Fuzzy Logic: Gradual halt of Car at Traffic Light

Intelligent Systems Project

Chasanidis Dimitris
Alexander Technological Educational Institute of Thessaloniki
Thessaloniki, Greece
dimitrischasanidis@gmail.com

Makris Giorgos
Alexander Technological Educational Institute of Thessaloniki
Thessaloniki, Greece
makris.giwrgos1234@gmail.com

Makrygiannaki Stefania
Alexander Technological Educational Institute of Thessaloniki
Thessaloniki, Greece
stefania.mak.project@hotmail.com

Abstract— **In order for us to understand better how fuzzy logic and fuzzy rules work, we decided to implement a gradual braking system that follows fuzzy rules in order to determine how much it has to use the braking system to avoid accidents.**

Keywords-component; Fuzzy Logic, Car, Traffic Light, Fuzzy Rules

I. Introduction

Some algorithmic problems do not have a black or white solution, but a greyer, not precise one. Fuzzy logic has been around for many years, and its uses varies. It is used to solve problems closer to the human rather than mathematics, and in order for us to fully understand what it is, and how it works we applied it to a simple halt of car problem and see its efficiency.

Our goal is to move forward without violating the Highway Code, providing the car with only two fuzzy information; one being the traffic light color, and the second how far is it from the car. With the success of this project, the car can judge what's the greatest speed to keep going forward.

II. Technologies Used/ p5.js

p5.js is a JavaScript sketchbook library made for learning and prototyping. It offers a wide range of modules including but not limited to drawing, sound manipulation, plotting and collision detection. Since it was created as a learning tool with detailed documentation, accessible to both hobbyists and seasoned programmers, it seemed the appropriate choice to create our small platform without spending too much time getting accustomed to it. In general, a great tool for not very complex visualizations that does not get in the way of creating something functional.

III. Fuzzy Logic

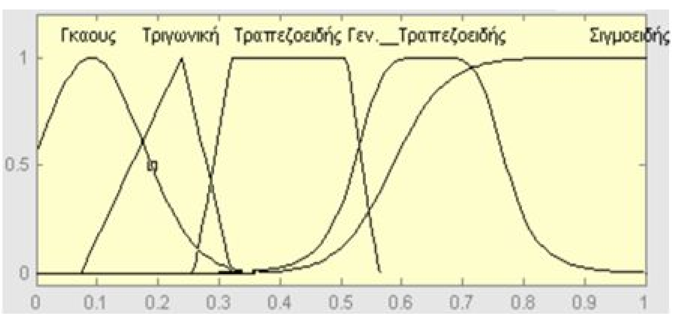
In order to understand fuzzy logic we need to familiarize ourselves with fuzzy systems. Fuzzy systems suggest a new information representation. Before those systems were introduced a system could only be represented by a set of characteristics that could only belong to one of two states; true or false. This binary encoding of information works quite well in simple systems. But when are dealing with complex systems/problems which require precise results, it does not provide great result, because when a problem gets even more complex its characteristics become more complex. People tend to think this way. When someone describes how much he enjoyed a movie he will most likely answer one of the following:

- Bad movie
- Could've been worse
- Good, I enjoyed it
- Excellent

We see that his possible answers are not strictly “Excellent” or “Terrible”. Instead there are states in the middle. When we try to represent a complex feature with a simple representation we lose much of the information that it carried. In our previous example if we try to binarize their possible answers we will not get all the information they provided us. We will lose information. This is where fuzzy systems thrive. Fuzzy systems introduce a new way of representing information, a way that is not binary. A variable now can take a real value (between 0 and 1). Also such a variable can affect more than one systems in different degrees. This way we have the means to describe uncertainty/vagueness and more importantly describe human knowledge. This idea looks quite similar with what probabilities are. In order to distinguish certainty/ possibility and probability we will use an example.

A person has a rather low probability of eating 10 eggs for breakfast, but it’s quite certain that he could.

In order to use this new information representation system we first must turn those values into expressions. This process is called fuzzification. A fuzzy system can take values $x \in [0, 1]$ and can affect the system differently.



In the picture we see different function that represent just how much a variable can affect the whole system. The width represents the values that x can take and the height represents the function of participation.

Fuzzy systems mathematical operations that are the counterparts of boolean operations:

Boolean	**Fuzzy**
AND(x,y)	MIN(x,y)
OR(x,y)	MAX(x,y)
NOT(x)	1-x

With that in mind, let’s dive in **fuzzy logic**. Fuzzy logic is a form of many-valued logic, a natural extension of boolean logic and provides approximations and conclusions based on fuzzy systems. This approximation thinking is a way to map and replicate how humans think. People think and arrive to conclusions based on approximation rules/criteria instead of strict mathematical rules. There are 2 types of fuzzy systems. Mandami and Sugeno. Mandami systems have outputs that fuzzy variables that need to combined in order to get a valuable conclusion/ result. Sugeno on the other hand produce real value results.

Fuzzy logic’s target is to create systems that produces conclusions in order to get actionable results from uncertain positions. This process has the following steps:

1. Input fuzzification

For each input variable / unclear value, we determine its degree of participation.

2. Apply fuzzy operations

Rules are applied to the new, defuzzed, variables.

3. Apply “if-else” rules

The conclusion is reshaped based on the degree of support of the rule by applying the AND operator.

4. Summation of all fuzzy exit variables

The various vague sets that are the outputs of the different rules (for the variable itself) are combined into an unclear set for each output variable.

5. Defuzzification

There are five ways to determine the defuzzed result, by computing:

- centroid (center weight)
- bisector (area)
- middle of maximum (the average of the maximum value of the output set)
- largest of maximum
- -smallest of maximum

Fuzzy logic is not a first choice problem solution. As a tool it is suggested when high value accuracy is not a necessity. In that case we should use other methods.

IV. Fuzzy Implementation

In our project, we decided to tackle a common problem in fuzzy logic examples; the car’s braking system problem. More specifically, given a car with speed V, we had to determine given its distance from the traffic light how much it would have to slow down in order to not get fined by the police.

First step had to be the fuzzification of inputs. For each p5.js tick we get the car's distance X from our traffic light mapping each possible value of X to 3 categories:

- Close
- Middle
- Far

We named the **"fuzzied" variable tlDistance,**

Next we take the traffic light state (Green, Orange, Red) and insert it into a **variable called tlColor**.

Second step, was to apply fuzzy operations. Specifically we applied the following:

1. If Car is <u>Close</u> to the Traffic Light OR Traffic Light is <u>Red</u>, then <u>Slow Down</u>.
   [if tlDistance == Close OR tlColor == Red then Slow Down]
2. If Car is <u>neither far or close</u> to Traffic Light OR Traffic Light is <u>Green</u>, then Keep <u>Middle Speed</u>.
   [if tlDistance == Middle OR tlColor == Green then Keep Speed]
3. If Car is <u>Far</u> from the Traffic Light OR Traffic Light is <u>Orange</u>, then <u>Speed Up</u>.
   [if tlDistance == Far OR tlColor == Orange then Speed Up]

In the third step, we had to calculate how much did each rule apply to the current state of the car. Specifically for the following cases:

Case 1: Slow Down

Case 2 : Middle Speed

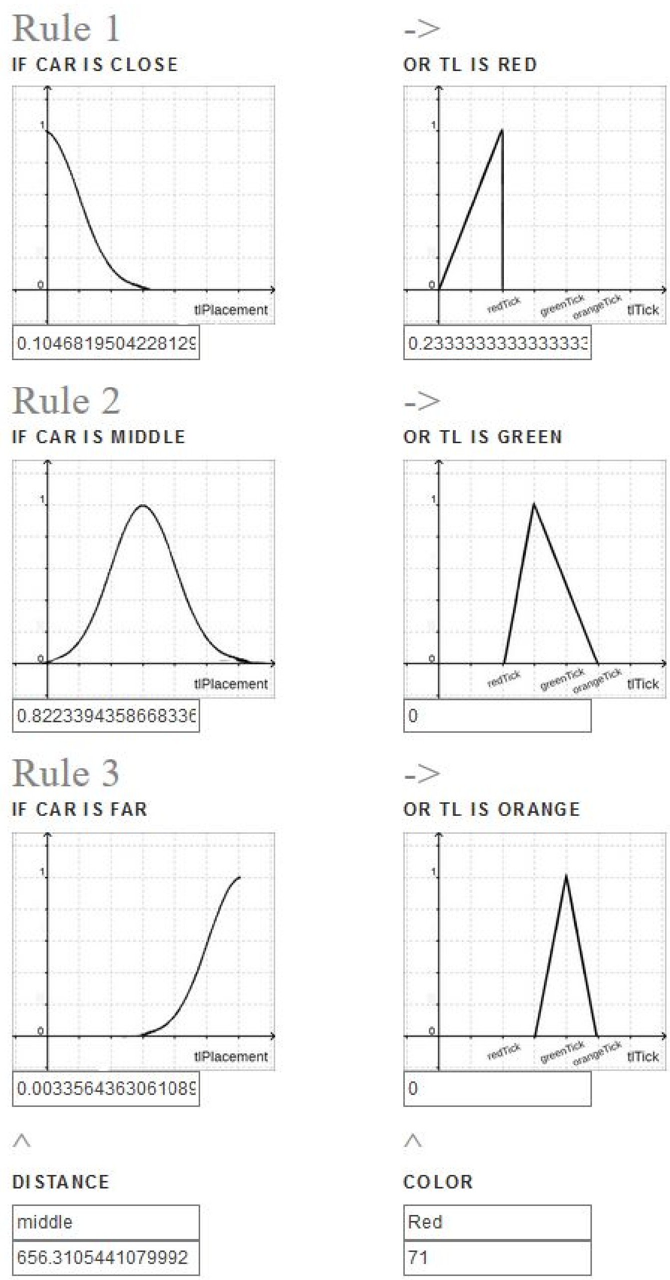
Case 3 : Speed up

Each case provides from a scale 0% to 100%, of how much should the car slow down, keed a middle speed or speed up.

1st case:
max(gaussianMF(tlDistance, tlPlacement / 5.5, 0), triangularMF(tlColor, 0 , redTick, redTick))

2st case:
max(gaussianMF(tlDistance tlPlacement / 5.5, tlPlacement / 2), triangularMF(tlColor, redTick, redTick + orangeTick, redTick + orangeTick + greenTick))

3st case:
max(gaussianMF(tlDistance tlPlacement / 5.5, tlPlacement), triangularMF(tlColor, redTick + orangeTick, redTick + greenTick, redTick + orangeTick + greenTick))

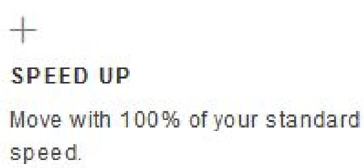


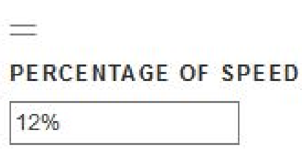
In the fourth step we sum the values of the above rules. Specifically:

$$car\ speed_{choice} = case_1 \cdot 0 + case_2 \cdot 0.5 + case_3 \cdot 1$$

The weights were chosen in a way that they would actually affect the final result just as it was needed.

Our final step is the defuzzification step, in which we translate our carSpeedChoice into a real usable value of speed. We do this by dividing by 3 and calculating the centroid.

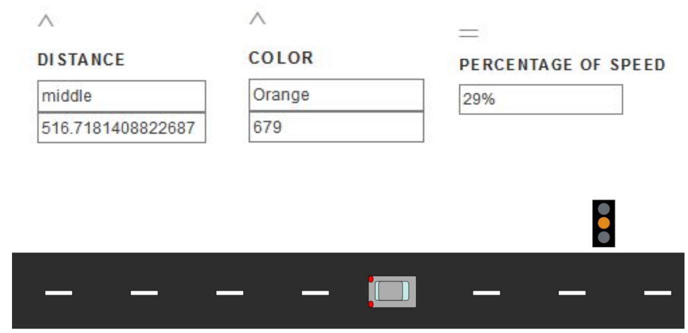




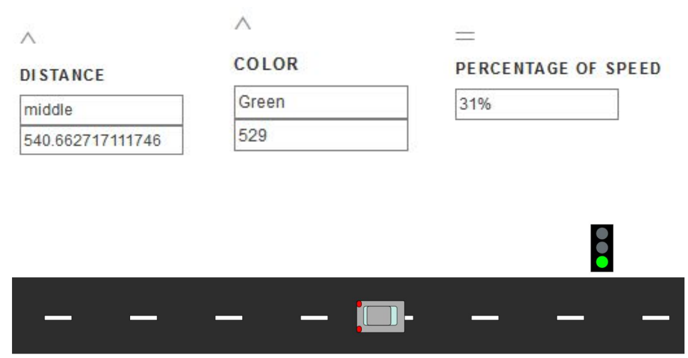
Some few of the fuzzy cases can be the following scenarios:

First scenario: Moving with 0% of the speed, meaning the car has stopped right before the traffic light.

Second scenario: Moving with 29% of the speed, as the car expects a red light while getting closer to the traffic light.



Third scenario: Driving with around 30-40% of its speed, when the green light is on.



While these being three cases, the possibilities are countless with a variety of results.

V. Conclusion

In conclusion, we developed an artificial road and car in order to test how fuzzy logic will help in order to determine how much should a car decrease its speed in front of a traffic light. We found out that indeed, fuzzy rules in this particular set of problems can be a valuable tool to solve them without overcomplicating them.Thus, we successfully saw the car following our three rules we setted, following the Highway Code

References

[1] Fuzzy Logic Toolbox: https://moodle.teithe.gr/pluginfile.php/73564/mod_resource/content/0/Fuzzy%20Logic%20Toolbox.pdf

[2] Fuzzy: https://aetos.it.teithe.gr/~adamidis/IntelSys/fuzzy.pdf

[3] Octave Forge, gaussfm: https://octave.sourceforge.io/fuzzy-logic-toolkit/function/gaussmf.html

[4] MathWorks, gaussfm: https://uk.mathworks.com/help/fuzzy/gaussmf.html

[5] MathWorks, trimf: https://octave.sourceforge.io/fuzzy-logic-toolkit/function/trimf.html

[6] I. Μαρινάκης, Μ. Μαρινάκη, Ν. Φ. Ματσατσίνης, and Κ. Ζοπουνίδης, Μεθευρετικοί και Εξελικτικοί Αλγόριθμοι σε Προβλήματα Διοικητικής Επιστήμης. Θεσσαλονίκη: Κλειδάριθμος.

[7] Javascript Documentation: https://developer.mozilla.org/en-US/

[8] P5.js Documentation: https://p5js.org/reference