REAL-TIME PLATOONING OF MOBILE ROBOTS: STUDY OF TRAJECTORY WITH OBSTACLE AVOIDANCE

DAVID BONG CHUNG HUA

ELECTRICAL AND ELECTRONICS ENGINEERING

UNIVERSITI TEKNOLOGI PETRONAS

JANUARY 2017

Real-time Platooning of Mobile Robots: Study of Trajectory with Obstacle Avoidance

by

David Bong Chung Hua

18427

Dissertation submitted in partial fulfilment of

the requirements for the

Bachelor of Engineering (Hons)

(Electrical and Electronics Engineering)

JANUARY 2017

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Real-time Platooning of Mobile Robots: Study of Trajectory with Obstacle Avoidance

by

DAVID BONG CHUNG HUA

18427

A project dissertation submitted to the

Electrical & Electronics Engineering Programme

Universiti Teknologi PETRONAS

in partial fulfillment of the requirement for the

BACHELOR OF ENGINEERING (Hons)

(ELECTRICAL & ELECTRONICS)

Approved by,

(Mr. Lo Hai Hiung)

UNIVERSITI TEKNOLOGI PETRONAS

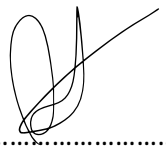
TRONOH, PERAK

January 2017

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgments, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

.



……

(DAVID BONG CHUNG HUA)

ABSTRACT

Robot platooning is a robot positioning method, whereby a few robots will be moving in formation. Platooning has been a very commonly discussed technique especially for its applications on vehicles. However, up until now, it still has many issues that are preventing it from being implemented in real life and one of them is its irregular trajectory in the maintaining phase. In this project, study will be made on the real-time platooning trajectory in its maintaining phase under the most two common factors in real life, which are changes in velocity and presence of obstacle. By studying its performance under the mentioned factors, this project aims to contribute by having algorithm that can improve the platooning trajectory during its maintaining phase.In order to do so, two robotic cars are used as models to simulate real vehicles. The first one would act as the leader while the second one would act as the follower. Both cars would be equipped with a visual sensor, Pixy camera while Fuzzy logic would be used as the main controller logic in this project due to the limited processing power of the microcontroller used, DFR0305, Atmega 328. Data will be collected throughout the experiment so that improved algorithm can then be developed for a better platooning performance.

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Ever since automobiles are invented, vehicles have been driven manually by a driver with the purpose of reaching the intended destination [1]. By having land transportation as the main commuting media nowadays, the increasing traffic and soaring energy costs has never been greater. In fact, about 60% of all surface freight transportation is done on roads [2]. Despite land transportation's significant role in the global trade as well as world economy, crucial challenges such as increasing fuel prices and the need to reduce greenhouse gas emissions have made man realizes the potential benefits of cooperative driving especially for freight transportation [1,2].

Platooning is formation movements of a group of vehicle in coordination and have since been studied for several decades [1,3]. Vehicles platooning can improve the capacity of the road by allowing more vehicles to use a given length of road [1,3,4]. According to [5, Table 1], it also improves energy efficiency, thus reducing fuel consumption by reducing aerodynamic drag [1, 3,5]. The reason is simple because the smaller inter-vehicle distance reduces the aerodynamic drag, thus leads to a higher energy efficiency.

Table 1: Fuel Consumption of Leading Truck and Its Platoon [5]

	Simul. Fuel consump. [%]	Exper. Fuel consump. [%]
Lead Truck	100	100
Time Gap 1	93.2	92.9
Time Gap 3	94.9	-
Time Gap 5	98.8	98.7

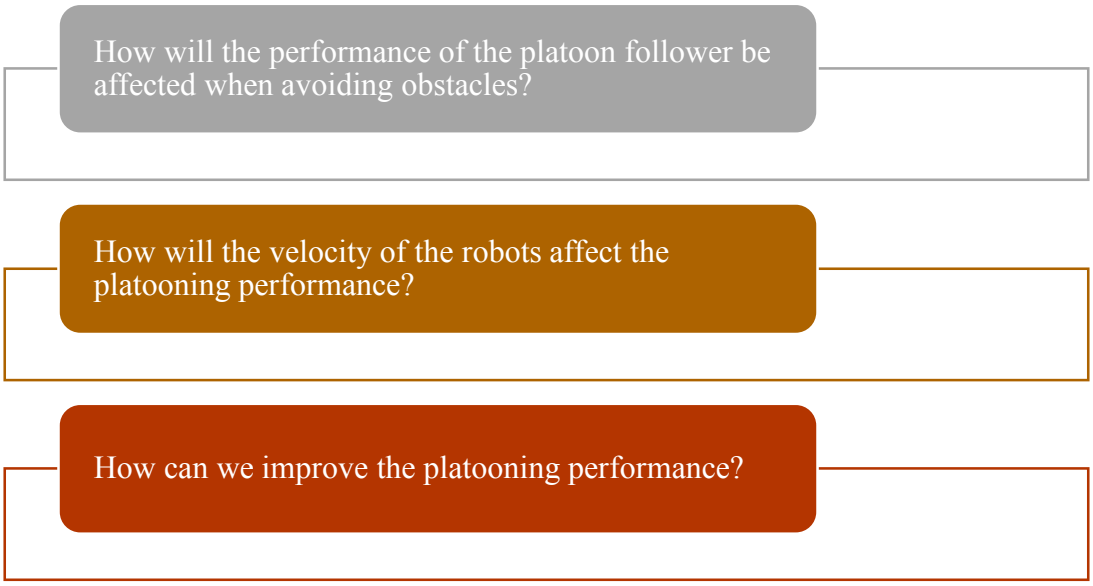
In addition, as smaller inter-vehicle distance tends to increase the risk of accidents, platooning would be crucial. With automated control, a small inter-vehicle distance can still be maintained even when vehicles are moving in high speed as safety is now ensured [2].

Due to the dynamic characteristics of vehicle platooning in natural environment, it is important that obstacle avoidance be taken into consideration [6]. This not only ensures the platooning performance, but also greatly increases its safety by eliminating what is unknown.

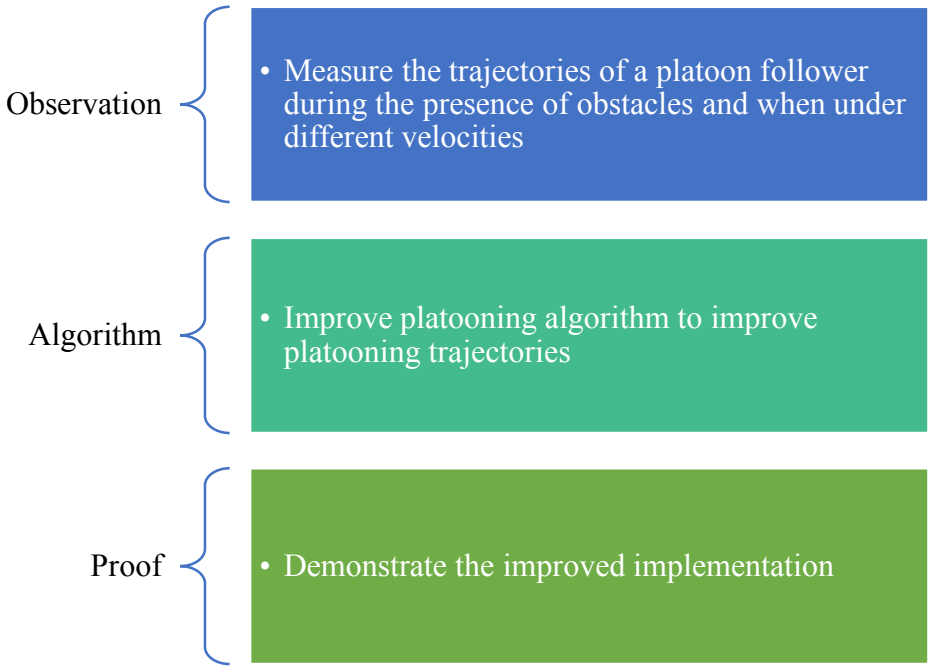
The two main approaches used in vehicles platooning are catch-up strategy or slow-down strategy. In catch-up strategy, the platoon follower speeds up to catch up with its leading platoon whereas in slow-down strategy, platoon leader slows down to allow the following platoon to catch up [7]. In both strategies, manipulation of velocity is essential in order to maintain the platoon formation [3]. Therefore, it is crucial that the effect of platoons' velocity on the performance of platooning be taken into consideration.

Lastly, study shows that platoon formation could be delayed up to 20% even on moderate traffic density [3]. Since traffic is unavoidable in real-life transportation, the performance of platooning when avoiding obstacle as well as changing velocity has to be observed in order to ensure a robust and safe platoon formation.

1.2PROBLEM STATEMENT



1.3 OBJECTIVES and SCOPE of STUDY



Vehicles platooning consists of three main phases as shown in Figure 1 [7]. The focus of this project would be on the second phase, which is the maintaining phase. In phase two, vehicles ought to make sure a certain separation from their preceding vehicle.

In this project, a monocular camera, CMUcam5 Pixy is used as the main sensor of the mobile robots. A total of two mobile robots would be used in order to show the platooning formation. In order to get accurate measurements on the platooning performance as well as to obtain the improvement in platooning trajectories through the improvement in algorithm, multiple experiments will be conducted with different platoon velocity, and presence of obstacles. From the experiment, performance data before and after the platooning algorithm improvement are obtained and tabulated. Moreover, demonstration would also be done in order to prove the obstacle avoidance capability during platooning.

This project is thus relevant, given all the mentioned advantages of platoon formation mentioned in background.

CHAPTER 2

LITERATURE REVIEW and/or THEORY

There will be two focus in this project, which are the ability to follow its leader and the ability to avoid obstacles while following its obstacles. Leader-follower mechanism [8] together with fuzzy logic would be used in order to achieve the main requirement of this project which is maintaining the platoon formation.

As for the second focus of this project, fuzzy logic controlling mechanism would be applied in order to have a robust obstacles avoiding system despite the dynamic characteristics of natural environment [9,10,11].

In spite of the robot's capability to follow its leader as well as avoiding obstacles, dilemma arises when following robot does not know whether turning to the right or left would be a better option so as to maintain the performance of platooning. This problem is illustrated in [9, Figure 1] below.

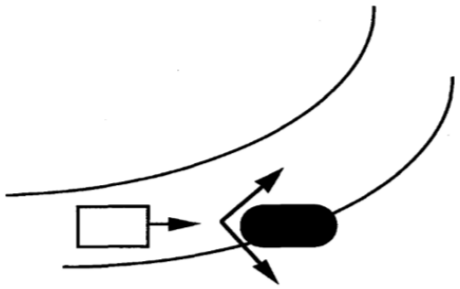


Figure 1:Dilemma when Avoiding Obstacles [9]

In order to maintain a robust performance of platoon formation even when the changing of velocity when avoiding obstacles, Payton and Rossenblatt's approach is used so as to set the priority right [9].

2.1 LEADER-FOLLOWER MECHANISM

This approach is commonly used to enable multiple robots moving in a formation, which is in our case to maintain the platoon formation. In leader-follower approach, only one robot will act as the leader of the platoon at a time. This leader robot is the only platoon which has the information for the reference path while the other robots in the platoon would simply follow the reference trajectory by the leader [8].

However, as there are more than two robots in a platoon, problem arises when there is only one robot acting as the leader. As a result, a simple extension of the leader-follower mechanism, that involves a simple cascaded system is created. In this system, the leader follower mechanism will be in a straight line whereby the first robot would be the leader to the following robot while the following robot will then be the leader to its next robot [8, 12].

2.2 FUZZY LOGIC CONTROLLING MECHANISM

Fuzzy logic is invented by A.Zadeh. It is developed as many concepts in the real word cannot be accurately represented by defined definition such as true or false and 1 or 0 [9]. As a result, fuzzy logic is often used to control autonomous robots in natural environments [10, 12]. The theory behind fuzzy logic enables vague concepts to be defined in term of partial membership function [9]. [9, Figure 2] below illustrates how fuzzy logic is used to define distance.

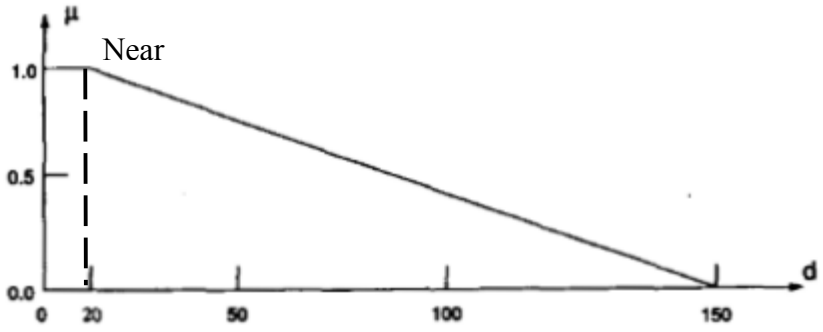


Figure 2: Fuzzy Set Representing Concept of Distance [9]

In Figure 2, d represents the interval distance between obstacle and robot in centimetre (cm) while μ represents the membership of the fuzzy set; Near with, 1 representing a full membership of Near and 0 representing zero membership of Near. As shown in Figure 2, 100cm away from the robot is represented by approximately the membership value of 0.4 while interval distances below 20cm are represented by a full membership value of 1. The ability of capturing vague concepts using fuzzy sets instead of the exact interval distance reduces the processing load for the microcontroller but most importantly, improves the robustness of the obstacle avoidance system by reducing the effect of sensor noise. This is because sensor noise can only affect the Near fuzzy membership degree slightly.

2.3 PAYTON AND ROSSENBLATT'S APPROACH

Payton and Rossenblatt's (P&R) approach is developed by David Payton, Ken Rosenblatt and David Keirsey. They invented this technique as they realize loss of information

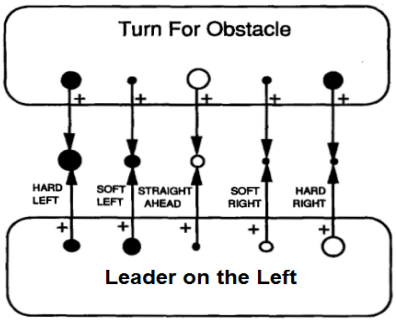


Figure 3: Decisions with Weighted-Priorities [9]

usually happens whenever two or more decisions are made based on fixed-priority [9]. As a result, they have come up with a decision making technique based on weighted priorities as shown in [9, Figure 3].

In P&R method, each options available is arranged in a set of nodes for easier decision making as shown in Figure 3. The white color of the node represents a negative activation while the black color represents a positive activation; with the bigger the node in the figure, the higher the priority of the behavior. For instance, higher desirable positive activation is given to both hard left and hard right, and a large negative activation given to moving straight ahead when meeting an obstacle. The final control chosen when the platoon follower meets an obstacle while having its leader at the left is a hard left. Similar decisions can be made by using the same P & R approach if the leader is on the right, or when the leader used to be on the right or left while it temporarily misses its leader when avoiding obstacle.

CHAPTER 3

METHODOLOGY/ PROJECT WORK

3.1 FLOWCHART

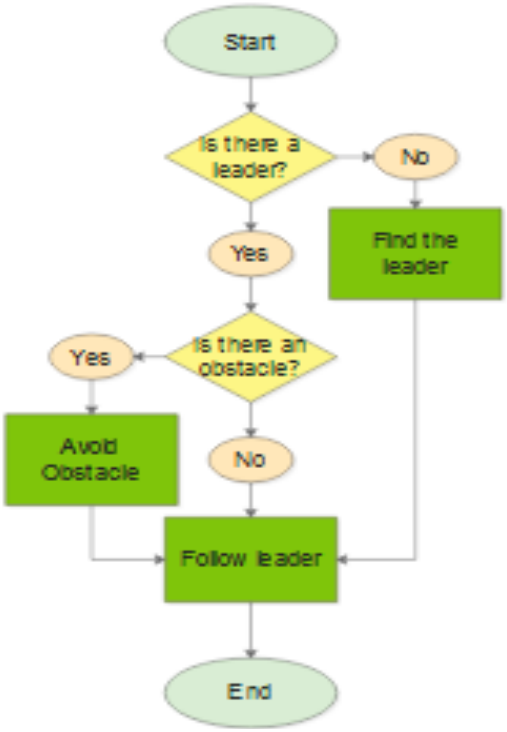


Figure 4: General Flowchart of Project

3.2 Fuzzy Logic

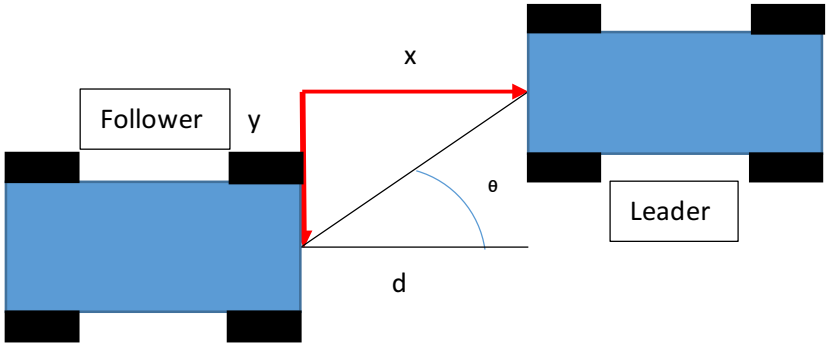


Figure 5: Overview of whole system

$$\text{Deviation angle, } \theta = \tan^{-1}\frac{y}{x}$$

Table 2:Fuzzy Logic Table for Leader-follower System

Angle, θ / Distance, d	Large Negative	Small Negative	Zero	Small Positive	Large Positive
Far	Left fast, Right reverse	Left fast, Right stop	Left Right, Fast	Left Stop, Right fast	Left stop, Right reverse
Close			Left Right, Medium		
Very Close			Left Right, Maintain Slow		

Table 2 shows the rotation of both left and right wheels of robot given the interval distance of the follower robot from the leader robot as well as the deviation angle of the leader platoon from the following platoon. According to the logic deduced from Figure 3 through P & R approach, the rotation of both wheels when obstacle exists can be derived. Depending on the deviation angle, θ, the following platoon should make a hard right if θ is positive but a hard left if θ is negative. If θ is zero, only a soft left or right is needed. Note that obstacle avoidance is only triggered if the distance of the obstacle falls within

close range. After successfully avoiding the obstacle, the fuzzy logic given in Table 2 is applied again until the appearance of the next obstacle.

In contrary, if leader is no way to be located, following robot is set to rotate statically in the same position until the leader is located. The logic in Table 2 is then applied once the leading robot is located. Lastly, the performance of platooning will be recorded by using a video camera and analyzed accordingly.

3.3 HARDWARE AND SOFTWARE

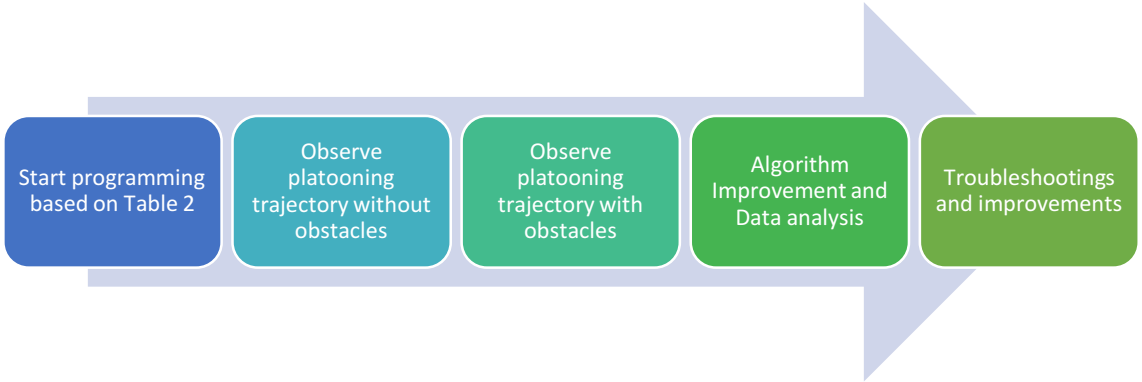
3.3.1 Hardware

- CMU Cam5 Pixy Camera
- DFR0305 Romeo BLE (Arduino Compatible Atmega 328)

3.3.2 Software

- Arduino
- Pixymon
- Matlab

3.4 KEY MILESTONES



3.5 Timeline (Gantt Chart)

Table 3: FYP1 and FYP2 Gantt Chart

Week	FYP1							FYP2						
	2	4	6	8	10	12	14	2	4	6	8	10	12	14
Testing hardware (cameras and motors)														
Studying on fuzzy logic controller and pixy camera														
Studying factors affecting the trajectory performance of platooning														
Determining the right fuzzy sets of logic														
Implementation in Arduino, Testing and Algorithm Troubleshooting														
Introducing obstacles, Testing														

and Algorithm Improvements														
Data Analysis														
Project Reporting														

CHAPTER 4

RESULTS and DISCUSSIONS

4.1 Pixy Camera Calibration

Figure 6: Platoon Leader Indicator

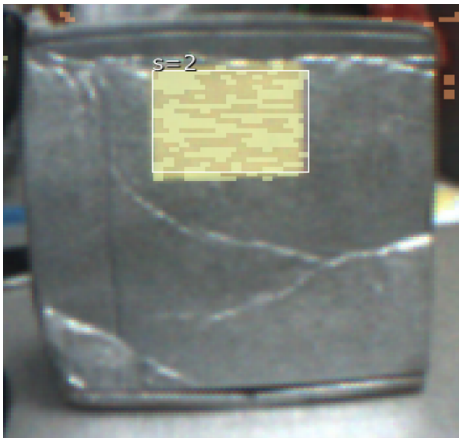


Figure 7: Obstacle

The indicator is detected as Signature 1 while the obstacle is detected as Signature 2 for the follower.

4.2 General Pseudocode

The pseudocode would be divided into two parts. The first part will be detecting leader's indicator, while the second part would be deciding how to react based on the position of the leader's indicator. Both parts would then contribute towards leader following.

4.2.1: Detecting Leader's Indicator

```
if detected blocks,
{calculate the area of all the detected blocks;
only retain the largest one;}
if Maximum Area < 1500 pixels^2,
{ignore;
do nothing;}
else
Indicator detected = true;
```

As PIXY camera detects objects based on only the colour parameter, there would be multiple blocks being detected other than the real indicator of the platoon leader. In order to remove all the noises or invalid blocks, only the block with the biggest size will be taken into consideration.

However, there is an issue when the size of the real indicator gets smaller than the size of the invalid blocks. In order to solve this issue, it is decided that the real indicator would declared as out of sight or invalid as its size gets smaller than 1500pixels^2. This value is obtained through numerous test and trials. As there are no valid blocks within the sight of Pixy camera, the robot is set to do nothing.

4.2.2: Reacting when Indicator is on the Left, Right or in the Middle of Camera

```
if indicator=True
        { if block is in the middle
                full speed ahead;
          else if block is on the left
                move to the left;
          else if block is on the right
                move to the right; }
```

In order to locate the position of the block, pixy.blocks().x function can be used. This function can tell the x position of the block within the camera view.

After numerous test and trials, it is decided that as long as the x values reside within 100 to 220, the indicator is considered as in the center while as the value gets smaller than 100, it means the indicator is now on the left. On the other hand, as the x values get larger than 220, this indicates the object is on the right. These data is tabulated as shown in Table 3 below. Platoon follower is set to move accordingly to the position of the indicator in order to keep following the indicator.

Table 4: Values of x for respective positions of block

Postion of indicator	**Values of pixy.blocks().x**
Centre	100<= x <=220
Left	x<100
Right	x>220

Note that, the above pseudocode is limited to just the ability to follow the detected object. It is the first attempt on the project. Fuzzy Logic as well as obstacle avoidance sets will then be included. Since the project is only focusing on the maintaining phase of platooning, the ability to keep track of the leader is set as the first priority. Full C code will be provided in the Appendix 1.

According to observation, the main disadvantage of using this code is the maximum distance of the indicator from the pixy camera can only be 20cm, limited by the size of 1500 pixels, further than that, it will be treated as an invalid block. This is unacceptable considering 20cm is a very short distance, and instead of doing nothing, the robot should move forward to the indicator.

Other than that, by using the maximum block size method, there are still possibilities that blocks other than the real indicator being detected. This happens as some of the object of the same colour that falls within the sight of the camera might be bigger than the real indicator.

4.3 Observations

4.3.1: Experiment of Trajectory Performance against Velocity of Follower

In order to measure the effect of velocity on the trajectory performance, an experiment, with the below variables, is done. Note that in this experiment, the leader is set to move in a fixed pattern, a circle.

Controlled Variables: Velocity of the Leader, Test Environment, Start-off distance and position of leader from the follower

Manipulated Variables: Velocity of the Follower

Responding Variable: Trajectory Performance of Platooning

Velocity of Leader = 100cm/s, Start-off Distance = 5cm, Start-off position: Centre

Table 5: Trajectory Performance of Platooning at Different Velocities

Velocity of Follower (cm/s)	Observation
50	The follower often misses the leader after attempting to follow it.
70	The follower is able to follow the leader.
200	The follower often knocks and pushes the leader away from its trajectory.

Table 5 shows that the follower can follow the leader properly only if they have the same velocity. When it is half the velocity of the leader, the follower often misses the leader as the leader is now moving too fast for the follower to follow. The leader tends to go out of the viewing angle of the follower. To solve this, a modification has been done on the code, as shown in Appendix 2. With this modification, the follower is now able to predict the position of the leader when the leader first leaves its viewing range.

From Table 5, it is observed that trajectory performance differs as velocity of follower changes. Therefore, it can be concluded that velocity plays a very important role in the trajectory performance of platooning.

4.3.2: Experiment of Trajectory Performance against Initial Interval Distance

Experiment 4.3.2 is conducted to investigate the effect of the starting interval distance between leader and follower against the trajectory performance of platooning. Similar to experiment 4.3.1, the leader moves in a fixed circle. Below shows the variables of this experiment:-

Controlled Variables: Velocity of the Leader and Follower, Test Environment, Start-off position of leader from the follower

Manipulated Variables: Starting Interval Distance

Responding Variable: Trajectory Performance of Platooning

Velocity of Leader and Follower = 100cm/s, Start-off position: Centre

Table 6: Trajectory Performance of Platooning with Varied Initial Position of the Leader

Starting Interval Distance	Observation
2cm	The follower often knocks and pushes the leader away from its trajectory
5cm	The follower is able to follow the leader.
10cm	The follower often misses the leader during platooning.

Results in Table 6 show that when the speed of both the follower and leader maintain the same, initial interval distance between the follower and the leader affects greatly the platooning performance of the follower. From Table 6, it can be seen that, even though it is found out that the follower can only follow the leader properly when they are of the same velocity from experiment 4.3.1, this conclusion does not yield truth when anymore when the starting interval distance changes from 5cm.

As a result, velocity of the follower has to be adjusted accordingly to the interval distance between the follower and the leader in order to maintain a good platooning performance.

4.3.3: Experiment of Trajectory Performance against Leader's Initial Position

Experiment 4.3.3 is conducted to investigate the effect of the leader's initial position against the performance of the platooning. This experiment would be similar to experiment 4.3.1 and 4.3.2 above except in this one, both velocity and initial interval distance would be remained constant. Variables of this experiment are as shown below:-

Controlled Variables: Velocity of the Leader and Follower, Test Environment, Start-off distance

Manipulated Variables: Leader's Initial Position

Responding Variable: Trajectory Performance of Platooning

Velocity of Leader and Follower = 100cm/s, Start-off Distance = 5cm

Table 7: Trajectory Performance at Different Leader's Position

Leader's Position	**Observation**
Left (30°)	The follower experiences difficulty in following the leader.
Center	The follower is able to follow the leader.
Right (30°)	The follower experiences difficulty in following the leader.

From both experiment 4.3.1 and 4.3.2, supposedly the follower platoon would not experience any difficulty in following its leader given it is having the same speed as the leader and having a start-off distance of 5cm. However, the results in Table 7 indicates this is not always the case. When the leader is position on the left or right of the follower, the follower starts to experience difficulty with following the leader. This is simply because if the leader is only slight left or right from the follower, by maintaining the same speed, the follower might overshoot instead of making sure that the leader always maintains at its center.

Thus, it can be concluded from experiment 4.3.3 that leader's position would also affect the platooning performance other than the two factors mentioned in both experiment 4.2.1 and 4.2.2.

From experiment 4.2.1, 4.2.2 and 4.2.3, it can be clearly seen that velocity of the follower needs to be adjusted accordingly not only to the interval distance between the follower and the leader but also to the leader's position from the follower's point of view. Therefore, this clearly indicates that interval distance and leader's position has to be fuzzified as inputs in order to determine the correct velocity needed for the follower in order to maintain a smooth platooning performance.

4.4 Area of Indicator at Different Interval Distances

Table 8: Area of Indicator at different interval distance

Interval Distance (cm)	Range of Area (pixels^2)
40	0-460
30	461-858
20	859-2000
10	2001-5000
4	>5000

The data collected in Table 8 is important so as to improve the accuracy of the pixy camera in determining the interval distance between the follower and leader. Besides that, this range of area determine can be used as fuzzy sets of logic for determining the speed of the follower.

40cm will be considered as "Very Far", 30cm as "Far", 20cm as "Near", 10cm as "Very Near" while 4cm as "Stop" as shown in Table 9 below. Full data together with methodology to collect the data above are shown in Appendix 3 and Appendix 4 respectively. Note that all measurements are taken when the indicator is at the center of the viewing angle of the follower.

Table 9: Fuzzy logic for each interval distance

Fuzzy Sets	Interval Distance (cm)
Very Far	40
Far	30
Near	20
Very Near	10
Stop	4

4.5 Area of Indicator at Various Positions

The outcomes produced by pixy.blocks[i].x fall within the range between 0 to 319 [13]. As mentioned in section 4.2, this function provides Arduino with the position of the indicator within the view of the Pixy camera. From this, theoretically, 5 sects of category, such as Left, Slight Left, Center, Slight Right, and Right, can be determined.

Calculations:

Range of each position sets:

$$\frac{\mathbf{0 + 319}}{\mathbf{5}} = \mathbf{63.8} \cong \mathbf{63}\ (\textit{rounded down as range cannot exceeds}\ 319)$$

Table 10 below shows the range values of pixy.blocks().x for each indicator's position sets.

Table 10: Area of Indicator at different leader's position

Fuzzy Sets	**Position of Indicator (°)**	**Values of** pixy.blocks().x
Left	-60	0 - 63
Slight Left	-30	64-127
Center	0	128-191
Slight Right	30	192-255
Right	60	256-319

The range calculated in Table 10 is important so as to determine the fuzzy sets for indicator position. By having both sets of fuzzy logic determined in both section 4.4 and section 4.5, an optimum velocity can now be determined for each condition.

4.6 Optimum Velocity of Follower

By using fuzzy sets determined in section 4.4 and 4.5, optimum velocity for each logic set are plotted as shown in Figure 8 and Figure 9 below.

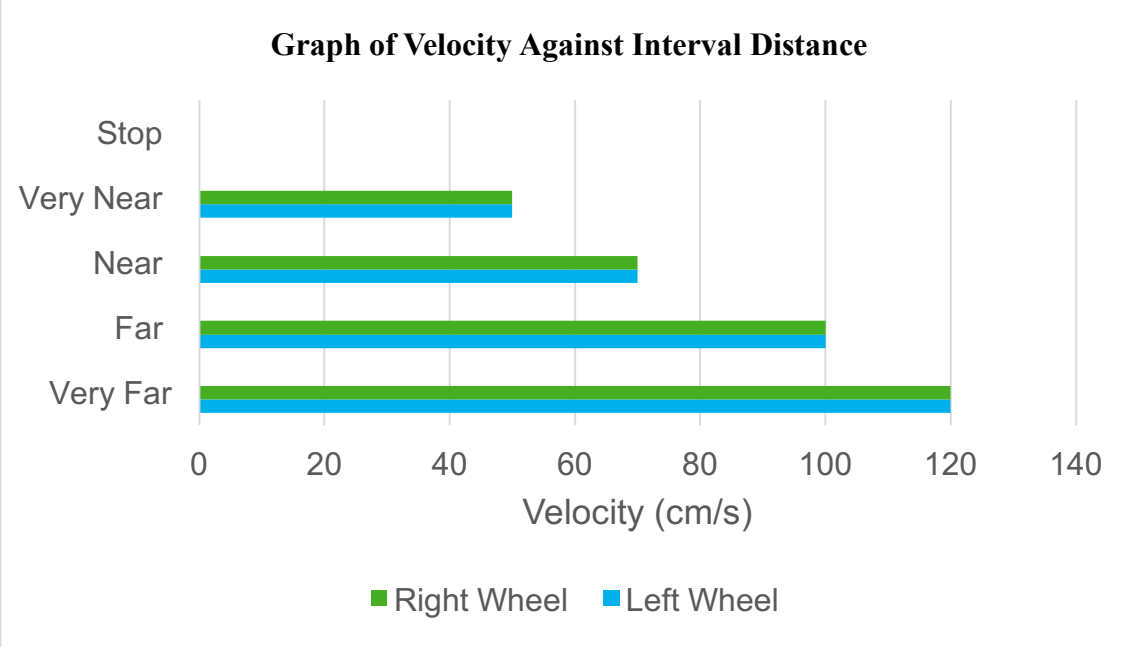


Figure 8: Optimum velocity of each set of interval distances

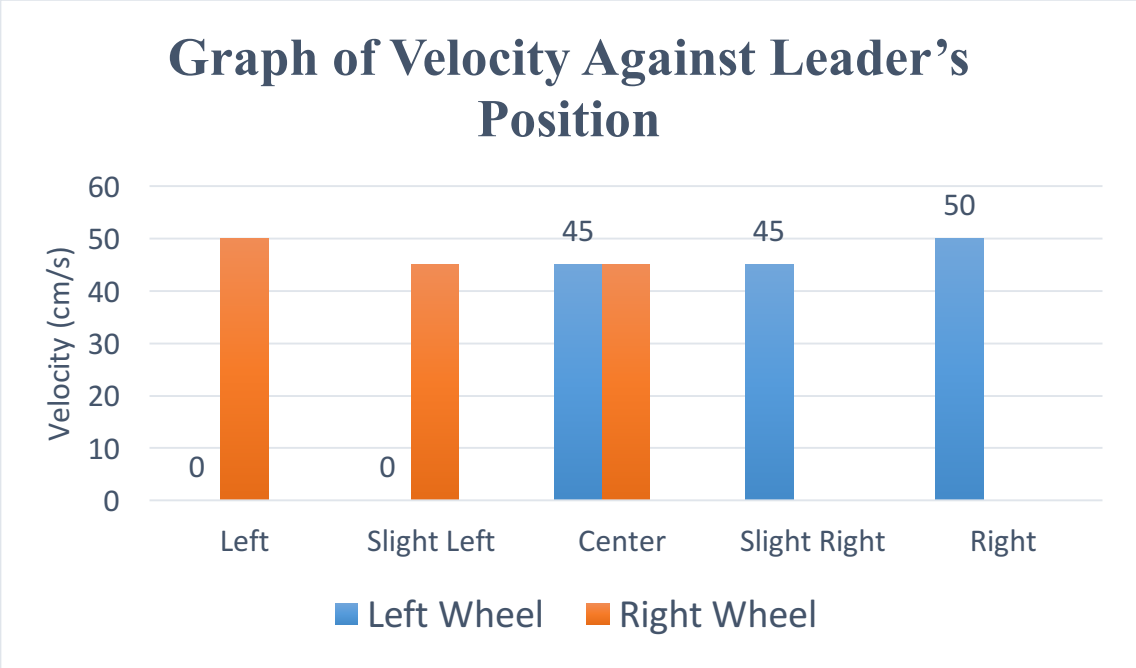


Figure 9:Optimum velocity for each category of leader's positions

Optimum velocities in Figure 8 and Figure 9 are obtained after numerous test and trials. Note that the follower will first adjust itself based on the velocity given in Figure 9 until the leader is at its center before proceeding towards moving according to the velocities given in Figure 8. Both of these fuzzy sets of logic make up the fundamental of the algorithm used in this project for platooning.

From Figure 9, when indicator falls within the slight left and left category, left wheel will be stopped, causing the follower to move to the left. However, as shown, the velocity of the right wheel in slight left category is lower than the one in left category. The reason is so that the car could adjust slowly if the indicator is just slight left while faster if the indicator is further left. Same concept applies to the right.

The full Arduino program is shown in Appendix 5.

4.7 Detectable Range of Pixy Camera

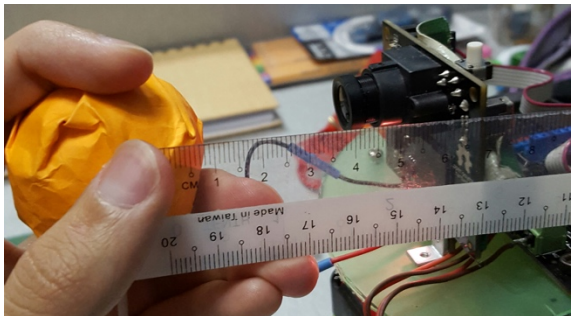


Figure 10: Minimum interval distance

From experiment as shown in Figure 10, it is found that the minimum distance required for the initial starting position is 4cm. Less than that, it is found that the follower starts to behave oddly and even begin to move towards the leader, even though it should have stopped immediately when it reaches the "Stop" category. The value of the block size returned by the pixy camera is as shown in the Figure 11 below:-

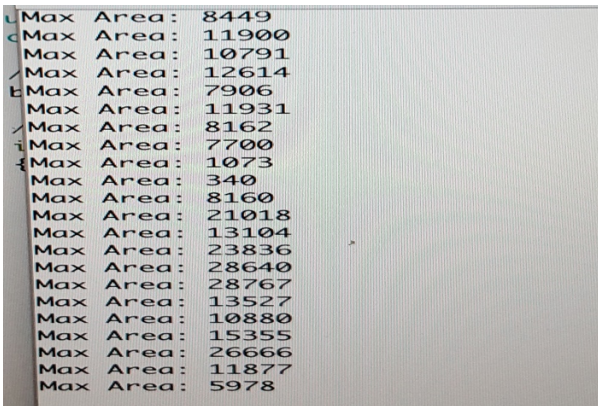


Figure 11: Area of block size when marker is at distance less than 4cm

As shown in Figure 11, the block size returned is very inconsistent as some of them dropped until 1073 pixels^2 or 340 pixels^2. As only area larger than 5000 pixels^2 falls within the "Stop" category, the car continues to move forward even though it is already less than 4cm from the leader in front.

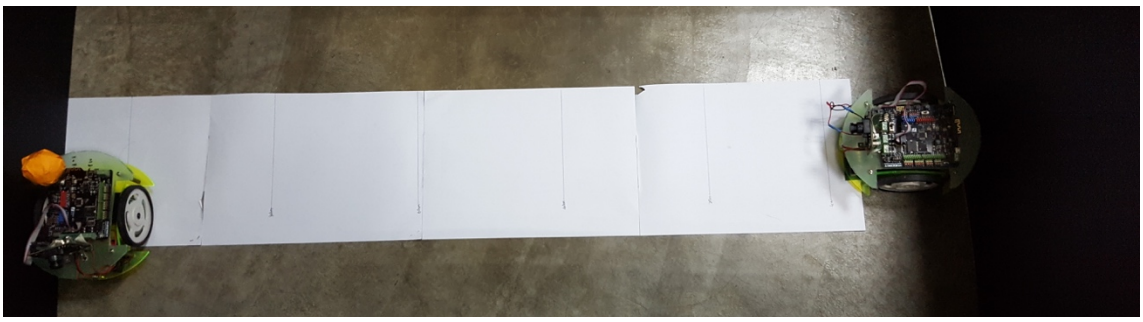


Figure 12: Maximum interval distance

Figure 12 shows the maximum interval distance recorded in this experiment is limited by the length of the controlled environment, which is 120 cm. At 120cm, the follower performs normally without any problem.

According to datasheet of Pixy camera, as long as the object falls within its view, the object will be detected. For an image of the size 4 pixels^2,Pixy camera can detect it as long as the object is not further than 304.8cm away. Therefore, the maximum distance detectable by Pixy camera depends on the size of the image as well.

Table 11 below shows the maximum and minimum distance Pixy camera can detect an object.

Table 11: Pixy Camera's Detectable Range

	Maximum	**Minimum**
Detectable Range	Greater than 120cm	4cm

4.8 Platooning Trajectories

After obtaining the right velocity for each fuzzy condition, platooning algorithm used in section 4.3 is then improved and implemented. Below shows platooning trajectories under different sets of movements:-

4.8.1 Straight Line

The straight-line trajectories are recorded in different distances such as 20cm, 40cm and 60cm as shown below.

- 20cm

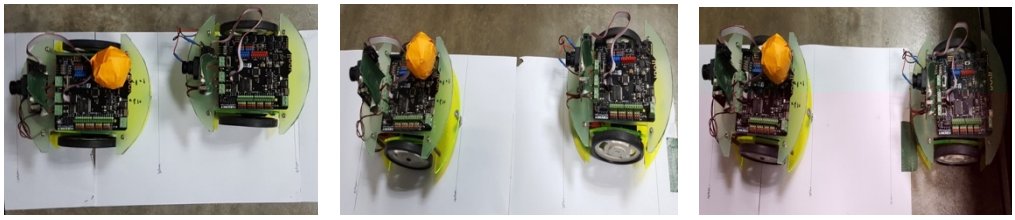


Figure 13: Trajectory of platooning at 20cm

- 40cm

Figure 14: Trajectory of platooning at 40cm

- 60cm

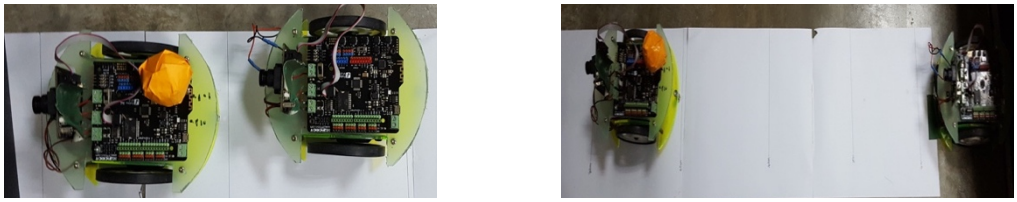


Figure 15: Trajectory of platooning at 60cm

From Figure 13, 14 and 15, it is observed that the follower's performance maintains as long as the leader falls within the detectable range as mention in section 4.7.

4.8.2 Square

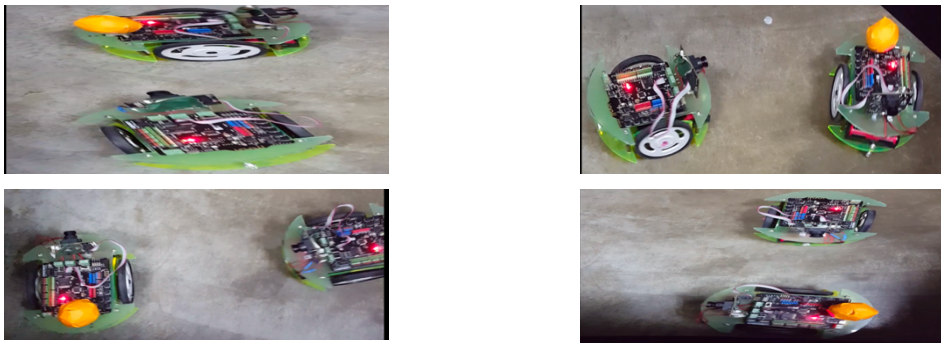


Figure 16: Platooning trajectory when moving in square

Figure 16 shows that follower is able to follow the leader properly even at the four right-angled corners of a square. As the leader is making a turn, the follower slows down and waits until the leader is moving further away from it.

4.8.3 Circle

Figure 17: Platooning trajectory when moving in circle

Figure 17 shows that as the leader is moving in a circle with small radius, the follower tends to stay in the middle, while adjusting its direction accordingly to follow the leader.

Results obtained in Figure 13 to Figure 17 shows that the performance of platooning improves after the implementation of the developed fuzzy-logic algorithm. Moreover, the consistency of the performance is also observed throughout different platooning patterns.

4.8.4 Straight Line with Obstacle

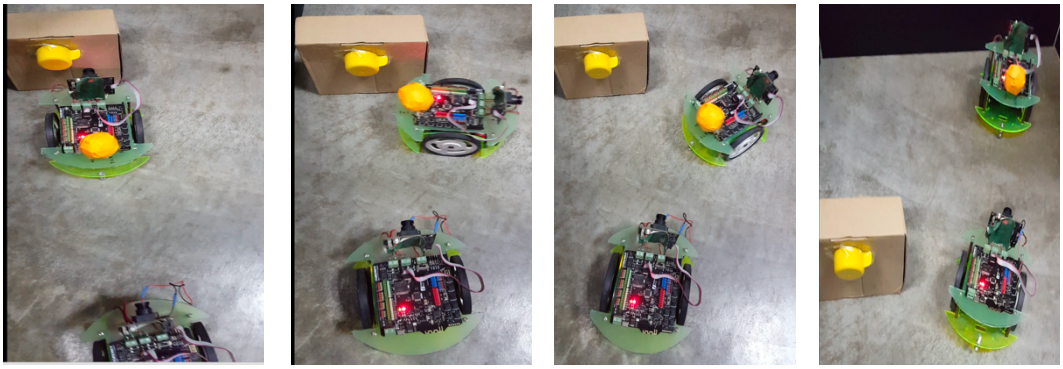


Figure 18: Platooning trajectory under obstacle

Figure 18 shows the movement of the follower when a static obstacle is introduced into the track. As the leader avoid the obstacle, the follower follows.

Under obstacles, it is observed that the platooning performance maintains the same, provided that the leader will still be in sight after the follower avoids the included obstacle.

CHAPTER 5
CONCLUSION and RECOMMENDATIONS

5.1 Conclusion

In conclusion, cooperative driving, platooning is useful and very relevant in current society. It is said to be the future of driving especially for freight transportation as well as to ensure a safe extensive driving on a highway. Other than land transportation, platooning is also advantageous in many fields like military, security, entertainment and so on.

Platooning performance is found to be greatly affected by changes in velocities. Therefore, tests have been conducted for numerous times in order to find the optimum velocity based on the leader's position and distance from the follower. The results are then implemented by using fuzzy-logic concept. After implementation, platooning performance increases tremendously as tested with different movement patterns of the leader.

Under obstacle, it is observed that the platooning performance maintains the same, provided that the leader is still detectable after the follower avoids the obstacle. In short, the platooning performance has been improved through developed fuzzy logic algorithm and has proven its consistency through different platooning patterns shown in section 4.8.

5.2 Recommendations

This project can be further enhanced by including platoon formation phase and separation phase other than the maintaining phase in order to make this an overall more complete platooning system.

In addition, sensors that can function more independently of lighting condition such as light detection and ranging (LIDAR) or infrared sensors can be added to complement the main sensor used, CMU CAM5 Pixy. This is to ensure a more robust performance even under poor lighting condition.

Besides that, this project can be further expanded with the inclusion of inter-robot communication into the platooning system instead of a full dependence on a single sensor, Pixy camera. Without inter-robot communication propagation of error could not be avoided if any of the follower fails to follow its leader since a cascaded leader -follower system is applied.

Lastly, for better obstacle avoidance, feedback system can be incorporated into the algorithm. By doing so, the follower or leader can avoid obstacles of all sizes no matter if it is placed at the most left, right or in the middle of the obstacles. This is because the interval distance between it and the edge of the obstacle is now taken into consideration.

REFERENCES

[1] J. Axelsson, "Safety in Vehicle Platooning: A Systematic Literature Review," *IEEE Transactions on Intelligent Transportation Systems*.

[2] A. Alam, B. Besselink, V. Turri, J. Martensson, and K. H. Johansson, "Heavy-duty vehicle platooning for sustainable freight transportation: a cooperative method to enhance safety and efficiency," *IEEE Control Systems,* vol. 35, pp. 34-56, 2015.

[3] K.-Y. Liang, Q. Deng, J. Mårtensson, X. Ma, and K. H. Johansson, "The influence of traffic on heavy-duty vehicle platoon formation," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 150-155.

[4] J. Ploeg, N. Van De Wouw, and H. Nijmeijer, "Lp string stability of cascaded systems: Application to vehicle platooning," *IEEE Transactions on Control Systems Technology,* vol. 22, pp. 786-793, 2014.

[5] A. Al Alam, A. Gattami, and K. H. Johansson, "An experimental study on the fuel reduction potential of heavy duty vehicle platooning," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, 2010, pp. 306-311.

[6] T. C. Ng, J. Ibañez-Guzmán, J. Shen, Z. Gong, H. Wang, and C. Cheng, "Vehicle following with obstacle avoidance capabilities in natural environments," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, 2004, pp. 4283-4288.

[7] M. Saeednia and M. Menendez, "A Consensus-Based Algorithm for Truck Platooning," *IEEE Transactions on Intelligent Transportation Systems*.

[8] A. Loria, J. Dasdemir, and N. A. Jarquin, "Leader–Follower Formation and Tracking Control of Mobile Robots Along Straight Paths," *IEEE Transactions on Control Systems Technology,* vol. 24, pp. 727-732, 2016.

[9] J. Yen and N. Pfluger, "A fuzzy logic based extension to Payton and Rosenblatt's command fusion method for mobile robot navigation," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 25, pp. 971-978, 1995.

[10] C.-J. Kim and D. Chwa, "Obstacle avoidance method for wheeled mobile robots using interval type-2 fuzzy neural network," *IEEE Transactions on Fuzzy Systems,* vol. 23, pp. 677-687, 2015.

[11] M. Benzaoui, H. Chekireb, M. Tadjine, and A. Boulkroune, "Trajectory tracking with obstacle avoidance of redundant manipulator based on fuzzy inference systems," *Neurocomputing,* vol. 196, pp. 23-30, 2016.

[12] S. M. Cristescu, C. M. Ionescu, B. Wyns, R. De Keyser, and I. Nascu, "Leader-follower string formation using cascade control for mobile robots," in *Control & Automation (MED), 2012 20th Mediterranean Conference on*, 2012, pp. 1092-1098.

[13] J. P. Lang. (2011): CMUcam5 Pixy [Online]. Available: http://cmucam.org/projects/cmucam5/wiki/Hooking_up_Pixy_to_a_Microcontroller_like_an_Arduino

Appendix

Appendix 1

```
//
#include <SPI.h>
#include <Pixy.h>

//Fuzzy
#include <FuzzyRule.h>
#include <FuzzyComposition.h>
#include <Fuzzy.h>
#include <FuzzyRuleConsequent.h>
#include <FuzzyOutput.h>
#include <FuzzyInput.h>
#include <FuzzyIO.h>
#include <FuzzySet.h>
#include <FuzzyRuleAntecedent.h>

#define dirRight  4  //Motor 2= Right Motor
#define pwmRight  5
#define dirLeft 7   //Motor 1= Left Motor
#define pwmLeft 6

Pixy pixy; //Declare a pixy global

void setup() {
 // pinMode(pwmLeft,OUTPUT);
  pinMode(pwmRight,OUTPUT);
  pinMode(dirLeft,OUTPUT);
```

```
  pinMode(dirRight,OUTPUT);

  pixy.init(); //initialize pixy global
  Serial.begin(9600);
}

  uint16_t blocks;
  int j, area, maxArea, maxJ;
  char buf[32];

void loop()
{
  maxArea=0;
  blocks= pixy.getBlocks();

   if (blocks) // if there were any recognized objects
   {
            //finds the largest object that fits the
signature
            for (j = 0; j < blocks; j++)
            {
                  area = pixy.blocks[j].width *
pixy.blocks[j].height;
                  if (area > maxArea) //save the new
largest obj
                  {
                        maxArea= area;
                        maxJ = j;
                  }
             }
```

```
            Serial.print ("Max area is ");
            Serial.println (maxArea);

            if (maxArea < 1500)
            {
             digitalWrite(dirRight, LOW);
             analogWrite(pwmRight, 0);
             digitalWrite(dirLeft, LOW);
             analogWrite(pwmLeft, 0);
            }

            else
            {
                  sprintf(buf, "  block %d: ", maxJ);
                  Serial.print(buf);
                  pixy.blocks[maxJ].print();

                  //Making sure object is in the middle of
camera.
                  if (pixy.blocks[maxJ].x >= 100 &&
pixy.blocks[maxJ].x <= 220)
                  {
                        digitalWrite(dirRight, HIGH);
                        analogWrite(pwmRight, 100);
                        digitalWrite(dirLeft, HIGH);
                        analogWrite(pwmLeft,100);
                  }
                  else
                  {
                        if (pixy.blocks[maxJ].x < 100)
                        {
```

```
                                    digitalWrite(dirLeft,LOW);
                                    analogWrite(pwmLeft,0);
                                    digitalWrite(dirRight,HIGH);
                                    analogWrite(pwmRight,100);
                              }

                              if (pixy.blocks[maxJ].x > 220)
                              {
                                    digitalWrite(dirRight,LOW);
                                    analogWrite(pwmRight,0);
                                    digitalWrite(dirLeft,HIGH);
                                    analogWrite(pwmLeft,100);
                               }
                        }
////////////////////////////////////////
               }

         }
   delay (200);
}
```

Appendix 2

```
//Pixy
#include <SPI.h>
#include <Pixy.h>

//Fuzzy
#include <FuzzyRule.h>
#include <FuzzyComposition.h>
#include <Fuzzy.h>
#include <FuzzyRuleConsequent.h>
#include <FuzzyOutput.h>
#include <FuzzyInput.h>
#include <FuzzyIO.h>
#include <FuzzySet.h>
#include <FuzzyRuleAntecedent.h>

#define dirRight  4  //Motor 2= Right Motor
#define pwmRight  5
#define dirLeft 7   //Motor 1= Left Motor
#define pwmLeft 6

Pixy pixy; //Declare a pixy global
int maxJ;
char dirCheck = ' ';

void setup()
{
  pinMode(pwmLeft, OUTPUT);
  pinMode(pwmRight, OUTPUT);
  pinMode(dirLeft, OUTPUT);
```

```
  pinMode(dirRight, OUTPUT);
  pixy.init(); //initialize pixy global
  Serial.begin(9600);
}

/*
   Car Movements
*/
void front() {
  //Serial.println("Go Front");
    digitalWrite(dirRight, HIGH);
    analogWrite(pwmRight, 75);
    digitalWrite(dirLeft, HIGH);
    analogWrite(pwmLeft, 75);
}
void back() {
  //Serial.println("Go Round Back");
    digitalWrite(dirRight, HIGH);
    analogWrite(pwmRight, 50);
    digitalWrite(dirLeft, LOW);
    analogWrite(pwmLeft, 50);
}
void right() {
  //Serial.println("Go Right");
    digitalWrite(dirRight, HIGH);
    analogWrite(pwmRight, 50);
    digitalWrite(dirLeft, HIGH);
    analogWrite(pwmLeft, 75);
}
void left() {
```

```
  //Serial.println("Go Left");
    digitalWrite(dirRight, HIGH);
    analogWrite(pwmRight, 75);
    digitalWrite(dirLeft, HIGH);
    analogWrite(pwmLeft, 50);
}

void no_move() {
  //Serial.println("No Move");
    digitalWrite(dirRight, LOW);
    analogWrite(pwmRight, 0);
    digitalWrite(dirLeft, LOW);
    analogWrite(pwmLeft, 0);
}

/*
   FInding blocks with maximum size
*/
int find_maximum_size(int blocks)
{
  int j, maxArea = 0, area;
  char buf[32];
  maxJ = 0;
  for (j = 0; j < blocks; j++)
  {
    area = pixy.blocks[j].width * pixy.blocks[j].height;
    if (area > maxArea) //save the new largest obj
    {
      maxArea = area;
      maxJ = j;
```

```
    }
  }
  return maxArea;
}

/*
  Control movement of the car
*/
void control_movement(int blockSize) {
  int x = 0;
  x = pixy.blocks[maxJ].x;
  //pixy.blocks[maxJ].print();
  //  if (blockSize > 100)
  //  {
  //    no_move();
  //    delay (1000);
  //  }
  //  else if (blockSize <= 100)
  //  {

  switch (x)
  {
    case 70 ... 275:
      front();
      break;

    case 0 ... 69:
      dirCheck = 'l';
      left();
      break;
```

```
    case 276 ... 319:
      dirCheck = 'r';
      right();
      break;
  }
  delay(50);
}

/*
    Main Program
*/
void loop()
{
  static int i = 0;
  static int k = 0;
  int blockSize;
  uint16_t blocks;
  char buf[32];
  // grab blocks!
  blocks = pixy.getBlocks();

  if (blocks > 0)
  {
    k = 0;
    i++;

    // do this (print) every 50 frames because printing
every
    // frame would bog down the Arduino
    if (i % 20 == 0)
```

```
    {
      //Serial.println("Blocks detected.");
      i = 0;  //to control the value limit of i
      blockSize = find_maximum_size(blocks);
      control_movement(blockSize);
    }
  }
  else
  {
    k++;
    if (k % 500 == 0) //this is important so that it does
not stop intermittently. Making sure the block is 0 because
there is really no object, not because of hardware scanning
issue.
    {
      switch (dirCheck)
      {
        case 'l':
          left();
          break;
        case 'r':
          right();
          break;
        default:
          back();
      }
      i = 0;
    }
  }
}
```

Appendix 3

```
Max area is 380
Max area is 460
Max area is 414
Max area is 420
Max area is 420
Max area is 418
Max area is 400
Max area is 399
Max area is 306
Max area is 418
Max area is 400
Max area is 360
Max area is 323
Max area is 396
Max area is 420
Max area is 420
Max area is 323
Max area is 340
Max area is 361
Max area is 342
Max area is 300
Max area is 400
Max area is 420
Max area is 420
```

Figure 19: Data collected at 40cm

```
Max area is 696
Max area is 750
Max area is 825
Max area is 725
Max area is 720
Max area is 780
Max area is 750
Max area is 780
Max area is 630
Max area is 780
Max area is 660
Max area is 858
Max area is 832
Max area is 630
Max area is 750
Max area is 638
Max area is 806
Max area is 864
Max area is 744
Max area is 660
Max area is 780
Max area is 744
Max area is 744
Max area is 750
```

Figure 20: Data collected at 30cm

```
Max area is 1932
Max area is 1848
Max area is 1848
Max area is 2021
Max area is 1890
Max area is 1892
Max area is 1935
Max area is 1716
Max area is 1845
Max area is 1892
Max area is 1892
Max area is 2021
Max area is 1848
Max area is 1892
Max area is 1716
Max area is 2091
Max area is 1755
Max area is 1665
Max area is 1610
Max area is 2091
Max area is 1512
Max area is 1656
Max area is 1720
Max area is 1764
```

Figure 21: Data collected at 20cm

```
Max area is 8004
Max area is 8091
Max area is 7830
Max area is 8091
Max area is 8004
Max area is 8360
Max area is 7998
Max area is 8004
Max area is 8004
Max area is 7826
Max area is 7912
Max area is 7912
Max area is 7820
Max area is 7905
Max area is 7912
Max area is 7998
Max area is 8342
Max area is 7650
Max area is 8178
Max area is 7568
Max area is 7644
Max area is 7917
Max area is 7912
```

Figure 22: Data collected at 10cm

```
Max area is 24420
Max area is 29785
Max area is 31396
Max area is 28490
Max area is 22680
Max area is 30420
Max area is 21901
Max area is 26901
Max area is 27600
Max area is 28704
Max area is 28800
Max area is 18437
Max area is 21594
Max area is 26390
Max area is 30251
Max area is 30589
Max area is 22680
Max area is 22479
Max area is 26901
Max area is 25810
Max area is 22932
Max area is 24660
Max area is 24920
Max area is 21411
```

Figure 23: Data collected at 4cm

Appendix 4

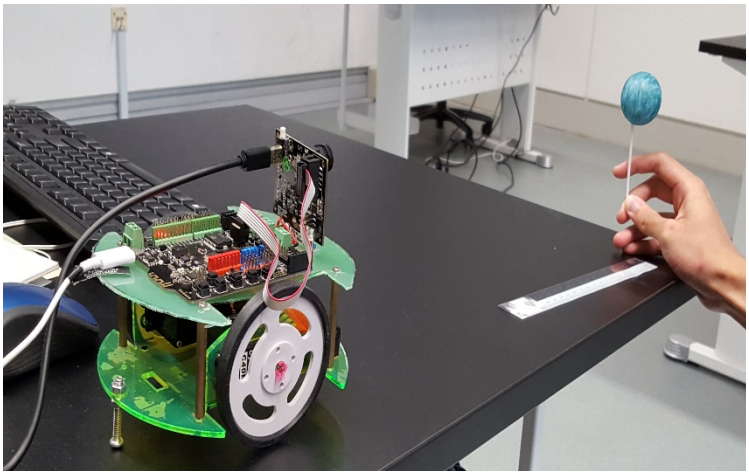


Figure 24: Data collection method at 40cm

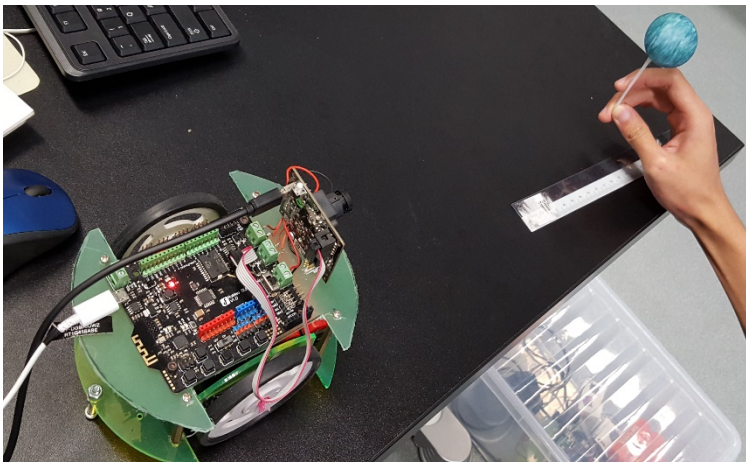


Figure 25: Data collection method at 30cm

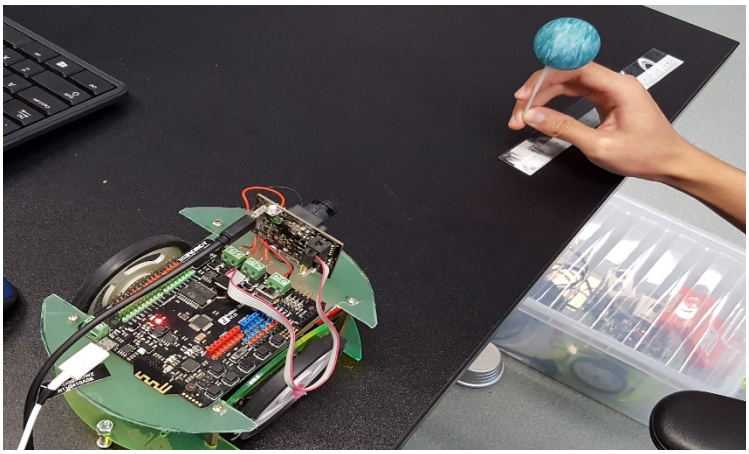


Figure 26: Data collection method at 20cm

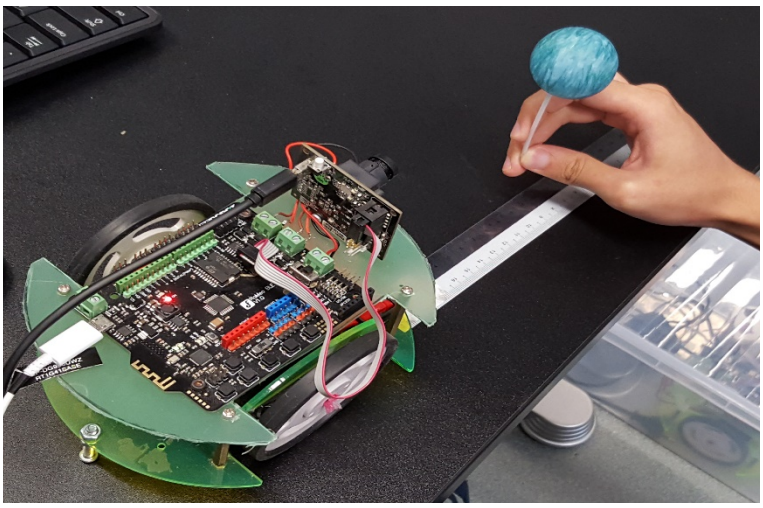


Figure 27: Data collection at 10cm

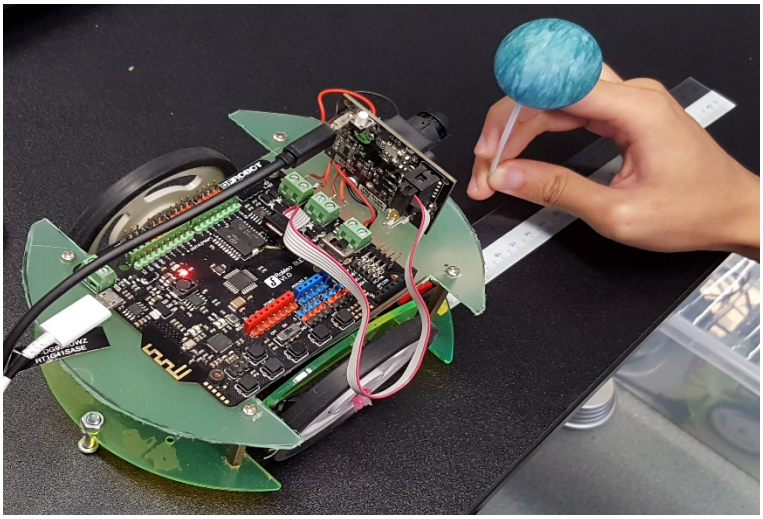


Figure 28: Data collection at 4cm

Appendix 5

```
//Pixy
#include <SPI.h>
#include <Pixy.h>
//Fuzzy
#include <FuzzyRule.h>
#include <FuzzyComposition.h>
#include <Fuzzy.h>
#include <FuzzyRuleConsequent.h>
#include <FuzzyOutput.h>
#include <FuzzyInput.h>
#include <FuzzyIO.h>
#include <FuzzySet.h>
#include <FuzzyRuleAntecedent.h>

#define dirRight  4  //Motor 2= Right Motor
#define pwmRight  5
#define dirLeft 7   //Motor 1= Left Motor
#define pwmLeft 6

Pixy pixy; //Declare a pixy global

int maxJ;
char dirCheck = ' ';
int fspeed = 0;
int lspeed = 0;
int rspeed = 0;
void setup()
{
  pinMode(pwmLeft, OUTPUT);
```

```
  pinMode(pwmRight, OUTPUT);
  pinMode(dirLeft, OUTPUT);
  pinMode(dirRight, OUTPUT);

  pixy.init(); //initialize pixy global
  Serial.begin(9600);
}
/*
   Car Movements
*/
void front() {

  //Serial.print("Fspeed: ");
  //Serial.println(fspeed);

  //Serial.println("Go Front");
  digitalWrite(dirRight, HIGH);
  analogWrite(pwmRight, fspeed);
  digitalWrite(dirLeft, HIGH);
  analogWrite(pwmLeft, fspeed);
}
void back() {
  //Serial.println("Go Round Back");
  digitalWrite(dirRight, HIGH);
  analogWrite(pwmRight, 0);
  digitalWrite(dirLeft, LOW);
  analogWrite(pwmLeft, 43);
}
void right() {
  //Serial.println("Go Right");
```

```
  digitalWrite(dirRight, LOW);
  analogWrite(pwmRight, rspeed);
  digitalWrite(dirLeft, HIGH);
  analogWrite(pwmLeft, lspeed);
}
void left() {
  //Serial.println("Go Left");
  digitalWrite(dirRight, HIGH);
  analogWrite(pwmRight, rspeed);
  digitalWrite(dirLeft, HIGH);
  analogWrite(pwmLeft, lspeed);
}
/*
   FInding blocks with maximum size
*/
int find_maximum_size(int blocks)
{
  int j, maxArea = 0, area;
  char buf[32];

  maxJ = 0;

  for (j = 0; j < blocks; j++)
  {
    area = pixy.blocks[j].width * pixy.blocks[j].height;
    if (area > maxArea) //save the new largest obj
    {
      maxArea = area;
      maxJ = j;
    }
```

```
  }
  return maxArea;
}


/*
   Checking Interval Distance VF, F, N, VN, S
{40cm,30cm,20cm,10cm,4cm}
   and change front speed accordingly
*/
void interval_distance_checking (int blockSize)
{
  Serial.print ("Block Size: ");
  Serial.println (blockSize);
  if (blockSize >= 0 && blockSize <= 460) //VF
  {
    fspeed = 120;
  }
  else if (460 < blockSize && blockSize <= 858) //F
  {
    fspeed = 100;
  }
  else if (858 < blockSize && blockSize <= 2000) //N
  {
    fspeed = 70;
  }
  else if (2000 < blockSize && blockSize <= 5000) //VN
  {
    fspeed = 50;
  }
  else if (blockSize > 5000) //S
  {
```

```
    fspeed = 0;
  }
}
/*
  Control movement of the car
*/
void control_movement(int blockSize) {
  int x = 0;
  x = pixy.blocks[maxJ].x;

  interval_distance_checking (blockSize);
  switch (x)
  {
    case 256 ... 319: //Right
      //Serial.println("Right");
      dirCheck = 'r';
      lspeed = 50;
      rspeed = 0;
      right();
      break;

    case 192 ... 255: //SlightRight
     // Serial.println("Slight Right");
      dirCheck = 'r';
      lspeed = 45;
      rspeed = 0;
      right();
      break;
    case 128 ... 191: //Center
      //Serial.println("Center");
```

```
      dirCheck = 'c';
      front();
      break;

    case 64 ... 127: //SlightLeft
      //Serial.println("Slight Left");
      dirCheck = 'l';
      rspeed = 45;
      lspeed = 0;
      left();
      break;
    case 0 ... 63: //Left
      //Serial.println("Left");
      dirCheck = 'l';
      rspeed = 50;
      lspeed = 0;
      left();
      break;
  }
  delay(50);
}
/*
    Main Program
*/
void loop()
{
  static int i = 0;
  static int k = 0;
  int blockSize;
  uint16_t blocks;
```

```
  char buf[32];
  // grab blocks!
  blocks = pixy.getBlocks();
  if (blocks > 0)
  {
    k = 0;
    i++;
    // do this (print) every 50 frames because printing
every
    // frame would bog down the Arduino
    if (i % 20 == 0)
    {
      //Serial.println("Blocks detected.");
      i = 0;  //to control the value limit of i
      blockSize = find_maximum_size(blocks);
      control_movement(blockSize);
    }
  }
  else
  {
    k++;
    if (k % 500 == 0) //this is important so that it does
not stop intermittently. Making sure the block is 0 because
there is really no object, not because of hardware scanning
issue.
    {
      Serial.println(dirCheck);
      switch (dirCheck)
      {
        case 'l':
          left();
          break;
```

```
        case 'r':
          right();
          break;
        default:
          back();
      }
      i = 0;
    }

  }
}
```