KU LEUVEN

FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN

CAMPUS DE NAYER SINT-KATELIJNE-WAVER

Visie voor semantische robotnavigatie in ziekhuisgangen

Olivier VAN DEN EEDE

Promotor(en): Prof. dr. ir. Toon Goedemé

Co-promotor(en): Filip Reniers

Masterproef ingediend tot het behalen van de graad van master of Science in de industriële wetenschappen: Electronica-ICT afstudeerrichting ICT

Academiejaar 2018 - 2019

Inhoudsopgave

Lijst van figuren

Acroniemen

AGV Autonoom Geleid Voertuig. 1

CNN Convolutional Neural Network. 4, 5, 6, 7, 8

EM Expectation-Maximization. 5

HOG Histogram of Oriented Gradients. 3

HSI Hue Saturation Intensity. 3

LSTM RNN Long Short Term Memory Recurrent Neural Network. 6

OCR Optical Character Recognition. 2, 3

RANSAC Random Sample Consensus. 3, 6

ReLU Rectified Linear Unit. 5

RGB Rood Groen Blauw. 1, 2, 4, 7

ROI Region Of Interest. 5

ROLO Recurrent YOLO. 6

SIFT Scale-Invariant Feature Transform. 3, 5, 8

SVM Support Vector Machine. 3, 6

YOLO You Only Look Once. 5, 6, 9

Hoofdstuk 1

Probleemstelling

Ziekenhuizen kampen al langer met personeelstekorten en een hoge werkdruk voor het zorgpersoneel. Een deel van dit probleem komt doordat ze ook instaan voor de textiellogistiek en de goederenstroom. Dit probleem zou aangepakt kunnen worden met behulp van automatisatie van de transporten van textiel, karren en bedden. Deze automatisatie staat momenteel nog niet zo ver, omdat vergeleken met de industrie het moeilijk is om de volledige infrastructuur aan te passen, en deze aanpassingen meestal niet overweg kunnen met het bestaande logistiek materiaal. Een Autonoom Geleid Voertuig (AGV) zou gebruikt kunnen worden om het transport van karren en bedden te automatiseren binnen de logistieke gangen van het ziekenhuis.

Dit voertuig moet vanzelf kunnen navigeren in de gangen van een ziekenhuis en weten waar het zich op elk moment bevind. Om dit te realiseren wordt het voertuig uitgerust met een aantal sensoren en een Rood Groen Blauw (RGB) camera om de omgeving te observeren. Voor navigatie beschikt het AGV over een semantische kaart. Dit is een kaart waarop aangeduid staat wat voor objecten er te zien zijn (muren, deuren, bordjes, verlichting, ..) samen met de afmetingen, positie en oriëntatie van deze tags.

Het doel van deze masterproef is het onderzoeken welke objecten/features er aanwezig zijn in de logistieke gangen van een ziekenhuis en op basis daarvan beeldverwerkingstechnieken te zoeken die geschikt kunnen zijn voor detectie en tracking van deze objecten. Deze detecties kunnen dan gebruikt worden om de locatie van de robot te bepalen op basis van de kaart. Vervolgens is het de bedoeling dat de robot vertrekt vanop een gekende locatie en d.m.v. zijn kaart en de objecten die hij detecteert in zijn omgeving autonoom kan navigeren naar een eindpunt.

Hoofdstuk 2

Literatuurstudie

2.1 Indoor navigatie & visie

Op visie gebaseerde navigatie is een onderwerp dat zeer vaak onderzocht wordt. Oudere onderzoeken zoals [17] maken gebruik van een robot met een RGB camera die zonder kaart informatie navigeert. De enige informatie die gegeven wordt is een eenvoudige object beschrijving van de gang en een beschrijving van een deur met een deurnummer ernaast. Met enkel een deurnummer als doel vertrekt de robot door de gang, en houdt zichzelf parallel met de muren door gebruik te maken van andere sensoren. Eens er een deur in beeld komt, worden er een aantal features(randen) herkent in het beeld. Nadat hun algoritme de deuren herkend kan er via Optical Character Recognition (OCR) op het deurnummer worden nagegaan of het doel bereikt is. Dit is uiteraard een zeer eenvoudige techniek omdat de robot geen begrip heeft van de omgeving, en moeilijk plaatsen t.o.v elkaar kan onderscheiden.

Nieuwere technieken zoals [6] maken gebruik van RGB-D camera's zoals bijvoorbeeld een kinect waardoor ze ook over diepte-informatie beschikken. Die diepte info kan dan gebruikt worden om heel de omgeving in 3d te mappen en op basis van de effectief gemeten positie te navigeren. Een andere manier om een 3d representatie van de omgeving te verkrijgen zoals [13] is gebruik te maken van stereovisie. Hierbij wordt de informatie van 2 RGB camera's die op een vaste afstand van elkaar staan gecombineerd om diepte informatie te verzamelen.

In dit onderzoek gaan we ons echter beperken tot één enkele RGB camera.

2.2 Object detectie

Een belangrijk aspect van dit onderzoek is het detecteren van individuele objecten in het beeld van één enkele RGB camera. De te detecteren objecten zijn op voorhand vastgelegd, en zijn afhankelijk van de ruimte waarin de robot zich bevindt.

In de logistieke gangen van een ziekenhuis zijn er heel wat objecten die we kunnen detecteren, een kleine selectie van deze objecten zijn:

- Pictogrammen;
- Brandblussers;
- Deurklinken.

Voor deze objecten gaan we kijken naar detectietechnieken uit de traditionele beeldverwerking, en naar meer *state-of-the-art* technieken.

2.2.1 Traditionele beeldverwerking

In openbare gebouwen zijn er heel wat pictogrammen te vinden zoals bijvoorbeeld nooduitgang, hoogspanning en brandblusser. Deze pictogrammen hebben steeds een specifieke vorm, kleur en symbool. De literatuur over pictogramdetectie is schaars. De techniek die voorgesteld wordt door [15] gebruikt een zeer eenvoudige edge detectie met OCR om eventuele letters op pictogrammen te lezen, dit is echter minder relevant omdat niet op elk pictogram tekst aanwezig is. Pictogrammen kunnen echter wel vergeleken worden met verkeersborden die bijna dezelfde kenmerken hebben. De aanpak van [5] is om 2 soorten features in een beeld te onderscheiden. Enerzijds detecteren ze vormen op basis van kleurranden en anderzijds wordt de afbeelding omgezet naar Hue Saturation Intensity (HSI) waaruit enkel de hue gebruikt wordt. De hue is de belangrijkste component voor het onderscheiden van kleuren omdat er zo geen rekening wordt gehouden met de hoeveelheid licht en schaduwen. Een recenter onderzoek [18] bouwt voort op deze technieken, maar hier berekenen ze de Histogram of Oriented Gradients (HOG) features van het beeld. Vervolgens wordt er gebruik gemaakt van een Support Vector Machine (SVM) om te bepalen waar er zich een match bevindt.

!!! TODO support vector machine

De vorm en kleur features kunnen dan gecombineerd worden om de positie van een mogelijke match te vinden. Eens er een mogelijke boundig box rond de mogelijke match gevonden is, kan er geprobeerd worden een template te matchen om het effectieve pictogram te achterhalen. Het grootste probleem bij de techniek van [5] is dat hun gebruikte template matching techniek niet robuust is tegen schaalinvariantie. Bij [18] maken ze voor de herkenningsfase gebruik van Scale-Invariant Feature Transform (SIFT)[9] features en kleur informatie. Hierdoor is het probleem van schaal invariantie grotendeels opgelost. Hierbij worden de SIFT features van de kandidaat matches en de templates vergeleken, en wordt er een gemiddelde genomen van de verschillen tussen hue, saturation en value. Door middel van Random Sample Consensus (RANSAC) en een treshold wordt er bepaald welke matches gebruikt worden. Deze techniek zou gebruikt kunnen worden voor het detecteren van pictogrammen.

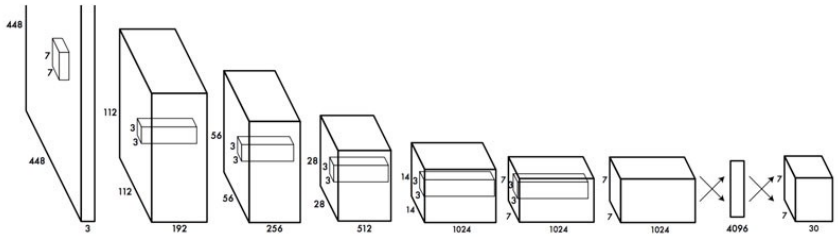
2.2.2 Convolutional neural nework

De laatste jaren in het domein van beeldverwerking wordt er steeds meer gegrepen naar deep learning technieken. Dit komt omdat de rekenkracht van computers steeds beter en beter wordt,

en de resultaten die bekomen worden met neurale netwerken de traditionele manieren overtreffen op verschillende vlakken. Een deep learning techniek die veel gebruikt wordt in de beeldverwerking is een Convolutional Neural Network (CNN).

Een CNN is een supervised deep learning techniek die gebruikt kan worden om verschillende beeldverwerkende taken uit te voeren. Als een traditioneel neuraal netwerk gebruikt zou worden voor een afbeelding van 448 bij 448 pixels, zou er bij elke inwendige laag een paar miljoen variabele gewichten aanwezig moeten zijn, dit zou leiden tot een veel te complex netwerk. Daarom wordt er gebruikgemaakt van een CNN, dit is eigenlijk een neuraal netwerk waarbij niet alle neuronen met elkaar verbonden zijn.

Een CNN kan bestaan uit meerdere lagen die meestal een combinatie zijn van 'convolutional-layers' en 'fully connected-layers'. Elk van deze lagen bevat een aantal neuronen met elk een eigen set van gewichten. Het doel van een CNN is om de gewichten zodanig bij te stellen dat data die aan de eerste laag (de input laag) gegeven wordt een verwacht resultaat geeft aan de laatste laag (de output/classificatie laag). Deze laatste laag kan men de classificatielaag noemen, en geeft een representatie van wat het netwerk denkt dat er aan de input staat. In figuur 2.1 is een voorbeeld zichtbaar van een CNN met de verschillende soorten lagen.



Figuur 2.1: De lagen van een CNN volgens het YOLO [11] detection system.

Een 'convolutional-layer' of convolutie-laag is een laag die een convolutie operatie uitvoert op zijn input. Dit wil zeggen dat de input verdeeld word in regio's van bijvoorbeeld 7x7 pixels, deze 49 pixels zijn verbonden met 1 neuron in de volgende inwendige laag. Als dit 7x7 masker wordt opgeschoven met 1 pixel, verkrijgen we de input voor aan andere neuron in de volgende laag. Dit proces kan gebeuren in meerdere dimensies tegelijkertijd, op dat moment spreekt men van een tensormasker. Een voorbeeld van een convolutie in meerdere dimensies tegelijkertijd is een RGB afbeelding die in een keer door het netwerk gaat, waarbij de dimensies de rode, groene en blauwe pixels van het beeld zijn.

De verschillende inwendige lagen van een CNN zullen na training op zoek gaan naar features. Deze features kunnen eenvoudig zijn zoals randen en lijnen, maar kunnen ook complexer zijn specifiek voor de getrainde data. Elke laag genereert dus een soort featuremap, die gebruikt wordt als input voor de volgende laag van het netwerk.

Om uiteindelijk een classificatie te verkrijgen moet er een dimensievermindering doorgevoerd worden, dit wordt gedaan door 'pooling layers' aan het netwerk toe te voegen na elke convolutie laag. Dit heeft ook als effect dat de featuremaps vereenvoudigd worden, en het aantal gewichten beperkt blijft.

Een CNN kan pas gebruikt worden nadat het getraind is. Voor de training van een netwerk zijn er 2 dingen noodzakelijk: veel voorbeeld data en per voorbeeld de verwachte output (label). Bij het trainingsproces wordt alle inputdata aangelegd, en wordt er gekeken wat het netwerk aan zijn output heeft. De loss functie is een maat van hoe goed een netwerk een voorspelling kan doen van de input data, met andere woorden een vergelijking tussen de input en de output. Trainen van een netwerk is het optimaliseren van de gewichten bij de neuronen in elke laag van het netwerk, bij een optimaal resultaat is de loss functie minimaal. Dit is een complex probleem dat enkel kan lukken indien er genoeg trainingsdata ter beschikking is. Trainen kan gedaan worden d.m.v 'backprogagation'. Backpropagation is het steeds een klein beetje aanpassen van de gewichten bij de neuronen in het netwerk om de classificatie ten gevolge van de input, en het trainingslabel beter op elkaar af te stemmen. Dit proces werkt door de kettingregel toe te passen, en voor elke laag een voor een de gewichten te zoeken.

Een belangrijke laag in een CNN is de Rectified Linear Unit (ReLU). Deze laag voegt een niet lineaire functie toe tussen de convolutielagen, dit is nodig omdat de convolutie een lineaire operatie is, en een eigenschap van lineaire operators laat toe om deze te combineren tot 1 operator. Dit zou willen zeggen dat alle convoluties in een netwerk samengesteld kunnen worden tot 1 laag, en dus het voordeel van meerdere lagen wegnemen. Het toevoegen van een ReLU laag tussen elke convolutielaag voorkomt dit probleem.

Een voorbeeld van een CNN is het 'You Only Look Once (YOLO) detection system' [11]. Het YOLO netwerk is opgebouwd uit 24 convolutielagen en 2 'fully connected' lagen. Dit netwerk heeft een uitgebreide training gehad op de ImageNet dataset[1] en kan gebruikt worden om objectdetectie en -classificatie te doen door één keer de input afbeelding door het netwerk te laten gaan. Door middel van een hertraining kan deze detector leren om alle objecten te detecteren en te classificeren en dus een mogelijke detector zijn voor onze toepassing.

Zoals [8] voorstelt is het niet moeilijk om het 'YOLO detection system' een hertraining te geven om deuren te herkennen. Zo kan dit ook toegevoegd worden aan de lijst met te detecteren kenmerken.

2.3 Object tracking

Object tracking of het volgen van objecten heeft als doel het bepalen van de positie van hetzelfde object over meerdere frames heen. In het geval van dit onderzoek kan het een indicatie geven van relatieve posities t.o.v. objecten die zich in de gangen bevinden. Een grote moeilijkheid bij het volgen van objecten die stilstaan t.o.v. de camera is dat ze veranderen in grootte, oriëntatie en perspectief. [20] stelt voor om gebruik te maken van SIFT voor het volgen van objecten. Ze zoeken een Region Of Interest (ROI) op het eerste frame waarop ze een kleurhistogram en SIFT features berekenen. Op het volgende frame worden dezelfde bewerkingen uitgevoerd in een regio die net iets groter is dan de originele ROI. Een overeenkomstregio wordt dan berekend door middel van een Expectation-Maximization (EM) algoritme. Volgens [3] is het beter om gebruik te maken van het KLT feature algoritme [16]. Hiermee wordt er een transformatie berekent waardoor de ROI

[1] http://www.image-net.org

tussen de 2 frames gelijkaardig wordt. De initiële transformatieparameters worden berekend via RANSAC.

[10] heeft een nieuwe techniek ontwikkeld om object tracking te combineren met object detectie CNN. Ze hebben een uitbreiding gemaakt op het YOLO detection system genaamd Recurrent YOLO (ROLO). De uitbreiding bevat een extra Long Short Term Memory Recurrent Neural Network (LSTM RNN) geplaatst achter de detectie fase van het originele netwerk. Een LSTM RNN is een neuraal netwerk geoptimaliseerd voor het maken van beslissingen op tijdsgebaseerde data. De data die aan het extra netwerk gegeven wordt is een van de tussenresultaten van het YOLO netwerk. Het systeem blijkt zeer goed te werken voor het volgen van objecten zelfs bij occlusies in één van de beelden.

2.4 Image segmentation

Het correct segmenteren van de beelden zal een belangrijke rol spelen. Niet in elk beeld zal er een distinctief object aanwezig zijn om te detecteren. Daarom is het belangrijk om de vloer van de muren te kunnen onderscheiden. Een eenvoudige aanpak zou kunnen zijn om via K-means een verdeling van een beeld te doen en met een soort regressie de regio's te labelen. Volgens [19] werkt de K-means aanpak met een op textuur en kleur gebaseerde aanpak redelijk goed, maar wordt steeds de muur verbonden met het plafond omwille van kleur en textuurgelijkenissen. Hun regressie gebaseerde labeling techniek blijkt dus een slechte oplossing. Verder zoals [7] aangeeft zijn reflecties en overbelichting eigenschappen van indoor omgevingen die het moeilijk kunnen maken om een correcte segmentatie te doen. [7] stelt een techniek voor die begint met het detecteren van verticale en horizontale lijn segmenten. Dit doen ze door eerst een Canny edge detector[4] toe te passen en vervolgens een line fitting. Een zelf geleerde SVM classifier verdeeld alle lijnsegmenten in twee categorieën: horizontaal en verticaal. De vluchtlijnen van de gang worden hierbij onderverdeeld in de horizontale categorie. Alle lijnstukken krijgen een score via een reeks van operaties waarna enkel de beste lijnen bijgehouden worden. Op basis van de kleur van de vlakken tussen de lijnstukken kan een segmentatie gemaakt worden. Dit geeft een resultaat waarbij de vloer meestal een mooi homogeen geheel is, maar de muren in meerdere vlakken gesegmenteerd worden door eventuele kleurverschillen en objecten aan de muur.

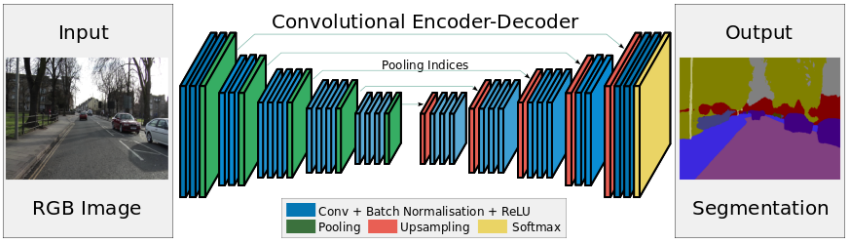
Een andere manier om de vloer te segmenteren is voorgesteld in [12]. Zij doen een superpixel segmentatie volgens het SLIC algoritme [1], vervolgens bekijken ze de randen van de superpixels. Na observaties blijkt dat de randen van superpixels onregelmatig worden bij objectovergangen. Door het aanduiden van een paar vloerpixels kan hun algoritme superpixels aanduiden die tot de vloer behoren. Deze aanpak geeft een goede schatting van vrije ruimte op de vloer, maar is minder bruikbaar voor segmentatie van muren.

Een meer recente technologie om afbeeldingen te segmenteren is gebruik te maken van een CNN. Het netwerk voor segmentatie is verschillend van een traditioneel CNN voor bijvoorbeeld object detectie. Een voorbeeld van een segmentatienetwerk is te zien in figuur 2.2.

Het segmentatienetwerk SegNet [2] is een combinatie van convolutielagen en pooling layers. Er zijn

geen fully connected layers aanwezig zoals het geval is bij een classificatie netwerk. De bedoeling van het SegNet netwerk is om als output opnieuw een afbeelding te genereren. Hiervoor zijn de lagen opgebouwd als een zandloper. Op deze manier is de output even groot als de oorspronkelijke afbeelding. Een segmentatienetwerk wordt getraind op gelijkaardige manier aan een traditioneel CNN met als verschil dat de labeling gebeurd op pixelbasis aangezien de output even groot is als de input van het systeem. Het systeem geeft als output een label voor elke pixel uit de afbeelding onderverdeeld in de classen waarmee het systeem getraind is.

Het SegNet netwerk is getraind op de SUN RGB-D [14] dataset. Deze dataset bevat een groot aantal indoor scenes, waarbij er onder andere segmentatie klassen zijn voor muren, vloeren en plafonds. De training is gebeurd met enkel de RGB gegevens van de dataset. Deze segmentatie-techniek zou nuttig kunnen zijn voor dit onderzoek om in beelden bijvoorbeeld de vloer te kunnen onderscheiden.



Figuur 2.2: Het SegNet [2] segmentatie netwerk.

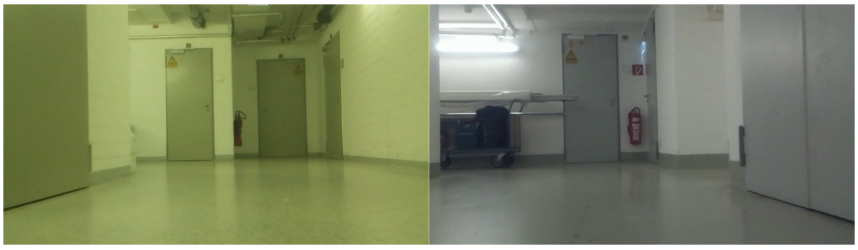
Hoofdstuk 3

Reeds gerealiseerd

Er zijn reeds een aantal experimenten uitgevoerd met de technieken die beschreven worden in de literatuur om voeling te krijgen met wat er wel en niet kan werken in situatie van dit onderzoek en het beschikbare beeldmateriaal. De experimenten zijn onder te verdelen in 2 categorieën die verder beschreven worden.

3.1 Object detectie

In hoofdstuk 2.2.1 is er beschreven hoe er op een traditionele manier pictogrammen herkent kunnen worden. Bij het zoeken naar de specifieke hue van de pictogrammen is gebleken dat het wel mogelijk is om dit als feature te gebruiken op het beschikbare beeldmateriaal, maar er is een te groot verschil in de belichting tussen de verschillende gangen waardoor de kleuren niet overeen komen en er moeilijk een juiste hue bepaald kan worden. Dit is geïllustreerd in figuur 3.1. Dit toont aan dat de hue van de pictogrammen geen absoluut uitsluitsel kan geven over de segmentatie, maar uitraard is het verschil in kleur in 1 beeld/omgeving nog steeds een feature die het pictogram onderscheid van de achtergrond.



Figuur 3.1: Kleurverschil tussen verschillende gangen

Voor het identificeren van de pictogrammen is er geprobeerd om gebruik te maken van 'local feature matching' door middel van SIFT. Hierbij is gebleken dat door de lage resolutie van het beeldmateriaal er zo goed als geen features gedetecteerd kunnen worden zoals te zien in figuur 3.2.

In hoofdstuk 2.2.2 is reeds beschreven hoe een CNN gebruikt kan worden om objecten te detec-

Figuur 3.2: SIFT features op gedetecteerde contour.

teren. Hiervoor zijn we begonnen met annoteren van het beeldmateriaal. Voor het tekenen van bounding boxes is er gebruik gemaakt van de opencv tool CVAT[1]. Als proof of concept zijn we begonnen met 4 object klassen om te detecteren:

- Brandblusser;
- Deurklink;
- Pictogram;
- Bordje.

In elk frame van het beeldmateriaal zijn er voor deze klassen bounding boxes getekend. De output van de CVAT tool is een XML-bestand met voor elke afbeelding de coördinaten van de objecten. Een voorbeeld output voor 1 enkele afbeelding is te zien in listing A.

Listing 3.1: Voorbeeld CVAT output

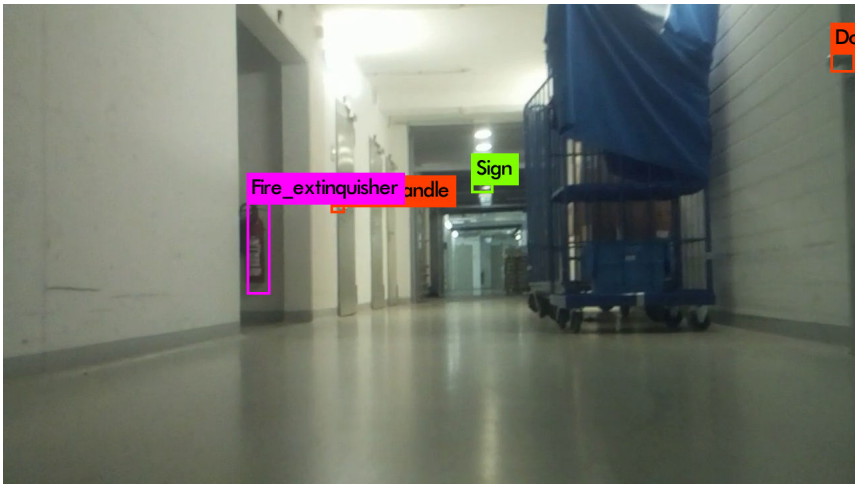
```
<image id="24" name="hospital_corridors_4Hz-025.png" width="1280" height="720">
    <box label="door_handle" xtl="863.66" ytl="237.20" xbr="881.28" ybr="248.22"/>
    <box label="sign" xtl="584.02" ytl="254.12" xbr="608.24" ybr="265.28"/>
    <box label="door_handle" xtl="435.83" ytl="321.53" xbr="448.33" ybr="329.06"/>
    <box label="sign" xtl="590.76" ytl="298.63" xbr="604.02" ybr="307.70"/>
</image>
```

De trainingsdata die aan het YOLO netwerk geleverd moet worden is in een volledig ander formaat dan het verkregen XML bestand, daarom is er een conversiescript geschreven om dit om te zetten naar het YOLO formaat. Het conversiescript bevind zich in bijlage A. Het formaat dat YOLO verwacht is per input afbeelding een .txt bestand met daarin per lijn een bounding box. Het YOLO formaat is beschreven in 3.1 waarbij $< x >$ en $< y >$ het centerpunt van een bounding box zijn.

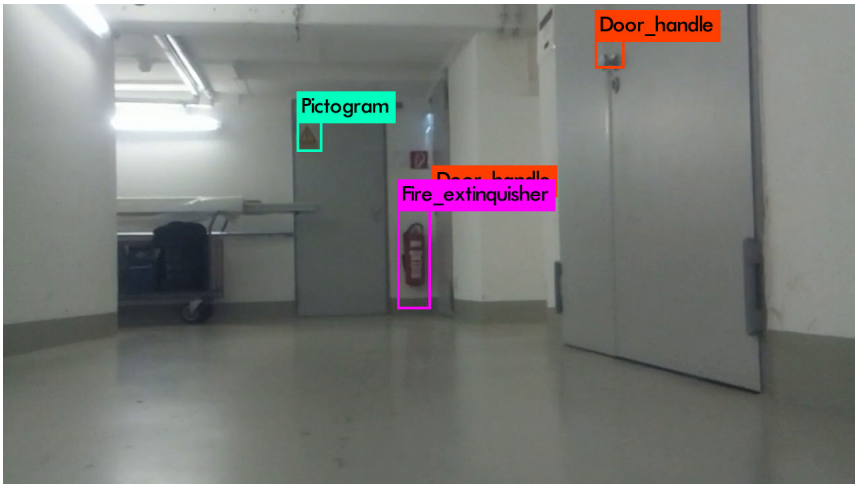
$$< x >< y >< width >< height > \tag{3.1}$$

In figuur 3.3 en 3.4 zijn 2 resultaten te zien van de YOLO detector op de dataset.

[1] https://github.com/opencv/cvat



Figuur 3.3: Resultaat 1 YOLO detector.

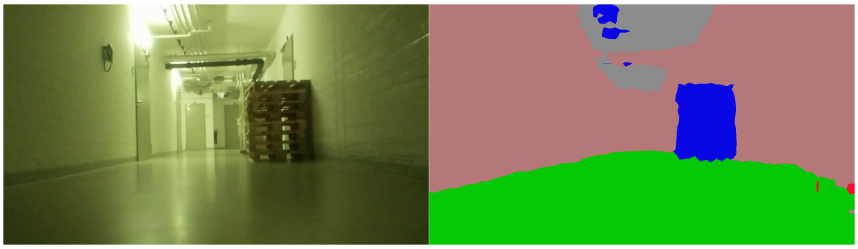


Figuur 3.4: Resultaat 2 YOLO detector.

3.2 Image segmentation

Zoals reeds voorgesteld in hoofdstuk 2.4 kan voor het segmenteren van de gangen gebruik gemaakt worden van het SegNet [2] segmentatie netwerk. Na enkele uren proberen is het niet gelukt om SegNet werkend te krijgen. Dit zal in de toekomst hernomen worden.

In plaats daarvan is er een ander segmentatie netwerk gevonden gebaseerd op ResNet dat reeds getraind werd op een dataset met indoor scenes en gangen. In figuur 3.5 is het resultaat te zien van het netwerk zonder een hertraining.



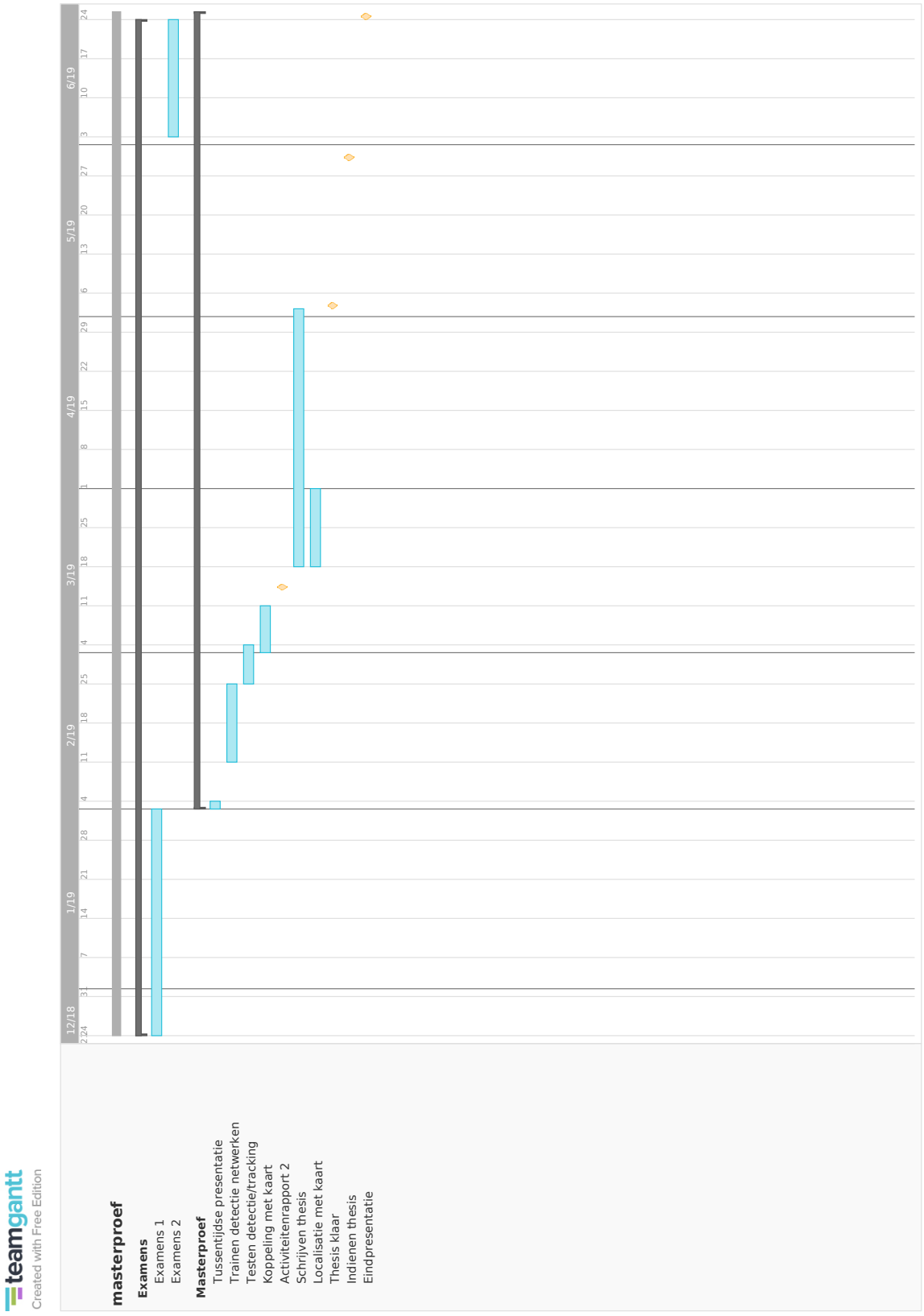
Figuur 3.5: Resultaat van ResNet segmentatie netwerk.

In de toekomst zal er vergeleken worden welke van de 2 netwerken het beste resultaat geeft, en zal er een hertraining gebeuren op basis van het beeldmateriaal om te proberen om ook deuren te segmenteren.

Hoofdstuk 4

Planning

De bijgewerkte planning is te zien in figuur 4.1.



Figuur 4.1: Herwerkte planning

Bibliografie

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. SÃ$\frac{1}{4}$sstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, Nov 2012.

[2] Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla, and Senior Member. SegNet : A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. pages 1–14.

[3] Bhakti Baheti, Ujjwal Baid, and Sanjay Talbar. An approach to automatic object tracking system by combination of SIFT and RANSAC with mean shift and KLT. *Conference on Advances in Signal Processing, CASP 2016*, pages 254–259, 2016.

[4] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986.

[5] Chiung Yao Fang, Sei Wang Chen, and Chiou Shann Fuh. Road-sign detection and tracking. *IEEE Transactions on Vehicular Technology*, 52(5):1329–1341, 2003.

[6] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *In the 12th International Symposium on Experimental Robotics (ISER*, 2010.

[7] Yinxiao Li and Stanley T. Birchfield. Image-based segmentation of indoor corridor floors for a mobile robot. *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pages 837–843, 2010.

[8] Adrian Llopart, Ole Ravn, and Nils A. Andersen. Door and cabinet recognition using Convolutional Neural Nets and real-time method fusion for handle detection and grasping. *2017 3rd International Conference on Control, Automation and Robotics, ICCAR 2017*, pages 144–149, 2017.

[9] D G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, sep 1999.

[10] Guanghan Ning. Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking. (1):1–4, 2017.

[11] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[12] F. Geovani Rodríguez-Telles, L. Abril Torres-Méndez, and Edgar A. Martínez-García. A fast floor segmentation algorithm for visual-based robot navigation. *Proceedings - 2013 International Conference on Computer and Robot Vision, CRV 2013*, pages 167–173, 2013.

[13] K. Schmid, T. Tomic, F. Ruess, H. HirschmÃ$\frac{1}{4}$ller, and M. Suppa. Stereo vision based indoor/outdoor navigation for flying robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3955–3962, Nov 2013.

[14] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[15] R. Swathika and T. S. Sharmila. Emergency exit sign detection system for visually impaired people. In *2016 International Conference on Inventive Computation Technologies (ICICT)*, volume 1, pages 1–7, Aug 2016.

[16] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. 1991.

[17] M Tomono and S Yuta. Mobile robot navigation in indoor environments using object and character recognition. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 313–320 vol.1, apr 2000.

[18] S. J. Zabihi, S. M. Zabihi, S. S. Beauchemin, and M. A. Bauer. Detection and recognition of traffic signs inside the attentional visual field of drivers. *IEEE Intelligent Vehicles Symposium, Proceedings*, (Iv):583–588, 2017.

[19] Zhong-Ju Zhang. Wall, floor, ceiling, object region identification from single image.

[20] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. Object tracking using SIFT features and mean shift. *Computer Vision and Image Understanding*, 113(3):345–352, 2009.

Bijlage A

CVAT naar YOLO conversie

```
import xml.etree.ElementTree as ET
import argparse
import os

parser = argparse.ArgumentParser()
parser.add_argument("file")
args = parser.parse_args()

classes = []


def getLabels(root):
    labels = root.find('task').find('labels')

    for label in labels:
        classes.append(label.find('name').text)

    with open('cust.names', 'w') as names:
        names.write("\n".join(classes))


if __name__ == '__main__':
    tree = ET.parse(args.file)
    root = tree.getroot()

    print(root.tag)

    if not os.path.isdir('Anotations_yolo'):
        os.mkdir('Anotations_yolo')

    for child in root:
        if child.tag == 'meta':
```

```
        getLabels(child)

    if child.tag == 'image':
        image = child.get('name')
        w = float(child.get('width'))
        h = float(child.get('height'))

        boxes = ''
        for box in child.findall('box'):
            xtl = float(box.get('xtl'))
            ytl = float(box.get('ytl'))
            xbr = float(box.get('xbr'))
            ybr = float(box.get('ybr'))

            width = xbr - xtl
            height = ybr - ytl
            x = xtl + (width/2)      #center
            y = ytl + (height/2)

            boxes += '{} {} {} {} {}\n'.format(classes.index(box.get('label')),
                                               x/w, y/h, width/w, height/h)

        with open('Anotations_yolo/{}.txt'
                  .format(os.path.splitext(image)[0]), 'w') as file:
            file.write(boxes)
```

FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
CAMPUS DE NAYER SINT-KATELIJNE-WAVER
J. De Nayerlaan 5
2860 SINT-KATELIJNE-WAVER, België
tel. + 32 15 31 69 44
iiw.denayer@kuleuven.be
www.iiw.kuleuven.be