is because the dense BEV representation has a greater number of pixels encoding information/density of the input RGB image as compared to the sparse BEV. This helps the GAN based translation model to generate more plausible dense BEV images given an RGB image. In the case of sparse BEV, the performance of the translation model decreases significantly as the object distance increases. This is due to the fact that as the distance increases, the sparse BEV points become more and more scattered and scanty making it hard for the model to train and generate credible images.

The 3DOD model is trained on both the dense (50m) and the sparse BEV (20m) images for only the 'car' category. The model is evaluated for the task of 3D object detection based on IoU and mAP scores. In this, the AP is first calculated for the easy, moderate and hard object labels and then averaged to get the mAP scores. Table 5.1 shows the result of the 3D object detection task on the dense (50m) and sparse BEV (20m) representations on the KITTI validation set. The sparse BEV accuracy is reported by considering only the objects which are 20m in the front view.

Table 5.1: Comparison of the 3DOD model when trained and evaluated on dense (50m) and sparse BEV (20m) for the 'car' object class. Results are shown on the KITTI 3D object detection validation set.

Method	**Testing Modality**	**mAP (%)**	
		IoU = 0.7	**IoU = 0.5**
Image-to-Sparse BEV (20m)	Mono	3.93	17.73
Image-to-Dense BEV (50m)	Mono	8.34	29.11

It is observed from Table 5.1 that the dense BEV (50m) outperforms the sparse BEV (20m) in terms of accuracy for all the IoU thresholds (0.7, 0.5). When the model is trained on the predicted dense BEV (50m), there is an improvement of more than ~50% over sparse BEV (20m) for IoU = 0.7. Figure 5.3 and Figure 5.4 shows the qualitative results of the 3DOD model when trained and evaluated on the dense BEV (50m) and sparse BEV (20m) images respectively. From Figure 5.4 it can be inferred that although the sparse BEV image is clipped at 20 meters, the accuracy decreases significantly for objects which are more than 15 meters away from the camera.

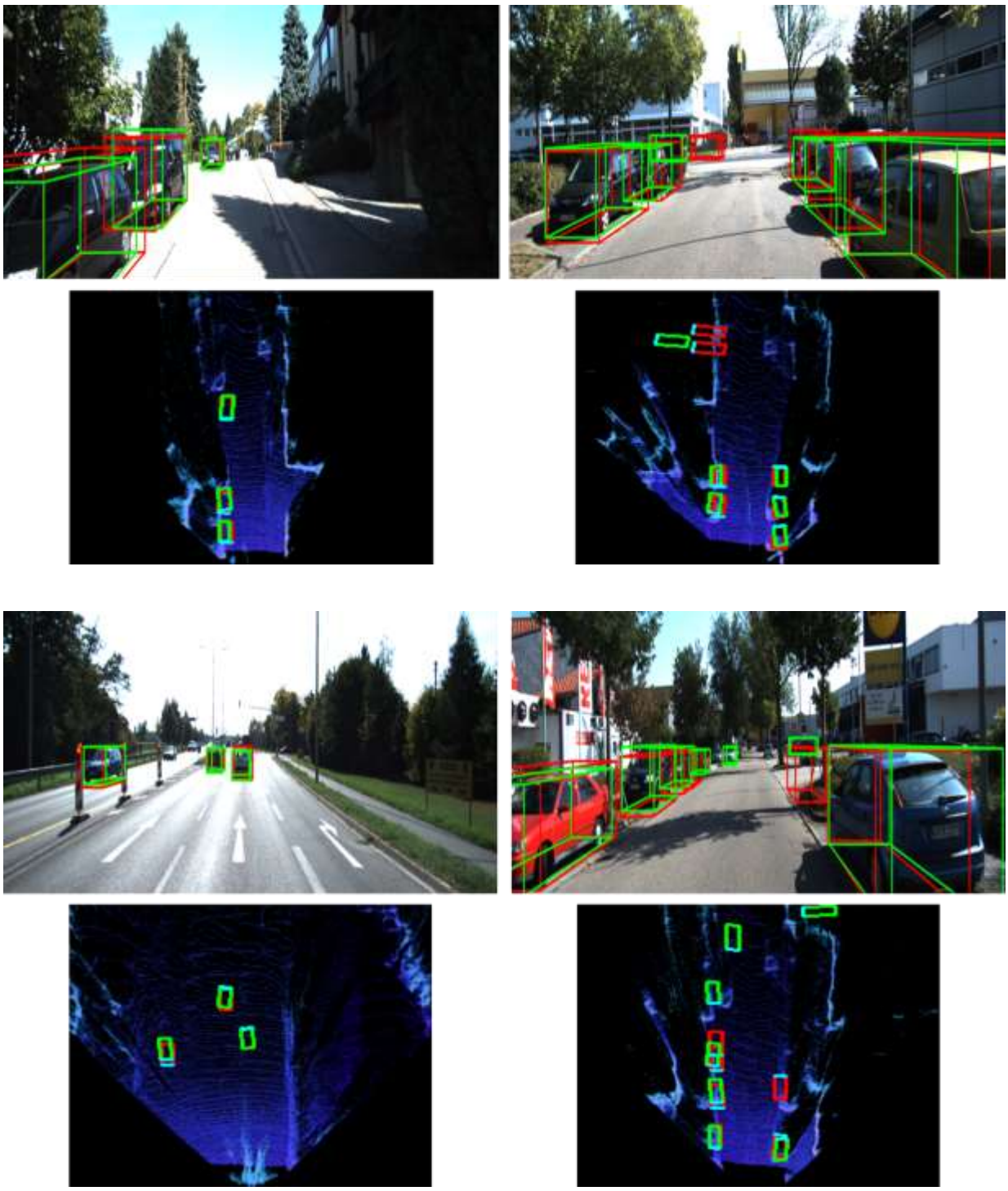

Figure 5.3: Qualitative results of the 3DOD model on the KITTI validation images when trained on the dense BEV (50m) predicted images. On the top are the RGB images and bottom are their BEV representations. Predicted boxes are in red and ground truth boxes are in green.

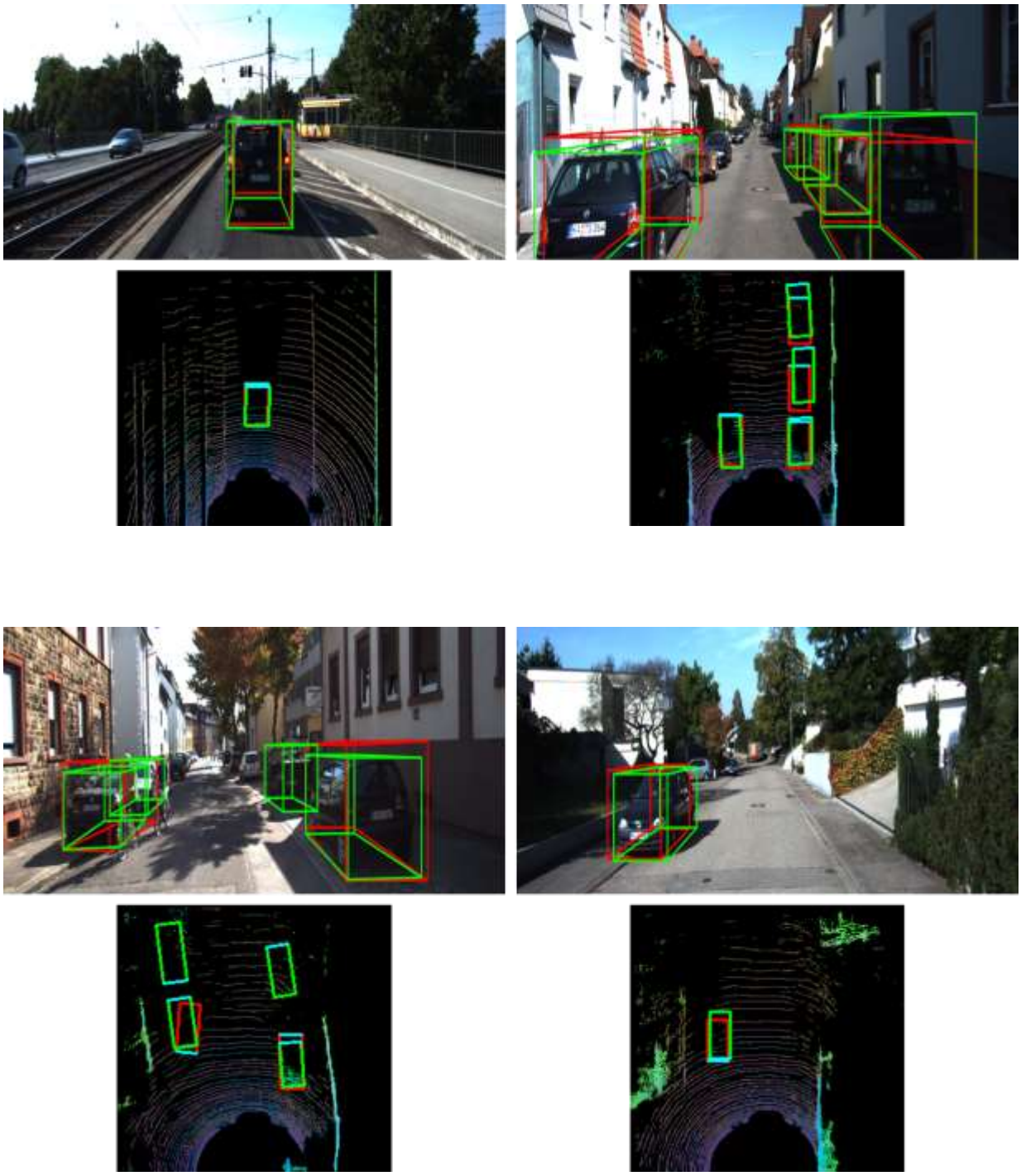

Figure 5.4: Qualitative results of the 3DOD model on the KITTI validation images when trained on the sparse BEV (20m) predicted images. On the top are the RGB images and bottom are their BEV representations. Predicted boxes are in red and ground truth boxes are in green.

Table 5.2 shows a comparison of the predicted dense BEV with other monocular image based 3DOD methods for the 'car' object class. The evaluation is performed for car objects which are 50 meters in front of the camera.

Table 5.2: Comparison of the 3DOD model trained on the predicted dense BEV with other monocular 3DOD methods. Results are shown on the KITTI 3D object detection validation set on the 'car' category. The results are reported by considering objects which are within 50m in front of the camera.

Method	**Testing Modality**	**Time (s)**	**mAP (%)**	
			IoU = 0.7	**IoU = 0.5**
Mono3D [1]	Mono	4.2	2.38	19.63
Deep3DBox [2]	Mono	2.7	4.19	19.67
MonoGRNet [53]	Mono	0.16	9.07	32.37
Image-to-Dense BEV	Mono	0.044	8.34	29.11

From Table 5.2 it is evident that the 3DOD model trained on the predicted dense BEV performs better than the Mono3D [1] and Deep3DBox [2] methods for all the IoU thresholds (0.7, 0.5). Both the Mono3D and Deep3DBox methods first detect 2D boxes and use it as a prior information for 3D object detection. MonoGRNet [53] performs slightly better than the predicted dense BEV based model for both the bounding box IoU thresholds 0.7 and 0.5. This may be because MonoGRNet uses multiple subnetworks which includes depth estimation of the object instances in the monocular image prior to localizing the 3D bounding box. However, the 3DOD model trained on the predicted dense BEV images is fast with less inference time taken per image as compared to other methods. This

is due to the use of a simple CenterNet based anchor free object detector implemented on the predicted dense BEV images.

Figure 5.5 shows the qualitative comparison of the Deep3DBox [2] method and our 3DOD model trained on dense BEV predicted images. The sample images are from the KITTI validation set. From the comparison, it is observed that the 3DOD model trained on the predicted dense BEV images has better bounding boxes aligned with the ground truth boxes than the Deep3DBox method. Also, the Deep3DBox method performs poorly for objects which are partially truncated or occluded. This is observed for cars which are very near to the camera or are partially hidden by another car or an object. The reason would be because the Deep3DBox method first detects a 2D bounding box over the object and then regresses a 3D bounding box from the detected 2D box coordinates.

We also test our method on other similar datasets like Cityscapes [55] to determine its robustness. However, the image-to-dense BEV translation model is not able to generate plausible BEV images of the given RGB front view images especially in the object regions. Further, the 3D object detection model fails to generate 3D bounding boxes from these predicted dense BEV images. This may be because the testing images from the Cityscapes dataset are distinct from the training images of the KITTI dataset. Hence, it could be inferred that the GAN based image-to-dense BEV translation model generates plausible BEV images with object details only when trained on images from the same dataset.

Figure 5.5: Qualitative comparison of our 3DOD model trained on dense BEV predicted images (left) and the Deep3DBox [2] method (right). Predicted boxes are in red and ground truth boxes are in green.

Chapter 6

Conclusions

6.1 Conclusions and Future Work

In this thesis, we introduced a method towards 3D object detection using 3D data representation while only using 2D monocular images at test time. We leveraged a GAN based architecture for mapping the 2D monocular image to its corresponding BEV representation. We demonstrated different training strategies for effective RGB image to BEV translation. We modify a CenterNet based object detection architecture to incorporate a BEV image as input and use it as our 3D object detection model. We compare the effects of different training strategies on the performance of the GAN based network and on our 3D object detection model. The results indicate that by converting the sparse BEV to its dense BEV representation prior to training improves the overall performance of the 3D object detection. Finally, we compared the performance of our 3D object detection model trained on the GAN predicted dense BEV images with other monocular 3D object detection methods. We infer that it is possible to use existing 3D data to train a network such as to generate 3D information at test time using only monocular 2D images and get comparable results.

This thesis research focused on introducing a general architecture for effective image to BEV translation and use it in a BEV based 3D object detection model. This architecture is scalable to be used as a plug-in module with any 3D object detection network which works on BEV images. Hence, the possible direction for future work would be to use the proposed

architecture with different state-of-the-art BEV based 3D object detection methods such as BirdNet [8], BirdNet+ [25] and MV3D [13]. This would further enhance the monocular based 3D object detection accuracy. Also, by increasing the resolution i.e., the point cloud range covered per pixel of the GAN predicted BEV images would further benefit the 3D object detection performance. This would be possible by leveraging a high-resolution generative network such as [52] to generate BEV images with higher dimensions such as 512 x 512.

Bibliography

[1] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler and R. Urtasun, "Monocular 3d object detection for autonomous driving", 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2147-2156, 2016.

[2] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká, "3D bounding box estimation using deep learning and geometry," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 5632–5640.

[3] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3D voxel patterns for object category recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2015, pp. 1903–1911.

[4] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau, "Deep MANTA: A coarse-to-fine many-task network for joint 2D and 3D vehicle analysis from monocular image," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 1827–1836.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[6] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 945–953.

[7] B. Wu, A. Wan, X. Yue, and K. Keutzer. (Oct. 2017). "SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time roadobject segmentation from 3D LiDAR point cloud." [Online]. Available: https://arxiv.org/abs/1710.07368

[8] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. García, and A. de la Escalera. (May 2018). "BirdNet: A 3D object detection framework from LiDAR information." [Online]. Available: http://arxiv.org/abs/1805.01195

[9] S. L. Yu, T. Westfechtel, R. Hamada, K. Ohno, and S. Tadokoro, "Vehicle detection and localization on bird's eye view elevation images using convolutional neural network," in Proc. IEEE Int. Symp. Saf., Secur. Rescue Robot. (SSRR), Oct. 2017, pp. 102–109.

[10] M. Simon, S. Milz, K. Amende, and H. Gross. (Mar. 2018). "ComplexYOLO: Real-time 3D object detection on point clouds." [Online]. Available: https://arxiv.org/abs/1803.06199

[11] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Sep. 2017, pp. 1513–1518.

[12] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2017, pp. 1355–1361.

[13] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 6526–6534.

[14] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2018, pp. 1–8.

[15] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," The International Journal of Robotics Research, vol. 32, no. 11, pp. 1231–1237, 2013. [Online]. Available: https://doi.org/10.1177/0278364913491297.

[16] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," 2019.

[17] 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," CoRR, vol. abs/1903.11027, 2019. [Online]. Available: http://arxiv.org/abs/1903.11027.

[18] M. Oberweger, M. Rad, and V. Lepetit. Making Deep Heatmaps Robust to Partial Occlusions for 3D Object Pose Estimation. ECCV, 2018.

[19] B. Xu and Z. Chen. Multi-Level Fusion based 3D Object Detection from Monocular Images. CVPR, 2018.

[20] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in CVPR, 2017.

[21] Phung, & Rhee, (2019). A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. Applied Sciences. 9. 4500. 10.3390/app9214500.

[22] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In Proceedings of NIPS, pages 2672– 2680, 2014.

[23] Z. Qin, J. Wang, and Y. Lu, "MonoGRNet: A Geometric Reasoning Network for Monocular 3D Object Localization", AAAI, vol. 33, no. 01, pp. 8851-8858, Jul. 2019.

[24] https://medium.com/@alexrachnog/gans-beyond-generation-7-alternative-use-cases725c60ba95e8. GANs beyond generation: 7 alternative uses.

[25] A. Barrera, C. Guindel, J. Beltrán and F. García, "BirdNet+: End-to-end 3D object detection in LiDAR Bird's eye view", *arXiv:2003.04188*, 2020, [online] Available: http://arxiv.org/abs/2003.04188.

[26] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. In ECCV, 2018.

[27] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang and Q. Tian, "CenterNet: Keypoint Triplets for Object Detection," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 6568-6577, doi: 10.1109/ICCV.2019.00667.

[28] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 9627–9636, 2019.

[29] Lichao Huang, Yi Yang, Yafeng Deng, and Yinan Yu. Densebox: Unifying landmark localization with end to end object detection. arXiv preprint arXiv:1509.04874, 2015.

[30] X. Zhou, D. Wang and P. Krähenbühl, "Objects as points" in arXiv:1904.07850, 2019, [online] Available: http://arxiv.org/abs/1904.07850.

[31] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.

[32] F. Yu, D. Wang, and T. Darrell. Deep layer aggregation. arXiv preprint arXiv:1707.06484, 2017

[33] https://medium.com/visionwizard/centernet-objects-as-points-a-comprehensive-guide-2ed9993c48bc CenterNet: Objects as Points - A Comprehensive Guide

[34] X. Chen et al., "3D object proposals for accurate object class detection," in Advances in Neural Information Processing Systems, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. New York, NY, USA: Curran Associates, Inc., 2015, pp. 424–432.

[35] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D Lidar using fully convolutional network," in Proc. Robot., Sci. Syst. XII, AnnArbor, MI, USA, Jun. 2016.

[36] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in Proc. IEEE Conf.

Comput. Vis. Pattern Recognit., Jul. 2017, pp. 6517–6525.

[37] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in Proc. Int. Conf. Comput. Vis. Pattern Recognit., Jun. 2017, pp. 77–85.

[38] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in Advances in Neural Information Processing Systems. New York, NY, USA: Curran Associates, Inc., 2017, pp. 5099–5108.

[39] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Weinberger, "Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving," in CVPR, 2019.

[40] Y. You et al., "Pseudo-LiDAR++: Accurate depth for 3D object detection in autonomous driving", arXiv:1906.06310, 2019, [online] Available: http://arxiv.org/abs/1906.06310.

[41] Siddharth Srivastava, Frederic Jurie, and Gaurav Sharma, "Learning 2d to 3d lifting for object detection in 3d for autonomous vehicles," in IROS, 2019.

[42] Maxim Shevtsov, Alexei Soupikov, and Alexander Kapustin. Highly parallel fast kd-tree construction for interactive ray tracing of dynamic scenes. In Computer Graphics Forum, volume 26, pp. 395–404. Wiley Online Library, 2007.

[43] J.-R. Chang and Y.-S. Chen. Pyramid stereo matching network. In CVPR, 2018

[44] Geiger, A.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). CVPR '12, Washington, DC, USA, IEEE Computer Society (2012) 3354–3361

[45] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. ´ Focal loss for dense object detection. arXiv preprint arXiv:1708.02002, 2017

[46] Chen, Zhong & Zhang, Ting & Ouyang, Chao. (2018). End-to-End Airplane Detection Using Transfer Learning in Remote Sensing Images. Remote Sensing. 10. 139. 10.3390/rs10010139.

[47] https://towardsdatascience.com/a-loss-function-suitable-for-class-imbalanced-data-focal-loss-af1702d75d75 A Loss Function Suitable for Class Imbalanced Data: "Focal Loss"

[48] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene

flow estimation. In CVPR, 2016.

[49] https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2 mAP (mean Average Precision) might confuse you!

[50] B. Xu and Z. Chen, "Multi-level fusion based 3d object detection from monocular images", The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[51] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[52] Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: Highresolution image synthesis and semantic manipulation with conditional gans. In: CVPR. (2018)

[53] Zengyi Qin, Jinglu Wang, and Yan Lu. Monogrnet: A geometric reasoning network for monocular 3d object localization. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 8851–8858, 2019.

[54] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In International Conference on Medical image computing and computer-assisted intervention, pp. 234-241. Springer, Cham, 2015.

[55] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 3213–3223.