Journal of Systems Architecture 149 (2024) 103105

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

An intelligent assistive driving solution based on smartphone for power wheelchair mobility

Zhiwei Wang [a,b,*], Jingye Xu [c], Jianqiu Zhang [a], Rocky Slavin [c], Dakai Zhu [c]

[a] Department of Electrical and Computer Engineering, The University of Texas at San Antonio, USA
[b] The Research Computing Support Group of The University Technology Solutions, The University of Texas at San Antonio, USA
[c] Department of Computer Science, The University of Texas at San Antonio, USA

ARTICLE INFO

Keywords:
Wheelchair driving
Deep learning
Smartphone

ABSTRACT

This research introduces a cost-effective, smartphone-powered, computer-vision-based system for Power Wheelchair Intelligent Assistive Driving (PWC IA-Driving). This system enables the safe, hands-free operation of a Power Wheelchair (PWC) with reduced attention required from the user in indoor environments, thereby easing the burden on disabled individuals and lessening their stress. Our objective is to provide an affordable and practical solution that can be easily incorporated into existing PWCs. The system leverages a customized and pre-trained ResNet-based model on a smartphone to derive driving commands from real-time imagery captured by the phone's camera. These instructions are then conveyed to the PWC via a control interface connected to the smartphone. We have developed a prototype of this assistive driving system on a Pixel-6 Android phone and tested its feasibility on a mobile robot as a proof-of-concept. Our evaluations indicate that the smartphone can process up to 25 images per second. This rapid processing rate allows for the generation of driving instructions in real time, enabling the mobile robot to navigate safely at reasonable speeds within the test environment, while requiring minimal user intervention.

1. Introduction

The US Census Bureau reported in 2013 that around 3.6 million individuals over the age of 15 used wheelchairs to assist with mobility in day-to-day tasks.[1] Power wheelchairs (PWCs) have been widely utilized to improve the independence of people with disabilities. In order to alleviate the operation difficulties, PWCs usually provide convenient joystick-type interfaces that can be easily operated with hands. Recently, new technologies have enabled a variety of ***hands-free*** controls, including using a headrest attached to a wheelchair, using breath by inhaling (sip) and exhaling (puff), or using face and mouth movement to operate a PWC.

However, for people with severe cognitive, motion, or sensory issues, it is still a difficult task to operate PWCs to their fullest extent. In particular, 17% of PWC users have reported severe pain, where over 50% of them attributed the pain to the operations of PWCs [1]. Furthermore, many PWC users complain about fatigue, insecurities, and general inconvenience while navigating through (mostly indoor) public spaces for an extended period. A fully automated PWC with a navigation system would offer an ideal solution. Unfortunately, such navigation systems are still in the design and development phases and are thus not accessible to socially disadvantaged demographics.[2] Furthermore, they often require specific types of wheelchairs to accommodate the navigation software and related hardware components to ensure compatibility for a wheelchair to navigate through spaces safely.

With the advancement of computer-vision technology, there have been many research studies focused on hands-free interfaces for PWCs, such as the use of head movements [2] to ease the operation of PWCs and enable different navigation schemes for indoor environments [3–5]. However, there is limited research on low-cost, assistive driving systems that can control PWCs in a semi-automatic fashion with reduced attention from users beyond hands-free operations. To achieve this goal, we present an affordable and intelligent assistive driving system that utilizes computer vision on a smartphone, leveraging its built-in camera for power wheelchairs (PWCs). The system is designed with availability and safety as top priorities.

Our work addresses the above concerns by focusing on the design and development of an affordable, mobile application-based assistive

* Corresponding author at: Department of Electrical and Computer Engineering, The University of Texas at San Antonio, USA.
E-mail address: zhiwei.wang@utsa.edu (Z. Wang).

[1] US Census Bureau. "US Disability Statistics: Facts and Figures". Disabled World, 29 May 2013, https://www.disabled-world.com/disability/statistics/info.php. Accessed 12 July 2022.

[2] https://innovationorigins.com/en/smart-wheelchair-corrects-involuntary-movements-using-ai/

https://doi.org/10.1016/j.sysarc.2024.103105
Received 5 November 2023; Received in revised form 4 March 2024; Accepted 5 March 2024
Available online 6 March 2024

driving solution that can be used on existing power wheelchairs to reduce the need for manual operations significantly. The system harnesses the affordability and widespread availability of popular smartphones with built-in cameras, utilizing deep learning models for automation. Mobile applications (apps) running on smartphones have become increasingly valuable, allowing users to streamline their lives through various functions, including media sharing and GPS navigation. These capabilities are made possible by the cameras and other sensors present in modern mobile devices, such as smartphones. Similarly, deep-learning-based innovations have been shown to outperform human beings in areas such as image classification, object detection, and voice recognition. For example, Man et al. [6] compared the performance of a CNN model and human observers for detecting lesions and concluded that the CNN model outperformed the human observers. Our framework integrates the capability of deep learning image classification with the availability of feature-rich smartphones to provide wheelchair users assistance in moving through indoor spaces safely.

The PWC IA-Driving system is designed as a driving assistance tool rather than a fully automated solution. It offers hands-free navigation in primarily obstacle-free indoor environments like hallways and corridors. Users of the PWC must still manually control the vehicle in areas crowded or dense with obstacles. When in more open spaces, such as uncrowded hallways or lengthy corridors, the system can take on most of the driving tasks, thereby providing substantial stress relief for the PWC user.

This work builds upon our previous work titled “A Vision-Based Low-Cost Power Wheelchair Assistive Driving System for Smartphones”, published in the Proc. of the IEEE International Conference on Embedded Systems and Software (ICESS) 2022 [7]. In the conference paper, we introduced a vision-based solution designed for smartphones to aid in navigating powered wheelchairs (PWCs) in an indoor environment.

We have significantly extended the work reported in our conference paper. The main differences and novel contributions are outlined below.

- We have re-designed the system by incorporating pre-trained Deep Neural Network (DNN) models through the TensorFlow Lite — a streamlined version of TensorFlow specifically tailored for mobile platforms and resource-constrained embedded devices. The newly developed app efficiently leverages the hardware acceleration capabilities of smartphones, achieving a processing speed of up to 25 images per second. This improvement enhances both the safety and robustness of the system. See Section 3-D (pages 6 to 7).
- To address the wide range of hardware architectures present in modern smartphones, we utilized various methods to generate multiple DNN models, each optimized for CPU, GPU, and PTU operations on these devices. App users now have the option to choose from a list of deployed models to optimize performance.
- To enhance the confidence in the model’s potential to generalize effectively across various environments (a requirement for high-risk applications), we employed class activation maps to validate that the DNN model effectively learns the appropriate features and patterns to generate correct driving instructions.

In summary, the PWC IA-Driving system presented in this paper brings forth several noteworthy contributions: (1) a vision-based system to assist the operation of PWCs in indoor environments; (2) the deployment of a deep learning model compressed and optimized using a variety of techniques provided by TensorFlow-lite on an Android smartphone and successful use of images captured by the built-in camera to provide real-time driving instructions; (3) the design of an Android app that interfaces with the PWC user, the deep learning model, and the PWC to safely facilitate navigation. To evaluate this vision-based, low-cost PWC assistive driving system, we measure the accuracy and response time of the system as well as the (reduced) attention time and hands-free period of PWC users.

The remainder of the paper is organized as follows. In Section 2, we provide a review of the closely related work and background that underpins our approach. Section 3 outlines the innovative design of a vision-based, low-cost assistive driving system suitable for smartphone deployment. Section 4 describes the evaluation of the assistive driving system, and Section 5 discusses the possible enhancement of the model. Finally, we conclude the paper and outline plans for future work in Section 6.

2. Background and closely related work

In this section, we survey closely related work and provide the background information necessary to motivate the need for our work.

2.1. Navigation for wheelchairs

Automated driving vehicles (ADVs) have drawn much attention in the past decade, and many industry leaders, including Google, Tesla, and Mobileye, have invested in ADVs. Many research works in the area of ADVs have been published [8–12]. However, automated or assistive driving technology for PWC systems has not been extensively explored. Below are some works that are closely related to this field. Kutbi et al. studied a hands-free wheelchair control approach based on egocentric computer vision and evaluated the scheme with 21 subjects [2]. Compared to joystick and chin-based control, their quantitative and qualitative evaluation results show that the vision-based control approach is viable for hands-free indoor use. However, their approach requires an Egocentric camera and an on-board laptop to support the complex computation to detect head movement. Similarly, other hand-free driving systems for power wheelchairs were developed [13], focusing on alternative means than joysticks to operate a PWC. Notable examples include voice control [14], chin-operated joystick [15], and head-tilt control [16], which significantly improved the usability of power wheelchairs for specific groups of people with disabilities. Unlike our approach, all of the above approaches require users to operate the power chairs with full attention while driving.

Navigation systems have been developed to assist the vision impaired while *walking* in indoor environments [17–20]. These approaches utilize techniques such as sign recognition, obstacle detection, and object positioning to improve user navigation. Working toward similar goals, Ohya et al. [21] proposed a vision-based navigation system for mobile robots in 1998. That approach was based on traditional image-processing algorithms without the use of modern deep-learning models. To our knowledge, such works have not been adapted to PWCs.

Kulhanek et al. proposed a reinforcement learning-based approach to navigate a robot to a target location in a virtual environment [22]. The model was trained to find a way to arrive at the destination given by an image. Our approach is different from this path planning approach in that it does not require a path map or similar instrumentation to the environment being navigated.

Lane-detection-based driving assistance systems for automobiles have been well-studied [23–25] and, at a cursory level, is similar to driving a PWC along corridors inside a building. However, unlike highways, indoor environments do not follow strict rules and regulations (e.g., lane width, road markings, etc.), so fewer assumptions can be made about the environment in the context of PWCs.

2.2. End-to-end autonomous vehicles

Numerous Automated Driving Vehicle (ADV) control systems are composed of various modules, each dedicated to a distinct aspect of autonomous driving. However, as outlined by Le Mero et al. [26], there exists an alternative approach involving a single, distinct module. This module is tasked with directly converting raw sensory inputs (such as those from cameras, LiDAR, etc.) into control signals (like steering and braking commands). Systems employing this methodology are commonly referred to as end-to-end autonomous driving systems. While modular systems may offer easier verification, they tend to lack computational efficiency. In contrast, end-to-end systems are generally more computationally efficient.

In end-to-end Automated Driving Vehicles (ADVs), a single module often employs Deep Learning techniques to train the system for the complex tasks associated with driving. This training is primarily conducted through Imitation Learning [27], a process where human experts supply data (expert policy) to guide the learning system. The system then attempts to learn the optimal policy by observing and emulating the decisions made by the expert. Essentially, Imitation Learning involves acquiring skills by replicating the actions of someone experienced.

Bojarski et al. [28] presents their end-to-end self-driving system that utilizes a convolutional neural network (CNN) trained to map raw pixels from a single front-facing camera directly to steering commands. This system autonomously learns internal representations, using solely the human steering angle as the training signal. Adopting the end-to-end approach, NVIDIA developed PilotNet, a neural network-based system that calculates steering angles from images of the road ahead [29,30]. Utilizing only camera inputs and without the aid of lidar, radar, or maps, the most advanced version of PilotNet is capable of autonomously steering a vehicle for an average of approximately 500 km on highways before requiring human intervention.

Despite PilotNet's success in facilitating ADVs on highways, the system cannot be directly applied to PWC driving systems due to considerable differences in the driving environment, such as the absence of lane markings in indoor settings.

2.3. Image classification

Navigation systems primarily rely on an image classification technique to analyze the images captured by the camera and generate driving instructions. The use of neural networks to classify images has been studied for decades. In 1998, LeCun et al. presented the first convolutional neural network, LeNet-5, to classify images of handwritten digits and achieved great success [31]. The success of LeNet-5 resulted in great interest in studying neural networks. However, given the limited computational power of the time, the architectures of the neural networks in the early 2000s remained shallow. As a result, the error rate of image classification with those models remained high until 2012 when Alex Krizhevsky's team built the famous AlexNet [32] which leveraged the power of GPUs to train the network consisting of eight layers, including five convolutional layers and three fully-connected layers. AlexNet achieved a top-5 error rate (the rate at which a model includes correctly labeled images in its five most confident predictions) of 15.3% and won the ILSVRC 2012 competition.[3] In later competitions, ZFNet [33], considered the extended version of Alexnet, won in 2013 with a top-5 error rate of 11.2%. Inception V1 (GoogLeNet), a 22-layer DNN using 1 × 1-sized filters and Relu to reduce the computation costs, won the competition in 2014 with a top-5 error rate of 6.67%. In the same competition of 2014, VGG [34] won second place with a top-5 error rate of 7.3% and subsequently became one of the most popular models for image classification. ResNet emerged as the victor in ILSVRC 2015 across image classification, detection, and localization. During the competition, ResNet achieved a validation top-5 error rate of 3.57%, surpassing the average human classification error rate. Due to this remarkable performance, our model is built upon ResNet, as detailed in Section 3.2.

Our approach involves the training and deployment of a single deep learning model directly on a smartphone. Such a combination has been successfully implemented for other works such as navigation and speech recognition [35–37]. Despite the significant differences in driving environments and speed limits between Automated Driving Vehicles (ADVs) and Personal Wheelchair (PWC) driving systems, our approach aligns with the principles of end-to-end autonomous vehicle systems. To the best of our knowledge, ours is the first implementation that integrates the Android platform specifically to cater to the unique constraints inherent in PWCs.

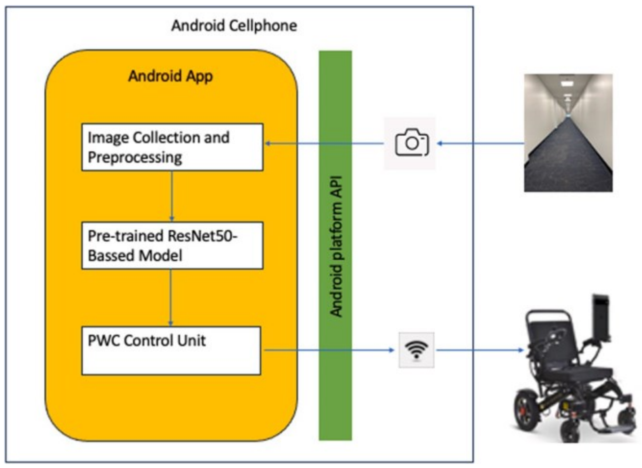

Android Cellphone
Android App
Image Collection and Preprocessing
Pre-trained ResNet50-Bassed Model
PWC Control Unit
Android platform API

Fig. 1. The PWC Intelligent Assistive Driving (PWC IA-Driving) system.

3. PWC IA-drive framework

As shown in Fig. 1, the proposed system comprises an Android app deployed on an Android smartphone. The app incorporates a pre-trained ResNet-based DDN model and code for image processing, PWC interfacing, and user input management. The app accesses essential Android sensors, such as the camera and wireless interface, through the Android platform API. It is important to note that advanced modification of the Android platform (e.g., rooting) is not necessary, and only permissions for the camera and network are required.

The flow of data from the PWC's environment through the pre-trained model to the PWC is illustrated in Fig. 1. Starting from the top right, real-time images of the environment are captured by the app through the Android device's physical camera. Subsequently, the app performs necessary preprocessing steps, including cropping (as detailed in Section 3.1) and down-sampling, to prepare the data for the ResNet-based DNN model. The app then forwards this preprocessed data to the model.

The pre-trained model processes the data and generates one of the four driving instructions, which are turning left, turning right, moving forward, or stopping. These driving instructions are then transmitted to the PWC using an implementation of a universal device interface. This interface can be customized to the specific PWC that the app is configured to communicate with. In our proof-of-concept experiments, we utilized a WiFi connection to interact with a REST API implemented on a Pioneer 3-AT robot.[4]

Fig. 2 shows the derivation of driving commands by training the ResNet-based DDN model. We train the model using images captured by a smartphone to automatically generate the four driving instructions: turning left, turning right, moving forward, and stopping.

[3] https://machinelearningknowledge.ai/popular-image-classification-models-in-imagenet-challenge-ilsvrc-competition-history/

[4] https://sites.google.com/a/nd.edu/discoverlab/robot-platform/ugv/pioneer-robots-1

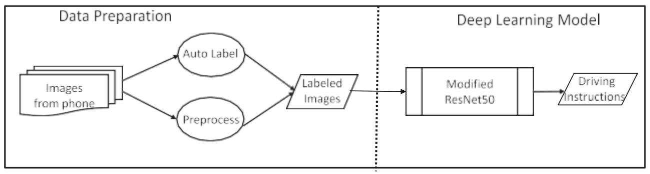


Fig. 2. The training process of the ResNet-based DDN model in PWC IA-Driving system.

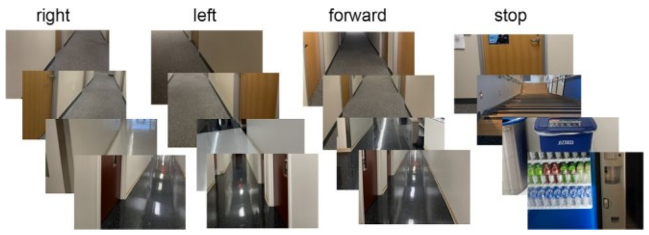


Fig. 3. Labeled training images.

Table 1

Image size vs model accuracy.

Model layers	Full-size	Half-size	Quarter-size
50 layers	96.05%	99.7%	99.21%
40 layers	95.66%	99.87%	98.42%
22 layers	94.61%	98.8%	94.87%
10 layers	92.63%	96.45%	82.50%
2 layers	69.21%	74.87%	76.18%

3.1. Data collection and preprocessing

Since the PWC IA-Driving system is designed to provide indoor driving assistance for PWCs, we gathered data in various indoor settings, including hallways, corridors, and staircases, within multiple buildings where PWCs are commonly utilized. This data was used for the pre-training of the ResNet-based DNN model.

The collected images are auto-labeled based on the orientation of the smartphone's camera in relation to its surroundings. Specifically, if the camera faces the left side of the corridor, the collected images are labeled as "right", indicating the PWC should turn right to avoid colliding with the wall. If the camera faces the right side of the corridor, the collected images are labeled "left", signifying the PWC should turn left to avoid colliding with the wall. If the camera is parallel to the corridor, the collected images are labeled as "forward", indicating that the PWC can continue moving forward without changing direction. If there are obstacles or stairs in front of the camera, the images are labeled as "stop", indicating that the PWC should halt to avoid collisions. Fig. 3 demonstrates some examples of these labels.

For this proof of concept, approximately 10,000 images were collected; 80% of them from different locations were used to train the model, and 20% of them were used to test the model. The images in the training set and test set were taken in different buildings with different architectural styles to demonstrate the generalizability of the system better.

Images taken by a smartphone generally have a 4:3 aspect ratio. We initially used the full images to train the model, resulting in non-ideal accuracy. We hypothesized that the upper portion of the images was less relevant for providing the driving directions, potentially introducing false correlations to the model. Based on this observation, we evaluated the use of three differently-cropped versions of images to determine how the different portions of the images may affect the model's accuracy.

Table 1 shows the results for three different image cropping strategies over models of different sizes that are based on ResNet (Section 3.2). The cropping strategies include the full-size, only the lower

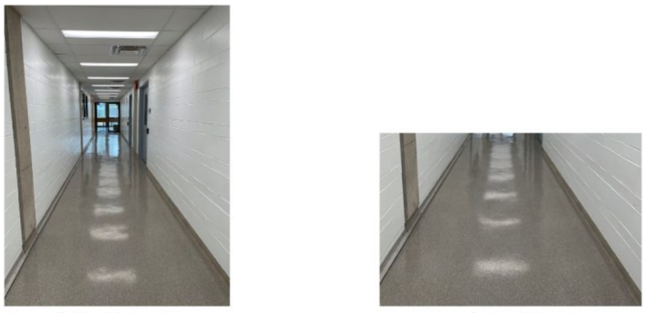


Fig. 4. Image cropping.

half, and only the lower quarter. The experiment results show that using the lower half of images produces the best result for almost all models except for the simple two-layer model, where the lower quarter images perform slightly better than lower half images (76.15% vs. 74.87%).

Based on the experiment results, all images are cropped in half for our approach; only the lower half is kept for model training and actual operation, as shown in Fig. 4. After the cropping, the image aspect ratio becomes 2:3. It is essential for all images collected, for both training and actual operation, to maintain this ratio to ensure that the deep learning model produces high-accuracy results. This pre-processing step significantly enhances the accuracy of the deep learning model. This improvement is likely because the edges of the corridors, where the floor meets the wall, are the most robust orientation indicators (Section 3.3).

3.2. The deep learning model

Safety is the foremost concern of the PWC IA-Driving system. After conducting thorough background research on image classification models (as discussed in Section 2.3), we selected the ResNet (Residual Network) architecture as the foundation for our model due to its superior accuracy in classifying images.

Unlike the previous deeper DNN models where layers are stacked, a residual network consists of several building blocks, allowing for more layers in the DNN without losing accuracy. The ResNet (Residual Network) architecture comes in several variants, often referred to by their depth. Some common variants of ResNet models include ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152, etc.

Considering the computing power and memory available on most smartphones, we selected ResNet-50 as our base model. ResNet-50 consists of five stages: Stage 1 consists of a convolution layer with 64 7×7 filters. Stage 2 includes 3 building blocks, stage 3 includes 4 building blocks, stage 4 includes 6 building blocks, and stage 5 includes 3 building blocks. Each building block consists of one 1×1 convolution layer, one 3×3 convolution layer, and one 1×1 convolution layer. Ultimately, there are a total of 50 layers, including the final output layer.

ResNet was initially designed to classify images into 1000 categories. Therefore, the final output layer consists of 1000 fully connected nodes. In our implementation, we modified the ResNet-50 model by removing the output layer. We added a global average pooling 2D layer and a fully connected layer with four nodes as the final prediction layer. These four nodes correspond to the classes "left", "right", "forward", and "stop".

Our modified ResNet-50 model comprises 23,595,908 parameters, of which 23,542,788 are trainable, and 53,120 are non-trainable. For training, we utilized a computer equipped with two Intel Cascade Lake CPUs (40 cores total), 384 GB RAM, and an NVIDIA V100 GPU featuring 32 GB of memory, 5120 CUDA Cores, and 640 Tensor Cores.

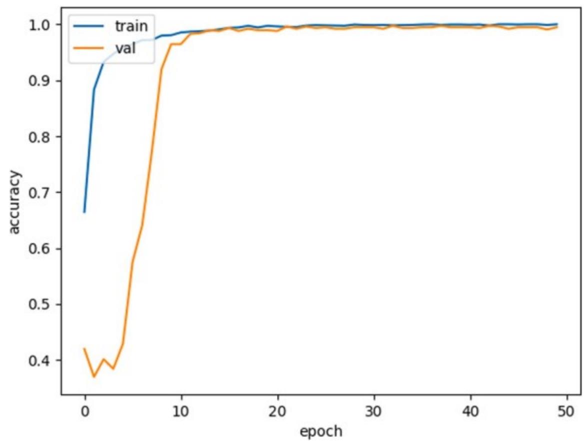


Fig. 5. Model accuracy.

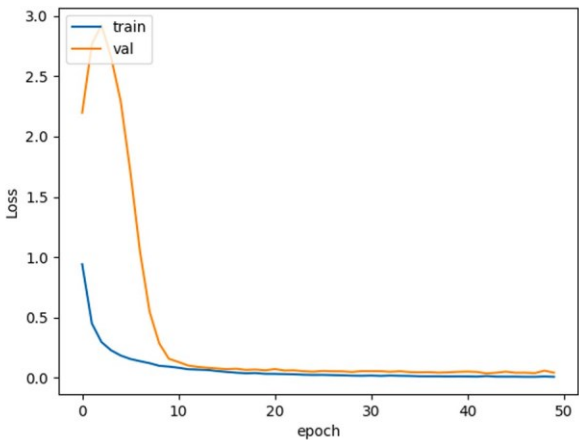


Fig. 6. Model loss.

Table 2
Model performance.

Model layers	Accuracy	Inference time (ms)
50 layers	99.7%	510
40 layers	**99.87%**	**410**
22 layers	98.8%	290
10 layers	96.45%	180
2 layers	74.87%	110

The model was trained using the Keras framework, with a learning rate set at 0.001, a mini-batch size of 32, and SGD as the optimizer, including a momentum of 0.9. The loss function used was "categorical_crossentropy". Conducting training over 50 epochs and incorporating "EarlyStopping" callbacks with "restore_best_weights" set to True, the model achieved an impressive test accuracy of 99.7%. Despite the remarkable accuracy of 99.7% achieved by the ResNet-50-based model, we notice that, apart from its relatively large size, the model took an average of 0.5 s to process an image on a Pixel 6 smartphone without any optimization or hardware acceleration. Since simpler models take less time to process images, it would be intriguing to find out if simpler models with fewer trainable parameters can achieve similar or even better results. With this objective in mind, we built four more models based on the ResNet-50 structure: attaching the two new layers to the output of stage 4 to form a 40-layer model; attaching the two new layers to the output of stage 3 to form a 22-layer model; attaching the two new layers to the output of stage 2 to form a 10-layer model; and attaching the two new layers to the output of stage 1 to form a simple 2-layer model with one convolution layer. We tested a Pixel 6 smartphone using all four models without any model optimizations or hardware acceleration. The accuracy and image processing speed of each model are shown in Table 2. From the table, we can see that the 40-layer model performs better on both accuracy and speed from the five tested models. Considering the hardware accelerations available in modern smartphones, processing times can be significantly improved, potentially up to ten times faster as shown in Table 3. Therefore, there is a lessened need to sacrifice accuracy for quicker processing by opting for a simpler model, such as the 22-layer one. Our approach focuses on the 40-layer model as our target for this project.

Fig. 5 shows the training accuracy vs. test accuracy over the number of epochs, and Fig. 6 shows the training loss vs. test loss over the number of epochs for our selected model. The training process has no obvious over-fitting, except the test loss increases slightly at epoch 43. Therefore, our final target model is trained for 43 epochs with an accuracy of 99.87%, as shown in Table 2.

3.3. Model interpretability

Although our constructed and trained model achieves commendable accuracy in producing driving instructions, it is imperative to conduct qualitative analysis to gain insights into its underlying mechanics for accurate predictions. This approach will enhance our confidence in the model's potential to generalize effectively across various environments. This is crucial for high-risk applications such as self-driving systems, where the consequences of wrong predictions can be deadly. Such an analysis is necessary to foster trust, particularly if we can confirm that the model has learned the correct features and patterns it is supposed to identify.

In general, the *explainability* of a model's prediction is linked to the collection of features contributing to the decision [38]. In our case, the features can be scores of regions in the input image contributing to the final decision. There are many interpretation methods for DNN image classification models (e.g., Saliency maps [39], Class Activation Map (CAM) [40], and Gradient-CAM [41]). Among those approaches, Class Activation Map (CAM) based heat maps visualize how the DNN model values different parts of the input image when it makes a prediction. This is done by highlighting the pixels of the input image most strongly supporting the classification decision. By examining the highlighted regions of predictions, users will have greater confidence in the model if these regions are genuinely meaningful to the prediction. We use such heat maps over randomly selected images from each decision type to visualize how the model decides.

To produce the heat map, we modify the model by using its final output and the extracted output of the last convolution layer to form a dual-output model: one is the output of the convolution layer, and one is the prediction result. The convolution layer output is a $7 \times 7 \times 2048$ tensor, which can be considered as 2048 features of 7×7 images. The results are based on each prediction's 2048 features and weight vectors. The heat map for an input image combines the input image and the product of features, multiplying the weight vector for the predicted result (dot product) [42]. We select one image from each of the four categories in the test set and produce the heat maps as shown in Fig. 7. As seen in the resulting heat maps in Fig. 7, the PWC model uses the baseboards or joints between the floor and the wall along the corridor as an indication of "lanes". For example, the model should predict a "turning right" if the PWC is moving toward the left lane (the baseboard on the left). In the corresponding heat map of the prediction, the regions of the edge between the wall and floor on the left side of the corridor should be highlighted. Conversely, for a prediction of moving forward, the edges on both sides should be highlighted in the corresponding heat map. For a "stop" prediction, the obstacles should be highlighted in the corresponding heat map. These findings confirm

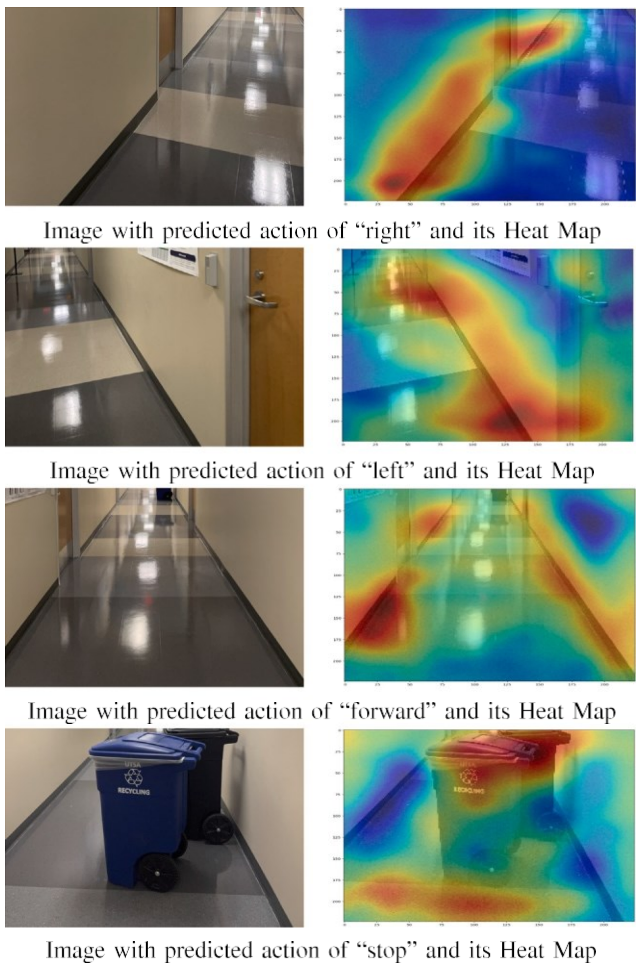


Image with predicted action of "right" and its Heat Map
Image with predicted action of "left" and its Heat Map
Image with predicted action of "forward" and its Heat Map
Image with predicted action of "stop" and its Heat Map

Fig. 7. Heat map.

that the model effectively utilizes relevant features for its predictions. Consequently, we can have confidence in the model's performance and ability to generalize to unseen data.

3.4. Model deployment and Android app

TensorFlow Lite (TF-lite), developed by Google, is a lightweight variant of TensorFlow explicitly designed for mobile platforms and embedded devices with limited resources. Its purpose is to enable the deployment of pre-trained TensorFlow models on resource-constrained devices such as smartphones. TensorFlow Lite converts the models into specialized formats suitable for such devices to achieve this. Optimization methods like operation fusion, quantization, and model compression enhance performance during the conversion process. Depending on the available hardware on the smartphone, the transformed models can run on a CPU via the CPU delegate or run on a GPU or Tensor Chip to further speed up via GPU and the Neural Networks API (NNAPI) delegates. We evaluated and compared the performance of the models optimized using these methods, ultimately selecting the most suitable one for deployment.

Our model evaluations were performed on a Google Pixel 6 smartphone, equipped with an ARM64-based system-on-chip (SoC) Google Tensor Processing Unit (TPU), a 20-core Mali-G78 MP20 GPU with 8 GB of memory, and an Octa-core CPU. The CPU features a tri-cluster configuration of 2+2+4: two Cortex-X1-based performance cores at 2.8 GHz, two Cortex-A76-based medium cores at 2.2 GHz, and four Cortex-A55-based efficiency cores running at 1.8 GHz. The evaluated models include four models: the original TensorFlow model, the TF-lite converted model without any optimizations, the TF-lite converted model with float16 quantization, and the TF-lite converted model with dynamic range quantization. We tested the non-TF-lite model using Pydroid, a Python environment for Android. To the authors' knowledge, Pydroid does not directly support device accelerations. Hence, our testing focused on running the model on the device's CPU. TF-lite is designed with built-in support for executing models on different hardware, including CPUs, GPUs, and Tensor Chips. Therefore, we assessed the performance of the three TF-lite converted models across each of these hardware units through TF-lite's CPU, GPU, and NNAPI delegates.

As shown in Table 3, The TF-lite converted models demonstrated significantly better performance than the original TensorFlow model when running with Pydroid, even without optimizations or hardware accelerations. Additionally, the sizes of the TF-lite converted models have been significantly reduced, simplifying the model development process. Among the three TF-lite converted models, the one without any optimization performs the best with the NNAPI delegate. It achieves a processing speed of nearly 30 images per second, which makes it an ideal model for smartphones with advanced Tensor Chip. On the other hand, the model with dynamic range quantitation performs the best with the CPU delegate. It achieves a processing speed of nearly 18 images per second, which makes it a perfect choice for smartphones without specialized processing capabilities.

As a proof of concept, we evaluated the navigational application using a Pioneer 3-AT robot. The robot was configured to run at speeds of up to 700 mm/s to simulate the approximate speed of a wheelchair. Additionally, we attached a bracket to hold the smartphone with the camera facing the front of the robot.

We have designed an Android app as shown in Fig. 8. The application includes a user-friendly interface with a four-direction control system for effortless manual control of the PWC. Users can easily switch between automatic and manual operation modes using a dedicated toggle switch. Furthermore, the app offers a range of customizable options, such as model selection, motor speed adjustments, hardware delegation preferences, and various essential tasks. Moreover, the application provides real-time feedback on preprocessing time, inference time, and decision confidence scores, enhancing the overall user experience.

The app leverages images captured by the smartphone's camera. It performs crucial preprocessing tasks such as cropping and resizing before passing them to the autonomous driving model to generate driving instructions, including commands to turn left, turn right, move forward, or stop. These instructions are then communicated to the robot.

3.5. Safety consideration

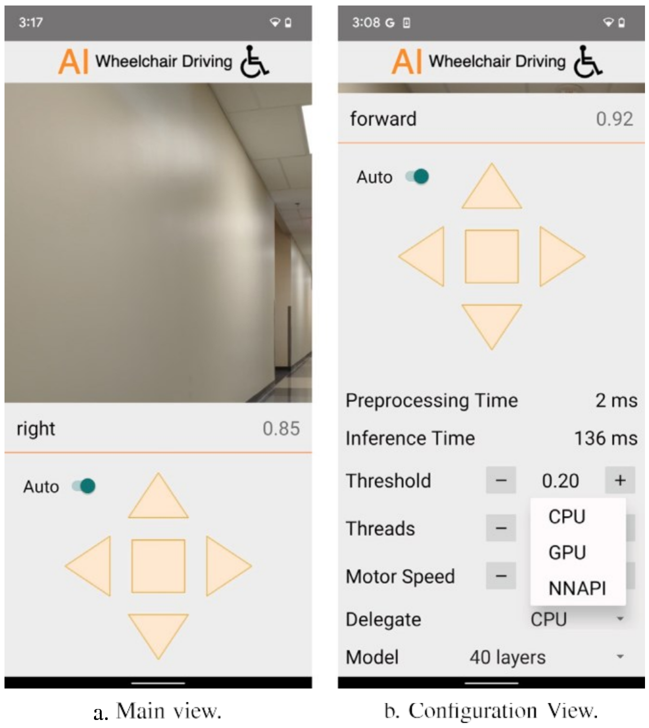
Safety is of utmost importance for any driving assistance system. Besides model accuracy, we consider response time, which is directly related to the PWC's velocity, to be a major indicator of safety. There are speed limits for Powered wheelchairs in most countries. For example, PWCs must not travel faster than 4 mph (1.79 m/s) in the UK.[5]; in the USA, most of PWCs cannot go faster than 5 mph (2.24 m/s)[6] With a significantly improved processing rate of 25 images/second over the previous version on the Pixel 6 using Google's tensor chip, the system could theoretically respond to obstacles as close as 0.07 and 0.09 m, respectively. On a lower-end smartphone without any hardware acceleration, the processing rate can still reach 18 images/second, ensuring a reaction range of 0.10 m and 0.124 m, respectively. Our model is trained to recognize obstacles within a two-meter range. The processing

[5] https://www.gov.uk/guidance/the-highway-code/rules-for-users-of-powered-wheelchairs-and-mobility-scooters-36-to-46

[6] https://www.wheelchairjunkie.com/speedselection/

Table 3
Model performance running on smartphone (PIXEL 6).

Model	Size (MB)	Accuracy	CPU delegate (ms)		GPU delegate (ms)		NNAPI delegate (ms)	
			Preprocessing	Inference	Preprocessing	Inference	Preprocessing	Inference
Original without TF-lite conversion	51.4	99.87%	2	410	–	–	–	–
TF-lite without optimization	25.2	99.87%	2	130	4	85	5	35
TF-lite optimized with float16	12.6	99.87%	2	128	3	92	2	167
TF-lite optimized with dynamic range	6.5	99.87%	2	55	3	86	2	57



Preprocessing Time 2 ms
Inference Time 136 ms
Threshold 0.20
Threads
Motor Speed
Delegate CPU
Model 40 layers

Fig. 8. Android app.

rate is more than sufficient to ensure that a PWC will receive a "stop" instruction when encountering an obstacle, even if the PWC is traveling at its maximum speed. The unlikelihood of high-speed travel within indoor environments further strengthens these safety characteristics.

Safety is further improved by the app's control override feature. The user can always override the instructions given by the model by operating the joy-stick-like buttons of the app.

4. Experimental evaluation and discussion

In this section, we report our experimental evaluation to address the following research questions.

- **RQ1:** Can the proposed system offer hands-free, and reduced-attention control in operating wheelchairs?
- **RQ2:** Can the vision-based assistive-driving system with deep learning models make timely and safe decisions to control the wheelchair's movement?

We have implemented a prototype mobile application that integrates the modified and pre-trained ResNet50 models, the image-capturing function, and the control interface for a mobile robot as the proof-of-concept design. The mobile application runs on a Google Pixel 6 smartphone. The Pioneer 3-AT mobile robot has three 12 V/9 Ah batteries that can power the robot to reach speeds up to 0.8 m/s. To simulate the average operation speed of wheelchairs for indoor spaces, where the regulation permits to run up to 4 mph (1.79 m/s) in an outdoor environment,[7] we adjust the control of the mobile robot to make it run at a speed up to 0.7 m/s.

Table 4
Travel time, hands-free ratio, and attention time of the assistive app under different test cases.

Test case	Travel time (s)	Hands-free (%)	Attention time (s)
1 (baseline)	24	0	24
2	24	100	0
3	27	100	3
4	35	100	11
5	31	83.8	7
6	26	100	0
7	26	100	0

4.1. Test cases and hands-free driving

To answer **RQ1**, we considered seven different test cases for the assistive-driving application described in Section 3.4 with or without various obstacles in a 20-meter-long corridor (as detailed below). We collected the respective total travel time, the ratio of hands-free time (i.e., total travel time minus the manual operation time over the total travel time), and attention time (i.e., the time requiring user attention due to stops at obstacles, though the user may not need to manually operate the robot). The results are shown in Table 4.

- **Test Case 1**: The user operated the web-based robot control interface to manually operate the robot at its highest speed of 0.7 m/s where there is no obstacle in the corridor. It took 24 s to reach the spot about 3 m before the end of the corridor. Here, the robot has a safety distance of 3 m to avoid bumping into walls. This is the baseline case for comparison, where the user needs to operate the robot all the time (i.e., 0% hands-free time) with full attention.

- **Test Case 2**: In this case, the robot was positioned in parallel with the corridor initially and there is no obstacle in the corridor. The assistive-driving application running on the Pixel 6 controlled the robot's movement based on the derived driving instructions from the images captured in real time through the phone's camera. The robot also took 24 s to reach the exact location about three meters before the corridor's end and stopped. Here, the assistive-driving application running on Pixel 6 could navigate the robot at the same speed as a human operator in Test Case 1 without any collision and stop in the middle of the operation. In this ideal case, no manual operation is needed (i.e., 100% hands-free) and no attention of the user is called since there is no obstacle and no stop.

- **Test Case 3**: This case used the same parameters as in Test Case 2 except that a stationary obstacle (trash can) was positioned in the middle of the corridor, 10 m from the end. In this case, the assistive-driving application controlled the robot until the application detected the trash can and made the robot stop about two meters before the obstacle. We assume that the user will realize and react when the robot stops, at which time the user could ask for help to move the obstacle away or manually drive the robot around it. In this case, the trash was moved away and the robot resumed moving forward under the control

[7] https://www.nidirect.gov.uk/articles/rules-users-powered-wheelchairs-and-mobility-scooters-36-46

of the application without the user's manual operation. It took 27 s for the robot to reach the target destination before the corridor's end, where the extra 3 s were due to the robot stopping and requiring the attention of the user.

- **Test Case 4**: This case used the same parameters as in Test Case 2 except that the stationary obstacle was replaced with a moving obstacle (person) moving across the corridor three times where the distances from the robot were 1, 1.5, and 2 m. In this case, the assistive-driving application detected all three occurrences of moving obstacles and directed the robot to stop safely without collision. Once the person walked away (beyond the two-meter range), the application continued driving the robot forward. It took 35 s for the robot to reach the target destination. The attention of the user was called each time the robot stopped for a total of 11 s. However, as with Test Case 3, no manual operation was required.

- **Test Case 5**: This case used the same parameters as in Test Case 3 where the stationary obstacle (trash can) was positioned in the middle of the corridor, 10 m from the end. Differently from Test Case 3, after the robot is stopped under the control of the assistive-driving application, instead of moving the trash can away, we emulated the case where the user takes control and manually operates the robot to move around the trash can. After manual intervention, the assistive-driving application took control again and navigated the robot to the destination. It took 31 s in total in this case where the user was called to attention for 7 s with 5 s used to manually navigate the robot around the obstacle.

- **Test Case 6**: This case used the same parameters as in Test Case 2, except that the robot was initially angled slightly towards the *left* of the corridor. The assistive-driving application detected that the robot was not parallel with the corridor at the beginning and navigated the robot to continuously turn *right* at reduced speeds until it faced directly parallel within the corridor. With the reduced turning speeds and refined driving commands derived by the assistive-driving application, we observed that there was no over-correcting of the angle during the experiment. Once the robot corrected its direction, it proceeded forward to the target destination. The robot took 26 s in total in this case. No manual operation was necessary, since the PWC did not need to stop.

- **Test Case 7**: This case used the same parameters as in Test Case 6, except that the robot was initially angled slightly towards the *right* of the corridor. As in Test Case 6, the robot successfully corrected the angle and navigated to the target destination in 26 s.

4.2. Safety

To answer **RQ2**, we observed the reaction times of our system during the test cases where obstacles were placed in front of the robot. For all the test cases, the robot was able to stop when it detected an obstacle within 1–3 m in front of it. As an eighth test case, we observed that the system would also stop if it was driving toward stairs. Moreover, to maintain the user's awareness in case manual override is necessary, the mobile application requires the user's interaction (e.g., touching the screen) at least every 35 s. The app will issue a warning message if there is no user interaction for more than 30 s, followed by issuing a "stop" command to the robot if there is still no user interaction for another five seconds. As part of our future work, we also plan to incorporate the phone's screen-side "selfie" camera to measure user attention.

5. Discussion

In this section, we discuss the results of the evaluation and potential implications.

5.1. Results and model enhancement

Our modified ResNet model was pre-trained on a server using images captured in a variety of indoor environments and deployed to an Android-based smartphone. The results of the test cases verified the efficacy of our system as the PWC was successfully navigated to its destination even in the presence of various obstacles and situations. Furthermore, the system exhibited adequate response time to ensure safety. In cases where the model makes incorrect driving decisions, our system will allow users to override the model-derived driving instructions by manually operating the in-app joystick.

To improve our system in the future, we kept the images that caused wrong decisions in the experiments to form a new data set. We plan to utilize the new data set to enhance the model using three different strategies: (1) Merge the dataset utilized for training the pre-trained model with the new dataset to conduct a retraining of the model; (2) use only the new data set to retrain the model with a much smaller learning rate; (3) create an ensemble model [43] to combine the pre-trained model and the new model trained only with the new data set. We will also consider using reinforcement learning to train a model.

5.2. Privacy protection

Since the introduction of the smartphone, millions of mobile apps, such as Google Maps, Weather apps, and Fitness apps, have been widely used to simplify and improve the quality of human lives. As with the app presented in this paper, many apps require access to sensors, including GPS and cameras, which can produce sensitive personal information. To protect privacy, regulations require mobile apps to inform each user if such sensitive data is collected and processed. Our assistive driving system uses the built-in camera of a smartphone to capture images of hallways, which can include the people walking in them. Privacy is a natural concern when real-time images are captured during operation. In our approach, the risk to privacy is lessened as only the lower portion of the images are saved for training and test purposes, and those cropped images contain no personally identifiable info such as human faces. Furthermore, the images captured in real-time are processed locally on the phone without interaction with other cloud services, and they are discarded immediately after the model produces the driving instructions.

6. Conclusions and future work

This paper describes an affordable, Intelligent Assistant driving (IA-driving) system that has the potential to be used in power wheelchairs to assist those with mobility impairments. We demonstrate that a smartphone app-based system could provide a highly accessible, low-cost solution to assist users in operating PWCs in a hands-free and attention-free manner.

As a continuation of this work, we will work to incorporate incremental learning techniques utilizing the data captured as the PWC user overrides the model decisions as discussed in Section 5.1 to improve the model. We will also expand the training data to include more obstacles in different environments, including home and outdoor settings. Finally, we will explore the use of virtual and augmented reality and Reinforcement Learning to improve the system.

CRediT authorship contribution statement

Zhiwei Wang: Writing – review & editing, Writing – original draft, Validation, Software, Investigation, Formal analysis, Data curation, Conceptualization. **Jingye Xu:** Writing – original draft, Validation, Software. **Jianqiu Zhang:** Writing – review & editing, Investigation, Conceptualization. **Rocky Slavin:** Writing – review & editing, Resources, Investigation. **Dakai Zhu:** Writing – review & editing, Supervision, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported in part by NSF Awards 1948244 and 2007718. This work received computational support from the University of Texas at San Antonio's HPC cluster Arc, operated by University Technology Solutions.

References

[1] A.O. Frank, L.H.D. Souza, J.L. Frank, C. Neophytou, The pain experiences of powered wheelchair users, Disab. Rehab. 34 (9) (2012) 770–778, http://dx.doi.org/10.3109/09638288.2011.619620, PMID: 22013954.

[2] M. Kutbi, X. Du, Y. Chang, B. Sun, N. Agadakos, H. Li, G. Hua, P. Mordohai, Usability studies of an egocentric vision-based robotic wheelchair, ACM Trans. Hum.-Robot Interact. 10 (1) (2020).

[3] G. Fusco, J.M. Coughlan, Indoor localization for visually impaired travelers using computer vision on a smartphone, in: Proceedings of the 17th International Web for All Conference, 2020.

[4] B. Li, J.P. Muñoz, X. Rong, Q. Chen, J. Xiao, Y. Tian, A. Arditi, M. Yousuf, Vision-based mobile indoor assistive navigation aid for blind people, IEEE Trans. Mob. Comput. 18 (3) (2019) 702–714.

[5] C. Manlises, A. Yumang, M. Marcelo, A. Adriano, J. Reyes, Indoor navigation system based on computer vision using camshift and d algorithm for visually impaired, in: The 6th IEEE Int'L Conference on Control System, Computing and Engineering, ICCSCE, 2016, pp. 481–484.

[6] R. De Man, G.J. Gang, X. Li, G. Wang, Comparison of deep learning and human observer performance for lesion detection and characterization, in: 15th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine, vol. 11072, SPIE, 2019, pp. 239–243.

[7] Z. Wang, K. Liu, J. Wang, J. Xu, J. Chen, Y. Jin, R. Slavin, A vision-based low-cost power wheelchair assistive driving system for smartphones, in: 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys), IEEE, 2022, pp. 1979–1986.

[8] Ö.Ş. Taş, S. Hörmann, B. Schäufele, F. Kuhnt, Automated vehicle system architecture with performance assessment, in: IEEE International Conference on Intelligent Transportation Systems, ITSC, 2017, pp. 1–8, http://dx.doi.org/10.1109/ITSC.2017.8317862.

[9] S. Liu, J. Tang, Z. Zhang, J.-L. Gaudiot, Computer architectures for autonomous driving, Computer 50 (8) (2017) 18–25.

[10] K. Jo, J. Kim, D. Kim, C. Jang, M. Sunwoo, Development of autonomous car—Part II: A case study on the implementation of an autonomous driving system based on distributed architecture, IEEE Trans. Ind. Electron. 62 (8) (2015) 5119–5132.

[11] C. Guo, C. Sentouh, J.-B. Haué, J.-C. Popieul, Driver–vehicle cooperation: a hierarchical cooperative control architecture for automated driving systems, Cogn. Technol. Work 21 (4) (2019) 657–670.

[12] D. Dominic, S. Chhawri, R.M. Eustice, D. Ma, A. Weimerskirch, Risk assessment for cooperative automated driving, in: Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy, 2016, pp. 47–58.

[13] M. Kutbi, Y. Chang, P. Mordohai, Hands-free wheelchair navigation based on egocentric computer vision: A usability study, in: Conference: Workshop on Assistance and Service Robotics in a Human Environment At: Vancouver, British Columbia, Canada, 2017.

[14] M. Fezari, M. Bousbia-Salah, Speech and sensor in guiding an electric wheelchair, Autom. Control Comput. Sci. 41 (1) (2007) 39–43.

[15] R. Lipskin, An evaluation program for powered wheelchair control systems, Bull. Prosth. Res. 6 (1970) 121–219.

[16] S.-H. Chen, Y.-L. Chen, Y.-H. Chiou, J.-C. Tsai, T.-S. Kuo, Head-controlled device with M3S-based for people with disabilities, in: Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No. 03CH37439), vol. 2, IEEE, 2003, pp. 1587–1589.

[17] C. Manlises, A. Yumang, M. Marcelo, A. Adriano, J. Reyes, Indoor navigation system based on computer vision using camshift and d* algorithm for visually impaired, in: 2016 6th IEEE International Conference on Control System, Computing and Engineering, ICCSCE, IEEE, 2016, pp. 481–484.

[18] B. Li, J.P. Munoz, X. Rong, Q. Chen, J. Xiao, Y. Tian, A. Arditi, M. Yousuf, Vision-based mobile indoor assistive navigation aid for blind people, IEEE Trans. Mob. Comput. 18 (3) (2018) 702–714.

[19] G. Fusco, S.A. Cheraghi, L. Neat, J.M. Coughlan, An indoor navigation app using computer vision and sign recognition, in: International Conference on Computers Helping People with Special Needs, Springer, 2020, pp. 485–494.

[20] A.G. Karkar, S. Al-Maadeed, J. Kunhoth, A. Bouridane, CamNav: a computer-vision indoor navigation system, J. Supercomput. 77 (7) (2021) 7737–7756.

[21] I. Ohya, A. Kosaka, A. Kak, Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing, IEEE Trans. Robot. Autom. 14 (6) (1998) 969–978.

[22] J. Kulhánek, E. Derner, T. De Bruin, R. Babuška, Vision-based navigation using deep reinforcement learning, in: 2019 European Conference on Mobile Robots, ECMR, IEEE, 2019, pp. 1–8.

[23] T. Sato, Q.A. Chen, Towards driving-oriented metric for lane detection models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2022, pp. 17153–17162.

[24] J.-W. Lee, S.-U. Choi, Y.-J. Lee, K. Lee, A study on recognition of road lane and movement of vehicles using vision system, in: SICE 2001. Proceedings of the 40th SICE Annual Conference. International Session Papers (IEEE Cat. No.01TH8603), 2001, pp. 38–41, http://dx.doi.org/10.1109/SICE.2001.977802.

[25] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, et al., An empirical evaluation of deep learning on highway driving, 2015, arXiv preprint arXiv:1504.01716.

[26] L. Le Mero, D. Yi, M. Dianati, A. Mouzakitis, A survey on imitation learning techniques for end-to-end autonomous vehicles, IEEE Trans. Intell. Transp. Syst. 23 (9) (2022) 14128–14147, http://dx.doi.org/10.1109/TITS.2022.3144867.

[27] A. Hussein, M.M. Gaber, E. Elyan, C. Jayne, Imitation learning: A survey of learning methods, ACM Comput. Surv. 50 (2) (2017) 1–35.

[28] M. Bojarski, D.D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L.D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, K. Zieba, End to end learning for self-driving cars, 2016, arXiv:1604.07316.

[29] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, U. Muller, Explaining how a deep neural network trained with end-to-end learning steers a car, 2017, arXiv:1704.07911.

[30] M. Bojarski, C. Chen, J. Daw, A. Değirmenci, J. Deri, B. Firner, B. Flepp, S. Gogri, J. Hong, L. Jackel, Z. Jia, B. Lee, B. Liu, F. Liu, U. Muller, S. Payne, N.K.N. Prasad, A. Provodin, J. Roach, T. Rvachov, N. Tadimeti, J. van Engelen, H. Wen, E. Yang, Z. Yang, The NVIDIA PilotNet experiments, 2020, arXiv:2010.08776.

[31] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[32] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[34] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.

[35] N.D. Lane, P. Georgiev, L. Qendro, Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning, in: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2015, pp. 283–294.

[36] A. Mathur, N.D. Lane, S. Bhattacharya, A. Boran, C. Forlivesi, F. Kawsar, Deepeye: Resource efficient local execution of multiple deep vision models using wearable commodity hardware, in: Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, 2017, pp. 68–81.

[37] M. Xu, M. Zhu, Y. Liu, F.X. Lin, X. Liu, Deepcache: Principled cache for mobile deep vision, in: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, 2018, pp. 129–144.

[38] W.J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, B. Yu, Definitions, methods, and applications in interpretable machine learning, Proc. Natl. Acad. Sci. 116 (44) (2019) 22071–22080, http://dx.doi.org/10.1073/pnas.1900654116.

[39] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013, http://dx.doi.org/10.48550/ARXIV.1312.6034, URL https://arxiv.org/abs/1312.6034.

[40] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, 2015, http://dx.doi.org/10.48550/ARXIV.1512.04150, URL https://arxiv.org/abs/1512.04150.

[41] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual explanations from deep networks via gradient-based localization, Int. J. Comput. Vis. 128 (2) (2019) 336–359, http://dx.doi.org/10.1007/s11263-019-01228-7.

[42] Seachaos, Get heatmap from CNN (convolution neural network), AKA cam, 2021, https://tree.rocks/get-heatmap-from-cnn-convolution-neural-network-aka-grad-cam-222e08f57a34.

[43] M.A. Ganaie, M. Hu, et al., Ensemble deep learning: A review, 2021, arXiv preprint arXiv:2104.02395.