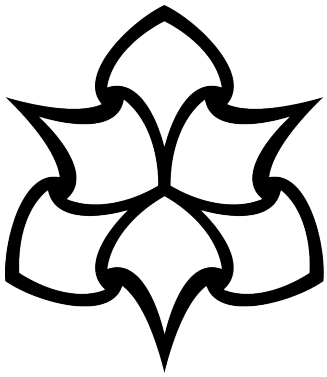

Pedestrian Recognition and Obstacle Avoidance for Autonomous Vehicles using Raspberry Pi

Charlie Day [1], Liam McEachen [1], Asiya Khan [1*], Sanjay Sharma[1] and Giovanni Masala[2]

[1] School of Engineering
[2] School of Computing, Electronics and Mathematics, University of Plymouth, Plymouth PL4 8AA UK
*asiya.khan@plymouth.ac.uk

Abstract. The aim of this paper is twofold: firstly, to use ultrasonic sensors to detect obstacles and secondly to present a comparison of machine learning and deep learning algorithms for pedestrian recognition in an autonomous vehicle. A mobility scooter was modified to be fully autonomous using Raspberry Pi 3 as a controller. Pedestrians were initially simulated by card board boxes and further replaced by a pedestrian. A mobility scooter was disassembled and connected to Raspberry Pi 3 with ultrasonic sensors and a camera. Two computer vision algorithms of histogram of oriented gradients (HOG) descriptors and Haar-classifiers were trained and tested for pedestrian recognition and compared to deep learning using the single shot detection method. The ultrasonic sensors were tested for time delay for obstacle avoidance and were found to be reliable at ranges between 100cm and 500cm at small angles from the acoustic axis, and at delay periods over two seconds. HOG descriptor was found to be a superior algorithm for detecting pedestrians compared to Haar-classifier with an accuracy of around 83%, whereas, deep learning outperformed both with an accuracy of around 88%. The work presented here will enable further tests on the autonomous vehicle to collect meaningful data for management of vehicular cloud.

Keywords: Pedestrian recognition, Obstacle avoidance, Ultrasonic sensors, Haar classifier, HOG descriptor, deep learning

1 Introduction

According to a recent report from the Department of Transport, from October 2015 to September 2016, there were around £183,000 casualties resulting from traffic accidents of which 1,800 were fatal and over 25,000 were life changing [1]. The vision for the autonomous car revolution is to reduce this figure by at least 76%. Cars are undergoing a revolution just like mobile phones did twelve years ago. They are increasingly becoming intelligent agents that have the capability to learn from their environment and be driven in an autonomous manner. Therefore, to improve road safety and traffic congestion, fully or partially autonomous vehicles offer a very

promising solution. Most modern vehicles are equipped with advanced driver assistance systems (ADAS) that assists in driving in a number of ways such as lane keeping support, automatic parking, etc. More recently, traffic sign recognition systems are becoming an integral part of ADAS.

Most new vehicles are capable of some form of autonomy e.g. automatic parking, lane recognition, etc. Raspberry Pi 3 is a small low cost computer and offers opportunity for research towards the roadmap of autonomy. Therefore, the objective of this paper is to use Raspberry Pi version 3 as a microcontroller for an autonomous vehicle connected with ultrasonic sensors and a camera and discusses the suitability of using Raspberry Pi as a controller. The mobility scooter was acquired from Betterlife healthcare [19] and adapted such that its controller communicated with the Pi which was connected to ultrasonic sensors and a camera. Ultrasonic sensors are cheap and have less power consumption and measures the accurate distance from the obstacle and transmits the measured data to the system. The ultrasonic sensor are connected to the Raspberry Pi in a way so that obstacles present in the front, back and side of the vehicle are being detected. The obstacles in the blind zone are also detected by the ultrasonic sensors. This emulates obstacle detection on the road. Presence of pedestrians is detected by computer vision using Haar-classifier and HOG descriptors and further by deep learning.

Therefore, the contributions of the paper are two-fold:

- To convert a mobility scooter to be fully autonomous with ultrasonic sensors and camera to detect an obstacle and find the reliable range.
- To present comparative analysis between HOG descriptors, Haar-classifiers and deep learning for pedestrian recognition.

The rest of the paper is organised as follows. Section 2 presents related work; Section 3 presents the conversion of the mobility scooter to an autonomous vehicle. In Section 4, the computer vision and deep learning algorithm implementation on Raspberry Pi 3 is presented. Section 5 presents the experiments, results and discussions. Section 6 concludes the paper highlighting areas of future work.

2 Related Work

An important feature of autonomous driving is recognizing pedestrians and obstacles. Computer vision allows autonomous vehicles to process detailed information about images that would not be possible with only sensors and has been increasingly used to study facial recognition. Two methods in computer vision have been widely applied in image recognition as firstly, the Haar feature like classifier [2] where a set of positive and negative images are used and Haar-like features are applied to each set. Critical analysis of the technique has shown that the background complexity plays a role in the quality of the classifier and can be easily corrupted by lighting [3]. This method was originally used for facial recognition, the same principles can be applied to almost any other object therefore a pedestrian cascade can be made using this method and has been presented here. Secondly, the Histogram of Oriented Gradients (HOG) descriptor and has been used in [4] for pedestrian

recognition. The principles behind the HOG descriptor [5] is that it is a type of 'feature descriptor' that has been trained using a Support Vector Machine (SVM), a type of supervised machine learning that works on the classification of positive and negatives samples of data. The HOG feature descriptor, unlike conventional techniques applies a general feature as opposed to a localized one to the image area. This is sent to the SVM which would then classify it as a pedestrian.

The authors in [6] present an efficient hardware architecture based on an improved HOG version and linear SVM classifier for pedestrian detection for full high definition video with a reduced hardware resource and power consumption. Image processing algorithms have been used in [7] to remove unwanted noise from the image based sensor. A vision optical flow based vehicle collision warning system is proposed in [8] based on computer vision techniques.

Raspberry Pi is a credit card sized single board low cost computer and provides the flexibility of using it as a microcontroller and is increasingly used in academic research, e.g. in [7] authors present the implementation of image processing operations on Raspberry Pi. In [9] ultrasonic sensors are used for object detection from the moving vehicle. In [10] authors have combined Haar detection with laser distance to recognize pedestrians. The work presented in [11] concludes that codebook representation of Haar and HoG features outperform detection based on only HoG and Haar. The codebook is generated from a set of features given by the Bag of Words [12] model. More recently, deep learning based on deep convolutional neural networks has been used for image classification of road signs [13] with 97% accuracy and for pedestrian recognition [14]. The work presented in [15] uses detection based on Haar features and classification based on HOG features with support vector machine. Pedestrian recognition using OpenCV on Raspberry Pi was implemented in [16]. In [17] authors have used Raspberry Pi for detecting road traffic signs using image processing techniques, whereas, in [18] a combination of MATLAB with Raspberry Pi is used for face detection using Haar classifier.

There has been an increasing interest from the research community in image classification for pedestrian recognition. Most modern vehicles now have some form autonomous features, confidence level in obstacle detection and pedestrian recognition has to increase towards the roadmap of fully autonomous vehicles. In addition, utilizing the computational capability of Raspberry Pi in research is still in its early stages. The novelty of our work from the work presented in literature is that we are implementing our pedestrian recognition algorithms on Raspberry Pi 3, comparing them and testing its suitability from a research point of view.

3 Autonomous Vehicle using Raspberry Pi

The vehicle used for this project is a Capricorn Electric Wheelchair from Betterlife Healthcare [19] as shown in Fig. 1(a). It is a small, four wheeled vehicle with caster type front wheels, two fixed driven rear wheels and powered by two 12V batteries. It is driven by two separate electric motors, which are connected directly to each of the rear wheels. It has a maximum speed of 4mph, a maximum incline of 6° and a turning circle of radius 475mm. The maximum range of the wheelchair is 9.5km. The tyres

are solid and have a larger radius than many other models of its type, helping to improve performance on rough or uneven surfaces.

This section will present the conversion of the mobility scooter into an autonomous vehicle controlled by Raspberry Pi 3. It will further describe the connection of ultrasonic sensors and camera.

3.1 Connecting the Raspberry Pi 3

The autonomous vehicle was built from a mobility scooter as shown in Fig. 1(a). The scooter had an inbuilt microcontroller shown in Fig. 1(b) which was used as a communicative tool between the Raspberry Pi version 3 and the vehicle's motors. The directional actions of the joystick voltage levels for each pin are shown in Fig. 1(c) (shown in the red circle in Fig. 1(b)) and are presented in Table 1.

Fig. 1(a). Original mobility scooter

Fig. 1(b). Autonomous vehicle microcontroller

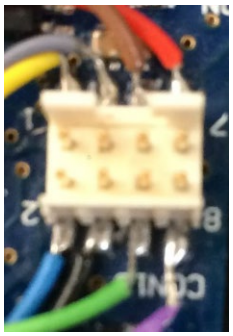

Fig. 1(c). Autonomous vehicle pins

Table 1. Voltage directional values.

Direction	Voltage applied (volts)	Pin colour
Forward	3.97	Green/Grey
Reverse	1.13	Green/Grey
Right	3.97	Purple/Yellow
Left	1.13	Purple/Yellow
Static/Stop	2.5	All
Turn ON	2.5	Black

In order to make space for a platform on which the system can be installed, the chair was removed, as was the housing surrounding the frame of the vehicle. The central column between the chair and the frame was also removed, allowing the new chassis to be placed over the frame. The new chassis is shown in Fig. 2(a), whereas, the block diagram is presented in Fig. 2(b) showing the connections of the Raspberry Pi with the sensors and the vehicle's controller. The chassis shown in Fig. 2(a) has

enough space for the control panel – rewired to connect the Raspberry Pi directly to the joystick input, the Pi itself, and two breadboards with which the circuitry could be modified during the built and testing process. The chassis is designed so additional components and sensors can be added. Two digital to analogue converters were installed, controlling both forwards/backwards motion and the yaw of the vehicle, respectively. The front wheels were fixed in place by the removal of the bearings contained in the shafts. This allowed the connecting bolts to be tightened fully and restricting the motion of the vehicle to forwards and backwards.

Fig. 2(a). The autonomous vehicle modified from the mobility scooter

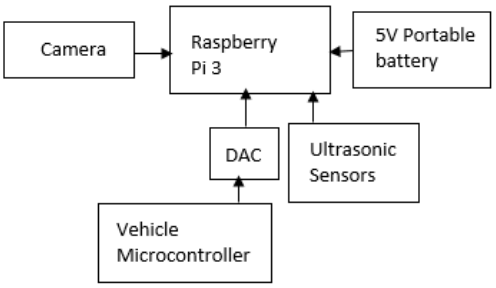

Fig. 2(b). Block diagram of the autonomous vehicle

For the vehicle to be autonomous it would have to be controlled by the General-Purpose Input Output (GPIO) pins on the Pi which would send signals emulating the joystick. The GPIO pins work with digital signals therefore a Digital to Analogue Converter (DAC) (Fig. 2(b)) would be required to alter the signal type. An Adafruit MCP4725 DAC [20] was used and functioned well with the Raspberry Pi.

To use the DAC effectively the Inter-Integrated Circuit (I^2C) bus on the Raspberry Pi had to be operated. This is an interface that utilises data exchange between components and microcontrollers, there are many 'slave' devices (the DAC/s) controlled by a 'master' device (the Raspberry Pi). The Pi's GPIO pins can only output 5V or 3.3V and the DAC is a 12-bit controller which means it ranges in values from 0 to 4096, Equation 1 shows the formula used,

$$DAC_{voltage} = \left(\frac{V_{desired}}{V_{max}}\right) X 2^{12} \quad (1)$$

This formula can then be applied directly to the DAC through a Python script. The DAC contains two inputs, V_{dd} and A_0 allowing two different I^2C bus addresses to be used on the Raspberry Pi at once which could then be referred to as parameters in a function as *0x62* and *0x63* for forwards/backwards and left/right, respectively.

3.2 Connecting Ultrasonic Sensors to the Raspberry Pi 3

Ultrasonic sensors provide basic object-detection autonomy to the vehicle. The HC-SR04 sensor was used which could work with the Pi's GPIO pins through jumper wires. The principle of the HC-SR04 is that there are four pins: power, trigger, echo, and ground. The power and ground were connected directly to the Pi's voltage and ground pins, the trigger acts as a 'starting gun' for the sensor signifying when to produce a soundwave and the echo receives the soundwave. While these are binary

input/output functions they can be used on Python to determine the distance of the closest object. The programming logic is shown in Flowchart in Fig. 3**.**

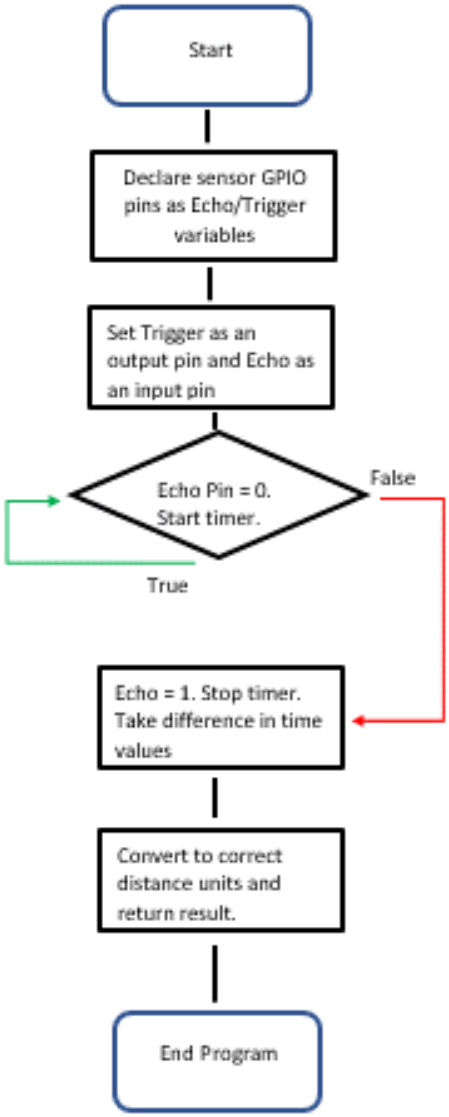

Fig. 3. Ultrasonic sensor function flowchart

This ultrasonic object detection function can be imported into the DAC script and tell the Pi to fire a STOP (2.5V) or 'Reverse' (1.13V) command to the vehicle through the DAC if the sensor function detects an object that is below a predefined threshold (e.g. < 2m) as shown in the flowchart in Fig. 4.

The script was modified for multiple sensors to incorporate multiple echo variables, however, a single trigger can be used to activate them all simultaneously and add each echo onto an array. Python has clocking functions which enable a user to determine the length of time between two points in a function, this is important to determine the latency of sensors and reaction time of the autonomous vehicle. A voltage divider was used to protect the Pi's GPIO pins from excessive signals generated by the ultrasonic sensors.

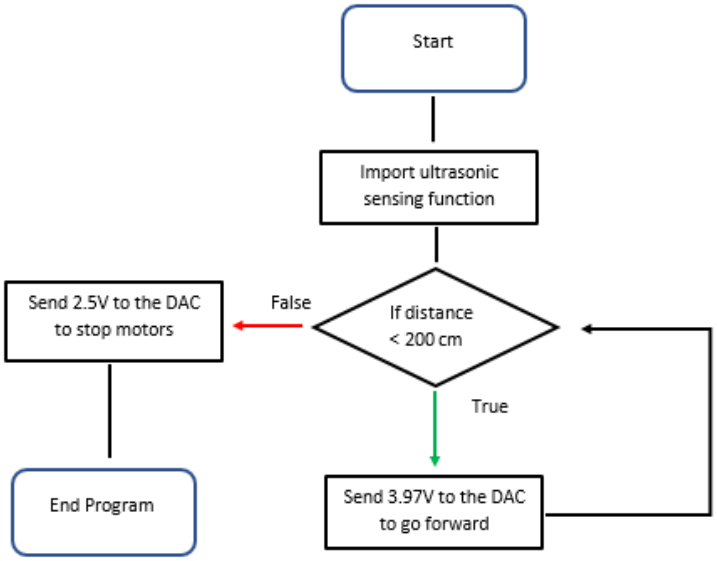

Fig. 4. Object detection function flowchart

To ensure the vehicle was mobile, a portable battery producing 5V was powering the Pi and a laser-cut housing was designed to hold the system in place and absorb forces from potential collisions (Fig. 2(b)). It is recognised that this methodology of using an autonomous vehicle is limited in that it is not the size of an autonomous commercial vehicle however the value comes from testing the principles.

In order to connect the ultrasonic sensors to the Raspberry Pi, the circuit was adjusted so that all of the triggers were controlled by the same i/o pin on the Pi, keeping the amount of i/o pins used to a minimum of x+1, where x is the number of sensors used in the design. Each of the sensors outputs the result to an array which is continually updated, and it is this array which can be communicated to an infrastructure hub.

4 Computer Vision and Deep Learning on Raspberry Pi

Computer Vision on the Raspberry Pi was run by importing the OpenCV (Open Source Computer Vision) [21] module through a stream. Open-CV is a library by Intel that consists of functions used for computer vision. The digital image is processed pixel by pixel, applying convolution and smoothing through a local operator, which operates threshold on the histogram. Python programming language has been used in this paper.

A pre-trained pedestrian Haar-classifier [2] was used where a cascade function is trained from a number of positive and negative images. This is based on the detection stage that generates the rectangular regions which may contain pedestrians. These regions of interest (ROI) are then classified and confirms the presence of a pedestrian or discard the ROI if the classification is negative. The system output consists of bounding boxes describing size and position of the pedestrians in each frame.

Computer Vision was run on the Raspberry Pi by importing the OpenCV module and run through a stream. A pre-trained pedestrian Haar-classifier was used that contains 19 Stages of Haar-training [21]. This included 5000 positive images and 3000 negative images [22].

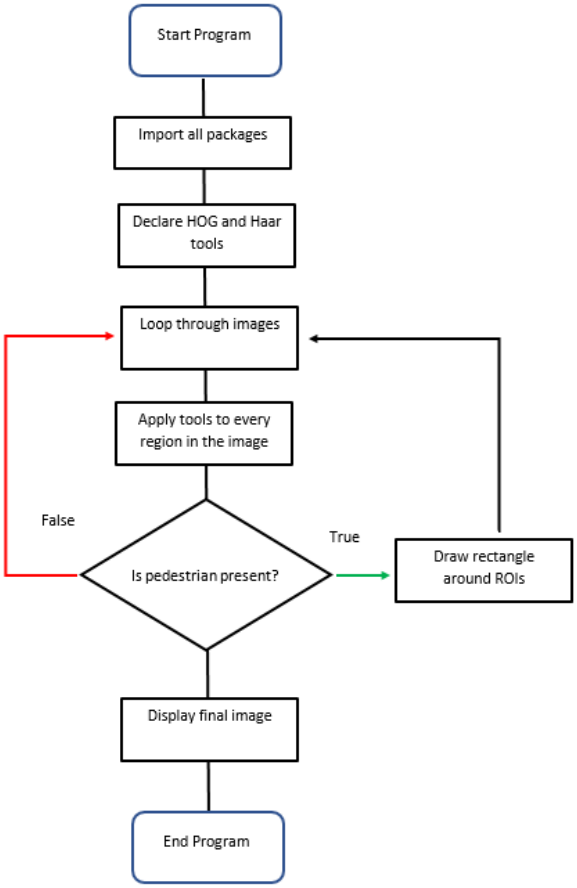

Fig. 5. Still images comparison flowchart

The Haar-classification uses an integral method that takes each rectangle as the sum of pixels and reduces them to an array of four values massively decreasing the time for a detection to take place. The Adaboost (Adaptive Boosting) machine learning algorithm [23] is used to sum all weak classifiers into a final strong classifier which is then applied to an image.The video stream from the Pi camera was parsed through this cascade and ROI returned a 4-index array which was used to determine the presence of a pedestrian. The Pi's Frames Per Second (FPS) on the camera was low even without running an algorithm for each frame, therefore a method called

threading was implemented to increase the FPS allowing for faster reaction times from the Pi. Threading (or Multi-threading) is the method of having processes run parallel independently with one another but transferring data, therefore instead of one main thread, there are multiple threads which while require more computational resources allow an increase in process quality, such as passing frames per second. The programming logic to execute the computer vision algorithms for still images and frames from a video stream are shown in Fig. 5 and 6, respectively.

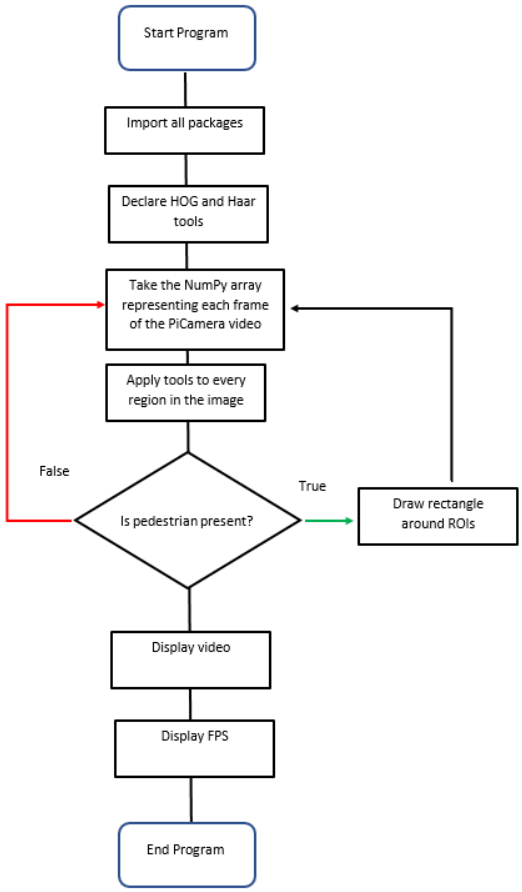

Fig. 6. Raspberry Pi 3 video comparison flowchart

Deep learning was applied with single shot detection method [21] based on a single neural network. The method [21] discretises the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. The scores are generated by the network at prediction time for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. In addition, predictions from multiple feature maps are combined by the network with different resolutions to naturally handle objects of

various sizes. The deep learning algorithm was trained with COCO and VOCO712 datasets [24] with a total of 82,943 images.

5 Experiments, Results and Discussions

The experiments comprised of two parts. The first part tests the hardware and software of the autonomous vehicle. This will include the evaluation of the vehicle's stopping properties on the Raspberry Pi and sensor connections along with the Python script to recognise obstacles and react accordingly. This has been tested by laying out an obstacle for the autonomous vehicle and sending it from two arbitrary points while avoiding solid objects which would translate in the real world as parked cars or pedestrians. The metrics used as results of this test are stopping distance, speed of recognition, accuracy, sensor latency and sensor clock of the ultrasonic sensors.

The second part of the investigation examines the quality of computer vision methods used for recognising pedestrians, the Haar Classifier, HOG Descriptor and deep learning, and their detection rates recorded. The computer vision methods will then be transferred onto the Raspberry Pi 3 and test distances will be run to measure the quality of the frames per second (FPS) on the Pi running the algorithms.
Sub-sections 5.1, 5.2 and 5.3 presents the ultrasonic sensor testing, Section 5.4 presents pedestrian recognition testing using Haar-classifier, HOG descriptors and deep learning on the vehicle, whereas, discussion of results is presented in Section 5.5.

5.1 Sensor Testing for Stopping Distance

The first test was performed with two different scenarios – a cardboard box and 'real' pedestrian. The dimensions of the cardboard box were 29cmx21cmx8cm. The vehicle is programmed to run at five different speeds as 0.2, 1.6, 2.4, 3.2 and 4 mph (maximum speed) and was tested for these speeds.

The two sets of data retrieved from the first test are shown in Fig. 7 and 8 at different speeds. Fig. 7 shows the results with simulated pedestrians (cardboard boxes), whereas, in Fig. 8, the cardboard boxes were replaced by a pedestrian. The black trend line signifies the minimum stopping distance showing that 7 tests resulted in a collision. These collisions occurred at higher speeds and lower ultrasonic sensor distances however for distances above 150cm all speeds have a successful object-detection result and no collisions were recorded.

To simulate real-world environments more effectively the cardboard box was replaced with a pedestrian. The results from the box and the pedestrian were compared. Analysis of results show that the data points remain in a similar position as shown in Fig. 7; however, as speeds increase the performance marginally decreased, specifically, at 200cm in the box test where it passed although results show a collision in the pedestrian test.

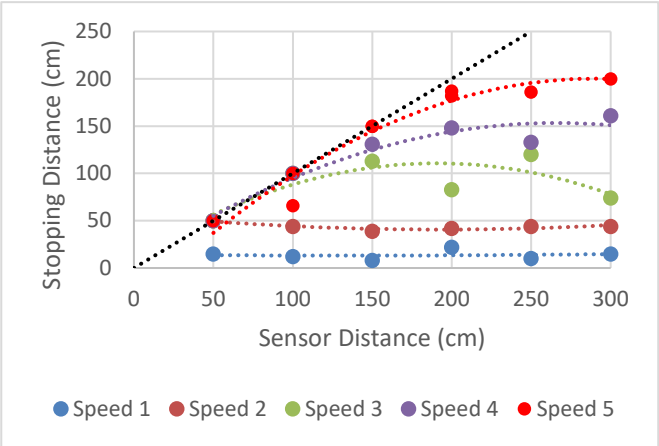

Fig. 7. Simulated pedestrians (rectangular boxes) stopping distances at different speeds of the autonomous vehicle

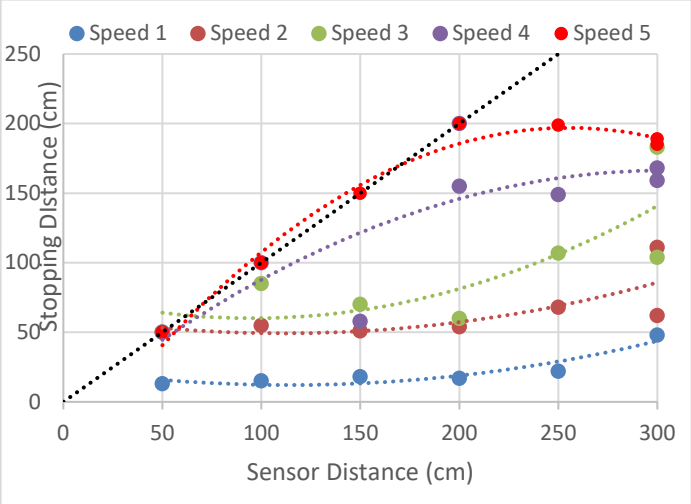

Fig. 8. Pedestrian stopping distances at different speeds of the autonomous vehicle

An additional test was passed on the final distance (300cm) for each speed, where the pedestrian stood front facing with their legs apart in contrast to a side facing pose (to simulate crossing the street) which demonstrated to no difference.

It can be concluded from Fig. 7 and 8 that the ultrasonic sensors combined in the autonomous vehicle working at practical speeds would work sufficiently at 100cm but for optimum use, 250cm would be the ideal distance range.

5.2 Sensor Latency Testing

Sensor latency directly affects the response time of the whole system and therefore was chosen as a parameter for testing. Sensor latency was tested according to the

flowchart of Fig. 9 where the program was allowed to run with the minimum allowable distance (MAD) set so that the forward sensor would not cause the program to terminate early. Initially, the test was run 'as-is' without changing any of the code. This was in order to obtain a baseline cycle time for comparison. Once the program had been running for approximately 3 minutes, the program was stopped by placing an obstacle in front of the forward sensor below the MAD. Initially, the main control panel attached to the motors was kept off for the duration of the experiment. This is because the Pi output signals via the input/output pins regardless of whether the board on the other end of the circuit is powered, this was not considered to be a potential factor in how long the sensors take to output data.

To set up the test, the vehicle was placed in the middle of the testing area, with obstacles placed at known distances from each of the four sensors locations as shown in Table 2. Table 2 describes the details of the tests conducted at four distances respective to the four ultrasonic sensors connected in four locations as Forward, Starboard, Aft and Port. This was not only to establish whether the cycle time had an effect on the accuracy of the ultrasonic sensors, but also to allow further analysis on the accuracy of the sensors at a variety of ranges.

Table 2. Obstacle distances.

Sensor location	Measured distances (cm)		
	Tests 1.1-1.3	Tests 2.1-2.2	Tests 3.1-3.5
Forward	120	200	350
Starboard	80	40	10
Aft	100	150	150
Port	50	30	20

The baseline cycle time was 2.58s. The cycle time was reduced for forward statement in the code. The default setting was at 0.5s. For each test it was decreased to 0.1s and then increased to 1s in order to test see the effect of changing known factors in the code.

Further, the cycle time was reduced for all of the four sensors as calling all four sensors adds a delay to the program due to the sleep time of 0.5s which allows the sensors to settle and receive the echo from any detected obstacle. This is not desirable from a design point of view since it introduces an unnecessary source of inconsistency into the readable results, especially if the sensors have an inherent inconsistency cycle to cycle. Fig. 10 shows the average cycle time for each test.

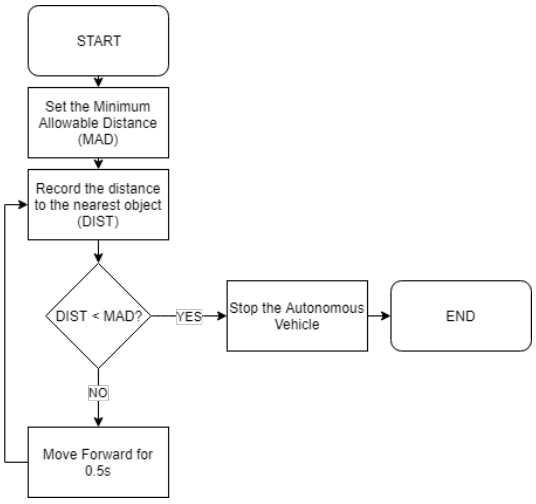

Fig. 9. Flowchart for sensor latency testing

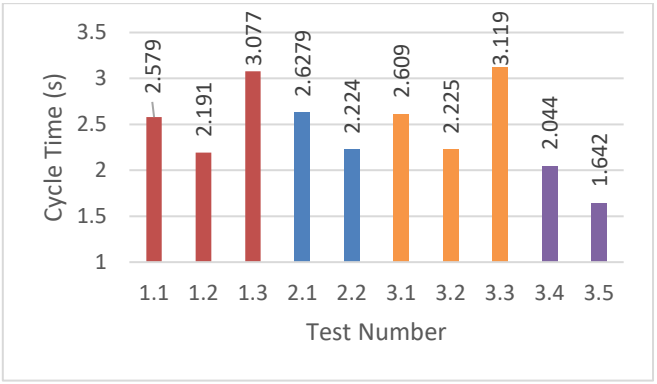

Fig. 10. Cycle time by test number

In Fig. 10 the red, blue and green bars show the cycle time for a series of different distances, none of which seem to have an appreciable effect on the cycle time. Each distance runs the forward statement for 0.5s, 0.1s and 1s, respectively from left to right. Tests 1.1-1.3, 2.1-2.2 and 3.1-3.5 were taken at the same distances as shown in Table 2. This has made a large difference to the cycle time of the program and would improve the response time of the program, thus improving the stopping distance.

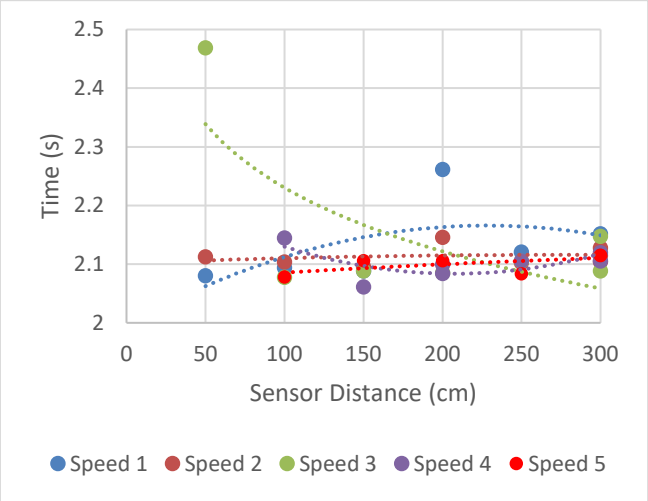

Fig. 11. Program clock time verses sensor distance

5.3 Sensor Clock Testing

The test measuring the clock time of the Python program is shown in Fig. 11 and shows the time taken from the sensor detecting that the vehicle is too close (under the distance threshold) to a complete stop function from the Python script at varying vehicle speeds. Immediate analysis of the data shows that there are two outliers (Speed 3 at 50cm and Speed 1 at 200cm), possible causes for these anomalies are sensor failures due to inconsistent connections which is the most likely reason. Trend lines show that the speed of the vehicle and sensor distance have negligible effects on the speed of the program, they are all generally inline between 2.05 and 2.15 seconds.

Five data points appear to be above the ideal trend line indicating that the system is stopping at a sensor value greater than the threshold suggesting a sensor error. This could potentially have been caused by overlapping soundwaves sent from previous triggers.

5.4 Pedestrian Recognition

The Haar-classifier and HOG descriptor were compared against one another initially to determine the more suitable computer vision algorithm for pedestrian detection. All images were converted to 300 pixels wide (with aspect ratios remaining constant) to ensure integrity of results. An example of the images with the ROI is shown in Fig. 12 (left, right and bottom). For the experimental results, confirmed bounding boxes would only be a success if they identified pedestrians, therefore an identification of a cyclist or a dog would be counted as a false positive (as they should have their own classifiers/descriptors). These methods were compared against further twelve images taken from a number of webpages and can be found in [25]. The results from Haar-classifier and HOG descriptors were then compared to deep learning on the same twelve images using the Single Shot Detection method [26] described earlier.

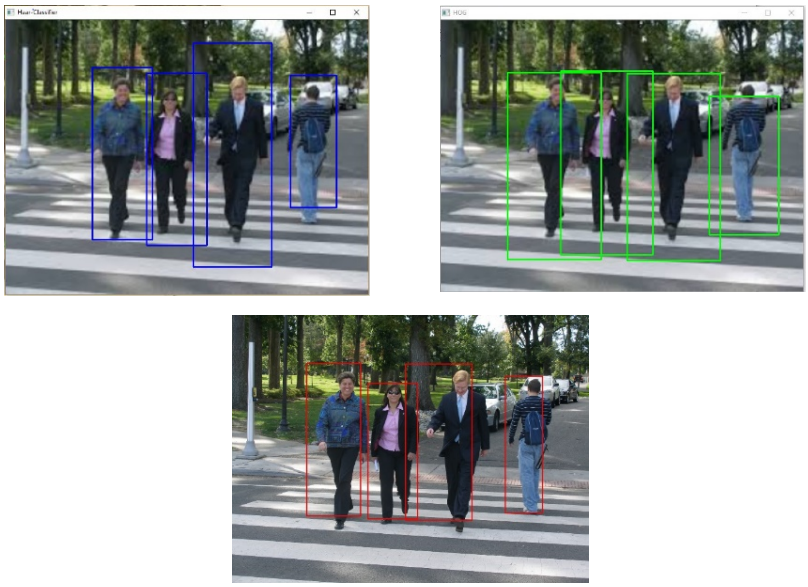

Fig. 12. Computer vision algorithm for Haar-classifier (left), HOG descriptor (right) and Deep learning (bottom)

Table 3 shows the numerical results for the comparisons of the two computer vision methods against deep learning. The bounding boxes refer to the rectangular boxes indicating the ROIs (Fig. 12 – blue for Haar (left), green for HOG (right) and red for deep learning (bottom)). The false positives refer to any bounding box that is outside a pedestrian or repeating an ROI that has already been identified. The detection failures indicate that there was no bounding box (or not sufficient to qualify as an ROI). The results show that out of 41 pedestrians, the Haar correctly identified 22 pedestrians giving it an accuracy of 53.66%, HOG achieved a success rate of 82.93% and deep learning outperformed both by achieving a success rate of around 88%.

Both Haar and HOG had high amounts of false positives and struggled when given an image of multiple pedestrians scattered across the image. However, comparison 2 in Table 3 shows that the deep learning algorithm recognised pedestrians that were not recognised by the HOG and Haar as they were deemed unnecessary because they were too far away. Similarly, detection failures in comparison 11 and 12 are due to the algorithm recognising multiple pedestrians together as a single pedestrian. To ensure the same pass parameters were taken for Deep Learning as well as HOG and Haar, these were taken as a detection failure even though practically the system would avoid those pedestrians.

Table 3. Comparison of Haar-classifier, HOG descriptor and Deep learning

	Bounding Boxes			Pedestrians	False positives			Detection Failures		
Com-parison	Haar	HOG	Deep learning	Haar/HOG/Deep learning	Haar	HOG	Deep learning	Haar	HOG	Deep learning
1	4	4	4	4	0	0	0	0	0	0
2	3	3	13	1	2	3	13	0	1	0
3	3	6	2	2	2	4	0	1	0	0
4	4	3	2	1	3	2	2	0	0	0
5	1	2	2	1	0	1	2	0	0	0
6	4	5	3	4	1	3	3	1	1	1
7	5	4	4	4	3	1	0	2	1	0
8	2	1	1	1	1	0	0	0	0	0
9	1	4	2	2	0	1	0	1	0	0
10	6	2	4	4	2	0	0	0	0	0
11	2	6	6	8	0	0	1	6	1	2
12	2	3	9	9	1	0	2	8	3	2
Total				41	30	30	23	19	7	5
Succ-ess Rate								53.66 %	82.93 %	87.8 %

The HOG and Haar-Classifier were further compared on the Raspberry Pi 3. The Pi camera was used and results from the FPS were taken and displayed in Table 4.

Table 4. FPS comparison.

	Haar (FPS)	HOG (FPS)
Comparison 1	3.974	0.808
Comparison 2	3.623	0.780
Comparison 3	3.776	0.815
Comparison 4	3.580	0.783
Comparison 5	3.603	0.792
Average FPS	3.711	0.796

Analysis of the results shows that the Haar-Classifier performs significantly better than the HOG Descriptor for maximising the FPS. This is likely due to the quality of HOG requiring more computational resources thus slowing down the frames being passed through the Pi per second. The test also showed repeated false positives with the Haar-Classifier which could be caused by lighting while the HOG descriptor could be fooled (Comparison (HOG) 4) if the pedestrian's clothing was of similar

colour to the background. All images on the Raspberry Pi comparison in Table 4 can be found in [27].

The implementation of pedestrian recognition via computer vision and deep learning demonstrates the effectiveness that the different machine learning algorithms have for autonomous cars. Computer Vision and deep learning however, can produce an accurate description of an object but it is a relatively new technology that has only recently had machine learning methods applied to it. The results had shown that both methods of computer vision worked poorly on images of dispersed pedestrians, however, was much improved with deep learning.

5.5 Discussion of Results

The stopping distance results have shown the need for a factor of safety to protect against the inconsistency of the ultrasonic sensors and the clock time results support this. The speed of the software and hardware play an important role in the stopping time. The sensor accuracy results have shown that while the ultrasonic sensors in most test runs avoided an obstacle, they cannot be used in this form in real systems. For example, the use of breadboard and jumper wires was adding to the delay for stopping the vehicle. Although ultrasonic sensors are simple to use and can give results sufficient enough for the system to avoid collisions they are unable to differentiate between objects or give detailed data back to the Raspberry Pi for analysis. Similarly, Raspberry Pi is running Raspbian which is not optimised for real world performance.

Our results agree with current studies [28] where HOG outperforms Haar-Classifiers, however, other peer-reviewed comparisons have shown a significantly higher number of false detections in the Haar-Classifier [29] compared to our study, however, a larger dataset would have to be used to confirm this difference. Results 5 from Table 3 demonstrate the Haar-Classifier's limits with lighting thus agreeing with [30] analysis although their study was performed on faces compared to ours where we are detecting pedestrians the principles behind the constraints are still valid.

The pedestrian recognition using computer vision and deep learning demonstrates the effectiveness and potential that machine and deep learning algorithms have for autonomous cars. The machine learning results have shown that both methods worked less effectively on images of dispersed pedestrians, however, deep learning outperformed machine learning.

6 Conclusions

This paper has presented results from converting a mobility vehicle to be fully autonomous with computer vision and deep learning implemented for pedestrian recognition. We further present results on ultrasonic sensors for obstacle detection. Our results show that at short distances ultrasonic sensor shows a delay, however, for distances above 100cm the sensors react very well.

The comparison between computer vision algorithms of Haar-classifier and HOG descriptor with deep learning show that deep learning outperform both algorithms, whereas, HOG descriptor gave better results than Haar-classifier. We conclude that

Raspberry Pi 3 is well suited as a microcontroller for research purposes, however, a more bespoke device would be recommended for 'real' vehicles.

The construction and evaluation of the autonomous vehicle shows that the Raspberry Pi functions as a possible microcontroller option for an autonomous system [31]. The speed is the main concern with the Raspberry Pi as it is running Raspbian which is not optimised for real world performance. However, Raspberry Pi as a microcontroller from our results has shown the benefits of autotomizing a system, the feasibility of use allows access to data which can be extrapolated to deduce problems within the system.

Future work will include increasing the training images on all algorithms to improve their performance, getting test data from the vehicle for vehicular cloud data management.

7 Acknowledgments

The authors would like to thank Mr Stuart MacVeigh, Mr John Welsh and Dr Toby Whitley for their support in building the autonomous vehicle and general electronics.

The work reported here is in part supported by the internal School of Engineering Research Grant.

References

1. Land transportation system report by Cisco http://www.cisco.com/c/dam/global/pt_pt/assets/ciscoconnect-2013/pdf/16_outubro_helder_antunes.pdf last accessed 2018/09/11.
2. Viola, P. M. J. 'Rapid object detection using a boosted cascade of simple features', IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2011, Kauai, USA.
3. Chaudhari M. S. S. 'A review on Face Detection and study of Viola Jones method'. International Journal of Computer Trends and Technology, 2015.
4. Takuya K. A. H. 'Selection of Histograms of Oriented Gradients Features for Pedestrian Detection'. Kitakyushu, Japan: Neural Information Processing, 2007.
5. Dalal, N. B. T. 'Histogram of oriented gradients for human detection'. San Diego: Computer Vision and Pattern Recognition, 2005.
6. Ameur H., Msolli, A., Helali, A., Maaref H. and Youssef, A. 'Hardware implementation of an improved HOG descriptor for pedestrian detection', ICCAD'17, Hammamet - Tunisia, January 19-21, 2017.
7. Shilpashree, K., S., Lokesha, H., Shivkumar, H. 'Implementation of Image Processing on Raspberry Pi', International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 5, May 2015.
8. Pan, J-S., Ma, S., Chen, S-H. and Yang, C-S. 'Vision-based Vehicle Forward Collision Warning System Using Optical Flow Algorithm', Journal of Information Hiding and Multimedia Signal Processing, Volume 6, Number 5, September 2015.
9. Mammeri, A., Zuo, T. and Boukerche, A. 'Extending the Detection Range of Vision-based Driver Assistance Systems Application to Pedestrian Protection System', Globecom-Communications Software, Services and Multimedia Symposium 2014.
10. Lin, S. F. and Lee, C. H., 'Pedestrians and Vehicles Recognition Based on Image Recognition and Laser Distance Detection', 16th International Conference on

Control, Automation and Systems (ICCAS 2016) Oct. 16-19, 2016 in HICO, Gyeongju, Korea.
11. Brehar R. and Nedevschi, S., 'A comparative study of pedestrian detection methods using classical Haar and HoG features versus bag of words model computed from Haar and HoG features', IEEE 7th International Conference on Intelligent Computer Communication and Processing (ICCP), 2011.
12. Fei-fei, L., 'A bayesian hierarchical model for learning natural scene categories,' in In CVPR, 2005, pp. 524–531.
13. Bruno D. R. and Osório, F. S., 'Image classification system based on Deep Learning applied to the recognition of traffic signs for intelligent robotic vehicle navigation purposes', Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR), 2017.
14. Orozco, C., I., Buemi, M., E., and Berlles, M., J., 'New Deep Convolutional Neural Network Architecture for Pedestrian Detection', 8th International Conference of Pattern Recognition Systems (ICPRS 2017).
15. Geismann P. and Schneider, G., 'A Two-staged Approach to Vision-based Pedestrian Recognition Using Haar and HOG Features', 2008 IEEE Intelligent Vehicles Symposium Eindhoven University of Technology Eindhoven, The Netherlands, June 4-6, 2008.
16. Manlises, C. O., Martinez, J. R., Belenzo, J. L., Perez, C. K. and Postrero, M. K. T. A., 'Real-Time Integrated CCTV Using Face and Pedestrian Detection Image Processing Algorithm For Automatic Traffic Light Transitions', 8th IEEE International Conference Humanoid, Nanotechnology, Information Technology Communication and Control, Environment and Management (HNICEM) The Institute of Electrical and Electronics Engineers Inc. (IEEE) – Philippine Section 9-12 December 2015, Cebu, Philippines.
17. Priyanka D, Dharani K, Anirudh C, Akshay K, Sunil M P, Hariprasad S A, 'Traffic Light and Sign Detection for Autonomous Land Vehicle Using Raspberry Pi', Proceedings of the International Conference on Inventive Computing and Informatics (ICICI 2017) IEEE Xplore Compliant - Part Number: CFP17L34-ART, ISBN: 978-1-5386-4031-9.
18. Shah, A. A., Zaidi, Z. A., Chowdhry, B. S. and Daudpoto, J., 'Real time Face Detection/Monitor using Raspberry pi and MATLAB', IEEE 10th International Conference on Application of Information and Communication Technologies (AICT).
19. Betterlife Healthcare, 2018. Betterlife Capricorn Electric Wheelchair.
20. Adafruit at https://www.adafruit.com/, 2018. [Last accessed 13th April 2018]
21. OpenCV https://github.com/opencv/opencv/tree/master/data
22. Natasha Seo training dataset for HOG and Haar http://note.sonots.com/SciSoftware/haartraining.html [Last accessed 15th April 2018]
23. Freund, R. E., 'A Short Introduction to Boosting', Journal of Japanese Society for Artificial Intelligence, 1-14, 1999.
24. COCO and VOCO712 datasets http://cocodataset.org/#download and https://github.com/chuanqi305/MobileNet-SSD
25. Images for pedestrian training and Raspberry Pi comparison https://github.com/asiyakhan0/images-
26. Liu, W., Anguelov, D., Erhan, D., Szegedy, C. and Reed. S. E., SSD: single shot multibox detector. CoRR, 2015, 5, 6.
27. Vinayak V. Dixit, S. C., 'Autonomous Vehicles: Disengagements, Accidents', and Reaction Times. *PLOS*. ONE, 11(12), e0168054, 2016.

28. Wei, T. Q. ‘An Improved Pedestrian Detection Algorithm Integrating Haar-Like Features and HOG Descriptors’, *Advances in Mechanical Engineering*, 2013, 5(546):206.
29. Xing, W., Zhao, Y., Cheng, R., Xu, J., Lv, S.,and Wang, X., ‘Fast Pedestrian Detection Based on Haar Pre-Detection’, International Journal of Computer and Communication Engineering, Vol. 1, pp. 207-209, 2012.
30. Chaudhari M., Sondur S. and Vanjare G., ‘A review on Face Detection and study of Viola Jones method’. *International Journal of Computer Trends and Technology*, vol. 25, no. 1, July 2015.
31. Chaudhari, H., ‘Raspberry Pi Technology: A Review’, *International Journal of Innovative and Emerging Research in Engineering*, vol. 2, issue 3, 2015.