

# Object detectie en lokalisatie op basis van stereo-beelden

**Maxim AELTERMAN**

Promotor: Prof. dr. ir. Patrick Vandewalle Masterproef ingediend tot het behalen van  
de graad van master of Science in de  
industriële wetenschappen: Electronica-ICT  
ICT

Academiejaar 2019 - 2020

©Copyright KU Leuven

Zonder voorafgaande schriftelijke toestemming van zowel de promotor(en) als de auteur(s) is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, kan u zich richten tot KU Leuven Technologiecampus De Nayer, Jan De Nayerlaan 5, B-2860 Sint-Katelijne-Waver, +32 15 31 69 44 of via e-mail [iiw.denayer@kuleuven.be](mailto:iiw.denayer@kuleuven.be).

Voorafgaande schriftelijke toestemming van de promotor(en) is eveneens vereist voor het aanwenden van de in deze masterproef beschreven (originele) methoden, producten, schakelingen en programma's voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

# **Voorwoord**

Ik zou graag mijn interne promotor Prof. dr. ir. Patrick Vandewalle bedanken voor de begeleiding doorheen dit project. Onze wekelijkse vergaderingen waren een grote hulp bij zowel het bepalen van de richting van het project als bij de vormgeving van de thesis.

Vervolgens wil ik Chen-Chou Lo bedanken voor zijn raad toen ik moeite had om PSMNet werkend te krijgen.

Verder wil ik mijn ouders bedanken. Naast ervoor zorgen dat ik niets tekort kwam hebben ze mijn thesis nagelezen en opmerkingen gegeven. Dit heeft bijgedragen tot de algemene kwaliteit van de tekst van deze thesis.

Maxim Aelterman

9 juni 2020, Sint-Katelijne-Waver

# Samenvatting

Met de opmars van de artificiële intelligentie in het voorbije decennium worden autonome voertuigen een realiteit. Om dit proces veilig te laten verlopen is er nood aan mogelijkheden om betrouwbaar objecten in de omgeving te herkennen en te lokaliseren. Niet alleen moeten deze technieken betrouwbaar zijn, maar ook zo snel en zo goedkoop mogelijk. De meeste autonome wagens die nu op de markt zijn maken gebruik van dure LIDAR-sensors om afstanden te bepalen. Om de kosten van deze technologie te drukken zodat dit een standaard kan worden, wordt er gekeken naar alternatieven voor LIDAR. Pseudo-LIDAR is een veelbelovende techniek waarbij gebruik wordt gemaakt van een stereo-camera om diepte in te schatten. Hierbij wordt eerst een dispariteitskaart gegenereerd uit de stereo-beelden. Vervolgens wordt elk punt in deze kaart getransformeerd naar een punt in een 3D-scène waardoor de gegevens gaan lijken op LIDAR-gegevens. Deze gegevens worden dan gebruikt in een detectie netwerk dat bedoeld is om te werken met LIDAR.

De vraag die dan gesteld kan worden is: Waarom werkt pseudo-LIDAR beter dan dispariteitskaarten, wetende dat beide in essentie dezelfde informatie bevatten?

In deze thesis worden allerlei technieken besproken en vergeleken om objecten in de omgeving te herkennen. Er wordt gestart vanuit het gebruik van AVOD, een 3D detectie netwerk dat gebruik maakt van LIDAR en RGB-beelden voor zowel de detectie als de lokalisatie. Hiermee zijn 3 experimenten uitgevoerd: LIDAR, pseudo-LIDAR (SGBM) en pseudo-LIDAR (PSMNet).

Het gebruik van echte LIDAR-gegevens levert, zoals verwacht, de beste resultaten op. Wanneer deze resultaten vergeleken worden met de pseudo-LIDAR technieken is er een merkbaar verschil in performantie. Het eerste punt dat opvalt is hoe belangrijk de kwaliteit van de dispariteitsschatter is voor zowel de object herkenning als de lokalisatie. De pseudo-LIDAR die gegenereerd is met behulp van neurale netwerken presteert beduidend beter dan degene die gebruik maakt van een simpele dispariteitsschatter.

Hoewel een 2D detectie netwerk ook goede prestaties levert, valt het op dat de pseudo-LIDAR techniek in bepaalde situaties beter presteert. De bepaling van de precieze de omvang en locatie van de detecties is echter minder eenvoudige kwestie.

# Abstract

With the advance of artificial intelligence in the past decade, autonomous vehicles have become a reality. In order for this process to proceed safely, there is a need for reliable identification and localization of objects in the vicinity. Not only should these techniques be reliable, but also as fast and as cheap as possible. Most autonomous cars currently on the market use expensive LIDAR sensors to determine distances. To reduce the costs of this technology so that it can become a standard, alternatives to LIDAR are being considered. Pseudo-LIDAR is a promising technique that uses a stereo camera to estimate depth. Here, a disparity map is first generated from the stereo images. Each point in this map is then transformed into a point in a 3D scene, making the data resemble LIDAR data. This data is then used in a detection network that is intended to work with LIDAR.

The question that can then be asked is: Why does pseudo-LIDAR work better than disparity maps, knowing that both contain essentially the same information?

In this thesis all kinds of techniques are discussed and compared to recognize objects in the environment. AVOD, a 3D detection network that uses LIDAR and RGB images for both detection and localization, is being used in a first phase. Three experiments were carried out with this: LIDAR, pseudo-LIDAR (SGBM) and pseudo-LIDAR (PSMNet).

As expected, using real LIDAR data will provide the best results. When these results are compared with the pseudo-LIDAR techniques, there is a noticeable difference in performance. The first point to note is how important the quality of the disparity estimator is for both object recognition and localization. The pseudo-LIDAR generated using neural networks performs significantly better than the one using a simple disparity estimator.

Although a 2D detection network also provides good performance, it is noticeable that the pseudo-LIDAR technique performs better in certain situations.

**Keywords:** Pseudo-LIDAR, stereo camera, disparity map, 3D object detection, convolutional neural network

# Inhoudsopgave

<b>Voorwoord</b>	iii
<b>Samenvatting</b>	iv
<b>Abstract</b>	v
<b>Inhoud</b>	viii
<b>Figurenlijst</b>	x
<b>Tabellenlijst</b>	xi
<b>Symbolenlijst</b>	xii
<b>Lijst met afkortingen</b>	xiii
<b>1 Inleiding</b>	1
1.1 Situering . . . . .	1
1.2 Probleemstelling . . . . .	1
1.3 Doelstelling . . . . .	2
<b>2 Literatuurstudie</b>	3
2.1 Dieptesensors . . . . .	3
2.1.1 Stereo-camera . . . . .	3
2.1.2 LIDAR . . . . .	5
2.2 Datasets . . . . .	5
2.2.1 KITTI . . . . .	6
2.2.2 Scene Flow . . . . .	7
2.2.3 Waymo . . . . .	9
2.2.4 COCO . . . . .	9
2.3 Convolutionele Neurale Netwerken . . . . .	9

2.3.1	Feature learning . . . . .	10
2.3.2	Classificatie . . . . .	11
2.3.3	Training . . . . .	11
2.3.4	Testing . . . . .	12
2.3.5	Object detectie netwerken . . . . .	12
2.3.6	Trainen van modellen . . . . .	13
2.4	Cloud-based platformen . . . . .	13
2.4.1	Google Colaboratory . . . . .	14
2.4.2	Microsoft Azure . . . . .	14
2.4.3	Amazon Web Service . . . . .	14
2.5	Object detectie en lokalisatie . . . . .	14
2.5.1	Pseudo-LIDAR . . . . .	15
2.5.2	Pseudo-LIDAR++ . . . . .	15
<b>3</b>	<b>Gebruikte technieken</b>	<b>16</b>
3.1	PSMNet . . . . .	16
3.1.1	Spacial Pyramid Pooling . . . . .	16
3.1.2	3D CNN . . . . .	16
3.1.3	Training . . . . .	17
3.2	Dispariteiten omzetten naar puntenwolken . . . . .	17
3.3	AVOD . . . . .	18
3.3.1	RPN . . . . .	18
3.3.2	Detectie . . . . .	19
3.3.3	Training . . . . .	19
3.4	YOLOv3 . . . . .	19
3.4.1	Feature extractor . . . . .	20
3.4.2	Detector . . . . .	20
<b>4</b>	<b>Experimenten</b>	<b>22</b>
4.1	Inleiding . . . . .	22
4.1.1	Opstelling . . . . .	22
4.1.2	Evaluatie . . . . .	22
4.2	LIDAR . . . . .	24
4.2.1	Opstelling . . . . .	24
4.2.2	Werkwijze . . . . .	24
4.2.3	Resultaten . . . . .	25

4.3 Pseudo-LIDAR . . . . .	27
4.3.1 Opstelling . . . . .	27
4.3.2 Werkwijze . . . . .	27
4.3.3 Resultaten . . . . .	30
4.4 Dispariteitskaart . . . . .	34
4.4.1 Opstelling . . . . .	34
4.4.2 Werkwijze . . . . .	34
4.4.3 Resultaten . . . . .	35
4.5 Analyse . . . . .	36
4.5.1 LIDAR vs. Pseudo-LIDAR . . . . .	36
4.5.2 Pseudo-LIDAR vs. dispariteitskaart . . . . .	37
4.5.3 Toekomstig werk . . . . .	38
<b>5 Conclusies</b>	<b>39</b>
5.1 Overzicht . . . . .	39
5.2 LIDAR vs. Pseudo-LIDAR . . . . .	40
5.3 Pseudo-LIDAR vs. dispariteitskaart . . . . .	40

# Lijst van figuren

2.1 Schematische voorstelling van een stereo-camera. O en O' zijn de twee lenzen, X is een punt in de ruimte, x is dit punt in de linker afbeelding, x' is dit punt in de rechter afbeelding. (1) . . . . .	4
2.2 Vergelijking tussen het origineel beeld (bovenaan) en de gegenereerde diepte-kaart (onderaan). Hoe lichter de kleur, hoe dichter dat het object zich bij de camera bevindt. (2) . . . . .	4
2.3 Punten-wolk gegenereerd uit de data van een LIDAR. (3) . . . . .	5
2.4 Auto voorzien van stereo-camera, LIDAR-sensor en GPS (4) . . . . .	6
2.5 Tabel van de 10 best scorende algoritmes voor het detecteren van auto's (4) . . . . .	7
2.6 Vergelijking tussen gemakkelijk (bovenaan), gemiddeld (midden) en moeilijk (onderaan) te detecteren straatbeelden uit de KITTI dataset (4) . . . . .	8
2.7 Een voorbeeld van de 3 categorieën uit de Scene Flow dataset (5) . . . . .	8
2.8 Segmentatie in de COCO dataset (6) . . . . .	9
2.9 Opbouw van een CNN (7) . . . . .	10
2.10 Convolutie van een 5x5x1 input en een 3x3x1 kernel. De rode getallen stellen de waarden voor van de kernel. (7) . . . . .	11
2.11 Confusion matrix voorzien van formules om precision en recall (sensitivity in deze figuur) te berekenen. (8) . . . . .	12
2.12 Proces dat wordt doorlopen met de pseudo-LIDAR techniek. (2) . . . . .	15
3.1 Opbouw van PSMNet (9) . . . . .	17
3.2 Proces dat wordt doorlopen door AVOD (10). De roze blokken stellen het region proposal network voor, de groene blokken zijn het detectienetwerk. . . . .	19
3.3 De opbouw van het YOLO netwerk, inclusief de structuur van een feature vector. (11) . . . . .	20
3.4 Visualisatie van de output van YOLO (voordat de bounding boxes worden bepaald). De kleur geeft de meest waarschijnlijke klasse aan voor die cel (11) . . . . .	21
4.1 Schema van de sensors die aanwezig zijn op de KITTI-wagen (4) . . . . .	23
4.2 Velodyne HDL-64E LIDAR-sensor . . . . .	24

4.3 De puntenwolk uit een Velodyne LIDAR-sensor . . . . .	25
4.4 Precision-recall plot van het AVOD-netwerk met echte LIDAR gegevens voor 2D detecties . . . . .	26
4.5 Precision-recall plot van het AVOD-netwerk met echte LIDAR gegevens voor 3D detecties . . . . .	26
4.6 Point Grey Flea 2 (FL2-14S3M-C) camera . . . . .	27
4.7 Kaart van dispariteiten (onderaan), gegenereerd met PSMNet (één van de beelden wordt bovenaan weergegeven) . . . . .	28
4.8 Kaart van dispariteiten (onderaan), gegenereerd met Open CV (één van de beelden wordt bovenaan weergegeven) . . . . .	29
4.9 Puntenwolk die gegenereerd is uit de output van PSMNet (9) . . . . .	29
4.10 Puntenwolk die gegenereerd is uit de output van Open CV (SGBM) . . . . .	30
4.11 Precision-recall plot van het AVOD-netwerk met pseudo-LIDAR gegevens, gegenereerd met PSMNet dispariteiten . . . . .	31
4.12 Precision-recall plot van het AVOD-netwerk met pseudo-LIDAR gegevens, gegenereerd met PSMNet dispariteiten, voor 3D detecties . . . . .	31
4.13 Fouten bij het inschatten van diepte met SGBM . . . . .	32
4.14 Precision-recall plot van het AVOD-netwerk met pseudo-LIDAR gegevens, gegenereerd met SGBM dispariteiten . . . . .	32
4.15 Precision-recall plot van het AVOD-netwerk met pseudo-LIDAR gegevens, gegenereerd met SGBM dispariteiten, voor 3D detecties. . . . .	33
4.16 De groene en gele rechthoeken zijn de bounding boxes die verkregen zijn uit het YOLO netwerk. De rode rechthoeken zijn de bounding boxes die achteraf aangepast zijn, rekening houdend met occlusie. De blauwe kaders zijn detecties die wegens de afstand niet relevant zijn. . . . .	35
4.17 Precision-recall plot van het YOLOv3 netwerk . . . . .	35
4.18 Gevisualiseerde detecties van AVOD met LIDAR (links) en Pseudo-LIDAR met PSM-Net (rechts). Pseudo-LIDAR scoort lager en maakt een valse detectie (rechts) . . . . .	36

# **Lijst van tabellen**

2.1 Overzicht van datasets . . . . .	6
--------------------------------------	---

# Lijst van symbolen

$c$	Lichtsnelheid	$[m/s]$
$d$	Afstand	$[m]$
$f$	Brandpuntsafstand	$[m^{-1}]$
$T$	Tijd	$[s]$

# Lijst van afkortingen

<i>AI</i>	Artificial Intelligence
<i>AOS</i>	Average Orientation Similarity
<i>AP</i>	Average Precision
<i>AUC</i>	Area Under Curve
<i>BEV</i>	Bird's Eye View
<i>CNN</i>	Convolutional Neural Network
<i>CPU</i>	Central Processing Unit
<i>CUDA</i>	Compute Unified Device Architecture
<i>FCN</i>	Fully Convolutional Network
<i>GPU</i>	Graphical Processing Unit
<i>HSV</i>	Hue Saturation Value (voorstelling van kleur)
<i>IaaS</i>	Infrastructure as a Service
<i>IoU</i>	Intersection over Union
<i>LIDAR</i>	Light Detection And Ranging / Laser Imaging Detection And Ranging
<i>ML</i>	Machine Learning
<i>NMS</i>	Non-Maximum Suppression
<i>PaaS</i>	Platform as a Service
<i>RGB</i>	Red Green Blue (voorstelling van kleur)
<i>RPM</i>	Revolutions Per Minute
<i>RPN</i>	Regional Proposal Network
<i>ReLU</i>	Rectifier Linear Unit
<i>SaaS</i>	Software as a Service
<i>SGBM</i>	Semi-Global Block-Matching
<i>TPU</i>	Tensor Processing Unit

# **Hoofdstuk 1**

## **Inleiding**

### **1.1 Situering**

Met de forse toename van onderzoek naar artificiële intelligentie en neurale netwerken zijn er veel toepassingen voor AI gevonden. De toepassingen zijn heel breed, gaande van kleine GSM apps tot analyse van gebruikersgegevens van tech-giganten zoals Google en Facebook voor bijvoorbeeld gerichte reclame. Er zijn echter toepassingen die nog niet helemaal op punt staan en nog nood hebben aan verbetering. Autonome navigatie van voertuigen, robots en drones zijn hier een voorbeeld van. Als de AI van deze toestellen een fout maakt kan dit schade toebrengen aan het toestel en de omgeving. Het is dus belangrijk dat deze systemen een accuraat beeld krijgen van de omgeving en voldoende inzicht hebben om ongevallen te vermijden.

### **1.2 Probleemstelling**

Hoewel we met LIDAR-sensors een accuraat beeld kunnen vormen van de omgeving heeft deze technologie ook zijn nadelen. Het grootste nadeel is de kost van dergelijke sensors. Voor een 360° LIDAR-sensor die snel en precies genoeg is, kan de prijs al snel oplopen tot verschillende duizenden euro's. Een ander probleem dat we tegenkomen is dat de toestellen die gebruik willen maken van 3D object-detectie voorzien moeten zijn van tamelijk veel rekenkracht. Dit drijft de kost van dergelijke systemen op en vermindert de autonomie van het toestel indien het op batterij moet werken.

### 1.3 Doelstelling

Momenteel bestaan er een aantal technieken die het mogelijk maken om in een 3D-omgeving objecten te identificeren en te lokaliseren. De nieuwkomer in dit gebied is pseudo-LIDAR, waarbij men vanuit stereo-beelden een puntenwolk genereert. Deze techniek ziet er veelbelovend uit om kosten te drukken. De doelstelling van deze masterproef is om verdergaand onderzoek hiernaar te doen. In de studie worden LIDAR en pseudo-LIDAR met elkaar vergeleken, met als voornaamste element de kost. Het weglaten van de LIDAR-sensor zal kostenbesparend werken, maar de behoefte aan meer rekenkracht van het systeem zal dan waarschijnlijk de kost weer doen toenemen.

De bestanden van het project staan op de github pagina:

<https://github.com/MaximAelterman/Masterproef>

# **Hoofdstuk 2**

## **Literatuurstudie**

### **2.1 Dieptesensors**

In volgend deel worden de technologieën besproken die essentieel zijn voor deze masterproef. Voor de visualisatie van de omgeving wordt er gebruik gemaakt van stereo-camera's en LIDAR. De detectie en lokalisatie van objecten in de omgeving wordt uitgevoerd door een neuraal netwerk, getraind op de KITTI-dataset.

#### **2.1.1 Stereo-camera**

Een stereo-camera bestaat uit twee camera's die op een bepaalde afstand uit elkaar staan, deze afstand wordt de baseline genoemd. De techniek is gebaseerd op het Parallax-effect: als de waarnemer van positie verandert, zullen dingen die dichterbij zijn meer lijken te verschuiven t.o.v. dingen die zich verder bevinden. Omdat de lenzen uit elkaar staan zullen ze de omgeving op een iets andere manier waarnemen. Een bepaald punt in het linker beeld zal op een iets andere locatie te vinden zijn in het rechter beeld. Het verschil in afstand tussen deze locaties wordt de dispariteit genoemd. Hoe dichter een object bij de lens geplaatst wordt, hoe groter de dispariteit zal zijn.

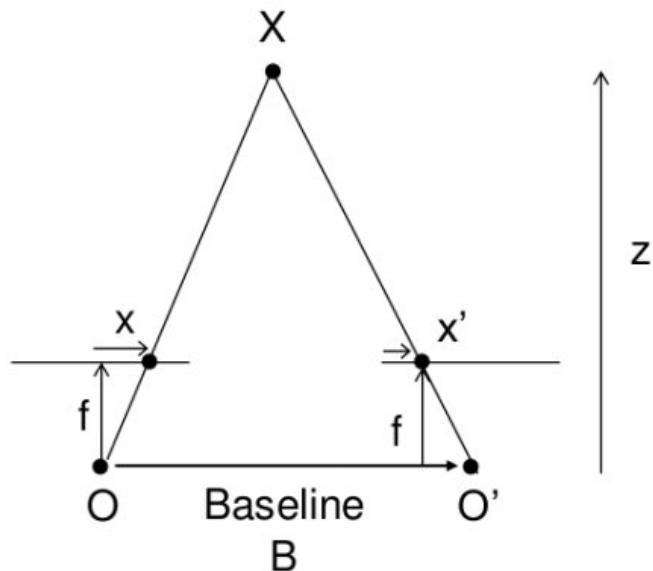
Als de baseline ( $B$ ), de brandpuntsafstand ( $f$ ) van de lens en de dispariteit ( $d$ ) van een punt in de afbeelding gekend zijn, kan de afstand ( $z$ ) berekend worden met formule 2.1.

$$z = \frac{f * B}{d} \quad (2.1)$$

##### **2.1.1.1 Diepte-kaart**

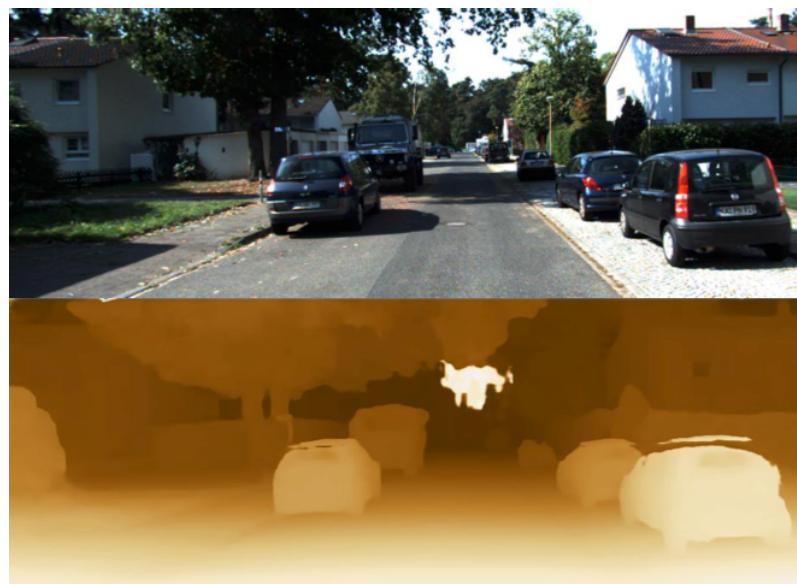
Om de diepte-informatie uit het stereo-beeld voor te stellen wordt er een diepte-kaart berekend. De diepte kan berekend worden op twee manieren: gebruik makend van beeldverwerking of door middel van een Convolutioneel Neuraal Netwerk of CNN (7).

De klassieke manier om de diepte te berekenen is met beeldverwerking. Een bepaald punt in het beeld wordt gelokaliseerd in beide beelden om de dispariteit van dat punt te bepalen. Vervolgens



**Figuur 2.1:** Schematische voorstelling van een stereo-camera. O en O' zijn de twee lenzen, X is een punt in de ruimte, x is dit punt in de linker afbeelding, x' is dit punt in de rechter afbeelding. (1)

kan voorgaande formule gebruikt worden om de diepte te bepalen. Pixels met een vergelijkbare afstand worden gegroepeerd en krijgen eenzelfde kleur. (Fig. 2.2)



**Figuur 2.2:** Vergelijking tussen het origineel beeld (bovenaan) en de gegenereerde diepte-kaart (onderaan). Hoe lichter de kleur, hoe dichter dat het object zich bij de camera bevindt. (2)

In dit project wordt gebruik gemaakt van zowel een CNN, als een meer klassieke pixel-match techniek. CNN technieken leveren accuratere resultaten op, maar vereisen meer rekenkracht. De werking van CNN's wordt besproken in sectie 2.3. PSMNet (9) is een voorbeeld van een dergelijk netwerk.

### 2.1.2 LIDAR

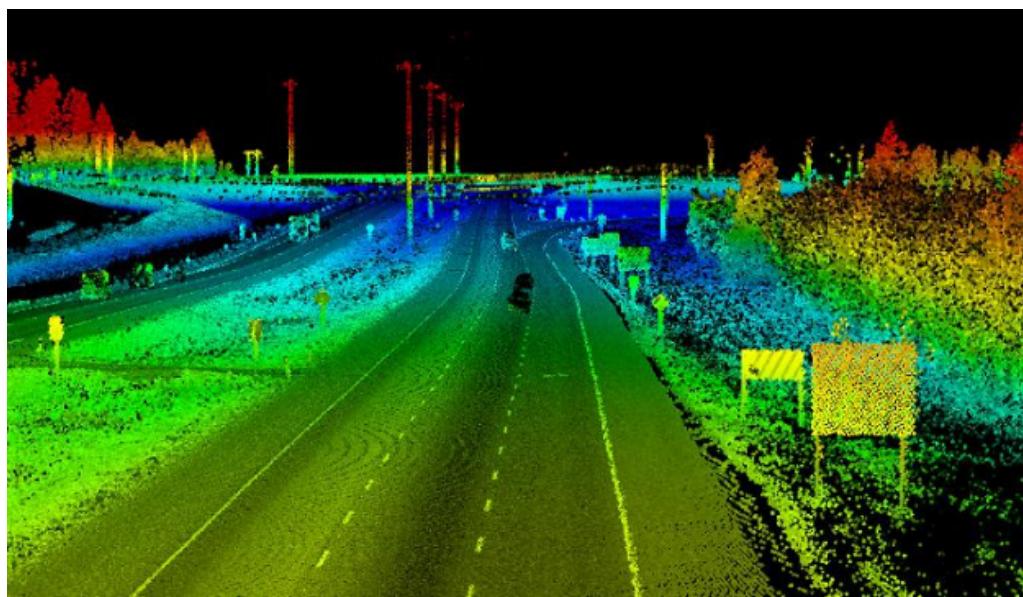
Een Light Detection and Ranging of kort LIDAR is een zogenaamde 'time-of-flight' sensor. Dit wil zeggen dat de sensor zelf licht, een laser, uitstraalt en meet hoe lang het licht ertoe doet om de sensor te bereiken na reflectie op de omgeving. De tijd die het licht nodig heeft om een afstand te overbruggen kan gebruikt worden om deze afstand te berekenen volgens formule 2.2.

$$d = \frac{c * T}{2} \quad (2.2)$$

Hierin stelt  $d$  de afstand tot het object voor,  $c$  is de lichtsnelheid en  $T$  is de tijd die het licht nodig had om tot het object te komen, en terug (daarom wordt dit resultaat nog eens gedeeld door 2).

Aangezien er met lasers gewerkt wordt kan hiermee slechts een zeer klein deel van de omgeving opgemeten worden. LIDAR maakt daarom gebruik van zeer korte, snel opeenvolgende lichtpulsen zodat er zeer veel metingen kunnen worden gedaan per seconde. Om niet telkens dezelfde plek in de ruimte te meten moet de sensor rond draaien om de omgeving te scannen.

De vergaarde informatie wordt in de vorm van een puntenwolk voorgesteld zoals in figuur 2.3.



**Figuur 2.3:** Punten-wolk gegenereerd uit de data van een LIDAR. (3)

## 2.2 Datasets

In deze sectie worden een aantal populaire datasets voor het trainen en testen van netwerken besproken. Om deze lijst overzichtelijker te maken zijn ze in tabel 2.1 opgenomen met hun belangrijkste kenmerken.

**Tabel 2.1** Overzicht van datasets

Naam	Type	Grootte	Labels	Extra gegevens
<b>KITTI</b>	Foto	14.000	Voertuigen Fietsers Voetgangers	LIDAR Multiview
<b>Scene Flow</b>	Synthetisch (3D)	39.000	Voertuigen Anderen	Segmentatie Dispariteiten beweging tussen beelden
<b>Waymo</b>	Video	1.950	Voertuigen Fietser Voetgangers Verkeersborden	LIDAR
<b>COCO</b>	Foto	200.000	80 klassen	Segmentatie

### 2.2.1 KITTI

KITTI (4) is een verzameling van stereo-beelden, voorzien van LIDAR-gegevens die gebruikt kunnen worden als ground truth voor de schatting van afstand op basis van stereo-beelden, of als extra gegevens voor de object detectie. De beelden zijn gemaakt vanuit een auto zoals in figuur 2.4 en bestaan dus uitsluitend uit straatbeelden.

**Figuur 2.4:** Auto voorzien van stereo-camera, LIDAR-sensor en GPS (4)

Naast training- en test-data voorziet KITTI een platform waarop onderzoekers de prestaties van hun algoritme kunnen vergelijken met anderen. Deze prestaties worden gerangschikt op de gemiddelde precisie die het algoritme behaald heeft (Fig. 2.5). De dataset kan opgedeeld worden in 3 moeilijkheidsgraden die worden bepaald door de grootte van het object in de afbeelding (in pixels) en de mate van oclusie van de te detecteren objecten.

[Additional information used by the methods](#)

- Stereo: Method uses left and right (stereo) images
- Flow: Method uses optical flow (2 temporally adjacent images)
- Multiview: Method uses more than 2 temporally adjacent images
- Laser Points: Method uses point clouds from Velodyne laser scanner
- Additional training data: Use of additional data sources for training (see details)

Car

	Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment	Compare
1	<a href="#">MMLab PV-RCNN</a>	<input checked="" type="checkbox"/>		81.43 %	90.25 %	<b>76.82 %</b>	0.08 s	1 core @ 2.5 Ghz (Python + C/C++)	<input type="checkbox"/>
	S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang and H. Li: <a href="#">PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection</a> . arXiv preprint arXiv:1912.13192 2019.								
2	<a href="#">SA-SSD</a>			79.79 %	88.75 %	74.16 %	0.04 s	1 core @ 2.5 Ghz (Python)	<input type="checkbox"/>
3	<a href="#">STD</a>			79.71 %	87.95 %	75.09 %	0.08 s	GPU @ 2.5 Ghz (Python + C/C++)	<input type="checkbox"/>
	Z. Yang, Y. Sun, S. Liu, X. Shen and J. Jia: <a href="#">STD: Sparse-to-Dense 3D Object Detector for Point Cloud</a> . ICCV 2019.								
4	<a href="#">3DSSD</a>			79.57 %	88.36 %	74.55 %	0.04 s	GPU @ 2.5 Ghz (Python + C/C++)	<input type="checkbox"/>
5	<a href="#">Point-GNN</a>	<input checked="" type="checkbox"/>		79.47 %	88.33 %	72.29 %	0.6 s	GPU @ 2.5 Ghz (Python)	<input type="checkbox"/>
6	<a href="#">EPNet</a>			79.28 %	89.81 %	74.59 %	0.1 s	1 core @ 2.5 Ghz (Python + C/C++)	<input type="checkbox"/>
7	<a href="#">CAASS-3D</a>			79.03 %	87.96 %	72.78 %	0.1 s	1 core @ 2.5 Ghz (C/C++)	<input type="checkbox"/>
8	<a href="#">Noah CV Lab - SSL</a>			78.99 %	85.50 %	71.75 %	0.1 s	GPU @ 2.5 Ghz (Python)	<input type="checkbox"/>
9	<a href="#">CPRCCNN</a>			78.96 %	87.74 %	74.30 %	0.1 s	1 core @ 2.5 Ghz (Python)	<input type="checkbox"/>
10	<a href="#">ORP</a>			78.50 %	87.38 %	71.49 %	0.06 s	1 core @ 2.5 Ghz (C/C++)	<input type="checkbox"/>

**Figuur 2.5:** Tabel van de 10 best scorende algoritmes voor het detecteren van auto's (4)

In fig. 2.6 is te zien wat het verschil is tussen de moeilijkheidsgraden:

**Gemakkelijk:** De te detecteren objecten zijn volledig zichtbaar en niet te ver weg.

**Gemiddeld:** In de beelden mogen de objecten wat bedekt zijn en verder van de camera zijn.

**Moeilijk:** In deze beelden mogen de objecten verder weg zijn en moeten ze gedetecteerd worden als er slechts kleine stukken zichtbaar zijn.

## 2.2.2 Scene Flow

De Scene Flow Dataset (5) bestaat uit meer dan 39000 synthetisch gegenereerde 3D scènes. Omdat deze scènes gegenereerd zijn door een computer, is het eenvoudig om een *groundtruth* te bepalen. Deze beelden vormen samen een bewegend beeld. Elke afbeelding is onder andere voorzien van een segmentatiekaart om objecten te kunnen onderscheiden, een dispariteitkaart en de intrinsieke and extrinsieke gegevens van de camera die gebruikt worden om diepte te berekenen met dispariteiten.

Er zijn 3 categorieën van scènes beschikbaar:

**FlyingThings3D:** een aantal willekeurige objecten die in de lucht zweven, zoals te zien in het linker deel van fig. 2.7

**Driving:** meer realistische scènes van straatbeelden (middelste deel van fig. 2.7)

**Monkaa:** scènes met een cartoon aap (rechter deel van fig. 2.7)



**Figuur 2.6:** Vergelijking tussen gemakkelijk (bovenaan), gemiddeld (midden) en moeilijk (onderaan) te detecteren straatbeelden uit de KITTI dataset (4)



**Figuur 2.7:** Een voorbeeld van de 3 categorieën uit de Scene Flow dataset (5)

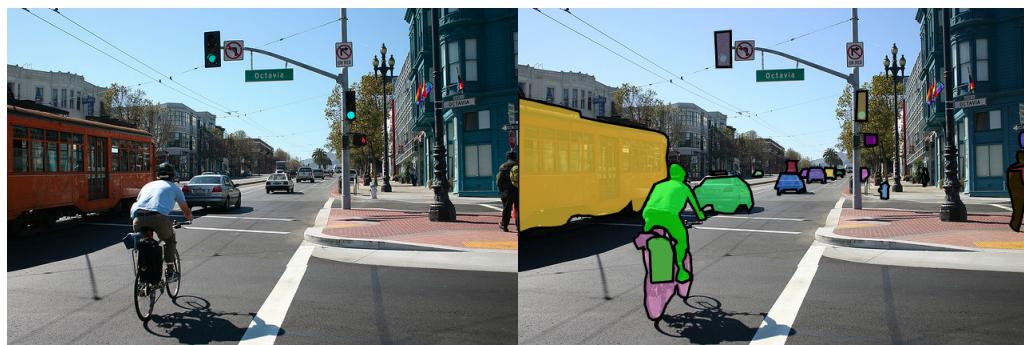
### 2.2.3 Waymo

De Waymo Open Dataset is specifiek bedoeld voor het ontwikkelen van software voor autonome wagens. De dataset bestaat uit 1950 video-segmenten van 20 seconden elk. Deze beelden zijn voor 4 klassen gelabeld: auto's, voetgangers, fietsers en verkeersborden. De wagen die de dataset heeft gemaakt is uitgerust met verschillende LIDAR-sensors en camera's (zowel vooraan als achteraan). Bij het registreren van de gegevens werd rekening gehouden met de diversiteit van het verkeer. Zo zijn er gegevens beschikbaar bij verschillende situaties zoals weersomstandigheden, omgevingen, wegenwerken en nachtverkeer.

Waymo organiseert ook wedstrijden in volgende categorieën: 3D of 2D detectie en 3D of 2D tracking (volgen van objecten over meerdere frames). In elke categorie is er een hoofdprijs van 15000 dollar te winnen.

### 2.2.4 COCO

Microsoft Common Objects in Contexts of COCO (6) is een verzameling van meer dan 200.000 gelabelde afbeeldingen. Deze dataset heeft labels voor 80 object klassen waaronder wagens, fietsers, verkeersborden en voetgangers. Alle objecten zijn naast de klassieke bounding boxes ook voorzien van segmentatie om preciezere detecties toe te laten (Fig. 2.8).

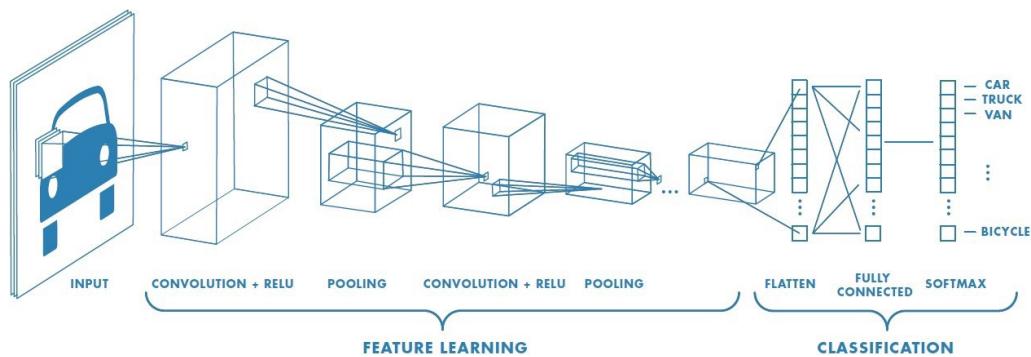


Figuur 2.8: Segmentatie in de COCO dataset (6)

## 2.3 Convolutionele Neurale Netwerken

De bedoeling van een convolutioneel neuraal netwerk of CNN is om in gegevens patronen te herkennen. In de beeldverwerking zijn deze gegevens gepresenteerd in de vorm van een afbeelding en zal de CNN trachten hierin objecten of situaties te identificeren.

Zoals in fig. 2.9 te zien is, bestaat een CNN gewoonlijk uit volgende onderdelen: Feature learning, gevolgd door classificatie.



**Figuur 2.9:** Opbouw van een CNN (7)

### 2.3.1 Feature learning

De feature learning stap bestaat uit een opeenvolging van convolutie-lagen die onderling verbonden zijn met behulp van pooling-lagen. Hierbij is het de bedoeling om zoveel mogelijk informatie uit de afbeelding te halen.

In een convolutie-laag wordt de input geconvoluteerd met een kernel. Een kernel is een matrix met bepaalde waarden en grootte. Deze kan voor elke convolutie-laag verschillen en bepaalt welke eigenschappen (features) uit de input worden gehaald. Bij convolutie wordt de kernel over de input geschoven. De waarden van de kernel worden dan vermenigvuldigd met de overeenkomstige input-waarden. Deze resultaten worden vervolgens opgeteld, dit is dan de berekende waarde voor die positie. In figuur 2.10 worden enkele iteraties van dit proces voorgesteld. In dit voorbeeld ziet de kernel er als volgt uit:

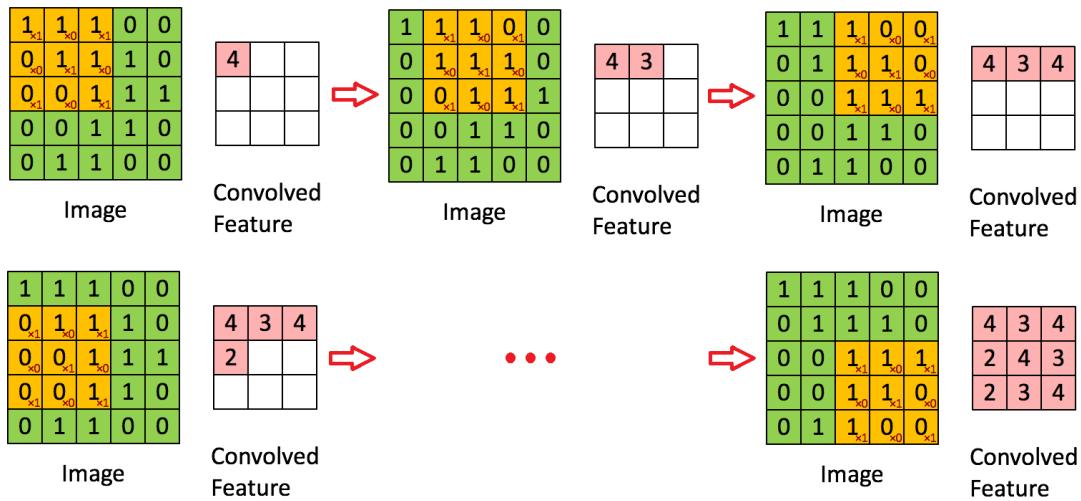
$$\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix}$$

Merk op dat dit een  $3 \times 3 \times 1$  kernel is, dus 2 dimensies, maar als bv. een afbeelding in 3 componenten wordt voorgesteld (RGB, HSV, ..) kan een kernel ook 3 dimensionaal worden.

Een convolutie-laag kan voorzien zijn van een Rectifier Linear Unit of ReLU. Deze rectifier units verkorten trainingstijd door de bekomen features te verbeteren aan de hand van simpele lineaire operaties.

De volgende stap in feature learning is pooling tussen de convolutie-lagen. Er zijn verschillende redenen om een neurale netwerk te voorzien van pooling lagen. Ten eerste wordt pooling gebruikt om de rekenkracht, die vereist is om de gegevens te verwerken, te reduceren. Ten tweede: als een netwerk rekening moet houden met teveel features kan dit leiden tot overfitting. Ook dit kan verholpen worden door met pooling de feature map down te samplen. Dit gebeurt door de verkregen features van voorgaande convolutie-lagen te bundelen en samen te vatten in dominante features.

Er zijn 2 vormen van pooling: max pooling en average pooling. In beide vormen worden features van de convolutie-laag gebundeld. Per bundel worden deze features herleid tot een dominante



**Figuur 2.10:** Convolutie van een 5x5x1 input en een 3x3x1 kernel. De rode getallen stellen de waarden voor van de kernel. (7)

feature. Bij max pooling wordt de feature met de hoogste waarde gebruikt, bij average pooling is dit het gemiddelde van de betreffende features.

### 2.3.2 Classificatie

De eigenlijke classificatie gebeurt door de fully-connected lagen na de feature learning. De output van de feature learning stap wordt gevormd door de meest significante features van de input-afbeelding. Deze worden in een vector geplaatst (= flattened) en gebruikt als input voor de fully-connected laag. Op basis van deze gegevens bepalen de fully-connected lagen een reeks van scores, 1 per classificatie-klasse waarop het netwerk getraind is. Deze score gaat van 0 tot 1. Hoe dichter deze score bij 1 ligt, hoe zekerder het is dat de afbeelding onder de bijhorende klasse valt.

### 2.3.3 Training

Een CNN wordt aan de hand van veel, hoe meer hoe beter, data getraind om de gewenste resultaten te bekomen. Deze data wordt geannoteerd zodat het algoritme voorbeelden heeft van de te herkennen patronen. Gewoonlijk wordt de beschikbare data opgedeeld in 3 delen. Het grootste deel wordt gebruikt als trainingsdata. De rest van de data wordt nog eens opgedeeld. Een deel wordt gebruikt als validatie-data om het netwerk te testen tijdens de trainingsfase. Het ander deel, de test-data, kan achteraf gebruikt worden om het netwerk te testen. Bij training wordt vooraf bepaald hoeveel keer het algoritme zichzelf zal proberen te verbeteren. Dit wordt het aantal epochs of iteraties genoemd.

Tijdens de eerste iteratie worden de kernels van de eerste convolutie-laag geïnitialiseerd op willekeurige waarden. Dit wordt de forward pass genoemd. Vervolgens wordt de loss-functie berekend. Deze functie geeft weer hoe goed het netwerk presteert. De volgende iteraties bestaan eruit om

deze loss-functie te minimaliseren door een aantal backward passes te doen. Op basis van de resultaten van de loss-functie worden de weights aangepast en kan een nieuwe loss-functie berekend worden. De backward pass gaat op zoek naar de weights die het meest hebben bijgedragen tot de verkregen loss. Door deze weights te veranderen zou de loss-functie een beter resultaat kunnen opleveren in de volgende iteratie.

### 2.3.4 Testing

Bij het testen van een CNN is het belangrijk dat er andere data wordt aangeleverd dan die die gebruikt werd om de CNN te trainen. Deze regel negeren kan namelijk resulteren in unrealistische resultaten. Een effectieve manier om de prestaties van een CNN voor te stellen is door de precision en recall (of sensitivity) van een test-dataset te berekenen. Deze methode maakt gebruik van het principe van de confusion matrix, zoals in figuur 2.11 geïllustreerd. De precision is de verhouding tussen het aantal juiste voorspellingen die het netwerk heeft gemaakt op de test-dataset en het totaal aantal voorspellingen. De recall is de verhouding tussen hoeveel voorspellingen juist zijn en het eigenlijk aantal voorspellingen die juist zouden moeten zijn.

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) <b>Type II Error</b>	Sensitivity $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) <b>Type I Error</b>	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
	Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$	

**Figuur 2.11:** Confusion matrix voorzien van formules om precision en recall (sensitivity in deze figuur) te berekenen. (8)

### 2.3.5 Object detectie netwerken

Er zijn veel neurale netwerken die gebruikt worden om objecten te detecteren uit RGB-beelden. Hieronder worden enkele interessante netwerken besproken. Een uitgebreide uitleg van deze technieken wordt gegeven in betreffende secties in hoofdstuk 3.

### 2.3.5.1 AVOD

AVOD (10) is een netwerk dat gebruik maakt van zowel RGB-beelden als van de puntenwolken die gegenereerd worden door een LIDAR-sensor. De combinatie van deze gegevens maakt dat AVOD meer accurate resultaten oplevert dan de meer traditionele netwerken die enkel gebruik maken van RGB-beelden.

### 2.3.5.2 YOLOv3

YOLOv3 (12) is niet het meest accurate netwerk, maar maakt dit goed met zijn snelle inferentie. Dit netwerk is dus zeer geschikt voor “real-time” toepassingen of voor mobiele/low-end toepassingen. Het belangrijkste kenmerk van YOLO is dat het de input-afbeelding opsplits in regio’s. Op elke regio wordt object detectie gedaan. Het resultaat hiervan wordt als label opgeslagen. Dit heeft als voordeel dat het inferentie versnelt, ten koste van de precisie van de detectie en bounding box. De opdeling in regio’s maakt het ook mogelijk om meerdere detecties in dezelfde afbeelding te doen.

## 2.3.6 Trainen van modellen

### 2.3.6.1 Libraries

Volgende libraries laten toe om netwerken op te bouwen en te trainen op één of meerdere GPU’s of CPU’s. Dit laat toe om grotere en complexere netwerken te bouwen.

**Pytorch** Een open-source machine learning library, ontwikkeld door Facebook. Deze library is gemaakt voor Python.

**Tensorflow** Een open-source library die gebruikt kan worden om neurale netwerken te maken, ontwikkeld door Google. Er bestaat een Python en een JavaScript versie.

## 2.4 Cloud-based platformen

Cloud-based platformen maken machine learning toegankelijker. Niet iedereen heeft namelijk toegang tot de dure hardware, vereist om complexere netwerken te kunnen trainen of gebruiken.

De 3 belangrijkste soorten van cloud-computing zijn: IaaS, PaaS en SaaS. Bij Infrastructure as a Service of IaaS wordt hardware beschikbaar gesteld aan de klant. Hierop draaien virtual machines waar de gebruiker op inlogt. Dit geeft het grootste flexibiliteit en de meeste controle over de service. Platform as a Service of PaaS maakt het eenvoudiger om van start te gaan omdat het betreffend platform waarop gewerkt wordt helpt bij de opstart. Taken zoals onderhoud en updates worden door de service provider geleverd. Bij Software as a Service of SaaS wordt een afgewerkt product aangeboden. Deze software wordt draaiende gehouden door de service provider. Dit is geen development tool en valt dus buiten de scope van deze thesis.

### 2.4.1 Google Colaboratory

Google voorziet een gratis (indien niet-commercieel) cloud-service waarmee men een project kan maken dat gesaved wordt op Google Drive. Deze omgeving wordt geprogrammeerd in Python, en laat het gebruik van libraries zoals Tensorflow toe. De omgeving laat toe om eenvoudig te wisselen tussen Python 2 en 3. Hiernaast is het ook mogelijk om de code uit te voeren op een CPU, GPU of TPU. Een TPU of Tensor Processing Unit is een processor, specifiek gemaakt om efficiënt tensorflow te kunnen gebruiken. Zowel het schrijven van de code als het trainen en uitvoeren van het netwerk kunnen vanuit een browser worden uitgevoerd. Datasets kunnen geupload worden naar Google Drive en rechtstreeks van daaruit gebruikt worden. Dit maakt deze service zeer draagbaar en toegankelijk; enkel een computer met internetverbinding is vereist.

### 2.4.2 Microsoft Azure

Microsoft Azure is een verzameling van cloud development tools, waaronder een tool voor machine learning. Het is mogelijk om een virtuele machine te huren op de servers van Microsoft Azure, uitgerust met veel rekenvermogen en een sterke grafische kaart. Gebruik maken van de service wordt per uur aangerekend en het tarief is afhankelijk van de gehuurde hardware.

### 2.4.3 Amazon Web Service

Amazon Web Service of AWS heeft een groot aanbod aan services in verschillende sectoren zoals development, machine learning, VR, etc. Deze services zijn van alle types: IaaS, PaaS en SaaS. Naast de klassieke frameworks zoals PyTorch en TensorFlow biedt AWS ook zijn eigen platformen aan zoals Amazon SageMaker. Amazon SageMaker is een service die bedoeld is voor het maken, trainen en uitvoeren van ML-modellen. Hier bestaat ook een uitbreiding voor om datasets te labelen en klaar te maken voor training.

## 2.5 Object detectie en lokalisatie

Momenteel zijn er 2 courant gebruikte technieken om objecten in de ruimte te kunnen identificeren: gebruik makend van stereo-beelden of van LIDAR-puntenwolken. Deze technologieën hebben echter elk hun voor- en nadelen.

**LIDAR** is zeer accuraat en levert dus ook bij object detectie en lokalisatie zeer goede resultaten op. Dit is enkel het geval bij de zeer dure hoge-resolutie modellen, die verschillende tienduizenden euro's kunnen kosten. Hierdoor is het niet realistisch om dit te implementeren in toestellen bedoeld voor de doorsnee consument, zoals bijvoorbeeld autonome voertuigen.

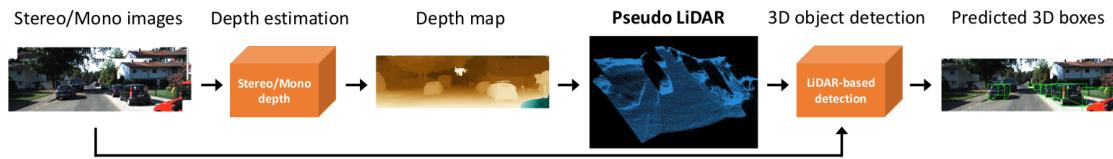
**Stereo-beelden** hebben als voordeel dat ze goedkoop en compact (kunnen) zijn. Helaas heeft deze technologie het nadeel dat ze niet zo accuraat is bij het bepalen van afstand tot het gede-

tecteerd object. Daarom doet men nu onderzoek naar pseudo-LIDAR, gegenereerd uit stereo-beelden, wat in volgende secties uitgelegd wordt.

### 2.5.1 Pseudo-LIDAR

Een interessante aanpak is die van Wang et al. (2), waar men op basis van stereo-beelden een puntenwolk genereert. Deze puntenwolk simuleert de output van een LIDAR sensor. Het resultaat levert geen extra diepte-data op, maar de voorstelling van de beschikbare data wordt wel drastisch veranderd. De onderzoekers beweren dat deze voorstelling veel gunstiger is voor een neurale netwerk om aan 3D object herkenning en lokalisatie te doen. Bijgevolg is dit een veel goedkopere, en dus meer realistische opstelling voor toepassingen zoals autonome wagens.

Deze techniek kan gezien worden als een soort van pre-processing. De traditionele werkwijze gaat als volgt (fig. 2.12): Uit de stereo-beelden wordt de afstand ingeschat per pixel. Deze afstanden kunnen vervolgens worden voorgesteld in een diepte map, waarop een neurale netwerk getraind kan worden om objecten te herkennen. Pseudo-LIDAR zal gebruik maken van deze diepte map om een puntenwolk te genereren, zoals dat gedaan wordt bij de data verkregen door een echte LIDAR. Dit laat ons toe om een netwerk te trainen op 3D-data, wat volgens de onderzoekers een beter resultaat oplevert. Of dit klopt gaan we in dit paper onderzoeken.



Figuur 2.12: Proces dat wordt doorlopen met de pseudo-LIDAR techniek. (2)

### 2.5.2 Pseudo-LIDAR++

You et al. (13) werken verder op de techniek van Wang et al. (2). Een probleem met deze techniek is dat de diepte niet zo accuraat ingeschatt kan worden met enkel stereo-beelden. De fout in de geschatte afstand met de stereo-beelden kan gecorrigeerd worden door gebruik te maken van een goedkope low-end LIDAR. Deze zijn nog steeds beter in het bepalen van afstand dan stereo-beelden, maar hebben een te lage resolutie om op zichzelf bruikbaar te zijn. De combinatie van de beeld-informatie van de stereo-camera en de diepte-informatie van de LIDAR zou een significante verbetering in de testresultaten opleveren.

# **Hoofdstuk 3**

## **Gebruikte technieken**

### **3.1 PSMNet**

De klassieke methoden voor het inschatten van dispariteiten, zoeken overeenkomstige pixels door pixelmatches of blockmatches. Deze technieken houden slechts rekening met een klein deel van de afbeelding en presteren daarom slecht in bepaalde situaties zoals occlusiegebieden en effen oppervlakken zonder textuur. De occlusiegebieden maken dat overeenkomstige pixels niet bestaan omdat betreffende plaats in één van de beelden bedekt wordt. Wanneer een oppervlak geen duidelijke textuur heeft lijken nabije pixels te veel op elkaar en kan er geen accurate match gevonden worden.

Pyramid Stereo Matching Network of PSMNet (9) is een neuraal netwerk dat uit stereobeelden een dispariteitskaart genereert. Door naar de hele afbeelding te kijken in plaats van enkel op pixelniveau, zoals de klassieke methodes, kunnen moeilijke situaties ook ingeschat worden. Het netwerk bestaat uit twee belangrijke delen: Spacial pyramid pooling (SPP) en 3D CNN (Fig. 3.1).

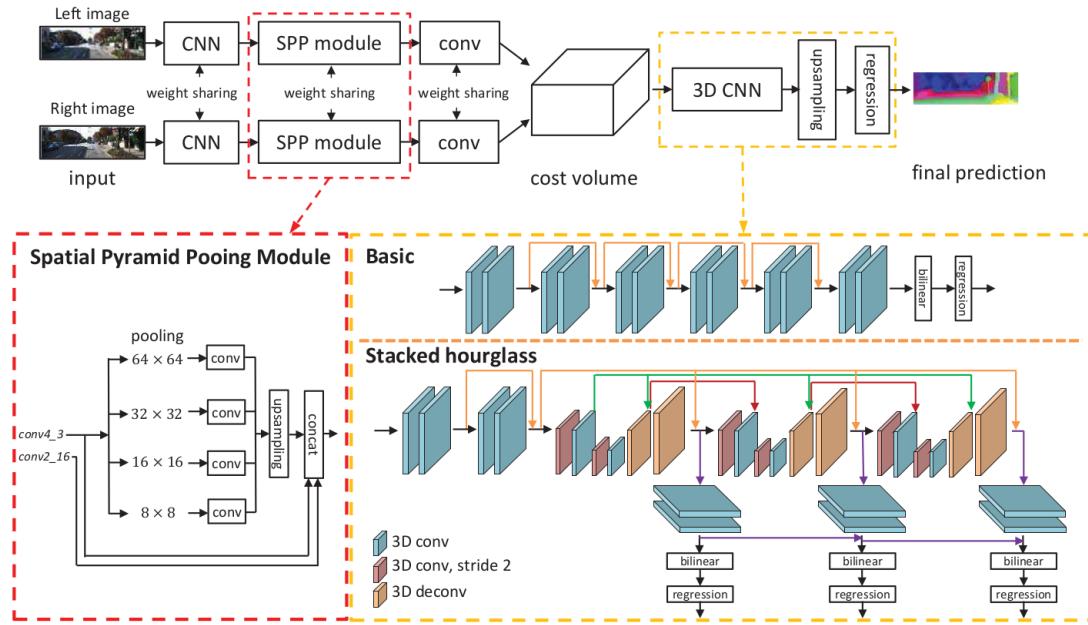
#### **3.1.1 Spacial Pyramid Pooling**

De Spacial Pyramid Pooling module of SPP gebruikt de informatie van de globale context van de afbeeldingen om de schattingen te verbeteren. Deze module leert een verband te leggen tussen een object (bv. wagen) en zijn sub-regio's (bv. vensters, wielen, koplampen, enz.) om zo context informatie te vergaren. De vergaarde features kunnen in de 3D CNN stap gebruikt worden om het matchen van overeenkomstige pixels in de stereo-beelden te vergemakkelijken.

#### **3.1.2 3D CNN**

PSMNet heeft twee 3D CNN architecturen: de basis en stacked hourglass. Deze architecturen worden in figuur 3.1 gevisualiseerd in de gele kaders.

De basis architectuur bestaat uit twaalf opeenvolgende  $3 \times 3 \times 3$  convolutionele lagen. Op het einde van deze reeks lagen wordt de cost volume geëupsampled via bilineaire interpolatie. Vervolgens



**Figuur 3.1:** Opbouw van PSMNet (9)

wordt er regressie toegepast om de dispariteitskaart te berekenen.

De tweede architectuur, stacked hourglass (encoder-decoder), wordt in dit project gebruikt omdat het toelaat om meer te leren uit de context informatie van de stereo-beelden. Deze bestaat uit drie opeenvolgende hourglass netwerken die elk een dispariteitskaart genereren. Tijdens training worden de drie outputs gebruikt om de loss te berekenen van de huidige iteratie, tijdens het gebruik van PSMNet wordt enkel de laatste van deze outputs gebruikt.

### 3.1.3 Training

Dit netwerk kan getraind worden door de gegevens van een LIDAR-sensor te gebruiken als ground-truth (streefdoel). Het trainen van dit netwerk vergt veel rekenkracht. Daarom hebben de makers van het PSMNet drie reeds getrainde modellen op hun Github pagina staan, getraind op volgende datasets: KITTI 2015, Scene Flow en KITTI 2012.

## 3.2 Dispariteiten omzetten naar puntenwolken

Om een dispariteitskaart om te zetten naar een pseudo-lidar puntenwolk wordt een python script gebruikt. Dit script berekent voor elk punt in de dispariteitskaart ( $Y(u,v)$ ) een diepte ( $D(u,v)$ ) op basis van brandpuntsafstand ( $f_u$ ) en baseline (B) van de camera's volgens formule 3.1.

$$D(u,v) = \frac{f_u * B}{Y(u,v)} \quad (3.1)$$

Na het omzetten van de dispariteitskaart naar een dieptekaart wordt een 2D matrix gemaakt met de breedte en hoogte van de afbeelding. Deze matrix wordt vervolgens uitgebreid met een derde dimensie: diepte. Het maken en bewerken van deze matrices is eenvoudig met de numpy library in python. De resulterende matrix is nu een puntenwolk die de afstand tot de camera aangeeft. Dit komt echter niet overeen met een echte 3D-scene en moet dus aangepast worden. Deze aanpassing of projectie wordt gedaan aan de hand van de kitti\_util library en de calibratie-files die gemaakt zijn voor de KITTI dataset. Elk punt in deze matrix heeft 3 waarden die samen een plaats in de 3D ruimte definiëren. Als de diepte en de lens-eigenschappen van de camera gekend zijn is het mogelijk om de pixel-waarden in de afbeelding om te zetten in echte afmetingen (meters). Dit kan berekend worden door formule 3.2 toe te passen voor de x-coördinaat ( $P_x$ ) en formule 3.3 voor de y-coördinaat ( $P_y$ ). In deze formule zijn  $c_u$ ,  $c_v$ ,  $f_u$ ,  $f_v$ ,  $b_x$  en  $b_y$  eigenschappen van de gebruikte camera. Deze gegevens zijn terug te vinden in de calibratie files van de KITTI dataset onder dezelfde naam. Ook diepte op de betreffende pixel (D) beïnvloedt de transformatie. De getransformeerde coördinaten ( $P_u$  en  $P_v$ ) kunnen vervolgens gebruikt worden om de 3D bounding box te tekenen.

$$P_u = \frac{(P_x - c_u) * D}{f_u + b_x} \quad (3.2)$$

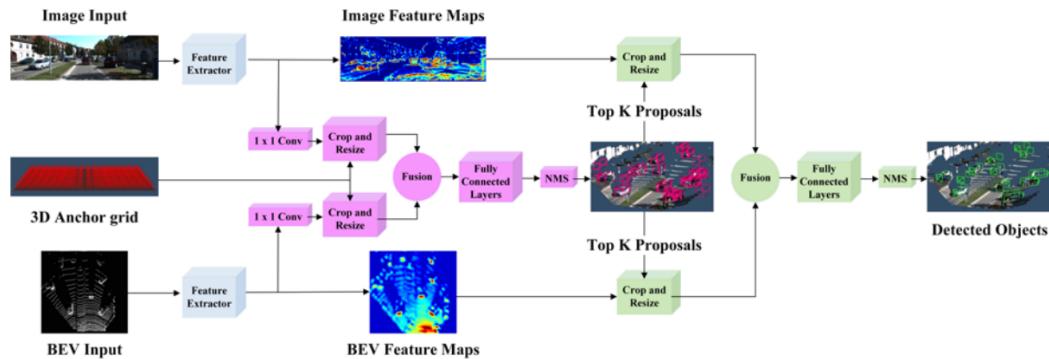
$$P_v = \frac{(P_y - c_v) * D}{f_v + b_y} \quad (3.3)$$

### 3.3 AVOD

Aggregate View Object Detection of AVOD (10) is een netwerk bedoeld voor de navigatie van autonome voertuigen. Het algoritme maakt gebruik van een combinatie van RGB-beelden en puntenwolken die gegenereerd zijn door een LIDAR. Alvorens dat de puntenwolk gebruikt kan worden, moet deze data eerst geconverteerd worden naar een bovenaanzicht of bird's eye view (BEV). Het netwerk bestaat uit 2 sub-netwerken: een region proposal network of RPN, dat instaat voor het zoeken naar potentiële 3D-objecten, en een detectienetwerk (Fig. 3.2). Met de hieruit voortkomende voorstellen kan het volgende sub-netwerk deze objecten proberen te identificeren. Deze aanpak laat toe om complexere en meer reken-intensieve netwerken te gebruiken voor de identificatie-stap zelf.

#### 3.3.1 RPN

De Regional Proposal Network of RPN maakt gebruik van de LIDAR-gegevens om regio's in de omgeving te vinden die het meeste kans maken om bv. een wagen te bevatten. Eerst wordt er een 3D BEV-kaart gegenereerd van de omgeving. Uit deze kaart wordt vervolgens in combinatie met de input afbeeldingen feature maps gegenereerd die potentiële detecties aangeven. De RPN maakt gebruik van 3 inputs: de feature map van de input afbeelding, die van de BEV en een vooraf gedefinieerde 3D anchor grid. Dit laatste is een 3D vlak voorzien van veel potentiële bounding



**Figuur 3.2:** Proces dat wordt doorlopen door AVOD (10). De roze blokken stellen het region proposal netwerk voor, de groene blokken zijn het detectienetwerk.

boxes of anchors waarover geïtereerd kan worden om detecties te maken. Deze techniek laat toe om meerdere (eventueel overlappende) objecten, op verschillende schaal te detecteren. De output van de RPN is een lijst van de  $K$  best scorende bounding boxes. Dit zijn de proposals of voorstellen die het detectienetwerk later gaat bekijken.

### 3.3.2 Detectie

Het detectienetwerk maakt gebruik van de output van de RPN, gecombineerd met de input afbeelding en BEV feature maps. Dit netwerk bestaat uit 3 fully connected layers met elk een grootte van 2048. Achteraf wordt de output van dit netwerk nog door een NMS of Non-Maximum Suppression module gehaald om overlappende detecties van hetzelfde object tegen te gaan.

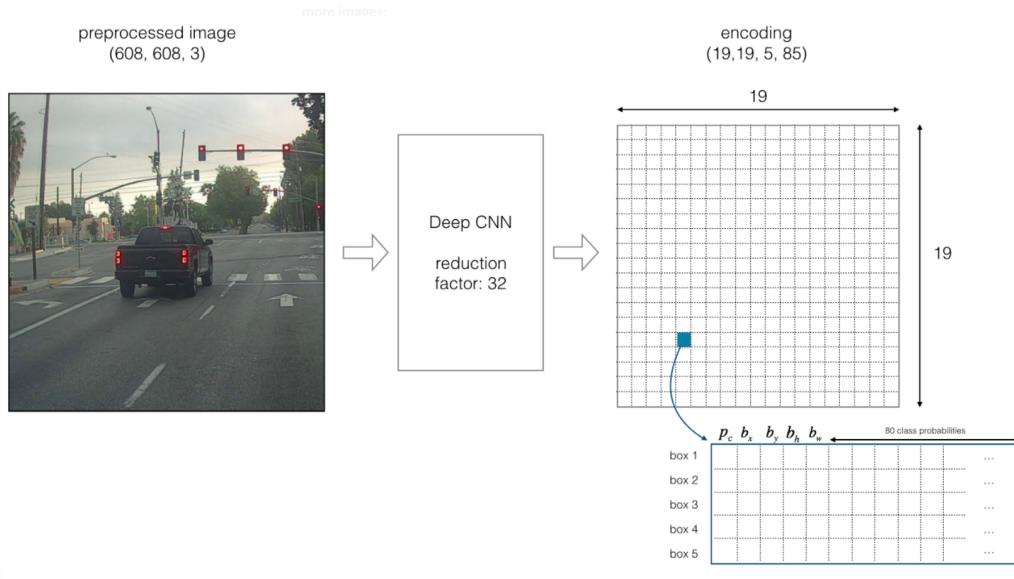
### 3.3.3 Training

Het trainen van AVOD vergt een krachtige GPU om het proces in een acceptabele tijd te kunnen voltooien. De modellen die gebruikt worden voor de experimenten in deze thesis zijn getraind op een Nvidia GTX 1080 Super GPU. Met deze hardware duurt een training sessie met 120.000 iteraties ongeveer 18 uur. Wanneer de LIDAR-gegevens wat veranderen van vorm (door gebruik van andere sensor of post-processing) moet het netwerk opnieuw getraind worden om de beste resultaten te verkrijgen.

## 3.4 YOLOv3

You Only Look Once of YOLO (12) is een CNN bedoeld voor object detectie. Het is niet het meest accurate netwerk, maar dit wordt goedgemaakt met de snelheid. Dit maakt het netwerk geschikt om te gebruiken in toepassingen zoals object detectie in videobeelden. Yolo maakt uitsluitend gebruik van convolutionele lagen en wordt daarom een fully convolutional network of FCN genoemd. Het netwerk maakt, zoals veel detectienetwerken, gebruik van anchors om de bounding boxes van

de detecties te bepalen. Dit laat toe om tegelijk meerdere objecten van verschillende klassen en schalen te detecteren. Figuur 3.3 illustreert de opbouw van een YOLO netwerk.



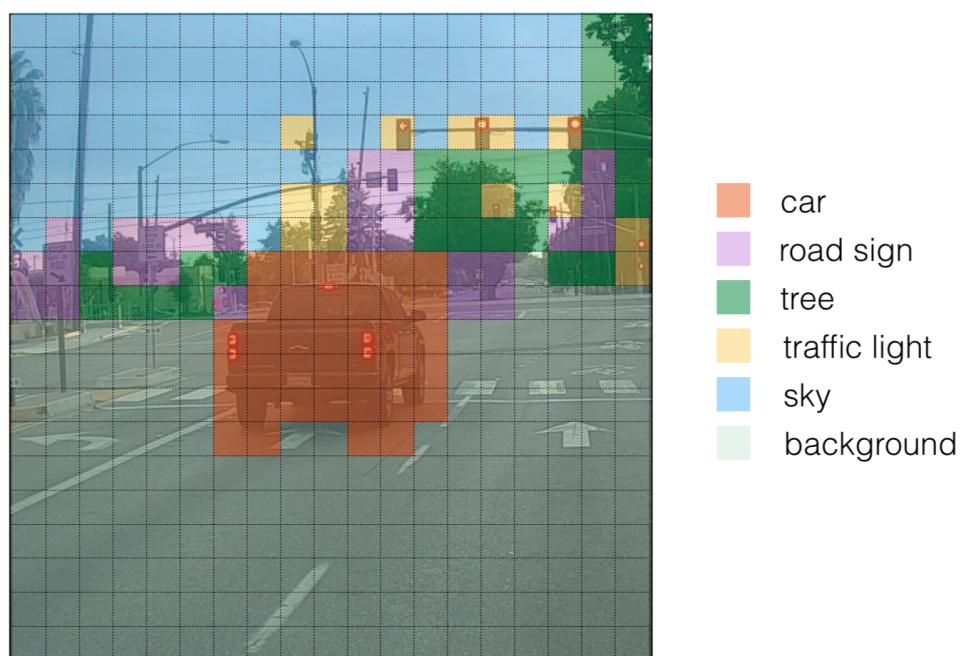
**Figuur 3.3:** De opbouw van het YOLO netwerk, inclusief de structuur van een feature vector. (11)

### 3.4.1 Feature extractor

De feature extractor van YOLO is een CNN die gebaseerd is op het Darknet framework (14). Dit framework is geschreven in C en CUDA en ondersteunt het gebruik van zowel CPU als GPU voor berekeningen. Het netwerk bestaat uit 53 convolutionele lagen en wordt daarom ook Darknet-53 genoemd. De CNN reduceert de input afbeelding met een factor 32. Het resultaat van dit netwerk is drie feature maps die elk een andere schaal hebben. Deze kunnen door de multi-scale detector gebruikt worden om objecten te herkennen.

### 3.4.2 Detector

Het detector gedeelte van YOLO werkt op meerdere schalen: multi-scale. Dit maakt de detectie schaal-onafhankelijk waardoor zowel verre als dichtbij zijnde objecten herkend worden. De output van dit netwerk is een matrix van vectoren met dezelfde afmetingen als die van de feature map. Elk van deze vectoren bevat 85 waarden: 4 om de positie en afmetingen van de bounding box te beschrijven, 1 die de "objectness score" voorstelt en de overige 80 waarden zijn een score per klasse. De objectness score reflecteert de zekerheid die het systeem heeft dat er zich in betreffende regio een object bevindt en dus geen achtergrond is. Door de hoogst scorende klasse uit de vector te gebruiken kan zoals in figuur 3.4 deze output gevisualiseerd worden.



**Figuur 3.4:** Visualisatie van de output van YOLO (voordat de bounding boxes worden bepaald). De kleur geeft de meest waarschijnlijke klasse aan voor die cel (11)

# **Hoofdstuk 4**

# **Experimenten**

## **4.1 Inleiding**

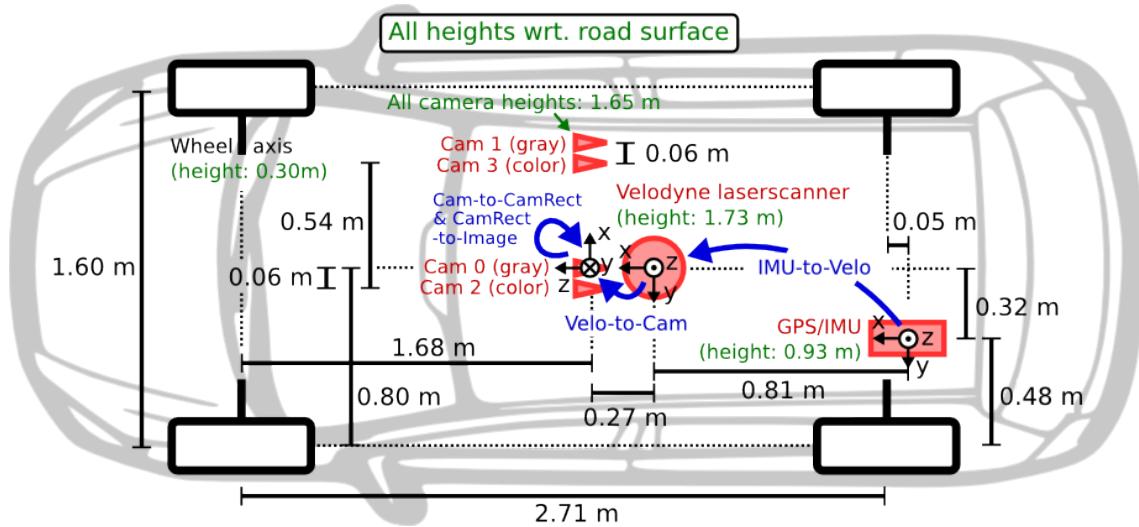
Dit hoofdstuk bespreekt de experimenten die in deze masterproef werden uitgevoerd. Volgende secties lichten de verschillende methodes toe die gebruikt gebruikt worden. Om de pseudo-LIDAR techniek te kunnen beoordelen, werd gebruik gemaakt van AVOD. Dit netwerk gebruikt LIDAR en RGB-afbeeldingen om objecten te detecteren. Door de echte LIDAR gegevens te vervangen met de gegenereerde pseudo-LIDAR kan bepaald worden hoe goed deze techniek werkt. Daarnaast wordt pseudo-LIDAR ook vergeleken met een 2D detectie techniek gecombineerd met een dispariteitskaart. Hiermee wordt de meerwaarde van het genereren van een puntenwolk aangetoond. De werking van voorgaande technieken werd in hoofdstuk 3 reeds besproken.

### **4.1.1 Opstelling**

Volgende experimenten zijn uitgevoerd op een deel van de KITTI-dataset (4). Deze set bestaat uit straatbeelden die vanop een auto zijn gemaakt. De KITTI-wagen is uitgerust met stereo-camera's, een LIDAR-sensor en GPS (Fig. 4.1). In deze experimenten wordt getracht de wagens in de straatbeelden te herkennen met behulp van deze sensors.

### **4.1.2 Evaluatie**

De KITTI-dataset is voorzien van een benchmark. Hiermee kunnen de resultaten van de evaluatie vergeleken worden met andere algoritmes. Deze benchmark wordt in volgende secties gebruikt om de resultaten te evalueren. De belangrijkste waarden die besproken worden bij de evaluatie van een techniek zijn de precisie en de recall. Deze waarden kunnen samen geplot worden op een precision-recall of PR-curve. Een grote oppervlakte onder de curve of AUC wijst op een goede precisie én recall, dus een goed resultaat. Naast deze curve geeft de gemiddelde precisie (AP) ook een idee van de performantie van een techniek. De technieken worden besproken volgens twee criteria: PR curve met 2D en met 3D bounding boxes. De 2D resultaten zeggen iets over de



Figuur 4.1: Schema van de sensors die aanwezig zijn op de KITTI-wagen (4)

kwaliteit van de detecties van objecten, terwijl de 3D resultaten meer zeggen over de kwaliteit van de object lokalisatie.

## 4.2 LIDAR

### 4.2.1 Opstelling

Voor afstandsbeperking wordt gebruik gemaakt van een Velodyne HDL-64E LIDAR-sensor (Fig 4.2). Dit is een accurate, maar dure sensor (ongeveer \$75.000). Deze sensor heeft een bereik tot 120m, zelfs op deze afstand maakt hij maximaal een fout van slechts 2cm. De sensor draait rond met een snelheid van 10 omwentelingen per seconde (of 600 RPM) zodat hij 10 beelden per seconde kan vormen van de omgeving ( $360^\circ$ ). Dit gebeurt met een hoekresolutie van  $0.08^\circ$  en hierbij genereert hij 2.2 miljoen punten per seconde. Om deze resolutie te bekomen maakt de sensor gebruik van 64 kanalen. Elk kanaal maakt gebruik van een verschillende golflengte van licht zodat deze gelijktijdig uitgestuurd kunnen worden zonder onderling te interfereren.

De KITTI-dataset stelt deze gegevens beschikbaar per scène. De object detectie gebeurt op het linker beeld van de stereo-camera.



Figuur 4.2: Velodyne HDL-64E LIDAR-sensor

### 4.2.2 Werkwijze

Het AVOD netwerk is getraind met behulp van de Velodyne LIDAR gegevens en linker RGB-beelden uit de KITTI dataset. De training set werd opgesplitst in twee gelijke delen: één deel om te trainen, het andere deel voor de evaluatie van het netwerk.

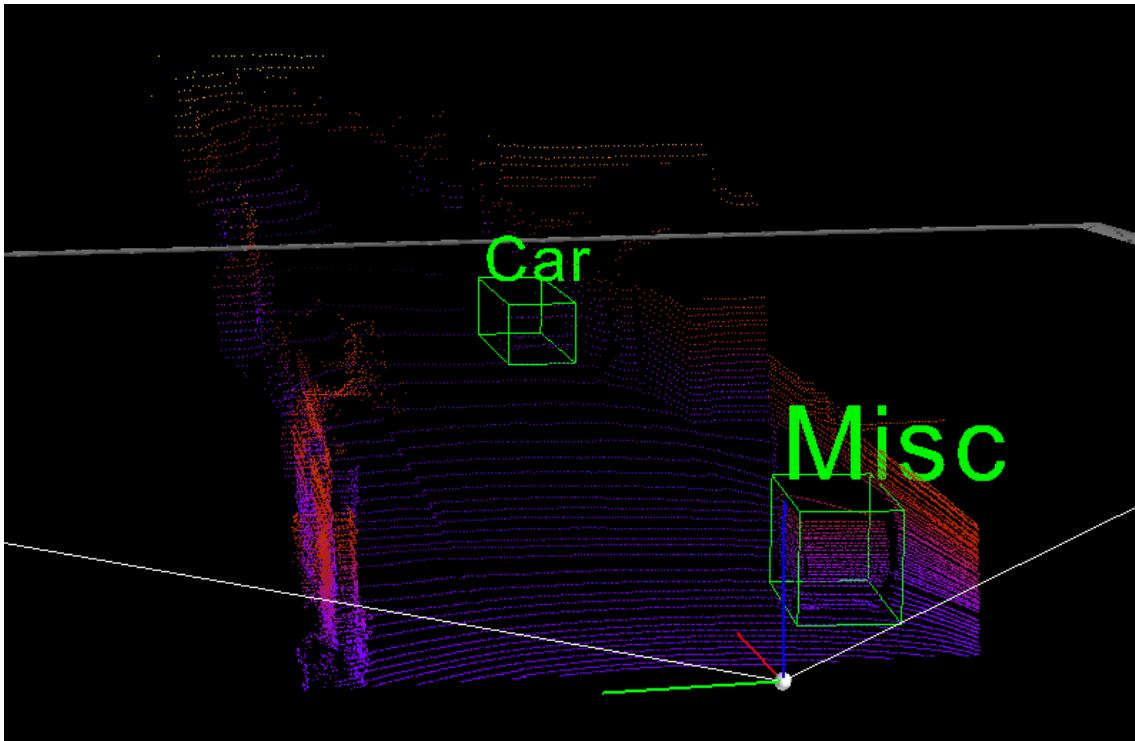
#### 4.2.2.1 Object detectie

Om de auto's te detecteren wordt gebruik gemaakt van het AVOD netwerk. Deze techniek maakt gebruik van twee subnetwerken: een RPN of Region Proposal Network en een object detectie netwerk. Bij de RPN wordt de data van een LIDAR-sensor vanuit vogelvlucht-perspectief (BEV)

gebruikt om regio's te vinden waar de kans groot is om een wagen te vinden. Op deze gevonden regio's kan dan het object detectie netwerk zijn werk doen om de auto's te identificeren.

#### 4.2.2.2 Object lokalisatie

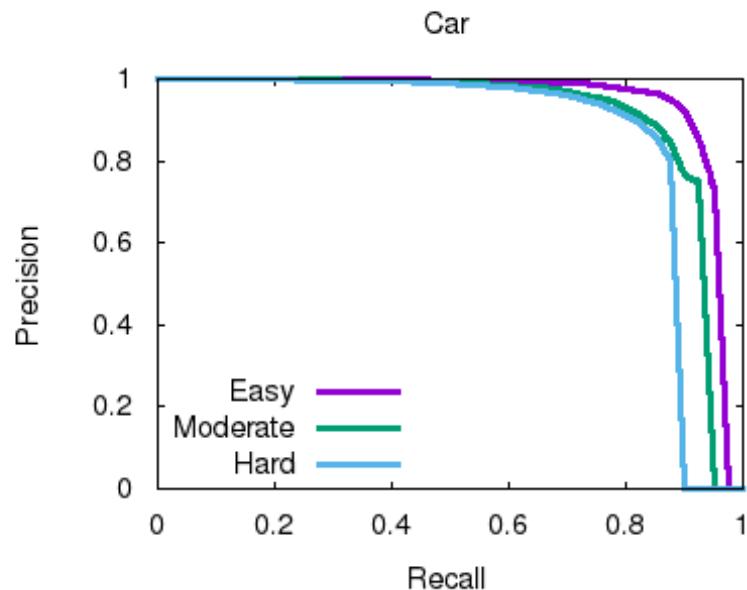
De LIDAR-sensor geeft een accurate afstandsmeting tot de gedetecteerde objecten. Door de bounding boxes, die gegenereerd werden in de object detectie stap, over de LIDAR-puntenwolk te tekenen kan de kwaliteit van de lokalisatie visueel beoordeeld worden. Figuur 4.3 illustreert deze visualisatie in de scène die gebruikt wordt in figuur 4.7.



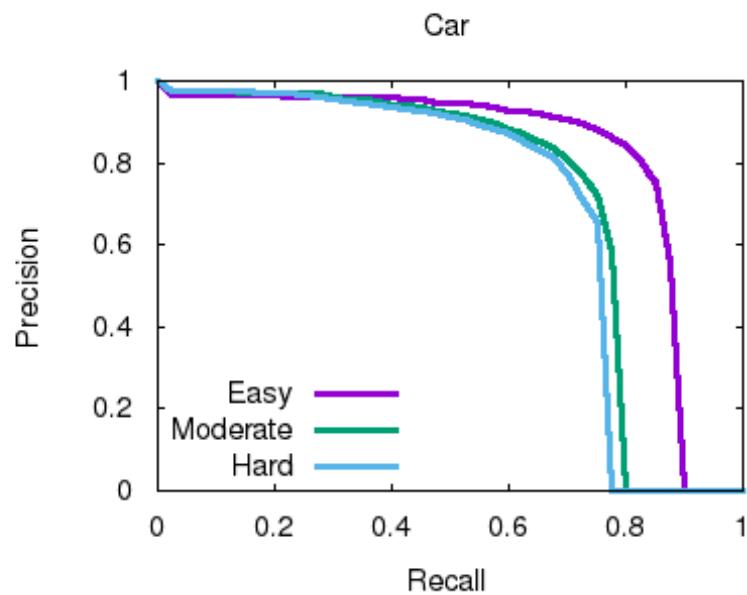
Figuur 4.3: De puntenwolk uit een Velodyne LIDAR-sensor

#### 4.2.3 Resultaten

De experimenten met het AVOD-netwerk met echte LIDAR gegevens geven goede resultaten: een grote AUC en AP voor elke moeilijkheid. Van gemakkelijk naar moeilijk bedraagt de AP voor 2D detecties 89.7%, 87.5% en 80.2%; voor 3D detecties is dit 77.0%, 67.9% en 67.3%. De moeilijkheid van de afbeelding heeft dus weinig impact op de betrouwbaarheid van de resultaten. In figuur 4.4 is te zien hoe deze opstelling presteert bij het herkennen van auto's. Figuur 4.5 illustreert de prestaties van deze opstelling bij het lokaliseren van auto's.



**Figuur 4.4:** Precision-recall plot van het AVOD-netwerk met echte LIDAR gegevens voor 2D detecties



**Figuur 4.5:** Precision-recall plot van het AVOD-netwerk met echte LIDAR gegevens voor 3D detecties

## 4.3 Pseudo-LIDAR

### 4.3.1 Opstelling

In plaats van gebruik te maken van een dure LIDAR-sensor, wordt de diepte bepaald door stereo-camera's. Deze zijn veel goedkoper, maar zijn minder precies voor het schatten van afstanden. De toestellen die gebruikt werden om de beelden te maken zijn twee 1.4 Megapixels Point Grey Flea 2 camera's (Fig 4.6). Deze staan op het dak van de wagen (1.6 meter boven de grond), 60 cm uit elkaar.



**Figuur 4.6:** Point Grey Flea 2 (FL2-14S3M-C) camera

### 4.3.2 Werkwijze

In deze sectie worden twee methodes beschreven om de dispariteit in te schatten uit de stereo-beelden. De eerste methode is met behulp van een neuraal netwerk of CNN. Dit belooft meer accurate schattingen, ten koste van hogere hardware vereisten. Ten tweede wordt er gebruik gemaakt van de algoritmen die Open CV ter beschikking stelt om de dispariteit te berekenen (SGBM). Ook deze methode maakt gebruik van AVOD om de detecties te maken. Omdat de resulterende pseudo-LIDAR puntenwolken onderling zo verschillen naargelang de gebruikte techniek, moet het AVOD netwerk bij elk van deze experimenten hertraind worden op de nieuwe data.

#### 4.3.2.1 Dispariteit inschatten met CNN (PSMNet)

Bij de initiële training van het PSMNet model werd gebruik gemaakt van de gegevens uit de Scene Flow dataset. Het getrainde model werd vervolgens bijgetraind met de LIDAR-gegevens uit de KITTI-dataset om dit meer te specialiseren. Deze gegevens worden beschouwd als de *groundtruth*, die gebruikt wordt als streefdoel voor het model tijdens training.

Een klassieke dispariteitsschatter zoekt overeenkomstige pixels in de stereo-beelden. Dit kan moeilijk zijn bij effen oppervlakken, occlusiegebieden en verre objecten. PSMNet kijkt verder dan enkel op pixelniveau. Door naar de globale context van de afbeelding te kijken kan de ruis die veroorzaakt is door deze moeilijke situaties verminderd worden. In sectie 3.1 werd PSMNet uitgebreid besproken. In figuur 4.7 wordt het linker beeld van de stereo-camera vergeleken met de output van PSMNet. Hoe lichter de kleur van de pixel, hoe dichter het overeenkomend punt bij de camera staat.



**Figuur 4.7:** Kaart van dispariteiten (onderaan), gegenereerd met PSMNet (één van de beelden wordt bovenaan weergegeven)

#### 4.3.2.2 Dispariteit inschatten met Open CV

Deze aanpak maakt gebruik van Semi-Global Block-Matching of SGBM om de dispariteit in te schatten. Dit is een variatie op SGM, een techniek voorgesteld door H. Hirschmuller (15), aangepast door Open CV. Dit is een populair algoritme omdat het relatief snel en accuraat werkt. Deze techniek matcht blokken uit de stereo-beelden met elkaar in plaats van het gebruikelijke pixel-matchen. Dit reduceert de ruis in de disparitieskaart, maar geeft wel een ruwer beeld van de scène. Geleidelijke veranderingen in afstand zijn abrupter waardoor de inschatting precisie verliest. Dit is duidelijk te zien wanneer de disparitieskaarten van figuren 4.8 en 4.7 vergeleken worden met elkaar. Dit is ook te zien in de gegenereerde puntenwolken van figuren 4.10 en 4.9.

#### 4.3.2.3 Dispariteit omzetten naar pseudo-LIDAR

In essentie bevatten dieptekaarten en LIDAR-puntenwolken dezelfde informatie. Het is dus relatief eenvoudig om een dieptekaart om te zetten in een puntenwolk. Hoe dit werkt is te vinden in sectie 3.2. Hoewel er geen extra informatie wordt toegevoegd aan de data zou de LIDAR-voorstelling betere resultaten opleveren dan rechtstreeks detectie te doen met de dieptekaart.

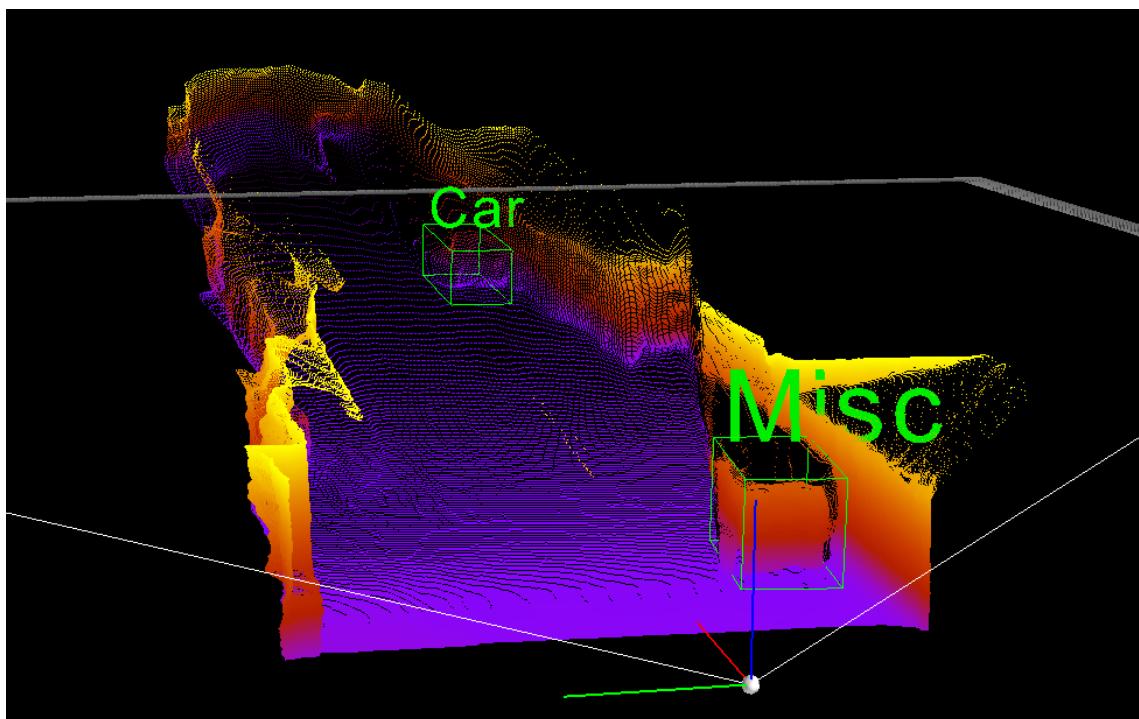
#### 4.3.2.4 Object detectie en lokalisatie

De detectie en lokalisatie van wagens gebeurt op dezelfde wijze als bij de techniek met echte LIDAR-gegevens (zie sectie 4.2.2.1 en 4.2.2.2). In plaats van echte LIDAR wordt nu gebruik ge-

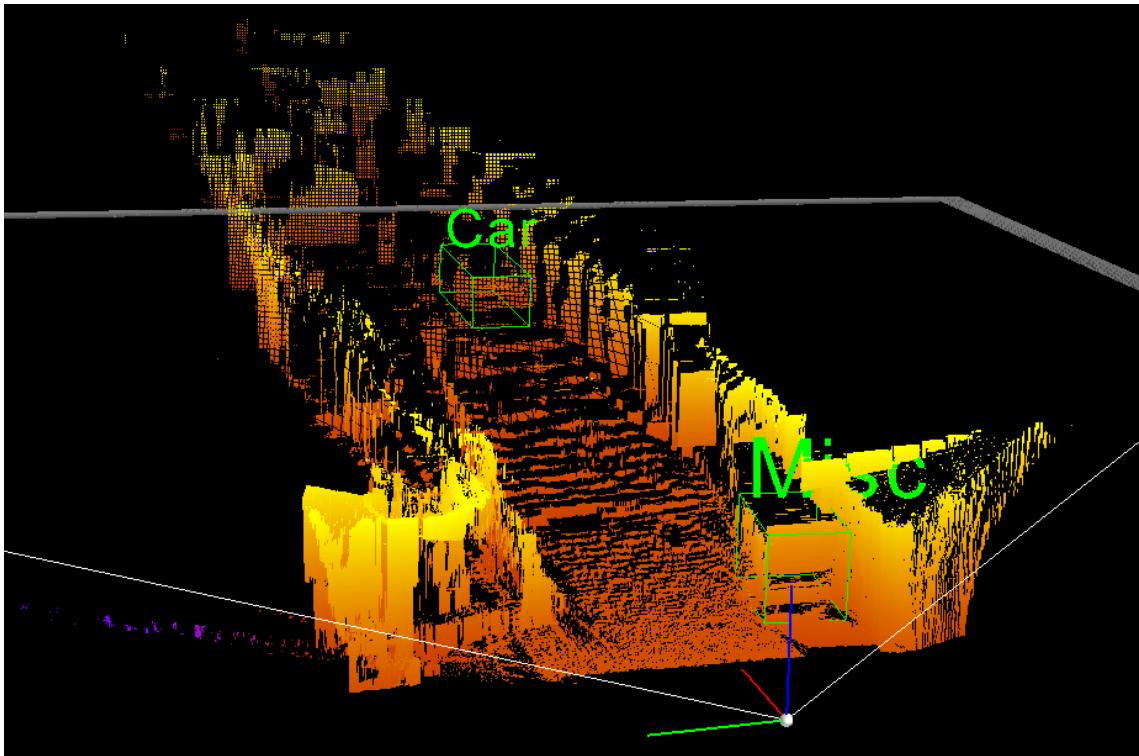


**Figuur 4.8:** Kaart van dispariteiten (onderaan), gegenereerd met Open CV (één van de beelden wordt bovenaan weergegeven)

maakt van de gegenereerde pseudo-LIDAR die in voorgaande sectie besproken is (sectie 4.3.2.3).



**Figuur 4.9:** Puntenwolk die gegenereerd is uit de output van PSMNet (9)



**Figuur 4.10:** Puntenwolk die gegenereerd is uit de output van Open CV (SGBM)

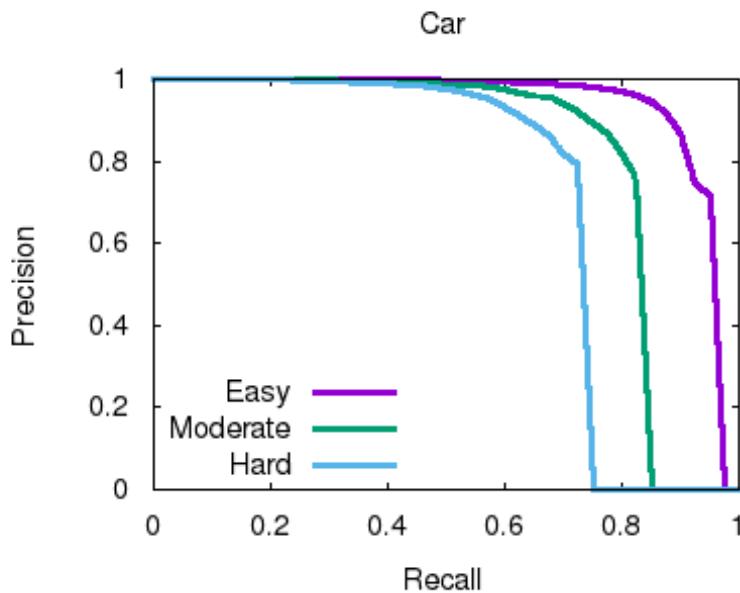
### 4.3.3 Resultaten

#### 4.3.3.1 CNN

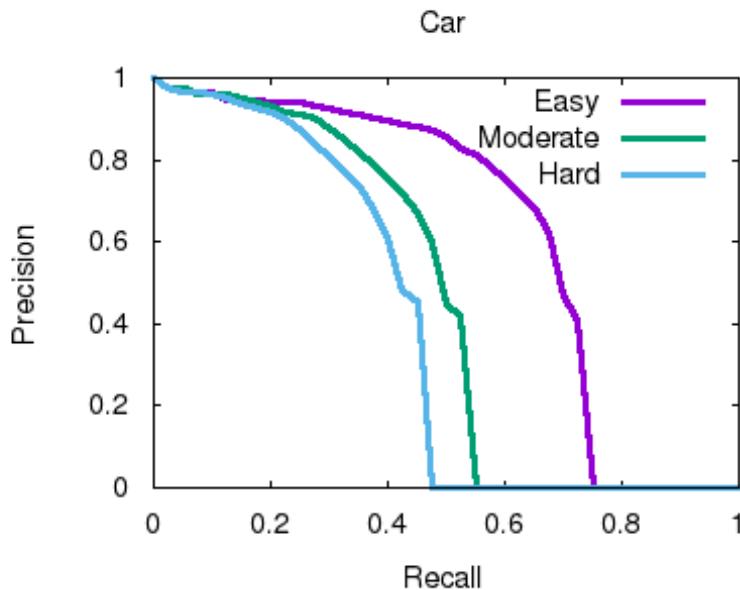
Deze techniek vereist behoorlijk veel rekenkracht en maakt dit dus minder praktisch voor mobiele toepassingen. De resultaten zijn echter wel goed: de meeste bounding boxes worden juist getekend en liggen dicht bij de ground truth (bepaald door LIDAR-gegevens). De AUC en AP van deze resultaten zijn hoog bij elke moeilijkheid. De AP gaat wel wat achteruit naarmate de moeilijkheid toeneemt: voor 2D is dit 89.2%, 79.2% en 70.0% (Fig. 4.11) en voor 3D is dit 61.9%, 44.6% en 39.0% (Fig. 4.12).

#### 4.3.3.2 Open CV

Het inschatten van afstand tot het object gaat sneller met deze techniek. Het valt echter op dat er inschattingsfouten voorkomen. Deze zijn amper te merken op dichtbij zijnde objecten, maar naargelang het object verder weg is van de camera's, neemt deze fout toe. In figuur 4.13 is te zien hoe de precisie van de diepteschatting afneemt als de afstand toeneemt. Als gevolg hiervan is er een opvallende fout op te merken bij de detectie van de auto. Deze fout verklaart de beduidend minder goede P/R curve voor 3D detecties (Fig. 4.15). Als de oriëntatie overeen komt, maar de afstand te veel verschilt zal de 3D detectie niet als juist beschouwd worden. De bekomen AP voor 3D detecties zijn dan ook maar 20.0%, 16.4% en 13.3%. De AUC en AP voor de 2D detecties zijn

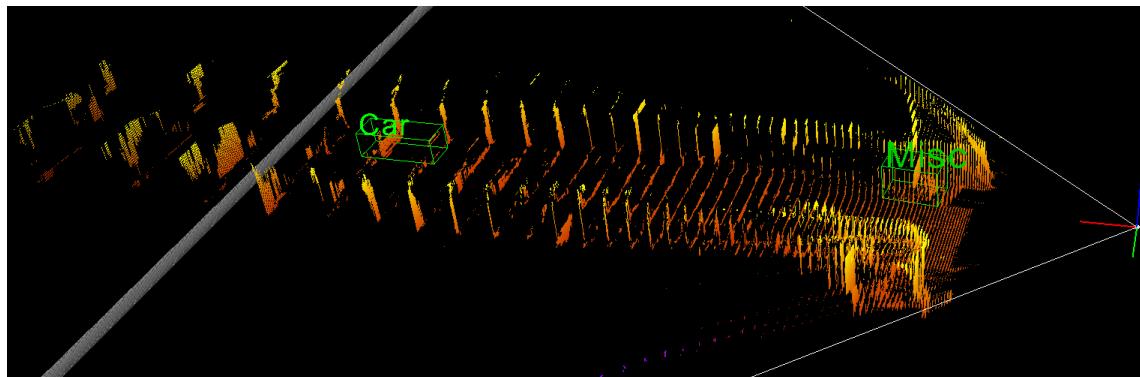


**Figuur 4.11:** Precision-recall plot van het AVOD-netwerk met pseudo-LIDAR gegevens, gegenereerd met PSMNet dispariteiten

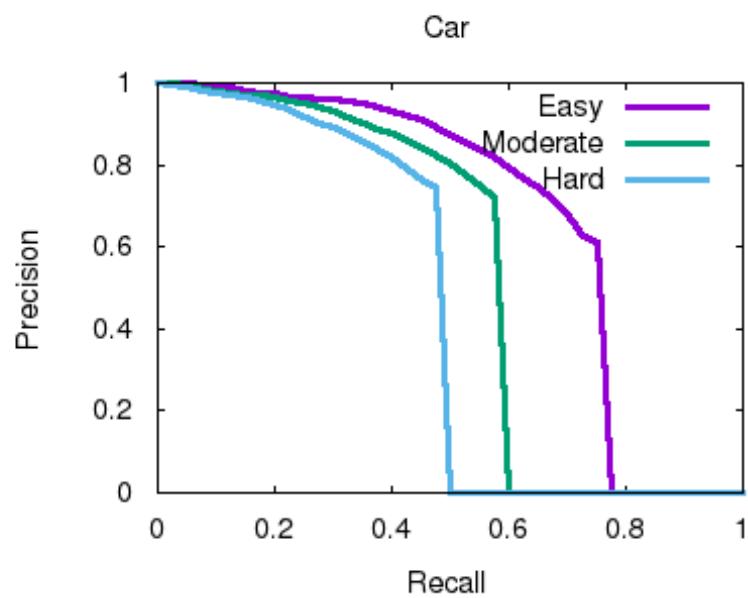


**Figuur 4.12:** Precision-recall plot van het AVOD-netwerk met pseudo-LIDAR gegevens, gegenereerd met PSMNet dispariteiten, voor 3D detecties

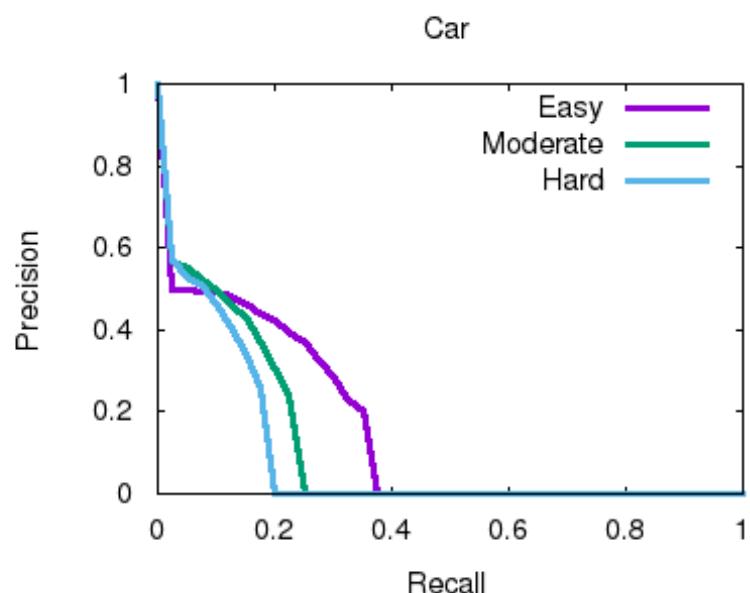
echter ook minder goed: een AP van 65.4%, 50.5% en 42.1% (Fig. 4.14). Dit komt omdat de input van het detectie netwerk in AVOD afhankelijk is van de output van het RPN netwerk. Wanneer de aangeleverde gegevens over de afstand minder accuraat zijn, zal dit invloed hebben op de performantie van het detectie netwerk.



Figuur 4.13: Fouten bij het inschatten van diepte met SGBM



Figuur 4.14: Precision-recall plot van het AVOD-netwerk met pseudo-LIDAR gegevens, gegenereerd met SGBM dispariteiten



**Figuur 4.15:** Precision-recall plot van het AVOD-netwerk met pseudo-LIDAR gegevens, gegenereerd met SGBM dispariteiten, voor 3D detecties.

## 4.4 Dispariteitskaart

### 4.4.1 Opstelling

In plaats van de omweg te nemen via pseudo-LIDAR wordt met deze techniek enkel gebruik gemaakt van de berekende dispariteiten uit de stereo-beelden. Omdat een dispariteitskaart in essentie dezelfde informatie bevat als pseudo-LIDAR, zou dit vergelijkbare resultaten moeten kunnen opleveren. Het overslaan van deze stap zou de berekentijd van een detectie verminderen, wat toelaat om meer beelden per seconde te verwerken.

### 4.4.2 Werkwijze

Zoals bij vorige methodes is de eerste stap het berekenen van een dispariteitskaart. Uit vorige experimenten blijkt dat PSMNet de betere techniek is om dit te verwezenlijken. Vervolgens wordt het YOLOv3 netwerk toegepast op het linker beeld van de stereo-beelden om de wagens te detec-teren. Details over het YOLO netwerk zijn te vinden in sectie 3.4. De locatie van de 3D bounding box wordt bepaald door de dispariteiten op de betreffende coördinaten te transformeren naar 3D-coördinaten. Deze transformatie is dezelfde als die om pseudo-LIDAR te genereren (sectie 4.3.2.3). Maar in plaats van elk punt in de omgeving te moeten transformeren worden enkel de relevante punten berekend wat de rekentijd aanzienlijk reduceert.

#### 4.4.2.1 Object detectie

Voor de detectie van objecten wordt op zich in eerste instantie, geen gebruik gemaakt van de dispariteitskaarten van PSMNet. Deze worden achteraf wel gebruikt om de detectie van deels bedekte auto's te verbeteren. YOLO tekent de bounding box namelijk enkel over het (herkenbaar) zichtbare deel van de wagen en schat niet in hoeveel van de wagen zichtbaar is. Door de omgeving van de bounding box te bestuderen in de dispariteitskaart kan bepaald worden waar er occlusie is. Op basis van de afmetingen van de bounding box wordt er geschat hoeveel van de wagen niet zichtbaar is. In figuur 4.16 wordt de occlusiecorrectie geïllustreerd.

#### 4.4.2.2 Object lokalisatie

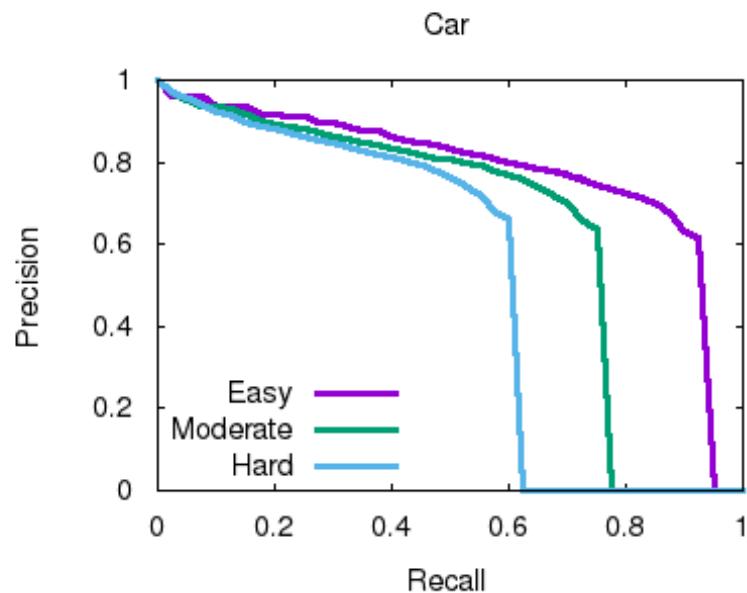
De lokalistatie van gedetecteerde objecten gebeurt door relevante pixels op te zoeken in de dis-pariteitskaart. Dit resulteert in de berekening van de afstand van de camera's tot aan het object. In plaats van een puntenwolk te genereren van de volledige omgeving, worden enkel de relevante punten berekend en gebruikt voor de lokalisatie. Dit reduceert aanzienlijk de rekenkracht vereist om de 3D bounding box te bepalen.



**Figuur 4.16:** De groene en gele rechthoeken zijn de bounding boxes die verkregen zijn uit het YOLO netwerk. De rode rechthoeken zijn de bounding boxes die achteraf aangepast zijn, rekening houdend met occlusie. De blauwe kaders zijn detecties die wegens de afstand niet relevant zijn.

#### 4.4.3 Resultaten

Het YOLO netwerk levert goede resultaten op voor 2D detecties. Er is een significant kwaliteitsverschil tussen moeilijkheidsgraden van de detecties. De bekomen AP voor 2D detecties zijn 76.1%, 61.8% en 53.5%. De 3D resultaten zijn nog niet beschikbaar. Meer hierover in de sectie toekomstig werk (sectie 4.5.3).



**Figuur 4.17:** Precision-recall plot van het YOLOv3 netwerk

## 4.5 Analyse

In deze sectie worden de resultaten van voorgaande experimenten vergeleken. Er worden mogelijke oorzaken voorgesteld die de verkregen resultaten kunnen verklaren. Verder wordt er een blik geworpen op de toekomst van dit project: zaken die nog niet af zijn en mogelijke verbeteringen.

### 4.5.1 LIDAR vs. Pseudo-LIDAR

Voor deze vergelijking werden 3 experimenten uitgevoerd: één met LIDAR gegevens en twee door pseudo-LIDAR te genereren. Deze experimenten maken allemaal gebruik van het AVOD netwerk, een 3D detectie netwerk dat gebruik maakt van LIDAR om auto's te detecteren. Pseudo-LIDAR wordt gegenereerd uit dispariteitskaarten. Hiervoor zijn twee technieken gebruikt die sterk verschillen in complexiteit en dus rekentijd. Ten eerste werd PSMNet gebruikt, een neuraal netwerk dat goede dispariteitskaarten inschat en hierbij rekening houdt met de globale context van de afbeelding. Dit is echter een zeer rekenintensieve techniek die een GPU met minstens 4GB video RAM vereist om te gebruiken. Als tweede techniek werd SGBM gebruikt, een schatter die in Open CV geïmplementeerd is. Deze techniek is beduidend minder rekenintensief, maar levert dan ook minder accurate resultaten op. De kwaliteit van de dispariteitskaart heeft een grote impact op de prestaties van het AVOD netwerk.

#### 4.5.1.1 Object herkenning

Op vlak van de 2D herkenning van objecten zijn de technieken vergelijkbaar. Het valt op dat bij LIDAR de kwaliteit van de detecties minder lijdt onder de factoren die een scène moeilijk maken. Deze factoren zijn de afstand tot het object en de mate van occlusie en truncatie van het object.

Wat opvalt is dat gemiddeld de score van juiste detecties hoger ligt wanneer LIDAR gebruikt wordt. Er gebeuren ook minder valse detecties. Dit fenomeen is te wijten aan het feit dat er ruis zit in de dispariteitskaart. Deze ruis kan als een detectie gezien worden. Figuur 4.18 illustreert een voorbeeld van het verschil in score en detecties tussen de technieken.



**Figuur 4.18:** Gevisualiseerde detecties van AVOD met LIDAR (links) en Pseudo-LIDAR met PSMNet (rechts). Pseudo-LIDAR scoort lager en maakt een valse detectie (rechts)

#### 4.5.1.2 Object lokalisatie

Het inschatten van afstanden voor de lokalisatie van objecten is zeer accuraat met echte LIDAR. In de KITTI dataset gebruikt men de LIDAR gegevens zelfs als de ground truth. Hierdoor kunnen deze gegevens perfect gebruikt worden als vergelijkingspunt. De diepte correct kunnen inschatten is uiteraard een belangrijk aspect bij het lokaliseren van objecten. Daarom speelt de kwaliteit van de gebruikte dispariteitschatter een grote rol bij de kwaliteit van de resultaten. Bij SGBM wordt de fout groter naarmate het object verder van de camera staat. Hoewel dit eigen is aan het concept van stereo-matching zelf, slaagt PSMNet erin om deze fout aanzienlijk te reduceren.

#### 4.5.2 Pseudo-LIDAR vs. dispariteitskaart

Door de grote impact die de kwaliteit van de dispariteitskaart heeft op de resultaten, zullen nu enkel nog de dispariteitskaarten van PSMNet gebruikt worden voor verdere vergelijkingen. De twee technieken die besproken worden zijn pseudo-LIDAR en YOLOv3 (met correcties aan de hand van dispariteitskaarten). Beide experimenten maken gebruik van PSMNet voor het genereren van de dispariteiten.

##### 4.5.2.1 Object herkenning

De resultaten van de 2D object herkenning zijn beter bij pseudo-LIDAR dan bij YOLO. Dit valt wel met een korrel zout te nemen: YOLO detecteert wagens die niet in de ground truth staan van KITTI. Deze detecties worden dan door de benchmark beschouwd als een valse detectie (false positive) en beïnvloeden de PR-curve negatief. In realiteit zou de precision hoger liggen en meer lijken op de PR-curve van pseudo-LIDAR. Verder is op te merken dat YOLO minder goed werkt wanneer er een wagen grotendeels bedekt is door een andere wagen. In deze situatie zijn er twee veel voorkomende uitkomsten. Ofwel wordt de wagen gedetecteerd, maar is de detectie te smal omdat enkel het zichtbaar stuk wordt gelabeld. Dit wordt opgelost door de dispariteitskaart te gebruiken om occlusiecorrectie te doen. Ofwel wordt de wagen helemaal niet gedetecteerd en beschouwt YOLO de bedekte wagen als een deel van de wagen die ervoor staat. Deze situatie maakt de bounding box van de voorste wagen te groot, wat de 3D detectie negatief kan beïnvloeden.

##### 4.5.2.2 Object lokalisatie

Dit deel van het onderzoek is nog lopende. Momenteel worden enkel de afstand tot de detectie en de afmetingen van de detectie in beschouwing genomen bij het bepalen van een 3D bounding box. De hoek waaronder de auto staat wordt nu niet berekend en heeft een vaste waarde. Het berekenen van deze hoek is echter complex, maar is essentieel om aanvaardbare resultaten te kunnen verkrijgen.

### 4.5.3 Toekomstig werk

Om te beginnen zou het lokalisatie gedeelte van de YOLO techniek verder afgewerkt moeten worden. Dit zou gerealiseerd kunnen worden door de detector aan te passen zodat hij delen van een wagen kan herkennen. Op deze manier kan dan de voorkant/achterkant en zijkant van de auto gevonden worden. Als de verandering in diepte over deze onderdelen bekend is, is het mogelijk om een oriëntatie te bepalen. Een andere aanpak zou kunnen zijn dat men enkel de pixels binnen de bounding box van de detectie transformeert naar 3D coördinaten. Zo wordt er een klein pseudo-LIDAR beeldje gevormd per detectie, dat dan op zijn beurt kan gebruikt worden om de oriëntatie te bepalen. Dit zou nog steeds minder rekenintensief zijn dan steeds opnieuw de volledige scène te moeten transformeren.

Een logische volgende stap zou zijn om het aantal te detecteren klassen uit te breiden met fietsers en voetgangers. Zowel AVOD als YOLO hebben de mogelijkheid om deze detecties uit te voeren. In het originele pseudo-LIDAR project (2) zijn er reeds testen uitgevoerd met deze klassen. Deze klassen zouden moeilijker zijn om te detecteren omdat ze lastiger te onderscheiden zijn in de pseudo-LIDAR gegevens. Hierdoor zou YOLO een beter resultaat kunnen geven dan AVOD wegens de manier waarop deze technieken werken. Een detectie bij AVOD komt enkel tot stand als de pseudo-LIDAR gegevens een potentieel zien in een gebied. Dit zou minder het geval zijn bij fietsers en voetgangers waardoor er minder gedetecteerd zouden worden. Omdat YOLO de detecties enkel baseert op RGB-beelden en pas achteraf rekening houdt met de dispariteitskaarten zou het hier minder last van kunnen hebben.

# **Hoofdstuk 5**

## **Conclusies**

### **5.1 Overzicht**

In deze masterproef werd een vergelijking gemaakt tussen allerlei technieken om objecten te detecteren en lokaliseren. Eerst werd LIDAR gebruikt in combinatie met AVOD om detecties te maken. Dit gaf een vergelijkingspunt voor de resultaten van de technieken die later aan bod kwamen.

Vervolgens werd de eerste alternatieve techniek besproken: pseudo-LIDAR. Hiervoor zijn twee experimenten uitgevoerd waarbij de variabele de dispariteitsschatter was. In het eerste experiment werd PSMNet gebruikt, een accurate maar rekenintensieve schatter die gebruik maakt van een neuraal netwerk. Het tweede experiment maakte gebruik van SGBM, een minder accurate maar snellere schatter die in Open CV werd geprogrammeerd.

Tot slot werd afgestapt van het LIDAR-concept en werd de diepte-informatie rechtstreeks gehaald uit de dispariteitskaart van PSMNet. Hierdoor was het niet meer mogelijk om AVOD te gebruiken omdat dit netwerk LIDAR-gegevens verwacht om detecties te kunnen maken. In plaats daarvan werd het YOLO netwerk gebruikt om de initiële detecties te maken. Deze werden vervolgens verbeterd, gebruikmakend van de diepte gegevens van de dispariteitskaart. Dit maakte het mogelijk om niet relevante detecties weg te filteren en om het occlusiegebied van deels bedekte objecten in te schatten.

De bestanden van het project staan op de github pagina:

<https://github.com/MaximAelterman/Masterproef>

## 5.2 LIDAR vs. Pseudo-LIDAR

Zoals verwacht is pseudo-LIDAR geen perfecte vervanger voor het gebruik van een LIDAR-sensor. De techniek biedt echter wel een vergelijkbaar alternatief voor slechts een fractie van de kost. De kwaliteit van de inschatting van de afstand tot objecten neemt af met het toenemen van die afstand. In vele toepassingen neemt het belang van de nauwkeurigheid van de afstandsbeperking logischerwijs ook af met de afstand tot het object. Het is mogelijk om voor low-end toepassingen een minder goede stereo-matcher te gebruiken om een dispariteitskaart te genereren, maar dit reduceert de kwaliteit van de resultaten aanzienlijk. Pseudo-LIDAR lijkt een optie te zijn om autonome wagens betaalbaarder te maken indien men erin slaagt om zeer accurate dispariteitskaarten te genereren.

## 5.3 Pseudo-LIDAR vs. dispariteitskaart

In essentie bevat een dispariteitskaart dezelfde informatie (mits enkele berekeningen) als zijn pseudo-LIDAR tegenhanger. Elke pixel in de dispariteitskaart wordt getransformeerd naar een positie in 3D-coördinaten. Dit maakt de voorstelling van deze gegevens interessanter voor de lokalisatie van objecten in een beeld. Bovendien laat de pseudo-LIDAR techniek toe om bestaande LIDAR gebaseerde netwerken te gebruiken om detecties te maken. Deze netwerken behoren tot de best scorende technieken volgens de KITTI benchmark.

# Bibliografie

- [1] A. K. Alexander Mordvintsev, “Depth map from stereo images.” [https://opencv-python-tutroals.readthedocs.io/en/latest/py\\_tutorials/py\\_calib3d/py\\_depthmap/py\\_depthmap.html](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html).
- [2] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Weinberger, “Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving,” in *CVPR*, 2019.
- [3] S. Gargoum and K. El-Basyouny, “Automated extraction of road features using lidar data: A review of lidar applications in transportation,” in *2017 4th International Conference on Transportation Information and Safety (ICTIS)*, pp. 563–574, Aug 2017.
- [4] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [5] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation,” in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134.
- [6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and L. Zitnick, “Microsoft coco: Common objects in context,” in *ECCV*, European Conference on Computer Vision, September 2014.
- [7] S. Saha, “A comprehensive guide to convolutional neural networks - the eli5 way,” 2018. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.
- [8] M. Sirsat, “What is confusion matrix and advanced classification metrics?,” 2019. <https://manisha-sirsat.blogspot.com/2019/04/confusion-matrix.html>.
- [9] J.-R. Chang and Y.-S. Chen, “Pyramid stereo matching network,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5410–5418, 2018.
- [10] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, “Joint 3d proposal generation and object detection from view aggregation,” *IROS*, 2018.

- [11] "Yolo v3 theory explained," 2019. <https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193>.
- [12] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.
- [13] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-li{dar}++: Accurate depth for 3d object detection in autonomous driving," in *International Conference on Learning Representations*, 2020.
- [14] J. Redmon, "Darknet: Open source neural networks in c." <http://pjreddie.com/darknet/>, 2013–2016.
- [15] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, 2008.
- [16] A. Aslam and M. S. Ansari, "Depth-map generation using pixel matching in stereoscopic pair of images," *CoRR*, vol. abs/1902.03471, 2019.

FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN  
CAMPUS DE NAYER SINT-KATELIJNE-WAVER  
J. De Nayerlaan 5  
2860 SINT-KATELIJNE-WAVER, België  
tel. + 32 15 31 69 44  
[iw.denayer@kuleuven.be](mailto:iw.denayer@kuleuven.be)  
[www.iw.kuleuven.be](http://www.iw.kuleuven.be)

