PBL Task 1 - Fuzzy Car Control

Feature: Collision Avoidance

SIT215 Computational Intelligence

Group 6

Matt Merrett
Will McGowan
Aiden Lyon
Benedict Nicholls
Sio Tou Lai
Brigette Hendersonhall

Table of Contents

Introduction

Adoption of intelligent vehicle control systems has been increasing as advances in technology including hardware, communications and computational and artificial intelligence have made such systems economically and practically viable.

Many heavy industries such as the resources sector and transport industries are increasingly looking to full automation of haulage systems and vehicles to improve safety and efficiency (Mining-technology.com, 2021). Features such as collision avoidance, cruise control and lane departure that provide decision guidance or driver augmentation have been more readily adopted in both personal and commercial vehicles over full autonomy.

The real world application of intelligent vehicle control system features such as collision avoidance is highly complex and deals with significant physical and mechanical variables ranging from physical dimension of the vehicle, to environmental factors like road slope, weather factors and dynamic interaction with other vehicles and objects.

The IEEE (Institute of Electrical and Electronics Engineers) has a dedicated Cybernetics Technical Committee for Intelligent Vehicular Systems & Control. Many other scientific and professional journals also have specific editions and issues related to intelligent vehicle control systems.

Traditionally, a number of car controller subsystems are based on control theory techniques such as various types of Proportional Integral Derivative (PID) controllers, various other error feedback loop controllers, and other more proprietary systems.

The more naive feedback approaches have issues with oscillation, and the more advanced systems using PID controllers need in-depth tuning, while still having issues with changing conditions such as weight and road slopes. They are also harder to reason about.

Fuzzy logic is a way to model logical reasoning using degrees of truth as opposed to the binary contrasting values offered by boolean logic. For example, the amount of pressure applied to a vehicle's acceleration or braking pedal for a given scenario. Using boolean logic, a statement is either only absolutely false or absolutely true, 0 or 1. Applying rigid boolean values to a car's pedal system would equate to an erratically chaotic driving experience as vehicles will be alternating between all gas and all brakes.

Using Fuzzy Logic, the fidelity of pedal operation can be enhanced to the point of mimicking how a person would operate the pedals. Our goal is to design a solution that can allow vehicles to automatically maintain their targeted speed and be able to slow down if possible collisions are going to happen.

The Collision Avoidance Problem

By applying problem-based learning methodology (PBL), subsystems of an autonomous car controller have been researched and explored to design a car controller using fuzzy logic. The main feature we are focussing on is a braking system for collision avoidance.

Collision avoidance features can include a number of different components of the vehicle subsystem and its environment, where the environment includes static and dynamic objects, and objects that can change from static to dynamic or correspondingly dynamic to static. One of the key considerations with autonomous vehicles and more advanced intelligent control systems is not only object detection but object classification. Desired responses vary if the object is classified as a person (on foot), a cyclist or another similar vehicle for example.

While detecting or 'sensing' the environment is certainly a major component of the collision avoidance problem, there are many additional factors that can then be identified that impact the system and how the vehicle should respond.

Object Detection

At a high level in order to avoid colliding with an object a vehicle requires mechanisms to detect;

- Objects around the vehicle
- Object distance relative to the vehicle
- Relative speed (if any) of objects around the vehicle
- Trajectory or path of objects relative to the vehicle
- The vehicle's own position, speed and trajectory relative to its environment and the objects contained in that environment.

Collision avoidance systems vary in their complexity and breadth of features. Some systems for example rely solely on the detection of the distance between the vehicle and another object(s) using a distance sensor. That distance sensor may have a restricted field of view. Depending on the nature of the response to be taken by the vehicle then the sensor regime required will vary.

Typically, collision avoidance involves modulating the vehicle acceleration or deceleration (braking) without also involving any change in the vehicle direction of movement. If there was a requirement to incorporate a change in vehicle trajectory in response to the detected environment then the sensor and detection requirements would be greater as would the response logic.

Environmental Factors

Environmental factors affect the vehicle's ability to avoid a collision. These can affect the performance of sensors used in detecting objects or the vehicle's response in adjusting its own behaviour. Examples of these include;

- Road slope or gradient
- Road conditions (wet, dry, icy, snowy)
- Light conditions
- Dust
- Precipitation (ranging from mist or fog to heavy rain)

Depending on the nature of the sensor(s) heavy rain or particulate matter in the air may affect the ‘visibility’, distance or relative speed perception of the objects in the environment.

Vehicle Conditions

Various aspects of the vehicle itself will also impact the implementation of the logic in modifying various vehicle behaviour to avoid a collision.

- Tyre Condition
- Brake Conditions
- Engine Power
- Vehicle Size
- Vehicle Weight
- General vehicle condition

Not every vehicle, even of the same age, make and model will perform or respond in exactly the same way. Certain generalisations can be made in principle for example that vehicles with good tyre condition will slow down more efficiently than a vehicle with poor tyre condition ie; balding.

Collision Avoidance - Auto-Braking

To minimise the complexity and reduce the scope of our solution, we aim to limit the inputs we consider in our fuzzy logic controller. Ultimately we have a simple and foundational level system that demonstrates the potential of fuzzy logic in this scenario.

The system is based around having a number of variables influencing the brake pressure required to avoid a collision with an object. Traditional boolean logic is ineffective at providing accurate results for complicated scenarios that have uncertain or contradictory conditions. Using fuzzy logic we can apply a research proven rule based system with membership functions to produce a calculated and intelligent braking system that responds to an object.

A number of inputs that could affect the braking distance and efficiency were discussed and evaluated. These included vehicle weight, road slope/gradient and tyre condition, relative speed, object distance and road conditions. Relative speed, object distance and road conditions were assessed as the primary factors that would impact the ability to affect brake pressure application to avoid hitting an object.

Therefore, in this simplified scenario, we consider three input variables which affect the brake pressure applied for the car to avoid hitting an object. Depending on the value of these variables, the brake pressure, the output of our solution, will generally change.

Inputs

- relative speed of the vehicle (The vehicle's current speed relative to the detected object)
- distance of the object (a ‘sensor’ that can detect an object in front 100m)
- condition of the road (wet, dry, and anywhere in between).

Output

- brake pressure to be applied to the wheels

This then allows us to formulate rules on what output should be returned based on given input conditions/states. Some examples of these are as follows;

- If the vehicle speed is very fast and an object is close, then the brakes should most likely be pressed as hard as possible, doing an emergency brake.
- If the object is close or at relatively safe distances but has a very slow speed, then it is safe to break lightly or no brakes at all depending on the vehicle controller, with brake pressure increasing the closer it gets to an object.
- When road wetness is introduced, there is an effect on the time it takes for a vehicle to stop, which will require heavier braking pressures

Solution Design

Solution Overview

The solution is based on standard Fuzzy Logic architecture using the Mamdani Fuzzy Inference system, has three crisp inputs; approaching vehicle speed, object distance, and road condition (wet or dry), and a single output, brake pressure.

Inputs

- relative speed of the vehicle
- detected object distance
- road conditions (wet or dry)

Output

- brake pressure

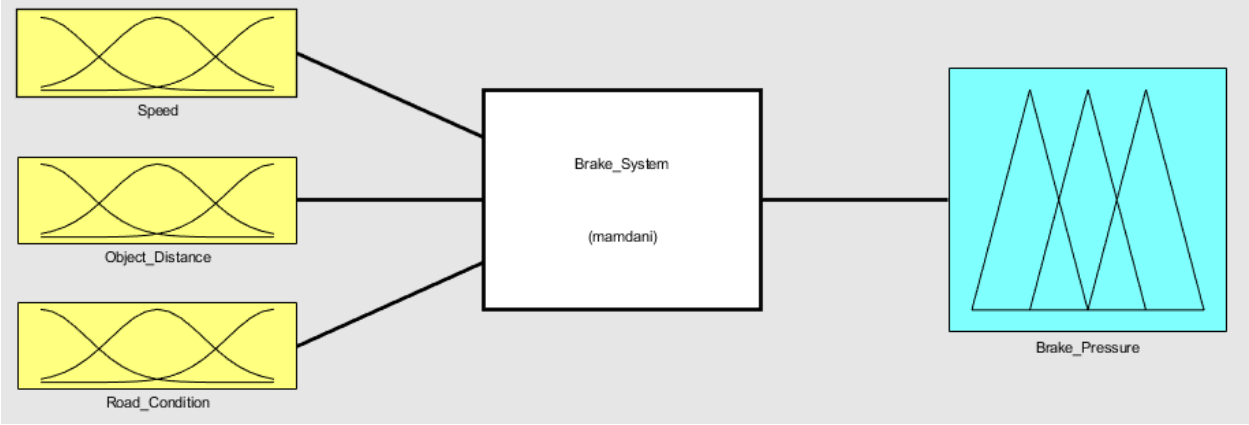


Figure 1. Overall Solution Design - Mamdani Fuzzy Inference Model, MATLAB

MATLAB's Fuzzy Logic Toolbox Add-On (Mathworks, n.d.) was chosen to create the overall model.

A python model using the Scikit-Fuzzy library (scikit-fuzzy development team, n.d.) was later developed using a simpler set of inputs, but that was more accurate to the braking statistics. The Python Model assumes dry conditions and does not include Road Conditions as a model input where the MATLAB model does.

The expert knowledge rules for the system were based on data sourced from the Queensland government's stopping distances statistics. (The State of Queensland, 2016)

The Fuzzy Model

Fuzzy logic works by projecting normal real inputs into a multi-value logic space that allows for values to be more than just true or false.

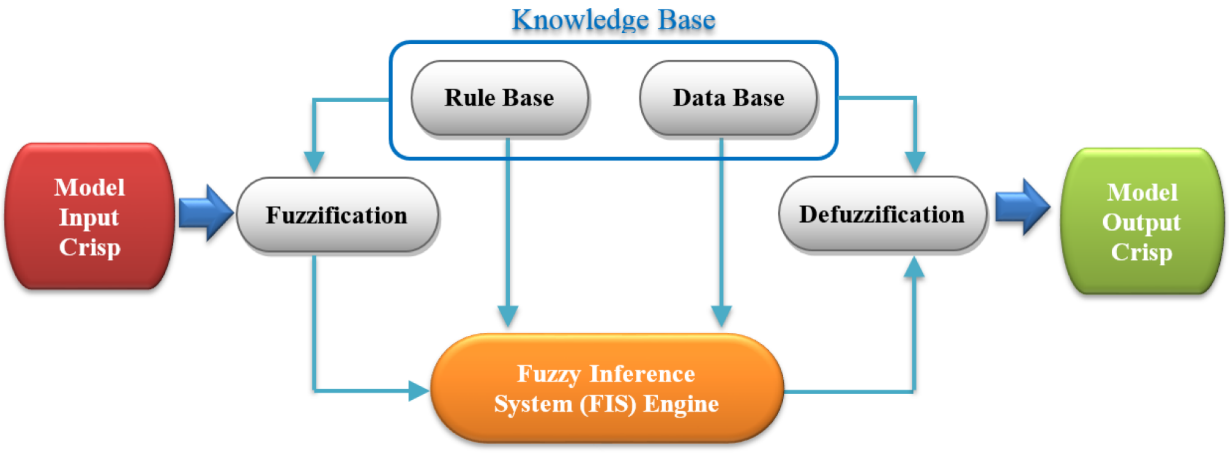


Figure 2. Fuzzy System Diagram (Üneş et al., 2020, Fig 7 page 9)

Chosen Model

We have chosen the Mamdani model for our system (Izquierdo & Izquierdo, 2018). It is easier to understand and develop for than Sugeno models, though not as performant. There are a few other differences such as the way the output is handled, but they are less important. The standard t-norms and t-conorms for the Mamdani model are the Łukasiewicz ones, which are the min and max functions respectively. They are easily readable and easy to reason about.
We are also using the centroids for our defuzzification for the same reason.

Stopping and Braking Distance Dataset

To make our model more accurate and reliable, real world and published data sets showing the relation between speed and braking distance should be used to calibrate and tune the rule base.

Our model is based on a data set of stopping distances shown in Table 1. from the Queensland Government (The State of Queensland, 2016). This data set is a good foundation for our fuzzy rule base. It gives statistical insight as to how long a standard vehicle takes to stop given its speed. For simplicity we will only be considering the speed and braking distance, and ignoring reaction distance.

Speed and braking distance have been incorporated into the membership functions.

		Dry Road		Wet Road	
Speed (km/h)	**Reaction Distance (m)**	**Braking Distance (m)**	**Total Stopping Distance (m)**	**Braking Distance (m)**	**Total Stopping Distance (m)**
40	17	9	26	13	30
50	21	14	35	20	41
60	25	20	45	29	54
70	29	27	56	40	69
80	33	36	69	52	85
90	38	45	83	65	103
100	42	56	98	80	122
110	46	67	113	97	143

***Table 1.** Distance needed to stop on wet and dry roads in an average family car.*
Adapted from Stopping distances on wet and dry roads (The State of Queensland, 2016)

Input Membership Functions

Each of our inputs have a number of membership functions of different types. The most common type of function is the triangular function but there are also linear z and s shaped functions among others.

Object Distance

We have multiple functions for degrees of closeness to make sure that as the object gets closer, it is more weighted towards braking harder, or at least can react faster to changes closer in.

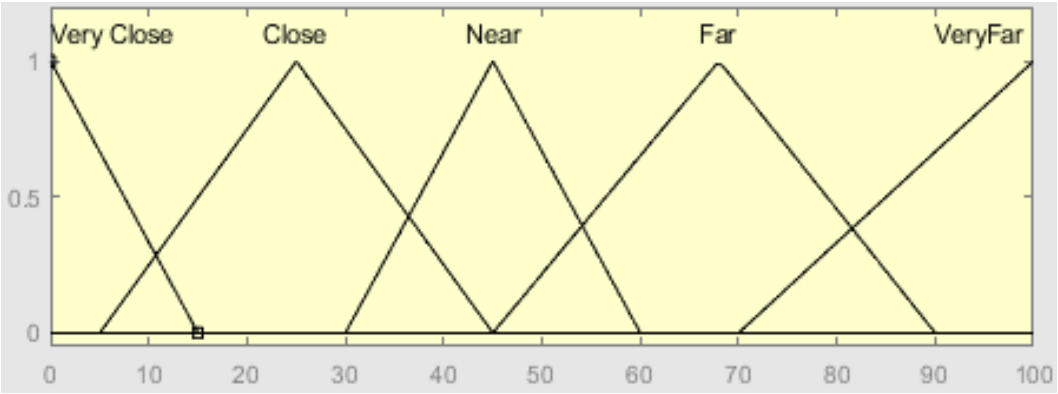


Figure 3. Object Distance Membership Functions used in our Model

Speed

For the speed, we need to know the current speed of the vehicle, we use very fast, fast, medium, slow and very slow for our model.

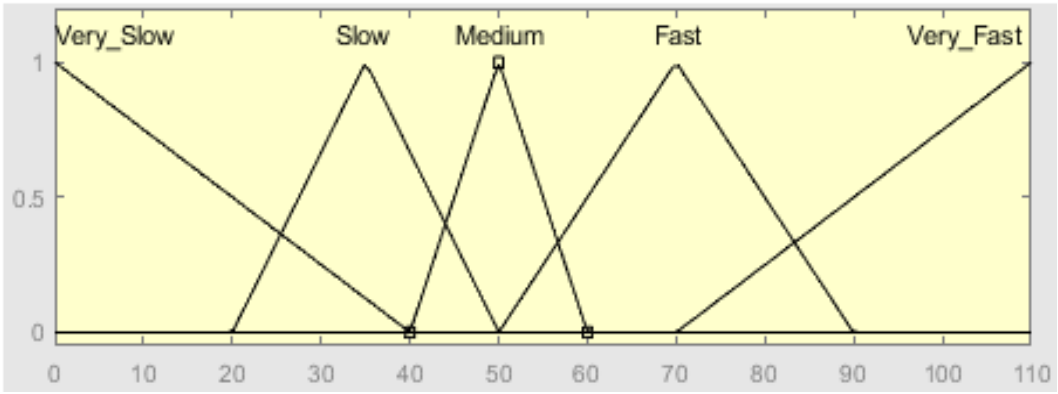


Figure 4. Speed Membership Functions used in our Model

Road Condition

The road condition can vary between dry and wet. Just as in real life, the conditions can vary between various levels of both dryness and wetness. To represent this we are using the z-shaped and s-shaped membership functions. Road Condition was not used as an input in the Python model.

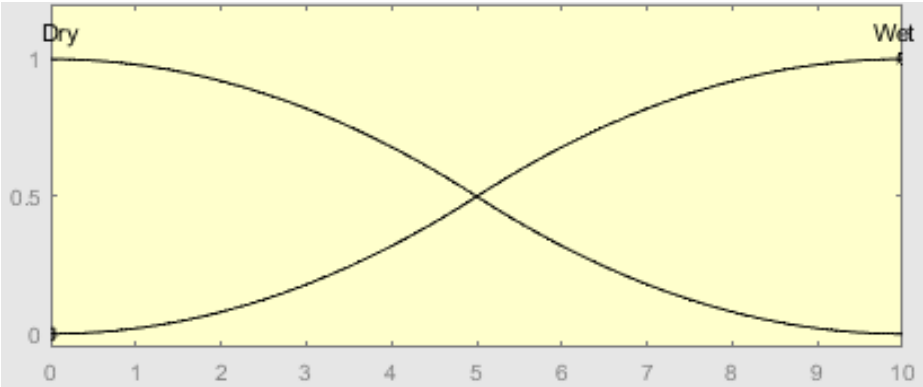


Figure 5. Road Condition Membership Functions used in our Model

Output Membership Function

Brake Pressure

The brake pressure we decided to differentiate between very light, light, medium, heavy and very heavy for the output set. It was decided that “None” would be taken out of the membership function set since the centroid method used for defuzzifying means it will never result in the pressure being zero. Instead, at least for the python simulation, when there is no detected object in range the brake pressure output will be ignored.

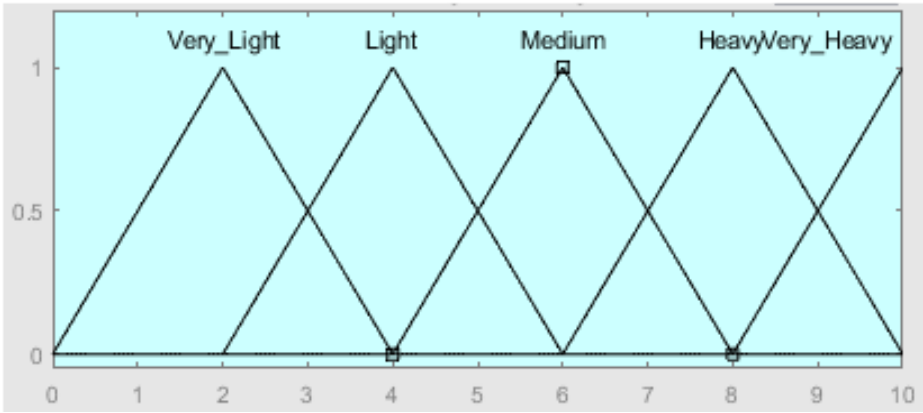


Figure 6. Road Condition Membership Functions used in our Model

Rules

The rules implemented were based on the Queensland Government's braking distance statistics shown in Table 1. Note that the Python Model assumes dry conditions and does not include Road Conditions as a model input or in the rule set where the MATLAB model does.

The MATLAB model contains 32 rules that are constructed using the functionality in the fuzzy toolbox based on the input and output membership functions.

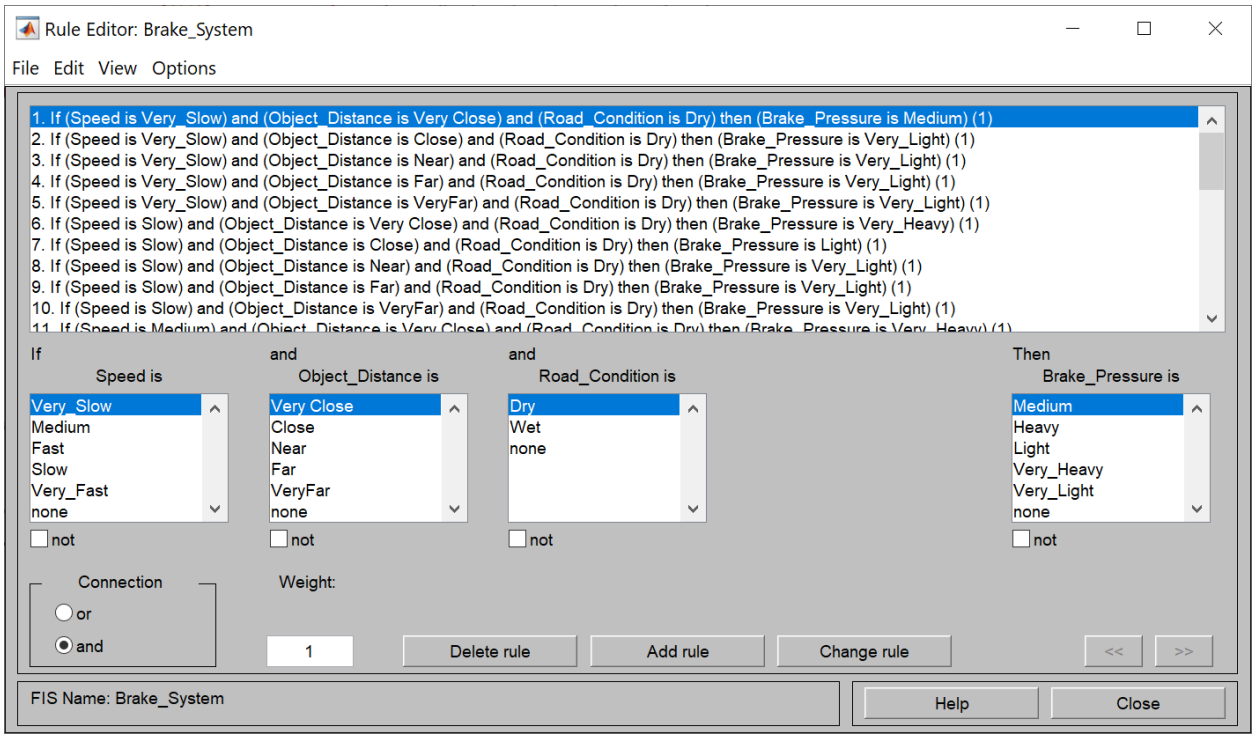


Figure 7. Rule Construction in the MATLAB model

The python model similarly constructs rules although not including road conditions to result in a set of 25 rules.

Rules take the form of logic statements built up of t-norms and t-conorms combining linguistic terms together to produce a resultant fuzzy set.

Example:

$$if\ \ Speed\ =\ Slow\ \ and\ \ Distance\ =\ Close\ \ then\ \ Brake\_Pressure\ =\ Medium$$

In order to satisfy the dataset, the rules had to result in a brake pressure that would be at the maximum braking pressure while the distance was equal to or greater than the braking distance specified for each of the speeds under dry and wet road conditions in the table.

Input Fuzzification

The first step that is taken to evaluate a fuzzy logic model is to take the normal “crisp” input values and turn them into fuzzified input variables, a list of degrees of membership to fuzzy sets. This is achieved by passing the input values to each membership function of the relevant input.

For example, if our speed is 30

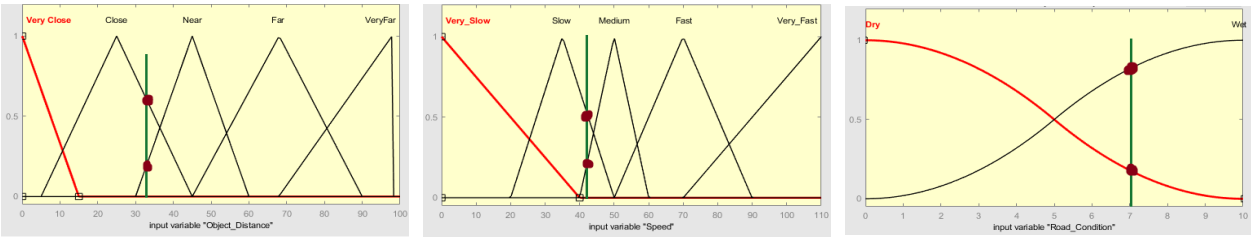
then it is passed to the “Very_Slow” membership function, the “Slow” membership function, the “Intermediate” membership function, etc.

Now our fuzzified speed input is:

$$[Very\_Slow(30),\ Slow(30),\ Intermediate(30),\ \ldots]$$

These lists are then passed to the inference engine.

In order to achieve the desired output in accordance with the Stopping Distance Dataset, some tuning of the membership functions was required.



Close: 0.6, Near: 0.2 Slow: 0.53, Medium: 0.2 Dry: 0.18, Wet: 0.82

Figure 8. Input Fuzzification, If Speed is 42, and Object Distance is 33, and the Road Condition is 7

Rule Evaluation - Inference

The rules are then applied with the fuzzy values as inputs, resulting in a number of truth values for each rule.

An example of such a rule is:

$$if\ \ Speed\ =\ Slow\ \ and\ \ Distance\ =\ Close\ \ then\ \ Brake\_Pressure\ =\ Medium$$

We use a few logical functions within these rules such as t-norms (min) and t-conorms (max), which are the abstraction of conjugation (and) and disjunction (or) respectively but in multi-value logic spaces.

Following this, the above rule could be rewritten as:
$Brake\_Pressure : Medium = min(Slow(speed), Close(distance))$

Taking the example from Figure 8 above we can see the following rules have been activated. Even though all the rules are running all the time, in this case we can see the ones below need to be considered in generating the output.

Rule 7: (Speed is Slow) and (Distance is Close) and (Condition is Dry)
then (Pressure is Light)
Rule 8: (Speed is Slow) and (Distance is Near) and (Condition is Dry)
then (Pressure is Very_Light)
Rule 12. (Speed is Medium) and (Distance is Close) and (Condition is Dry)
then (Pressure is Very_hard)
Rule 13. (Speed is Medium) and (Distance is Near) and (Condition is Dry)
then (Pressure is Light)
Rule 32. (Speed is Medium) and (Distance is Near) and (Condition is Wet)
then (Pressure is Medium)

Implication of the Rule Outputs

We now need to implicate all of those truth values. For Mamdani models, this is done using the min() function.

The implication “reshapes” the membership functions of the output consequent based on the truth values as shown in the below figures, resulting in a number of truncated fuzzy sets.

As described by MatLab (*Fuzzy Inference Process - MATLAB & Simulink*, n.d.)

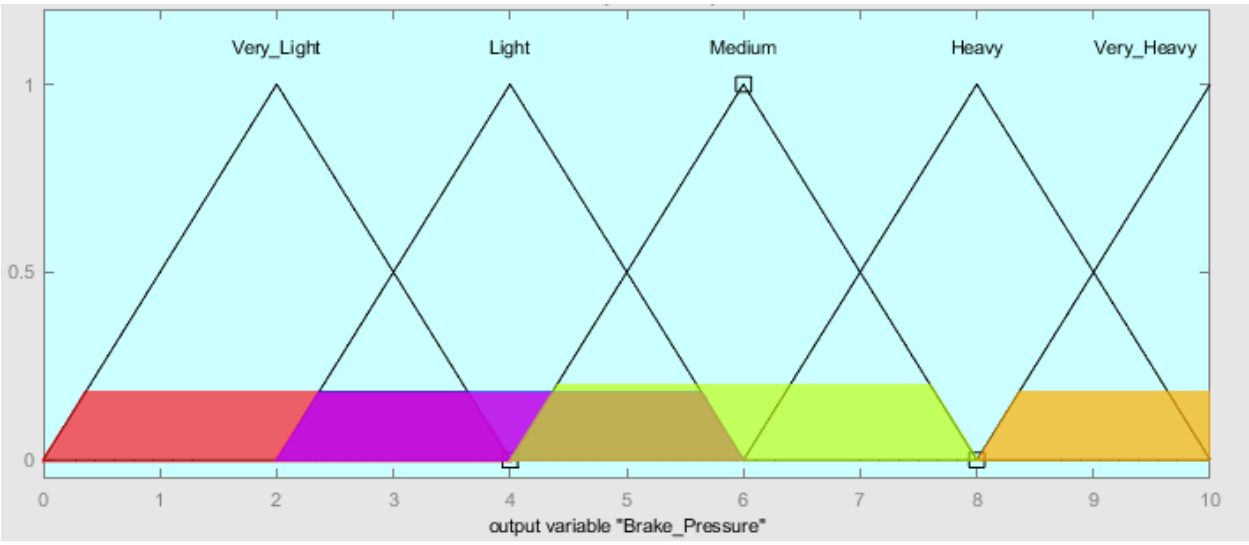


Figure 9. Output consequent based on truth values from inference

Fuzzy Set Aggregation

Now that all the rules have been evaluated and implicated, there will be a number of fuzzy sets, some of them for the same output sets. They all need to be combined.

For Mamdani, this uses the max() function. The result is one combined fuzzy set for the output variable.

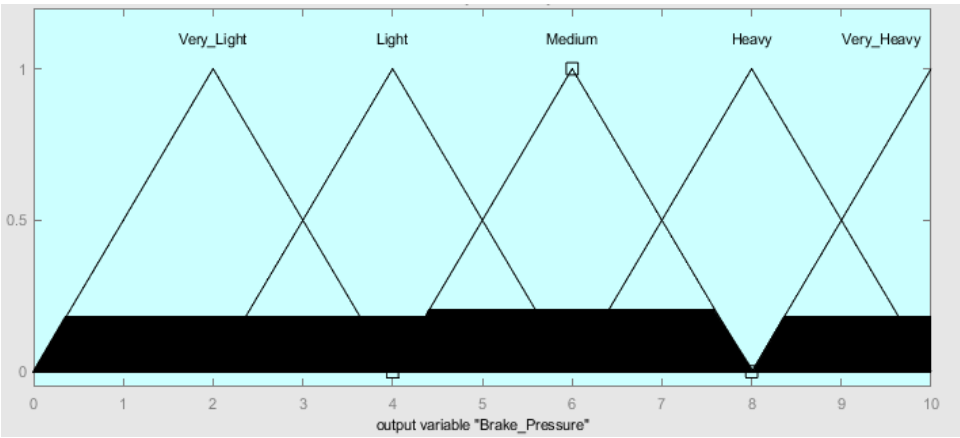


Figure 10. Aggregated Output fuzzy set

Defuzzification - Centroid

The final step is to find a final, defuzzified output value from that fuzzy set, the crisp output.
For our model, we use the centroid method.
The centre of mass of the area under the line is found and that value is the final output value.

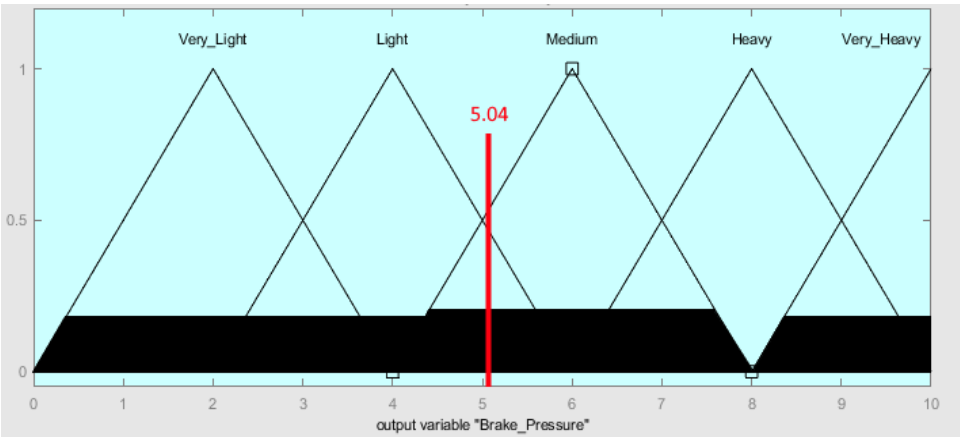


Figure 11. Find centroid of the aggregated output fuzzy set

Results

MATLAB Fuzzy model

Taking the results of each crisp input combination and then generating a subsequent crisp output these can be plotted to generate a surface that reflects the model. This is our final MATLAB fuzzy model below.

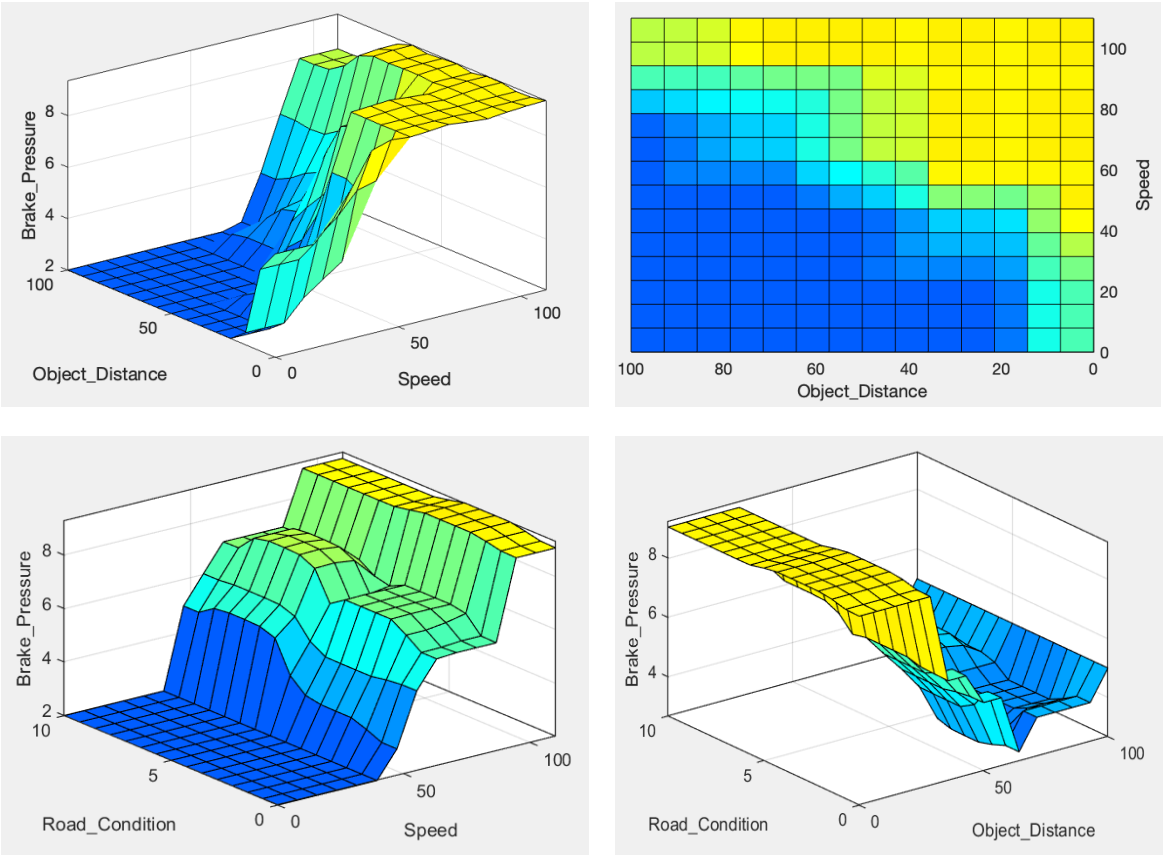


Figure 12. Final fuzzy model (MatLab only displays the surface with two inputs and the one output, different combinations of inputs can be viewed via the interface)

The MATLAB model was tested and evaluated using the rule viewer window to demonstrate its results.

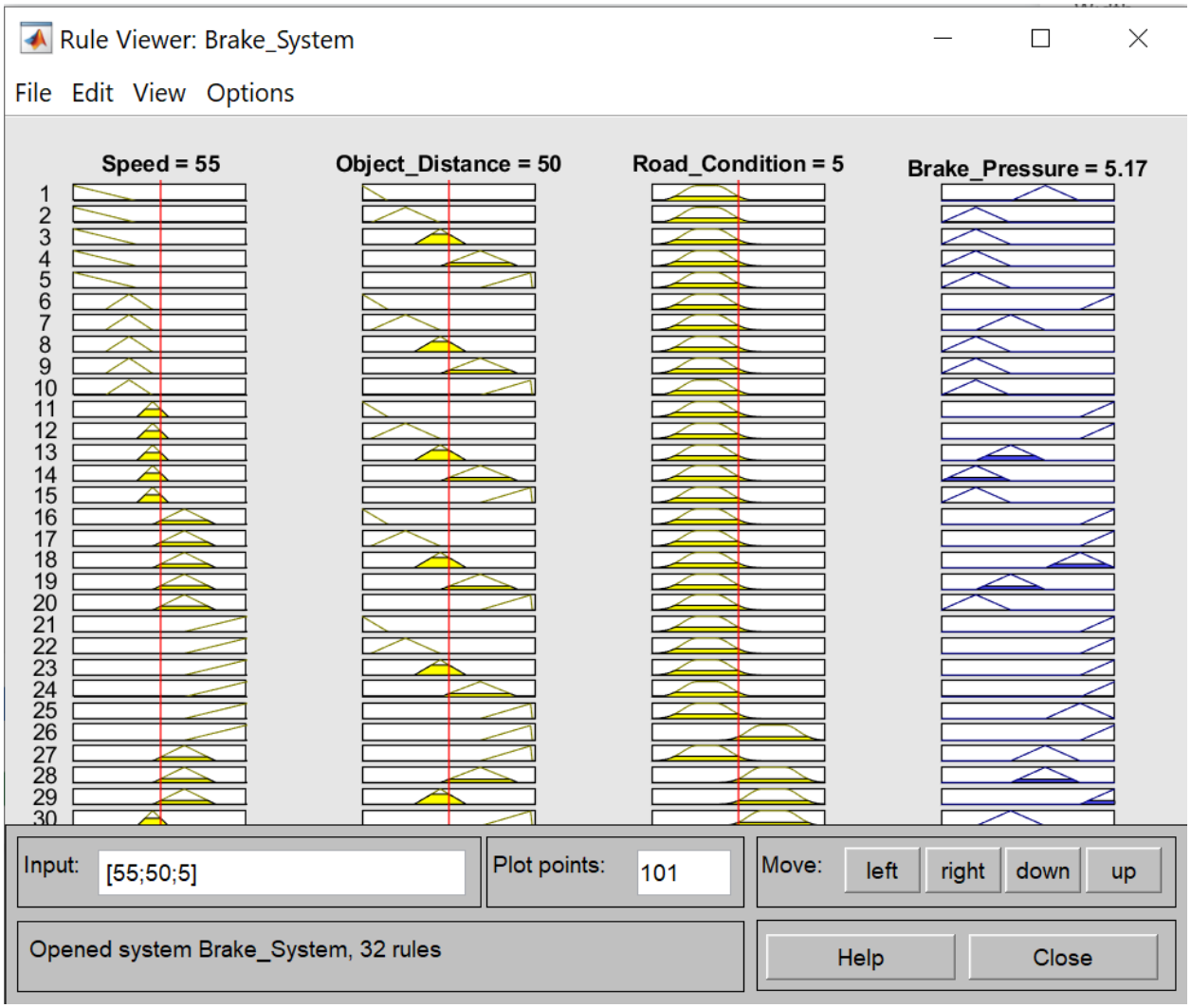


Figure 13. Viewing the Inputs versus the outputs in MATLAB Rule Viewer

Python Model

In order to help test, tune, and ensure the rules were able to satisfy the braking distance requirements, a python module was written and included in the Python model project. To verify this, the Distance vs Braking Pressure graph at a constant speed was analysed. Note that the Python model does not include Road Conditions as an input.

The system should slow down proactively, doing some light braking as it approaches the detected object to match speed without fully engaging the brakes. In practice this would increase occupant comfort and help to maintain vehicle separation.

Some problems were found while testing and tuning the rules.

For example, it was found that there was an issue when the distance was around 28m [Close] to 40m [Near] and speed between 50 km/h to 60 km/h [Medium].

An attempt was made to adjust the rules such that the brake pressure would be less than the maximum when the speed is “Medium” and distance is “Close” or “Near”.

However when tested, while the brake pressure in that situation was less than the maximum, it also reduced the pressure when the distance was 14m and the speed was 50 to medium brake pressure

That causes the model to no longer fit the Braking Distance statistics, and while that means the brakes may be maxed out at a greater distance, that is acceptable, as long as the brakes are always maxed before the distance in the statistics.

Speed	Distance	Brake Pressure
40	18	4.55
50	28	9.31
60	40	9.22
70	54	6.12
80	72	4.84
90	90	8
100	112	8
110	134	8

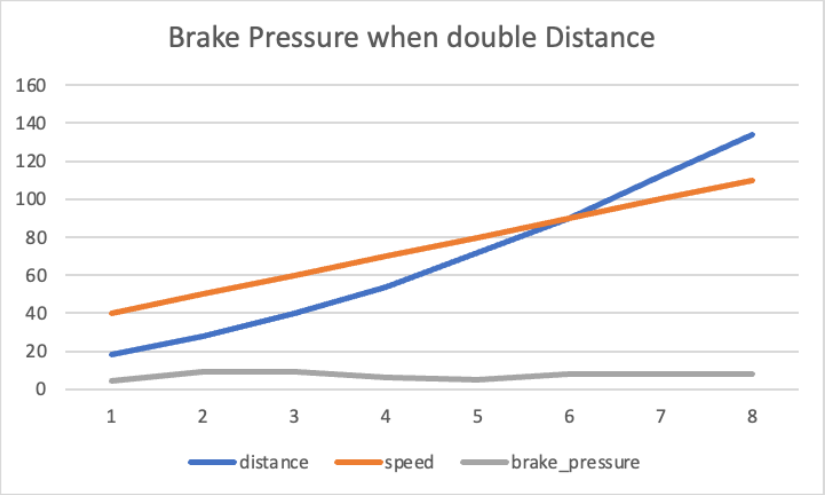
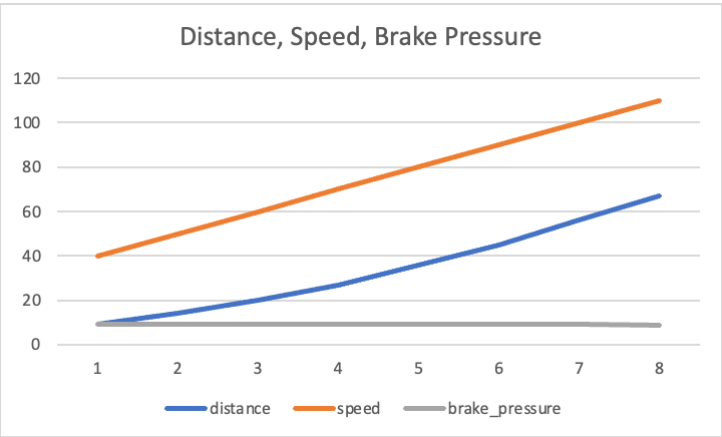


Figure 14. Brake Pressure when Double Distance compared as Stopping Distance

The reason this may have happened is that some of the membership functions may have needed to be tuned further. Possibly by giving a more even spread over the graph space.

After some tuning, The fuzzy model not only successfully managed to have the maximum brake pressure at the braking distance mentioned in the Queensland Data Sheet, but also began applying brakes gently at a greater distance for more occupant comfort.



Speed	Braking distance	Braking Pressure
40km/h	9m	9.24
50km/h	14m	9.14
60km/h	20m	9.22
70km/h	27m	9.32
80km/h	36m	9.18
90km/h	45m	9.22
100km/h	56m	9.18
110km/h	67m	9.07

Figure 15. Model Inputs and Output Results compared to the Stopping Distance Dataset

To further demonstrate, the model was tested with a speed of 70 km/h, demonstrating this brake force curve. The curve matched expectations.

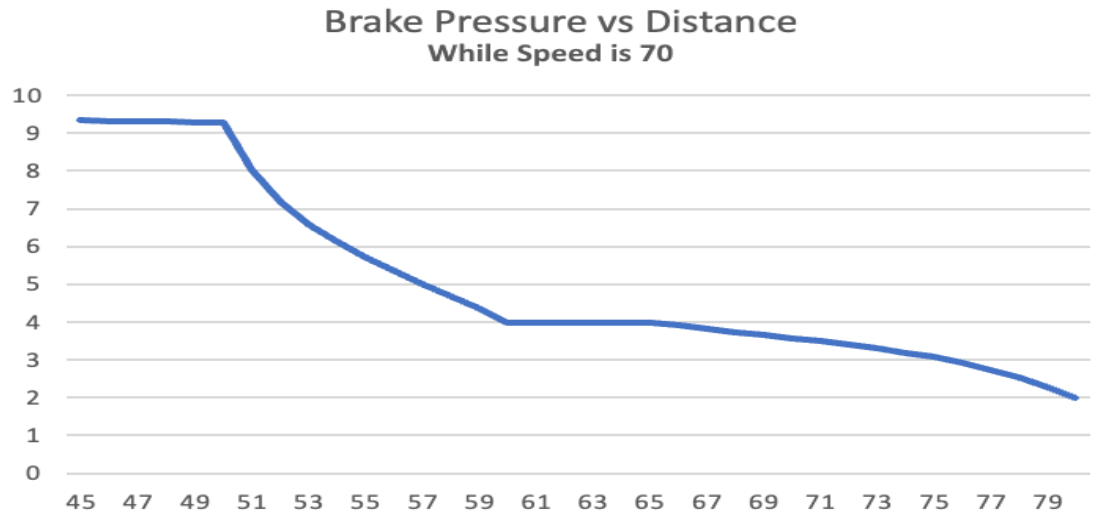


Figure 16. Brake Pressure vs Distance, Speed is 70km/hr

To further ensure that the braking pressure is applied following this curve, the model was tested at a number of other distances over the graph's speed range.

The results show that the pressure matches what was described over that range.

distance	speed	brake_pressure		distance	speed	brake_pressure
9	40	9.240229885		18	40	4.546801773
14	50	9.141176471		28	50	9.311111111
20	60	9.222222222		40	60	9.222222222
27	70	9.322875817		54	70	6.120454545
36	80	9.183333333		72	80	4.83725141
45	90	9.222222222		90	90	8
56	100	9.183333333		112	100	8
67	110	9.070589488		134	110	8
distance	**speed**	**brake_pressure**		**distance**	**speed**	**brake_pressure**
27	40	4		36	40	2.818376068
42	50	4		56	50	2.818376068
60	60	4		80	60	2
81	70	2		108	70	2
108	80	4.210526316		144	80	4.210526316
135	90	8		180	90	8
168	100	8		224	100	8
201	110	8		268	110	8

Figure 17. Brake Pressure evaluated at different speeds and distances

Pygame was used with a simple test situation consisting of the vehicle and a stationary object in front of it. The simulation was run with the speed of the car set to 100 km/h. And the resulting simulation shows the car braking as it approaches the object, before coming to a stop, avoiding collision with it.

A video of it in action can be watched here.

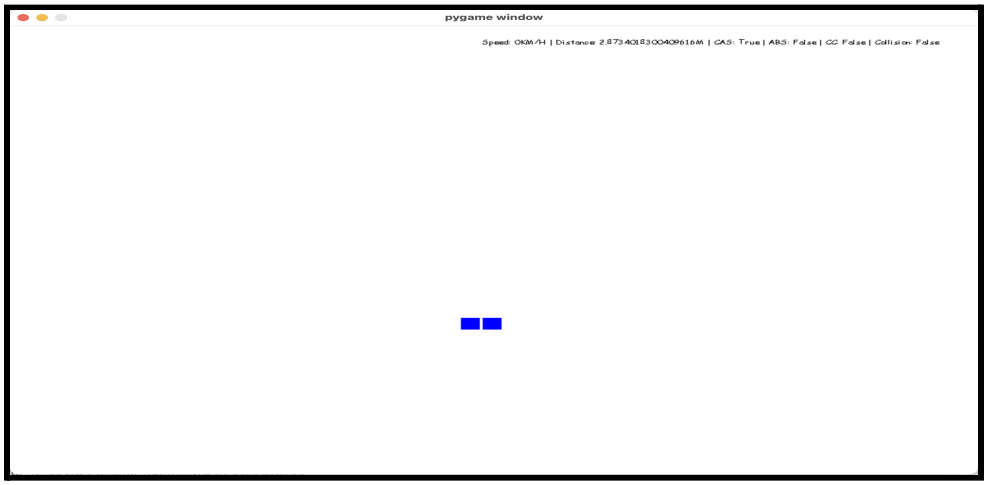


Figure 18. Pygame simulation screenshot, vehicle coming to stop

Our model will eventually brake when it is activated, therefore our system needs to communicate with the vehicle controller and let it decide when our system should be activated.

Limitations and Constraints

The design is limited in that it only tackles a surface level of control, with only some variables considered. The complex nature of a car's braking system is not fully represented in the model.

The design is also limited to just the collision avoidance braking system.
The current model could have been integrated into a larger system and composed with other components to create even more complex behaviour.

It is important to note that our simulation does not strictly follow physics equations. For example if the vehicle does not crash in the simulation, it does not mean that it will definitely not crash in the real world. It only demonstrates that the fuzzy model would theoretically work.

According to our source for braking distances, there are at least 10 factors affecting the stopping distances (Queensland Government, 2016)**.** However with fuzzy logic, it is not possible to add as much as features we want, the reasons are listed below:

- The more input/output variables, and linguistic sets within each variable, the more potential rules there may be.
  Depending on the input, there may not be an easy, smart way to implement a rule that covers a large number of situations, leading to rules potentially needing to be added for every permutation.

- Realistically, some inputs are hard to measure. For example, it is extremely non-trivial to detect the amount of tyre wear or to measure (potential) weight transfer.

These limitations and constraints were considered and it was decided that the model would:

- Not include the weight of the vehicle. This is a big limitation as weight can potentially have a non-negligible effect on the stopping distance of a vehicle. The maths involved to calculate how weight affects stopping distance is non-trivial as can be seen in a number of forum posts (*Braking Distances, and Weight*, 2006). The amount of friction a tyre generates is non-linear with regards to weight and a number of other physical factors make it a far more complex issue

  - Ignore tyre condition, worn treads can negatively impact performance by extending the required braking distance due to reduced wheel traction.

- Not account for physical processes, such as wheel slip, that may significantly affect results. The reason is similar to the point above.
  According to our calculation, the stopping distance will be increased by **44.62% on average**, therefore we think we can apply some mathematical equation to increase our brake pressure accordingly.

Dry vs Wet difference in %
40 km/h = +1.4444 50 km/h = +1.4286 60 km/h = +1.45 70 km/h = +1.4815 80 km/h = +1.4444 90 km/h = +1.4444 100 km/h = +1.4286 110 km/h = +1.4478 avg = +1.4462

Table 2. Difference in Stopping Distance, Wet and Dry Road Conditions

- We never considered programming ethics into the system. The ethical dilemmas that accompany autonomous vehicles are still being explored by world class experts through large scale surveys. The topic produces profound questions that require heavy consideration. Is it possible to have a global unified moral code for self-driving vehicles to adhere to, or will the programmed ethics differ country by country.
- Only accounts for obstacles in the same and straight lane that the object is approaching
    - It does not avoid collisions that may result from other directions, i.e. being rear ended, or objects going around a corner.
    - It doesn't predict other issues, i.e. a car attempting to suddenly change into the cars lane.
- Not be active at all times. The vehicle must have some controller to control when the collision avoidance system should be activated, for example it should be inactive when the vehicle is at low speeds, otherwise the system would constantly be trying to brake while the driver is trying to complete ordinary manoeuvres, such as exiting a parking spot.

Who did what

Python model and implementation
- Nelson

MatLab model
- Will
- Aiden

Report Graphs and Maths
- Nelson
- Matt

Report and Presentation Content
- Everyone

Report Formatting and General Editing
- Bridget

Presentation Slides
- Benedict

References

Braking distances, and weight. (2006, May 10). Physics Forums. Retrieved April 9, 2022, from https://www.physicsforums.com/threads/braking-distances-and-weight.120420/

Fuzzy Inference Process - MATLAB & Simulink. (n.d.). MathWorks. Retrieved April 9, 2022, from https://au.mathworks.com/help/fuzzy/fuzzy-inference-process.html

Izquierdo, S. S., & Izquierdo, L. R. (2018, June 30). Mamdani Fuzzy Systems for Modelling and Simulation: A Critical Assessment. *Journal of Artificial Societies and Social Simulation*, *21*(3), 2. doi: 10.18564/jasss.3660

Mathworks. (n.d.). *Fuzzy Logic Toolbox - MATLAB*. MathWorks. Retrieved April 8, 2022, from https://au.mathworks.com/products/fuzzy-logic.html?s_tid=srchtitle_fuzzy_1

Mining-technology.com. (2021, November 16). *Revealed: the mining companies leading the way in autonomous vehicles*. mining-technology.com. Retrieved April 8, 2022, from https://www.mining-technology.com/

Queensland Government. (2016, November 14). *Stopping distances: speed and braking | Transport and motoring*. Queensland Government. Retrieved April 8, 2022, from https://www.qld.gov.au/transport/safety/road-safety/driving-safely/stopping-distances

scikit-fuzzy development team. (n.d.). *skfuzzy 0.2 docs — skfuzzy v0.2 docs*. PythonHosted.org. Retrieved April 8, 2022, from https://pythonhosted.org/scikit-fuzzy/

The State of Queensland. (2016, November 14). *Stopping distances on wet and dry roads | Transport and motoring*. Queensland Government. Retrieved April 7, 2022, from https://www.qld.gov.au/transport/safety/road-safety/driving-safely/stopping-distances/graph

Üneş, F., Demirci, M., Zelenakova, M., Çalışıcı, M., Taşar, B., Vranay, F., & Kaya, Y. Z. (2020, August 29). River Flow Estimation Using Artificial Intelligence and Fuzzy Techniques. *Water*, *12*(9), Figure 7. http://dx.doi.org/10.3390/w12092427