T.C.
HASAN KALYONCU UNIVERSITY
GRADUATE EDUCATION INSTITUTE
DEPARTMENT of ELECTRONICS AND COMPUTER ENGINEERING

TOWARDS ONLY-VISION AUTONOMOUS WHEELCHAIR: A DEEP LEARNING OBSTACLE DETECTION AND IMAGE-BASED AVOIDANCE

Yahya TAWIL

MASTER THESIS

GAZIANTEP - 2023

TOWARDS ONLY-VISION AUTONOMOUS WHEELCHAIR: A DEEP LEARNING OBSTACLE DETECTION AND IMAGE-BASED AVOIDANCE

Electronics and Computer Engineering

Hasan Kalyoncu University

M.Sc. Thesis

Supervisor
Assoc. Prof. Dr. Abdul Hafiz ABDULHAFIZ

Yahya TAWIL
Jan 2023

© 2023 [Yahya TAWIL]

GRADUATE EDUCATION INSTITUTE
MSc ACCEPTANCE AND APPROVAL FORM

M.Sc. (Master Of Science) graduate program student **Yahya TAWIL** of the Electronics and Computer Engineering department prepared a thesis titled "**Towards Only-vision Autonomous Wheelchair: a Deep Learning Obstacle Detection and Image-based Avoidance**". It was **accepted** as a **Master's Thesis** by our jury as a result of the defense exam held on 12/1/2023.

Position	Title, Name and Surname	Department/University	Signature:
Thesis Supervisor	Assoc. Prof. Dr. ABDUL HAFIZ ABDULHAFIZ	Computer Engineering Department Hasan Kalyoncu University	
Jury Head	Assoc. Prof. Dr. Tolgay Kara	Electrical and Electronics Engineering Department Gaziantep University	
Jury Member	Asst. Prof. Dr. Saed ALQARALEH	Computer Engineering Department Hasan Kalyoncu University	

This thesis is accepted by the jury members selected by the institute management board and approved by the institute management board.

Prof. Dr. M.Serhat YENİCE
Institute Director

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Yahya TAWIL

19/1/2023

HASAN KALYONCU UNIVERSITY
GRADUATE EDUCATION INSTITUTE
DEPARTMENT of ELECTRONICS AND COMPUTER ENGINEERING

TOWARDS ONLY-VISION AUTONOMOUS WHEELCHAIR: A DEEP LEARNING OBSTACLE DETECTION AND IMAGE-BASED AVOIDANCE

Yahya TAWIL

MASTER THESIS

Advisor
Assoc. Prof. Dr. Abdul Hafiz ABDULHAFIZ

ABSTRACT

A rapidly increasing number of people need to use a wheelchair (WC). WC users face several challenges while using the WC. Obstacle avoidance is one of these challenges. Avoidance for some users cannot be done simply using manual control. We propose that the WC should be able to achieve the avoidance automatically. Previous systems offered solutions to similar problems using a fusion of expensive depth sensors. This system uses vision-only technology, via a single camera, to achieve detection and avoidance at a cost that makes it accessible to a large number of disabled users. Our approach integrates functionalities from deep learning, computer vision and mobile robotics fields into the standard powered wheelchair (PWC). A deep-learning model is adapted using learning-transfer techniques to detect obstacles. A dataset of sidewalks has been developed to be used in the learning-transfer process. Any obstacle detected in front of the WC is avoided using a developed image- space avoidance method. The system was deployed during experiments on a Hardware setup using a real PWC. Object detection accuracy is reported as 61% mAP, which is comparable to methods implemented using standard computers. The control module generated the required motor speeds to avoid the obstacle successfully. The overall system achieves a speed of 5 FPS. We conclude that our cost-effective system can work effectively without the need for redundant and costly depth sensors. Adopting our system will increase the mobility of people with disabilities both indoors and outdoors. This opens the way for a vision-only fully autonomous WC.

Key Words: Autonomous wheelchair, obstacle-detection and -avoidance, assistive technology, deep-learning, computer vision

HASAN KALYONCU ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
ELEKTRONIK VE BILGISAYAR MÜHENDISLIĞI ANABİLİM DALI

SADECE GÖRÜŞLÜ OTONOM TEKERLEKLİ SANDALYEYE DOĞRU: DERİN ÖĞRENMELİ BİR ENGEL ALGILAMA VE GÖRÜNTÜ TABANLI KAÇINMA

Yahya TAWIL

YÜKSEK LİSANS TEZİ

Danışman
Doç. Prof. Abdul Hafiz ABDULHAFIZ

ÖZET

Hızla artan sayıda insan tekerlekli sandalye (WC) kullanmaya ihtiyaç duymaktadır. WC kullanıcıları, WC'yi kullanırken çeşitli zorluklarla karşılaşırlar. Engellerden kaçınma bu zorluklardan biridir. Bazı kullanıcılar için kaçınma, basitçe manuel kontrol kullanılarak yapılamaz. WC'nin kaçınmayı otomatik olarak başarabilmesi gerektiğini öneriyoruz. Önceki sistemler, pahalı derinlik sensörlerinin bir füzyonunu kullanarak benzer sorunlara çözümler sunuyordu. Bu sistem, çok sayıda engelli kullanıcının erişebileceği bir maliyetle algılama ve kaçınma sağlamak için tek bir kamera aracılığıyla yalnızca görüş teknolojisini kullanır. Yaklaşımımız, derin öğrenme, bilgisayar görüşü ve mobil robotik alanlarındaki işlevleri standart elektrikli tekerlekli sandalyeye (PWC) entegre eder. Engelleri tespit etmek için öğrenme-aktarma teknikleri kullanılarak bir derin öğrenme modeli (deep-learning model) uyarlanmıştır. Öğrenme-aktarma sürecinde kullanılmak üzere bir kaldırım veri seti geliştirilmiştir. WC'nin önünde tespit edilen herhangi bir engel, gelişmiş bir görüntü alanından kaçınma yöntemi kullanılarak engellenir. Sistem, gerçek bir PWC kullanan bir Donanım kurulumundaki deneyler sırasında dağıtıldı. Nesne algılama doğruluğu, standart bilgisayarlar kullanılarak uygulanan yöntemlerle karşılaştırılabilir olan %61 mAP olarak rapor edilmiştir. Kontrol modülü, engelden başarıyla kaçınmak için gerekli motor hızlarını üretti. Genel sistem 5 FPS hıza ulaşır. Uygun maliyetli sistemimizin gereksiz ve maliyetli derinlik sensörlerine ihtiyaç duymadan verimli bir şekilde çalışabileceği sonucuna vardık. Sistemimizi benimsemek, engelli kişilerin hem içeride hem de dışarıda hareketliliğini artıracaktır. Bu, yalnızca vizyona dayalı tamamen otonom bir WC'nin yolunu açar.

Anahtar Kelimeler: Otonom tekerlekli sandalye, engel algılama ve kaçınma, yardımcı teknoloji, derin öğrenme, bilgisayar görüşü

DEDICATION

To my belief and faith which drive me always to accomplish what may make me a better person and the world a better place... To Islam.

To my parents who did their best to raise me and never saved any effort... To my father Dr. Walid Tawil and my mother Dr. Yomn Alghadban.

To my wife, who was patient with my busyness and extraordinary life conditions, yet she bore many burdens and provided love and support to me and our lovely kids all the time... To beloved Alaa and my lovely little daughters Maria and Sirin.

To my wounded home country which provided excellent education to me to be a successful engineer... To Syria.

ACKNOWLEDGEMENTS

I would like to express my acknowledgment to my supervisor Dr. Abdul Hafiz for his guidance and support during my master's degree and my work for TÜBİTAK 117E173 project. Dr. Abdul Hafiz was also the PI for TÜBİTAK 117E173 project. He was always motivated to make the project successful and accomplished. His inputs and suggestions are reflected clearly in this work's problem formulation and control law design.

I would like to thank TÜBİTAK for providing the scholarship during my work for 117E173 project. I would also like to thank Hasan Kalyoncu University represented by the rector Prof. Turkey Dereli and the head of electroincs and computer engineering department Prof. Dr. M. Fatih HASOĞLU.

I would like to thank the thesis committee for the brilliant comments and suggestions.

Finally, I would like to acknowledge the support provided by 16Lab Inc. company represented by Mr. Ko Kijima who was a very understanding and helpful manager while doing my part-time job on weekends during the M.Sc. program, and to MIT Technologies Sdn. Bhd. company represented by Mr. Ahmad Tahoun who made my leave from Malaysia to Türkiye to join the Master's degree program seamlessly. Also, to my friend Eng. Ammar Tello who provided a lot of support in different approaches during the past 2 years.

TABLE OF CONTENTS

ABSTRACT i
ÖZET ii
DEDICATION iii
ACKNOWLEDGEMENTS iv
TABLE OF CONTENTS v
LIST OF TABLES vii
LIST OF FIGURES viii
LIST OF SYMBOLS/ABBREVIATIONS x
CHAPTER 1 1
INTRODUCTION 1
1.1 Problem Statement 1
1.2 Motivation and objectives 2
1.3 Contributions 3
1.4 Organization of the thesis and publication 4
CHAPTER 2 5
BACKGROUND 5
2.1 Tradition Computer Vision vs Deep Learning Computer Vision 5
2.2 Evolution of object detection models 6
2.3 Visual Serviong 7
CHAPTER 3 12
LITERATURE REVIEW 12
3.1 Reviewing the powered wheelchairs with obstacle avoidance 12
3.2 Reviewing the existing datasets 17
CHAPTER 4 20
DEEP-LEARNING OBSTACLE DETECTION AND VISION-ONLY AVOIDANCE 20
4.1. Our system overview 20
4.2 Deep-learning model selection for obstacle detection 22
4.3 The new WODD dataset 24
4.3.1 Data collection 24
4.4 Dataset images annotation and learning transfer 25
4.5 The usage of EdgeImpulse platform 28
4.6 Novel image space wheelchair obstacle avoidance 30
4.6.1 Control law design 31

CHAPTER 5 34
EXPERIMENTAL EVALUATION AND ANALYSIS 34
5.1 The experimental setup and wheelchair model 34
5.2 Evaluating the obstacle-detection model 37
5.3 Experiments on deep-learning model inference speed 40
5.4 Experiments on detection and avoidance using the real wheelchair system 43
CHAPTER 6 47
CONCLUSION 47
REFERENCES 49

LIST OF TABLES

Table 3.1 A Review of research and commercial PWCs with obstacle detection functionality. 15
Table 3.2 The different datasets built for or used in object detection for robots that use side-walk environments 19
Table 5.1 A compression between Raspberry Pi 4 and Nvidia Jetson Nano 36
Table 5.2 Evolution of our dataset (WODD) during fine-tuning experiments. 40
Table 5.3 MobileNet V2 SSD FPS results on Raspberry Pi 4 & Jetson Nano (CPU) under different setups..... 42

LIST OF FIGURES

Figure 1.1 A bounding box is used as a feature by the control law to work on moving the obstacle to the safety margin of the image after it is detected in the Area of Interest (AOI) in front of the WC. The WC then returns to its primary task. 1
Figure 2.1 An illustration, from (O'Mahony et al., 2019), showing the difference between traditional computer vision (a) and deep learning workflow (b). 5
Figure 2.2 Object detection steps in R-CNN. 6
Figure 2.3 Object detection steps in SSD 7
Figure 2.4 MobileNet V2 Architecture. 7
Figure 2.5 The basic steps in IBVS. Figure is adapted from (Corke and Khatib, 2011) . 8
Figure 2.6 An example of a target square shape (blue) and the initial shape (red). The task is to move the feature points indicated by o-marks to the points indicated by *-marks. The figure is adapted from (Corke and Khatib, 2011) 8
Figure 2.7 Shows the central projection model which consists of Camera coordinate C with image plane (in yellow) with distance f, a point P in homogeneous coordinate and a projected point on the image plane p. Figure is adapted from robotacademy.net. 9
Figure 2.8 Another illustration showing the principle of point projection between 3D world and image plane with respect to the camera center. This figure is adapted from Notes on Motion Estimation. 10
Figure 3.1 Primitive way of finding of object detection using a sliding window with fixed stride and size. 13
Figure 3.2 Samples from Sidewalks dataset after drawing the bounding boxes using the provided information in bbox_sample.xml file. Only samples can be downloaded publicly. 17
Figure 3.3 Samples from SOID dataset showing the 96x96 pixles size. 18
Figure 3.4 code snippet used to load samples from SOID dataset pickle file. 18
Figure 3.5 Samples from Mapillary Vistas dataset show some examples of sidewalk views. 19
Figure 4.1 An illustration of our developed system blocks 21
Figure 4.2 A flowchart showing the main tasks in our system. 21
Figure 4.3 Object detection steps in our deep learning model. The first one is an image feature extractor backbone which is selected to be similar to the one adopted in (Sandler et al. 2018). The output of conventional layers at different scales are fed to non-maximum suppression predictor similar to (Lee and Kim, 2020). This architecture does the object detection in one shoot. 22
Figure 4.4 The model size and accuracy among different network architectures trained on ImageNet dataset. Image is adapted from (Choudhary et al., 2020). 23
Figure 4.5 Sample images from WODD showing the four augmentation effects. They are from left to right: Contrast adjustment with fc = 1.5, Gamma effect transformation with $\gamma = 1.5, C = 0.9$, Gaussian noise with $\mu = 0.0, \sigma = 0.05$, and at the most right is Quality adjustment with encoding quality=12. 25
Figure 4.7 A bar chart showing the distribution of the 3976 instances over classes in our WODD dataset. The Traffic Sign and trunks classes are generally common among places. This explains the relatively larger number of instances in these two classes. 26
Figure 4.6 Samples from the annotated images contained in our WODD dataset. 26

Figure 4.8 An illustration for (a) undersampled dataset and (b) oversampled dataset. . 28
Figure 4.9 Our dataset was visualized using feature points after dimensionality reduction to 3D space. This graph is exported from Edge Impulse using the ’feature explorer’ tool. 28
Figure 4.10 A diagram showing the ecosystem that Edgeimpulse platform provides... 29
Figure 4.11 A screenshot showing the usage of Edgeimpulse annotation tool during building our WODD dataset. 30
Figure 4.12 A screenshot showing the capability in Edgeimpulse to formulate a deep learning model using building configurable blocks. 30
Figure 4.13 An illustration of the problem in the image plane. AOI, safety margin, error function e , position of feature point (u,v), and the position of the desired location of feature point (u∗,v∗) are depicted here. 32
Figure 5.1 The Hardware components of our wheelchair setup 34
Figure 5.2 Raspberry Pi Camera Module v2 35
Figure 5.3 Motor driver Sabertooth 2x32 35
Figure 5.4 The PWC used during experiments and development with the installed hardware (battery, Raspberry Pi, motor controller and camera). 36
Figure 5.5 WC modeling with camera, robot and world frames (a) Side view of the WC (b) Top view of the WC 37
Figure 5.6 A prediction example in the image that depicts the Intersection over Union (IoU) calculation. 38
Figure 5.7 Samples of the validation images with bounding boxes as ground truth, and under each one the output from the object-detection 38
Figure 5.8 Precision-recall curve for our fine-tuned model. 39
Figure 5.9 P-R curves during the evolution of our dataset (WODD) during fine-tuning experiments. 39
Figure 5.10 Running MobileNet V2 SSD on Jetson Nano at 2.8 FPS speed. 41
Figure 5.11 CPU load on the 4 cores available in Jetson Nano CPU while running MobileNet V2 SSD. 41
Figure 5.12 Achieving 5.69 FPS while running the Mobilenet V2 SSD using Tensorflow Lite and Raspberry Pi 64-bit OS. 42
Figure 5.13 CPU and memory usage on Rasbperry Pi 4 while both dectection and avoidance sub-modules are operating. 43
Figure 5.14 Different scenes show the detected obstacles (green bounding box) inside AOI (green trapezoid area) and how to control law is bringing the feature point (green dot) to the safety margin based on the distance to it (blue line). Under each image the action is generated using the velocities calculated in the control law. 44
Figure 5.15 (a) The calculated linear and angular speed by the control law changing per frame whilst avoiding the obstacle. (b) Our WC estimated trajectory path while avoiding the obstacle. 46

LIST OF SYMBOLS/ABBREVIATIONS

AOI	Area Of Interest
CV	Computer Vision
CNN	Convolutional Neural Network
FP	False Positive
FPS	Frame Per Second
FN	False Negative
IBVS	Image Based Visual Servoing
IOU	Intersection Over Union
JSON	JavaScript Object Notation
LSTM	Long Short Term Memory
PR	Precision-Recall
MAP	Mean Average Precision
NMS	Non-Maximum Suppression
PBVS	Position Based Visual Servoing
PWC	Powered Wheelchair
RCNN	Recurrent Convolutional Neural Network
RGB-D	RGB Depth
SBC	Single Board Computers
SSD	Single Shot multibox Detector
TP	True Positive
TN	True Negative
UMAP	Uniform Manifold Approximation and Projection
WC	Wheelchair
WODD	Wheelchair Obstacle Detection Dataset
YOLO	You Only Look Once

CHAPTER 1

INTRODUCTION

This work focus on obstacle detection and avoidance on the sidewalk and outdoor environment to facilitate mobility amongst the elderly and disabled using a vision-only system. Figure 1.1 depicts an abstract description of our system. The developed system involves a low-cost Hardware setup.

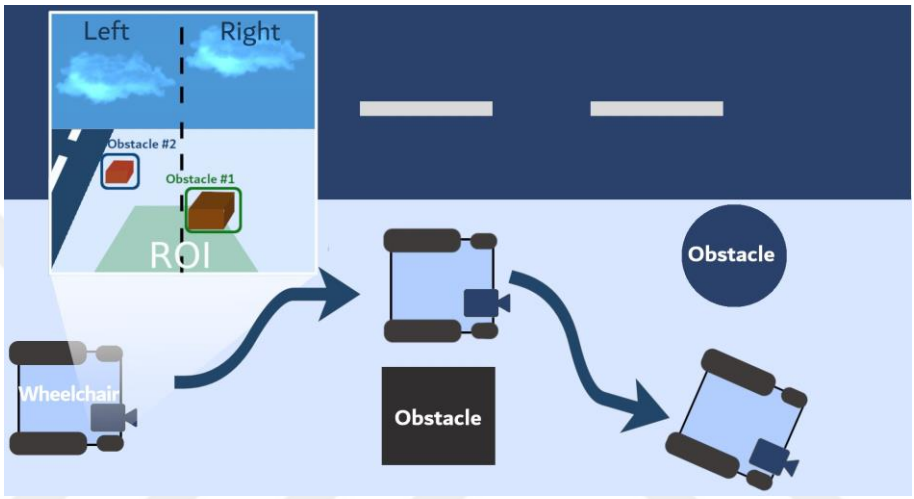


Figure 1.1 A bounding box is used as a feature by the control law to work on moving the obstacle to the safety margin of the image after it is detected in the Area of Interest (AOI) in front of the WC. The WC then returns to its primary task.

1.1 Problem Statement

The world faces an increasing need for mobility support such as WCs (World Health Organization, 2015). Assisting people with limited mobility, like those who need WC support, is essential to keeping all members of the community involved, active, and enjoying a healthy lifestyle. The absence of such support can lead to disorders such as depression. Such support and participation require proper technology and an accessible physical environment (Edwards and McCluskey, 2010; Sidik et al., 2018).

Barriers to power mobility, such as PWCs, were surveyed by Edward, who found that 52% of the participants reported obstacles in front of the WC as a barrier to their mobility experience (Edwards and McCluskey, 2010). Although PWCs are affordable nowadays, using them in constrained environments is challenging or sometimes impossible. According to users surveyed in (Sidik et al., 2018), particularly those with certain types of paralysis or other disabilities, the risk of falling, collision or injuries is high in some environments as such pavements with potholes.

Smart PWCs have been a subject of research since the 1980s (Leaman et al., 2017). The review of Lehman et al. stated that more than a hundred studies had been carried out on smart WCs by 2017. The survey summarizes related works on PWCs between 1983 and 2016. A survey of smart WC prototypes dating from 1992 to 2012 was also reviewed in (Simpson et al., 2005;Desai et al., 2017) with a focus on the sensors used, prototype form, functionality and features. For more about the history of smart WC development.

Torkia et al. assessed user perceptions on PWC driving in the community and identified that obstacle avoidance is a major challenge that users face (Torkia et al., 2015). These findings inspired our work, which aims to increase the mobility of people with disabilities in both indoor and outdoor environments at a cost that will allow access by numerous disabled users with special mobility needs.

A variety of PWC driving strategies are investigated in the literature. These strategies vary between the standard joystick and new assistive driving tools such as gaze-based (eye tracker) control (Maule, 2022;Eid et al., 2016;Elliott et al., 2019), voice and speech control (Priyanayana et al., 2018; Joshi et al., 2019; Kadmin, 2016), leap motion control (Fereidouni et al., 2020), remote control (Soma et al., 2018; Akash et al., 2014), and infrared sensor headset control (Chen et al., 2007).

Despite the overall success of these strategies, it is reported that several challenges need yet to be addressed and solved. For example, the need for a stable internet connection, the eye movement and blinking, focusing on the control and the path at the same time in the gaze-based controller, and the external noise and interventions in general. All the above strategies are user interaction-based and require user attention while driving.

1.2 Motivation and objectives

In this work, we focus on a solution that requires minimal user interaction. This approach helps in several scenarios where users are not able to pay attention to the driving task, or they are looking for a more seamless driving experience. Such scenarios are widely experienced outdoors and in busy indoor environments such as hospitals and airports.

Providing an expensive solution to this problem limits that solution to those WC users with enough money, who are in a minority in our world. The low income that most WC users have is a serious barrier to implementing new technology. Shore and Juillerat studied more than 500 disabled persons from 3 countries and emphasized the importance of providing low-cost WCs to the developing world (Shore and Juillerat, 2012).

Researchers like Vidana-Zavala et al. led research and design efforts to bring low-cost WC solutions for low-income countries to improve the quality of life for WC users in these societies (Vidana-Zavala et al., 2019). We strive to answer the question of how to provide smarter WC technology at an affordable price.

In this work, we demonstrate user-independent obstacle detection and avoidance as an autonomous smart WC feature using computer vision. It is achievable thanks to advanced technologies in computer vision and deep learning. Using a single low-cost camera sensor with a WC, we aim to reduce the costs of using expensive sensors such as LiDAR and RGB-D cameras, while retaining the same functionality.

For robots using the sidewalk, as with WCs, there is a lack of datasets for deep-learning model training. This motivated us to build our own essential dataset, the Wheelchair Obstacle Detection Dataset (WODD). Obstacle detection is achieved by retraining the deep-learning model using the dataset that we developed for that purpose. The ability of the network to detect objects in the image has enabled us to develop image-space-obstacle avoidance, which uses an image-based visual servo control law (Corke and Hutchinson, 2001).

1.3 Contributions

This work, to the best of our knowledge, is the first to contribute the following to the WC system:

(i) Deep-learning obstacle detection for a WC system, deployed using SBC.
(ii) Development of a publicly accessible dataset of potential obstacles linked to sidewalk environments to allow the learning transfer of the model.
(iii) Image-space WC obstacle avoidance without using any 3D information.

During the design and development of our system, we focused on setting the following criteria:

(1) Affordable: Many users already have their own PWC and cannot afford a new smart WC. Thus, we designed a low-cost system to upgrade any PWC, with a target price of <$300. We used only commercial off-the-shelf electronics, which have lower prices than industrial ones.
(2) Deployable: To make our solution adaptable by the community, it must be easy to install with minimal modifications to the existing WC. Therefore, any PWC can be

adjusted by bypassing the joystick and connecting the motor's lines directly to our controller, which fits in a single box. Other avoidance systems that require additional depth sensors, such as ultrasonic transfers and LiDAR will need additional installation to the WC body. With ours, this is not the case.

(3) Portable: Our system runs on embedded computational platforms, such as single board computers (SBC) with a single monocular camera input, and a power bank or external battery. This avoids the need for a notebook to run process-heavy code. It only needs to be connected to the WC motor terminals.

Meeting these criteria gives our proposal a higher degree of usability and makes it easily adaptable. Indeed, it may help a wider range of WC users throughout the world.

1.4 Organization of the thesis and publication

The content of this thesis is organized as follows:

1. "Background" (chapter 2) explains the main techniques used in this thesis for obstacle detection using deep learning and obstacle avoidance using visual-servoing technique.
2. "Literature review" (chapter 3) contains a review of the important works for WC with obstacle detection and avoidance feature.
3. "Deep-learning obstacle detection and vision-only avoidance" (chapter 4) contains details about the proposed approach.
4. "Experimental evaluation and analysis" (chapter 5) presents the results of the experiments done through this thesis using a PWC.
5. "Conclusion" (chapter 6) is to give a brief about the outcomes of this work and some headlines for possible directions for future work.

We would like to mention that we have published part of this thesis work in an IEEE conference in Turkey called Innovations in Intelligent Systems and Applications Conference for year 2022 (ASYU 2022). The paper is published in IEEE Xplore under the title "Innovations in Intelligent Systems and Applications Conference" (Tawil and Hafez, 2022). Another submission is made to Disability and Rehabilitation: Assistive Technology journal under the title "A practical realization of a deep learning low-cost obstacle detection and avoidance system for powered wheelchair using a single camera" and it is revised and resubmitted after getting a major revision.

CHAPTER 2

BACKGROUND

This work suggests utilizing a refined deep-learning model to identify barriers in the collected photos as a solution to the obstacle detection and avoidance problem. Later on, a developed visual servoing control law will calculate the needed speeds to push the obstacle to the safety margin of the image. Therefore, we will introduce in this chapter the background of using deep-learning models for the object detection task. We will also talk about the principles of visual servoing which is used to develop the control sub-module.

2.1 Tradition Computer Vision vs Deep Learning Computer Vision

Convolutional neural network (CNN) is commonly used after training to extract feature maps for visual analysis, which is a main requirement in object detection. CNN models can be re-trained using a custom dataset for any use-case, contrary to computer vision CV algorithms, which tend to be more domain-specific. A deep learning model is 'trained' on the given data, where neural networks discover the underlying patterns in classes of images automatically. As shown in Figure 2.1, in the traditional CV approach, it is necessary to choose which features are important in each given class/case (O'Mahony et al., 2019).

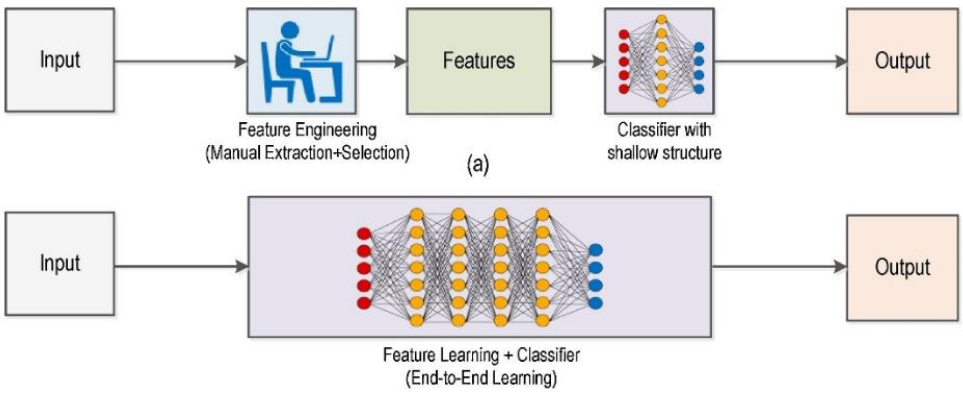


Figure 2.1 An illustration, from (O'Mahony et al., 2019), showing the difference between traditional computer vision (a) and deep learning workflow (b).

2.2 Evolution of object detection models

In image classification, we pass an image to a deep learning model to determine what that image composes of, and this will classify the entire image. Object detection comes when we need to identify multiple objects that can be anywhere in the image and it figures what size they are and where they occur. Many times object detection uses bounding boxes around of regions in the image and classifies the objects that have been detected. The naive way to find the regions that contain the object in the image is to set a sliding window with a predefined size that scans the whole image with a known stride. Each window in each step is sent to a classification model to know if that window has background or object.

Recent models grow out of this and one of the first break through models we got is R-CNN. R-CNN use the sliding windows as the first step, but the windows are sent to a region proposal model that defines different interest regions that could be containing an object. The proposals are fed to a CNN to extract feature points, and then using a classifier like SVM a class is predicted. Later, the regression step will give the final output of bounding boxes. The R-CNN steps are depicted in Figure 2.2. Feeding the regions of interest sequentially to the CNN with numerous iterations, slows down detection speed of R-CNN.

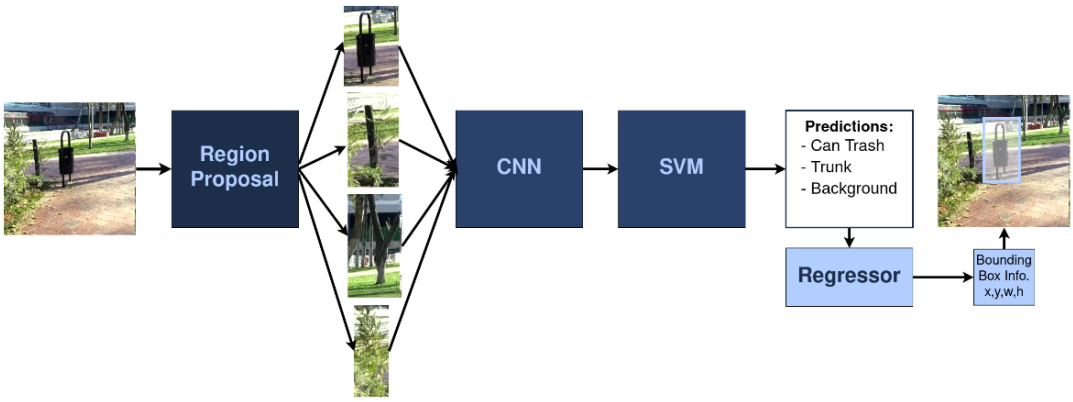


Figure 2.2 Object detection steps in R-CNN.

Object detection models in many cases will run in a processing-contained device and R-CNN is not suitable for such. To solve this, other object detection methods proposed a new structure that can classify the whole proposed regions in one pass which doubled the speed of detection (Lee et al., 2017). Single Shot multibox Detector (SSD) uses a backbone CNN to extract image feature map and then do convolution at different scales. Finally, using the output of the convolution layers, a non-maximum suppression

(NMS) block gives the bounding boxes with class predictions. As shown in Figure 2.3, all is done in a single shot.

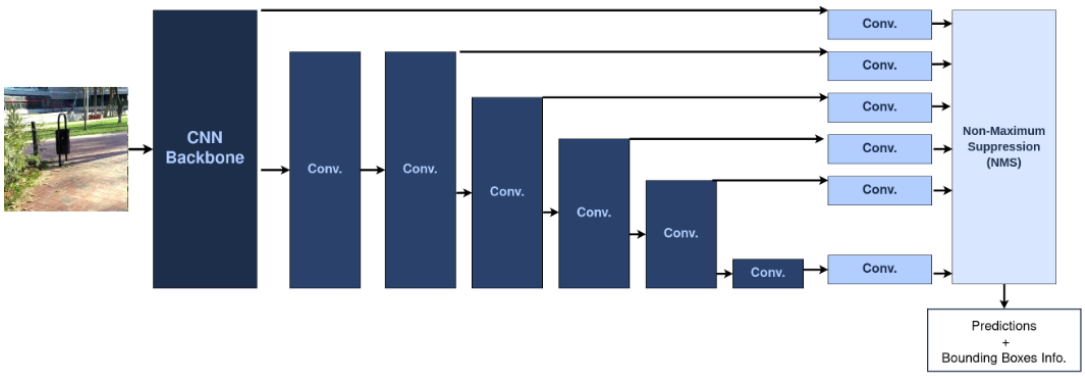


Figure 2.3 Object detection steps in SSD

The backbone CNN of SSD can be any classification model like MobileNet, shown in Figure 2.4. Depth-wise separable layers are used in MobileNetV2 with SSD combination instead of regular convolutions for the object detection part of the network, with better real-time performance. MobileNetV2 architecture contains an initial convolution layer with 32 filters, followed by 19 residual bottleneck layers whereas SSD is a one-stage detector.

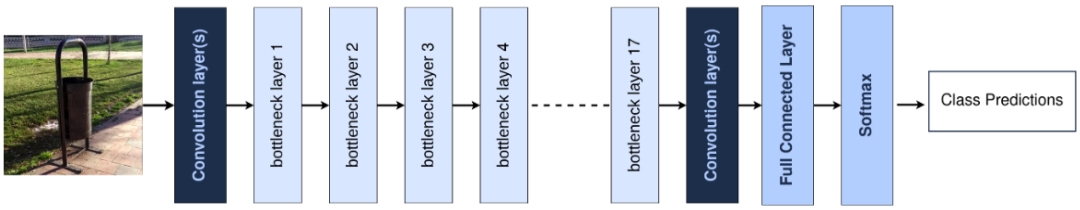


Figure 2.4 MobileNet V2 Architecture.

2.3 Visual Serviong

Visual servoing extract from a goal image some of its visual features to be used in the main task of controlling the pose of the robot ‘s end effector (camera). There are 2 approaches in visual servoing: Position-Based Visual Servo (PBVS) and Image-Based Visual Servo (IBVS). In PBVS, the pose of the goal is determent with respect to the camera using the extracted visual features and a geometric model of the goal. In IBVS, the visual features are directly used in the control law. In this work, we will use the IBVS as described later in control law design chapter.

Figure 2.5 shows the basic idea behind IBVS, which is bringing the initial selected feature point(s) to the target feature point(s).

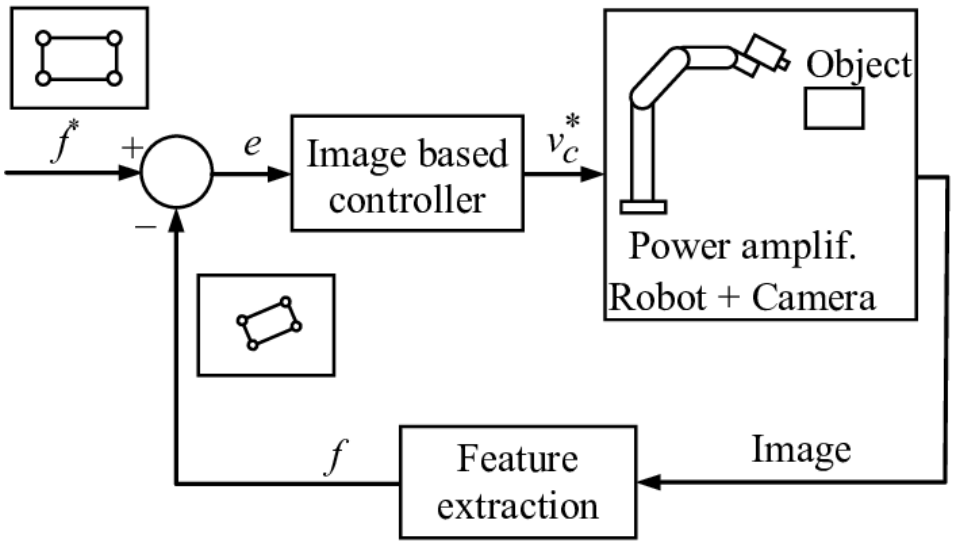


Figure 2.5 The basic steps in IBVS. Figure is adapted from (Corke and Khatib, 2011)

Figure 2.6 provides a clearer illustration of bringing the original selected feature point(s) to the target feature point(s) in IBVS.

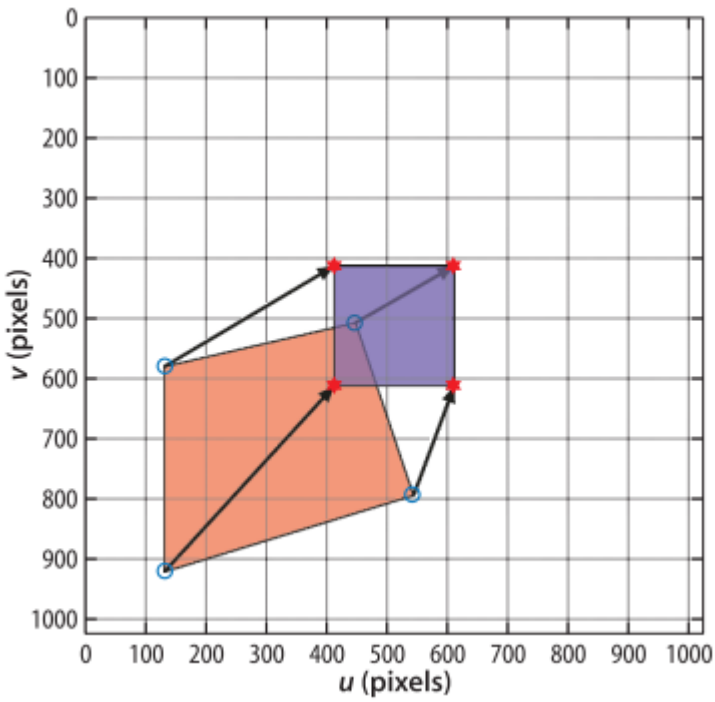


Figure 2.6 An example of a target square shape (blue) and the initial shape (red). The task is to move the feature points indicated by o-marks to the points indicated by *-marks. The figure is adapted from (Corke and Khatib, 2011)

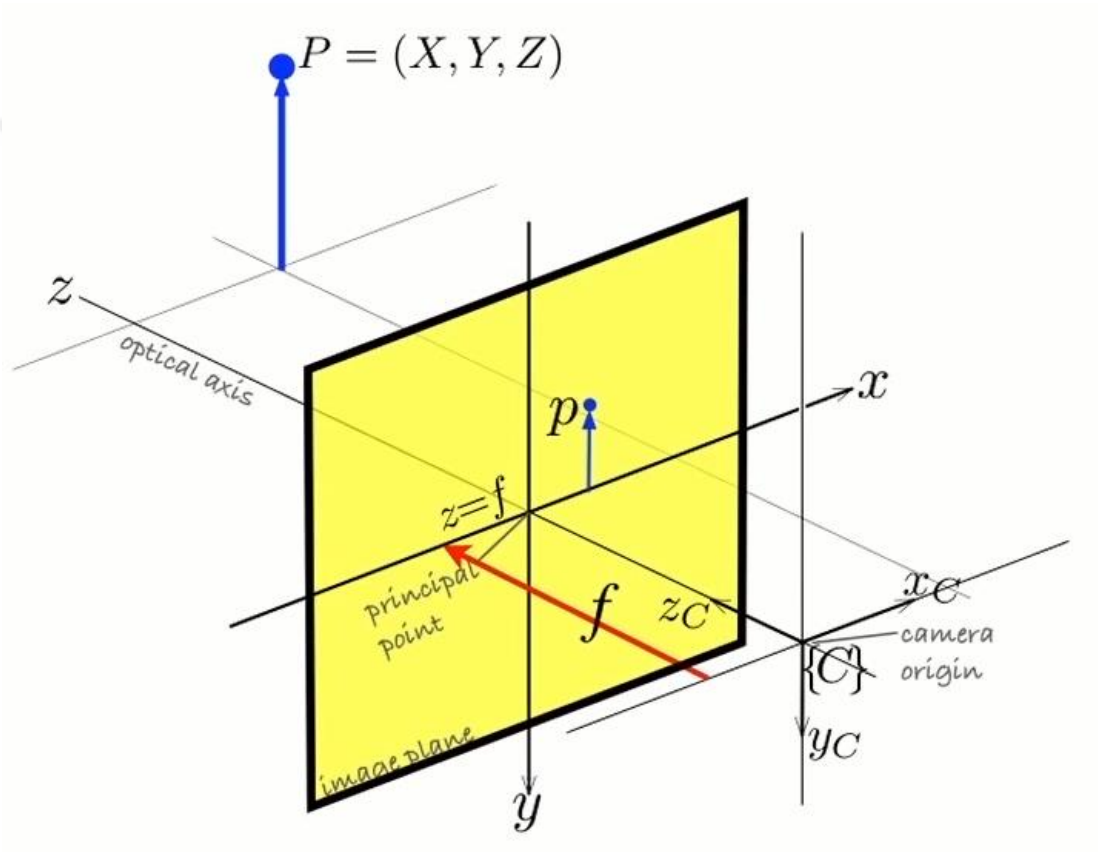
To map the movement in the world frame with the camera plane, we use the interaction matrix.

$$\mathbf{L}_x = \begin{pmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{pmatrix} \tag{2.1}$$

The value Z is the depth of the point relative to the camera frame, and x,y are the pixel position. This matrix is used in the following equation which maps the change of feature point position in image-space with the moving velocities in the physical world:

$$\dot{x} = \mathbf{L}_x v_c \tag{2.2}$$

The interaction matrix is derived starting with the central projection model to normalize image-plane coordinates to get the perspective projection of 3D world point in the 2D image plane. Figure 2.7 illustrates the projection's fundamental concept.



With kind permission of Springer Science+Business Media

Figure 2.7 Shows the central projection model which consists of Camera coordinate C with image plane (in yellow) with distance f, a point P in homogeneous coordinate and a projected point on the image plane p. Figure is adapted from robotacademy.net.

The projection is expressed mathematically for a three-dimensional world point with coordinates X; Y; Z in the camera frame projects into the image plane of a conventional perspective camera as a twodimensional point with normalized coordinates x; y as follows:

$$x = \frac{X}{Z} = \frac{u - c_u}{f\alpha}$$
$$y = \frac{Y}{Z} = \frac{v - c_v}{f} \tag{2.3}$$

Where u,v are the position in the image space and f is the camera focal length. As we are tying to relate the change of the pixel position with the physical speed, then we will take the time derivative of the projection equations

$$\dot{x} = \frac{\dot{X}}{Z} - \frac{X\dot{Z}}{Z^2} = \frac{\dot{X} - x\dot{Z}}{Z}$$
$$\dot{y} = \frac{\dot{Y}}{Z} - \frac{Y\dot{Z}}{Z^2} = \frac{\dot{Y} - y\dot{Z}}{Z} \tag{2.4}$$

We relate the 3D point velocity to the camera spatial velocity using the following equation

$$\dot{X} = -v_c - \omega_c \times \boldsymbol{X} \Leftrightarrow \begin{cases} \dot{X} = -v_x - \omega_y Z + \omega_z Y \\ \dot{Y} = -v_y - \omega_z X + \omega_x Z \\ \dot{Z} = -v_z - \omega_x Y + \omega_y X \end{cases} \tag{2.5}$$

Figure 2.8 present the concept of mapping the move of the 3D point in real word with the image point projected to the camera plane.

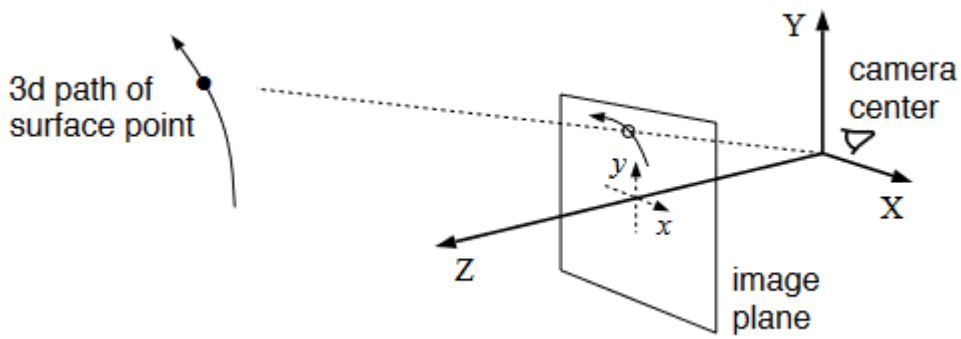


Figure 2.8 Another illustration showing the principle of point projection between 3D world and image plane with respect to the camera center. This figure is adapted from Notes on Motion Estimation.

By using projection equations and inserting the 3 components of speed and acceleration into the last equation we find

$$\dot{x} = \frac{-v_x}{Z} + \frac{xv_z}{Z} + xy\omega_x - (1 + x^2)\omega_y + y\omega_z$$
$$\dot{y} = \frac{-v_y}{Z} + \frac{yv_z}{Z} + (1 + y^2)\omega_x - xy\omega_y - x\omega_z \tag{2.6}$$

And this can be written as $\dot{x} = \mathbf{L}_x v_c$, leading us to the interaction matrix shown in equation 2.1 .

CHAPTER 3

LITERATURE REVIEW

In this chapter, we will review the most important previous works related to PWC obstacle detection and avoidance, either in academia or commercially. We will highlight the main contribution behind each work and highlight the limitations in the past work, when needed, compared to our proposal. We will also review the datasets that target similar environments to ours and prove the gap that leads us to develop our own dataset.

3.1 Reviewing the powered wheelchairs with obstacle avoidance

Several researchers investigated WC obstacle-avoidance. For example, (Diao et al., 2017) and (Petry et al., 2010) used an artificial potential field controller to avoid obstacles detected using a redundant number of ultrasonic sensors, which proved to be less reliable and accurate than the camera sensor. In another work by (Özkörs et al., 2014), they used the Kinect sensor to extract the depth information which is used to calculate the forces of the potential field. Nevertheless, the Kinect sensor is a costly alternative to the Raspberry Pi Camera used in our work, in addition to which they used Cartesian space information, rather than the image space that we adopted.

A few WC studies have been done using classical computer vision algorithms (Kim, 2016; Lee et al., 2017), which are currently outperformed by a deep-learning approach. Kim in (Kim, 2016) used a combination of multiple ultrasonic sensors surrounding the WC body and a camera to implement driving assistance system functionality for obstacle detection and avoidance, which can recognize two critical environmental situations in front of the WC, traffic intersections, or roadways in front of the WC.

Lee et al. in (Lee et al., 2017) provided an obstacle-detection algorithm using a single camera, but the performance was unreliable, due to their use of primitive thresholding-based image processing and Canny Edge detection. Moreover, their system required a notebook to run which limits the system portability.

WC avoidance, localization and negotiation of doorways are considered by works (Li, 2017) and (Burhanpurkar et al., 2017) where a dynamic window method is used for avoidance and adaptive Monte Carlo for localization. Additionally, Burhanpurkar et al.

used costly RGB-D camera and they targeted indoor environments. To reduce the costs of using 3D imaging, we used an affordable monocular camera and targeted an outdoor environment.

Classical computer vision methods mostly rely on feature points extracted from the scene, and later an algorithm is developed with an expert analysis. In deep learning, the model is trained to find the required visual features based on the data provided. The resultant model can be reused with new objects and requires only fine-tuning using transfer-learning with enough data for the new objects. Moreover, deep-learning models have shown superior performance in computer-vision applications like object detection and classification. This fact is thoroughly investigated in the literature (O'Mahony et al., 2019).

Object detection comes when we need to identify multiple objects that can be anywhere in the image and work out what size they are and where they occur. Object detection often uses bounding boxes around regions in the image and classifies the objects that have been detected. Figure 3.1 shows a very basic way of finding the areas in the image that contain an object, using a sliding window.

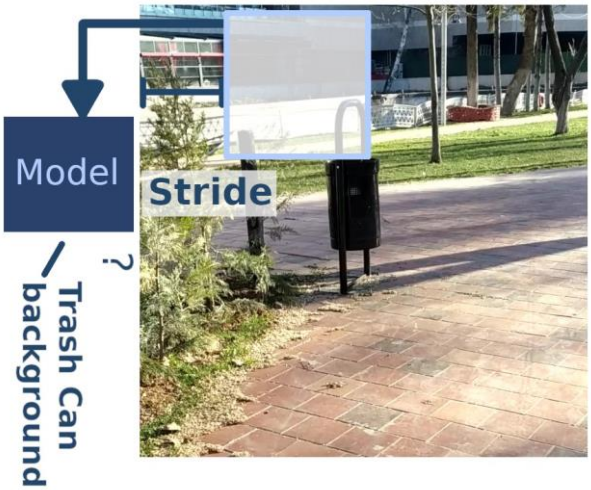


Figure 3.1 Primitive way of finding of object detection using a sliding window with fixed stride and size.

A CNN-based system is used by Ali et al. in (Ali et al., 2018) to make an autonomous boarding to the bus by detecting the bus and whether its door is open. They used YOLO and a LiDAR to judge whether the door was wide enough to take the WC. This work is a step in the development of the autonomous WC, yet it has limited application. Although the autonomous bus-boarding work is valuable and novel, obstacles still need to be detected and avoided on the way to the bus station.

The work captured in (Haddad et al. 2020) suggests a set of steering actions for WC users as an assisting tool rather than autonomous driving and uses a deep-assessing neural network based on the Long Short Term Memory (LSTM) neural network. Haddad et al. work takes distance information input from three ultrasonic transceivers to sense the surrounding obstacles and suggest a correct direction in response.

The work in (Day et al., 2019) used deep-learning object detection and an SBC (Raspberry Pi 3) for a PWC, but this was for pedestrian detection with collision avoidance using ultrasonic sensors. The collision avoidance in their work is simplified by using a two commands algorithm (stop and reverse) depending on the distance-to-obstacle threshold. To the best of our knowledge, Day's is the only work to use deep-learning object detection on Raspberry Pi for WCs. However, their work was potentially limited by the fact that deep-learning object detection was not combined with avoidance in a single system. Moreover, they used a pre-trained model that detected pedestrians only. Therefore, other obstacle hazards, such as fire hydrants, potholes and rocks could not be detected. Our fine-tuned deep-learning model overcomes these limitations by using an image-based control law to make the avoidance.

Table 3.1 summarizes the related past works, and focuses on past research and commercial works that use vision sensors. Although commercial smart WCs are widely available, we found only two products with an anti-collision feature. The first is WHILL, which uses a pair of stereo camera sensors. The second is Smile Smart System PWC, which uses a pair of ultrasonic transceivers. Other products like Scewo Bro, Invacare TDX SP2, Scoozy, Delta Autour A1 and Permobil F3 Corpus focus on remote control through smartphones and on adding innovation to WC mechanical functionality and chassis. We conclude from the Table that object detection using computer vision is not yet adopted in the commercial PWC products. Moreover, previous researches presented costly solutions using industrial-type sensors or require a notebook to run the system code. Alternatively, they use affordable sensors but use less reliable or limited techniques. This both demonstrates the "cheap or robust" dilemma and highlights the importance and novelty of our work.

It can be concluded from the above reviews that most previous works proposed costly systems and only a few employed deep-learning algorithms, but with limited applications. By contrast, we present a novel low-cost system that employs an off-the-shelf computer and single-vision sensor, using a deep-learning model for obstacle detection and image-space avoidance.

Table 3.1 A Review of research and commercial PWCs with obstacle detection functionality.

Type	**Reference**	**Hardware Setup**	**Cost**	**Main contribution (+) limitations (-)**
Research	Kim 2017	1 x CCD Dragonfly camera 1 x NI DAC USB-6009 8 x Ultrasonic sensor	$700*	+ Anti-collision by combining ultrasonic and computer vision. + Road with sidewalk intersection recognition. – Assistive system. Not autonomous. – No control law provided; system gives fixed commands (Left, Right, straight, Back). – Lack of deployment on SBC or embedded platform.
Research	Lee et al. 2017	1 x Logitech webcam 2 x Compass module	$70*	+ Low-cost. – Primitive thresholding-based image processing. – Lack of deployment on SBC or embedded platform.
Research	Burhanpurkar et al. 2017	1 x Microsoft Kinect V2 1 x Motor encoder 1 x On-board computer	$2000**	+ WC Navigation with obstacle avoidance. – High-cost.

Commercial	Whill 2022	x Stereo camera sensors 1 x Customer interface tablet 1 x On-board Computer	$4000	+ Navigation using preinstalled maps. + Remote Real-time monitoring through dashboard. – Obstacle collision preventing without avoidance functionality. – Not portable.
Research	Maule et al. 2021	1 x Microsoft Kinect V2 1 x Motor encoder 1 x On-board computer	$2000	+ Hybrid driving opinions (eye tracking & semiautonomous). – High-cost. – Lack of deployment on SBC or embedded platform.
Research	Day et al. 2017	1 x Raspberry Pi 3 1 x Raspberry Pi camera HC-SR04 ultrasonic Sensors	$200	+ Running using affordable SBC (Raspberry Pi). + Cost efficient. – No control law provided; system gives fixed commands (stop and reverse). – Deep learning and computer vision methods are not integrated with WC control. – Destructively reused the WC's Joystick, making it undeployable by normal users.

* Excluding notebook used for deployment.

** Reported by the authors

3.2 Reviewing the existing datasets

The research community working on autonomous vehicles has produced several datasets. However, only a few of them, and with small sizes, are suitable for the autonomous WC in general and for travel on sidewalks in particular.

One example of the sidewalk datasetes is SideGuide (Park et al., 2020). It has been developed to bridge the dataset gap for robots' use of sidewalks. Samples from sildwalks dataset are shown in Figure 3.2.

Figure 3.2 Samples from Sidewalks dataset after drawing the bounding boxes using the provided information in bbox_sample.xml file. Only samples can be downloaded publicly.

The Chinese Driving from a Bike View (CDBV) dataset (He et al., 2019) is another example captured using a moving bike. Unfortunately, there is no access to SideGuide and CDBV beyond the countries in which they were developed.

The Sidewalk Obstacle Image Dataset (SOID) presented in (Ahmed, 2017) aimed to develop a deep-learning model to allow the recognition of objects in different lighting conditions. SOID is used for classification only, meaning it is not annotated. In addition, it consists of small images, 96x96 pixels as shown in Figure 3.3, which makes it unusable in our research. The dataset is packed in pickle format which is a file format to convert the images in Numpy array (or python objects in general) to a stream of bytes that can be stored in the hard disk. So you can load it in the code. Figure 3.4 shows the snippet used to load samples from SOID dataset.

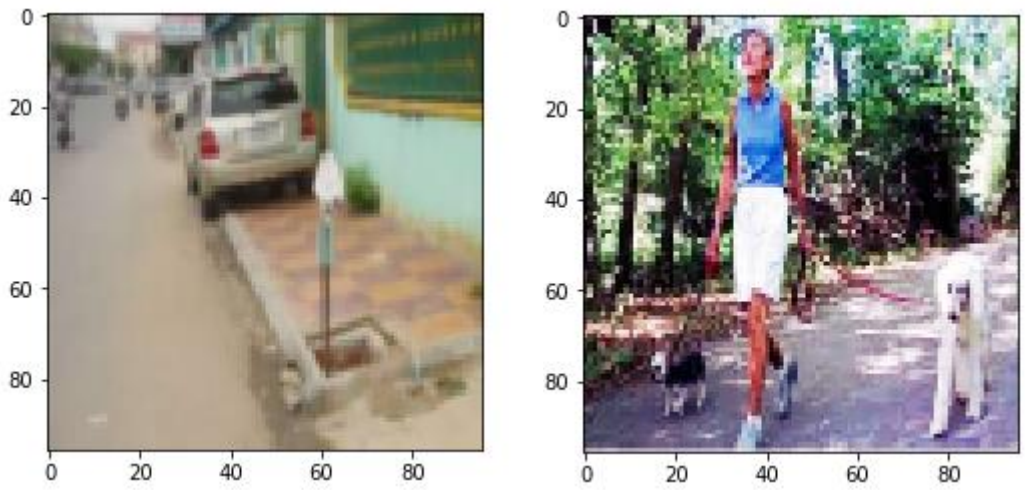


Figure 3.3 Samples from SOID dataset showing the 96x96 pixles size.

```
import pickle
import matplotlib.pyplot as plt

infile = open("sidewalk_rgb_all.pkl",'rb')
u = pickle._Unpickler(infile)
u.encoding = 'latin1'

new_dict = u.load()
infile.close()

(X_train,Y_train),(X_valid,Y_valid),(X_test,Y_test) =
new_dict

plt.imshow(X_train[200].reshape((96,96,3)))
```

Figure 3.4 code snippet used to load samples from SOID dataset pickle file.

A dataset called PESID (Sun et al., 2019) was created to develop a solution to alert pedestrians using smart devices while walking to possible accidents. The last is divided into sub-datasets to deal with different sidewalk environments. However, we could not get access to it either.

The Mapillary Vistas dataset (Neuhold et al., 2017) can be used for similar applications for sidewalks, as shown in Figure 3.5, and although it was not created specifically for sidewalks, it contains both sidewalk and road views. However, the images for sidewalks are very limited.

Figure 3.5 Samples from Mapillary Vistas dataset show some examples of sidewalk views.

Table 3.2 summarizes the number of images, number of classes and annotation type in each dataset we have mentioned. These datasets cannot be effectively used in other projects, as all the datasets mentioned in the table, other than Vistas and SOID, are proprietary without open access, which led us to build our own dataset.

Table 3.2 The different datasets built for or used in object detection for robots that use side-walk environments

Dataset	**Size**	**Annotation**	**Classes**
SideGuide	350K	Bounding box and semantic polygon	29
SOID	50K	X	5
Vistas	25K	Semantic polygon	37
PESID	1.9K	Bounding box	10
CDBV	13.5K	Bounding box	6

We chose nine common obstacles usually found on sidewalks. The instances are annotated with bounding boxes in the image. The dataset is open-access with no restriction policies.

CHAPTER 4

DEEP-LEARNING OBSTACLE DETECTION AND VISION-ONLY AVOIDANCE

This chapter describes our proposed system for WC obstacle detection and avoidance, which consists of an obstacle-detection module built on a deep-learning network and an avoidance controller built using a simple visual serving control law to avoid the detected obstacle. A low-cost computer, Raspberry Pi, along with a motion controller and power drive are used to perform the computation of the motion commands and feed them to the WC's motors.

4.1. Our system overview

We have integrated different technologies in our design to create a single and novel system for WC obstacle detection and avoidance. The deep-learning model is carefully chosen to provide the required precision and to run on a portable and limited resources computer such as Raspberry Pi. This is discussed in sub-section 4.2. In addition, we have built our dataset to represent the environment of the WC. The WODD is used to transfer the learning from the original object-detection domain, in which the deep-learning model was initially trained, to the WC obstacle-detection domain in which the WC is supposed to operate. The dataset is presented in sub-section 4.3. Designing and evaluating a visual servoing-based control law in the image space that pushes the obstacle from the WC's area of interest (AOI), as it appears in the camera field of view, forcing the WC to avoid the detected obstacle. This is shown in sub-section 4.4.

The overall process is illustrated in Figures 4.1 and 4.2. The block diagram in Figure 4.1 shows the functionality flow of our system with hardware and software components and how they interact with each other, while Figure 4.5 shows a flow chart of our algorithm with the avoidance steps, which are followed by the system when the obstacle is inside the AOI.

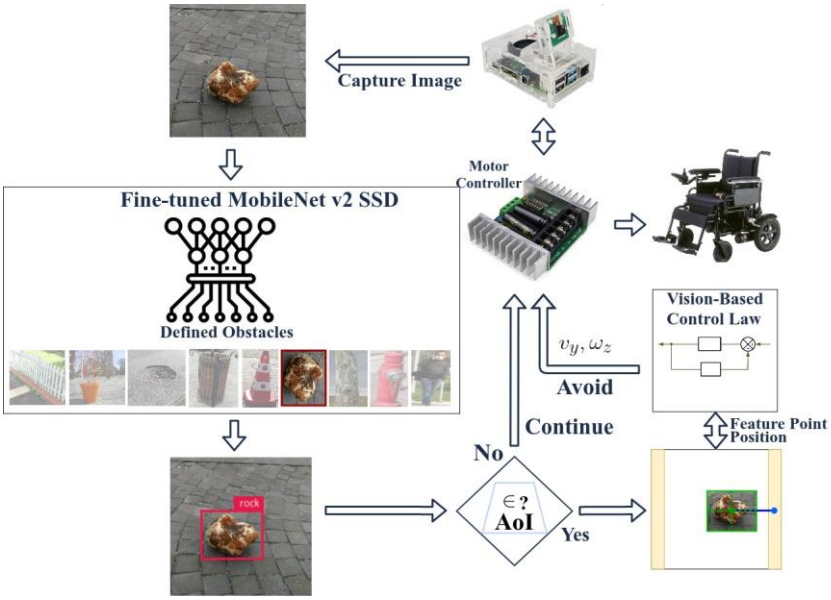


Figure 4.1 An illustration of our developed system blocks

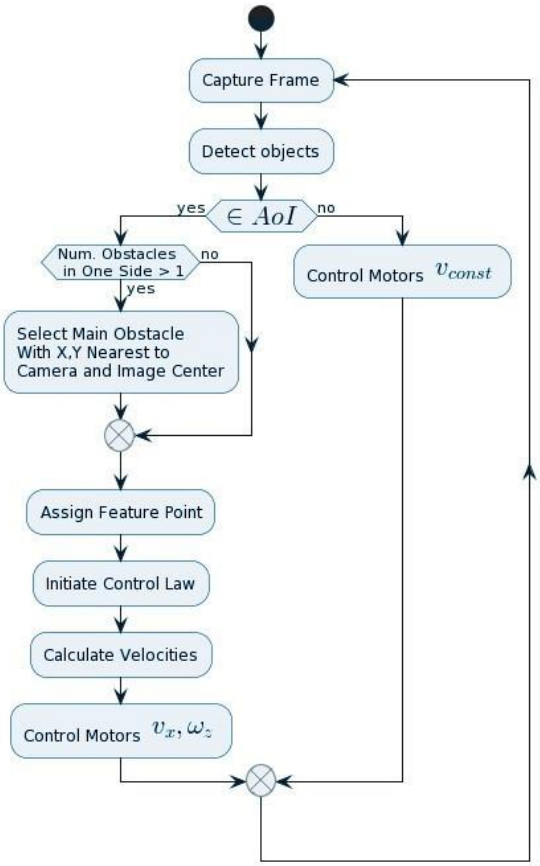


Figure 4.2 A flowchart showing the main tasks in our system.

4.2 Deep-learning model selection for obstacle detection

A computationally expensive way to find the regions in an image that contain an object is to set a sliding window with a predefined size that scans the whole image with a known stride. Each window in each step is sent to a classification model to know if that window has just a background or contains an object. After that, a regression step predicts the size and the position of the bounding boxes. Region-based detection models use a region-proposal module to produce different regions through different window sizes and a CNN for feature extraction. Such detection models are called region-based convolution neural network (R-CNN) models.

Unfortunately, such models are not suitable for deployment in our low-cost hardware, Raspberry Pi, whose computational resources are comparatively limited, as the R-CNN models are computationally expensive, due to their sequential nature. To meet our computation-speed requirements, we adopt newer models for detection. They achieve faster detection as they pass the regions to the regression model in parallel. Such models are well-known currently in the literature and are called single-shot multi-box detectors (SSD). As shown in Figure 4.3, all is done in a single shot.

The CNN model is built above the one presented in (Sandler et al., 2018), which is called MobileNetV2. Its CNN architecture serves as an image feature extractor for detection and classification. It is suitable to meet our real-time requirements, as it has a light architecture and was originally designed to run on mobiles and embedded hardware.

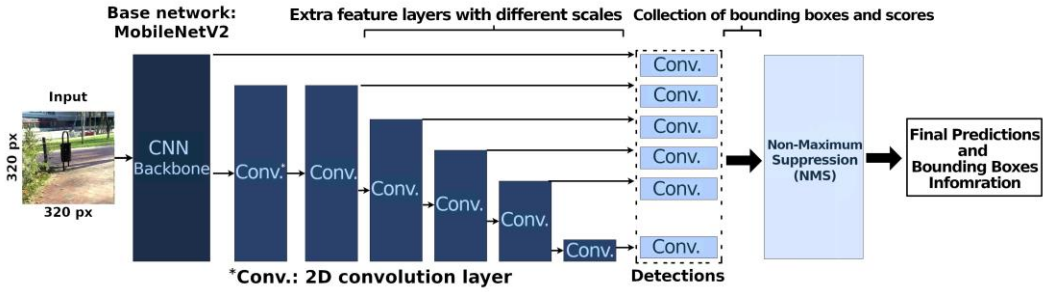


Figure 4.3 Object detection steps in our deep learning model. The first one is an image feature extractor backbone which is selected to be similar to the one adopted in (Sandler et al., 2018). The output of conventional layers at different scales are fed to non-maximum suppression predictor similar to (Lee and Kim, 2020). This architecture does the object detection in one shoot.

Figure 4.4 shows a comparison between different state-of-art image classification networks in terms of the number of parameters and model accuracy. We can spot how MobilenetV2 compromises between size and accuracy.

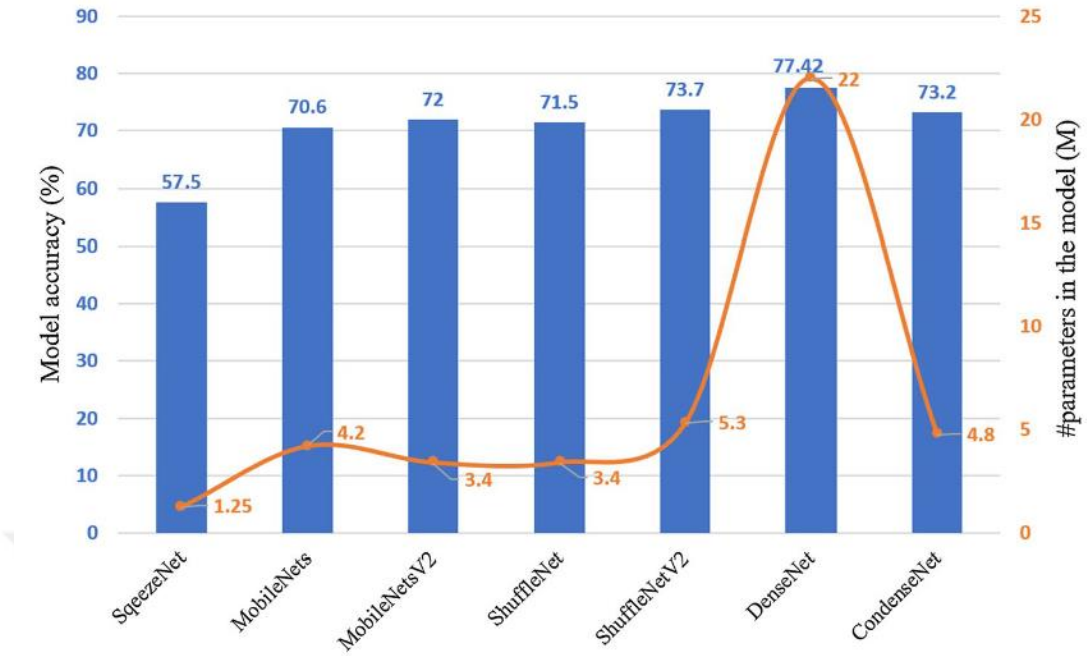


Figure 4.4 The model size and accuracy among different network architectures trained on ImageNet dataset. Image is adapted from (Choudhary et al., 2020).

The combination of the two selected models produces highly desirable features in terms of lightness and speed of execution. The resultant model has satisfactory performance and accuracy on Raspberry Pi 4, which is our adopted hardware. Similar combinations in the literature have reported a superior performance compared to classical region-based detection methods. This model takes an RGB image with a 320x320 pixel size as input and returns the positions of the detected boxes in the output, along with its prediction of percentage value.

General purpose selected deep-learning models are originally pre-trained using the COCO 2017 dataset with 80 classes. This is retrained to be customized to our application through a process known in the literature as “learning transfer”. In this process, a new dataset is collected in such a way that it represents the new application environment. After that, the model is retrained using the dataset to learn the new environment. The next sub-section presents the process of collecting and annotating the dataset, and then customizing the model to suit our application. We defined nine obstacle classes in our dataset to be detected and yet the model is customized to have only nine classes. The box regression and classification heads of the detection model were retrained without the feature extraction part as the feature extraction is common to all obstacles and objects.

4.3 The new WODD dataset

In this sub-section, we describe the process of data collection, annotation, and preparation for the learning transfer. We captured and prepared the images in Gaziantep, Turkey, over two months (one month in summer and one in winter) and by traversing around 2.5 KM of sidewalks multiple times.

4.3.1 Data collection

Our dataset (WODD) was captured using a mobile-phone camera fixed to a WC that was used to collect videos on our university campus and our city sidewalks. Additionally, we used web scraping using Google images and Flicker services for collecting images of sidewalks. The images we collected from our WC view represent 75% of the total images in WODD. The frame size of the mobile camera was 1920x1080 pixels, which were cropped to 1080x1080 pixels to match the input format of the CNN. We extracted a total of 1411 annotated images from the captured videos, with bounding boxes available in our dataset and distributed over nine different classes.

We added various types of noise and visual effects to the images to replicate the different lighting conditions that might be encountered in an outdoor environment. Figure 4.5 presents a set of sample images showing the different types of applied augmentation. Quality adjustment is the first type of augmentation, converting an image to UINT8 representation, encoding it to a jpeg using the desired encoding quality, decoding it, and then converting back to the original data type. This was done to address The difference in image quality between the high-quality camera used to collect the dataset and the Raspberry Pi Camera.

The second type of augmentation was brightness correction using a Gamma effect to slightly darken the images, most of which were collected on sunny days. This converts our RGB image to a float representation, adjusting each pixel value r to $C.r^{\gamma}$ where C is a gain constant and γ is variable and, after that, converting them back to the original data type. The third type of augmentation is the addition of noise using random values from a Gaussian normal distribution added to the image pixels value. The final type of augmentation was the contrast modification. The contrast was changed for each image channel. The mean m_r of the image pixels in the channel is computed each value r of each pixel, and is adjusted to $(r - m_r)f_c + m_r$. Here, f_c is the contrast factor.

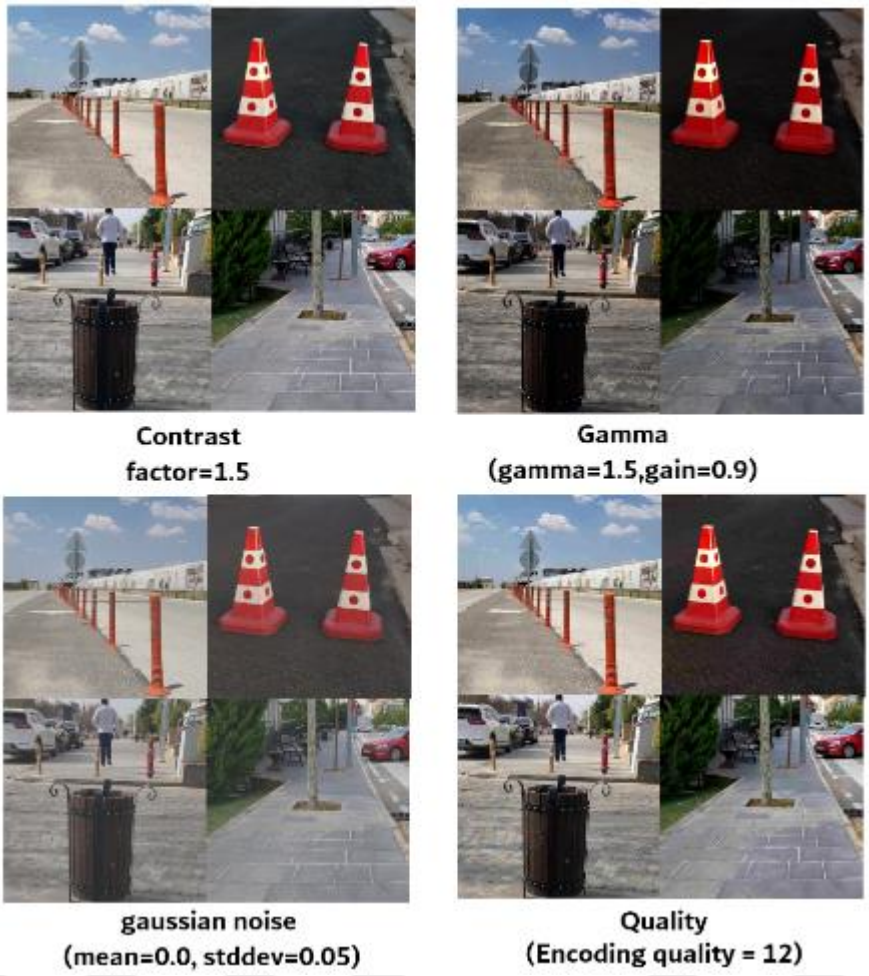


Figure 4.5 Sample images from WODD showing the four augmentation effects. They are from left to right: Contrast adjustment with fc = 1.5, Gamma effect transformation with $\gamma = 1.5, C = 0.9$, Gaussian noise with $\mu = 0.0, \sigma = 0.05$, and at the most right is Quality adjustment with encoding quality=12.

4.4 Dataset images annotation and learning transfer

We carefully selected the obstacle classes that WC users found challenging (Edwards and McCluskey, 2010), and the most common obstacles found on the sidewalks in our city. The nine classes are: pothole, potted plant, fence, traffic sign, trash can, fire hydrant, person, rock, and trunks. The distribution of data over classes is depicted in Figure 4.6. The ratio of training to testing sets is 90:10.

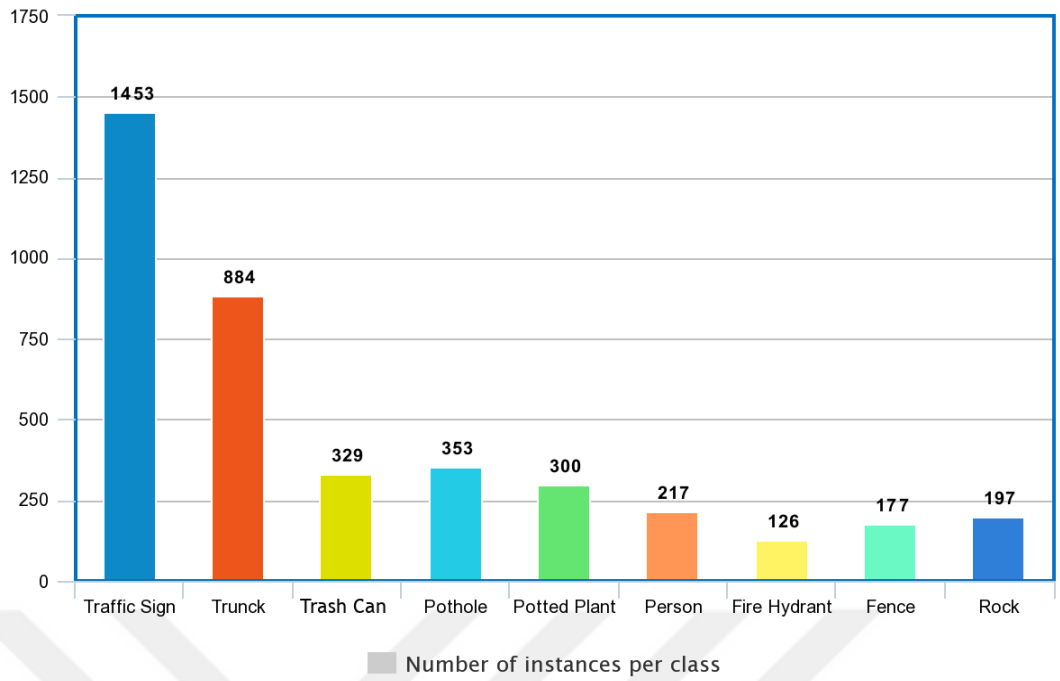


Figure 4.7 A bar chart showing the distribution of the 3976 instances over classes in our WODD dataset. The Traffic Sign and trunks classes are generally common among places. This explains the relatively larger number of instances in these two classes.

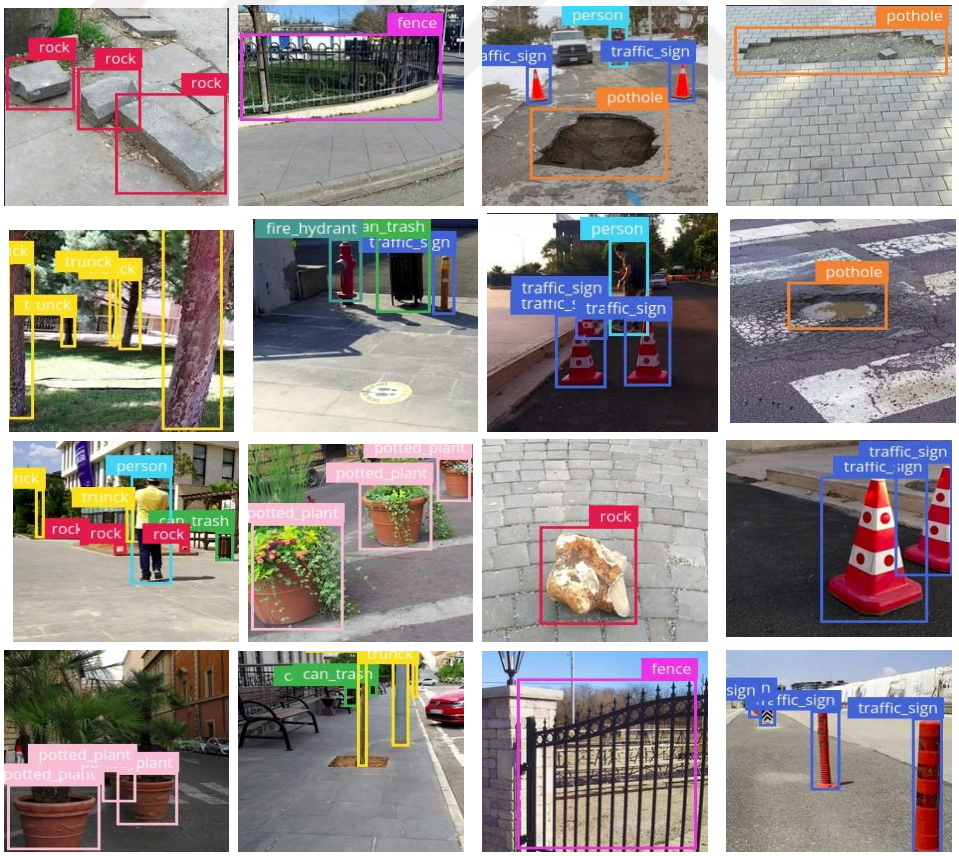


Figure 4.6 Samples from the annotated images contained in our WODD dataset.

It is shown in (Lin et al., 2018) that 150─500 images per class are enough to get reasonable image-classification accuracy. We show some samples of annotated images from each class in Figure 4.7.

The 1411 images in the dataset were annotated in the format of a bounding box. The WODD dataset size is 1.0 GB and can be downloaded publicly too.[1] The dataset is divided into two folders (training and testing). Each folder has a JavaScript Object Notation [JSON] file that includes the required information of each image bounding box as a JSON object. The information for each bounding box is: (a) image file name; (b) label name; (c) box position and (d) box dimension (min x, min y, max x and max y) in pixels. As we prepared the images with square dimensions 1080x1080, it is compatible with the 320x320 input for our network and makes scaling the bounding boxes' position and dimension using a single division factor.

The number of instances in our dataset is not balanced. This is a normal result of how frequently the objects appear in the surrounding environment. For example, traffic signs will be seen more than fire hydrants. Many machine-learning and deep-learning models are affected by the biased dataset during training. This leads to a partial or entire degradation in performing predictions for the minority class.
To solve the problem of having an unbalanced dataset, we use a resampling strategy during training. The resampling has two approaches, as shown in Figure 4.8. undersampling and oversampling. The first compose of removing samples from the dominated class until it becomes similar to the minority class. The latter is composed of adding more samples to the minority class. The oversampling can be done using data augmentation. Readers may refer to (Liu et al., 2014) for more information about the resampling in object detection deep- learning models.

Velasco et al. (Oksuz et al., 2020) showed in their research the improvement of Mobilenet model classification results when resampling techniques were used. However, our experiments show that the accuracy we have achieved is accepted in our problem as we have tolerance in detecting the "right" class of obstacle.

[1] https://bit.ly/3BPR06U

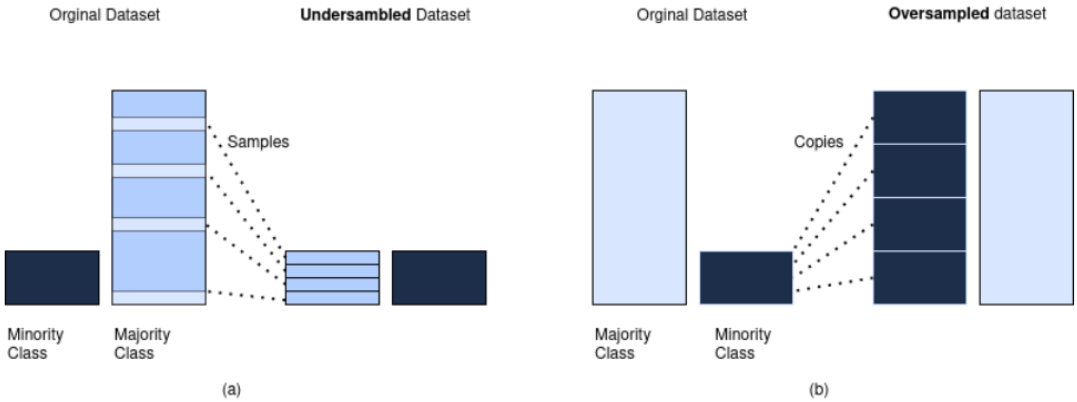


Figure 4.8 An illustration for (a) undersampled dataset and (b) oversampled dataset.

During building the dataset we used a tool called Feature Explorer presented in Figure 4.9 to detect outliers in the dataset and better visualization of how good the samples can be classified. This technique is based on Uniform Manifold Approximation and Projection (UMAP) (McInnes et al., 2018) for data dimensionality reduction. Additional enhancements, based on feature explorer, will be considered in a next dataset version.

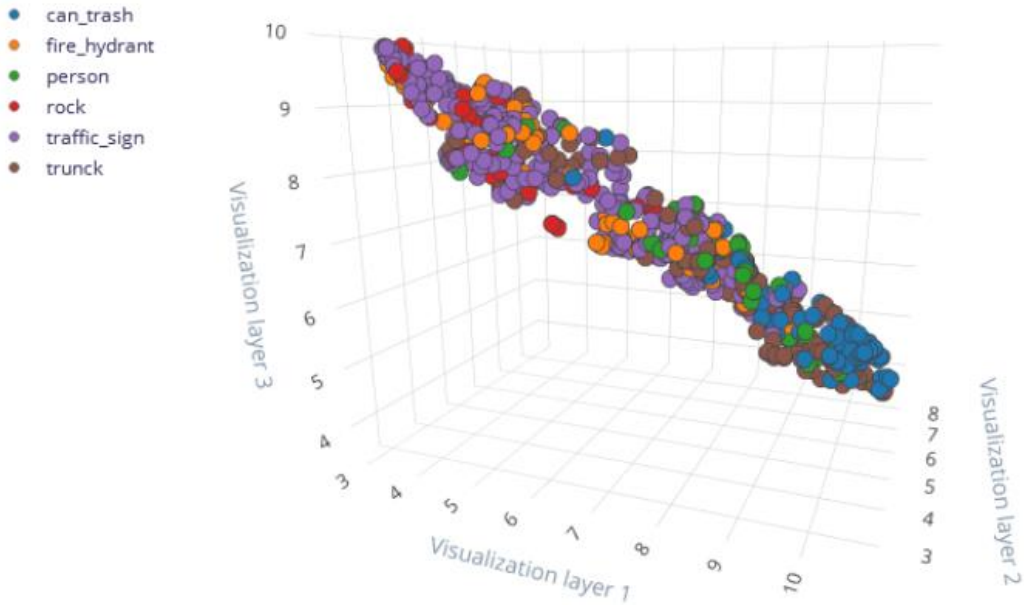


Figure 4.9 Our dataset was visualized using feature points after dimensionality reduction to 3D space. This graph is exported from Edge Impulse using the 'feature explorer' tool.

4.5 The usage of EdgeImpulse platform

Developing a new deep-learning system requires multiple tools related to data collection for dataset creation, data annotation tools, training resources, and monitoring performance. We have chosen a cutting-edge platform called Edge Impulse, which is

increasingly adopted in the research community (Hymel et al., 2022), because it is an end-to-end cloud development platform that provides tools for annotation and training on one side.

Edge Impulse briefly has the features, as shown in Figure 4.10: 1- It provides most tools need for developing new Machine learning-based solutions. The provided tools include rich data-capturing experience with many options, an annotation tool, as shown in Figure 4.11, data visualization, model design with building blocks based on TensorFlow with the ability to modify the source, as shown in Figure 4.12, versioning system and finally training. 2- Edge Impulse is a recent cutting-edge solution to enable embedded machine learning (TinyML) which is an important future research target for us.

We aim to rebuild our system with Hardware with less computation resources to reduce power consumption and costs. Edge Impulse provides multiple deployment options for different embedded system development boards and provides an optimized model. This includes quantized model (int8 and float32 options) and the availability of special compilers like EON Compiler which is claimed to achieve compiling neural networks in 25-55% fewer in data space, and up to 35% less in program space.

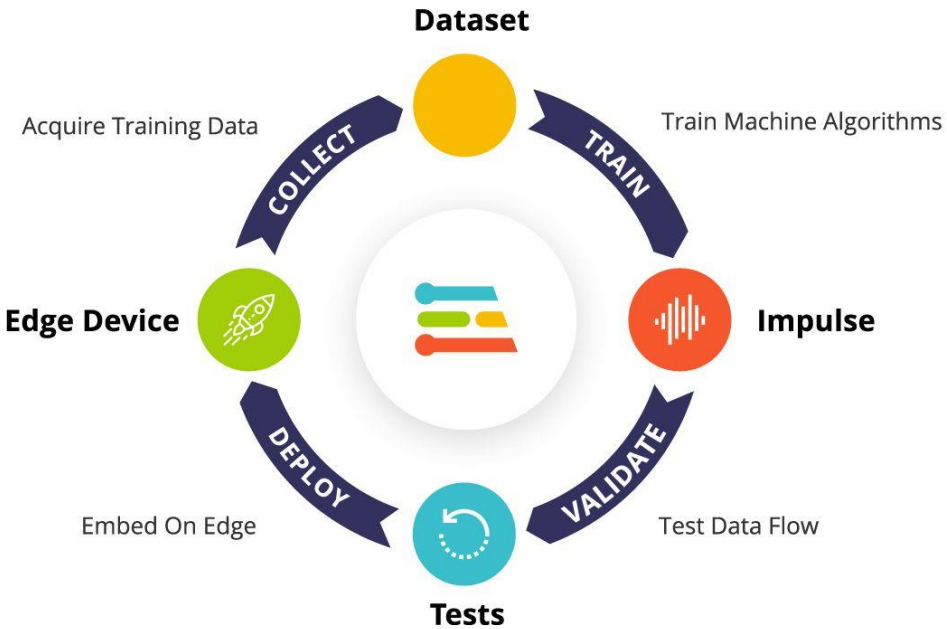


Figure 4.10 A diagram showing the ecosystem that Edgeimpulse platform provides.

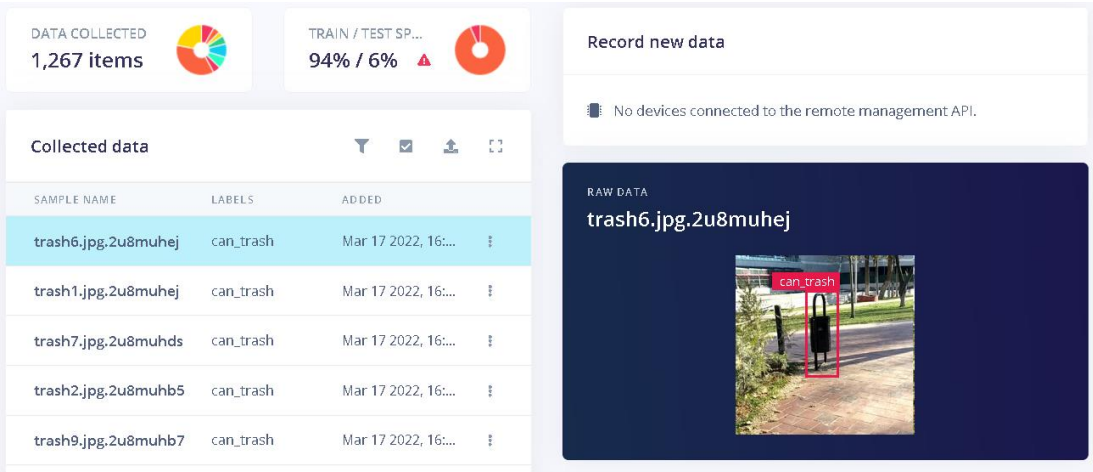


Figure 4.11 A screenshot showing the usage of Edgeimpulse annotation tool during building our WODD dataset.

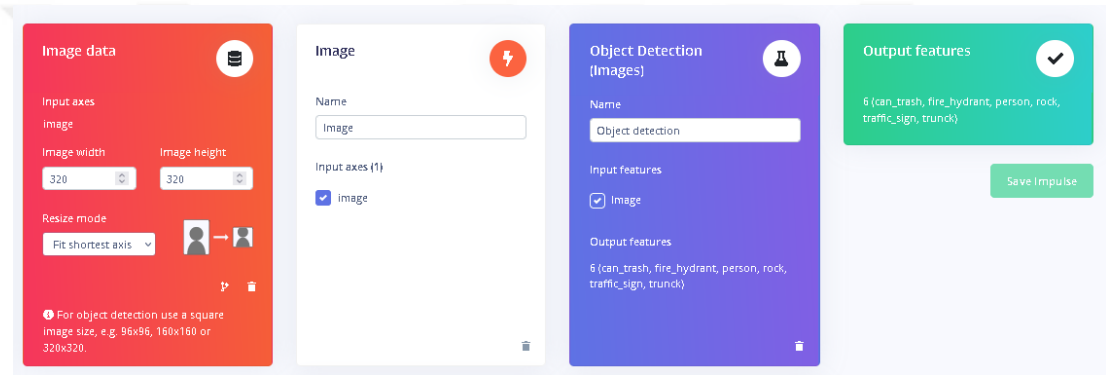


Figure 4.12 A screenshot showing the capability in Edgeimpulse to formulate a deep learning model using building configurable blocks.

We get a self-contained (.eim) package from Edge Impulse with all used processing blocks inside including the TensorFlow model. Our Edge Impulse project for obstacle detection including the annotated dataset can be accessed publicly. Finally, We would like to acknowledge the support from Aurelien Lequertier from EdgeImpulse to let us extend the training time on the platform upon our request during the very early phases of our system development.

4.6 Novel image space wheelchair obstacle avoidance

This section presents the proposed obstacle-avoidance method, which is an image-space visual-avoidance method. The image acquired by the vision sensor is fed as an input to the obstacle-detection module. The output of this module is a bounding box that defines the location of the obstacle in the image. The obstacle coordinates are sent to the avoidance module so it can take the required avoidance action.

The proposed method in this section assumes the following settings. The WC is navigating in an outdoor environment that is expected to contain a few potential obstacles. Before detecting the obstacle and activating the avoidance module, the WC is guided to achieve its main navigation task. The control-design and motion-planning used to define the main task are outside the scope of this work.

To simplify the presentation, we assumed that the WC was following the sidewalk in a straight line through the main task and will resume the main task as soon as the avoidance task is completed. Once an object is detected inside the defined area of interest (AOI), the avoidance motion is initiated to move the WC in such a way that the obstacle falls outside its field of vision and pathway.

The AOI represents a rectangle in front of the WC of 1m width and 2m depth. We consider the obstacle inside the AOI when the bottom edge of its bonding box enters the trapezoid area.

4.6.1 Control law design

The design of the control law that achieves this motion is based on an Image-Based Visual Servo IBVS controller. An error function is defined in the image space and the motion that moves the obstacle outside the image space is generated by regulating the error function to zero. The problem formulation in the image space is illustrated in Figure 4.13.

We selected the coordinates of the centre of the bounding box around the detected obstacle as a feature. The size of the image is $I_c \times I_r$. If the obstacle is in the left half of the image then it will be moved to the left side and will be moved to the right side if it is in the right half of the image. The coordinates of the bounding box around the obstacle are u,v, and are aimed to be moved to the desired values u^*,v^*. As the aim of the avoidance process is to move the obstacle to the right or to the left outside the image, the coordinate v^* is selected to be the same value of v. The coordinate u^* is selected in the middle of the margin depending on whether the obstacle is in the right or left half

$$u^* = \begin{cases} I_c - \frac{m}{2} & \text{: feature point } \in \text{ right half} \\ \frac{m}{2} & \text{: feature point } \in \text{ left half} \end{cases} \tag{4.1}$$

Here, m is the safety margin width in pixels.

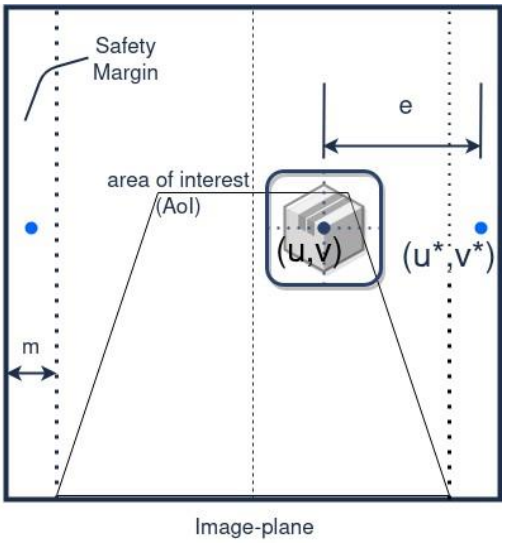


Figure 4.13 An illustration of the problem in the image plane. AOI, safety margin, error function e , position of feature point (u,v), and the position of the desired location of feature point (u∗,v∗) are depicted here.

If multiple obstacles are detected inside the AOI on one side, then we select the one with position closer to image centre (the closest u to $I_c/2$) and to Camera (the largest v, knowing that image position starts from the top-left corner). We call the selected one as the main obstacle. Avoiding the main obstacle will lead to avoiding the others.

We define the task function $e = (s - s^*)$ to be regulated to zero:

$$e = \begin{bmatrix} u & - & u^* \\ v & - & v^* \end{bmatrix} \tag{4.2}$$

where $s^* = (u^*, v^*)$ and $s = (u, v)$ represent the desired and current feature point position respectively in image frame. Let $\dot{r} = (v_y, \omega_z)^T$ represents the velocity to move the WC. It is composed of linear velocity along Y-axis (v_y) and angular velocity around Z-axis (ω_z).

To derive the velocity commands we consider a visual servoing task that uses the image Jacobian matrix, which relates the velocity of image features to the WC velocity in Cartesian frame (Velasco et al., 2019; Chaumette et al., 2006; Hutchinson et al., 2007), knowing that $\dot{e} = \dot{s}$:

$$\dot{s} = J\dot{r} \tag{4.3}$$

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} \frac{\lambda}{Z} & 0 & \frac{-v}{Z} & \frac{-uv}{\lambda} & \frac{\lambda^2+u^2}{\lambda} & -v \\ 0 & \frac{\lambda}{Z} & \frac{-v}{Z} & \frac{-\lambda^2-v^2}{\lambda} & \frac{uv}{\lambda} & u \end{pmatrix} \begin{pmatrix} 0 \\ v_y \\ 0 \\ 0 \\ 0 \\ \omega_z \end{pmatrix} = \begin{pmatrix} -v\omega_z \\ \frac{\lambda v_v}{Z} + u\omega_z \end{pmatrix} \tag{4.4}$$

After a little mathematical development considering an exponential convergence of the task function to zero (Hafez, 2007), we find the velocity components are calculated as

$$\omega_z = \frac{(u-u^*)}{v}, \text{ and } v_y = -\alpha \frac{u(u-u^*)}{v} \frac{Z}{\lambda} \tag{4.5}$$

Where u and v are the image point coordinates, and Z is the depth of the 3D point. Also, λ is the camera focal length in pixels $\lambda = \frac{l}{\rho}$ where l is focal length in mm and ρ is the pixel size in mm, and α is a gain constant. From Raspberry Pi camera V2 specifications, we compute: $\lambda = 3.04mm/0.00112mm = 2714.3$ px. In the ideal case $v - v^* = 0$. Z is considered constant and equal to 1.5 meters.

The above-calculated values of ω_z and v_y ensure the error e is zero. This means the centre of the obstacle is moved to the safety margin in the image because the WC has moved away from the obstacle. This control system guarantees a significant reduction in error through an exponential rate.

CHAPTER 5

EXPERIMENTAL EVALUATION AND ANALYSIS

This chapter presents our experimental setup and our experimental evaluation of the proposed obstacle-detection and -avoidance method. We show the details of the used Hardware, and then how we evaluated the performance of our fine-tuned object detection model. Later on, we presented the results of running the system in a real PWC inside our university campus in multiple experiments. We also discuss how we increased the model execution speed from 0.8 FPS to around 5 FPS.

5.1 The experimental setup and wheelchair model

Our developed system runs on Raspberry Pi 4 with a Raspberry Pi camera v2.1 module. Figure 5.1 shows the hardware setup we used during experiments and the development of our system. We used a Sabertooth driver to control the WC motors and experimented with the WC around our university campus. In the following experiments, we will discuss the performance of obstacle detection and avoidance. The WC system, along with its kinematic and dynamic models, are described in this sub-section.

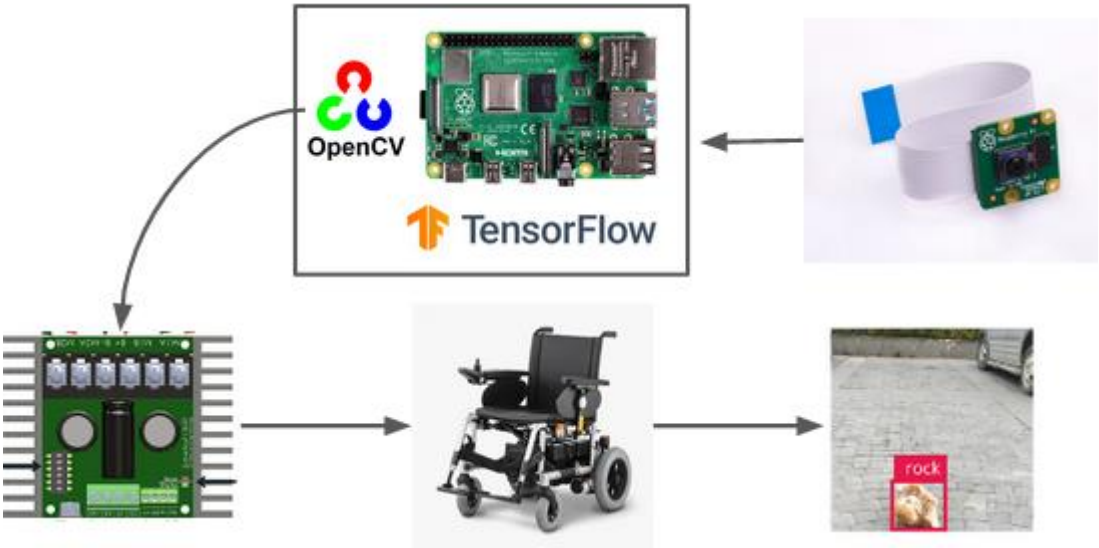


Figure 5.1 The Hardware components of our wheelchair setup

The v2 Camera Module, shown in Figure 5.2, has a Sony IMX219 8-megapixel sensor. Horizontal FoV 62.2 degrees. Vertical FoV 48.8 degrees. The Camera Module can be accessed from picamera python library. Its price is around 33 USD.

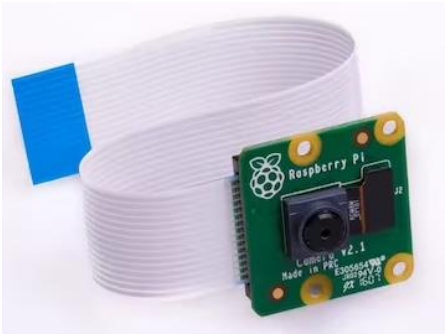


Figure 5.2 Raspberry Pi Camera Module v2

The used Sabertooth 2x32, shown in Figure 5.3, can drive up to two motors at up to 32A continuous and 64A peak each. It can be operated from radio control, analog, TTL serial or USB inputs. The Raspberry Pi controls the Sabertooth using serial command over USB. its price is around 135 USD.

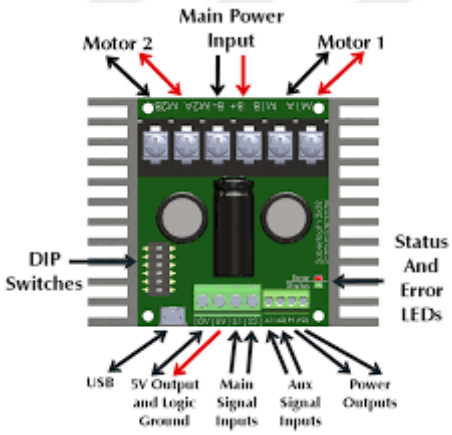
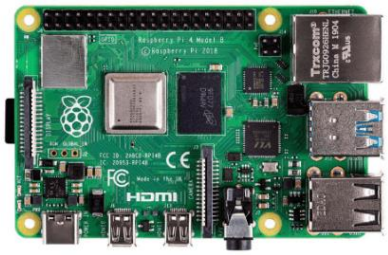
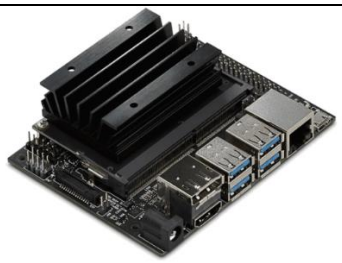


Figure 5.3 Motor driver Sabertooth 2x32

We have used during development Raspberry Pi 4 and Jetson Nano SBCs to compare the performance of the tunned deep-learning model on both of them with different setups as will be explained in the next chapter. In Table 5.1, we made a comparison between both of Raspberry Pi 4 and Jetson Nano in what matters in this work. However, only Raspberry Pi is used during the practical experiments with the PWC.

Table 5.1 A compression between Raspberry Pi 4 and Nvidia Jetson Nano

		
CPU	quad-core 64-bit ARM Cortex-A72 CPU @ 1.5 GHz	quad-core ARM Cortex-A57 64-bit @ 1.43 GHz
GPU	No built-in GPU	NVIDIA Maxwell™ architecture GPU with 128 NVIDIA CUDA® cores
RAM	4GB of RAM (upto 8 GB)	4GB of RAM
Connectıivity	dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0	No built-in Wifi. Requires additional module.

We used TensorFlow framework which is an end-to-end open-source deep learning framework developed by Google. TensorFlow and targets Mobile/Embedded devices like Raspberry Pi with TensorFlow Lite (TFLite). Types of optimization provided in TFLite: quantization, pruning and clustering.

The WC is modeled as a six-wheel robot moving on a horizontal plane. The two large rear wheels are actuated by two DC motors. The other four wheels are passive and used to support the WC. The PWC and the installed Hardware are depicted in Figure 5.4.

Figure 5.4 The PWC used during experiments and development with the installed hardware (battery, Raspberry Pi, motor controller and camera).

The world coordinate frame is defined as F_w and the WC frame (robot frame) is defined as F_r. This frame is attached to the middle of the segment formed by the centres of the differentially actuated wheels. The camera frame is defined as F_c. These frames are similarly defined in (Chaumette et al., 2008).

Following the unicycle robot model shown in Figure 5.5, the velocity of the WC in the global coordinate frame $u = (v,\omega)$ is given as follows (Pasteau et al., 2016).

$$v_x = v\cos\varphi\,, v_y = v\sin\varphi\,, \omega_z = \omega \tag{5.1}$$

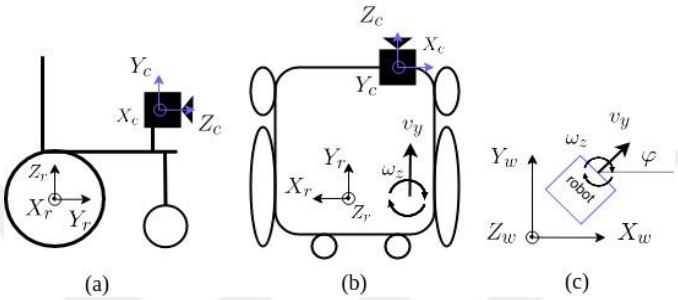


Figure 5.5 WC modeling with camera, robot and world frames (a) Side view of the WC (b) Top view of the WC

In our obstacle-avoidance task, we control the motion of the WC during the avoidance using the linear velocity along Y-axis v_y, and the angular velocity around Z-axis ω_z. These two are calculated finally in the world coordinate frame F_w.

5.2 Evaluating the obstacle-detection model

Object-detection models do predictions with a bounding box and a class label. We use the mean Average Precision (mAP) metric value for evaluation. It is a well-used metric for object-detection problems.

To calculate the mAP, the overlap between the predicted bounding box and the ground-truth bounding box is measured. This is called the Intersection Over Union (IoU). Figure 5.6 demonstrates IoU calculation on a sample image.

A threshold δ_{IoU} of the metric IoU is selected. Any prediction with an IoU measure value larger than or equal to the threshold is considered as True Positive (TP). If the measure is less than the threshold, it is considered as False Positive (FP). For example, if IoU threshold is $\delta_{IoU} = 0.3$, and the value for a IoU of prediction is 0.5, then we classify the prediction as True Positive (TP). But if IoU of prediction is 0.2, we classify it as False

Positive (FP). Average Precision (AP) is finding the area under the precision-recall curve (PRC). Precision P and Recall R are calculated according to:

$$\text{Precision } = \frac{TP}{TP+FP}, \text{ Recall } = \frac{TP}{TP+FN} \tag{5.2}$$

where, True Positive (TP) is when an object is predicted as positive matching to the ground truth. False Negatives (FN) is when an object failed to be predicted, although it is there. Ultimately, mAP is reached by calculating the AP for each class and averaging them.

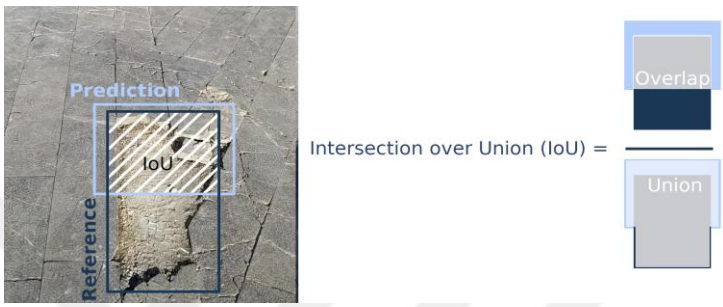


Figure 5.6 A prediction example in the image that depicts the Intersection over Union (IoU) calculation.

In figure 5.7, we show samples from the outputs of the proposed model during the validation process using the validation set of images.

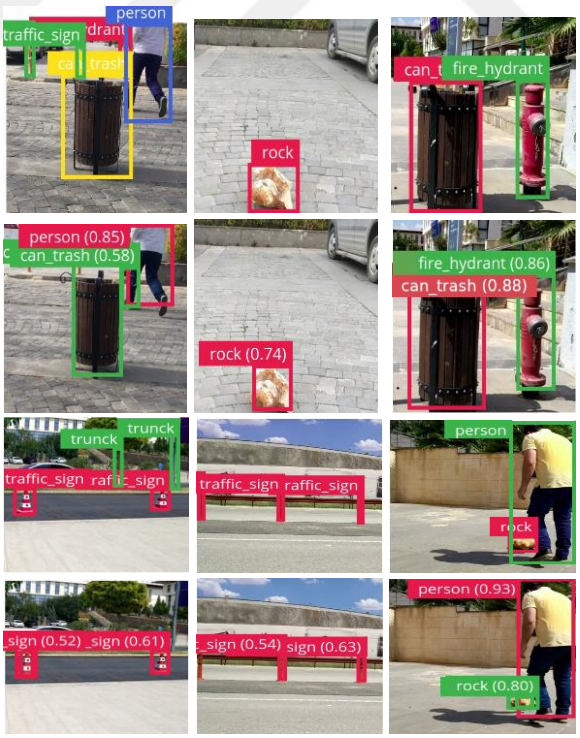


Figure 5.7 Samples of the validation images with bounding boxes as ground truth, and under each one the output from the object-detection

Our experiments have reported 61% mAP accuracy after training the model with a total of 2559 training images available in our dataset. We present in Figure 5.8 the precision-recall graph which shows the trade-off between precision and recall of our fine-tuned model using different thresholds.

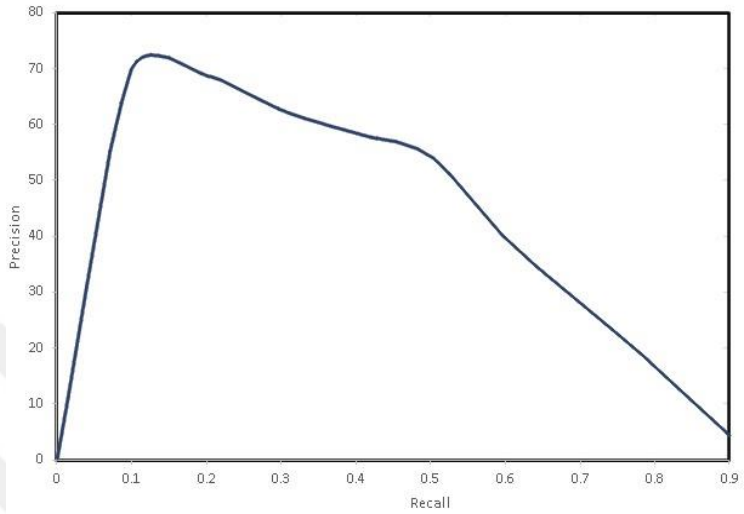


Figure 5.8 Precision-recall curve for our fine-tuned model.

However, a 66% mAP for MobileNetv2 SSD is reported in (Gulati et al., 2008) after training the model with the PASCAL VOC dataset that contains 20 classes. Our model was originally trained using the COCO dataset and then fine-tuned by us using our WODD dataset, which has fewer classes. This gives some insights into our resultant mAP. The development of WODD went through different stages to increase the precision. We had to increase the dataset size, as shown in table 5.2, to achieve better results. This was reflected to the P-R as shown in Figure 5.9.

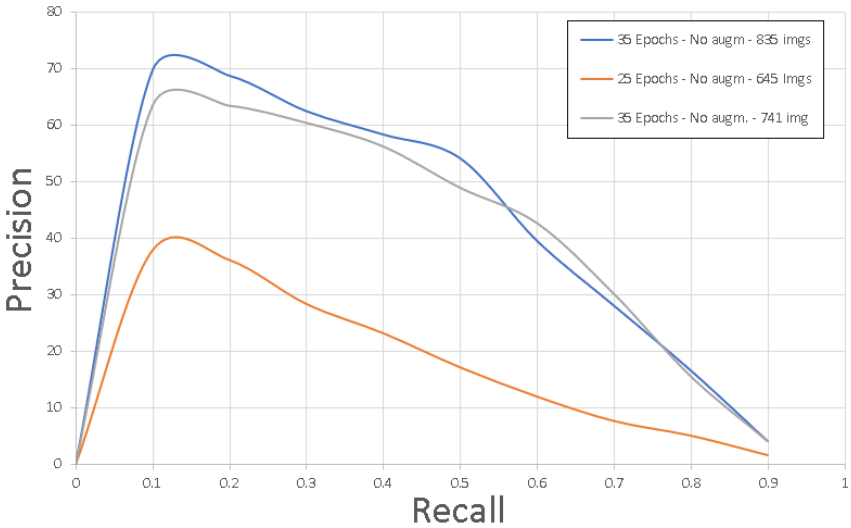


Figure 5.9 P-R curves during the evolution of our dataset (WODD) during fine-tuning experiments.

Table 5.2 Evolution of our dataset (WODD) during fine-tuning experiments.

Trail	Remark	Images#	Epochs	Instances#	mAP
#1	Initiate	645	25	2422	27.4
#2	Increase images number	741	35	2896	30.7
#3	Increase images number	853	35	3061	52.8
#4	Augmentation	853 + 1706	30	3061 (+ augmentation)	60.1

5.3 Experiments on deep-learning model inference speed

Having a usable system execution speed is a crucial target. The avoidance will fail if the Frame Per Second (FPS) processing speed is low. Therefore, we made several experiments to enhance the system speed. Experiments were done using Tensorflow, our adopted platform, Lite model (.tflite) which is designed and optimized for small computers such as smartphones and SBCs like Raspberry Pi and Nvidia Jetson Nano.

The object detection model speed on Jetson Nano using the Tenorflow lite was not as expected to be for a GPU. Our experiments reported ~2.8 FPS, as shown in Figure 5.10. It was found that the best performance can be achieved only when the model is compiled using a tool by Invidia called Tensor-RT. Tensor-RT is a C++ compiler developed by Nvidia to optimize the Deep model blocks for the GPU.

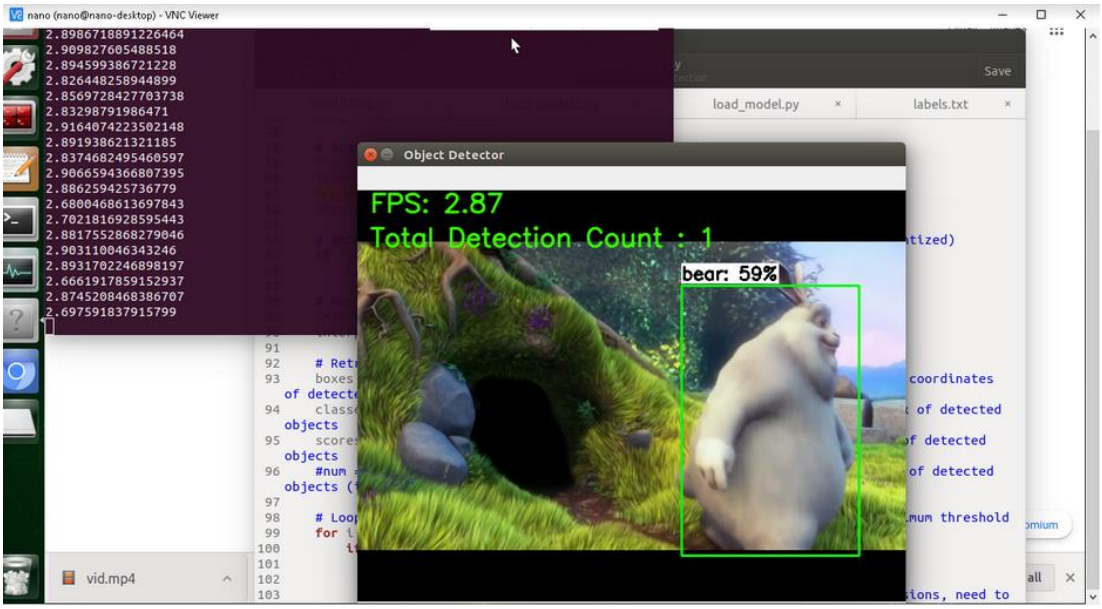


Figure 5.10 Running MobileNet V2 SSD on Jetson Nano at 2.8 FPS speed.

Jetson Nano runs the generated model on the quad-CPU only without the GPU if Tensor-RT was not used. The multi-threading performance while doing object detection on Jetson Nano looks like Figure 5.11.

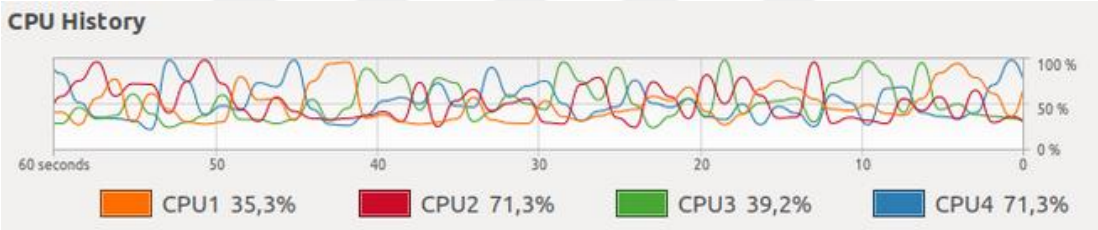


Figure 5.11 CPU load on the 4 cores available in Jetson Nano CPU while running MobileNet V2 SSD.

Applying different setups led us to increase the FPS, as shown in Figure 5.12. We achieved the best FPS speed, as shown in Table 5.3, by using the 64-bit version of Raspberry Pi OS with the Tensorflow Lite.

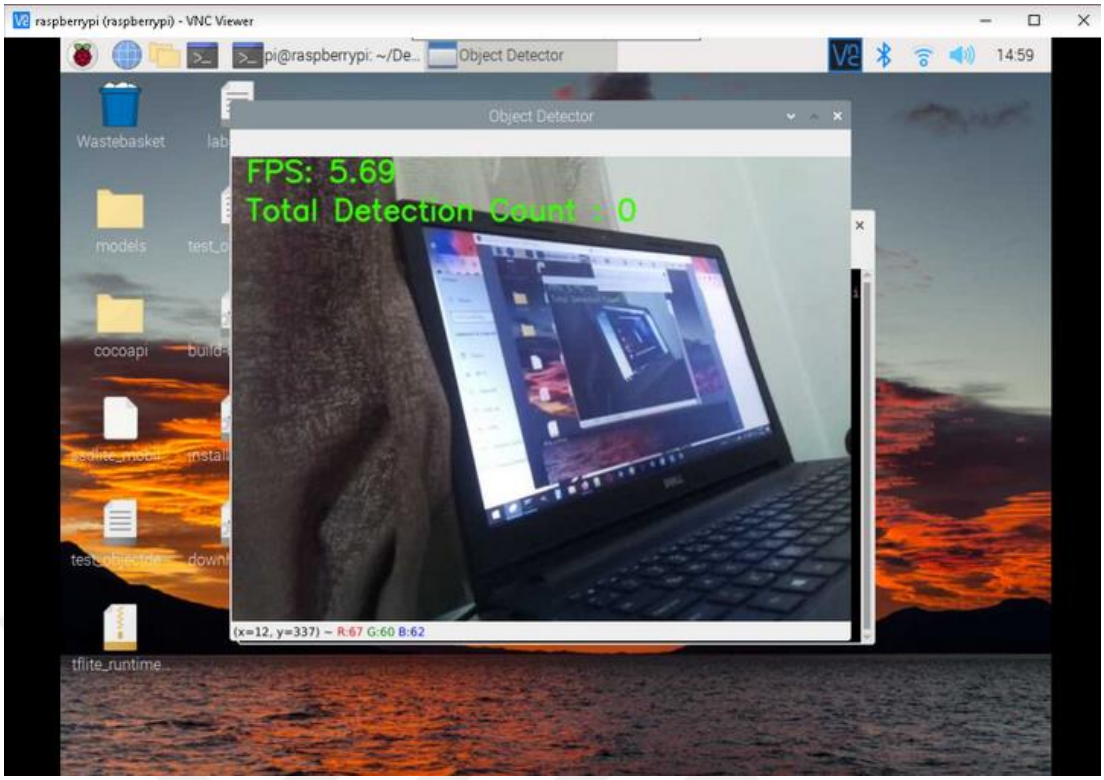


Figure 5.12 Achieving 5.69 FPS while running the Mobilenet V2 SSD using Tensorflow Lite and Raspberry Pi 64-bit OS.

Table 5.3 MobileNet V2 SSD FPS results on Raspberry Pi 4 & Jetson Nano (CPU) under different setups

Hardware	Raspberry Pi 4	Raspberry Pi 4	Jetson Nano / CPU	Jetson Nano/ CPU
OS	Raspberry Pi OS 64-bit	Raspberry Pi OS 32-bit	Nvidia-jetpack 4.6	Nvidia-jetpack 4.6
Model Format	.tflite model	.tflite model	.PB model	.tflite model
tflite Runtime	2.8.0	2.5.0	2.5.0	2.5.0
Python	3.9	3.7	3.6	3.6
FPS	**5.7**	**1.8**	**0.5**	**2.96**

To get a better execution speed in Raspberry Pi, we applied different settings: i) CPU clock: We increased the Raspberry Pi CPU clock (overclocking) to 900 MHz. Raspberry Pi OS default CPU clock is 700 MHz, while the maximum speed is 1.5GHz; ii) Execution priority: By increasing python interpreter process priority from the OS, we

increased the speed too; iii) Power source: To get the full execution speed, Raspberry Pi needs to be powered from a DC power source that can provide 5V and up to 3A. We noticed the increase in execution speed when we compared running the same code with the official 3A DC adapter and with a 2A power bank. However, our experiments required 20-watt power bank usage to provide enough power to Raspberry Pi.

Our computential system can work with any portable 15─20 watt DC power source or ideally regulate the 5V from the WC's main battery, using a step-down converter. However, the motor's controller needs to be connected to the main WC battery.

An example of CPU and memory usage for our system on the Rasbperry Pi 4 is shown in Figure 5.13. On average, the CPU usage is 46% while operating in detection and avoidance mode. 36% out of the 46% is for running the detection deep-learning model and the rest 10% is for running the avoidance code including image processing, logging and camera view display for debugging purposes. On the memory side, the system consumes around 150 MB out of 3649 MB of RAM. 50 MB of them is for the detection part and 100 MB is for the avoidance and other debugging tools included in the code.

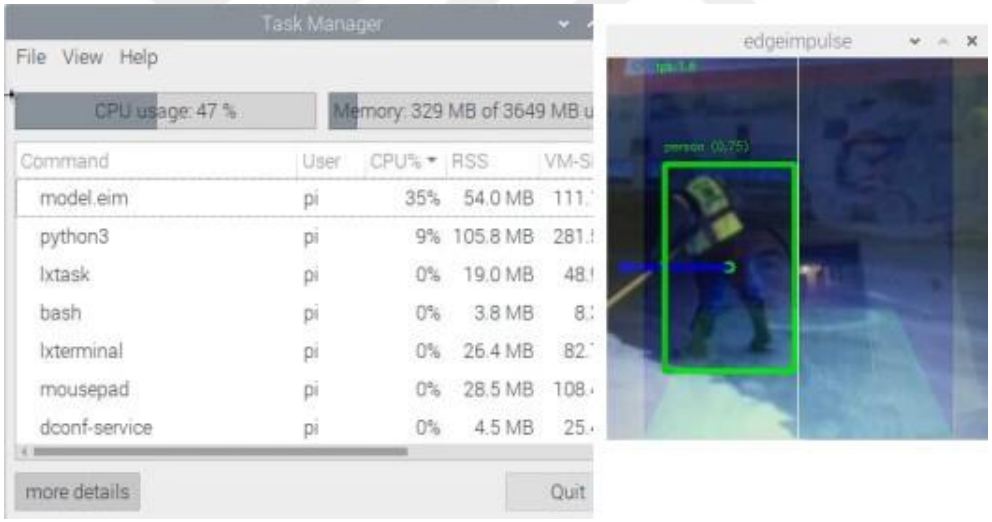
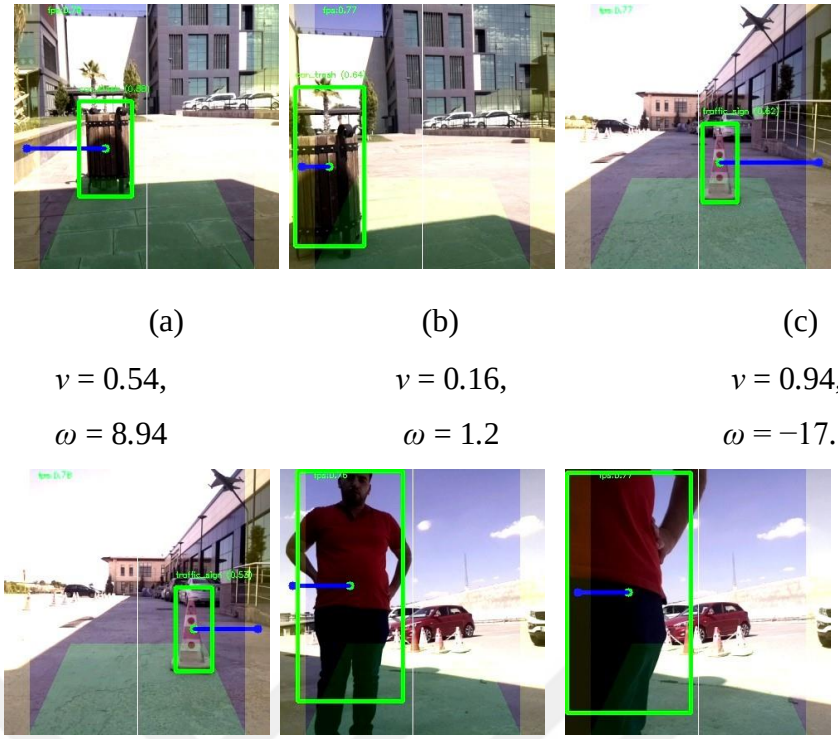


Figure 5.13 CPU and memory usage on Rasbperry Pi 4 while both dectection and avoidance sub-modules are operating.

5.4 Experiments on detection and avoidance using the real wheelchair system

The object-detection and avoidance system was tested inside our university campus using our WC system built in our lab for this purpose. Multiple experiments with different obstacles in different scenarios were conducted. Figure 5.14 shows a few samples from them.



(a) $v = 0.54$, $\omega = 8.94$

(b) $v = 0.16$, $\omega = 1.2$

(c) $v = 0.94$, $\omega = -17.2$

(d) $v = 0.65$, $\omega = -13.6$

(e) $v = 0.49$, $\omega = 6.18$

(f) $v = 0.41$, $\omega = 4.68$

Figure 5.14 Different scenes show the detected obstacles (green bounding box) inside AOI (green trapezoid area) and how to control law is bringing the feature point (green dot) to the safety margin based on the distance to it (blue line). Under each image the action is generated using the velocities calculated in the control law.

We fine-tuned the WC forward speed by trial and error to correspond to the achieved FPS. We did the experiments with different speeds in the range of 0.1 to 0.5 meters per second. The best performance was using 0.4 m/s which falls in the range of the wheelchair mobility speed according to (Pan et al., 2019; Ding et al., 2004; Lund, 2010).

In images (a) and (b) from figure 5.14, the WC detects the trash can inside the AOI, the control law computed the positive velocity to push the obstacle feature point to the safety margin by moving the WC to the right. However, the required velocity in (a) is higher than in (b) as the obstacle is closer to the safety margin in (b). In images (c) and (d), the traffic-sign obstacle is on the right side of the image, so the WC rotation is to the left, the angular velocity is negative. Again, the velocity in (c) is higher than in (d) because the distance of the object image to the margin is shorter. Similar observations can be made for images (e) and (f).

An estimated trajectory path of the WC for another experiment is shown in Figure 5.15. It represents the path of the WC during the experiment and the location of the obstacles relative to the WC. The path clearly reflects the ability of the WC to avoid the obstacle. The images at the beginning, the end, and at intermediate points of the path show the scene through the WC's eye (internal view).

The WC is initially moving straight with a speed corresponding to the main task requirement. We assume that the main task requirement is to move the WC in a straight line with constant velocity. Once the obstacle (traffic sign) enters the AOI from the right side of the images, the control law calculates the required velocity vector to push the obstacle, represented by the center point of the bounding box, to the safety margin in the image.

The calculated linear and angular velocities per frame are shown in Figure 5.15. According to the sub-figure (a) in Figure 5.15, the avoidance started after around two frames (around one second) once the obstacle was detected inside the AOI (scene (c) in Figure 5.15). It took around 14 frames (around seven seconds) to get the obstacle out to the safety margin (scene (d) in Figure 5.15). Once the object is outside the AOI, the velocity profile returns to the main task trajectory. The reader can watch the video at[2] for recordings from Raspberry Pi Camera made during the experiments.

[2] https://youtu.be/MQc9g3VEyzM

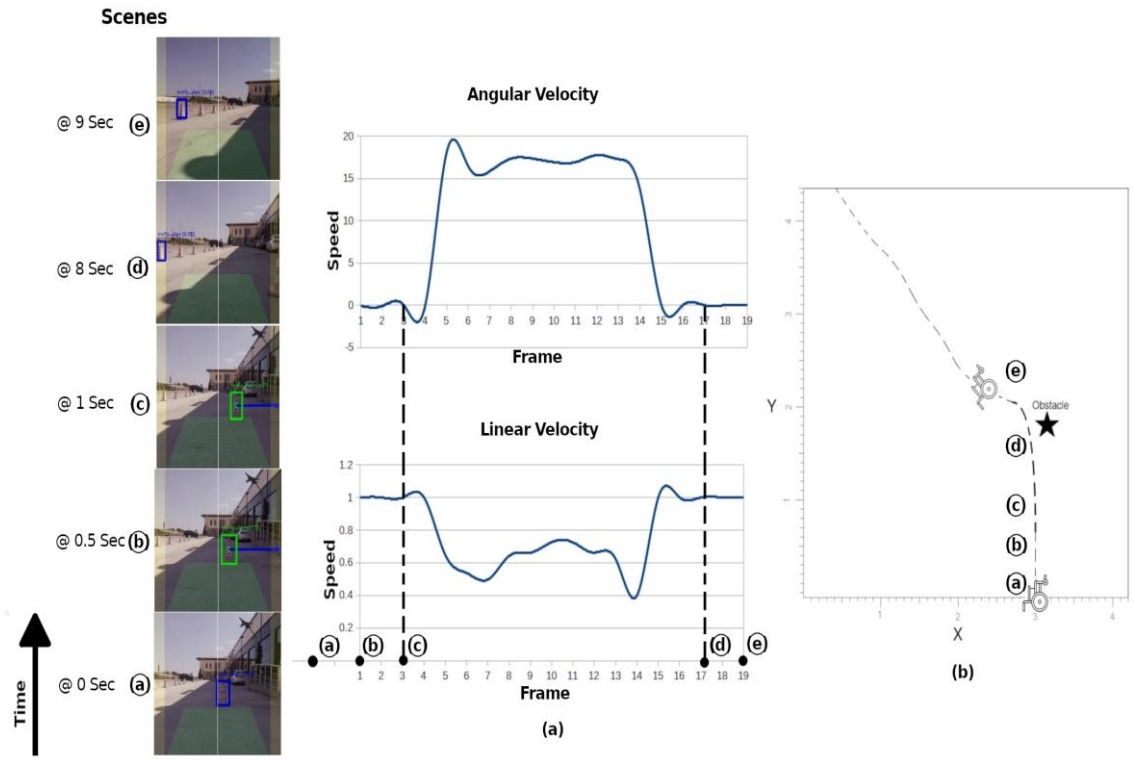


Figure 5.15 (a) The calculated linear and angular speed by the control law changing per frame whilst avoiding the obstacle. (b) Our WC estimated trajectory path while avoiding the obstacle.

CHAPTER 6

CONCLUSION

WC users may need to move forward and backward, and obstacles may appear in both cases. The avoidance can be done in the same proposed vision-only manner in both cases. Therefore, our system is expandable as cameras can be installed to the front and rear of the wheelchair. This requires a higher-level mechanism to select the wheelchair movement direction and corresponding camera. On other hand, avoidance can be formulated as a task working with a front or back camera view. Therefore, the side of the wheelchair seems to be less important in monitoring our problem. However, a higher supervisor mechanism that assures the coordination of the different modules of the smart wheelchair is beyond the scope of this article, which focuses on the detection and avoidance module only. Ideally, the control law in one direction will be capable of avoiding obstacles without the need to switch between the front and back cameras.

We recognize that multiple obstacles may appear in front of the WC. Our proposed system is capable of detecting the defined obstacles in the image. Once multiple obstacles enter the area of interest, avoidance of one will result in avoidance of the others. We refer to the “main obstacle”. This is the one with a position closest to the camera and closest to the middle of the image centre. However, we assume multiple obstacles to be on one side of the image (left or right). Yet in real life, this may not be so. This is for consideration in future work.

Usually, a navigation task steers the WC to its target and navigation needs to include path correction after the completion of the obstacle-avoidance task. Yet our work focuses on adding a new avoidance system to an autonomous wheelchair, so we assumed movement in a straight line in an outdoor environment without route planning. We are looking forward to integrating other modules in a single system, aiming to create a multi-functional autonomous WC. Such a system would be able to navigate in a real-life environment and position itself in respect of that environment, having the ability to avoid obstacles with just a single camera.

Another real scenario that may happen in a surrounding environment includes more obstacles than the nine defined in WODD. For example, we did not include the benches in WODD, as the original deep-learning model was trained using the COCO

dataset, and COCO already has benches. We wanted to include new classes that are not found in COCO. Namely, traffic sign, truck, pothole, trash can, truck, fence and rock. Other obstacles may be added to our publicly accessible dataset as part of future work.

Our system, combining as it does robust working with affordability, presents a solution to the “cheap or robust” dilemma. It will allow PWC users to upgrade their WCs by installing our system. Simultaneously, this should bring the WC industry's attention for applying deep learning and computer vision to their future models with an obstacle-detection and -avoidance functionality.

Our system results in terms of:

1) detection and avoidance speed at 5 FPS running on Raspberry Pi with single camera;
2) achieving a good obstacle-detection precision;
3) total estimated cost of $250;

is promising and addresses one of the key challenges users face during outdoor mobility. Ultimately, it is a step toward building a cost-efficient and fully-autonomous WC system that can be adopted by the market.

REFERENCES

Akash, S., Menon, A., Gupta, A., Wakeel, M., Praveen, M. & Meena, P. (2014) "A novel strategy for controlling the movement of a smart wheelchair using internet of things". *IEEE Global Humanitarian Technology Conference-South Asia Satellite (GHTC-SAS)*, pp. 154−158.

Ahmed, F. & Yeasin, M. (2017) "Optimization and evaluation of deep architectures for ambient awareness on a sidewalk", in 2017 *International Joint Conference on Neural Networks (IJCNN),* pp. 2692–2697.

Ali, S., Al Mamun, S., Fukuda, H., Lam, A., Kobayashi, Y., Kuno, Y. (2018) "Smart robotic wheelchair for bus boarding using CNN combined with Hough transforms", in *International Conference on Intelligent Computing* 2018, pp. 163–172.

Burhanpurkar, M., Labb´e, M., Guan, C., Michaud, F. &, Kelly, J. (2017) "Cheap or robust? the practical realization of self-driving wheelchair technology". In 2017 *International Conference on Rehabilitation Robotics (ICORR)*. pp. 1079–1086.

Chen, W., Liou, A., Chen, S., Chung, C., Chen, Y. & Shih, Y. (2007) A novel home appliance control system for people with disabilities. *Disability And Rehabilitation: Assistive Technology*, 2, pp. 201−206.

Corke, P, Hutchinson, S. A (2001) new partitioned approach to image-based visual servo control. *IEEE Transactions on Robotics and Automation*, 17(4):507–515.

Chaumette, F. & Hutchinson, S. (2006) "Visual servo control. I. Basic approaches", *IEEE Robotics Automation Magazine*, 13, pp. 82−90.

Chaumette, F. & Hutchinson, S. Visual servoing and visual tracking. (*Springer*,2008)

Pasteau, F, Narayanan, V, Babel, M, Chaumette, F. (2016) "A visual servoing approach for autonomous corridor following and doorway passing in a wheelchair", *Robotics and Autonomous Systems*, 75 pp.28–40.

Corke PI, Khatib O. Robotics, vision and control: fundamental algorithms in MATLAB. Berlin: Springer; 2011 Nov 3.

Choudhary T, Mishra V, Goswami A, Sarangapani J. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*. 2020 Oct;53(7):5113-55.

Day, C., McEachen, L., Khan, A., Sharma, S. & Masala, G.(2019) "Pedestrian recognition and obstacle avoidance for autonomous vehicles using raspberry pi", *Proceedings Of SAI Intelligent Systems Conference*, pp. 51-69.

Desai, S., Mantha, S., Phalle, V. (2017) "Advances in smart wheelchair technology", *International Conference on Nascent Technologies in Engineering (ICNTE)*, pp. 1–7.

Diao, C, Jia, S, Zhang, G, Sun, Y, Zhang, X, Xue, Y, Li, X. Design and realization of a novel obstacle avoidance algorithm for intelligent wheelchair bed using ultrasonic sensor". In 2017 *Chinese Automation Congress (CAC)* 2017 (pp. 4153–4158).

Ding, D., Cooper, R., Guo, S. & Corfman, T. (2004)"Analysis of driving backward in an electric powered wheelchair", *IEEE Transactions On Control Systems Technology*. 12, pp. 934_943.

Dorbala, V., Hafez, A., Jawahar., C.A. (2019). "Deep Learning Approach for Robust Corridor Following", in 2019 *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3712–3718.

Eid, M., Giakoumidis, N. & El Saddik, A. (2016) "A novel eye-gaze-controlled wheelchair system for navigating unknown environments: case study with a person with ALS", *IEEE Access*, 4, pp. 558−573.

Elliott, M., Malvar, H., Maassel, L., Campbell, J., Kulkarni, H., Spiridonova, I., Sophy, N., Beavers, J., Paradiso, A., Needham, C. & Others. (2019) "Eye-controlled, power wheelchair performs well for ALS patients", *Muscle Nerve*, 60, pp. 513−519.

Erickson, B., Hosseini, M., Mudhar, P., Soleimani, M., Aboonabi, A., Arzanpour, S. & Sparrey, C. (2016) "The dynamics of electric powered wheelchair sideways tips and falls: experimental and computational analysis of impact forces and injury. *Journal Of Neuroengineering and Rehabilitation*, 13, pp. 1−10.

Frank, A., Neophytou, C., Frank, J. & Souza, L. (2010) "Electric-powered indoor/outdoor wheelchairs (EPIOCs): users' views of influence on family, friends and carers". *Disability And Rehabilitation: Assistive Technology",* (5), pp 327−338.

Fereidouni, S., Sheikh Hassani, M., Talebi, A. & Rezaie, A. (2020) A novel design and implementation of wheelchair navigation system using Leap Motion sensor. *Disability And Rehabilitation: Assistive Technology*, pp. 1−7.

Gulati, S. &, Kuipers, B. (2008) High performance control for graceful motion of an intelligent wheelchair. In 2008 *IEEE International Conference on Robotics and Automation*, pp. 3932–3938.

Haddad, M. &Sanders, D. (2020) "Deep learning architecture to assist with steering a powered wheelchair", *IEEE Transactions on Neural Systems and Rehabilitation Engineering*; 28(12) pp. 2987–2994.

Hafez, A. & Jawahar, C. "Visual servoing by optimization of a 2D/3D hybrid objective function" (2007) Proceedings 2007 *IEEE International Conference On Robotics And Automation*. pp. 1691−1696.

Hutchinson, S. & Chaumette, F. (2007) "Visual servo control, part ii: Advanced approaches. *IEEE Robotics And Automation Magazine*, 14, pp. 109-118.

Hymel S, Banbury C, Situnayake D, Elium A, Ward C, Kelcey M, Baaijens M, Majchrzycki M, Plunkett J, Tischler D, Grande A. Edge Impulse: An MLOps Platform for Tiny Machine Learning. *arXiv preprint arXiv*:2212.03332. 2022 Nov 2.

He, Y., Yang, H.B., Wang, S.J. (2019) "CDBV: A Driving Dataset With Chinese Characteristics From a Bike View. *IEEE Access* ,7, pp.51714–51723.

Joshi, K., Ranjan, R., Sravya, E. & Baig, M. (2019) "Design of voice-controlled smart wheelchair for physically challenged persons", *Emerging Technologies in Data Mining and Information Security*, pp. 79−91.

Kadmin, A., Jidin, A., Bakar, A., Aziz, K. & Abd Rashid, W. (2016) "Wireless voice-based wheelchair controller system", *Journal Of Telecommunication, Electronic and Computer Engineering (JTEC)*, 8, pp. 117−122.

Kim, E. (2016) "Wheelchair navigation system for disabled and elderly people", *Sensors*, 16(11) p.1806.

Leaman, J, La, H. A comprehensive review of smart wheelchairs: past, present, and future. (2017) *IEEE Transactions on Human-Machine Systems*, 47(4), pp.486–499.

Li, Z., Xiong, Y. & Zhou, L. (2017) ROS-based indoor autonomous exploration and navigation wheelchair. In 2017 10th *International Symposium on Computational Intelligence and Design (ISCID)* 2017, pp. 132–135.

Lee, Y. & Kim, Y.(2020) "Comparison of CNN and YOLO for Object Detection", *Journal Of Semiconductor Display Technology*. 19, 85−92.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. & Berg, A.(2016) "Ssd: Single shot multibox detector", in *European conference on computer vision* 2016, pp. 21–37.

Lund, M., Christiensen, H., Caltenco, H., Lontis, E., Bentsen, B. & Struijk, L. (2010) "Inductive tongue control of powered wheelchairs". 2010 *Annual International Conference Of The IEEE Engineering In Medicine And Biology*, pp. 3361−3364.

Lee, Y., Lim, J., Eu, K., Goh, Y. & Tew, Y. (2017) "Real time image processing based obstacle avoidance and navigation system for autonomous wheelchair application", in 2017 *Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 380–385.

Maule, L., Zanetti, M., Luchetti, A., Tomasin, P., Dallapiccola, M., Covre, N., Guandalini, G. & De Cecco, M. (2022) "Wheelchair Driving Strategies: a comparison between standard joystick and gaze-based control", *Assistive Technology*, pp. 1−13.

Maule, L., Luchetti, A., Zanetti, M., Tomasin, P., Pertile, M., Tavernini, M., Guandalini, G. & De Cecco, M. (2021) "RoboEye, an Efficient, Reliable and Safe Semi-Autonomous Gaze Driven Wheelchair for Domestic Use", *Technologies*. 9(16).

McInnes, L., Healy, J. and Melville, J., (2018). "Umap: Uniform manifold approximation and projection for dimension reduction.", *arXiv preprint arXiv:1802.03426*.

Neuhold, G., Ollmann, T., Rota Bulo, S. & Kontschieder, P. (2017) "The mapillary vistas dataset for semantic understanding of street scenes. In Proceedings of *the IEEE international conference on computer vision* 2017, pp. 4990–4999.

Organization, W. World health statistics 2015. (World Health Organization,2015)

Edwards, K. & McCluskey, A. (2010) "A survey of adult power wheelchair and scooter users. Disability And Rehabilitation", *Assistive Technology*, (5), pp 411-419.

Ozkors, M. A., Say, M., Yazici, A., Yayan, U, & Ak, M. (2014) “Kinect based Intelligent Wheelchair navigation with potential fields”, in 2014 *IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA)* Proceedings, pp. 330–337.

O’Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G., Krpalkova, L., Riordan, D., Walsh, J. (2019) “Deep learning vs. traditional computer vision”, in *Science and Information Conference* 2019, pp. 128–144.

Oksuz, K., Cam, B., Kalkan, S. & Akbas, E. (2020)“Imbalance problems in object detection: A review”, *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 43, pp. 3388−3415.

Park, K., Oh, Y., Ham, S., Joo, K., Kim, H., Kum, H. & Kweon. (2020) ”SideGuide: A Largescale Sidewalk Dataset for Guiding Impaired People”, in 2020 IEEE/RSJ *International Conference on Intelligent Robots and Systems (IROS)*, pp. 10022–10029.

Pan, J., Sun, H., Song, Z, & Han, J. (2019) “Dual-Resolution Dual-Path Convolutional Neural Networks for Fast Object Detection”, *Sensors*, 19(14) p. 3111.

Priyanayana, S., Buddhika, A. & Jayasekara, P. (2018) Developing a voice controlled wheelchair with enhanced safety through multimodal approach. *IEEE Region 10 Humanitarian Technology Conference* (R10-HTC). pp. 1−6.

Petry, M., Moreira, A., Braga, R. &, Reis, L. (2010) “Shared control for obstacle avoidance in intelligent wheelchairs”, In *IEEE Conference on Robotics, Automation and Mechatronics* 2010 (pp. 182–187).

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. & Chen, L.C. (2018) “Mobilenetv2: Inverted residuals and linear bottlenecks”, in Proceedings of the *IEEE conference on computer vision and pattern recognition* 2018, pp. 4510–4520.

Sun, C., Su, J., Ren, W. &, Guan, Y. “Wide-View Sidewalk Dataset Based Pedestrian Safety Application”. *IEEE Access* 2019 (7), pp.151399–151408.

Shahinfar, S., Meek, P. & Falzon, G. (2020) How many images do I need? Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring. *Ecological Informatics*, 57, pp. 101085

Sidik, M., Ghani, S., Ishak, M., Harun, W., Daud, N. (2018) “A review on electric wheelchair innovation to ease mobility and as a rehabilitation tool for spinal cord impairment patient”, *Internet journal on engineering and technology,* 3(10), pp.803–815.

Soma, S., Patil, N., Salva, F. & Jadhav, V. (2018) “An Approach to Develop a Smart and Intelligent Wheelchair”. *9th International Conference On Computing, Communication And Networking Technologies (ICCCNT)*, pp. 1−7.

Shore, S. & Juillerat, S. (2012) “The impact of a low cost wheelchair on the quality of life of the disabled in the developing world”. *Medical Science Monitor: International Medical Journal Of Experimental And Clinical Research*. 18, (CR533).

Simpson, R. (2005) “Smart wheelchairs: A literature review”, *Journal of rehabilitation research and development,* 42(4), p.423.

Tawil Y, Hafez AA. Deep Learning Obstacle Detection and Avoidance for Powered Wheelchair. In 2022 *Innovations in Intelligent Systems and Applications Conference (ASYU)* 2022 Sep 7 (pp. 1-6). IEEE.

Torkia, C., Reid, D., Korner-Bitensky, N., Kairy, D., Rushton, P., Demers, L., Archambault, P. (2015) “Power wheelchair driving challenges in the community: a users’ perspective”, *Disability and Rehabilitation Assistive Technology*, 10(3), pp. 211–215.

Tello, A, Hafez, A, Sarakbi, B. ”Real-time GP-based Wheelchair Corridor Following”. In 2021 29th *Signal Processing and Communications Applications Conference (SIU)* 2021 (pp. 1–4).

Vidana-Zavala, D. & Munno, M. (2019) “Design of a Wheelchair for Low-Income Countries, the Second Stage of a Project. Advances In Design For Inclusion”, Proceedings Of The AHFE 2019 International Conference On Design For Inclusion And The AHFE *2019 International Conference On Human Factors For Apparel And Textile Engineering*, July 24-28, 2019, Washington DC, USA. 954, p. 301.

Velasco, J., Pascion, C., Alberio, J., Apuang, J., Cruz, J., Gomez, M., Molina Jr, B., Tuala, L., Thio-ac, A. & Jorda, R. Jr (2019) “A smartphone-based skin disease classification using mobilenet cnn”, *ArXiv Preprint ArXi,*1911.07929.

Whill.inc. Autonomous Solutions. (2022). Available from: https://whill.inc/gb/autonomous-solutions/