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

A Vision-Based Low-Cost Power Wheelchair Assistive Driving System for Smartphones

Zhiwei Wang[1,2], Kevin Liu[3], Jeffrey Wang[4], Jingye Xu[5], Jingjing Chen[5], Yufang Jin[1], and Rocky Slavin[5]

[1]Department of Electrical and Computer Engineering, The University of Texas at San Antonio, USA
[2]The Research Computing Support Group of The University Technology Solutions, The University of Texas at San Antonio, USA
[3]Department of Computer Science, University of Western Ontario, Canada
[4]Keystone School, San Antonio, USA
[5]Department of Computer Science, The University of Texas at San Antonio, USA

***Abstract*—Power wheelchairs (PWC) are essential for people with mobility impairment, and many research studies have been reported to ease their operations. However, the existing approaches either rely on extra hardware components or demand complex software that incurs high costs. In this work, we propose a low-cost, computer-vision-based assistive driving system that runs on a smartphone with the objective of safely driving a PWC in a *hands-free* manner with *reduced attention* in an indoor environment to relieve the arduous operations of disabled users and reduce their stress. The system adopts a *modified* and *pre-trained* ResNet-50 model running on a smartphone to derive the driving instructions using the images captured in real-time with its built-in camera. The smartphone interacts with a control interface to send the driving instructions to the PWC. A prototype of the proposed driving assistive system is implemented on a Pixel-6 Android phone and evaluated on a mobile robot as the proof-of-concept design. The experiments show that the smartphone can process input at up to 3.4 images per second to generate driving instructions in time to safely navigate the mobile robot at reasonable speeds in the testing environment with minimal intervention from the user.**

I. 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 which 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, motor, 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 [6]. 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 navigation system would offer a more ideal solution. Unfortunately, such navigation systems are still in design and development phases in labs and are prohibitively expensive[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 [19] to ease the operation of PWCs and enable different navigation schemes for indoor environments [8], [24], [27]. 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 this end, we propose a low-cost, semi-autonomous assistive driving system for 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 driving solution that can be used on existing power wheelchairs to significantly reduce the need of manual operations. The system harnesses the affordability and availability of mobile devices with the automation enabled by deep learning models. Mobile applications (apps) running on smartphones have become increasingly useful, enabling users to simplify their lives in various ways such as sharing media, making purchases, ordering food, and navigating maps thanks to the cameras and other sensors available on mobile devices. 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. [3] compared the performance of a CNN model and human observers for detecting lesions and concluded that the CNN model outperformed the human observers in general.

[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/

979-8-3503-1993-4/22/$31.00 ©2022 IEEE
DOI 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00295

Our framework integrates the success of deep learning image classification with the availability of feature-rich smart phones to provide wheelchair users assistance in moving through indoor spaces safely.

In this paper, we present the following contributions: (1) a vision-based model to assist the operation of PWCs in indoor environments; (2) the deployment of a deep learning model 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 using modified ResNet models, 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 II, we review the closely related work and background upon which we base our approach. Section III presents the novel design of a vision-based, low-cost assistive driving system which can be deployed on smartphones. Section IV describes the evaluation of the assistive driving system and Section V discusses the possible enhancement of the model. Finally, we conclude the paper and present plans for future work in Section VI.

II. Related Work and Background

In this section, we survey closely related work and provide the background information necessary to understand our approach.

A. Closely Related Works

Many works exist regarding the use of image classification for navigation. We describe the most closely related works to ours here.

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. Although many research works in the area of ADVs have been published [4], [11], [14], [26], [35], automated or assistive driving for PWCs has not been well studied. Some closely related works are described below.

Kutbi et al. studied a hands-free wheelchair control approach based on Egocentric computer vision and evaluated the scheme with 21 subjects [19]. Compared to joystick control 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 [18], focusing on alternative means than joysticks to operate a PWC. Notable examples include voice control [5], chin-operated joystick [25], and head-tilt control [2], 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 100% of the time during driving.

Navigation systems have been developed to assist the vision impaired while *walking* in indoor environments [7], [15], [23], [28]. These approaches utilize techniques such as sign recognition, obstacle detection, and object positioning to improve user navigation. Working toward similar goals, Ohya et al. [30] 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.

Our approach deploys a deep learning model directly on a mobile device. Such a combination has been successfully implemented for other works such as for navigation and speech recognition [20], [29], [36]. To our knowledge, ours is the first which incorporates the Android platform to address the specific constraints inherent to PWCs.

Kulhanek et al. proposed a reinforcement learning-based approach to navigate a robot to a target location in a virtual environment [17]. The model was trained to find a way to arrive at the destination given by a 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 has been well-studied [13], [22], [31] 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.

B. Image Classification

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 [21]. LeNet-5 consists of two convolution layers, three fully-connect layers, and an input layer taking 28x28x1 images. 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 [16] 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 incorrectly labeled images in its five most confident predictions) of 15.3% and won the ILSVRC 2012 competition[3]. Before AlexNet, models typically achieved a top-5 error rate no lower than 25% [16]. As a result

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

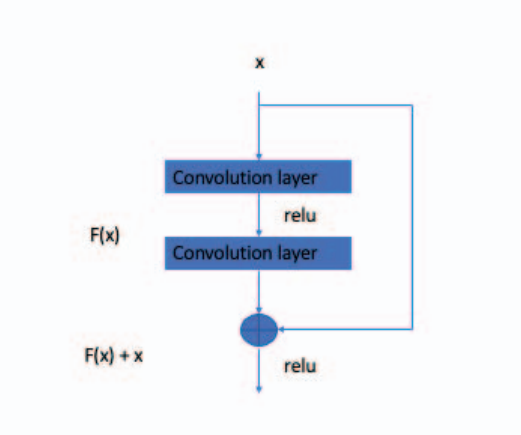

Fig. 1. Building Block of a Residual Network.

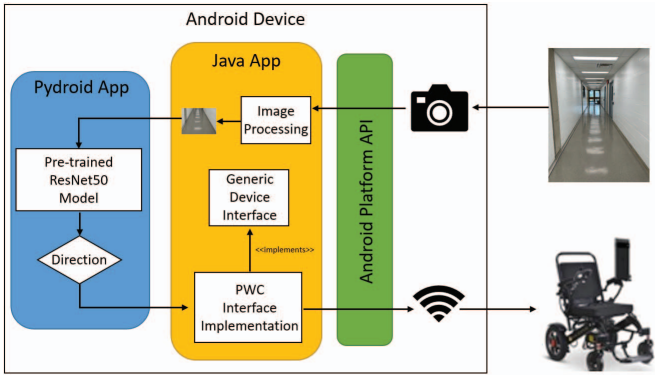

Fig. 2. Overview of the vision-based assistive driving system for PWC.

of this breakthrough, AlexNet is considered as the starting point of the Deep Neural Network (DNN) boom. In later competitions, ZFNet [34], considered as 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 1x1-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 [32], won the second place with a top-5 error rate of 7.3% and became one of the most popular models for image classification.

Although researchers realized that network depth is critical to improving model accuracy, a very deep neural network is difficult to train because of how gradient explosion and vanishing issues hinder model convergence [10]. Many works have been done to address the issues [1], [10]. However, even if a very deep neural network converges, simply adding more layers to a network will decrease the accuracy rapidly when the depth reaches certain level [33]. He et al. effectively solved the problems of gradient explosion / vanishing and accuracy degradation through the use of a Residual Network (ResNet) framework [12]. In this framework, they introduced the concept of residual building block with a shortcut connection, as shown in Figure 1, guaranteeing the deeper version of a neural network performs no worse than the shallow counterparts. ResNet assemble won ILSVRC 2015 in image classification, detection, and localization. In that competition, ResNet archived a validation top-5 error rate of 3.57%, which surpasses the average human classification error. Given this performance, we base our model on ResNet as described in Section III-B.

III. Approach

As shown in Figure 2, the proposed system consists of two applications deployed within an Android platform: a Pydroid[4] application which includes the pre-trained ResNet-50-based model and a Java application (the typical Android programming language) which handles image processing, interfacing with the PWC, and input from the user. The Java application accesses the necessary Android sensors (camera and wireless interface) through the Android platform API.

Flow from the PWC's environment through the pre-trained model to the PWC can be seen in Figure 2. Starting from the top right, images of the environment are captured in real-time by the Java app via the Android device's physical camera. The app then performs the necessary cropping (Section III-A) and downsampling for the model before it is relayed to the DNN model. Our model is implemented in Python, so we utilize Pydroid to facilitate a Python environment on the Android system. The Pydroid application feeds the data to the model which produces a direction (left, right, forward, stop) based on its training. This direction is then relayed back to the Java app and signaled to the PWC through an implementation of a generic device interface for the specific PWC the app is configured to communicate with. Communication is done through the Android API using the appropriate sensor for the PWC. As a proof of concept, our experiments are conducted using WiFi to connect to a REST API implemented on a Pioneer 3-AT robot[5].

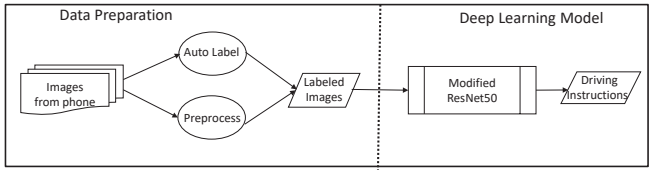

Fig. 3. Vision-based assistive driving model based on ResNet-50

Figure 3 shows the derivation of driving directions by training a DNN 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 stop moving.

[4]https://pydroid-3-ide-for-python-3.en.softonic.com/android

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

TABLE I
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%

A. Data Collection and Pre-processing

Images taken by a mobile phone 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 in 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 portion of the images may affect the model's accuracy.

Table I shows the results for three different image cropping strategies over models of different sizes that are based on ResNet-50 (Section III-B. The cropping strategies include the full-size image, only the lower half of the image, and only the lower quarter of the image. The experiment results show that the use of the lower-half of images produces the best training 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 results of the experiment, all images are cropped in half for our approach; only the lower half is kept for both model training and actual operation, as shown in Figure 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 keep this ratio to ensure that the deep learning model produces a high accuracy result. This pre-processing step significantly improves the accuracy of the deep learning model. We believe this is likely because the edges of corridors (i.e., where the floor meets the wall) are the strongest indicators of orientation.

Fig. 4. Image Cropping

The collected images are labeled based on the directions the phone camera faces with respect to its surroundings. Specifically, if the camera faces to the left side of the corridor, the collected images are labeled as "right", meaning the PWC would need to turn right to avoid colliding with the wall; if the phone faces to the right side the corridor, the collected images are labeled "left", meaning the PWC needs to turn right to avoid colliding with the wall; if the phone faces to the direction parallel to the corridor, the collected images are labeled as "forward", meaning PWC can move forward without any direction changes; if there are any obstacles or stairs in front of the phone, the images are labeled as "stop" meaning the PWC needs to stop to avoid collisions. Figure 5 demonstrates some examples of these labels.

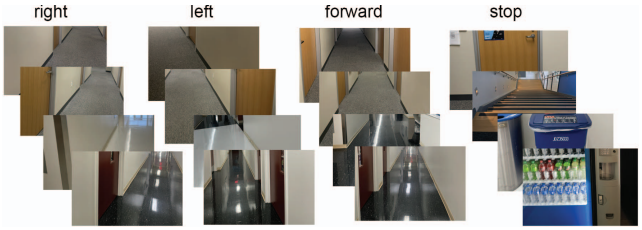

Fig. 5. Labeled Training Images

For this proof of concept, approximately 10,000 images were collected for training and testing. 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 better demonstrate the generalizability of the system.

B. The Deep Learning Model

Based on our background research on image classification models (Section II-B), we chose the ResNet architecture as a basis for our model. Unlike the previous deeper DNN models where layers are simply stacked, a residual network consists of several building blocks allowing for more layers in the DNN without a loss of accuracy. Our deep learning model is based on the 50-layer ResNet and 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 1x1 convolution layer, one 3x3 convolution layer, and one 1x1 convolution layer. Ultimately, there are a total of 50 layers, including the final output layer.

ResNet was originally designed to classify images into 1,000 categories; therefore the final output layer consists of 1,000 fully-connected nodes. We modify the ResNet-50 model by removing the output layer and adding a global average pooling 2D layer and a fully-connected layer with four nodes as the final prediction layer, corresponding to "left", "right", "forward", and "stop".

With 10,000 training images (80% in the training set and 20% in the test set), the training with 50 epochs takes about

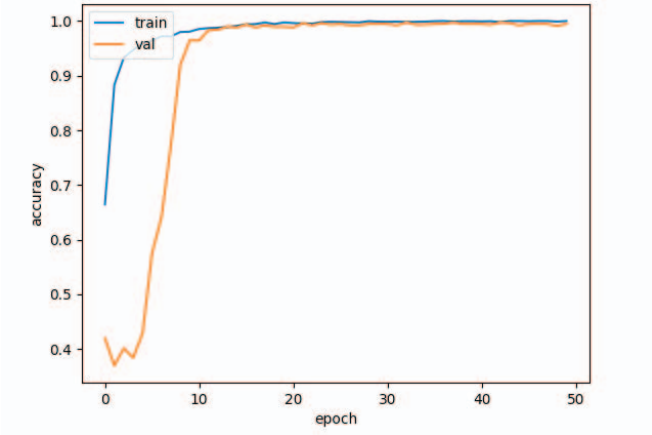

Fig. 6. Model Accuracy

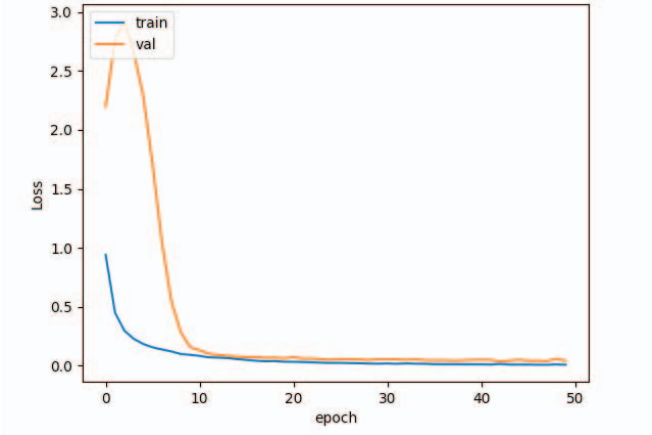

Fig. 7. Model Loss

TABLE II
MODEL PERFORMANCE

Model Layers	Accuracy	Second per Image
50 Layers	99.7%	0.51
40 Layers	99.87%	0.41
22 Layers	98.8%	0.29
10 Layers	96.45%	0.18
2 Layers	74.87%	0.11

two hours on a computer with an NVIDIA V100 GPU. With a learning rate of 0.001, the test accuracy reached 99.7%.

For our purposes, the ResNet-50-based model achieved an accuracy of 99.7%. However, it took an average of 0.5 seconds to process an image on a Pixel6 phone to generate a driving instruction. To try to achieve a better balance of speed and accuracy, we built four more models also based on ResNet-50: attaching the two new layers to the output of stage 4 to form a 40-layer model; attaching the two new layers to the out 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. The accuracy and image processing speed of each model are shown in Table II. We considered the 22-layer model to have the best balance of accuracy and speed at 98.8% and 0.29 seconds respectively. As a result, the system would be able to process one more image per second with minimal cost to accuracy. Furthermore, it is likely that increasing the number of training images would further improve accuracy and thus the performance of the system.

Figure 6 shows the training accuracy vs. test accuracy over the number of epochs, and Figure 7 shows the training loss vs. test loss over the number of epochs, for our selected 22-layer model. There is no obvious overfitting in the training process, except the test loss increases slightly at epoch 43. Therefore, our final target model is trained for 43 epochs with an accuracy of 98.8%, as shown in Table II.

C. Model Deployment to Android Phone

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

We have designed an Android app that takes the images at a fixed rate of ten images per second and saves the images to the phone's filesystem. While our model only handles up to three images per second, the Android app is able to accommodate future improvements to performance. The pre-trained deep learning model is deployed to the phone via a Pydroid3 app that provides a full Python environment with all necessary packages such as Tensorflow, Keras, OpenCV, etc. The Python code runs in parallel with the Android app, taking the images saved by the app to the pre-trained model to produce the driving instructions such as turning left, turning right, moving forward, and stopping. The instructions are sent to the robot via the RESTful API.

D. Safety

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 is a speed limit for Powered wheelchairs in most countries. For example, PWCs must not travel faster than 4mph (1.79 m/s) in the UK[6]; in the USA, most of PWCs cannot go faster than 5mph (2.24 m/s)[7]. With a process rate of 3 images/second, the system could theoretically react to obstacles as close as 0.60 and 0.75 meters, respectively. Our model is trained to recognize an obstacle within two meters. The process rate is more than enough to guarantee that a PWC will receive a "stop" instruction in front of an obstacle if the PWC travel is at top speed, which is very unlikely in an indoor environment.

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

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

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.

IV. 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, which has a 2.8GHz Octa-core CPU and a Mali-G78 MP20 GPU with 8GB memory. The Pioneer 3-AT mobile robot has three 12V/9Ah batteries that can power the robot to reach speeds up to 0.8 m/s. To simulate the normal operation speed of wheelchairs for in-door spaces, where the regulation permits to run up to 4mph (1.79 m/s) in an outdoor environment[8], we adjust the control of the mobile robot to make it run at a speed up to 0.7 m/s.

A. Test Cases and Hands-free Driving

To answer **RQ1**, we considered seven different test cases for the assistive-driving application described in Section III-C 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 III.

- **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 seconds to reach the spot about 3 meters before the corridor end. Here, the robot has the safety distance of 3 meters 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 controled 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 seconds to reach the same 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.

TABLE III
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

- **Test Case 3**: This case used the same parameters as in Test Case 2 except that a stationary obstacle (trash can) was positioned at the middle of the corridor, 10 meters 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 of the application without the user's manual operation. It took 27 seconds for the robot to reach the target destination before the corridor's end, where the extra 3 seconds 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 meters. 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 seconds 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 seconds. 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 meters 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 seconds in total in this case where

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

the user was called to attention for 7 seconds with 5 seconds 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 seconds 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 seconds.

B. 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 meters 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 seconds. The app will issue a warning message if there is no user interaction for more than 30 seconds, 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.

V. Discussion

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

A. 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 decision, 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) combine the training data set with the pre-trained model with the new data set to retrain the model; 2) use only the new data set to retrain the model with a much smaller learning rate; 3) create an ensemble model [9] 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.

B. 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 while assisting the operation of a PWC are also cropped and discarded immediately after the model produces the driving instructions.

VI. Conclusions and Future Work

This paper describes an affordable, vision-based assistive driving system which has the potential to be used in power wheelchairs to assist those with mobility impairments. We have demonstrated a proof of concept by applying our deep learning model to a robot to successfully navigate corridors in buildings of a university. We demonstrate that a mobile app-based system could provide a highly accessible, low-cost solution to assist users to operate PWCs in a hands-free and attention-free manner.

As a continuation of this work, we first plan to implement different versions of the app-PWC interface for existing PWCs. We will also work to incorporate incremental learning utilizing the data captured as the PWC user overrides the model decisions as discussed in Section V-A. Effectively, any signal sent by the user to contradict the system's decision serves as a ground truth. By collecting such data, we may improve the model through techniques such as online and / or reinforcement learning. 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 to improve the system's training and user experience.

VII. Acknowledgment

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] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[2] S.-H. Chen, Y.-L. Chen, Y.-H. Chiou, J.-C. Tsai, and 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)*, volume 2, pages 1587–1589. IEEE, 2003.

[3] R. De Man, G. J. Gang, X. Li, and 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*, volume 11072, pages 239–243. SPIE, 2019.

[4] D. Dominic, S. Chhawri, R. M. Eustice, D. Ma, and A. Weimerskirch. Risk assessment for cooperative automated driving. In *Proceedings of the 2nd ACM workshop on cyber-physical systems security and privacy*, pages 47–58, 2016.

[5] M. Fezari and M. Bousbia-Salah. Speech and sensor in guiding an electric wheelchair. *Automatic Control and Computer Sciences*, 41(1):39–43, 2007.

[6] A. O. Frank, L. H. D. Souza, J. L. Frank, and C. Neophytou. The pain experiences of powered wheelchair users. *Disability and Rehabilitation*, 34(9):770–778, 2012. PMID: 22013954.

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

[8] G. Fusco and 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.

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

[10] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

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

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] 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. *arXiv preprint arXiv:1504.01716*, 2015.

[14] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo. Development of autonomous car—part ii: A case study on the implementation of an autonomous driving system based on distributed architecture. *IEEE Transactions on Industrial Electronics*, 62(8):5119–5132, 2015.

[15] A. G. Karkar, S. Al-Maadeed, J. Kunhoth, and A. Bouridane. Camnav: a computer-vision indoor navigation system. *The Journal of Supercomputing*, 77(7):7737–7756, 2021.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[17] J. Kulhánek, E. Derner, T. De Bruin, and R. Babuška. Vision-based navigation using deep reinforcement learning. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–8. IEEE, 2019.

[18] M. Kutbi, Y. Chang, and 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.

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

[20] N. D. Lane, P. Georgiev, and 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*, pages 283–294, 2015.

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[22] J.-W. Lee, S.-U. Choi, Y.-J. Lee, and 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)*, pages 38–41, 2001.

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

[24] B. Li, J. P. Muñoz, X. Rong, Q. Chen, J. Xiao, Y. Tian, A. Arditi, and M. Yousuf. Vision-based mobile indoor assistive navigation aid for blind people. *IEEE Transactions on Mobile Computing*, 18(3):702–714, 2019.

[25] R. Lipskin. An evaluation program for powered wheelchair control systems. *Bull. Prosth. Res*, 6:121–219, 1970.

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

[27] C. Manlises, A. Yumang, M. Marcelo, A. Adriano, and 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)*, pages 481–484, 2016.

[28] C. Manlises, A. Yumang, M. Marcelo, A. Adriano, and 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)*, pages 481–484. IEEE, 2016.

[29] A. Mathur, N. D. Lane, S. Bhattacharya, A. Boran, C. Forlivesi, and 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*, pages 68–81, 2017.

[30] I. Ohya, A. Kosaka, and A. Kak. Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing. *IEEE transactions on robotics and automation*, 14(6):969–978, 1998.

[31] T. Sato and 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)*, pages 17153–17162, June 2022.

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

[33] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks, 2015.

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[35] Taş, S. Hörmann, B. Schäufele, and F. Kuhnt. Automated vehicle system architecture with performance assessment. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2017.

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