Stereo End-to-End 3D Object Detection

by

Jakob Gürtl, BSc

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur
Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisors:

Prof. Dr. Horst Bischof
Institute for Computer Graphics and Vision

Dr. Christian Leistner
Amazon Prime Air

Graz, March 2021

GRAZ UNIVERSITY OF TECHNOLOGY

Abstract

Faculty of Computer Science
Institute for Computer Graphics and Vision

Master's degree programme: Software Engineering and Management

Stereo End-to-End 3D Object Detection

by Jakob Gürtl, BSc

Autonomous vehicles rely heavily on 3D object detection for safely navigating without human intervention. Existing 3D object detection approaches are often based on LiDAR data. While producing more accurate results than methods relying on stereo image data, LiDAR sensors are very expensive. Recently, Pseudo-LiDAR approaches based on stereo image data showed notable results. However, there is still a performance gap compared to LiDAR-based 3D detectors. We investigate improvements for Pseudo-LiDAR to increase 3D object detection accuracy.

We propose to include object masks during training of Pseudo-LiDAR for exact estimation of 3D bounding boxes in border areas of foreground objects. Additionally, we enable both the stereo and point cloud component of the Pseudo-LiDAR framework to interact with each other by training in an end-to-end way. Further, this joint training collectively minimizes a global learning loss.

Our approach is evaluated on a publicly available dataset. The experiments confirm that the integration of object masks and end-to-end training of Pseudo-LiDAR improves the 3D detection accuracy.

Kurzfassung

Autonome Fahrzeuge verlassen sich stark auf 3D-Objekterkennung, um ohne menschliches Eingreifen sicher navigieren zu können. Bestehende 3D-Objekterkennungsansätze basieren häufig auf LiDAR-Eingabedaten. LiDAR-Sensoren liefern zwar genauere Ergebnisse als Methoden die auf Stereobildern basieren, sind jedoch sehr teuer. Kürzlich zeigten Pseudo-LiDAR-Ansätze, die Stereobilddaten als Eingabe verwenden, bemerkenswerte Ergebnisse. Im Vergleich zu LiDAR-basierten Methoden besteht jedoch immer noch eine Leistungslücke. Zu diesem Zweck untersuchen wir in dieser Arbeit Verbesserungen für Pseudo-LiDAR, um die Genauigkeit der 3D-Objekterkennung zu erhöhen.

Wir schlagen vor, Objektmasken während des Trainings von Pseudo-LiDAR für genauere Schätzungen in Randbereichen des Objekts einzuschließen. Darüber hinaus ermöglichen wir beiden Komponenten des Pseudo-LiDAR-Frameworks die Interaktion miteinander, indem wir End-to-End-Training durchführen. Weiters optimiert dieses gemeinsame Training eine globalen Zielfunktion.

Wir evaluieren unseren Ansatz anhand eines öffentlich verfügbaren Datensatzes. Unsere Experimente bestätigen, dass das End-to-End-Training von Pseudo-LiDAR und die Integration von Objektmasken die Genauigkeit der 3D-Objekterkennung verbessern.

Declaration of Authorship

Contents

Acronyms

CNN Convolutional Neural Network

DNN Deep Neural Network

FPN Feature Pyramid Network

IoU Intersection Over Union

LiDAR Light Detection and Ranging

RADAR Radio Detection and Ranging

NMS Non-Maximum Suppression

RPN Region Proposal Network

SGD Stochastic Gradient Descent

Chapter 1

Introduction

Contents

The first chapter describes challenges in 3D object detection. We then discuss motivations and the research objective. Furthermore, we will address our contributions and present the structure of this master thesis.

1.1 3D Object Detection

1.1.1 Definition

3D object detection estimates oriented rectangular boxes and a class description for physical objects from various sensor data [1]. Ideally, a 3D bounding box captures object information while having the minimum volume. In other words, the goal is to know the location and size of all objects of interest.

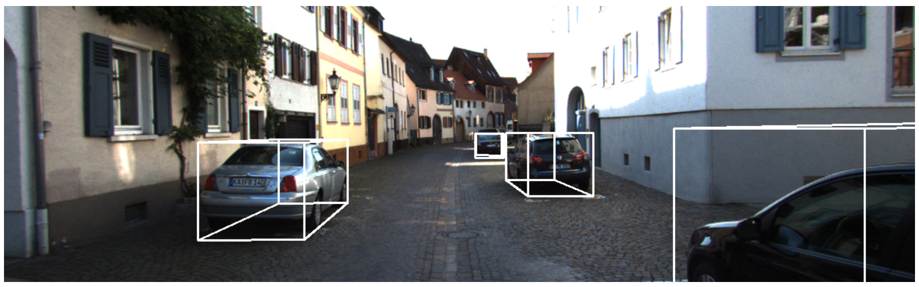

Figure 1.1: Visualization of 3D object detection results.

Figure 1.1 shows an example image after applying an 3D object detector. As described before, the task is to recognize all objects from a specified input data.

1.1.2 Motivation

The rise of autonomous vehicles development brought new challenges into the field of modern computer science. They need to be aware of their environment and navigate safely without human intervention [2]. In particular, the localization and detection is one of the most important tasks of autonomous vehicles vision perceptual systems.

Currently, most of the 3D object detection and recognition methods heavily rely on LiDAR (Light Detection and Ranging) data for providing accurate scene information. LiDAR uses light waves to measure the range to a target [3]. It shoots off thousands of rays simultaneously to build up a 3D measurement of the current surroundings, commonly referred to as a 3D point cloud. Due the high precision and data rate, LiDAR systems are used by a majority of modern autonomous vehicles. However, LiDAR sensors have the disadvantage of very high costs and lower density of captured points compared to traditional cameras. Additionally, weather-based reflections caused by rain or fog can present issues for LiDAR systems [4].

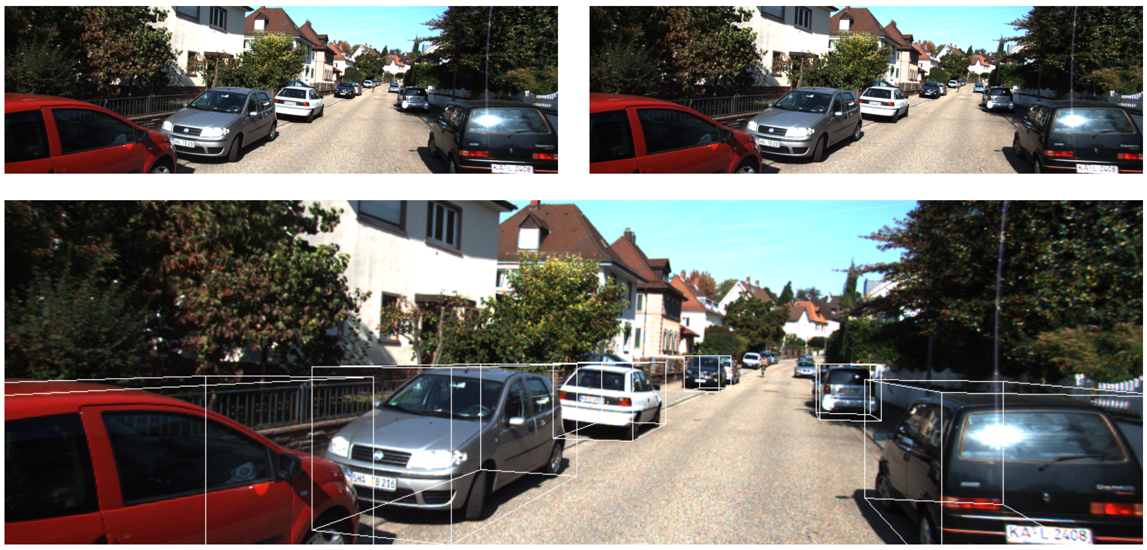

FIGURE 1.2: Example from the KITTI dataset. Top: Stereo input images. Bottom: 3D bounding boxes of objects projected into the input image.

Other methods estimate 3D bounding boxes from stereo input images. Here, the input consists of two separate images of the same scene, taken simultaneously from two nearby points of view. For example, these two cameras could be placed on top of a car. Using triangulation, the distance to objects can be provided via correspondence estimation of the input image pairs [5]. In combination with the original input images, this depth information is used for predicting the 3D bounding boxes. Figure 1.2 shows an example input and output of 3D box estimation from stereo input. Due to the low cost of camera systems, stereovision is an interesting alternative compared to LiDAR sensors. Since stereovision consists of two cameras, it provides a built-in redundant system. Additionally, LiDAR sensors only measure depth sparsely which may loose details of a scene and do not provide as high resolution as modern camera systems. [6].

3D object detection based on a single camera uses only one image as an input. This approach is often used for autonomous vehicles without space for mounting a second camera. However, a monocular camera system is not capable of capturing enough spatial information of the environment. Therefore, current single image approaches have significantly worse performance compared to stereo and LiDAR methods.

Accordingly, we will focus on 3D object detection using stereo input images in our thesis.

1.1.3 3D Object Detection From Stereo Input Images

3D object detectors rely heavily on the underlying stereo vision algorithm. These algorithms estimate the depth of objects by obtaining the disparity between two input images. Traditionally, the disparity is calculated by searching for corresponding pixels in the input image pair [7].

In recent years, the progress of neural networks also pushed the performance in the field of 3D object detection from stereo input images. Presently, most object detection systems are based on convolutional neural networks (CNNs). We will shortly introduce recent methods in this section.

Chen et al. [8] formulated the problem of placing pre-defined bounding boxes in the 3D space as an energy function. After calculating the depth using traditional stereo methods, they estimate the bounding box coordinates and direction using a CNN.

Li et al. [9] introduced the extraction of region proposals [10] from the input image pair. Using the corresponding left and right region proposals, they regress 3D bounding boxes at different scales using a pyramid structured network [11].

Recently, Wang et al. [12] proposed a modular pipeline which first converts a predicted disparity map into a 3D representation structure. Afterwards, this point-cloud representation is fed into an existing LiDAR point cloud detector. This method performs at state-of-the-art level on current public datasets. However, it needs careful tuning of several hyper-parameters and both components cannot be trained jointly.

Li and colleagues [13] showed that focusing on objects of interest during depth estimation increases the overall accuracy of 3D object detection. In particular, pixels associated with foreground objects are crucial to the object detection accuracy.

Accordingly, we propose a 3D object detection pipeline with integrated foreground object masks. Additionally, we suggest modifications to train the network in an end-to-end manner.

1.2 Contributions

This thesis seeks to address the challenge of estimating the 3D bounding box of an object using the left eye's and right eye's view of a fixed camera setup as an input. In particular, we want to employ a CNN to estimate 3D bounding box parameters from calculated input image depth information. The final step will be to compare the proposed method to current stereo 3D methods on public benchmarks like the KITTI 3D dataset.

To summarize, the goal of the thesis is:

- **Integration of foreground object masks during training**
  Li et al. [13] showed that 3D detection accuracy heavily relies on pixels associated with foreground objects. Defined as split stereo matching, they estimate depth maps individually for foreground and background objects. However, they did not investigate into further usage of segmenting foreground objects. Therefore, we propose a modified network architecture that includes object masks into the point cloud detector module of 3D object detection. In our experiments, we show promising results on public datasets using additional object masks during training.

- **End-to-end 3D object detection**
  The object detection pipeline proposed by [14] is based on a modular design consisting of two independent components which need to be trained separately. According to Bojarski et al. [15], this decoupled approach of training two networks using a loss objective that does not acknowledge dependence may not be the best fit for the final learning objective of 3D object detection. To this end, we study an approach to train whole pipeline for a global objective. We enable joint training of both modules and optimize for a global loss function. Our approach is able to increase the object detection accuracy compared to the decoupled structure used as a baseline.

- **Comparison with other methods**
  Finally, we want to compare the proposed approach with other current 3D detection methods on public datasets.

1.3 Outline

The first chapter gives a short introduction into the topic of the thesis. Chapter 2 contains related work and theoretical backgrounds in the field of 3D object detection.

In Chapter 3, we define a framework for estimating 3D bounding boxes from stereo input images. Additionally, we report the network architecture and details of the applied model.

Chapter 4 focuses on the integration of object masks. Further, we describe how these masks can help to increase detection accuracy.

End-to-end learning is described in Chapter 5. We first show how individual components of the object detection framework can interact with each other. Afterwards, we describe a global learning objective for jointly training the framework components.

The results of the conducted experiments are reported and discussed in Chapter 6. The final chapter will summarize the aspects of this thesis and consider topics for future extensions.

Chapter 2

Related Work

Contents

The second chapter gives an overview about principles that are of particular interest for this work and provides a survey of related research.

We start with disparity estimation from stereo images and 2D object detection. After that, we discuss point cloud object detectors, 3D object detectors and semantic segmentation.

2.1 Stereo disparity estimation

Autonomous vehicles heavily rely on the quality of depth estimations, which can be obtained from the disparity between two images. The disparity is defined as the displacement of a pair of corresponding pixels in the image pair. The estimation happens by taking an input image pair and calculating the difference in image location of the same 3D point [7]. Rectifying the input images limits the search space for the matching to a one-dimensional scan line called the epipolar line. The difference of corresponding points on the same epipolar line results in the disparity, as shown in Figure 2.1.

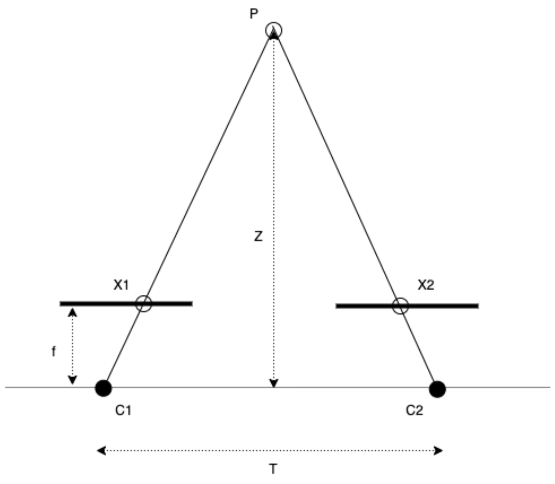

FIGURE 2.1: Visualization of a stereo camera setup, where C1 and C2 relate to the left and right camera centers, f relates to the focal length and T is defined as the stereo baseline . Point X1 on epipolar line in the left image plane can appear anywhere on the epipolar line in the right image plane.

Traditional stereo vision algorithms can be separated into global, local and semiglobal approaches. Global approaches formulate a cost function over the whole image. Then, this cost function gets minimized by applying global matching methods such as Graph Cut [16]. Global methods achieve high accuracy for the disparity output. However, minimizing the global cost function step-by-step results in more computational complexity.

Local methods displace a predefined support window to search for points with high correlation. This behavior achieves better runtimes while suffering from lower quality than global approaches.

Semiglobal methods [17] combine the efficiency of local methods with the accuracy of global methods. They approximate a 2D minimization problem with several 1D scanline optimizations, which can be solved efficiently via dynamic programming. However, traditional methods are of limited accuracy due to their hand-crafted feature selection [18].

With the advent of machine learning methods in computer vision, Convolutional Neural Networks (CNNs) replaced several components of the traditional vision pipeline. Zbontar et al. [19] pioneered by leveraging CNNs to compute the matching cost of a stereo vision task. Their Matching Cost Convolutional Neural Network (MC-CNN) learns a similarity measure of small image patches which gets trained in a supervised manner. The outputs of the network are used as an initial matching cost before applying several post-processing steps. Compared to traditional methods, MC-CNN achieves considerable gains in accuracy.

Mayer et al. [20] replaced the whole stereo pipeline by introducing end-to-end stereo learning. Their disparity network first constructs a cost volume using correlation layers and then estimate the disparity map by end-to-end training. Additionally, they released a synthetic dataset with over 35,000 stereo frames. On public benchmarks, their network achieves smaller stereo error results compared to traditional multi-stage approaches.

eometry and Context Network (GC-Net), introduced by Kendall et al. [21], proposes to regularize the cost volume with 3D convolution layers. In contrast to traditional distance metric methods, they construct their 4D cost volume by concatenating the unary features of the input images to maintain the original geometric input information. The innovative usage of 3D convolutions results in higher memory usage while producing more accurate disparity values.

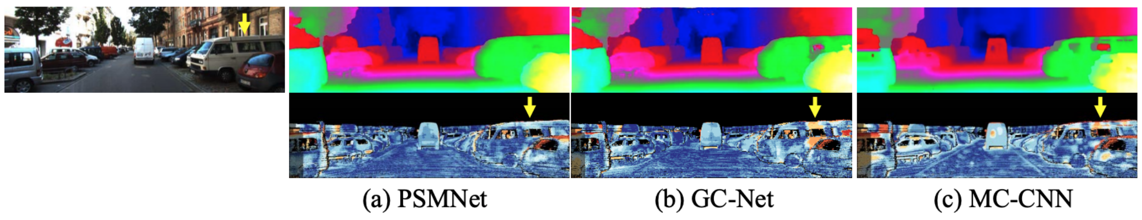

FIGURE 2.2: Qualitative results of different stereo methods. The left column shows the according input image. On the right side, the disparity and error map for each method is obtained. Image taken from [11].

Pyramid Stereo Matching Network (PSMNet) by Chang et al. [11] tackles this problem by proposing a pyramid stucture. While extracting multi-scale features, the network decreases the high-resolution input into a lower resolution. This way, it can take advantage of global information in addition to pixel-level features by aggregating context in different scales. The pyramid feature performs the matching in a hierarchical manner, constructing a single cost volume using multiscale features. As visualized in Figure 2.2, PSMNet outperforms other modern methods especially in ill-posed regions while having the same run-time speed.

2.2 2D Object Detection

The task of 2D object detection is to estimate a 2D bounding area and a class description for all objects of attentation in a given input image. The 2D bounding box is described by four parameters and should fit the desired object closely. An example visualization is shown in Figure 2.3.

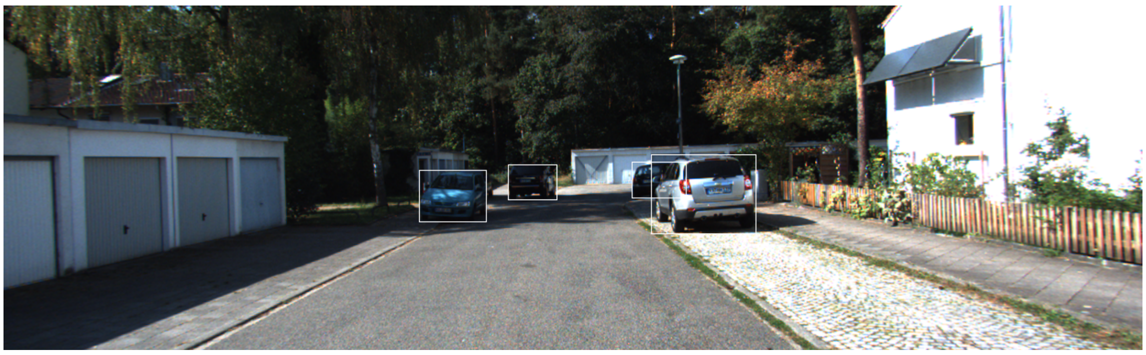

FIGURE 2.3: Example 2D bounding box groundtruth annotations from the KITTI dataset. Image taken from [22].

There have been large improvements in 2D prediction accuracy on account of the advent of CNNs for feature extraction and the availability of larger and more complexe training data. The modern detector pipeline has been introduced by Region Based Convolutional Neural Networks (R-CNN) [23].

The first step of R-CNN is that several proposed regions from the input image on multiple scales are selected via a predefined method like selective search [24] through basic color segmentation. The aim of this search is to filter out objects of interest from background scenes. After that, the selected regions are resized and fed into a pretrained AlexNet [25] CNN which extracts a feature vector for each region. Finally, support vector machines (SVM) predict a bounding box and object category from the extracted feature vectors. Although producing appealing results, the main downside of R-CNN is the massive computing load due to the high number of individual forward computations of the CNN.

Since the feature extraction for each individual region produces a large amount of repeating calculations, Fast R-CNN [26] combines them into one CNN forward pass over the entire image. Then, the output of the CNN and the regions from the selective search are combined using a region of interest pooling layer. Before predicting bounding boxes and classes, the systems applies a fully connected layer for transforming the merged feature vector. Since only one CNN forward computation is necessary, Fast R-CNN decreases both training and inference time compared to R-CNN.

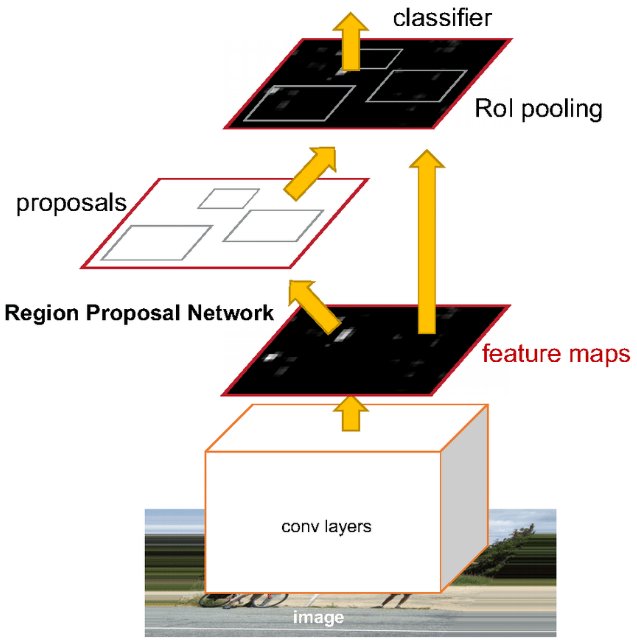

FIGURE 2.4: Overview of Faster R-CNN. The model consists of convolutional layers and a final RoI pooling layer. Image taken from [27]

The framework was quickly followed by Faster R-CNN [27], which introduced the region proposal network (RPN) as displayed in Figure 2.4. This results in a single model consisting of an RPN and Fast R-CNN which can be trained together using weight sharing.

The RPN is a fully convolutional network which uses every element of the extracted map as a base for anchor generation. Anchors are predetermined bounding boxes which serve as a default size for the region proposals to assume. Finally, all positive anchors get refined by a regressor to produce a final output. The sharing of computation of Faster R-CNN increases the pipeline performance further, proceeding in state-of-the-art (SOTA) scores on public benchmarks.

In contrast to two-stage approaches described before, one-stage object detection models directly predict a set of bounding boxes on an input image. They aim to encapsulate all calculations in one network, resulting in an easier training process compared to two-stage approaches. One of the first attempts in this direction was SSD (Single Shot Detector) [28] introduced by Liu et al. which treats object detection as a regression problem.

Similar to RPN, the network places pre-defined anchors on extracted feature maps. The feature maps in different scales can be extracted from individual layers of the CNN. Finally, matching anchors from different scales are merged and pruned by non-maximum suppression (NMS) to produce scores for every class besides width and height of the bounding area.

Another one-stage approach named YOLO (You Only Look Once) [29] aims toward real-time object detection. It uses fully convolutional layers for feature map extraction. After that, anchors are placed on grid locations similar to SSD. Additionally, YOLO predicts a confidence score for each resulting anchor apart from the anchor size and offset.

In summary, one-stage detectors defined new benchmarks in the field of 2D object detection inference time. However, they reach lower accuracy scores than two-stage detectors while being significantly faster [30]. Furthermore, one-stage detectors often have problems to detect small objects compared to region based networks [31].

Recently, fully-convolutional anchor-free methods like FCOS [32] and CenterNet [33] have been introduced. One major advantage of anchor-free object detectors is the removal of hyper parameter tuning.

In general, FCOS uses Feature Pyramid Networks (FPN) [34] for feature map extraction. On every pyramid level, a detection head consisting of fully connected layers is used for prediction. The design of this pyramid allows to share the rich semantics of higher level feature maps to lower level feature maps.

Besides a class label and four bounding box values FCOS also predicts a centerness score for each pixel in an image. This means that the number of predictions matches the pixel count exactly. The centerness score is defined as

$$ctr = \sqrt{\frac{min(l,r)}{max(l,r)} \times \frac{min(t,b)}{max(t,b)}}, \tag{2.1}$$

where l, r, b and t denote the bounding box regression targets. It refers as an value describing the distance to the center of the ground truth bounding box for suppressing low-quality predictions. FCOS surpasses previous one-stage detectors on public benchmarks with the advantage of being much simpler. However, it does not reach the accuracy of recent two-stage detectors.

2.3 Deep Learning on Point Clouds

As deep learning has shown promising results on images, there has been interest to transfer this techniques into geometric spaces like 3D point clouds as found in Figure 2.5. Initial methods [35][36] applied traditional deep learning models on point clouds transformed into multiple views. This way, existing methods could be used for geometric spaces without expensive network modifications.

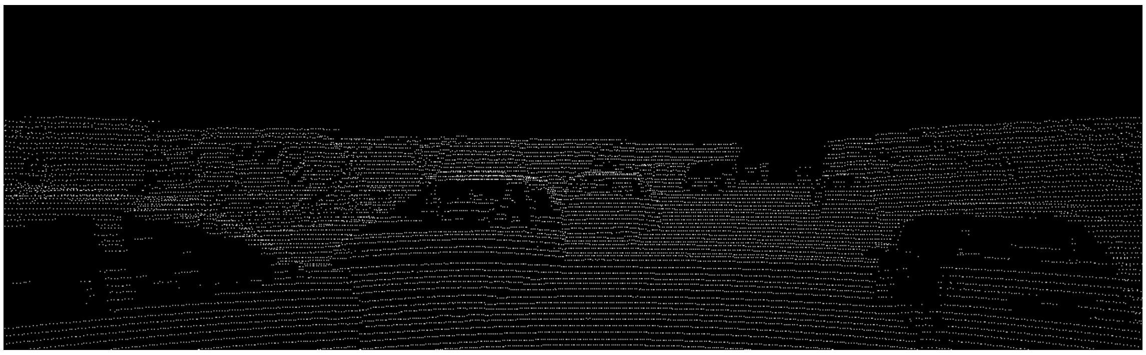

FIGURE 2.5: Example lidar visualization from the KITTI dataset. Image taken from [22].

Squeezeseg [37] introduced by Wu et al. adopts a spherical projection to transform sparse 3D point clouds to dense 2D grid representations. This way, they can apply conventional CNN models on the generated front view camera images. Another preprocessing method transforms the input point cloud into a birds-eye-representation before applying a CNN [38].

Currently, there are two streams on how to apply deep learning on point clouds: voxel-based and point-based.

Conventionally, VoxelNet [39] proposed by Zhou et al. divides 3D data points into equally-sized 3D elements named voxels. It introduced voxel feature encoding (VFE), which allows interactivity inside a voxel-element, by merging point-wise structures with a multi-voxel extracted feature. It then groups up all points inside a voxel to a representive feature. Finally, a region proposal network with stacked structure is applied to integrate multi-scale representations.

However, transformation operations in the prepocessing step of voxel-based methods often leads to information losses. Additionally, they produce intense computational cost [40]. Therefore, deep learning models which operate on raw point clouds have been introduced.

Processing on raw point clouds has been pioneered by PointNet [41]. Instead of 3D voxels, PointNet directly consumes a point cloud without any preprocessing, which also respects the invariance of points in the input using symmetric functions. One big advantage of PointNet is that it can learn both individual point feature vectors and global point aggregation feature vectors. Figure 2.6 show the general architecture of PointNet.

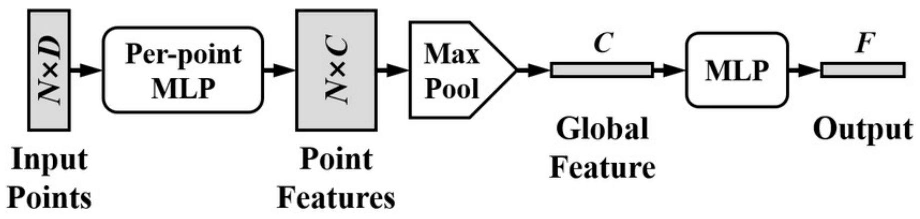

FIGURE 2.6: Basic architecture of PointNet. Image taken from [42]

PointNet first applies a multi layer perceptron (MLP) to each indivual point p_c to get a feature vector f_c. Using max pooling, all feature vectores result into one global feature vectore f_g describing the whole point cloud. Eventually, this global feature vector is fed into another MLP to output a final classification score.

Furthermore, an extension to PointNet uses the same base network for classification and segmentation by adding a segmentation network. It therefore concatenates local geometric information and global semantics to compute output scores for each individual point.

PointRCNN [43] exploits PointNet by predicting and segmenting objects directly on point-clouds. It is composed of two stages.

The first phase generates 3D bounding area proposals and acts as a classic region proposal network. First, it uses PointNet as a base network to extract a description vector f_c for every input point p_c. Next, all foreground features are put into a box regression head to generate a 3D bounding box proposal.

The distinction between foreground and background features helps to predict better proposals and is naturally given by the groundtruth annotations. Finally, NMS from bird's eye view is applied to filter out redundant bounding box proposals.

The aim of the second stage is to refine the locations and orientations of the generated box proposals. It starts with a pooling of 3D locations and their correlated extracted features from the first phase. Therefore, PointRCNN enlarges proposed bounding boxes by a constant factor to include additional information from the context.

After that, the refinement sub-network in the second stage merges the locally extracted features with the global description. Then, the combined features are fed into a network related to [44] for generating a discriminative feature vector which is necessary for the following steps.

Finally, the output layer directly regresses the size offset from the average dimension of each training set object. For the orientation a bin-based regression loss is applied. Additionally, a confidence score is trained using a traditional cross-entropy loss. All 3D boxes with a IoU score higher than 0.55 are counted as training samples.

With PointRCNN, Shi et al. claim to address key flaws of traditional point cloud detectors. Using the bottom-up proposal approach, they claim to achieve significantly higher recall than previous methods [39]. The proposed concatenation of local and global semantic feature vectors in combination with the bin-based loss demonstrate the efficiency and effectiveness of PointRCNN for 3D bounding box regression. Compared to other methods [45], PointRCNN reports improved performance for 3D object detection on public datasets while having the same inference time.

2.4 3D Object Detection

3D object detection estimates oriented 3D bounding areas and a class label for physical objects from various sensor data. A 3D box is described by the box core point (x,y,z), dimensions (h,w,l), and an observation angle. Ideally, a 3D bounding box captures all necessary information while having the minimum volume. Example boxes are visualized in Figure 2.7.

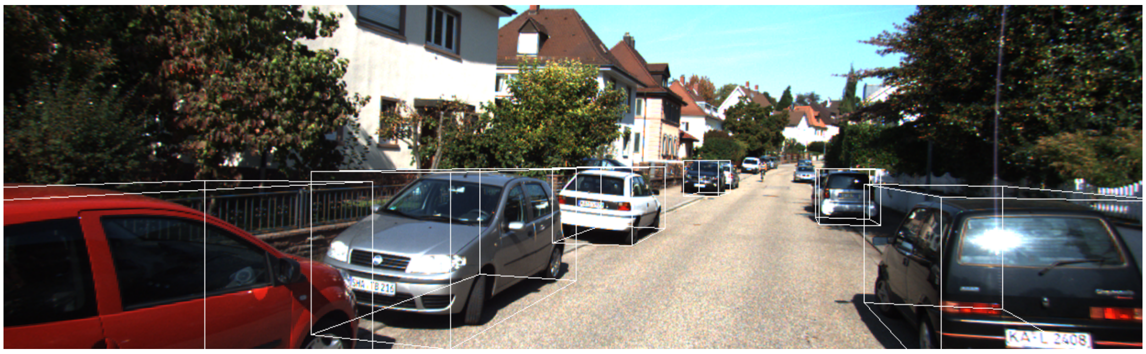

FIGURE 2.7: Example 3D bounding box groundtruth annotations from the KITTI dataset. Image taken from [22].

There are various sources for sensor data, such as e.g. RADAR (Radio Detection and Ranging), LiDAR, monocular cameras, or stereo cameras. Due the focus of research on autonomous cars, there has been more attention on LiDAR, stereo and monocular input sources for 3D frameworks.

However, stereo cameras have shown to be a good trade-off between efficiency and computation time. Due their low acquisitions cost, they are a viable alternative to LiDAR sensors. Additionally, they can be used to generate dense depth maps. In comparison, LiDAR describes scenes by sparse point clouds.

3DOP (3D Object Proposals) [8] proposed by Chen et al. was one of the first deep learning 3D pipelines in the context of autonomous vehicles using stereo images as an input. They formulate the problem of placing pre-defined anchors in the 3D space as an energy function. After calculating the depth using traditional stereo methods, they estimate the bounding box dimensions and direction by applying a CNN. Using this combination, this approach outperformed all non-CNN based approaches until then.

In 2019, Li et al. [9] introduced Stereo R-CNN as an extension to Faster R-CNN which simultaneously detects and identifies objects in stereo input images. In contrast to the classic R-CNN pipeline, Stereo R-CNN uses a modified RPN to output corresponding left and right proposals. Similar to FPN, the RPN of Stereo R-CNN uses a pyramid structure as visualized in Figure 2.8. It merges the extracted left and right features at every scale before continuing. The RPN regresses proposals 2D bounding box coordinates for the left image and a center offset for the right image.

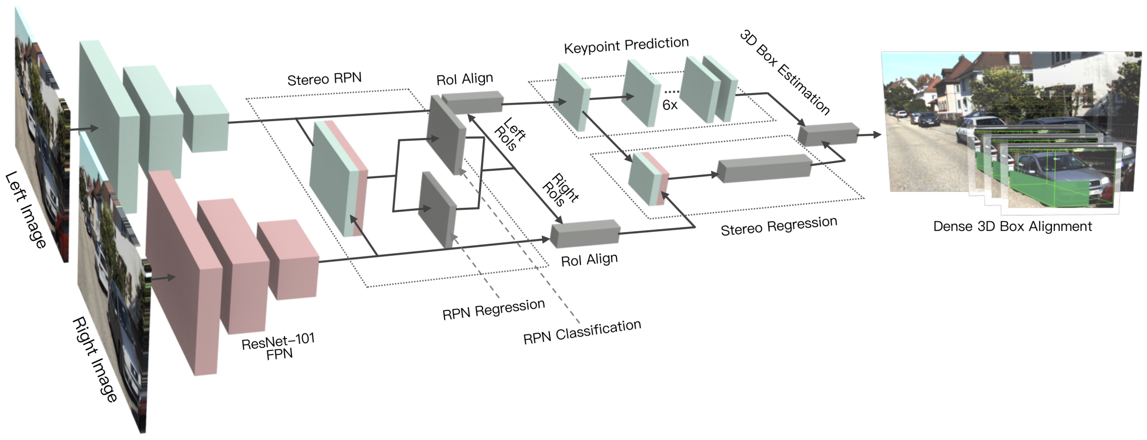

FIGURE 2.8: Stereo R-CNN architecture. Image taken from [9]

After that, RoI Align [46] is applied on these left-right proposal pairs. The aligned features get passed into fully connected layers for semantic information extraction. The output of these layers are all bounding box regression terms defined above.

Besides bounding boxes, Stereo R-CNN also predicts a perspective keypoint. Keypoints are defined as the four corners at the bottom of a 3D cubicle. Stereo R-CNN defines the perspective keypoint as the only visible keypoint projected to the box middle. This keypoint prediction helps to provide more constraints for estimating the 3D bounding box. The final step of Stereo R-CNN is to predict a dense bounding box by solving the disparity problem on the final left-right object pair.

More recent methods convert an estimated depth map from stereo input into a 3D representation before applying a point cloud detector. Methods based on the Pseudo-LiDAR framework [12] receive highest scores on public datasets by taking advantage of the best networks from stereo estimation and point cloud detection. They argue the remaining gap between image and LiDAR-based detection results that stereo images are poorly represented inside CNN-based 3D object detection pipelines. In comparison, computations on transformed point clouds [47] are invariant to depth changes. Therefore, they propose a two-step approach based on existing techniques which get aligned together.

Instead of a direct 3D estimation on a given stereo image pair, Pseudo-LiDAR starts with a per-pixel disparity estimation. After calculating the depth value from each per-pixel disparity, the depth map gets transformed into a sparse 3D point cloud representation as shown in 2.9, which they refer as Pseudo-LiDAR. This transformation happens by back-projecting all the pixels and their individual depth into 3D coordinates.

To imitate a real LiDAR signal, the dense Pseudo-LiDAR point cloud gets sub-sampled and cropped to a height of 1 m. Treating this fictitious signal as a LiDAR point cloud, any 3D detector can be applied. Specifically, they apply AVOD [45] and achieve double the performance of previous state of the art. Additionally, they prove the compatibility of their approach by applying different 3D detectors [47].

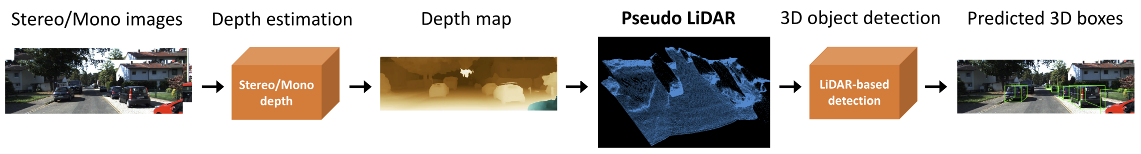

FIGURE 2.9: Overview of the Pseudo-LiDAR pipeline. Image taken from [12]

Pseudo-LiDAR++ [14] modifies the framework to be more tailored for autonomous driving. They argue that the calculation of pixel-level depth by inverting the disparity is a major source of error. The over-emphasizes on near objects of traditional stereo networks results in a poor depth estimation for far-away objects. Especially, they adjusts the stereo loss function to predict distant objects more precisely by directly regressing the per-pixel depth and enabling 3D convolutions.

In addition, they propose to use an extremely sparse LiDAR sensor to boost object detection accuracy. Since standard LiDAR sensors using 32 beams are expensive, they connect a 4 beam sensor input to their pipeline. Paired with dual image input, the sparse signal can help to de-bias the depth error from stereo estimation. Pseudo-LiDAR++ shows consistent improvement over its predecessor on all benchmarks.

The work of Li et al. [13] is also based on the same framework while using two separated depth estimation networks for foreground and background scenes. This separation of depth networks causes less depth estimation errors. They prove that this modification also yields a better 3D object localization accuracy. Additionally, they leverage the predicted confidence scores from the foreground stereo prediction as an supplementary input for the 3D point cloud detector. This approach pushes the CNN to aim for points with high scores. Naturally, the extension of the depth network increase the training and inference speed by large margins. On the other hand, they surpass the next best-performing method by 1.4 % on average.

In 2020, DSGN (Deep Stereo Geometry Network) [48] introduced an one-stage detection pipeline which detects and localizes 3D bounding boxes in an end-to-end way. DSGN intermediately creates a plane sweep volume [49] which encodes accurate geometric information of the 3D space. They create this volume from the input images to learn stereo correspondences in the camera frustum as show in Figure 2.10.

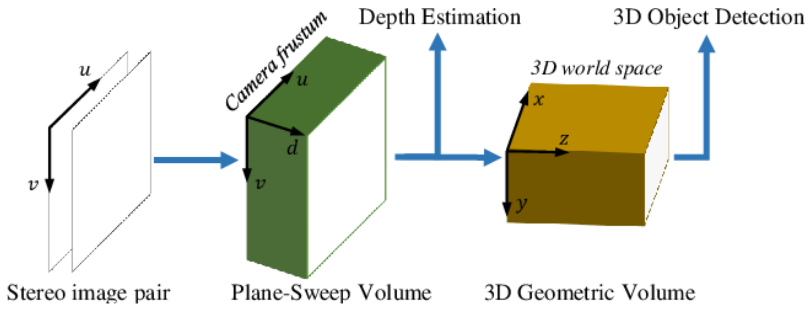

FIGURE 2.10: DSGN detection pipeline. Image taken from [48]

After that, this plane sweep is lifted into a 3D geometric volume where 3D convolution layers can be applied. This results in a top-view representation of the setting. By learning the necessary object structure in this 3D world space, DSGN outruns all current stereo networks. However, multi-task training and lifting into 3D space requires more GPU memory than other methods.

To get a good starting point for our research, we conducted a comparison of current methods. The results of this comparison can be found in Table 2.1. Besides the 3D object detection scores, we also noticed the inference runtime for one image-pair and the memory consumption during training.

Method name	KITTI AP@0.7 Val Set			Inference	Memory
	Easy	Moderate	Hard	Runtime [s]	consumption
Stereo R-CNN [9]	54.11	36.69	31.07	0.35	11 GB
Pseudo-LiDAR [12]	61.90	45.30	39.00	0.4	8 GB
Pseudo-LiDAR++ [14]	67.90	50.10	45.30	0.4	8 GB
DSGN [48]	73.21	54.27	47.71	0.67	24 GB
CG-Stereo [13]	76.17	57.82	54.63	0.57	11 GB

Table 2.1: Different 3D detectors applied on the KITTI validation dataset.

We can clearly see that CG-Stereo yields the best 3D detection scores of all cited methods. However, the inference speed of 0.57 seconds is considerably slower compared to e.g. Pseudo-LiDAR. DSGN produces very good results for 3D detection while consuming 24 GB of GPU memory.

Pseudo-LiDAR++ ranks both in the midfield in terms of detection accuracy, runtime and GPU memory. Stereo R-CNN has the fastest runtime while having the worst 3D detection results of all compared methods.

2.5 Semantic Segmentation for Street Scenes

The objective of semantic segmentation is to estimate a class to every pixel in an picture [50]. Since every pixel gets classified, semantic segmentation is commonly referred as dense prediction. In contrast to instance segmentation, there is no distinction of separate objects of the same class.

The first methods to solve the problem of semantic segmentation were base on Random Forests [51] and K-means Clustering [52]. Modern approaches like U-NET [53] are based on fully convolutional networks to take input images of arbitrary size. They consist of an encoder-decoder system and take advantage of skip connections to overcome information loss during sampling of images.

In 2018, Yang et al. proposed SegStereo [54] for pixel-level semantic segmentation on the KITTI semantic dataset. It exploits semantic cues into a backbone disparity network. While originally constructed for stereo estimation, SegStereo yields appealing results in the task of semantic segmentation.

Ladder-style DenseNets (LSDN), [55] introduced by Kreso et al., construct an effective architecture for semantic segmentation which augments the base CNN with ladder-style skip-connections. LSDN supports calculation on very large natural images without high-end hardware, by having only few feature maps at higher resolution.

Since the annotation of training data for semantic segmentation networks is very expensive, recent work has proposed the integration of synthetic data. Particularly, VideoPropLabelRelax [56] has shown significant improvements in accuracy on the KITTI semantic segmentation dataset by scaling up training data with video prediction networks. It incorporates new training samples by predicting future frames and future labels. These predictions result from a given sequence of past video frames which have been sparsely labeled at regular intervals.

Method	IoU class
SegStereo	59.10
LSDN	63.51
VideoPropLabelRelax	72.83

Table 2.2: Different semantic segmentation methods applied on KITTI.

Table 2.2 compares the segmentation performance of the discussed methods. We can clearly observe that VideoPropLabelRelax yields the best results on the KITTI validation dataset.

In semantic segmentation, pixel-wise cross entropy [57] is often used to evaluate class label predictions. This can be problematic when there is an imbalance of class labels as the most widespread class may dominate the training.

In order to counteract this circumstance, Focal loss [58] down-weights well-classified classes and adjusts on rare classes. Particularly, this approach can be very useful for autonomous driving scenes where there is an imbalance between background and foreground classes.

2.6 Chapter Summary

This chapter gave an introduction into the current research of 3D object detection. It started the concept of disparity estimation from stereo input pairs. This disparity estimation is fundamental for any 3D object detector using stereo input. After that, we described current point cloud and 3D object detectors. We concluded with insights about current semantic segmentation networks for road scenes.

Chapter 3

3D Object Detection

Contents

In this section we present a detailed explanation of the base architecture and methods for 3D object detection which have been implemented. We begin with a discussion of chosen methods and continue with the proposed network architecture. We conclude with an in-depth assessment of the depth estimation and 3D detection modules.

3.1 Methodology

As already discussed, the task of 3D object detection is to estimate oriented 3D boxes and class labels for physical objects from various sensor data. Due to reason described in the section before, we focus on stereo image-pairs as an input source for our detection pipeline.

For better understanding the possibilities for improvement of the current methods, a comparison of these methods has been conducted. We compared them on several benchmarks and also tried to weigh up their respective strengths and weaknesses. The findings of this assessment can be found in Table 2.1.

As a result, we were able to identify the advantages and disadvantages of these methods, creating our own modifications to overcome the observed limitations. Based on these findings, we decided for one approach as starting point.

Pseudo-LiDAR++ [14] has shown to produce visually compelling results for 3D object detection problems and to close the gap between image- and LiDAR-based detection. By both enforcing low GPU memory usage during training and a focus on short inference runtime, we chose Pseudo-LiDAR++ as a baseline for our modifications. Furthermore, we claim that the modularity of Pseudo-LiDAR++ offers chances of improvement while being a good starting point for 3D object detection.

We also notice the trend of effective end-to-end 3D learning approaches like DSGN [48]. The joint training of one unified network is able to learn both pixel- and high-level features. Using a multi-task loss, 3D geometry encoding of point clouds and expensive 3D convolutions, they achieve comparable accuracy with a few LiDAR-based methods.

Additionally, we observe the impact of better object stereo estimation on 3D detection results implemented in CG-Stereo [13]. Li et al. have shown that the 3D detection performance directly depends on the quality of the preceding stereo estimation. By using a separate stereo estimation network for foreground objects, they outperformed all current approaches evaluated the KITTI 3D dataset.

3.2 Network Architecture

The default pipeline is based on a modular design consisting of two independent components which need to be trained separate. Starting with a left-right image pair, a modified stereo estimation network predicts a depth map with the left image as a reference image. After that, this depth map gets transformed into a point cloud. This transformation happens via back-projecting all pixels into 3D coordinates. After subsampling the resulting Pseudo-LiDAR signal, we apply a 3D point cloud detector to estimate final object bounding boxes.

Figure 3.1 gives an overview of the proposed network architecture. In the next chapters we will describe the individual modules and modifications of this pipeline.

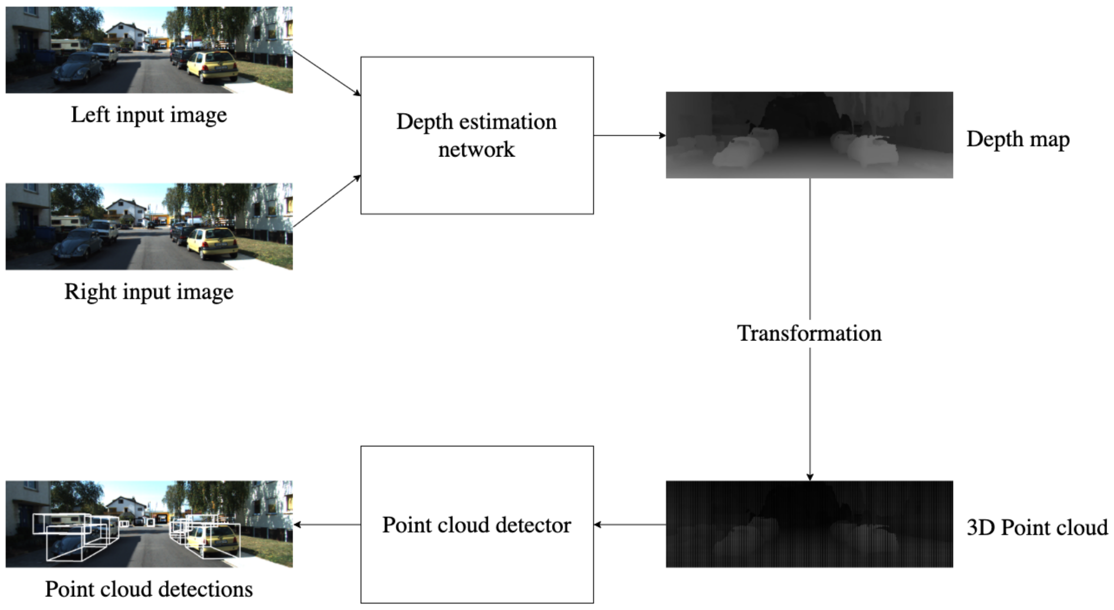

Figure 3.1: Visualization of the proposed pipeline architecture

3.3 Depth Estimation

The whole framework relies on the precision of the previous depth prediction. Depth errors get propagated through the network which directly affects the accuracy of 3D detection negatively.

Therefore, we chose a proven stereo architecture similar to PSMNet [11] as a foundation. As shown in Section 2, it surpasses other approaches both in quantitative and qualitative results. Additionally, the pyramid structure of PSMNet can handle multiple depth map objects at different scales.

Haeusler et al. [59] have shown that stereo estimation networks perform inferior for far-away objects. Especially for small objects like cars, a small disparity error may result in a huge error in depth. Additionally, Izzat et al. [60] showed that disparity estimation networks spend more time on correcting errors of nearby objects than of distant objects.

Since the aim of PSMNet is to estimate disparities, we adapt it to better align with the learning objective of depth estimation. We modify the loss to directly regress the depth values instead of disparity values. Additionally, we transform the constructed disparity cost volume towards a depth cost volume before applying 3D convolutions.

A schematic overview of our pipeline can be found in Figure 3.2.

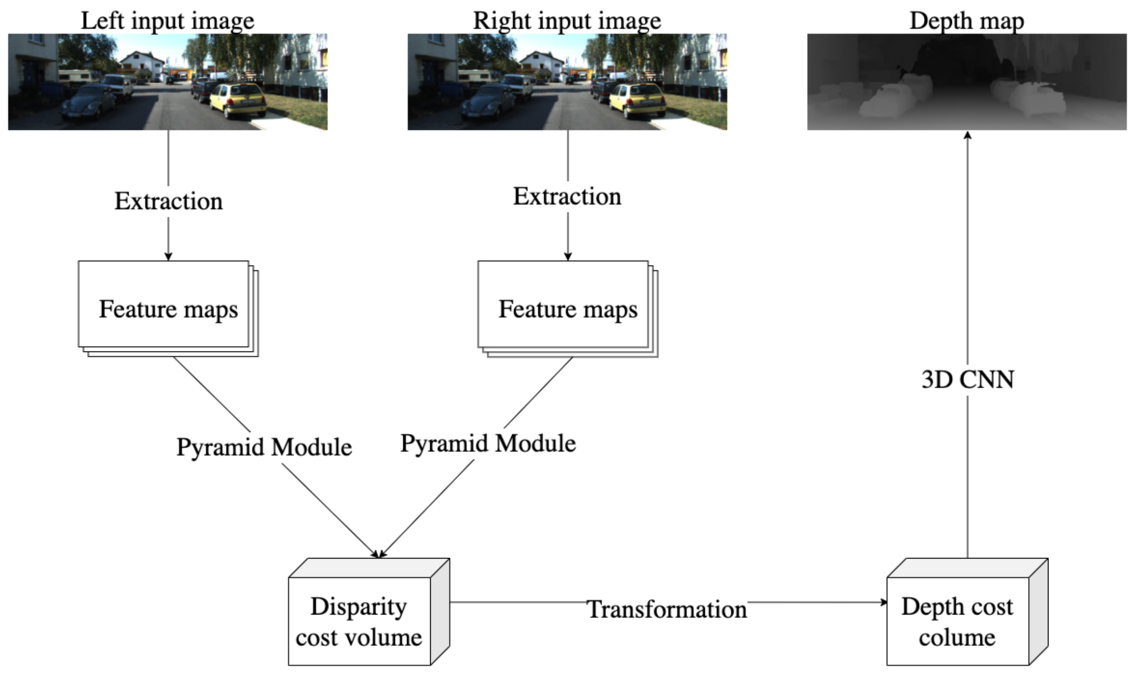

Figure 3.2: Our proposed depth network estimation network.

The feature extractor network uses two input pictures I_l and I_r as inputs and extracts feature maps m_l and m_r from them using a weight-sharing CNN. We can see the details of the feature extractor network in Table 3.1.

Contrary to large filter sizes proposed in other approaches [61], our network starts with three stacked convolutional layers with a filter size of 3x3 to develop a deeper network. Each layer uses batch normalization [62] and is followed by a ReLU activation function. Batch normalization is a method to stabilize a deep neural network by normalizing the inputs of nonlinear activation functions.

After that, several residual blocks with 3x3 filter size are applied. A residual block [61] uses skip connections to add the output from a previous layer to the layer ahead.

As shown in the last column of the table, the feature extractor CNN reduces the pixel size of the original image to one fourth. This downsampling happens via the layers using a stride of 2. Stride describes the amount of movement between the application of the filter.

The final residual block contains convolutional layers with a dilation of 2. Introduced by Yu et al. [63], dilated convolutions are normal filters applied to an input with defined gaps. This way, the receptive field can be enlarged.

Feature Extractor CNN					
Block	Filters	Filter Size	Stride	Dilation	Output dimension
Conv + ReLU	32	3 x 3	2	1	H/2 x W/2 x 32
Conv + ReLU	32	3 x 3	1	1	H/2 x W/2 x 32
Conv + ReLU	32	3 x 3	1	1	H/2 x W/2 x 32
3 x ResBlock	32	3 x 3	1	1	H/2 x W/2 x 32
16 x ResBlock	64	3 x 3	1 (first block: 2)	1	H/4 x W/4 x 64
3 x ResBlock	128	3 x 3	1	1	H/4 x W/4 x 128
3 x ResBlock	128	3 x 3	1	2	H/4 x W/4 x 128

Table 3.1: Feature extractor network structure.

To incorporate hierarchical context information, He et al. [61] introduced spatial pyramid pooling. This approach also removed the fixed input size constraint of CNNs. Spatial pyramid structures in stereo estimation networks have shown to produce significantly improved performance on challenging scene tasks [64].

Having this advantages in mind, we feed the extracted feature maps of both images into a pyramid pooling module. Table 3.2 shows the structure of this network.

Pyramid Module			
Block		Details	Output dimension
Branch 1	Average Pooling	8 x 8	H/4 x W/4 x 32
	Conv + ReLU	32 Filter, 1 x 1	
	Bilinear Upsampling		
Branch 2	Average Pooling	16 x 16	H/4 x W/4 x 32
	Conv + ReLU	32 Filter, 1 x 1	
	Bilinear Upsampling		
Branch 3	Average Pooling	32 x 32	H/4 x W/4 x 32
	Conv + ReLU	32 Filter, 1 x 1	
	Bilinear Upsampling		
Branch 4	Average Pooling	64 x 64	H/4 x W/4 x 32
	Conv + ReLU	32 Filter, 1 x 1	
	Bilinear Upsampling		
Concat	Branch 1-4		H/4 x W/4 x 320
	Feature Extractor Output		
	Res16 Block Output		
Fusion	Conv + ReLU	128 Filter, 3 x 3	H/4 x W/4 x 32
	Conv + ReLU	32 Filter, 1 x 1	

TABLE 3.2: Feature extractor network structure.

We have split up the module into 4 branches on different fixed scales. Each pyramid branch starts with an average pooling layer to downsample the input feature to the desired size. After that, a convolutional layer with 32 filter of size 1x1 and a ReLU activation function is applied. The intermediate representation gets upsampled via bilinear interpolation to match the pyramid input size.

The output of all 4 branches gets concatenated with the final output of the feature extractor CNN and the output of the ResBlockx16. Finally, the concatenated features get fused by two standard convolutional layer.

These features are then used to form a 4D disparity cost volume CV_{disp} which expresses the pixel difference among the left and the right picture.

As discussed before, disparity regression can be a source for errors when used for depth prediction. Instead of directly determining the disparity values from CV_{disp}, we modify the original architecture to better fit into the Pseudo-LiDAR pipeline.

In Equation 3.1 we can see the formula to convert a disparity map *D(u,v)* into a depth map *T(u,v)*. Opposed to homogeneous convolutions applied in traditional stereo networks, we can clearly observe the non-linear ratio between disparity and depth. The usage of the same kernel for the whole disparity cost volume may lead to a negative outcome when using disparity networks for depth estimation. Therefore, we transform the disparity cost volume CV_{disp} into a depth cost volume CV_{depth} using Equation 3.1.

$$T(u, v) = \frac{focal\ length \times baseline}{D(u, v)}. \tag{3.1}$$

After forming the 4D cost volume and transforming it as described, we apply the 3D CNN architecture shown in Table 3.3 for the final depth map prediction.

3D CNN			
Block		Details	Output dimension
Cost volume aggregation	Concat left & right pooling module outputs		H/4 x W/4 x D/4 x 64
3D Block x 5	3D Conv + ReLU	32 Filter, 3 x 3 x 3	H/4 x W/4 x D/4 x 32
	3D Conv + ReLU	32 Filter, 3 x 3 x 3	
Upsampling		Trilinear interpolation	H x W x D
Depth Regression		Softmax	H x W

Table 3.3: Cost volume 3D CNN network structure.

The network starts with 5 residual blocks. Each block contains two 3D convolution layers with a filter size of 3 x 3 x 3 followed by a ReLU activation function. As already discussed, the residual blocks use skip connections to ensure the gradient flow. The final output gets upsampled to match the input image dimensions.

The depth regression of the 3D CNN takes place using a softmax operation. Softmax is defined as a function that turns a vector of real values into a vector of real values of the same size that sums up to 1. The final depth value is computed by the sum of each depth value weighted by its probability. Bahdaneu et al. [65] argue that this weighting makes the prediction more robust.

Furthermore, we modify the default loss to directly estimate the depth values instead of disparity values via the following combination,

$$\sum L(T(u, v) - T^{\dagger}(u, v)). \tag{3.2}$$

L denotes the L1 loss and $T^{\dagger}$ the annotated depth image. We use the L1 loss because of its stability and low awareness to deviations [26]. Based on our knowledge from Section 2, we argue that this network structure and modifications improve the depth prediction for autonomous car scenes compared to traditional stereo estimation pipelines.

3.4 3D Object Detection

We chose the PointRCNN model proposed by Shi et al. in [43] as our reference architecture for the 3D object detection component. This network was chosen because of its high results on public datasets [22] and the approach of taking raw point clouds as an input. Figure 3.3 shows an overview of the network.

As PointRCNN is classified as a two-stage network, it first generates bottom-up 3D proposals which get refined afterwards to get final results. Originally, the first stage directly works on 3D point clouds. Since our depth estimation network outputs 2D depth maps, we have to transform them into the right input shape for the 3D object detection. A pixel (u, v) from depth map T will be reshaped into three-dimensional space by

$$x = \frac{(u - q_{\mathrm{U}}) \times z}{f_{\mathrm{U}}}, \quad y = \frac{(v - q_{\mathrm{V}}) \times z}{f_{\mathrm{V}}}, \quad z = Z(u, v). \tag{3.3}$$

$(q_{\mathrm{U}}, q_{\mathrm{V}})$ denotes the camera origin and f_{U} and f_{V} describes the focal length up- and sideways. Furthermore, the transformed three-dimensional represenations needs to be sub-sampled according to [14] because of the high amount of points compared to sparser point clouds captured with LiDAR scanners.

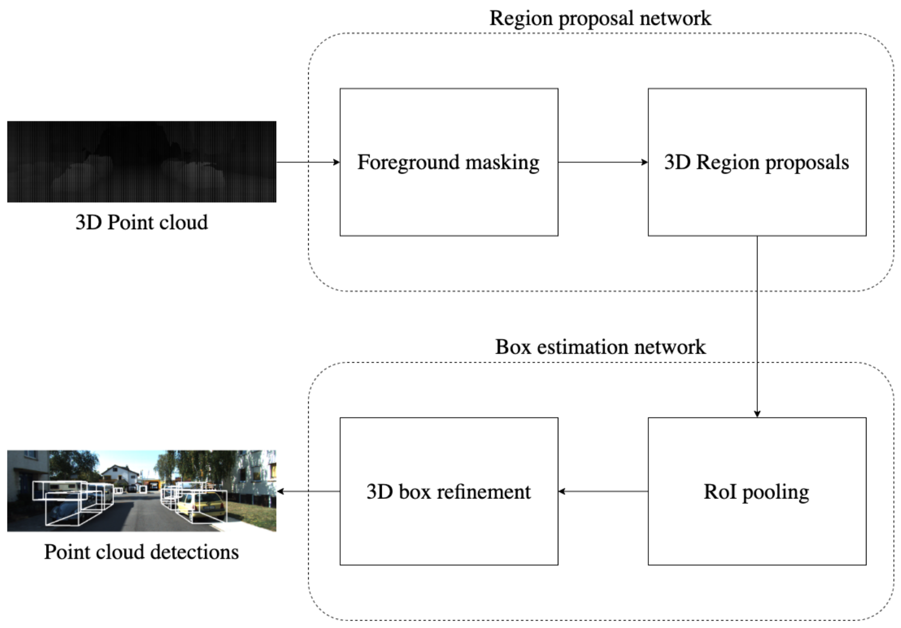

FIGURE 3.3: Visualization of the modified PointRCNN architecture.

3.4.1 Region Proposal Network

Once the sub-sampling is completed, the sparse point-cloud is fed into the first stage of 3D object detection. There, point-wise features get extracted using PointNet++ [44] as a backbone network. All extracted semantic features get pushed as input to the first stage region proposal network. Its structure is presented in Table 3.4.

RPN			
Block		Details	Output channels
Downsampling	4 x SA module		512
Upsampling	4 x FP module		128
Foreground Segmentation	Conv + ReLU	1 x 1	128
	Conv	1 x 1	1
Regression	Conv + ReLU	1 x 1	128
	Conv	1 x 1	7

Table 3.4: Region proposal network structure.

The RPN starts with 4 set abstraction (SA) modules introduced by [44]. Set abstraction can be described as a downsampling of point clouds. New sets are created by finding one centroid point and assigning points to them in a defined radius. Sets are completed by encoding local region patterns into feature vectors.

On the other hand, feature propagation (FP) modules work similar to upsampling layers on point cloud. It uses linear interpolation to upsample the point cloud. In detail, points which get dropped during downsampling are assigned feature vectores based on weighted distances.

Once all features have been extracted, they are fed into two separate heads. Both of them first apply a convolutional layer with ReLU activation function. Finally, the network estimates an output with pre-defined output channels.

As already discussed, we define a bounding box as (x, y, z, h, w, l, θ) in the 3D space. The object dimension (h, w, l) are estimated by computing residuals respective to the pre-calculated category-wise average target class size in the dataset. Similar to [66], the yaw rotation θ for each object is represented in bins using eight scalars. The object center location (x, y, z) gets directly regressed using residuals.

Since generating 3D proposals for every single point is very time-consuming, the model segments the input into foreground and background points via a foreground segmentation head. This improves the performance of box proposal generation due to the limited area of potential bounding box spaces. Furthermore, this segmentation head also builds a synergy among different tasks of segmentation and object detection by providing contextual information.

We formulate the posterior probability as

$$p = \begin{cases} p & if\, foreground \\ (1-p) & if\, background, \end{cases} \tag{3.4}$$

where p is the calculated prediction.

Next, we can define the segmentation loss as

$$\mathcal{L}_{\text{Segmentation}} = -w_{\text{s}} \log(p), \tag{3.5}$$

where w_{s} is the segmentation weight. For our experiments we us a weight of 15, as this is the default value from PointRCNN.

For all foreground points, we formulate the RPN regression loss as

$$\begin{aligned} \mathcal{L}_{\text{RPN}} = \mathcal{L}_{\text{CE}}(\theta_{bin}, \theta^*_{bin}) + \\ \mathcal{L}_1(\theta_{res}, \theta^*_{res}) + \\ \sum_{p\in\{x,y,z\}} \mathcal{L}_1(p, p^*) + \sum_{q\in\{h,w,l\}} \mathcal{L}_1(q_{res}, q^*_{res}), \end{aligned} \tag{3.6}$$

where $\mathcal{L}_1$ refers to the smooth L1 loss, q_{res} are the predicted size residuals, q^*_{res} are the groundtruth size residuals, $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss, θ_{bin} is the predicted orientation bin, and θ^*_{bin} is the groundtruth orientation bin. We predict the orientation θ via first estimating a θ_{bin} and then estimating an offset θ_{res} inside the respective bin.

3.4.2 3D Box Refinement Network

The refinement and classification part of PointRCNN takes all foreground point box proposals from the first stage as an input. An overview of all steps included can be found in Table 3.5.

Before starting, we merge the output of the RPN head with the original point cloud and extracted features. Similar to NMS, we perform RoI pooling to remove redundant box proposals. Next, we perform an inside-outside test for every point from the input point cloud with each box proposal. If inside, we keep the point and the extracted feature for refining the box. This way, we can encode local features into the 3D box refinement network.

Box Refinement Network			
Block		Details	Output channels
Merge RPN outputs			128
Downsampling	3 x SA module		512
Confidence Prediction	2 x Conv + ReLU	1 x 1	256
	Conv	1 x 1	1
3D Bounding Box Refinement	2 x Conv + ReLU	1 x 1	256
	Conv	1 x 1	7

Table 3.5: Box refinement network structure.

After merging, we apply a similar approach as before by downsampling using 3 set abstraction modules. The final prediction happens via two separate heads. Each of them first applies two convolutions with ReLU activation. As before, the output channels are pre-defined for confidence prediction and 3D box prediction. As a last step, non maximum suppression (NMS) strains out overlapping boxes to come up with the ultimate bounding box location and dimension.

For refining the boxes, we use a similar loss as in Eq. 3.6. This regression loss is responsible to find the correct localization, size and orientation. Additionally, we add a loss to optimize the correct class for individual objects. We formulate it as

$$\mathcal{L}_{\text{confidence}} = \mathcal{L}_{\text{CE}}(p, p^*). \tag{3.7}$$

$\mathcal{L}_{\text{CE}}$ is the cross entropy loss which acts as a target for the predicted confidence.

3.5 Chapter Summary

This chapter started with the motivation and reasons for our proposed 3D object detection architecture from stereo input images including a description of the whole algorithm. Next, we explained the details of the stereo estimation module and the transformation from depth maps to point clouds. We finish with the in-depth explanation of the applied Point-RCNN 3D object detector.

Chapter 4

Integration of Object Masks

Contents

This sections starts with an analysis on how segmentation masks can help to estimate 3D bounding boxes. Further, we describe how to generate these object masks and how to integrate them in the Pseudo-LiDAR framework. Finally, we discuss methods to optimize class imbalances in the dataset used for our experiments.

4.1 How Can Object Masks Help?

In the previous sections we described that there is a direct dependency between stereo estimation accuracy and the subsequent 3D object detection accuracy. Also small errors in the stereo estimation can result in huge differences during 3D box detection. As described in Section 2, we noticed that deep stereo networks without modifications advocate near and big objects for several reasons.

One of them is the inverse relation between disparity and depth values. We already solved this problem by adjusting the default PSMNet network to internally estimate depth values.

Another reason is the class imbalance in the KITTI dataset. Wang et al. [12] analyzed that 90 % of all pixel in this dataset refer to background pixels. To overcome this problem, we will issue a loss modification to balance out inequal classes.

Pon et al. [67] showed that three-dimensional representation acquired from depth images can contain false predictions at object edges. To address this problem, they present object-centric stereo matching. It only estimates the disparity value for foreground objects using segmentation masks. After transforming, they only cover object point clouds for further processing.

As discussed in the previous section, we apply PointRCNN as our 3D detector in the Pseudo-LiDAR framework. Besides proposing regions of interest, the RPN module also segments between foreground and background object. Originally, the object masks for training the segmentation classifier are given by the groundtruth 3D bounding boxes. Naturally, these object masks are not precise enough to exactly distinct between foreground and background.

This leads us to the idea of including external segmentation masks into the RPN module of Point-RCNN. For our approach, we adapt the network to use object masks provided by an external segmentation network. The details and additional benefits of this method are described in the next section.

4.2 Integration of Object Masks

As stated in Section 3.3, PointRCNN segments foreground objects from input point clouds before predicting final 3D boxes. Originally, it uses proximate masks given by the groundtruth 3D bounding boxes. We modify the point cloud detector to use external objects masks predicted by an external provider instead.

Figure 4.1 shows an overview of our modified 3D bounding box detector. Besides the converted point cloud from the depth estimation, we also include a front-view object mask of the same scene. This helps to make more accurate point-wise box proposals as non-object points are now excluded. Additionally, the reduced amount of potential target points increases the overall performance of the point cloud proposal component.

Since the stereo module already has pixel-level information about objects via the disparity matching, we do not include the segmentation masks there.

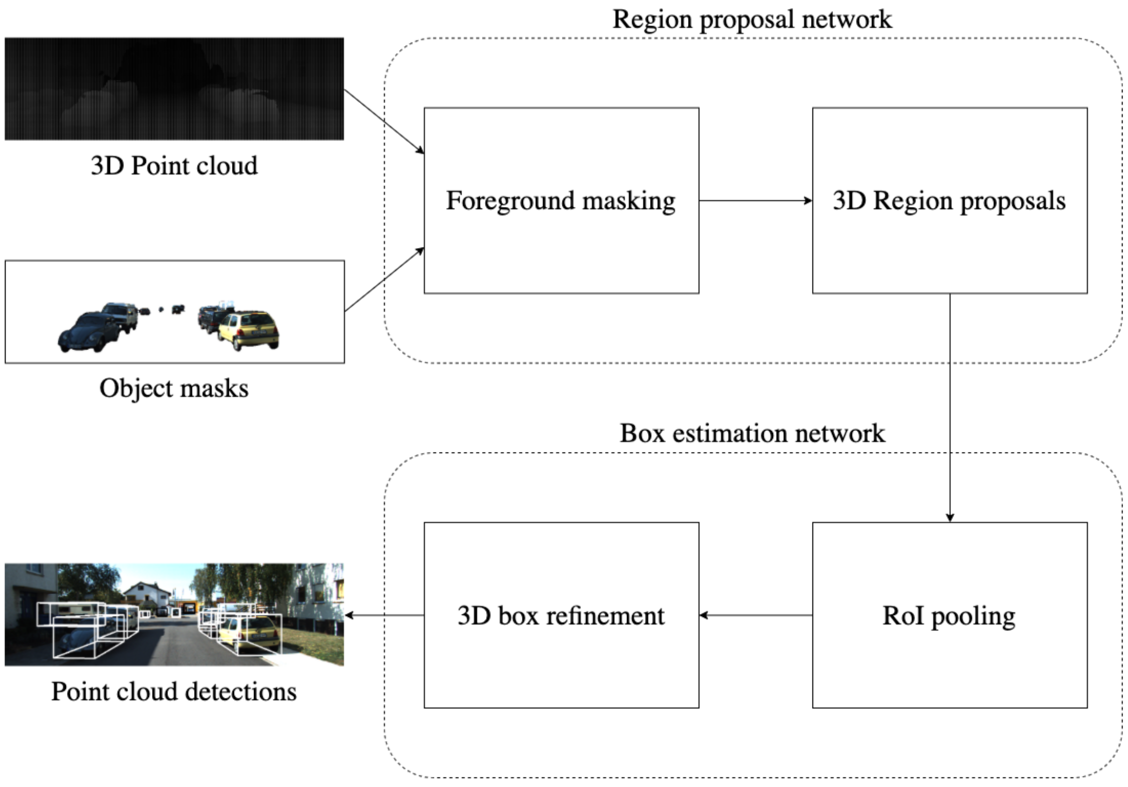

Figure 4.1: Visualization of the modified PointRCNN architecture with external object masks.

For estimating object masks, we compared different image segmentation networks in Chapter 2. Due the high accuracy and low memory usage we chose VideoPropLabelRelax [56] as a base network for integration. It is an encoder-decoder architecture with spatial pooling layers based on DeepLabV3Plus [68] by Chen et al. Combining the multi-scale encoder with the sharp low-level upsampling of the decoder results in high quality segmentation masks.

Table 4.1 shows the structure of the implemented encoder network. It starts with extracting features from the input image using a Resnet50 [25] backbone network. Next, we apply a Atrous Spatial Pyramid Pooling (ASPP) module. It consists of several parallel atrous convolution layers with different rates. Atrous convolutions allow to explicitly control the resolution by adjusting the kernel's point of view. This way, the ASPP module can probe convolutional layers at multiple scales. The encoder ends with concatenating all ASPP module outputs. After that, a final 1x1 convolutional layer with ReLU activation is applied. The output feature map contains 256 channels and encodes rich semantic information.

Encoder			
Block		Details	Output channels
Backbone	Resnet50	1 x 1	2048
ASPP Module	Conv + ReLU	1 x 1	256
	Conv + ReLU	3 x 3, Atrous: 6	256
	Conv + ReLU	3 x 3, Atrous: 12	256
	Conv + ReLU	3 x 3, Atrous: 18	256
	AvgPool + Conv + ReLU + Upsample	1 x 1	256
Head	Concat all ASPP outputs		1280
	Conv + ReLU	1 x 1	256

Table 4.1: Image segmentation encoder.

Once completing the encoder network, a decoder network as described in Table 4.2 follow. We first apply a 1 x 1 convolution on the backbone outputs to cut down the amount of channels. The high number of channels could outweigh the rich semantics of the encoder output. In parallel we bilinearly upsample the ASPP outbut by a factor of 4. After that, we merge the reduced low level backbone features with the upsampled ASPP output features.

Decoder			
Block		Details	Output channels
Reduce low level features from backbone	Conv + ReLU	1 x 1	48
Concat	Reduced low level feature + ASPP output		304
Classifier	Conv + ReLU	3 x 3	256
	Conv	1 x 1	No. classes

Table 4.2: Image segmentation decoder.

Once merged, we apply a 3x3 convolutional layer to refine the features. During the final classification, the amount of channels matches the number of classes to predict. The output image gets bilinearly upsampled by a factor of 4.

Figure 4.2 shows a comparison of cropped target objects using the different methods. We can clearly see the difference of precision especially in the border areas of objects. Since especially border areas are a source for errors, we expect to support the point cloud detector during region proposal and thus increase bounding box accuracy.

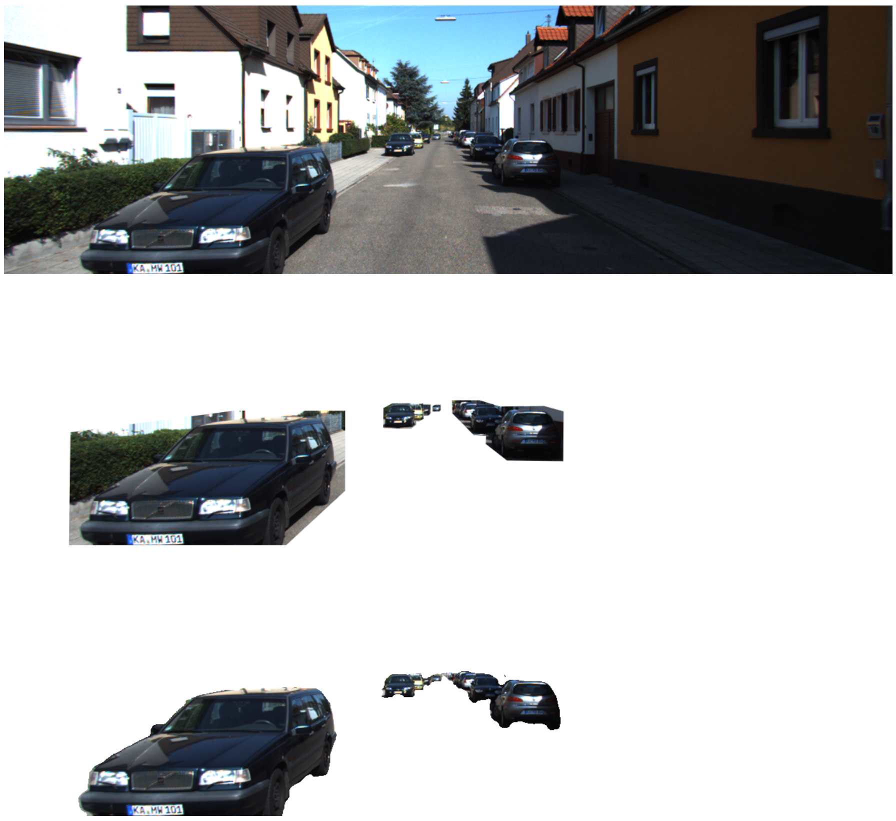

Figure 4.2: Top: Original image.
Middle: Segmented target objects using groundtruth 3D bounding boxes.
Bottom: Segmented target objects using VideoPropLabelRelax [56] masks

Due to the focus of VideoPropLabelRelax [56] on semantic segmentation, we expect the network to estimate better object masks than the groundtruth masks. However, there may be cases where VideoPropLabelRelax produces inferior results or misses objects. Therefore, we use the groundtruth masks as a backup during training. Whenever the predicted masks exceed the groundtruth masks, we crop it to fit into the groundtruth mask. Additionally, we dismiss predicted masks when there is no matching groundtruth objects with a specified overlap.

4.2.1 Focal loss

According to Abdou et al. [69], imbalanced distribution of classes in point cloud input data can lead to wrongly classified objects. Therefore, they propose a weighted loss function to overcome discrimination of under-represented classes. Opposed to previous methods, they do not down-weight objects of over-represented classes.

Since there is an imbalance among front- and background objects in the chosen dataset, we adapt the segmentation head of the point cloud detector to use Focal loss [21]. This loss function is a modification of the standard cross entropy function by down-weighting the loss allocated to easily-classified objects.

We reformulate the RPN segmentation loss from Eq. 3.5 as

$$\mathcal{L}_{\text{Segmentation}} = -w_{\text{s}}(1-p)^{\gamma}\,\log(p), \tag{4.1}$$

where the hyperparameters γ and w_s are used to tune the weight of different samples. This gives us more control for prioritizing classes than using the standard cross entropy loss.

For all our experiments, we set the hyperparameters to their default settings $\gamma = 2$ and $w_s = 0.25$ as described in the paper.

4.3 Chapter Summary

This section showed how external object masks can be integrated in our framework during training. We described the beneficial effects of modifying the default segmentation head, allowing detailed bounding box proposals from the boarder areas of objects. Finally, we explained the class imbalance problem in the KITTI dataset and how to overcome it using a modified loss function.

Chapter 5

End-to-End Learning

Contents

In this section, we first investigate into improvements of the decoupled design of the Pseudo-LiDAR framework. After that, we describe modifications to train the model for a global optimization function. We conclude with the definition of an unified loss function to optimize for a single objective.

5.1 Power of End-to-End Learning

The default pipeline is based on a modular design consisting of two independent components which need to be trained separately. However, this decoupled approach of training two networks may not be the best fit for the learning objective of 3D object detection. For instance, standalone depth estimation networks accurately predict near and over-represented objects while producing more errors for far-away objects [14].

Therefore, we investigate into end-to-end neural networks. End-to-end learning is defined as training a possibly complex system by applying gradient-based techniques to the model as a whole. They have the benefit that single components can interact and jointly minimize a global learning loss. This means that there is no need to optimize for any auxiliary task unrelated to the main objective. Besides, training a system in a holistic manner results in conceptual beauty.

End-to-end learning models yield powerful results in different domains [70]. Bojarski et al. [15] have shown that autonomous vehicles can operate in diverse conditions with sparse training data using a unified loss function. They argue that the non-decomposition of their self-driving car optimization will eventually lead to better performance and smaller systems.

On the contrary, unifying a complex system offers potential for inefficiencies. As discussed by Shalev et al. [71], the traditional limitations of gradient-based learning models can occur. Especially the chance of vanishing gradients and slow convergence increases by coupling different networks into one system.

Additionally, interacting components can restrict each other's learning progress which may result in a complete breakdown of training. Therefore, the stacking of multiple networks with too complex learning tasks is not recommended [72].

With all the merits and limitations in mind, we propose end-to-end learning for both the depth and object detection module in our Pseudo-LiDAR pipeline. The details will be described in the next section.

5.2 Joint Training

As described earlier, Pseudo-LiDAR enables to replace single components by being designed in a modular way. This allows to integrate new state-of-the-art models nearly immediately. However, this modularity is not suited to optimize the ulimate learning objective of object detection. There is no need to train modules on an auxiliary objective.

Compared to the explicit decomposition of components, such as stereo estimation and point cloud detection, our end-to-end system optimizes all processing steps simultaneously. This will lead to better performance and smaller system, which has been proven in various domains [73]. Better accuracy will result because the internal components self-optimize to maximize the overall learning objective of 3D object detection.

Figure 5.1 shows an overview of the end-to-end learning system.

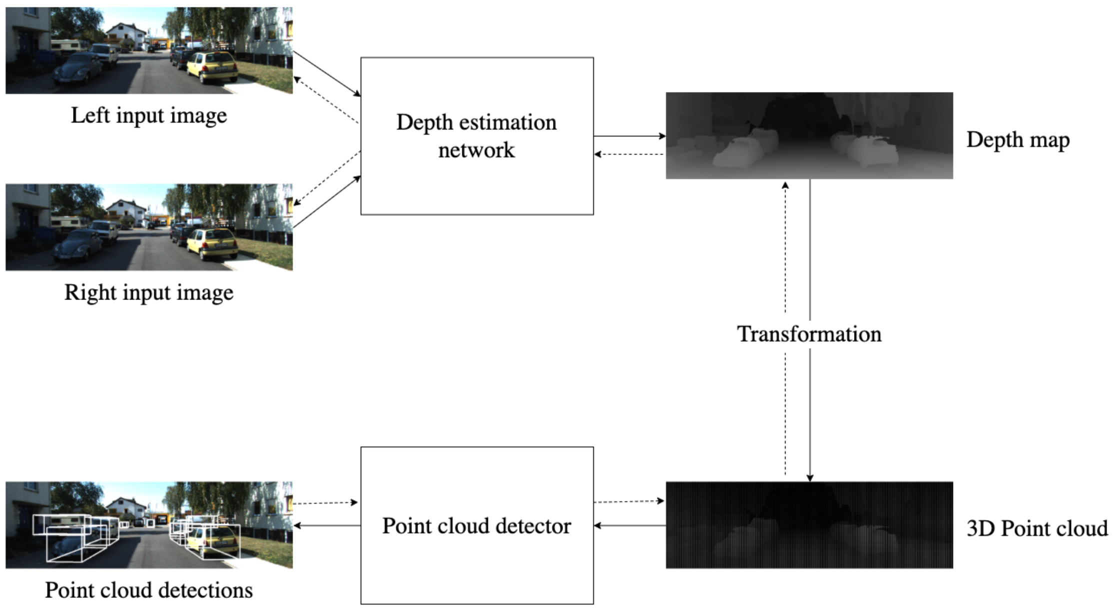

FIGURE 5.1: Joint training.

To make end-to-end learning possible, we need to connect both components to permit the detection gradients to back-propagate to the stereo estimation module. This includes the upscaling of the traditional backpropagation algorithm [74] for our complex architecture. In addition, the individual components need to be differentiable in terms of all adjustable parameters. Furthermore, we optimize our model to use a global, supervised loss function which will be described in the next sections.

The architecture shows the united design of depth estimation and point cloud object detection. In between, we have the transformation from the depth map to a point cloud. We have already described this transformation in the previous chapter. Since the transformation includes the subsampling of points and thus outputs a sparse representation of the original scene, we need to re-prioritize the loss for the depth network in our experiments.

As we already know, the backpropagation algorithmn for calculating gradients consists of forward and backward passes. The forward pass multiplies the input by weights and passes the results forward to the next layers. After calculating the error, we can update weights using backward passes. Compared to our default network architecture shown in Figure 3.1, we can clearly see the dotted-lined arrows in Figure 5.1. The flow of these arrows describes the backward passes of the losses.

We argue that this back-flowing gradient works similar to a soft-attention [75] to the depth estimation network. This way the final error from the point cloud detector can guide the depth network where to put focus and improve. Often, we observed that the errors occur in and around objects to detect. When using the external masks from the previous chapter, we can softly guide the depth network in border regions more precise than without external masks.

As we know from the previous section, deep networks offer the chance of vanishing gradients. If the gradient is too small, it will prevent changing the value of weights. The reason for this problem is the gradient computation by the chain rule.

We try to solve this problem using several approaches:

- First, we heavily use residual layers in both the depth estimation and point cloud estimation network. Using these skip connections permits the gradient to proceed into deeper structures of the model.
- Additionally, we employ the sigmoid-like non-linearity ReLU activation in our network. However, ReLU does not feed deeper parts of the network with additional information.

5.3 Loss Functions

The proposed end-to-end model described in the previous section optimizes a global learning function.

We define the end-to-end loss as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{3D}}, \tag{5.1}$$

where $\mathcal{L}_{\text{depth}}$ denotes the depth loss as defined in Eq. 3.2 and $\mathcal{L}_{\text{3D}}$ describes the point cloud detector loss from Chapter 5.

5.3.1 Weighted Global Loss Function

Since we subsample the converted depth map to better suit existing 3D detectors, it outputs a sparse point cloud. Therefore, we need to introduce weights to distinct between depth and point cloud gradients.

To conclude, we define the global objective function as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{depth}} + w_{\text{3D}}\mathcal{L}_{\text{3D}}, \tag{5.2}$$

where w_{3D} denotes the point cloud detector weight.

For our experiments, we aim to find the best trade-off between weighting the depth and point cloud detector loss. Due the subsampling of depth maps to point cloud, we anticipate to start with an down-weighted 3D loss for a good balance.

Optimizing for this global learning goal, we anticipate the network to learn the optimal internal weights of the architecture in an end-to-end way. The model should improve in terms of accuracy compared to the modular design consisting of two independent components which need to be trained separately.

5.4 Chapter Summary

In this section, we discussed methods to improve information flow between the individual components via end-to-end learning. Additionally, we described details for implementing end-to-end learning in the Pseudo-LiDAR framework via a global loss function using backward passes.

Chapter 6

Experimental Evaluation

Contents

This chapter begins with a explanation of the dataset and evaluation metric used for the performed experiments. Next, we explore the details of the implementation of the proposed model. We finalize this chapter with results obtained from experiments and a comparison with other methods.

6.1 Dataset Description

We evaluate the proposed network on the openly obtainable KITTI dataset [22]. It consists of 7,481 training and 7,518 images, where every instance includes two pictures from a stereo camera setup and a LiDAR point cloud from a Velodyne laser sensor [76]. In total, it comprises 80,256 labeled objects. All images were taken of streets in daylight around the city of Karlsruhe. A typical scene of the dataset can be seen in Figure 6.1.

We also investigated into other public datasets for our experiments. The Waymo Open Dataset [77] features 3,000 driving scenes totalling 16.7 hours of video data. However, it does not offer stereo camera input.

Another popular dataset named nuScenes [78] comprises over 1,000 driving scenes for several object detection challenges. Even so, it also only exposes LiDAR and mono-camera sensor data.

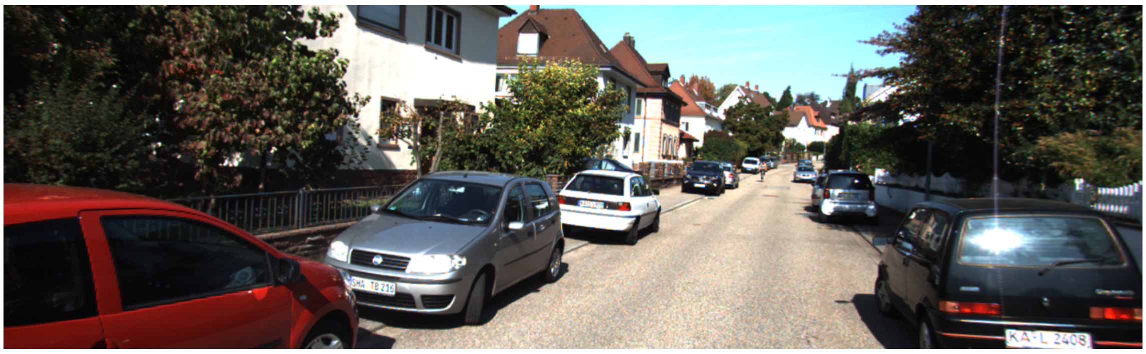

FIGURE 6.1: Example image from the KITTI dataset. Image taken from [22].

Table 6.1 describes the KITTI annotation format used for our experiments. Note that the provided 2D bounding box is not processed by the model and therefore not necessary. The score field is only necessary for predicted labels and therefore not available during training. Additionally, we only perform experiments on the 'Car' class and ignore any other classes.

6.2 Evaluation Criteria

To evaluate the 3D detection performance of the presented architecture, we use the PASCAL criteria [79] for a fair comparison with other methods.

Values	Name	Description
1	type	Describes the type of object: 'Car', 'Van', 'Truck', 'Pedestrian', 'Person_sitting', 'Cyclist', 'Tram', 'Misc' or 'DontCare'
1	truncated	Float from 0 (non-truncated) to 1 (truncated), where truncated refers to the object leaving image boundaries
1	occluded	Integer (0,1,2,3) indicating occlusion state: 0 = fully visible, 1 = partly occluded, 2 = largely occluded, 3 = unknown
1	alpha	Observation angle of object, ranging [-pi..pi]
4	bounding box	2D bounding box of object in the image (0-based index): contains left, top, right, bottom pixel coordinates
3	dimensions	3D object dimensions: height, width, length (in meters)
3	locations	3D object location x,y,z in camera coordinates (in meters)
1	rotation_y	Rotation ry around Y-axis in camera coordinates [-pi..pi]
1	score	Only for results: Float, indicating confidence in detection

TABLE 6.1: KITTI annotation format as specified by [22].

6.2.1 Intersection over Union

The prediction of bounding boxes in 3D space require a metric to measure how accurate these outputs are. Therefore, the intersection over union evaluates the magnitude of overlap between two objects as:

$$IoU(A,B) = \frac{A \cap B}{A \cup B}. \tag{6.1}$$

In the described datasets, an 3D bounding box IoU overlap of 70 % is necessary for all objects to be counted as correct.

6.2.2 Average Precision

Basic concepts for calculating objected detection metrics are:

- True Positive (TP): A correct detection, where the IoU is above a specific threshold.
- False Positive (FP): A wrong detection, where the IoU is below a specific threshold.
- False Negative (FN): A groundtruth bounding box which gets not detected.

Precision is defined as the probability of correct positive predictions:

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{All\,Detections}. \tag{6.2}$$

Recall is defined as the probability of groundtruth objects being correctly detected:

$$Precision = \frac{TP}{TP + FN} = \frac{TP}{All\,Groundtruth\,Boxes}. \tag{6.3}$$

For better comparison, a single numerical metric called average precision (AP) gets calculated. It averages the precision by interpolating across 40 spaced recall values from 0 to 1.

6.3 Implementation Details

We now proceed with the training details for all conducted experiments. We have implemented all models using the PyTorch [80] framework. It is an open-source machine learning library optimized for achieving state of the art results in research without sacrificing flexibility.

6.3.1 Data Augmentation

For all experiments we apply input augmentation using horizontal flip, scaling by a factor in [0.90, 1.10] in addition to rotating the Y axis of [-15, 15] degrees.

6.3.2 Semantic Segmentation

We use a starting learning rate of 0.002 and train the network for 90 epochs. As already described, we use a pretrained Resnet50 [25] feature extractor as a backbone network.

6.3.3 Stereo Depth Estimation

Before feeding the images into the network, we apply color normalization and randomly crop them according to [25].

We pre-train the stereo network for 200 epochs with a 0.002 learning rate. For training, we transform the groundtruth point clouds onto the images to obtain the depth. Therefore,

6.3.4 3D Object Detection

As an input, we subsample every point cloud to a defined amount of 16,384 points. This amount of points refers to the mean of points in the dataset per scene.

For 3D detection, we pre-train the RPN network for 200 epochs using a 0.002 learning rate in addition to the batch size of 8. We keep 300 proposals after NMS has been applied. This way, we can ensure high-quality proposals for further refinement.

We pre-train the refinement network for 250 epochs with a learning rate of 0.002 in addition to a batch size defined as 8.

6.3.5 End-to-End Training

We apply stochastic gradient descent for jointly optimizing the global objective function. We set the weight decay and momentum to 0.0004 and 0.95.

For end-to-end optimization, we set both modules of the pipeline as trainable. This allows the gradients of the point cloud detector to flow back into the depth estimation network. We experiment with different loss weights to find the best relationship between point cloud and depth gradients. We report the weight and parameter settings in the different experiment results.

For the final fine-tuning, we train the network for 15 epochs with the batch size specified to 4. For the beginning the learning rate is set to 0.0002.

6.4 Experimental Results

This section presents the quantitative results of training our end-to-end 3D object detection architecture using the methods described in the previous chapters. The aim of the experiments is the demonstration of improvement by integrating objects masks and joint training of separate pipeline modules.

For the validation of this idea, we measured the mean IoU with a minimum overlap of 70 of our modified model and compare it with a reference. As a baseline model, we used the public implementation of Pseudo-LiDAR++ [14]. Both the results from the baseline implementation and our implementation can be found in Table 6.2.

Model	Input	IoU (0.7)		
		Easy	Moderate	Hard
Baseline	Stereo	58.03	42.48	36.65
Baseline + Objects masks	Stereo	59.79	43.85	37.91
Baseline + Objects masks + E2E Learning	Stereo	61.43	45.69	39.48

Table 6.2: Obtainend results of the proposed modifications compared with the baseline model on the KITTI validation dataset.

The outcomes of the experiments in combination with the qualitative results in Figure 6.2 confirm our intention of combining the separate components to optimize a common learning objective. We also notice that end-to-end learning mainly improves the 'Hard' category on KITTI compared to the baseline implementation. This may be an outcome of the back-propagated loss into the depth estimation module, making it easier to propose far-away and occluded objects.

We also see improvements in all categories by adding object masks and a segmentation head during training of the network. Naturally, generated object masks specifically for 3D object detection are cost-intensive. However, often these masks are already provided by any other module of an autonomous vehicle vision system and can therefore be integrated without any additional cost.

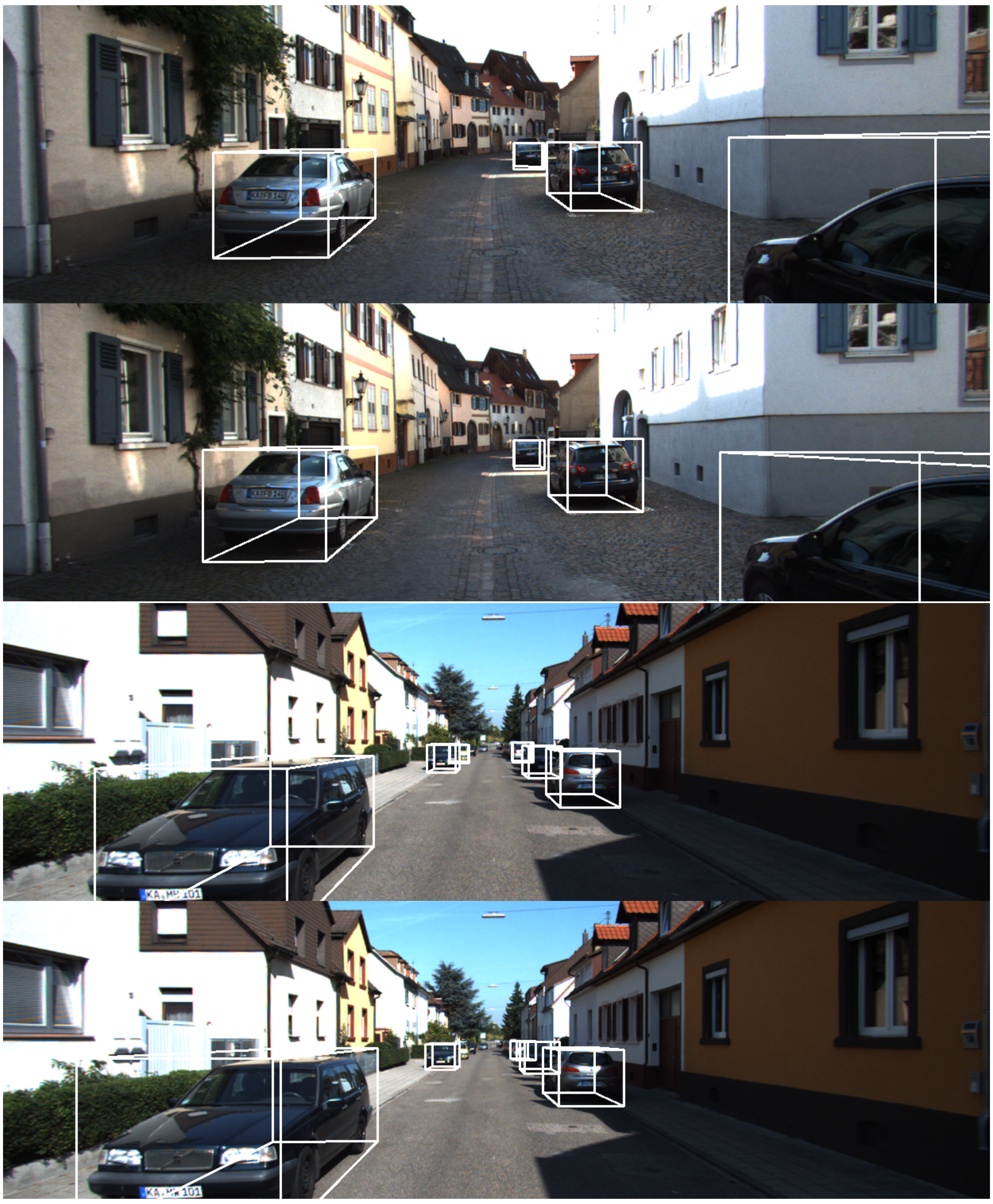

Figure 6.2: Qualitative results obtained from the experiments. The top row of each image pair shows the groundtruth bounding boxes, the bottom row presents the detected bounding boxes from our model.

Figure 6.3 shows a comparison of 3D detection results on the middle and right side and the groundtruth annotation on the left side. In the middle column we see the detections using external object masks during training. As already described, 3D bounding boxes need an IoU overlap of 70 % to be counted as correct.

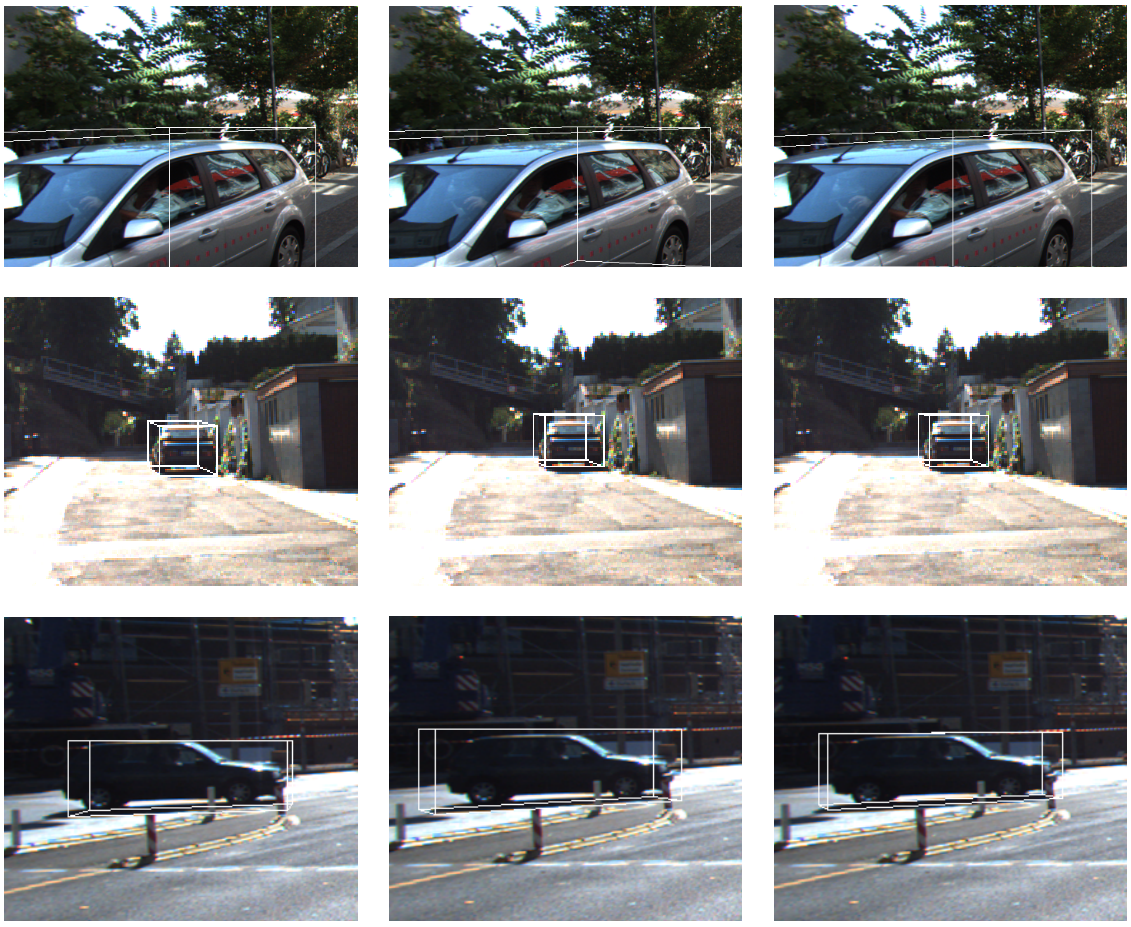

FIGURE 6.3: Comparison of qualitative results with different modifications applied. From left to right: groundtruth annotation, results from model with integrated object masks, results from end-to-end model with integrated object masks

In the first row, we can clearly see that the bounding box in the back of the car becomes more narrow using our model with all modifications. For smaller and further away object as shown in the second row, there is not a huge difference in bounding box location and size. The last row also presents that end-to-end learning helps for tine-tuning the bounding box size for narrow cars. Overall, we see how using external masks and end-to-end learning improves the size of predicting bounding boxes.

For an better overview, we compare our implementation on the KITTI 3D dataset. The results are presented in Table 6.3. We observe that our model performs better than traditional stereo methods like Stereo-RCNN. However, memory-intensive approaches like DSGN and CG-Stereo outperform all other methods based on stereo input. These methods accurately estimate depth maps by internally applying more high computational operations as shown in Table 2.1 than we use our architecture. Additionally, there is still a huge gap between approaches based on stereo input and approaches based on LiDAR-only input like Voxel R-CNN.

Model	Input	IoU (0.7)		
		Easy	Moderate	Hard
SMOKE [81]	Mono	14.76	12.85	11.50
Stereo-RCNN [9]	Stereo	54.1	36.7	31.1
PL++ [14]	Stereo	58.03	42.48	36.65
Ours	Stereo	61.43	45.69	39.48
DSGN [48]	Stereo	72.31	54.27	47.71
CG-Stereo [13]	Stereo	76.17	57.82	54.63
PointRCNN [43]	LiDAR	89.2	78.9	77.9
Voxel R-CNN [82]	LiDAR	89.41	84.52	78.93

Table 6.3: Obtained results our model compared with other methods on the KITTI validation dataset.

Since the KITTI dataset shows a class imbalance between foreground and background objects, we introduced a loss modification in Eq. 4.1. We defined a focusing parameter γ for individually down-weighting easy examples. For our experiments, we smoothly adjust the parameter. In Table 6.4 we observe that setting $\gamma = 3$ yields the best object detection results.

γ	IoU@0.7 (Moderate)
1.0	42.48
2.0	44.62
3.0	45.69
4.0	44.02

Table 6.4: Obtained results of our model with varying focal detector loss weight.

In Eq. 5.2, we use parameters to individually weigh the balance between depth and point cloud detector loss. We conducted the experiments with different settings to find the best parameter setup. As shown in Table 6.5, setting the parameter $w_{3\text{D}} = 0.05$ turned out to work well on the validation dataset.

$w_{3\text{D}}$	IoU@0.7 (Moderate)
0.01	43.48
0.05	45.69
0.1	41.35

TABLE 6.5: Obtained results of our model with varying 3D detector loss weight.

Figure 6.4 show the tradeoff between precision and recall at a threshold of 0.7. The figure compares the true positive rate and the positive predictive value for our model. For generating this curve, all recall results get approximated to a linear function. From the figure we can observe that compared to Pseudo-Lidar++ [14] our model does not show any outstanding drops in the curve.

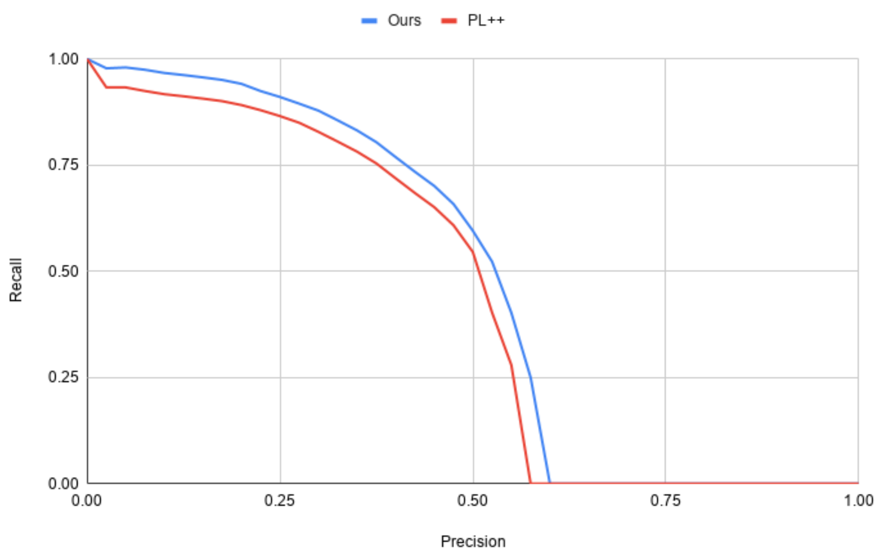

FIGURE 6.4: Precision-recall curve on the KITTI dataset moderate level.

We compared different semantic segmentation networks in Chapter 2. Due the reported results on the KITTI dataset, we chose VideoPropLabelRelax [56] for our implementation. However, the object masks can be generated by any arbitrary solution. For our experiments, we visually compared the results of SegStereo [54] to our selected network. The results can be found in Figure 6.5. We can clearly see the higher error of SegStereo [54], especially in the boundary areas of cars.

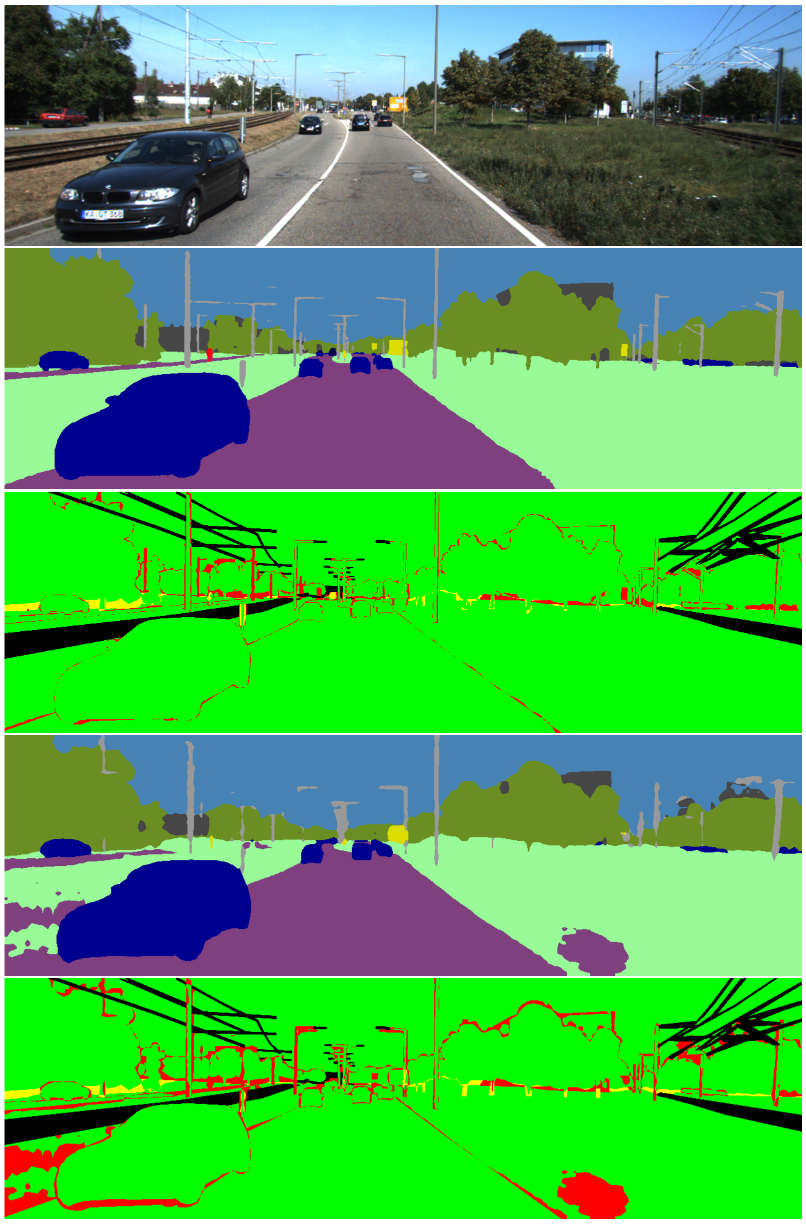

FIGURE 6.5: Visual comparison of different semantic segmentation methods. From top to bottom: image, color-coded result from VideoPropLabelRelax [56], error from VideoPropLabelRelax [56], color-coded result from SegStereo [54] and error from SegStereo[54]

6.5 Runtime

Given its importance, a lot of research effort has been made to push 3D object detection accuracy. Most current methods are unable to obtain real-time speeds and require plenty of memory footprint in the training phase. However, autonomous vehicles often have limited hardware resources, processor power and memory.

Having those hardware restrictions in mind, we are aiming to find a trade-off between detection accuracy and performance. To validate this idea, we measure the runtime of our 3D object detection implementation. First, we identify the used hardware specifications. For all our experiments, we utilized a GeForce RTX 2070 GPU with 8 GB memory.

All models were implemented in the PyTorch [80] framework. It supports designing network architectures in Python, while remaining structured and supporting hardware accelerators such as GPUs.

We analyze the performance by measuring the runtime to infer the 3D bounding boxes of a stereo input image pair. On the GeForce GPU, the end-to-end model requires 0.54 seconds for a single input pair with prepared object masks.

Since object detection is often applied with complimentary tasks, we assume that the segmentation masks are already given. The segmentation of objects takes up 0.17 seconds per input image using the reference implementation of VideoPropLabelRelax [56]. The total runtime can be reduced by running several processes in parallel using more GPUs.

6.6 Chapter Summary

The final chapter summarizes the results obtained of the experimental evaluation. It started with a description of the used dataset and the metrics for comparison with other methods.

Further, we described the details of our implementation. Due to different modification each component of the whole pipeline is capable of optimizing the learning objective of 3D object detection.

Finally, we presented our experiment results and showed that our presented approach increases the accuracy on public datasets compared with traditional methods. We complete with a runtime evaluation of our model.

Chapter 7

Conclusions and Future Work

Contents

The final chapter will summarize the aspects of this thesis and consider topics for future studies.

7.1 Conclusions

Our thesis shows that convolutional networks are able to successfully detect and localize objects in road scenes using stereo input images. Based on the existing Pseudo-LiDAR framework [14], we proposed modifications to the network architecture to increase the object detection accuracy. Using segmentation masks from an external network, we could eliminate wrong bounding box proposals especially in border areas of objects. Additionally, we implemented end-to-end learning to optimize for a global learning function. We assessed our approach on a public 3D dataset [22] and showed improvements compared to the chosen baseline.

By default, the PointRCNN [43] 3D object detector of the Pseudo-LiDAR framework distinguishes between foreground and background objects for proposing 3D bounding boxes. However, the network uses the groundtruth 3D boxes for segmenting objects which are imprecise in border areas. To solve this issue, we leverage foreground object masks predicted by the VideoPropLabelRelax [56] network. This way we can improve the proposal generator of the two-stage network by precisely dismissing any out-of-object bounding box proposals. Further, analysis show that the KITTI dataset contains class imbalances. Therefore, we modified the model to use Focal loss [58] for optimization. We saw that these modifications are beneficial for the task of 3D object detection.

Current stereo object detection pipelines are built on a modular design consisting of two independent components which need to be trained separately. According to other studies this decoupled approach of training two networks may not be the best fit for the final learning objective of 3D object detection. Therefore, we adapt the baseline architecture to train in an end-to-end fashion. Our modifications enable both components of the Pseudo-LiDAR framework to interact with each other and collectively minimize a global learning loss. Our evaluations proved that training jointly improves the object detection accuracy.

7.2 Future Work

As indicated by the reported results, there is still a gap among camera-based 3d object detectors and LiDAR-based 3d object detectors. However, we showed that image-based methods are improving continuously. There are multiple directions in which our work could be improved.

To generate the synthetic point clouds, we applied a pyramid-based stereo estimation network. Future work could investigate more complex stereo network architecture designs. Neural Architecture Search (NAS) can achieve promising results in the area of stereo estimation, as shown by Cheng et al. [83].

In our framework, we first estimate a depth map based on stereo input images and then transform this map into a point cloud. While being fast and conceptually simple, this method does not directly encode 3D geometric structures and thus offers the chance of loosing information. Therefore, it would be compelling to think about better suited representations for processing 3D data from stereo input. For example, DSGN [48] and the work of Bewley et al.[84] proposed ideas for other representation.

In the case of 3D detection, our transformed depth map gets downsampled before getting fed into the point cloud detector. Future work could improve current point cloud detectors for fusion-based input. This way, the complementary strengths of Pseudo-LiDAR, LiDAR and input images may result in higher detection accuracy.

As shown in the thesis, current stereo object detection pipelines including our model do not operate in real-time. With the rising of autonomous vehicles and drones, future work could try to lower the inference time of stereo object detection for concurrent usage. With a view on edge computing, exploiting simpler network architectures could be advantageous for decreasing GPU usage. Königshof et. al [85] showed promising results in terms of inference time using image-based detectors.

Bibliography

[1] Henry Schneiderman and Takeo Kanade. A statistical method for 3d object detection applied to faces and cars. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 1, pages 746–751. IEEE, 2000.

[2] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[3] Michael Himmelsbach, Andre Mueller, Thorsten Lüttel, and Hans-Joachim Wünsche. Lidar-based 3d object perception. In *Proceedings of 1st international workshop on cognition for technical systems*, volume 1, 2008.

[4] Matti Kutila, Pasi Pyykönen, Werner Ritter, Oliver Sawade, and Bernd Schäufele. Automotive lidar sensor development scenarios for harsh weather conditions. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 265–270. IEEE, 2016.

[5] James Davis, Ravi Ramamoorthi, and Szymon Rusinkiewicz. Spacetime stereo: A unifying framework for depth from triangulation. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–359. IEEE, 2003.

[6] Jeffrey S Deems, Thomas H Painter, and David C Finnegan. Lidar measurement of snow depth: a review. *Journal of Glaciology*, 59(215):467–479, 2013.

[7] David Marr and Tomaso Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283–287, 1976.

[8] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432, 2015.

[9] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo r-cnn based 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7644–7652, 2019.

[10] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 328–335, 2014.

[11] 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.

[12] 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.

[13] Chengyao Li, Jason Ku, and Steven L Waslander. Confidence guided stereo 3d object detection with split depth estimation. *arXiv preprint arXiv:2003.05505*, 2020.

[14] 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. In *International Conference on Learning Representations (ICLR)*, 2020.

[15] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[16] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 508–515. IEEE, 2001.

[17] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.

[18] Rostam Affendi Hamzah and Haidi Ibrahim. Literature survey on stereo vision disparity map algorithms. *Journal of Sensors*, 2016, 2016.

[19] Jure Žbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *The journal of machine learning research*, 17(1): 2287–2318, 2016.

[20] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016.

[21] 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.

[22] 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.

[23] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[24] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[26] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[28] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[29] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[30] Jeong-ah Kim, Ju-Yeong Sung, and Se-ho Park. Comparison of faster-rcnn, yolo, and ssd for real-time vehicle type recognition. In *2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–4. IEEE, 2020.

[31] Minh-Tan Pham, Luc Courtrai, Chloé Friguet, Sébastien Lefèvre, and Alexandre Baussard. Yolo-fine: One-stage detector of small objects under various backgrounds in remote sensing images. *Remote Sensing*, 12(15):2501, 2020.

[32] 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*, pages 9627–9636, 2019.

[33] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[34] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[35] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[36] Biao Leng, Shuang Guo, Xiangyang Zhang, and Zhang Xiong. 3d object retrieval with stacked local convolutional autoencoder. *Signal Processing*, 112:119–128, 2015.

[37] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.

[38] Zining Wang, Wei Zhan, and Masayoshi Tomizuka. Fusing bird's eye view lidar point cloud and front view camera image for 3d object detection. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–6. IEEE, 2018.

[39] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[40] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, pages 965–975, 2019.

[41] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[42] Liuhao Ge, Yujun Cai, Junwu Weng, and Junsong Yuan. Hand pointnet: 3d hand pose estimation using point sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8417–8426, 2018.

[43] 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.

[44] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[45] 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.

[46] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[47] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[48] 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.

[49] Robert T Collins. A space-sweep approach to true multi-image matching. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 358–363. IEEE, 1996.

[50] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[51] Mahesh Pal. Random forest classifier for remote sensing classification. *International journal of remote sensing*, 26(1):217–222, 2005.

[52] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[53] 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.

[54] Guorun Yang, Hengshuang Zhao, Jianping Shi, Zhidong Deng, and Jiaya Jia. Seg-stereo: Exploiting semantic information for disparity estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 636–651, 2018.

[55] Ivan Kreso, Sinisa Segvic, and Josip Krapac. Ladder-style densenets for semantic segmentation of large natural images. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 238–245, 2017.

[56] Yi Zhu, Karan Sapra, Fitsum A Reda, Kevin J Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8856–8865, 2019.

[57] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836*, 2018.

[58] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[59] Ralf Haeusler and Reinhard Klette. Analysis of kitti data for stereo analysis with stereo confidence measures. In *European Conference on Computer Vision*, pages 158–167. Springer, 2012.

[60] Izzat Izzat and Feng Li. Stereo-image quality and disparity/depth indications, May 12 2015. US Patent 9,030,530.

[61] 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.

[62] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[63] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[64] Jie Liao, Yanping Fu, Qingan Yan, and Chunxia Xiao. Pyramid multi-view stereo with local consistency. In *Computer Graphics Forum*, volume 38, pages 335–346. Wiley Online Library, 2019.

[65] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[66] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[67] Alex D Pon, Jason Ku, Chengyao Li, and Steven L Waslander. Object-centric stereo matching for 3d object detection. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8383–8389. IEEE, 2020.

[68] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[69] Mohammed Abdou, Mahmoud Elkhateeb, Ibrahim Sobh, and Ahmad Elsallab. End-to-end 3d-pointcloud semantic segmentation for autonomous driving. *arXiv preprint arXiv:1906.10964*, 2019.

[70] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[71] Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of gradient-based deep learning. In *International Conference on Machine Learning*, pages 3067–3075. PMLR, 2017.

[72] Tobias Glasmachers. Limits of end-to-end learning. In *Asian Conference on Machine Learning*, pages 17–32. PMLR, 2017.

[73] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[74] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[75] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[76] Craig Glennie and Derek D Lichti. Static calibration and analysis of the velodyne hdl-64e s2 for high accuracy mobile scanning. *Remote Sensing*, 2(6):1610–1624, 2010.

[77] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[78] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[79] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[80] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[81] Zechen Liu, Zizhang Wu, and Roland Tóth. Smoke: single-stage monocular 3d object detection via keypoint estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 996–997, 2020.

[82] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. *arXiv preprint arXiv:2012.15712*, 2020.

[83] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Tom Drummond, Hongdong Li, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. *arXiv preprint arXiv:2010.13501*, 2020.

[84] Alex Bewley, Pei Sun, Thomas Mensink, Dragomir Anguelov, and Cristian Sminchisescu. Range conditioned dilated convolutions for scale invariant 3d object detection. *arXiv preprint arXiv:2005.09927*, 2020.

[85] Hendrik Königshof, Niels Ole Salscheider, and Christoph Stiller. Realtime 3d object detection for automated driving using stereo vision and semantic information. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1405–1410. IEEE, 2019.