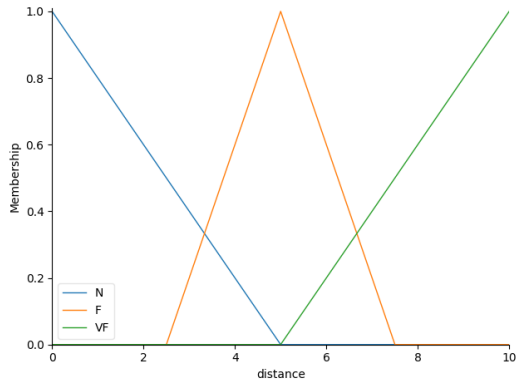
ECE657 Assignment 4 – Hammad Rehman, 20293340, harehman

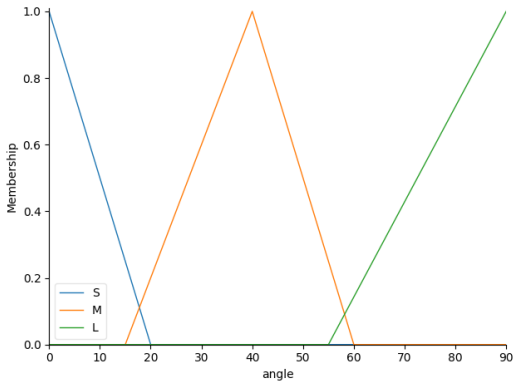
3) Genetic algorithms almost seem to be at a lower level than genetic programming. According to the lecture slides, genetic algorithms work to optimise the output of a given function f by changing its inputs (i.e. genotype). Genetic programming takes this one step further by optimising entire programs which could consist of multiple functions, and indeed, could contain DAGs. Genetic algorithms mutate (bit-flips, crossovers etc.) the individual "chromosomes" which feed the function in order to find a global minimum or maximum. Genetic programming will take an entire program, and attempt to optimise it to obtain the program with the highest fitness. Mutations in this case could include swapping different functions out at different points in the DAG, replacing an entire sub-DAG with another DAG etc.

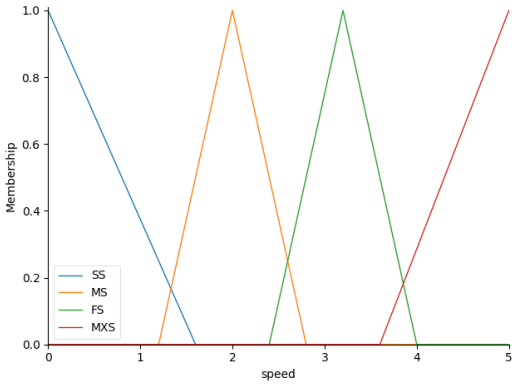
4) Rules:

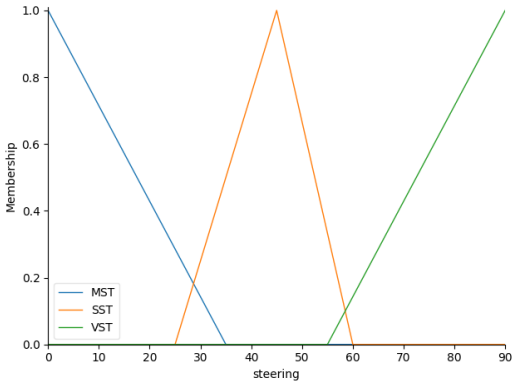
- If D is N and A is S, S is SS and ST is VST (obstacle in front)
- Else if D is N and A is M, S is SS and ST is SST (obstacle near but not in front)
- Else if D is N and A is L, S is MS and ST is MST (obstacle out of the way)
- Else if D is F and A is S, S is MS, ST is VST(obstacle in the way but not close)
- Else if D is F and A is M, S is MS, ST is SST (obstacle far but not in front)
- Else if D if F and A is L, S is FS, ST is MST (obstacle far and out of the way)
- Else if D is VF and A is S, S is FS, ST is VST (obstacle very far but on collision course)
- Else if D is VF and A is M, S is MXS, ST is SST (obstacle very far and medium angle)
- Else S is MXS, ST is MST (obstacle very far and at big angle -> no obstacle)

The membership function graphs I chose are shown below:









I experimented with various degrees of overlap, but this seems to produce optimal results. I considered using Sugeno initially, but it appears that Sugeno is more suited to "PID style controllers", similar to what would operate in an industrial environment (my undergraduate background is in Chemical Engineering, so I have a fair bit of experience with those type of PID-loops) [1]. According to the same reference, Mamdani is more suitable for a system with human input, which is closer to this system. As an example, humans don't mathematically model the angle and distance of an obstacle to determine the degree of speed/steering adjustment; we just go by intuition, and Mamdani's ability to generate a fuzzy output is more suitable for this.

For the defuzzification, I used MOM for speed and LOM for steering. I could have gone with LOM for speed as well, but then that wasn't suited to the situation where the obstacle was too close. Going with SOM for speed would have meant speed wouldn't be maximised with no obstacle in the way. However, for safety reasons, LOM seemed most appropriate, as it allowed maximum steering to avoid close obstacles, at the expense of minimal steering when the obstacles weren't close.

Here is a table detailing 3 scenarios I tested, showing inputs (A/D) and outputs (S/ST):

A	D	S	ST
90	10	5	0
15	1	0.5	90
60	8	4.4	30

Note that I have marked the points in my code you can change A and D to test the system yourself (P1/P2).

Bibliography

[1] Mathworks, "Mamdani and Sugeno Fuzzy Inference Systems," 2020. [Online]. Available: https://www.mathworks.com/help/fuzzy/types-of-fuzzy-inference-systems.html. [Accessed 2 August 2020].