THE UNIVERSITY OF WESTERN AUSTRALIA
SCHOOL OF MECHANICAL AND CHEMICAL ENGINEERING

Autonomous SAE Car – Drive By Platoon System

Final Year Project Thesis

Calvin Wen-Loong Yapp
20760241

Supervisor: Professor Dr. Thomas Bräunl
School of Electrical & Electronic Engineering
University of Western Australia

Date of Submission: 28th October 2013

PROJECT SUMMARY

This project is a part of the joint project, Autonomous SAE Car under the Renewable Energy Vehicle (REV) program headed by Thomas Bräunl at the School of Electrical and Electronic Engineering at the University of Western Australia. The Autonomous SAE project involved three final year students including Thomas Drage (School of Electrical & Electronic Engineering), Jordan Kalinowski (School of Electrical & Electronic Engineering) and Calvin Yapp (School of Mechanical Engineering).

Traffic congestion is an issue on Perth roads and is steadily getting worse. A possible solution to help improve the congestion is introducing a system called drive by platoon in which cars autonomously follow each other allowing drivers to perform other tasks while in traffic. The objective of this project is to implement a drive by platoon on the autonomous SAE car which will allow the car to follow an object, person or vehicle driving in front of it.

This is achieved using the IBEO LUX sensor that is able to detect objects using the time of flight principle. The data received from the sensor is analysed and processed to create a plot of points obtained and an algorithm is used to identify the objects present in front of the autonomous SAE car. The object detection algorithm is able to detect objects to within 6 centimetres laterally and 24 centimetres longitudinally.

Using this data, the object is tracked and commands including acceleration, brake and steering are sent to the control unit of the autonomous car. A program has been written that incorporates the object detection algorithm, the plotting of objects and communication to the autonomous SAE and using this program the autonomous SAE car is able to track and follow a person walking in front of the car.

Calvin W. Yapp
6 Ballan Court
Morley, W.A, 6062

28th October 2013

Winthrop Professor John Dell
Dean
Faculty of Engineering, Computing and Mathematics
University of Western Australia
35 Stirling Highway
Crawley, W.A, 6009

Dear Professor Dell,

I am pleased to submit this thesis, entitled "Autonomous SAE Car – Drive by Platoon System", as part of the requirement for the degree of Bachelor of Engineering.

Yours Sincerely,

Calvin W. Yapp
20760241

ACKNOWLEDGEMENTS

I would like to thank Thomas Bräunl for his continuing support and guidance throughout this project, the past and current members of the REV team for the help and assistance throughout the project, the staff at the University of Western Australia Mechanical Workshop, especially Mark Henderson, the staff at EV works for help in the construction of various parts of the car and all the faculty staff at the UWA faculty of Mathematics and Engineering.

CONTENTS

1 DESCRIPTION OF PROJECT

According to a survey conducted by the Royal Automobile Club of Western Australia, 90% of businesses surveyed have commented that workers have spent an increase in time on the road over the past 12 month. With the traffic congestion in Perth getting steadily worse (RAC & CCI 2012) the current infrastructure can't keep up with the growing demands. One long term solution that would help decrease the congestion on Perth's freeways and major highways is to introduce a drive-by-platoon system.

Drive-by-platoon involves vehicles following each other autonomously communicating signals to each vehicle wirelessly and allowing the driver to perform other tasks while on the road. The concept of platooning has been developed since the mid 90's (Avanzini et al. 2008) with the design of platooning urban vehicles (small electric cars suitable for 2 passengers). In 2009 the SAfe Road TRains for the Environment (SARTRE) project funded by the European Commission under the Framework 7 programme began, aiming to develop a prototype of a platoon system that would run on existing highways. This was successfully tested on a public highway in May 2012. (Tulloch 2012) Another project working on the concept of driving in a platoon is the Automated Highway System (AHS) designed by California PATH (Partners for Advanced Transportation Technology). PATH aims to create a fully automated highway system that will only consist of fully automated vehicles.

1.1 PROJECT OBJECTIVES AND METHOD

The aim of this project is to implement a drive-by-platoon system on the electric 2010 SAE car. By the completion of this project, the objective is to have the SAE car follow a single vehicle around a set course. Both longitudinal and lateral control will need to be considered as well as safety systems to prevent the loss of control of the vehicle and possible collisions between the car, the leading vehicle and any surrounding obstacles.

This project will be a part of joint project working alongside Jordan Kalinowski and Thomas Drage, both Electrical Engineering students from the University of Western Australia. Both Kalinowski and Drage are working on a section of the autonomous SAE car as part of their thesis. Drage will be working on the high level control of the autonomous SAE car and will be trying to create a fully autonomous SAE race car that

will be able to follow a predefined path on a race track. Kalinowski will be working on the low level control of the autonomous car.

The low level control involves converting the 2010 electric SAE car from a manually driven car to an autonomously controlled vehicle. To achieve this, hydraulic breaks will be controlled by a servo motor, the accelerator will be controlled electronically and the steering will be controlled by a DC motor. Safety is of the upmost importance and systems will need to be created to prevent the car from colliding into obstacles and continuing to drive when communication has been lost from the base station. Even though this car will be used primarily under autonomous control, the car will need to be modified so that it can be switched to manual control when needed.

An IBEO Lux sensor will be used as the main piece of equipment in implementing the drive by platoon system on the autonomous SAE car. Research into how to obtain the information from the IBEO will need to be completed as well as analysing and interpreting the information. The data received from the sensor will also need to be plotted and a suitable program will need to be found to complete this. Using the data obtained from the sensor, an algorithm will need to be created that will identify objects from a set of points. Using the algorithm the object will be tracked and commands need to be sent to the low level control to control the car.

When running the program to track the object, key issues that need to be considered include keeping the car at a safe distance to ensure that it won't collide with the vehicle in front of it in the case of an emergency brake, minimising the oscillations in speed and setting the distance between vehicles at a range of distances. When the steering of the system is considered the following challenges will need to be taken into account, the speed the car is travelling, the angle required for the wheels to turn so that the vehicle can line up with its leading vehicle and the prevention of oscillating changes in direction that will be very inefficient.

1.2 PROJECT PURPOSE

There are many advantages of driving in a platoon including increased safety, reduction in fuel consumption, increased comfort and convenience for the driver and saving of time (Bergenhem, Hedin & Skarin 2012)

According to the Department of Planning, Transport and Infrastructure in South Australia, driver fatigue is a major cause of accidents on the road. Between 2004 and 2008 an average of 14% of fatal crashes were highly likely due to driver fatigue. (Government of South Australia: Department of Planning, Transport and Infrastructure 2012). Especially on long trips in rural areas fatigue has a large impact on vehicle accidents. If driving in a platoon is implemented on these long trips there will be a great decreases in the number of accidents as drivers will be able to rest and the human aspect of driving will be taken out of the picture.

In the past 10 years from May 2003 to May 2013 there has been an increase in the average fuel price of unleaded fuel of 50.4c from 85.7c to 136.1c (Department of Commerce 2013). This is a huge increase of 58.8% over 10 years. Driving in a platoon has proven to increases fuel efficiency (as long as cars are within close proximity of each other) (Bergenhem et al. 2010) and other systems are looking into urban vehicles which run on electricity and not petrol. With fuel prices increasing by so much even a small saving in fuel by driving in a platoon can make a large difference overall.

An increase in fuel efficiency does not only make an economic difference but also an environmental difference. On the 9th May 2013 the concentration of carbon dioxide in the Earth's atmosphere (recorded at the National Oceanic and Atmospheric Administration's (NOAA) Mauna Loa Observatory) reached 400ppm (National Oceanic and Atmospheric Administration 2013). In a study conducted by the Australian Government's Department of Climate Change and Energy Efficiency, 14% of the nation's greenhouse gas emissions in 2009 were due to transport (Department of Climate Change and Energy Efficiency 2010). They also believe that between 2010 and 2020 there will be an increase of 15% of emissions contributed by transport. One way this contribution can be reduced significantly is by introducing more electric platoon vehicles especially in built up urban areas. By using electric vehicles, zero emissions are released into the atmosphere.

The last significant advantage of driving in a platoon is the drivers comfort and convenience. According to the RAC BusinessWise and CCI Congestion survey conducted in 2012 almost 1 in every 5 people surveyed said they spent more than 50 hours in additional time on the road than compared to 12 months ago (RAC & CCI 2012). This means that people are arriving to work tired and there is a decrease in productivity during the day. If driving in a platoon was introduced, the time spent on the

road will decrease as cars will be able to travel faster and closer together on highways but more importantly drivers will be able to perform other tasks on the road like reading the newspaper, eating breakfast, catching up on work or just resting. This will mean workers arrive at work better rested, more relaxed and ready for another day at work.

With all these benefits, driving in a platoon can make a big difference to way people commute around the world and could possibly be the way we all travel in the future.

By researching the current available technologies and implementing a similar system on the 2010 electric SAE car, this project aims to provide a better understanding of how platoon systems work, a basic implementation of a drive by platoon system and look into the possibility that one day a similar system will be implemented on the roads of Perth.

2 LITERATURE REVIEW

The following paragraphs are an introduction into the various projects developing around the world in regards to driving in a platoon. These projects will be analysed and reviewed, comparing the different ideas presented in each project.

2.1 CYBERCARS

Cybercars are small fully automated vehicles that can be used for passenger transport or goods transport in an urban environment. The concept of cybercars has been around since the early 1990's and multiple projects have been run in Europe including CyberCars, CyberMove, EDICT, Netmobil, and CyberC3 (Naranjo et al. 2009). Current transportation systems inefficiently use space and a solution to this problem is to introduce cyber cars (Parent 2004). As cyber cars are fully automated, computers or microcontrollers will control the acceleration, braking/deceleration and steering of the vehicle.

There are many different types of navigation methods available but each has their advantages and disadvantages. In a report by Michel Parent, he lists the following navigation methods; using wire guidance, transponders, magnetic sensors or GPS (Parent 2004). The disadvantages of wire guidance and transponders are that they are expensive and hard to install and maintain, while GPS systems can have problems in built up areas as satellite signals can be obstructed. (Parent 2004). Another two methods

suggested by P. Avanzini, E. Royer, B. Thuilot and P. Martinet are using camera and laser rangefinders. Cameras are cheap to install but have a problem with slightly distorted representations in the virtual world when compared to the actual world. Laser rangefinders are more expensive but have very accurate measurements. (Avanzini et al. 2008)

In Parents report he suggests three techniques to implement a platoon system. The first is a camera based approach where the following cybercar extracts certain features of the car it is following. The second is a laser scanner with reflective beacons which can be used at distances between 2 and 15m. The third approach is to use a camera based technique with infrared lamps.

Using platooning with cybercars can have two significant advantages. Firstly, empty vehicles can be moved from one location to another by following a car with a driver and space on the road can be saved by having cybercars follow closely behind each other without collision. As cybercars are also electric running costs are low and zero greenhouse emissions are released.

2.2 SAFE ROAD TRAINS FOR THE ENVIRONMENT (SARTRE)

The Safe Road Trains for the Environment (SARTRE) is a European Commission Co-Funded Framework 7 programme project. Companies involved in the project include Idiada and Robotiker-Tecnalia of Spain, Institut for Kraftfahrwesen Aachen (IKA) of Germany, SP Technical Research Institute of Sweden, Volvo Car Corporation and Volvo Technology of Sweden. (SARTRE 2012) The project was started in 2009 and aims to develop vehicles that will allow it to travel in a platoon. They also aim to use current infrastructure on highways so that platoon vehicles can be run alongside non-platoon vehicles.

The SARTRE system comprises of two types of vehicles, a leading vehicle that is manually driven by a professional driver and following vehicles which are controlled by the leading vehicle. In the first road test conducted in late May 2012 the lead vehicle was a truck and the following vehicles comprised of another truck and three cars. (Tulloch 2012) The test took place just outside Barcelona and lasted for 200km.

Figure 1: First Road test of SARTRE System (Tulloch 2012)

Each vehicle in the platoon was equipped with cameras, radars, laser sensors and wireless communication gear allowing each vehicle to transmit information to each other. (Tulloch 2012). The following vehicles operate by mimicking the lead vehicles movement which is wirelessly transmitted to the car. The other sensors attached to the vehicle monitor the vehicles distance and movement to ensure the vehicle remains at a close distant.

Fuel efficiency is a big benefit when driving in a platoon however to make a significant difference the cars need to be following in close proximity (Bergenhem et al. 2010). Therefore there are many safety procedures that need to be taken into consideration including adaptive cruise control, collision mitigation by braking and lane departing.

Safety is a very important part in the designing of the SARTRE system. As the vehicle is not completely automated and human interaction is still involved in the system, human factors need to be taken into consideration. In a report by Maider Larburu, Javier Sanchez and Domingo José Rodriguez they look into the design of the human machine interface (HMI) that would be used to interact and communicate with the driver. When considering the design they had to take into account the following factors:

- Situational Awareness – Will the driver be aware of their surroundings while under autonomous control?

- Behavioral Adaption – Knowing that the car is under autonomous control how will this change the behavior of the driver? What happens if the system breaks down, how will the driver react?
- Transitioning from Autonomous to Manual driving – Will the driver be alert enough to manually drive the car after been in autonomous mode?

Taking these factors into consideration it is important that the HMI communicates efficiently and effectively what the automated system is doing, what mode the system is in and keep the driver aware of their surroundings. The communication to the driver should not only be visual but should be auditory and haptic. (Larburu, Sanchez & Rodriguez 2010)

2.3 PATH AUTOMATED HIGHWAY SYSTEM

California PATH (Partners for Advanced Transportation Technology) was established in 1986 and is a collaboration of work between the Institute of Transportation studies and the University of California and Caltrans. Together they aim to create new developments in transportation research and develop solutions to improve surface transportation systems. (California PATH n.d.)

One project that was developed by California PATH is the Automated Highway System (AHS). (Chin, Hall & Gadgil 2003) The AHS looks into the accessibility and flexibility of roadway systems, the susceptibility of crowding and congestion and the large space requirements between vehicles. As drivers do not have the skill to drive at a high speed and a close distance to other vehicles the majority of space on a highway is empty as it is the space between vehicles.

According to Chin, Hall and Gadgil, adding automation to the road system will increase the capacity of roads as vehicles will be able to travel faster and closer together. However some challenges involved in automation include:

- The construction of new infrastructure to accommodate for the increase in traffic flow and to inspect vehicles as they enter and exit the highway.
- Real time control of vehicles entering and exiting the highway
- Safe transitions between autonomous and manual driving control
- New management structures to construct and operate automated highways.

In the design of the AHS, PATH looks into changing the infrastructure of highways to maximise the effectiveness of the automated system. One consideration is the entrances and exits of highways. As vehicles are inspected and sorted space is needed for this to occur. To maximise the benefits of a platoon, it is desirable to have a large number of vehicles in the platoon travelling to the same destination. (Chin, Hall & Gadgil 2003) If one vehicle is to leave the platoon, vehicles may not be able to travel as close as possible to each other and there will be wasted space.

Chin, Hall and Gadgill have proposed possible platoon formation strategies and these are as follows:

- Destination group – Grouping vehicles according to their destinations. Separating vehicles into various lanes if they are exiting at adjacent exits.
- Dynamic Group – Each group is assigned a certain range of distance that it will travel and vehicles are sorted if their destination is within the range. If the vehicles destination is out of the range, the largest platoon is sent off and the vehicle makes a new platoon.
- Dynamic Grouping and Platooning Splitting – Sorting lanes so that the vehicles are in order of furthest to nearest exit. This way platoons can continue as vehicles from the back exit.

To test which was the most effective formation strategy in terms of platoon ratio, waiting time and highway throughput Chin, Hall and Gadgill performed a simulation with the following variables: trip length, exit spacing, number of entrance lanes, average trip length, highway length and formation method.

The results from the simulation were that dynamic grouping produced the largest platoon ratio and highway throughput for most situations however with the number of lanes limited to 2 or 3 the dynamic grouping performed better (Chin, Hall & Gadgil 2003). In regards to the waiting time, it was a linear relationship to the number of lanes. As the number of lanes increased it took longer for platoons of a given size to be formed and therefore the average waiting time was higher.

In a report written by Roberto Horowitz, Chin-Woo Tan and Xiaotian Sun from the University of California Berkeley, they propose a manoeuvre for vehicles to change lanes within a platoon. This manoeuvre works by aligning adjacent platoons parallel to each other and locking their longitudinal positions. A space adjacent to the vehicle

changing lanes is made by moving the front and back vehicles apart and once a sufficient space is made the vehicle moves laterally to fill up the space. Once the manoeuvre is complete the two platoons disengage and a new manoeuvre can be made. Through the introduction of this manoeuvre, efficiency and highway utilisation can be increased. (Horowitz, Tan & Sun 2004)

In regards to the control of the AHS, driving will be completely controlled by automation and therefore the driver will be out the control loop. The vehicles on the AHS must also be able to communicate with other vehicles as well as the roadway infrastructure. (Shladover 2008)

2.4 COMPARISON OF THE THREE PROJECTS

Each of the three projects, Safe Road Trains for the Environment (SARTRE), Automated Highway System (AHS) and Cybercars have different methods of implementation.

SARTRE plans to implement their system by installing sensors and communication devices on vehicles and using existing infrastructure. The AHS will need extra infrastructure added to existing highways and modifications will also need to be made to vehicles. As the Cybercar is a completely new vehicle they will be produced with the sensors and communications inbuilt.

In terms of cost, the SARTRE project and the cybercars will have a lot lower start-up cost than the AHS as no new infrastructure needs to be built. (Avanzini et al. 2008) Therefore no time will be needed for planning infrastructure and platoons can be rolled out onto existing streets and highways. However with the SARTRE project, as it relies on a professional driver to drive the lead vehicles, running cost will be a lot higher than the AHS. In the first public road test the leading vehicle was a truck. (Tulloch 2012) If they continue to use a truck as the leading vehicle, the fuel consumed will greatly contribute to the running costs. For both the SARTRE and AHS project the cost of vehicles will greatly decline once the annual production of vehicles pre-installed with autonomous equipment increases. Already in the market, vehicles are being installed with many expensive components that would be used in both the SARTRE and AHS. This means that by the time the project is ready for implementation the majority of cars will already be equipped with a lot of the components required for autonomous driving. For the AHS project the majority of infrastructure costs will be due to the construction

of the entry and exit ramps. Therefore these costs can be decreased if fewer ramps are installed depending on the required capacity. (Shladover 2008)

Capacity wise the AHS is capable of handling a much larger throughput than the SARTRE project. Through the AHS, all lanes on the highway can be used autonomously however to install a system this large, the whole highway will need to be closed and this would cause major traffic interruptions. Another problem with changing the entire highway is that only people who have new cars with the modifications pre-installed or can afford the modifications to be put on existing vehicle will be able to use the new highway. It will also mean that cars without the modifications won't be able to use the autonomous highway which may cause traffic problems on other streets. Over time as the older cars phase out, this will become less of a problem but it is unknown how long it will take for this to occur.

Both the SARTRE and AHS employ different methods in terms of driver control. With the SARTRE, the driver is given the ability to decide when they want to join and leave the platoon while with the AHS, the vehicle is completely autonomous and the driver has no input into the system from the time they enter the highway to the time they exit. Having the ability to leave a platoon at the choice of the driver creates flexibility and allows the driver to change their route at any given time. However this can cause issues as it becomes the responsibility of the driver to leave the platoon when they wish to exit and drivers may not be fully alert when transitioning from autonomous control to manual control creating the possibility of accidents to occur. A fully automated system like the AHS means that the transition from manual to autonomous driving will occur when the car is stationary. This gives the driver time to be aware of their surroundings and time to adjust to manual driving controls.

As technology is continually evolving it won't be long til these projects or similar projects will be installed in cities around the world. For the new future it would be more likely to see systems like SARTRE and cybercars as they do not require new infrastructure to be built and can easily be implemented on existing infrastructure. They will also be a good way to slowly introduce the idea of autonomous vehicles to the public as the success of the project depends on the acceptance of autonomous vehicles among the community. For the project to be successful people need to feel more comfortable and safer with the system than without. Once the idea of autonomous

vehicles is accepted by the community and the demand of autonomous vehicles increases, the AHS system will be the way of the future.

3 IBEO LUX SENSOR

3.1 DESCRIPTION OF THE IBEO LUX SENSOR

The IBEO LUX Sensor will be the main piece of equipment used in this project.

IBEO Automotive Systems GmbH is a company based in Hamburg Germany that was founded in 1998, focusing on time-of-flight technology to measure distances. Their mission is to work towards a future with no road accidents and to produce products that enhance drivers comfort and reduce the loss of lives on the road. (IBEO Automotive Systems 2013)

The IBEO LUX Sensor makes use of the LIDAR (Light Detection and Ranging) system and uses the time of flight technology to detect objects. The LIDAR system works by emitting rapid pulses of light that bounces of objects and returns to the sensor where the time of flight of the pulse can be used to calculate the distance the sensor is away from the object. (CSIRO 2008)

The sensor has a range of up to 200m has a horizontal field of view between 85 and 110 degrees depending on the number of layers used. The sensor has the capabilities of scanning at up to 4 parallel layers with a vertical field of view up to 3.2 degrees. The sensor also features multi echo detection which enables the sensor to be used in all weather conditions including rain and dust clutter. The accuracy of the sensor is to within 10cm independent of the distance and has a distance resolution of 4cm. The angle resolution is up to 0.125 degrees horizontally and 0.8 degrees vertically. (IBEO Automotive Systems GmbH 2013)

The power supply required for the sensor is 9 to 27V. On the SAE autonomous car the sensor will be powered by the 12V power supply on the car. Data input and output can be transferred through 100 MBit Ethernet, Controller Area Network (CAN) cable or RS232.

3.2 INITIAL TESTS OF THE IBEO LUX SENSOR

The objective of these initial tests were to get an understanding of what data was outputted by the IBEO LUX sensor and to see if it is possible to identify objects from the data.

On the 10th March the 2010 Electric SAE car was taken to the Electrikhana Event at the RAC Driving Centre to test the functionality of the car and obtain data from the IBEO sensor and the GPS.

Unfortunately after half a lap of the driving on the race track the laptops hard drive crashed, possibly due to the vibration of the car or the bumpiness of the race track. This issue has been fixed and a solid state hard drive has been installed into the laptop.

From the data obtained before the hard drive crashed a few issues in regards to the IBEO sensor were noticed. The first issue was that the mechanical mount of the IBEO sensor was not strong enough to support the sensor and therefore moved over time as the car was driven around the track. This is a big problem especially with object detection as objects are tracked by comparing previous and current data. The program won't be able to determine if the change in position of the object is due to movement of the object or the movement of the sensor. This problem was fixed by installing a new bracket and mount designed by Thomas Drage that is placed above the drivers head as seen in figure 2.

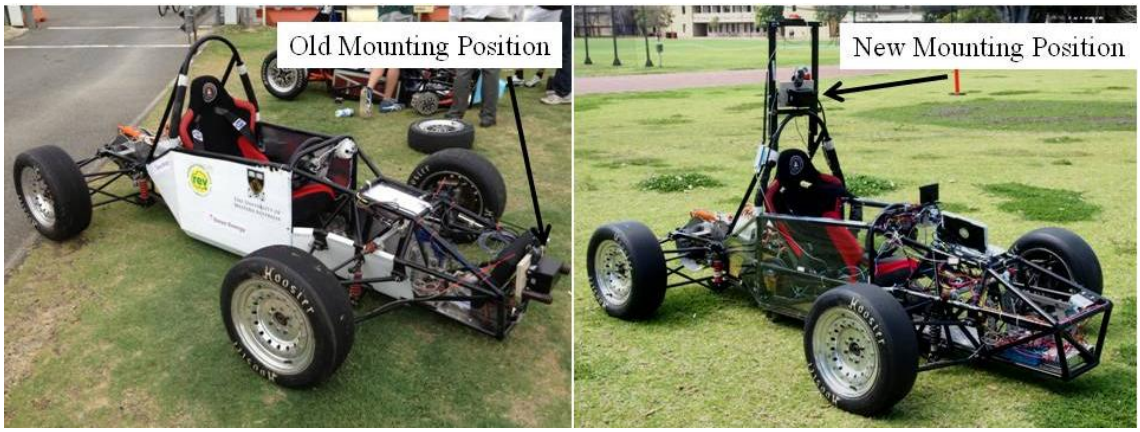


Figure 2: Old and new mounting positions of the IBEO sensor

When taken back to the university another experiment was conducted to compare the difference in results when the IBEO was placed above the drivers head and when placed at the front of the car. The experiment took place on a path that was lined with a grass

on one side and a wall on the other. Traffic cones were placed in front of the car to see if they could be detected. Below is a plot of the data received from the IBEO when placed above the drivers head.

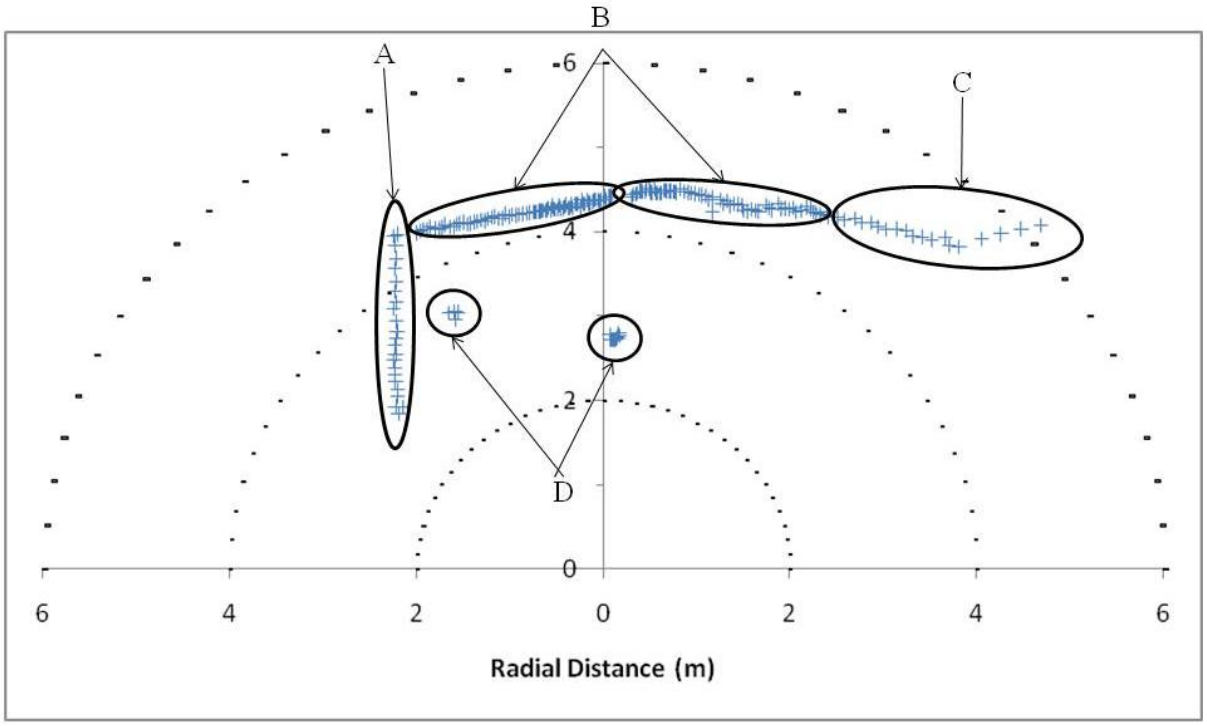


Figure 3: Polar plot of data from the IBEO Sensor placed above the drivers head

These are the features noticed in the plot:

A. The wall alongside the path
B. The flat surface of the path
C. Points corresponding to the uneven surface of the grass alongside the path
D. Traffic cones placed in front of the car

From the plot the traffic cones can be easily identified and it can be observed that objects are made of a multiple points closely grouped together. As the points are closely grouped together the distance between points can be used to determine which points are part of objects are which points are not.

3.3 DETERMINING THE ANGLE OF THE IBEO LAYERS

The IBEO sensor scans at four parallel layers which allows the user to get scan points from different heights. In order to figure out which scan layer would be the most suitable to use when scanning for a various objects the angle of each layer needed to be determines. The following test was conducted to work out the angle of each layer with respect to the horizontal when the IBEO sensor is mounted at the top of the roll cage on the SAE car.

In the lab a piece of square cross section metal tubing was placed on a stack of tyres, books and a bin. The height of the metal tubing was constantly adjusted until the metal tubing was visible in a single layer of points from the IBEO scanner. The height of the metal tube was recorded and the process was repeated for each layer of the IBEO scanner. Figure 4 shows the experimental set up.

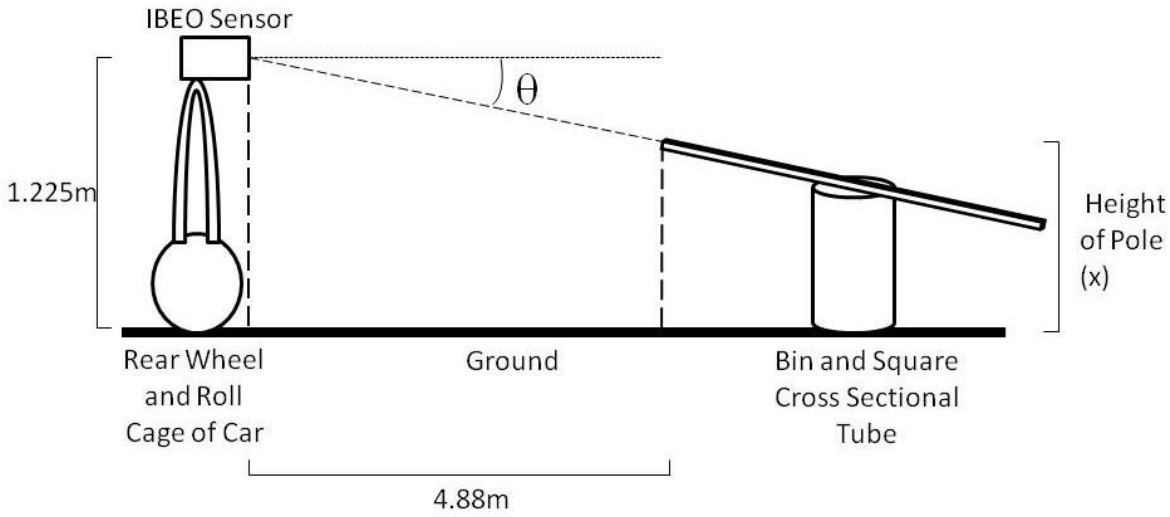


Figure 4: Experimental set up to determine the angle of IBEO layers

Where:

$$\theta = \tan^{-1}\left(\frac{1.225-x}{4.88}\right) \quad (1.1)$$

Figures 5 to 8 show the plots of points for each of the layers.

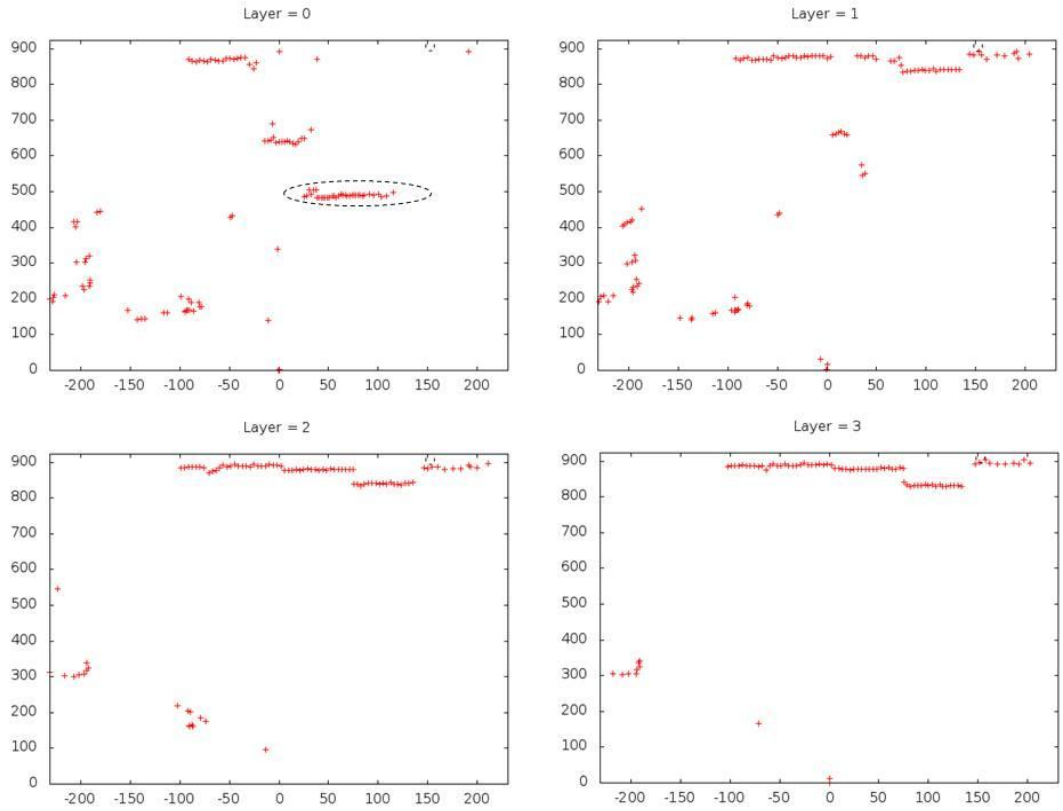


Figure 5: Pole at height of 0.97m can be seen in layer 0

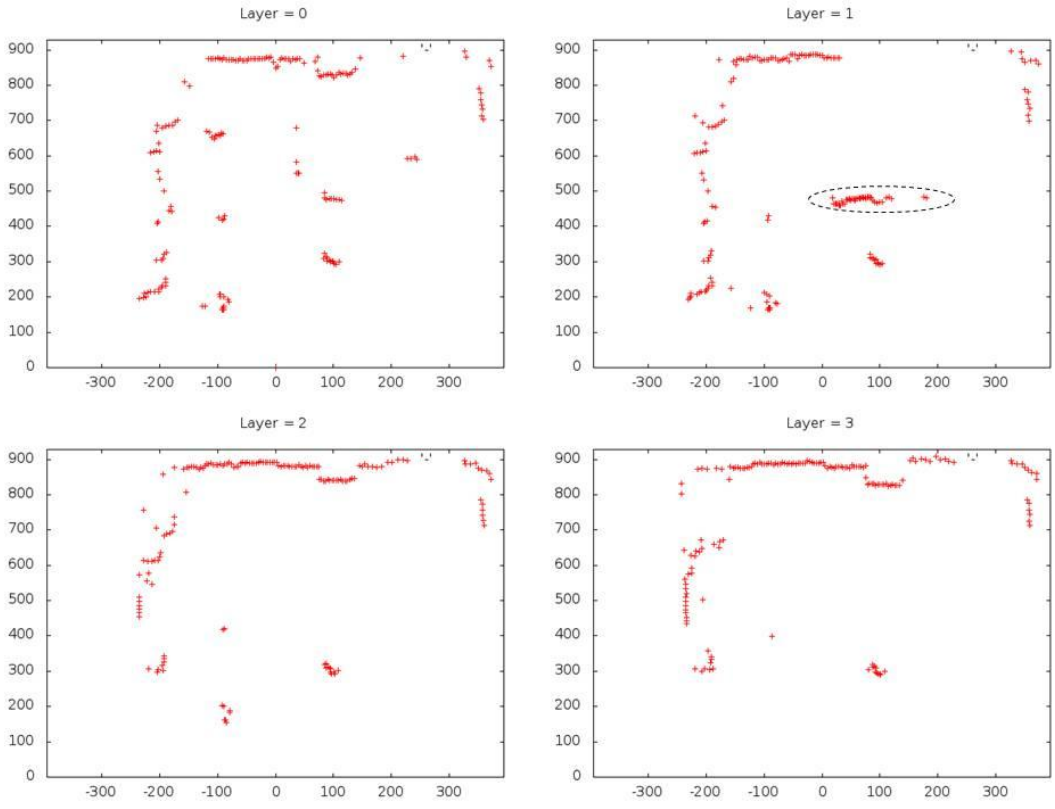


Figure 6: Pole at height of 1.09m can be seen in layer 1

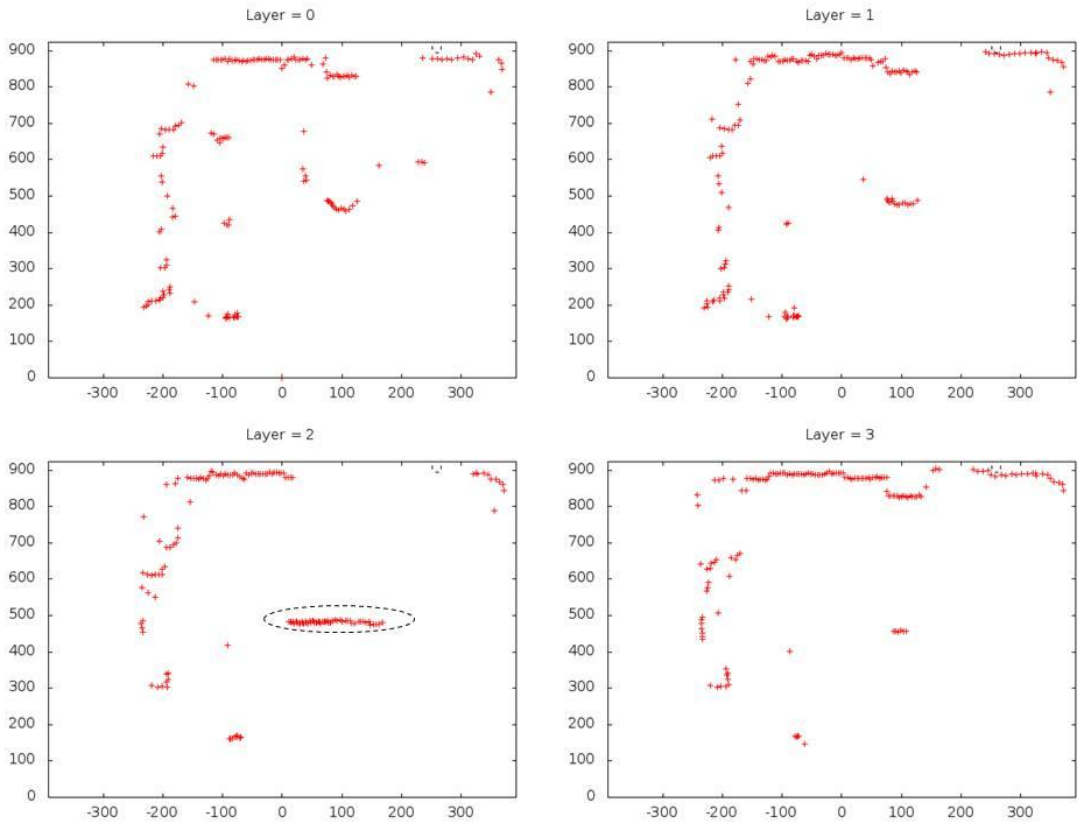


Figure 7: Pole at height of 1.17m can be seen in layer 2

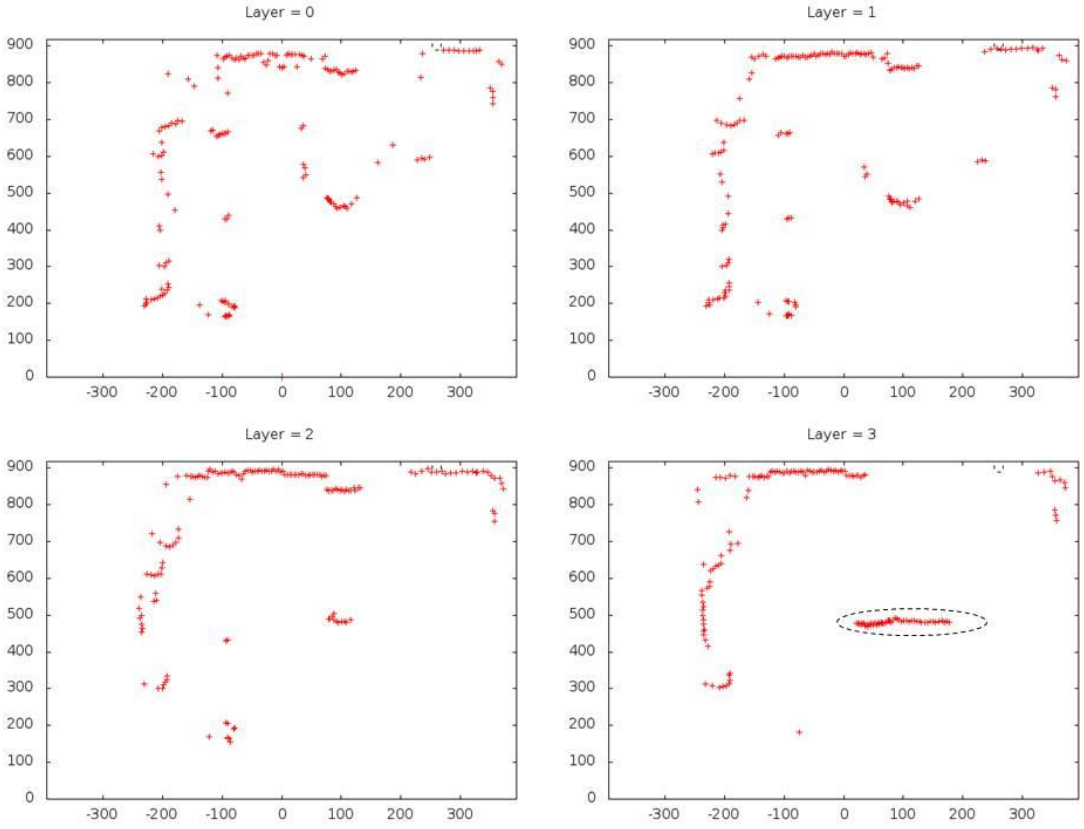


Figure 8: Pole at height of 1.25m can be seen in layer 3

Using the height of the pole visible in each layer, the angle of the each layer was calculated and the results can be seen in table 1.

Layer	Height of Pole (m)	Angle
0	0.97	2.9°
1	1.09	1.58°
2	1.17	0.65°
3	1.25	-0.29°

Table 1: Heights and angles of the layers of the IBEO sensor

4 PROGRAMMING FOR OBJECT DETECTION

Creating a program to detect and follow objects in front of the SAE Autonomous car involved multiple parts. Firstly, data from the IBEO lux sensor needed to be obtained and analysed to produce useful data that could be plotted. Using the data, an algorithm needed to process the data to determine which points obtained were objects and by constantly looping through the algorithm the object needs to be tracked outputting information about the change in position of the object. Using this information commands are sent to the low level control which in turn controls the car. Below is a description of the various sections of the program.

4.1 OBTAINING DATA FROM THE IBEO LUX SENSOR

For this project, Ethernet will be used as the method of connection between the IBEO LUX sensor and the computer. The default Ethernet configuration of the IBEO sensor is as below:

Default IP address: 192.168.0.1
Subnet Mask: 255.255.255.0
The default port: 12002

To obtain data from the IBEO, a program written by Timothy Black a former student from the University of Western Australia who has previously worked with the IBEO sensor and a main function written by Thomas Drage which has been modified to change the order information is received and processed.

The program written by Black works in conjunction with the header file "IBEO.h". This needs to be included in the program for the program to work. Blacks program is a set of functions that starts and stops communication between the IBEO and the connected device, reads the data outputted by the IBEO and stores it in various arrays.

In order to obtain data from the IBEO the following order of functions must be called in the 'main' function.

1. A new instance of the IBEO object must be created through the use of the function *new IBEO()*;
2. Communication between the IBEO and computer is started by calling the function 'StartScanner'
3. Data from the IBEO can be obtained by calling 'ReadMessages'

Every time the function *ReadMessages* is called, new data from the IBEO is obtained and stored in the corresponding array. Therefore when new data is required, calling *ReadMessages* will update all the arrays with the most current information received from the IBEO.

The scan data points are stored in the array *scan_data_point*[]. To obtain all the points from the IBEO, a for loop from 0 to the total number of scan points is used (The total number of scan points can be accessed from the variable *scan_data_header[Lux->curScanDataSource].scan_points*). Within the loop, each member of the structure SCAN_DATA_POINTS can be accessed by Lux->scan_data_points[Lux->curScanDataSource][i].*memberName*. The structure SCAN_DATA_POINTS has the following members with their corresponding type and size.

Member	Data Type	Size (Bytes)
Layer	UINT4	1*
Echo	UNIT	1*
Flags	Bit field 8 bits	1
Horizontal Angle^	INT16	2
Radial Distance	UNIT16	2
Echo Pulse Width	UNIT16	2

Table 2: Members of the structure SCAN_DATA_POINTS (IBEO 2009)

* The member's layer and echo share one byte. To access the layer, the low nibble/bits 0 to 3 of the byte need to be used. The echo value can be obtained by using the high nibble/bits 4 to 7 of this byte.

^The horizontal angle is given in angle ticks. One rotation is equal to 11520 ticks. Therefore to convert the angle from ticks to radians:

$$angle\ (rads) = 2\pi\ \times \frac{angle\ ticks}{11520} \tag{1.2}$$

4.2 PLOTTING DATA POINTS

In order to make sense of the information received from the IBEO sensor the data needs to be plotted to create a visual representation of what the IBEO sensor is seeing.

For the initial tests, the data received from the IBEO sensor is saved in files with the extension ".lux". For the first few tests only the horizontal angle and radial distance was recorded for each data point. The program Microsoft excel was used to initially plot the points. In order to open the .lux files in Microsoft Excel the extension needed to be changed to .csv. Using Microsoft Excel the horizontal angle was converted from ticks to degrees and using the radial distance and angle, all the data points were converted to the Cartesian coordinate system. Using the Cartesian coordinates a scatter plot was made which can be seen below in figure 9.

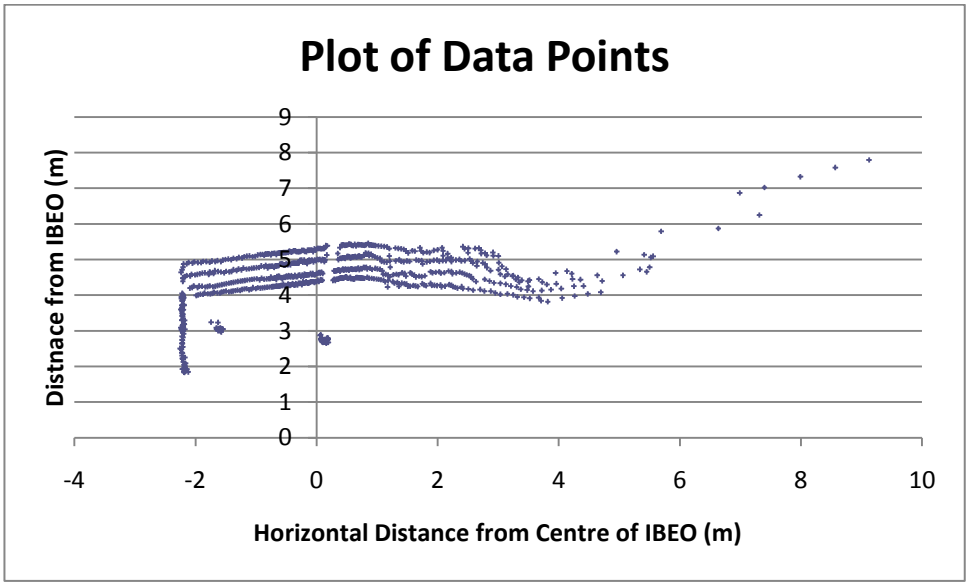


Figure 9: Microsoft Excel plot of the points obtained from the IBEO sensor

Looking at the plot, there appears to be a shadow of four horizontal lines from between a horizontal distance of approximately -2m to 3m. This is because the layer of the data point hasn't been taken into account and all the data points from all four layers have been plotted. Between -2m and 3m there is an object that is large enough so that it is seen by all three layers.

Using Excel to initially plot the data points was useful to have an idea of what the information from the IBEO meant but it is impractical in the long run as it is too time consuming and does not automatically generate a plot for every set of data points recorded.

A better option to plot data points was to use a program called gnuplot. Gnuplot is a "freely distributed source code that is a portable command-line driven graphing utility for Linux, OS/2, MS Windows, OSX, VMS and many other platforms." (Gnuplot 2013)

In order to use Gnuplot, the program must be installed on your device which is available from http://www.gnuplot.info/download.html.

Gnuplot can be used to plot specific files through the terminal or command prompt but for convenience, Gnuplot is called directly in the program. Using an example program written by Ben Blumer on the forum site stackoverflow, gnuplot is called from the C program and is able to plot points stored in an array in the C program. Blumer achieves this by using a pipe, a method to redirect information from one process to another.

To create a new pipe the file pointer needs to be initialised using the function popen. The function popen takes in two arguments. The first is the command argument which is a string containing a shell command line and the second is the mode which can either be 'r' for read or 'w' for write. (die.net n.d.)For this program the command is "*gnuplot*" and the mode is 'w'. Once a pipe has been created commands can be written through the pipe to gnuplot using the function fprintf with the first argument being the name of the file pointer and the second argument being the command sent to gnuplot.

Gnuplot features the capability to customise plots including:

- Adding a title – "*set title \"TITLE\" \n*"
- Setting the range of the x and y axis – "*set xrange [0:10]\n*" to set the x range from 0 to 10

Note that after every command sent a new line character (\n) needs to be sent otherwise an error will occur. To finally plot the points the command "*plot '-'\n*" needs to be sent followed by the values of the x and y coordinates separated by a space and ending in a new line character.

In order to plot the data points received from the IBEO sensor they first need to be stored in arrays. Instead of storing the raw data, the array consists of x and y coordinates for each point. To achieve this, a structure called point has been created which contains two floats xdist and ydist. The xdist and ydist is calculated by converting the horizontal angle from ticks to degrees and multiplying the radial distance by the sin and cos of the angle in degrees. In order to scale the plot appropriately a loop has been created that scans through all the points finding the largest value for x and y in absolute value and setting the x and y range to equal the maximum value plus an extra twenty. The final plot is created by looping through all the x and y coordinates of the array and sending them through the pipe to gnuplot after the "*plot '-'\n*" command is sent.

To enable the user to plot single layers from the IBEO, a new member was added to the structure which contains the layer information of each data point. Using the layer member of every point a conditional statement was added within the loop of x and y coordinates that meant that only points with the same layer as the user defined layer are plotted.

Another function called "plotcolour" was created which plots each layer in different colours and would later be used to plot each object in different colours. To achieve this, the command "*set multiplot*" is used which enables the user to have multiple plots appear on a single window. Styles are initialised for each colour that will be used using the command "set style line". For example to initialise line style 1 to be of thickness 1 and line colour red, the following command would be sent "*set style line 1 lt 1 lc rgb \"red\"\n*". This is repeated for the four colours used; red, blue, green and orange. To plot the various layers in different colours the line style is added to the plot command making the overall command "*plot '-' ls 1 \n*" to plot in the colour defined by set style line 1. The plot command needs to be repeatedly called before a new set of points for a different layer is plotted.

Another useful feature of Gnuplot is the ability to generate plots that are directly saved to file. Gnuplot does not allow the user to save plots generated in a window so the

following command can be used if plots are to be used later in the future. The first command is "*set output png* \n" which forces the output file to be a .png file and the second command is "*set output* \"*home/data.png*\"\n" which will save the plot under the file name data.png in the home file.

4.3 FINDING AN OBJECT

Using the point data from the IBEO sensor, an algorithm was created to identify objects within a plot. The basic idea of the algorithm is that objects will be comprised of multiple points and points that are close enough to adjacent points will be part of the same object. Not only was an algorithm created but a new structure was created to store information about each object found and the point structure was also edited to now contain information about which object it belonged to. The plotting function was also edited to display each object in a different colour. Two different methods were used to identify objects and these will be explained in this section.

4.3.1 OBJECT STRUCTURE

A new structure called *object* was created to store various information about each object indentified in the *findobject* functions. The members of the structure object include:

- xstart – The x coordinate of the point furthest right in an object
- ystart – The y coordinate of the point furthest right in an object
- xfinish – The x coordinate of the point furthest left in an object
- yfinish – The y coordinate of the point furthest left in an object
- length – The length of the object calculated using the xstart/finish and ystart/finish.
- xmidpoint – The x coordinate of the middle of an object
- ymidpoint – The y coordinate of the middle of an object
- numpoints – The number of points that make up an object.

To store the information of all the objects found in a plot, an array of *objects* is used. A new member called *object* was added to the *point* structure which contains the array index of the object the point belongs to.

4.3.2 OBJECT DETECTION ALGORITHM 1

In this first algorithm, objects were identified by scanning through all the points saved in the points array in increasing index order. To determine whether or not two points in consecutive array entries were part of the same object, the points were required to have x coordinates and y coordinates within a user defined *POINTDIST*. If this returned true and the current index of the object array had zero points, the xstart and ystart members are assigned and the *numpoints* of the object is incremented. If the current index of the object array is already comprised of points then only the *numpoints* member is incremented. This continues to happen as the function loops through all the points until two consecutive points are greater than *POINTDIST* apart from each other, either in the x or y direction. When this occurs an if statement checks if *numpoints* of the current object index is greater than a user defined variable *smallestobj*. If this is true, the xfinish and yfinish of the object is equal to the previous points x and y coordinates and the index of the object array is incremented. If it is not true the *numpoint*s of the current object index is set to zero which means that in the next loop, if a new object is found the xstart and ystart data will be overwritten. This has been done to prevent random adjacent points being considered objects. A flow chart of the algorithm can be found in figure 10.

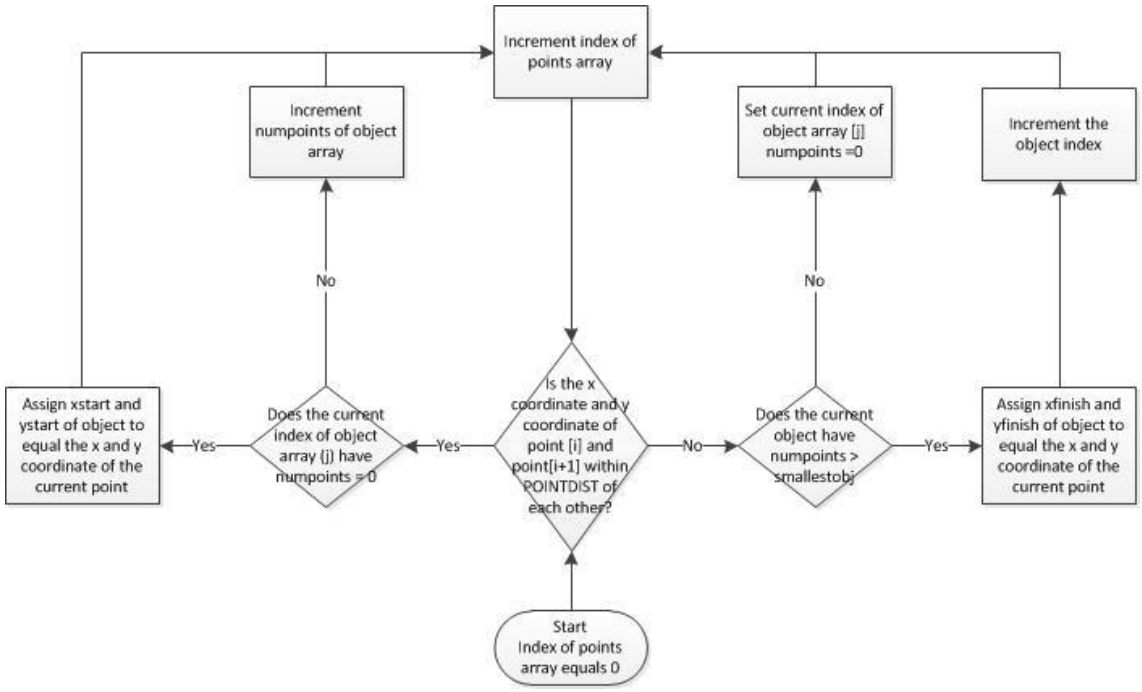


Figure 10: Flow chart of algorithm 1

However in initial tests not all the objects in the plot were identified by the program. Looking at the raw data obtained from the IBEO points were obtained from the far right to the far left with decreasing horizontal angle. This meant that in the overall plot it appeared that the points went from largest to smallest x coordinate but when looking up close at certain groups of points this was not the case.

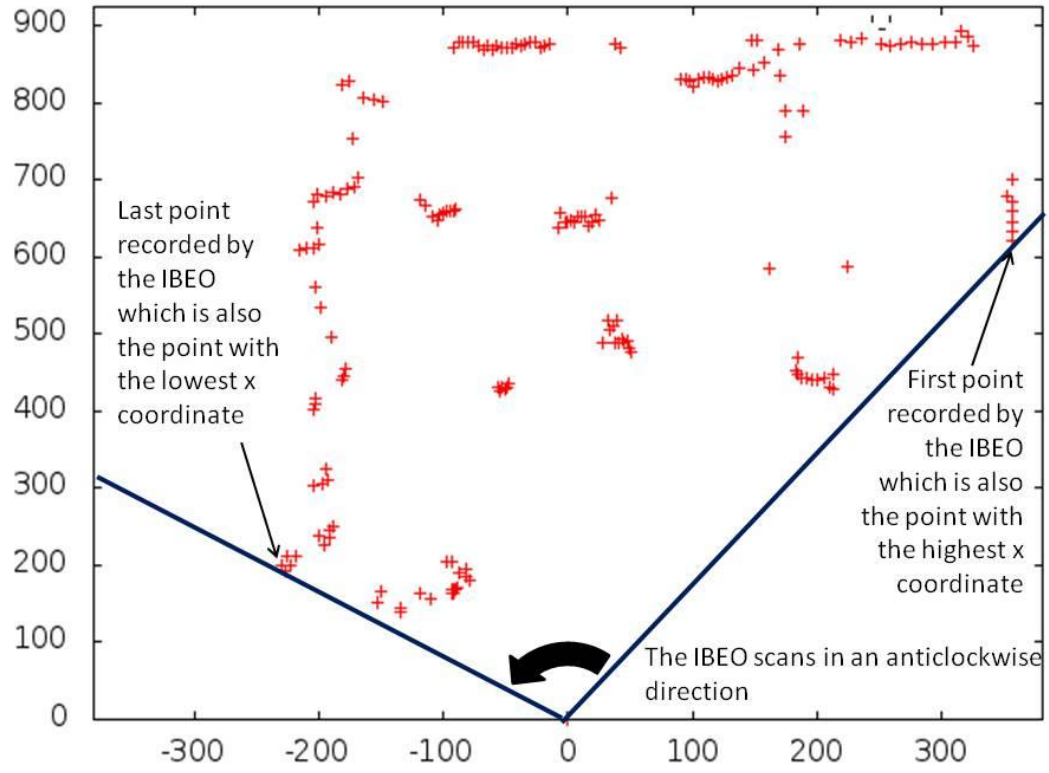


Figure 11: Plot of points from IBEO sensor showing the direction the points are scanned

Looking at figure 12, the left plot shows the order points would be scanned by the IBEO. When used in the 1st algorithm, points 1 and 2 would be part of the same object but point 3 would not be, as it is greater than *POINTDIST* away from point 2 in the x direction. However, if the points were ordered in decreasing x coordinate as shown in the right plot, all three points would be part of the same object as they are all within *POINTDIST* of each other.

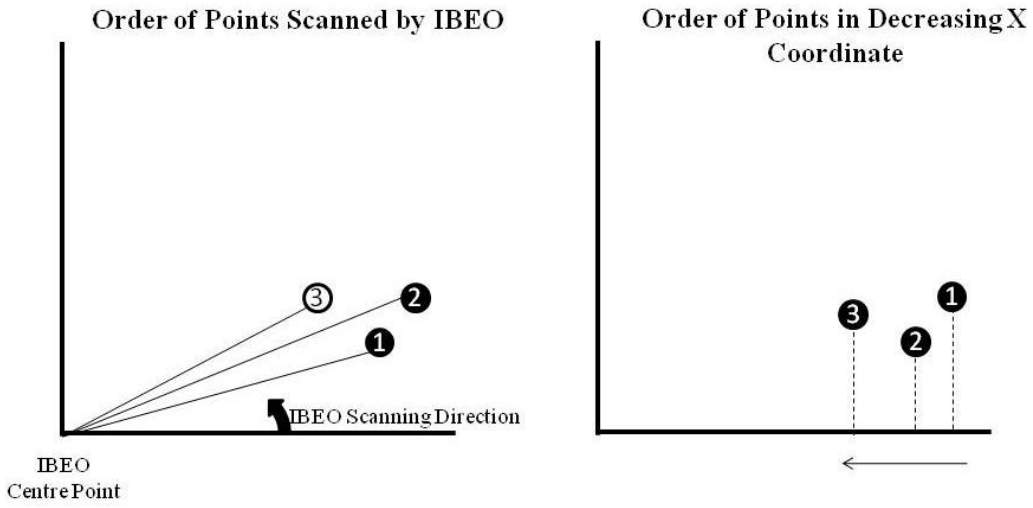


Figure 12: The order points are scanned by the IBEO sensor and the order of points if sorted by decreasing x coordinate

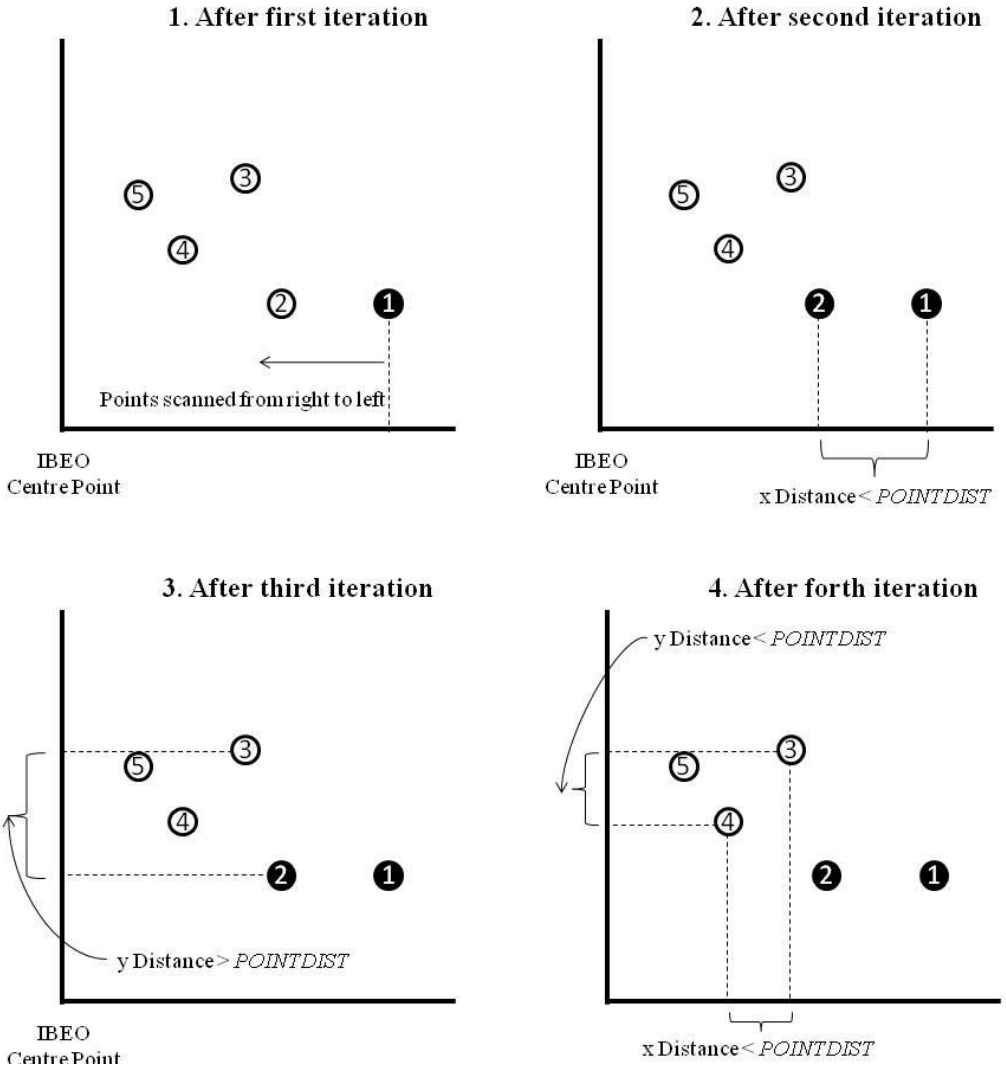
To solve this problem all the points were sorted from highest x coordinate to lowest x coordinate before being passed to the *findingobject* function. The sorting algorithm used is a quick sort and follows the example taken from mathbits.com. According to mathbits.com, quick sort is a quick and efficient way to sort arrays. (mathbits.com 2013) This is done be recursively splitting the array into partitions and sorting the partitions with numbers above the pivot point on one side and numbers below the pivot point on the other side. An example of how the quick sort algorithm works is shown below.

	Element	1	2	3	4	5	6	7
1. First Element is chosen as the pivot point and moved to the end	Original Array	8	5	10	15	7	2	11
2. Starting at element 1, the pointer i moves right until it reaches a value higher than the pivot. Starting at element 6, the pointer j moves left until it reaches a value lower than the pivot When they both stop moving the two values are switched	After step 1	11 i→	5	10	15	7	2 j ←	8
3. This continues until i passes j	After step 2	2	5	10	15	7	11	8
4. The pivot point is placed back in to the	After step	2	5	7	15	10	11	8

	middle of the array	3				i j			
5.	The array is split into 3 groups and steps 1 to 4 is repeated for each group	After step 4	2	5	7	8	15	10	11

Table 3: Steps of the quick sort algorithm

Now that the array of points are ordered from highest x coordinate to lowest x coordinate more objects were identified however another problem became obvious. Larger objects were being split into multiple smaller objects and therefore the number of identified objects was higher than the number of objects present.



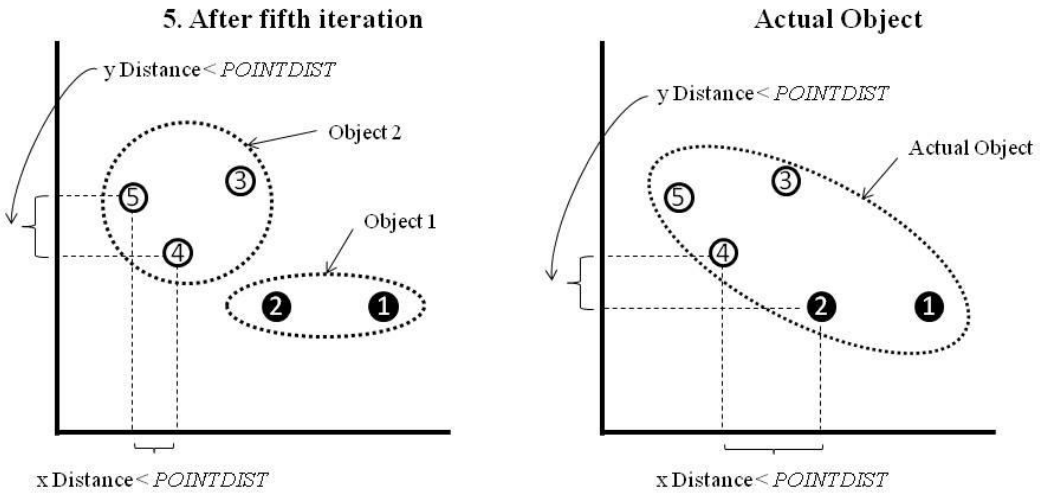


Figure 13: Pictorial representation of algorithm 1

1. Points are scanned from right to left, from the highest x coordinate to the lowest x coordinate. When a point is not within *POINTDIST* of the previous point a new object is created.
2. As the distance between point 1 and point 2 is within *POINTDIST* in both the x and y direction point two becomes part of the same object as point 1.
3. The y distance of between point 2 and point 3 is greater than *POINTDIST* and therefore point 3 is not part of the same object as point 1 and 2. Point 3 becomes part of a new object
4. Point 4 is within *POINTDIST* of point 3 and becomes part of the same object as point 3
5. Point 5 is within *POINTDIST* of point 4 and becomes part of the same object as points 3 and 4. Therefore 2 objects have been formed, one containing points 1 and 2 and another containing points 3, 4 and 5. Please note that only five points have been shown in this example. The objects would be made up of many more points and objects with less than *smallestobj* would not normally form objects.
6. When you look at the distance between point 2 and point 4, both the x and y distance is within *POINTDIST* and therefore should be part of the same object. However as the points are scanned from left to right, points 2 and 4 are never compared to each other. This means that the single object that is made up of five points is split into two separate objects.

Due to this problem, algorithm 1 could not be used and a new algorithm was created.

4.3.3 OBJECT DETECTION ALGORITHM 2

The basic idea of object detection algorithm 2 is to find all the points within an arc of radius *POINTDIST* and make them all part of the same object. By identifying objects this way, it eliminates the problem from algorithm 1 where objects were identified by adjacent points. Similar to algorithm 1, all the points are ordered from highest to lowest x coordinate. In the program a for loop loops through all the points, but unlike algorithm 1, there is another loop for each point that looks at all the points until the difference between the two points "x distance" is larger than *POINTDIST*. Figure 14 is a flow chart of how algorithm 2 works.

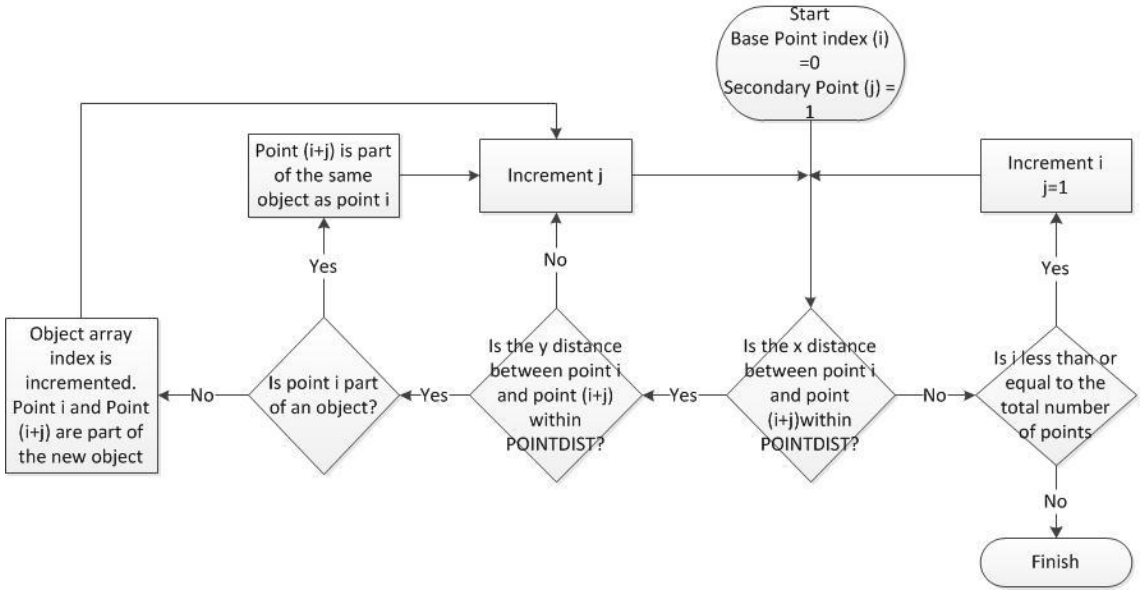


Figure 14: Flow chart of algorithm 2

After the algorithm has run through all the points, the objects that have less than *smallestobj* number of points are removed from the array of objects. This is done by looping through the objects array, finding objects that have less than *smallestobj* number of points and removing the object from the array by storing the next object at the current index. The object member in the points array also needs to be changed and so another for loop, loops through all the points changing the object member to 0 if it equals the index of the object being removed. The variable for the total number of objects is also decremented.

The members of the object structure are also assigned after the algorithm and not during. The start and ending points are found by looping through all the points and finding the point with the highest x coordinate (starting point) and the point with the

lowest x coordinate for each object (finishing point). The length of each object is also calculated using the starting and finishing points and Pythagoras' theorem. The midpoint of each object is also calculated using the starting and finishing points.

By using algorithm 2 instead of algorithm 1, objects were identified more consistently and accurately.

4.3.4 TESTING OF ALGORITHM 2

To test the object detection algorithm the car was taken outside the lab with objects placed in front of it. Below in figure 16 is a colour plot of the image shown in figure 15. The x and y distance of the objects was measured using a tape measure and these have been compared to the distances calculated by the algorithm which is shown in table 4.

Figure 15: Objects placed in front of the Autonomous SAE Car

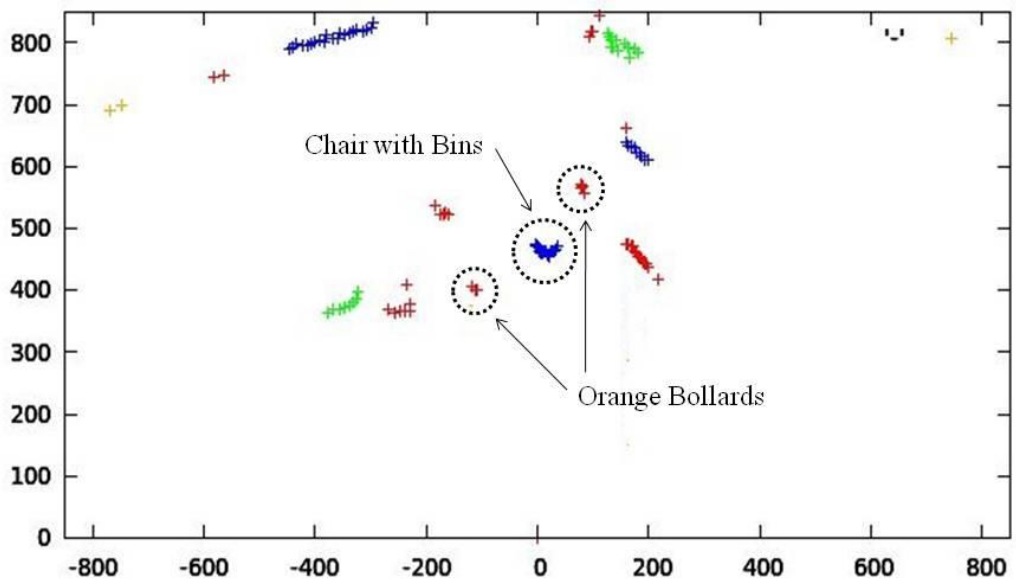


Figure 16: Plot of the objects in front of the Autonomous SAE car as shown in figure 15

	Program Distance (cm)		**Actual Measured Distance (cm)**		**Difference**	
Object	x midpoint	y midpoint	x midpoint	y midpoint	x midpoint	y midpoint
Left Bollard	-94	378	-100	380	6	2
Chair	15.5	473.5	20	450	4.5	23.5
Right Bollard	86.5	545.5	90	560	3.5	14.5

Table 4: Program distance and measured distance of objects seen in figure 15

Looking at the results above, it can be shown that the program is accurate to within 24cm. The largest difference was the y midpoint of the chair at 23.5cm. These are very good results as the IBEO scanner specifications state an accuracy of up to 10cm.

4.4 SELECTING OBJECT TO FOLLOW

When the program is executed, the object detection function is run a single time and a colour plot of the objects identified is displayed in a new gnuplot window. The data for

each object identified is also shown in the terminal window and allows the user to select which object they would like to follow.

Once the user has selected an object they wish to follow, the data for the specified object is assigned to a variable called fobject which is of the same type as the object structure. Within the main loop of the program, objects are identified again and the information is stored in the objects array and overwrites the previous object data. However as the information of the object the user has chosen to follow is saved in the variable *fobject*, the new objects identified are compared to *fobject* to find the same object.

To find the object the user has chosen, a function called *findselected* has been created. The function loops through all the new objects and compares the length, xmidpoint and y midpoint to *fobject*. To be considered as the same object it must have a length within 10cm of *fobject* and an xmidpoint and ymidpoint within 100cm of *fobject*. The x and y midpoints are used to prevent the program from choosing an object of similar length that is far away from the original object. The function returns the index of the object that it has identified as the same object or zero if no object that matches the criteria is found. If a zero is returned, the program continues to run in the main loop but increments a variable called *notfound* until *notfound* exceeds a user defined value called *NUMBEROFCYCLES*. Once this occurs the program terminates with an error message "object not found again".

However if the object is found the *notfound* variable is reset back to zero and the acceleration and steering variables are evaluated. The new information about the object being followed is then saved as the *fobject* and the program loops to find the object again.

4.5 SETTING THE ACCELERATION AND STEERING VALUES

The acceleration and steering values are calculated in two separate functions. The acceleration function is called *setAcceleration* and requires two input arguments. These are the y midpoint of the object in the previous loop (which is saved under *fobject*) and the y midpoint of the current loop. For this reason, the acceleration function must be run before the members of the new object in the current loop are saved to *fobject*. The *setAcceleration* function is a PID controller (Braunl 2012). For safety measures a new user defined variable *MINDIST* has been added which represents the distance from the

car that full brakes is applied. This is to ensure that the car will never come within a set distance of the object it is following. The error function in the PID loop is the difference between another user defined variable *SETDISTANCE* and the y midpoint. The *SETDISTANCE* is the distance the user would like to maintain between the car and the object it is following. If the output of the PID loop is negative then the brake is set to 0 and the acceleration is set to the minimum value between the absolute value of the PID output or 255. The acceleration and brake must be limited to 255 as the low level control will only accept a value between 0 and 255. If the output of the PID loop is positive then the acceleration is set to 0 and the brake is set minimum value between the output of the PID and 255.

The steering value is calculated in the function *setSteering* and takes in two arguments, the *xmidpoint* and *ymidpoint* of the new object. Using these two values the angle of the object relative to the vertical axis along the middle of the car is calculated. In a previous experiment the range of the steering was determined to be 20° at full left and full right. The lower level control requires a value of -128 for full left and +127 for full right therefore the steering value $= \frac{\text{angle}}{20} * 127$. The steering value is then passed through a min and max function to ensure that the steering value is either at a maximum of 127 or a minimum of -128.

4.6 SENDING SERIAL COMMANDS TO LOWER LEVEL CONTROL

Communication between the object detection program and the lower level control is achieved through serial. The library boost.asio was used in conjunction with the program BufferedAsyncSerial written by Terraneo Federico and available from http://www.webalice.it/fede.tft/serial_port/serial_port.html. Using Federico's program is very simple and allows the user to send and receive messages through serial from the device. For this application the device that will be receiving the messages is an arduino. To open the serial connection the function *BufferedAsyncSerial serial ("/dev/ttyACM0",9600)* is executed with "*/dev/ttyACM0"* is the device name of the serial port and 9600 is the baud rate. To ensure that communication is successful the baud rate of the computer and the arduino must be the same. Once the serial connection is open it remains open until the program is terminated.

To send commands through serial to the ardunio, the function *serial.writeString("string")* is used. The arduino accepts three commands and these are

A#\n for acceleration at a value of "#", *B#\n* for brake at a value of "#" and "*S#\n"* for steering at a value of "#". To read data sent back from the ardunio which includes the current steering, acceleration and braking values and any error codes the following code is used *cout<<serial.readStringUntil("\n")<<endl*. For safety purposes the arduino must receive a command from the program every 0.3 seconds otherwise the emergency brake is applied. This is to ensure that the car does not continue to run if the program crashes or connection between the arduino and computer is is no longer present. When the program is terminated *serial.close()* is executed to close the serial connection.

5 IMPLEMENTING THE OBJECT DETECTION PROGRAM WITH THE AUTONOMOUS SAE CAR

The implementation and testing of the object detection program with the autonomous SAE car was completed in three stages. This was to ensure that the program worked and that the car, the surroundings and the people involved were not in any danger.

In order to implement the object detection program on the autonomous SAE car all the programming files including the source code and header files were copied to the laptop attached to the car. For the first stage of testing, the lines of code which sent commands to the lower level control through serial were commented out. Instead, these values were displayed in the terminal for debugging purposes. To test the program, a person was selected to be tracked by the program. For the first test, the person walked laterally in front of the car with the steering values watched on the terminal to ensure that the correct values were being outputted by the program relative to the position of the person at that point in time. The second test involved checking the longitudinal control with the brake and acceleration values. A distance of 3m was used as the *SETDISTANCE* and it could be seen that as the person passed a distance of 3m the acceleration value increased and as the person started to walk within a distance of 3m the brake value started to increase as the acceleration remained at 0.

As the values of all three parameters were as expected, the second stage involved testing the program with the low level control switched on. In order to do this, the serial commands were uncommented and the car was raised on 4 axle stands. In order to run the program with the lower level control on, the following procedure needs to be followed:

1. The power button (Big red button behind the driver's seat) needs to be pulled up into the on position.
2. The acceleration toggle switch must be in the autonomous acceleration position
3. The autonomous switch needs to be pulled up in the on position
4. The program is to be executed in the terminal with "sudo" in front of the execution line. If the program is not run in "sudo" mode the program will not be able to access the serial port.
5. The layer at which the user wishes to scan at is selected. (For objects up 20m away, either layer 0 or 1 should be used)
6. Turn on the motor controller
7. The terminal displays the objects it has detected and the user selects the object they wish to follow
8. Once the object has been selected the program will start and the steering, acceleration and brake will adjust depending on the location of the object it is following.
9. Program continues to run until any of the switches are turned off or the program losses the object it is tracking.

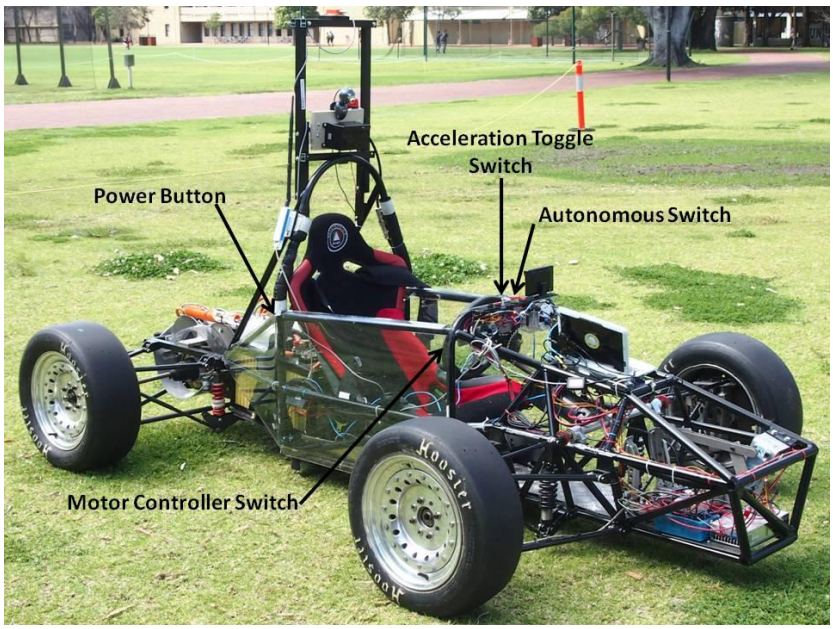


Figure 17: Location of the buttons and switches on the autonomous SAE car used to run the program

Similar to the first stage of testing, a person was used as the object to track. With commands being sent to the lower level control it could be observed that the steering responded to the movement of the person. As the person moved laterally it could be

observed that the angle of wheels changed to reflect the position of the person. Longitudinal movements affected the brake position and the acceleration of the wheels. Initially when the program was executed the person was within the *setdistance* and therefore the brake was on full, but as the person walked backwards away from the car it could be seen that the brake slowly released until the person was the *setdistance* away from the car. At this point the brake was fully released and the rear wheels started to spin. This proved that the commands sent to low level control were successful and the reactions of the car reflected the person's location at a given point in time.

The third stage of testing involved taking the autonomous SAE car out to the chemistry oval and running the program to track a person walking in front of the car. For safety reasons the speed of the car was limited to 10% of the maximum speed and three people were always present for each test. One person was used as the object to track while another person followed along the car ready to turn off the power in case of an emergency. The third person was positioned in the driver's seat allowing them to execute the program but also manually override the brake if the car became too close to the person in front. The *setdistance* for these tests was 5m and the starting position of the person was 4.5m. The test was a success and the car was able to track and follow the person walking in front of the car. However a couple of problems observed during the test were the jerkiness of acceleration and braking and the inconsistent distance between the car and the person followed. As the person walked away from the car, the car would accelerate to maintain the set distance, but the acceleration was too high and the car would need to brake as it became too close to the person. The car was unable to maintain a constant speed to prevent the jerkiness of braking and accelerating.
An experiment was conducted to investigate the distance the car maintained when transitioning from acceleration to brake. The experiment involved having a person at a set distance of 4.5m from the IBEO sensor, executing the program to track the person, having the person walk 4 paces backward allowing the car to accelerate to maintain the set distance, and then brake once it had reached the set distance. Figure 18 shows a graph of the distance between the IBEO sensor and the person in front of the car as recorded by the program in one of the tests.

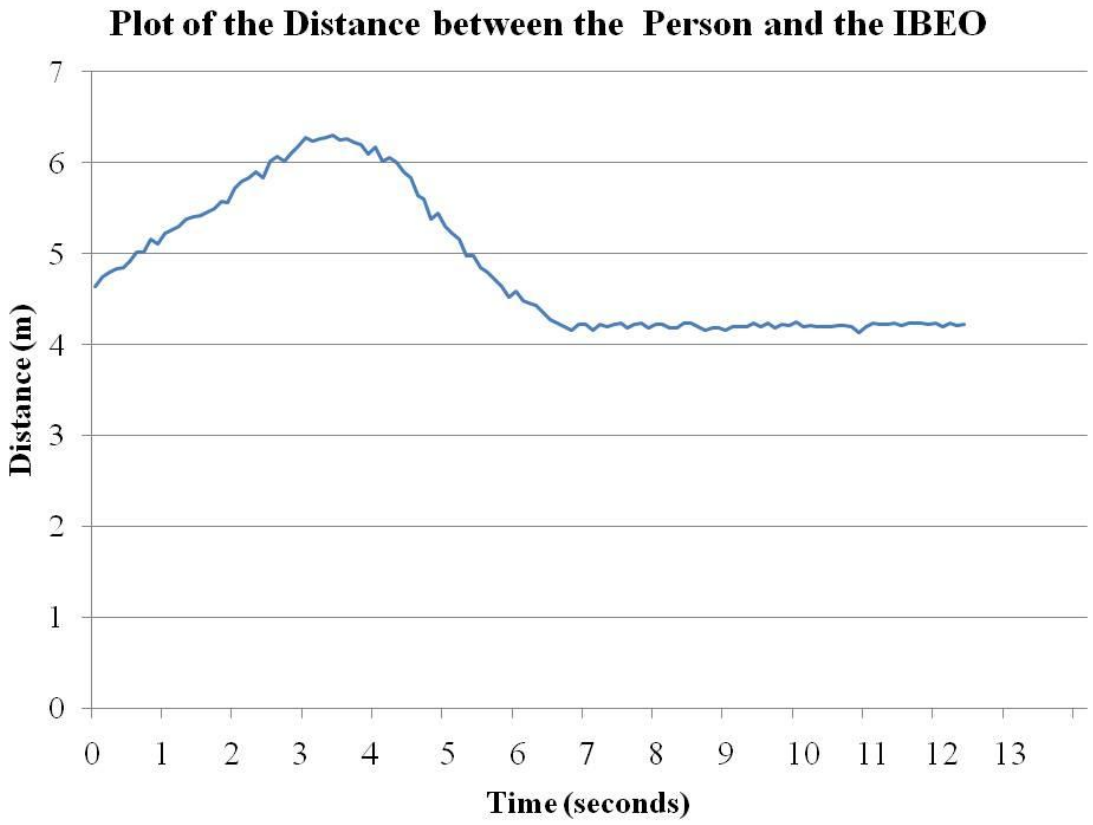


Figure 18: Plot of the distance between the person in front of the autonomous SAE car and the IBEO sensor over time.

From the graph it can be seen that the distance between the IBEO sensor and the person reached a maximum of 6.3m after 3.5 seconds and took approximately 3 seconds to reach a final distance of 4.2m. Even after the car passed the set distance of 5m it continued to move closer to person for 1.5 seconds and a distance of 0.8m. This is due to the momentum of the car and fact that even though the brake is being applied, it is not applied enough to cause the car to come to a complete stop. The experiment was repeated 10 times with the final distance between the IBEO sensor and the person in front of the car measured with a tape measure and recorded by the program. Table 5 shows the results of the experiment.

Test Number	Measured Distance (m)	Distance from Program (m)	Difference
1	4.40	4.54	0.14
2	4.49	4.58	0.09
3	4.21	4.15	-0.06
4	4.79	4.83	0.04
5	4.66	4.61	-0.05
6	4.99	5.00	0.01
7	4.08	4.08	0
8	4.12	4.22	0.1
9	4.15	4.22	0.07
10	4.27	4.45	0.18
Average	4.416	4.468	0.052

Table 5: Results from testing the stationary distance of the person to the IBEO sensor

On average the car drove 0.584m past the set distance of 5m. To improve this value more work needs to be done on calibrating the PID controller and changing the sensitivity of the brake and accelerator. However, the average between the measured distance and the distance from the program was only 0.052m (5.2cm) which means the object detection algorithm is working as intended.

6 USER INTERFACE FOR THE AUTONOMOUS SAE CAR

A user interface has been designed and installed which will allow the autonomous SAE car to be fully self-sufficient and be able to run without the use of a laptop connected wirelessly. This will be very beneficial when doing demonstrations as all commands can be executed from the driver's seat. The interface consists of two push buttons, two LEDs and a screen connected to a Raspberry Pi.

6.1 EXTENDING THE CURRENT DASHBOARD

Before adding the new interface to the car, a smaller dashboard had been created which featured an emergency stop, four 5 safety LEDs, a toggle switch to switch from manual

to autonomous acceleration and a push button to arm the safety. With all these components on the current dash it left very little room to add the new buttons and LED. Mounting the screen was also another problem as there was no flat surface to mount the screen on to. As all of the components of the old dashboard had been hard wired it would be very difficult to remove the current dash and replace it with a longer one.

To overcome this issue a new plate was designed that would fit over the current buttons, switches and LED and would be affixed by the nuts holding the switches down. In order to exactly copy the outline of the current buttons and LEDs a piece of ply carbonate was used to trace the outline of the current dash and make a template which would be transferred to the new metal plate. The metal plate was constructed out of aluminium as it is light and easy to work with.

6.2 CIRCUIT DIAGRAM FOR BUTTONS AND LEDS

In order to connect the push buttons to the Raspberry Pi the following circuit was used.

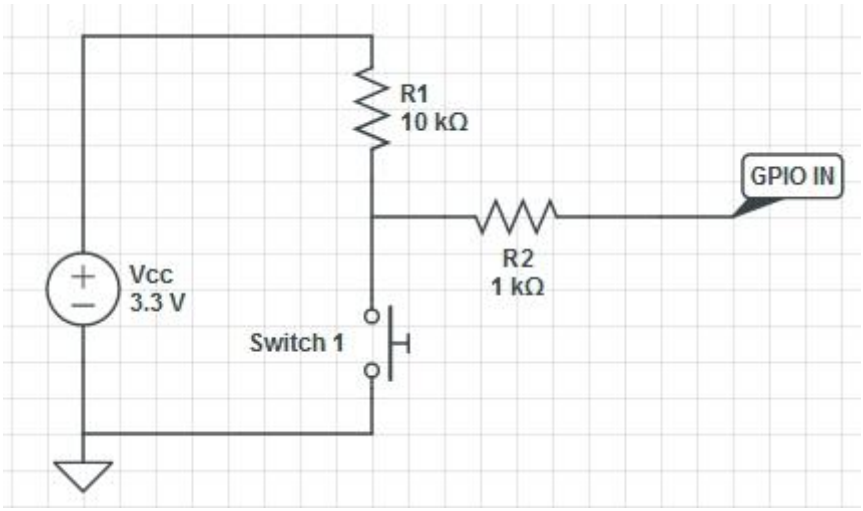


Figure 19: Circuit Diagram of the push button (Andrade et al. 2012)

In this configuration, a pull up resistor of 10kΩ has been used. This means that the GPIO input pin will read high while the switch is open and will read low when the switch is pressed as there will be a path to ground. A current limiting resistor of 1kΩ resistor has been used to ensure that the raspberry Pi can handle the current drawn. The raspberry Pi's positive rail is the voltage source of this circuit and supplies 3.3V.

The following circuit is used to connect the LEDs to the GPIO pins of the raspberry Pi.

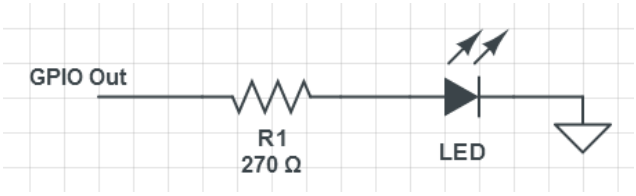


Figure 20: Circuit to connect LED to the Raspberry Pi's GPIO pin

A current limiting resistor of 270Ω was used to prevent damaging the LED.

6.3 RASPBERRY PI GPIO PINS

The raspberry Pi features General Purpose Input and Output (GPIO) pins which allows the user to connect external hardware to the raspberry Pi. The raspberry Pi gives access to 8 GPIO pins as well as connection to a +3.3V and ground supply.

In order to connect to the GPIO pins of the raspberry Pi I have used a pin header and soldered wire to the pins I will be using. These pins include:

- +3.3V supply located on Pin1
- Ground supply located on Pin 6
- GPIO pin 17 and GPIO pin 21 located on Pin 11 and 13 respectively for the LEDs
- GPIO pin 16 and GPIO pin 18 located on Pin 16 and 18 respectively for the Push Buttons

The pins are counted from left to right, up then down with pin 1 marked by P1. To ensure that the remaining pins that are not used are not short circuited, heat shrink has been placed over each pin.

6.4 PROGRAMMING THE RASPBERRY PI TO READ FROM GPIO PINS

In order to program the raspberry Pi to use the GPIO pins WiringPi needs to be installed. This is available from http://wiringpi.com/. WiringPi is a library written in C that that can be used in programs written in C and C++.

To use the WiringPi library #include <wiringPi.h> needs to be added to the program. At the beginning of the program the setup function "wiringPiSetup()" needs to be called before any WiringPi library functions can be called. The GPIO pins mode also needs to

be initialized by calling the function "pinMode (*pin number*, *mode*)" where mode is either INPUT or OUTPUT.

When using WiringPi the *pin number* does not correspond to the physical pin number or the pin name. WiringPi's pins 0 through to 6 correspond to the GPIO 17, 18, 21, 22, 23, 24 and 25 respectively. Therefore in the interface program WiringPi pin 0 and 2 will be used for the start/stop reecord LED and start/stop autonomous LED respectively. These pins will be set to OUTPUT mode. WiringPi pin 4 and 5 will be used for the record button and start button respectively. Appendix 1 has a summary of the pins used.

6.4.1 DEBOUNCING PUSH BUTTONS

The switches used for the interface are push buttons and these are susceptible to the problem of bouncing. This is due to moving contacts that "bounce" up and down when the button is pressed. This causes the digital output of the button to change from high to low multiple times even though the button has only been pressed once.

To compensate for the bouncing I have used a program written by Kenneth Kuhn available from http://hackaday.com/2010/11/09/debounce-code-one-post-to-rule-them-all/. In his program, Kuhn uses a variable called the "integrator" which increments or decrements depending on the output signal of the push button. Using the integrator variable, the signal must be high for a "maximum" number of times (defined by the user) before it outputs a signal of 1. This produces a clean signal that only changes from 1 to 0 when the button has been pressed.

6.4.2 COMMUNICATING WITH CONTROL PROGRAM

The purpose of the interface program is to communicate with a control program written by Thomas Drage which controls the operation of the car. To communicate with Drage's program, a pipe has been created that allows the interface program to send commands as strings through the pipe to the control program.

In the first attempt to communicate through the pipe, the function *popen* was used as according to the IEEE and the Open Group popen is used to "initiate pipe streams to or from a process". (The IEEE and The Open Group 2004) However when the program was run, a "sh:/ permission denied" error appeared in the terminal and prevented any

communication through the pipe. The permissions of the pipe were checked and all read and write permissions were for all users.

Instead of using function *popen*, *open* was used and this has seemed to work. In order to use the function *open*, a char pointer containing a string with the location of the pipe needs to be initialized. The pipe can then be accessed using the function *open(const char* filename, mode)*. The mode used in this situation will be *O_WRONGLY* which stands for write only. The function *open* returns an integer which will be used to identify the pipe when writing to it. To write to the pipe the function *write (int pipe, char* string, streamsize n)* is used, where pipe is the integer returned from the function *open*, string is the command sent to the control program and n is the size of the string being sent. After each use of the pipe, the pipe is closed using the function *close(int pipe)*.

6.4.3 STRUCTURE OF THE PROGRAM

The program comprises of four parts. The first part is debouncing the output of both push buttons which outputs a "clean" signal. The other three parts consists of code for 3 different events.

The debouncing code reads the signal received from the raspberry Pi GPIO pins connected to the push button using the function *digitalRead()*. As a pull up resistor has been used in the circuit with the push button, the output signal will change from 1 to 0 when the button is pushed. The debouncing function, therefore increments the integrator when the output of the *digitalread()* function is 0 and decrements when the output is 1. This continues to happen until the integrator reaches either 0 or 20 (the user set maximum). When this occurs a variable called *newRecOutput* or *newStartOutput* for the record and start buttons respectively changes from 0 to 1 when the integrator reaches 20 or 0 to 1 when the integrator reaches 0. Before the new output is set the old output is set to the variable *oldRecOutput* and *oldStartOuput* which will be used in the later parts of the program.

The first event that can occur is if the record button is pressed. To determine whether or not the button has been pressed the *oldRecOutput* and *newRecOutput* need to equal 0 and 1 respectively. Another two conditions have been added which include checking that the start button hasn't been pressed, as you don't want to start recording if the car is currently running and checking that the last time a button was pressed was more than a

second ago to ensure that accidentally pressing the button twice only reads as a single button press. If the button has been pressed and the car is not already recording (variable *RecPressed=0*), the control Pipe is opened, the command "STARTREC" is sent through the pipe to Drage's control program, the pipe is closed and the record LED is turned on using *digitalWrite(pin,output)* where the pin is 0 and the output is 1. If the button is pressed and the car is already recording (variable *RecPressed=1*), the controlPipe is opened again, the command "STOPREC,a.wyp" is sent, the pipe is closed, the record LED is turned off and *RecPressed* is set to 0 again.

Similar code is used in the event of the start button being pressed. The difference being the variable names and the commands sent through the pipe. When the start button is pressed and the car is not already running two commands are sent through the pipe. These are "LOADMAP,a.wyp" and "AUTOSTART". When the start button is pressed and the car is already running the command "AUTOSTOP" is sent through the pipe.

The last event is if both buttons are pressed. This may occur simultaneously or one button after the after. When this occurs, the current function that is happening is stopped either through "STOPREC,a.wyp" or "AUTOSTOP" and another command "UNTRIP" is sent to the control program. Both LED's will flash four times to indicate to the driver that the untrip command has been sent successfully. The flashing of the LED's was achieved by setting the output of both LEDs to 1, using the function *delay(500)* , then setting the output back to 0 and using the delay function again, all within a *for* loop that repeated for 4 iterations.

6.5 OVERALL INTERFACE BETWEEN CAR AND RASPBERRY PI

Future plans have been made to add a toggle switch that would allow the user to switch the control of the autonomous SAE car from following way points to drive by platoon mode. The existing buttons will be used to select the object the user wishes to follow and to execute the program.

By having a new interface, it allows the user easy access to the functions of the car and means the car can be taken to expos and conventions without the need of a complicated set up.

7 CONCLUSION

With the amount of traffic, fuel prices and green house gas emissions all increasing over the past few years, driving in a platoon is the way to go in the future. Technology is constantly evolving and with many projects like the cybercars, SARTRE and AHS it won't be long before we see platoons on our cities freeways, highways and streets.

Using the IBEO LUX sensor and the data obtained from the sensor an object detection algorithm has been created that is able to identify which points obtained from the sensor makes up objects. Using the algorithm objects have been able to be indentified to within 6 centimetres laterally and 24 centimetres longitudinally of its location. In order to have a visual representation of the objects identified, a program called GNUplot has been used to display a plot of the objects.

Using the object detection algorithm, a program has been created to repeatedly execute the algorithm and identify objects. By comparing the length and number of points of the objects, a user selected object is tracked. Using information about the movement of the object commands are sent to the low level control of the autonomous SAE car to control the steering, brake and acceleration. Using the program, the car has been successfully able to track and follow a person walking in front of the car. However, the movement of the car is jerky and future work needs to be conducted to improve this. Suggestions to improve this problem include, recalibrating the PID control and calculating the steering and acceleration values differently.

Another unresolved issue is the reliability and robustness of the object detection. Currently objects are only tracked by their length and number of points which can become a problem when two similar sized objects are placed in front of the car. Possible solutions to this problem include, integrating a camera into the object detection program and using image processing to help identify objects; using all four layers of the IBEO scanner when identifying objects instead of using just the single layer.

With these improvements hopefully one day it will be possible to test this program on a platoon of multiple vehicles at a faster speed and maybe on one of Perth's highways.

8 APPENDICES

8.1 APPENDIX 1 – INTERFACE PINS

Component	Program Name	Physical Pin on Raspberry Pi	GPIO Number	WiringPi Pin Number
Record LED	RecLED	11	GPIO 17	0
Autonomous LED	StartLED	13	GPIO 21	2
Record Button	RedButton	16	GPIO 23	4
Autonomous Button	StartButton	18	GPIO 24	5

9 REFERENCES

Andrade, O, Jones, E, Lee, A & Bates, D 2012, *Physical Computing with Raspberry Pi - Buttons and Switches*, viewed September 2013, <http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/buttons_and_switches/>.

Avanzini, P, Royer, E, Thuilot, B & Martinet, P 2008, 'Vehicle Platooning using Monocular Vision and a Laser Rangefinder', *2008 10th Intl. Conf. on Control, Automation, Robotics and Vision*, Hanoi.

Avanzini, P, Thuilot, B, Dallej, T, Martinet, P & Derutin, JP 2009, 'On-line Reference Trajectory Generation for Manually Convoying a Platoon of Automatic Urban Vehicles', *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis.

Bergenhem, C, Hedin, E & Skarin, D 2012, 'Vehicle-to-Vehicle Communication for a Platoon System', *Procedia - Social and Behavioural Sciences*, no. 48, pp. 1222 - 1233.

Bergenhem, C, Huang, Q, Benmimoun, A & Robinson, T 2010, 'Challenges of Platooning on Public Motorways'.

Bräunl, T 2012, 'ELEC2303 Embedded Systems Lecture Notes', University of Western Australia.

California PATH, *About California PATH*, viewed 17 May 2013, <http://www.path.berkeley.edu/About/Default.htm>.

Chin, C, Hall, R & Gadgil, N 2003, 'The Automated Highway System/Street Interface: Final Report', University of Southern California.

CSIRO 2008, *Light Detection and Ranging*, viewed 10 September 2013, <http://www.csiro.au/Outcomes/Food-and-Agriculture/LightDetectionLidar.aspx>.

Department of Climate Change and Energy Efficiency 2010, *Australia's Emissions Projections*, viewed 17 May 2013, <http://www.climatechange.gov.au/~/media/publications/projections/australias-emissions-projections-2010.pdf>.

Department of Commerce 2013, *FuelWatch Historical Price Search*, viewed 17 May 17, <http://www.fuelwatch.wa.gov.au/fuelwatch/pages/public/historicalPriceSearch.jspx>.

die.net, *popen(3) - Linux Man Page*, viewed 2013, <http://linux.die.net/man/3/popen>.

Gnuplot 2013, viewed 2013, <http://www.gnuplot.info/>.

Government of South Australia: Department of Planning, Transport and Infrastructure 2012, *Road Safety: Driver Fatigue*, viewed 17 May 2013, <http://www.dpti.sa.gov.au/roadsafety/Safer_behaviours/fatigue__and__distractions>.

Horowitz, R, Tan, C-W & Sun, X 2004, 'An Efficient Lane Change Maneuver for Platoons of Vehicles in an Automated Highway System', University of California Berkeley, California.

IBEO 2009, *Ethernet data protocol ibeo LUX and ibeo LUX Systems*, viewed July 2013, <http://www.springfieldelectric.com/images/cm/20089171340547087617423 31/EthernetOut_ibeoLUXPrototype.pdf>.

ibeo Automotive Systems 2013, *From light pulse to emergency braking*, viewed March 2013, <http://www.ibeo-as.com/index.php/en/ihow-it-works>.

IBEO Automotive Systems 2013, *IBEO Mission*, viewed 10 September 2013, <http://ibeo-as.com/index.php/en/imission>.

IBEO Automotive Systems GmbH 2013, *ibeo LUX*, viewed 10 September 2013, <http://ibeo-as.com/images/stories/pdf/neu/salesblatt%20lux.pdf>.

Investment Mine 2013, *5 Year Crude Oil Prices and Price Charts*, viewed May 2013, <http://www.infomine.com/investment/metal-prices/crude-oil/5-year/>.

Larburu, M, Sanchez, J & Rodriguez, DJ 2010, 'Safe Road Trains for the Environment: Human factors' aspects in dual mode transport systems'.

mathbits.com 2013, *Arrays in C++ - Qucik Sort*, viewed 2013, <http://mathbits.com/MathBits/CompSci/Arrays/Quick.htm>.

Naranjo, JE, Bouraoui, L, García, R, Parent, M & Sotelo, MÁ 2009, 'Interoperable Control Architecture for', *IEEE Transactions on Intelligent Transportation Systems*, vol 10, no. 1, pp. 146 - 154.

National Oceanic & Atmospheris Administration 2013, *Trends in Atmospheric Carbon Dioxide*, viewed May 2013, <http://www.esrl.noaa.gov/gmd/ccgg/trends/global.html>.

National Oceanic and Atmospheric Administration 2013, *Carbon Dioxide at NOAA's Mauna Loa Observatory reaches new milestone: Tops 400 ppm*, viewed 17 May 2013, <http://researchmatters.noaa.gov/news/Pages/CarbonDioxideatMaunaLoareaches400ppm.aspx>.

Parent, M 2004, 'Automated Urban Vehicles: State of the Art and', *2004 8th International Conference on Control, Automation, Robotics and Vision*, Kumming, China.

RAC & CCI 2012, 'Congestion Crisis', Perth.

SARTRE 2012, *The SARTRE Project*, viewed 13 May 2013, <http://www.sartre-project.eu/en/Sidor/default.aspx>.

Shladover, SE 2008, 'AHS Research at the California PATH Program', *Proceedings of the 2008 IEEE International Conference on Vehicular Electronics and Safety*, Columbus.

The IEEE and The Open Group 2004, *popen*, viewed September 2013, <http://pubs.opengroup.org/onlinepubs/009696799/functions/popen.html>.

Tulloch, J 2012, *Future transportation: Remote control convoys*, viewed May 2013, <http://knowledge.allianz.com/mobility/transportation_safety/?1908/future-transportation-remorte-control-convoys>.

Whitely, S 2012, 'Embedded Systems Tutorial 8', University of Western Australia.