Department of Mechatronic Engineering

MXEN4000 - Mechatronic Engineering Research Project 1

Progress Report

Navigation Assistance for a Semi-Autonomous
Smart Wheelchair

Semester 1, 2022

Jakob Wyatt
19477143

Supervisor(s): Yifei Ren & Siavash Khaksar

Abstract

Table of Contents

List of Figures

List of Tables

1 Introduction

Many people with motor disabilities rely on wheelchairs for movement, and powered wheelchairs have enabled greater independance for people with disability. Despite the huge benefit powered wheelchairs have granted, the use of this technology can be inaccessible or unsafe for people with amyotrophic lateral sclerosis (ALS) or vision impairment, who may be unable to use a joystick or see their environment clearly.

1.1 Aims

The aim of this research is to develop a semi-autonomous smart wheelchair system. This research is done in collaboration with Glide, a WA wheelchair manufacturer, who have provided an existing powered wheelchair (CentroGlide) to use as a base for this functionality (fig. 1). By developing assistive technology for the wheelchair, the user is granted greater mobility, confidence, and independence.

Figure 1: CentroGlide in Reclined Configuration

1.2 Problem Definition

There are multiple engineering research project students who are part of this team, working on elements such as controller design, navigation assistance, and object detection. This work specifically focuses on pathway assistance, which identifies suitable paths for the wheelchair to drive on. If a user unintentionally drives off their desired path, this can lead to uneven terrain and possibly falling from the wheelchair. By guiding the user along a path, these safety issues can be mitigated.

Emphasis is placed on the ’semi-autonomous’ aspect of the wheelchair. An important requirement of this project is that the user still has control over their wheelchair, and can override any autonomous functionality if required. If the smart wheelchair system mistakenly detects an obstacle, the users mobility should not be compromised.

Another requirement of the system is that any sensors mounted to the wheelchair should not impede the users comfort or the wheelchairs manouverability. Many wheelchair users have specific requirements for wheelchair seat adjustments, to avoid pressure sores and discomfort. Figure 1 shows the wheelchair configuration when fully reclined, demonstrating that some sensor mounting locations are infeasible.

The smart wheelchair system should also be commercially viable - high-cost components and sensors are infeasible. Internet connectivity should not be a requirement for the system to operate either - the round trip time required to communicate with a server would compromise the safety of a user. Because of this, all processing is performed locally on the wheelchair.

Name	Type	Cost (AUD)[1]	Dimensions (mm)	FOV (Horizontal, Vertical, Depth)	Operating Range (m)
Stereolabs Zed Mini [8]	Passive	$595	$124.5 \times 30.5 \times 26.5$	$90° \times 60° \times 100°$	0.1-15
Stereolabs Zed 2 [9]	Passive	$670	$175 \times 30 \times 33$	$110° \times 70° \times 120°$	0.3-20
Intel RealSense D455 [10]	Active IR (Stereo)	$595	$124 \times 26 \times 29$	$90° \times 65° \times 87°$	0.6-6
Microsoft Azure Kinect DK [11]	Active IR (Time of Flight)[2]	$595	$103 \times 39 \times 126$	$75° \times 65° \times 75°$	0.5-3.86

Table 2: Stereo Camera Options

2 Literature Review

2.1 Sensors and Hardware

2.1.1 Sensor Types

Smart wheelchairs have used a varied range of sensor types to percieve the surrounding environment. RGB-D stereo cameras have been widely used in the field [1][2][3], alongside 2d Lidar [4] and ultrasonic sensors [5]. Self-driving cars built by companies such as Tesla and Waymo use cameras, mmWave Radar, and 3d Lidar to avoid traffic and pedestrians [6].

Selecting a sensor to use is not necessarily an either-or decision. Sensor fusion algorithms such as the Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF) [7] allow outputs from multiple sensors to be used together to improve their accuracy. Additionally, sensors can be used for different applications on the smart wheelchair - a stereo camera could be used to sense the surrounding environment while an inertial measurement unit (IMU) could be used for wheelchair odometry. Table 1 gives a comparison of several available sensor types, taking into account factors such as resolution, cost, and accuracy.

Sensor	Advantages	Disadvantages
RGB-D Stereo Camera	Very high resolution	Low field of view (FOV)
mmWave Radar	High accuracy	Low resolution
3D Lidar	High resolution and accuracy	Very high cost
2D Lidar	High FOV and accuracy	Only detects obstacles within the same plane
Ultrasonic sensor	Low cost	One-dimensional

Table 1: Sensor Comparisons

2.1.2 RGB-D Cameras

One advantage RGB-D cameras have over alternative sensors is high RGB resolution, allowing them to utilize advances in machine learning and computer vision. Sensors such as LIDAR may fail at path detection, as path markings cannot be detected.

When comparing these cameras, factors such as package size, field of view, and depth accuracy are important to consider due to the available mounting points on the wheelchair. Several commercial options are compared in Table 2 - all of the listed units come with an integrated IMU.

[1]Costs are taken at RRP with an exchange rate of 1 AUD = 0.74 USD

[2]The Microsoft Azure Kinect DK has multiple operating modes which tradeoff between FOV, operating range, and resolution. The `NFOV unbinned` mode was compared, which provides good tradeoff between operating range and resolution.

A caveat of the Stereolabs products is that they require a seperate CUDA enabled GPU (manufactured by Nvidia) to generate the point-cloud and RGB-D image. In contrast, the Kinect DK only requires a CPU for processing, while the Intel RealSense performs processing onboard and requires a USB-C 3.1 interface to communicate.

2.1.3 Compute Element

A compute element inside a semi-autonomous driving system generally consists of several components - a microcontroller to process user inputs and send signals to the motors, a general purpose computer to run pathfinding algorithms and log information, and an AI accelerator to improve performance of on-board machine learning (ML) algorithms.

AI accelerators use specialized hardware to perform operations common in ML algorithms (such as matrix multiplication and convolution) more efficiently than a CPU can. GPUs have been used widely for this application, however their high power usage is infeasible for some applications. Embedded AI accelerators aim to provide greater power efficiency at the cost of specialization.

Machine learning models often use mixed-precision (FP16) datatypes to store weights while training. Although improving the accuracy of the model, FP16 datatypes are slow to manipulate during inference. Model quantization [12] is a process where this FP16 datatype is replaced with an INT8 datatype (using a scaling factor and bias) after training. This gives a large speed improvement, while only losing a small amount of model accuracy. For this reason, modern AI accelerators focus on the performance of INT8 operations (TOPS, 10^9 operations per second), whereas earlier accelerators state the performance of FP16 operations (TFLOPS, 10^9 floating-point operations per second).

The Nvidia Jetson and Google Coral products are both popular options for embedded AI acceleration. These accelerators are compared in Table 3 alongside a gaming GPU.

Name	Cost (AUD)[1]	Release Year	Speed	Power	Notes
Nvidia Jetson Nano [13]	$150	Early 2019	0.5 TFLOPS (FP16)	10 W	-
Nvidia Jetson Xavier NX [14]	$595	Late 2019	21 TOPS (INT8)	20 W	-
Nvidia RTX 2080 [15]	$1040	2018	80.5 TFLOPS (FP16) 161.1 TOPS (INT8)	215 W	Doesn't include single-board computer
Google Coral Edge TPU [16]	$190	2020	4 TOPS (INT8)	2 W	Only supports TensorFlow Lite

Table 3: AI Accelerator Options

2.2 Scene Understanding

Scene understanding is a broad field, and involves using computer vision methods on visual or spatial data to gain better knowledge about the surrounding environment. Convolutional Neural Networks (CNNs) are commonly used for this application, as they are able to exploit the local nature of image features to reduce the number of required computations.

2.2.1 Image Classification

Image classification is a core problem within this field, and involves identifying the subject of an image (such as an animal or object). AlexNet [17], based on the earlier digit-recognition CNN LeNet-5 [18], was one of the first deep CNNs applied to this problem. AlexNet was trained on the large ImageNet dataset [19], which consists of 15M images and 22K categories, and achieved an error of only 15.3% on a 1000 class subset. The underlying architecture uses a series of 5 convolutional layers and 3 fully connected layers, which can be seen in fig. 2.

Neural network architectures have become deeper and more accurate over time, enabled by both growth in computational power and dataset size. VGG-16 [20] and GoogLeNet [21] are 16 and 22 layers deep respectively, and approached human performance on the ImageNet dataset. ResNet [22] is up to 156 layers deep, and exceeds human performance at image classification with an error of 3.57%. ResNet uses a ’skipping’ architecture to improve network training, where the output of a layer relies directly on the input of a previous layer.

2.2.2 Object Localization

Object localization is another core problem within this field, and involves identifying the location of objects within an image as well as classifying them. Object localization can be used on a semi-autonomous wheelchair to identify a pedestrian or obstacle within the environment. R-CNN [24] was one of the first object classification models which utilized convolutional networks, by identifying potential bounding boxes and running an image

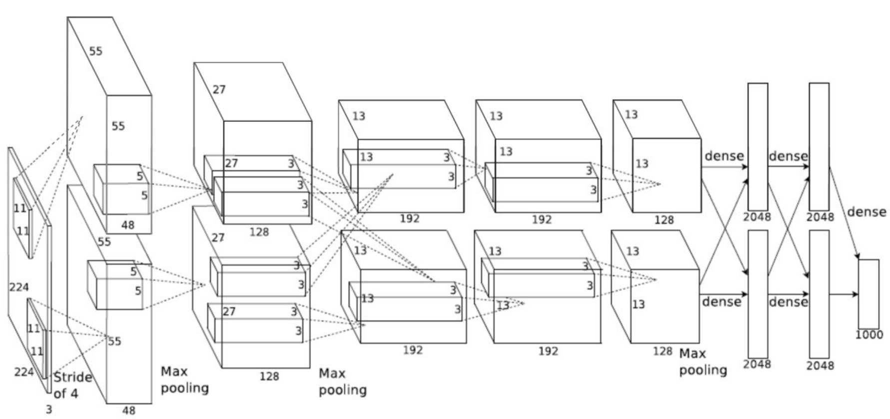

Figure 2: Architecture of the AlexNet image classification network. Reproduced from Krizhevsky et al. [17])

classifier on these bounding boxes. Fast and Faster R-CNN [25][26] improved the speed of this model by running an image classifier backbone once on the entire image, and using a CNN to improve identification of bounding boxes. Pascal VOC [27] and MS COCO [23] are datasets which are commonly used to evaluate object classification models.

YOLO (You Only Look Once) [28][29][30][31] is another object classification model which focuses on improving the speed of the model. In particular, YOLOv4 [31] reaches over 60 fps on the Tesla V100, which enables its use in real time applications such as autonomous driving and security camera footage. YOLO divides an image into an $S \times S$ grid, and uses a single convolutional network to output both bounding box predictions and image classification for each grid square. Low-probability and overlapping bounding boxes are then removed before the final output.

2.2.3 Semantic Segmentation

Semantic segmentation involves labelling each pixel of an object, rather than drawing a bounding box around the entire object. This technique is often used in medical applications, where different components of a scan need to be labelled. Another application semantic segmentation can be used for is drivable area detection, as a bounding box would not be able to cleanly identify a road or kerb. Figure 3 compares the output of semantic segmentation to image classification and object localization.

Most semantic segmentation algorithms use an encoder-decoder architecture, where information about the image is encoded into a small feature space. This feature space is then decoded back to the size of the original image using deconvolutional layers to obtain the segmented output. An issue with this architecture is that the resulting image can be low quality, as the image encoding is low resolution. U-net [32] is a semantic segmentation network which helps to rectify this issue, by using higher resolution features during deconvolution. This makes the segmented output sharper and more accurate.

Another semantic segmentation algorithm is DeepLab [33][34][35]. DeepLab uses atrous convolution (otherwise known as dilated convolution), which is a type of convolution which widens the FOV of a convolutional layer. It does this by leaving gaps in the

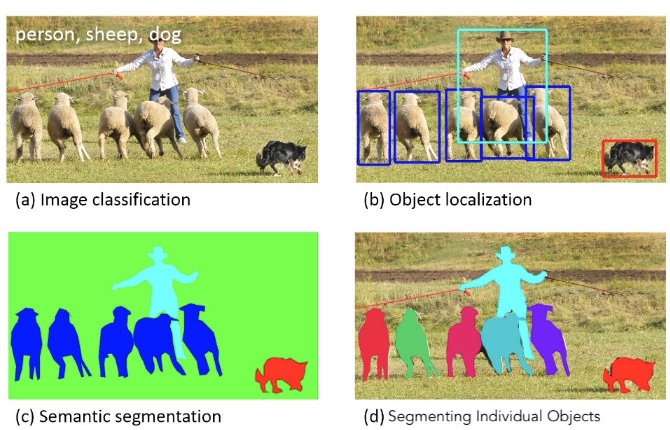

Figure 3: Types of Classification in Machine Vision. Reproduced from Lin et al. [23])

convolutional layer, as illustrated in fig. 4. By widening the FOV of each convolution, less downscaling is required during encoding. By encoding the image in a much higher resolution feature space, the segmented output is more accurate. To obtain the segmented output, a technique called atrous spatial pyramid pooling (ASPP) is used. ASPP samples the feature space at different scales using atrous convolution to classify each pixel in the image. These techniques improve both the accuracy and speed of the network - DeepLabv3 obtained 86.9% accuracy on the PASCAL VOC 2012 test set.

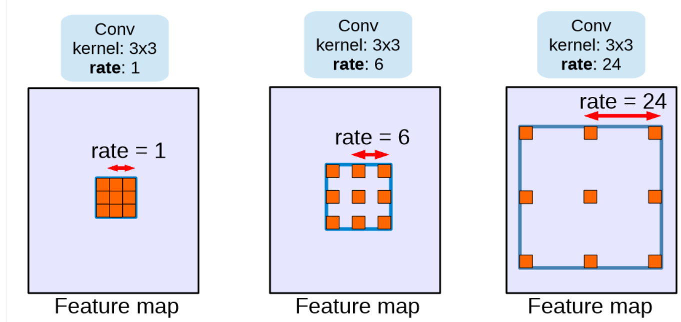

Figure 4: Atrous convolution with a 3x3 kernel, showing increasing FOV. Reproduced from Chen et al. [35])

2.3 Assistive Control

Once an understanding of the 3d scene has been built, the user can be navigated through the environment. The surrounding environment is generally represented as an occupancy grid [36], which is a top-down view of the area where each grid cell indicates the probability that it is occupied by an obstacle. It is possible to include more detailed information about paths and obstacles by adding more information to the occupancy grid.

In semi-autonomous control, the user decides the desired speed and direction of the wheelchair, with any intervention only occuring before a collision takes place. This is in contrast to full autonomy, where the user specifies the desired end goal and the wheelchair navigates to that goal [1].

2.3.1 Path-Based Algorithms

Path-based algorithms take an occupancy grid as an input, and output a path between the start point and a goal point. A* is an example of this, and uses a heuristic to efficiently find the shortest path between the start and end point. Other algorithms such as RRT* (rapidly-exploring random tree) [37] build a tree from randomly sampled points to find a path to the goal node. RRT* may not find the shortest path initially, however can find an efficient path with much less computation required.

One potential issue with these two algorithms is that they fail to take into account the smoothness of the resulting path. Although the path may be short, sharp changes in the trajectory could be uncomfortable to the user. Trajectory planning algorithms aim to solve this - one such algorithm is optimal-control in a Frenét-Frame [38], which can be used in autonomous vehicle control. This algorithm takes a reference path as an input, and outputs a local path which avoids collisions and minimizes jerk (rate of change of acceleration). This is done by generating sample trajectories (represented with quintic polynomials), removing those which cause collisions, and choosing the remaining trajectory with the lowest change in acceleration. Figure 5 illustrates the reference path, obstacles, and generated local path in an example scenario. It should be noted that this algorithm still requires a reference path, which could be generated with one of the path finding algorithms mentioned above.

2.3.2 Local Algorithms

Local algorithms only consider obstacles currently in the proximity of the wheelchair, and use this information to set the current speed and direction of the wheelchair. VFH+ (vector field histogram) [40] is one example, which has been applied to wheelchair control algorithms in prior work [41]. VFH+ calculates a polar obstacle density histogram around the robot based on the occupancy grid. The histogram is then binarized, to classify sectors around the robot as either occupied or not occupied. Next, a masked polar histogram is generated, which excludes paths that are not possible given the robots turning radius and kinematics. Finally, a safe direction is chosen which is nearest to the users desired direction. An example binary histogram is shown in fig. 6; the chosen direction avoids the obstacle in the desired direction.

An advantage to this algorithm is that it gives the user more fine-grained control over their speed and direction, rather than planning a path to their assumed goal. However an issue with VFH+ is that it does not control the wheelchairs speed, and instead only finds a safe direction. Ideally the wheelchair should slow down if an obstacle is present.

Another approach to assistive control with local algorithms is haptic feedback. Rather than blending inputs from the autonomous software and the user, the joystick itself is actuated to make it more difficult to move in the direction of obstacles [43][44]. An advantage of this approach is that it gives the user total control over which direction of movement they choose, however the additional force required to actuate the joystick may fatigue the user.

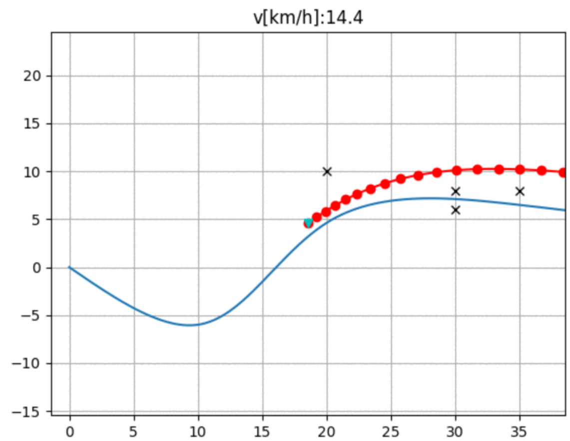

Figure 5: Frenét-Frame path planning, with reference path and local path illustrated. Reproduced from Sakai et al. [39])

3 Methodology

3.1 Hardware

The smart wheelchair should have the ability to sense, process, and manouver within the surrounding environment. To do this requires some necessary hardware, including a sensor system, compute element, and motor controller. Due to the 2021-2022 chip shortage, hardware selection was identified as a process that should occur relatively quickly.

The literature review provides a comparison between different sensor types and models. For outdoor navigation assistance, a forward facing RGB-D camera was selected as the best option for this project - specifically, the Zed 2 Mini. This was chosen due to the high resolution provided by RGB-D cameras, and the long distance range of the Zed cameras due to their passive operation (does not use an IR emitter).

The front of the joystick control unit was selected as the best mounting point for the stereo camera, due to several reasons:

1. Clear view of the environment in front of the wheelchair.
2. Not obstructed by the user in any wheelchair configuration.
3. When needed, the user can move the joystick control unit out of the way, which also moves the camera out of the way.

However, there are some challenges faced when using this mounting point, which must be addressed.

1. Shaky video footage due to low rigidity in joystick mount.
2. Close to the front of the wheelchair, which reduces visibility of the sides of the wheelchair.
3. Maximum camera width of 150 mm before doorway manouverability is affected.

Due to the size constraints, the Mini form factor was chosen.

An additional sensor with a wider field of view will be needed for doorway navigation

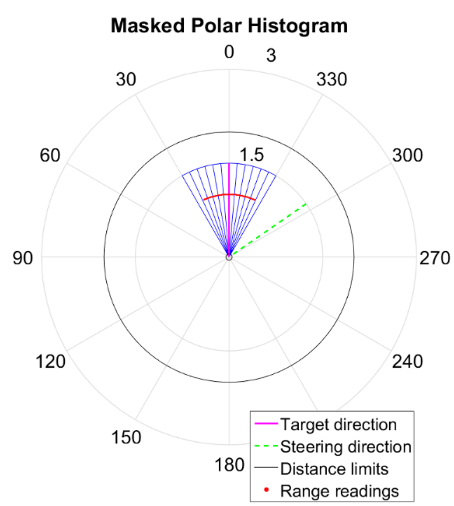

Figure 6: VFH+ binary histogram, representing direction of obstacles. Reproduced from MathWorks [42])

and wheelchair docking, due to the limitations of this mounting point. The sensor selected for this purpose was a 2D Lidar unit attached to the base of the wheelchair. However, this sensor will not be used for our specific application (pathway assistance).

For the compute element, a Nvidia Jetson Xavier NX will be used, due to compatability with the Zed API and a wide range of ML frameworks, as well as low power use. The motor controller is being designed by a different thesis student, however will connect with the compute element in a standard way.

3.2 Dataset Collection

To train and evaluate machine learning models, a 34 minute driving dataset was collected around Curtin University. The camera used to collect this dataset was a GoPro Hero 4. Although this camera does lack a depth component, having a basic dataset for model evaluation is valuable prior to procurement of the RGB-D camera.

3.3 Software

Our software system requires several components. Interaction with the Zed 2 Mini will be done via the proprietry Stereolabs API, which will provide RGB-D imaging to the computer. PyTorch [45] will be used for training and evaluation of machine learning models, as it provides good compatability with existing models such as YOLOv5 [46] and DeepLabv3 [35]. OpenCV [47] will be used to encode and decode video.

4 Current Work

Part of the current work this semester has included selection of the sensor and compute element, as well as dataset collection.

5 Future Work

6 References

[1] H. Wang, Y. Sun, R. Fan, and M. Liu, "S2P2: Self-Supervised Goal-Directed Path Planning Using RGB-D Data for Robotic Wheelchairs," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11 422–11 428, 2021. DOI: 10.1109/ICRA48506.2021.9561314. [Online]. Available: https://sites.google.com/view/s2p2.

[2] H. Wang, Y. Sun, and M. Liu, "Self-Supervised Drivable Area and Road Anomaly Segmentation Using RGB-D Data For Robotic Wheelchairs," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4386–4393, 2019, ISSN: 2377-3766. DOI: doi.org/10.1109/LRA.2019.2932874.

[3] S. Jain and B. Argall, "Automated perception of safe docking locations with alignment information for assistive wheelchairs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 4997–5002. DOI: 10.1109/IROS.2014.6943272.

[4] M. Scudellari, "Self-driving wheelchairs debut in hospitals and airports [News]," *IEEE Spectrum*, vol. 54, no. 10, pp. 14–14, 2017, ISSN: 0018-9235. DOI: doi.org/10.1109/MSPEC.2017.8048827.

[5] S. Levine, D. Bell, L. Jaros, R. Simpson, Y. Koren, and J. Borenstein, "The NavChair Assistive Wheelchair Navigation System," *IEEE Transactions on Rehabilitation Engineering*, vol. 7, no. 4, pp. 443–451, 1999, ISSN: 1063-6528. DOI: 10.1109/86.808948.

[6] A. Karpathy, director, *Tesla AI Day*, 2021. [Online]. Available: youtube.com/watch?v=j0z4FweCy4M.

[7] E. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, IEEE, 2000, pp. 153–158. DOI: 10.1109/ASSPCC.2000.882463.

[8] Stereolabs, *ZED Mini Camera and SDK Overview*, 2018. [Online]. Available: https://cdn.stereolabs.com/assets/datasheets/zed-mini-camera-datasheet.pdf.

[9] Stereolabs, *ZED 2 Camera and SDK Overview*, 2019. [Online]. Available: https://cdn.stereolabs.com/assets/datasheets/zed2-camera-datasheet.pdf.

[10] Intel, *Intel RealSense Product Family D400 Series Datasheet*, 2022. [Online]. Available: https://www.intelrealsense.com/wp-content/uploads/2022/03/Intel-RealSense-D400-Series-Datasheet-March-2022.pdf.

[11] Microsoft. "Azure Kinect DK Hardware Specifications." (2021), [Online]. Available: https://docs.microsoft.com/en-us/azure/Kinect-dk/hardware-specification.

[12] B. Jacob, S. Kligys, B. Chen, *et al.*, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," 2017. DOI: 10.48550/ARXIV.1712.05877. [Online]. Available: https://arxiv.org/abs/1712.05877.

[13] Nvidia, *Jetson Nano System-on-Module Data Sheet*, 2019. [Online]. Available: https://developer.nvidia.com/embedded/downloads.

[14] Nvidia, *Jetson Xavier NX Series System-on-Module Data Sheet*, 2019. [Online]. Available: https://developer.nvidia.com/embedded/downloads.

[15] Nvidia, *Turing GPU Architecture*, 2018. [Online]. Available: https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf.

[16] Google Coral, *Coral Dev Board Datasheet*, 2020. [Online]. Available: https://coral.ai/static/files/Coral-Dev-Board-datasheet.pdf.

[17] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, pp. 1097–1105, 2012, ISSN: 0001-0782. DOI: 10.1145/3065386.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, ISSN: 0018-9219. DOI: 10.1109/5.726791.

[19] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[20] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014.

[21] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going Deeper with Convolutions," 2014.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2016, IEEE, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[23] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft COCO: Common Objects in Context," 2014. DOI: doi.org/10.48550/arXiv.1405.0312.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2013.

[25] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision*, vol. 2015, IEEE, 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[26] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2015.

[27] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2009, ISSN: 0920-5691. DOI: 10.1007/s11263-009-0275-4.

[28] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015. DOI: doi.org/10.48550/arXiv.1506.02640.

[29] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” 2016. DOI: doi.org/10.48550/arXiv.1612.08242.

[30] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” 2018. DOI: doi.org/10.48550/arXiv.1804.02767.

[31] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” 2020. DOI: doi.org/10.48550/arXiv.2004.10934.

[32] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” 2015. DOI: 10.48550/ARXIV.1505.04597. [Online]. Available: https://arxiv.org/abs/1505.04597.

[33] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs,” 2014. DOI: 10.48550/ARXIV.1412.7062. [Online]. Available: https://arxiv.org/abs/1412.7062.

[34] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs,” 2016. DOI: 10.48550/ARXIV.1606.00915. [Online]. Available: https://arxiv.org/abs/1606.00915.

[35] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking Atrous Convolution for Semantic Image Segmentation,” 2017. DOI: 10.48550/ARXIV.1706.05587. [Online]. Available: https://arxiv.org/abs/1706.05587.

[36] A. Elfes, “Using Occupancy Grids for Mobile Robot Perception and Navigation,” *Computer (Long Beach, Calif.)*, vol. 22, no. 6, pp. 46–57, 1989, ISSN: 0018-9162. DOI: 10.1109/2.30720.

[37] S. Karaman and E. Frazzoli, “Sampling-based Algorithms for Optimal Motion Planning,” *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011, ISSN: 0278-3649. DOI: 10.1177/0278364911406761.

[38] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, “Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenét Frame,” *IEEE International Conference on Robotics and Automation*, pp. 987–993, 2010, ISSN: 1050-4729. DOI: 10.1109/ROBOT.2010.5509799.

[39] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, “PythonRobotics: A Python code collection of robotics algorithms,” 2018. DOI: 10.48550/ARXIV.1808.10703. [Online]. Available: https://arxiv.org/abs/1808.10703.

[40] I. Ulrich and J. Borenstein, “VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots,” *IEEE International Conference on Robotics and Automation*, vol. 2, 1572–1577 vol.2, 1998, ISSN: 1050-4729. DOI: 10.1109/ROBOT.1998.677362.

[41] M. R. M. Tomari, Y. Kobayashi, and Y. Kuno, "Enhancing Wheelchair's Control Operation of a Severe Impairment User," in *The 8th International Conference on Robotic, Vision, Signal Processing & Power Applications*, H. A. Mat Sakim and M. T. Mustaffa, Eds., Singapore: Springer Singapore, 2014, pp. 65–72, ISBN: 978-981-4585-42-2. DOI: 10.1007/978-981-4585-42-2.

[42] MathWorks. "Vector Field Histogram." (2022), [Online]. Available: https://au.mathworks.com/help/nav/ug/vector-field-histograms.html.

[43] Y. Kondo, T. Miyoshi, K. Terashima, and H. Kitagawa, "Navigation Guidance Control Using Haptic Feedback for Obstacle Avoidance of Omni-directional Wheelchair," in *2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, IEEE, 2008, pp. 437–444. DOI: 10.1109/HAPTICS.2008.4479990.

[44] E. B. Vander Poorten, E. Demeester, E. Reekmans, J. Philips, A. Huntemann, and J. De Schutter, "Powered wheelchair navigation assistance through kinematically correct environmental haptic feedback," *IEEE International Conference on Robotics and Automation*, pp. 3706–3712, 2012, ISSN: 1050-4729. DOI: doi.org/10.1109/ICRA.2012.6225349.

[45] A. Paszke, S. Gross, F. Massa, *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8024–8035, 2019.

[46] Ultralytics, *YOLOv5*. [Online]. Available: https://github.com/ultralytics/yolov5.

[47] G. Bradski, *The OpenCV Library*, 2000. [Online]. Available: https://opencv.org/.