Project 1:
Fuzzy Logic
Control System

Report Created by: Avi Tombak

avitomba@buffalo.edu

University at Buffalo

CSE454 Fall 2021

10/4/2021

Table of Contents

Summary

This project is an autonomous control system for a robot. The control system is implemented using a fuzzy logic controller, with the goal of moving the robot from one location to another. The fuzzy logic control system follows a design process and is programmed using a Matlab script. This fuzzy logic design process consists of three stages; a fuzzifier, a fuzzy inference engine, and defuzzifier. The fuzzifying process involves the creation of fuzzy sets and membership functions. Real inputs are then fuzzified with these using these membership functions. The fuzzy inference engine creates fuzzy rules for the fuzzy sets. The defuzzifying process involves the outputs of these fuzzy rules and assessing membership functions with Lukasiewicz t-norms. Once membership functions are assessed with t-norms, they can be used alongside another method called Sugeno Fuzzy Integrals or s-norms in order to translates a fuzzy output into a real output. In the case of this system, it's real inputs are user entered and consist of a current location and destination. It's real outputs are changes in both forward acceleration and angular acceleration in order to move and turn the robot. This control system also has feedback, using saved inputs and outputs in order to determine it's next output, or in this case movement.

Keywords: Fuzzy Logic, Control System, Fuzzifier, Fuzzy Inference Engine, Defuzzifier, Fuzzy Sets, Fuzzy Rules, Membership Functions, Lukasiewicz T-Norm, Sugeno S-Norm, Matlab

System Inputs

User Inputs:

These inputs are entered by the user in the project's Matlab script. They consist of:

A timestamp, t which indicates how often the robot updates it's movement.

The current location of the robot as a Cartesian coordinate, $(x_, y)$.

A destination for the robot to move towards is entered by the user as a Cartesian coordinate, (x_{dest}, y_{dest}).

Control System Inputs (Sensed values):

The system's angle to the destination, θ and distance from current location to the destination, d are used by the robot's fuzzy controller in order to calculate its movement. Both θ and d are derived by using the current location and destination coordinates.

System Outputs

System Outputs:

The fuzzy logic control system can move the robot towards the destination by applying a forward acceleration, a and an angular acceleration, α. Both of these system outputs are derived values of d and θ with respect to time.

Matlab Script Outputs:

The Matlab script will output information in a command window as specified in the project instructions. The following values are outputted at each timestamp:

Weights of each membership function at that given time

New angular velocity, which can be derived from α, θ, and t.

New forward force, which can be derived from a.

Constraints

Project Requirements Specified Constraints:

The robot updates it's movement based on the fuzzy logic controller every one-hundred milliseconds. The initial angle to the destination and forward velocity are defined to be zero. This also implies initial forward acceleration is zero.

Other Constraints:

Initial timestamp will be zero and initial position will be at (0,0).

The angle to the destination, θ will have a range of $0 \leq \theta \leq \pi$.

The distance from current location to the destination, d will have a range of $0 \leq d \leq 100$.

The angular acceleration of the robot, α will have a range of $-1 \leq \alpha \leq 1$.

The acceleration of the robot, a will have a range of $0 \leq a \leq 1$.

Definitions

Current Location: (x, y)	Destination: (x_{dest}, y_{dest})	Timestamp: t
Angle to destination: θ	Angular velocity: ω	Angular acceleration: α
Distance: d	Forward Velocity: v	Forward Acceleration: a

where,

$t_o = 0$, $(x_o, y_o) = (0,0)$, $v_o = 0$ m/s, $\omega_o = 0$ rad/s, $a_o = 0$ m/s^2 , $\alpha_O = 0$ rad/s^2

$0 \leq \theta \leq \pi$ rad, $-1 \leq \alpha \leq 1$ rad/s^2, $0 \leq d \leq 100$ m, $0 \leq a \leq 1$ m/s^2

Physics Equations for Robot Movement

The project requirements specify using the $d = \sqrt{(a^2 + b^2)}$ formula for the distance calculation. Using the defined variables for current location and destination, distance is defined as:

$$d = \sqrt{[(x_{dest} - x)^2 + (y_{dest} - y)^2]}$$

Theta can be calculated trigonometrically as:

$$\theta = \arccos((x_{dest} - x) / d) \qquad \text{for } 0 \leq \theta \leq \pi \text{ rad}$$

The equations for average acceleration, velocity, angular acceleration and angular velocity are used for this project.

$$a = \Delta v/\Delta t,\ v = \Delta d/\Delta t,\ \alpha = \Delta\omega/\Delta t\ ,\ \omega = \Delta\theta/\Delta t$$

Applying them to the robot's control system will give equations for the system outputs a and α.

$$a = \Delta(\Delta d/\Delta t)/\Delta t = [(d_{new} - d_{old}) / 0.1 - (d_{old} - d_{older}) / 0.1] / 0.1$$

$$\alpha = \Delta(\Delta\theta/\Delta t)/\Delta t = [(\theta_{new} - \theta_{old}) / 0.1 - (\theta_{old} - \theta_{older}) / 0.1] / 0.1$$

The Matlab outputs of new angular velocity and new forward force can be calculated by:

$$\omega_{new} = 0.1\alpha - \omega_{old} = 0.1\alpha - (\theta_{old} - \theta_{older}) / 0.1$$

$$F = ma$$

Where $\Delta t = 100ms = 0.1s$, m = mass of the robot

Fuzzifier Design Process

The fuzzy design process can start by defining fuzzy sets linguistically for the control system's inputs and outputs. With these fuzzy sets, membership functions can be represented graphically. These membership graphs show the relation between an input/output and its membership on a scale of zero to one. These graphs can be created with linguistic reasoning instead of performing any calculations. With these graphs of the membership functions, piece-wise functions for each membership can be defined mathematically.

Fuzzy Sets:

This fuzzy logic control system will use four fuzzy sets, two mapping the system inputs θ and d to fuzzy inputs, and two mapping the system outputs α and a to fuzzy outputs.

θ = {VR, SR, ZE, SL, VL}

very right (VR)

somewhat right (SR)

straight (ZE)

somewhat left (SL)

very left (VL)

d = {ZE, VC, SC, SF, VF}

zero (ZE)

very close (VC)

somewhat close (SC)

somewhat far (SF)

very far (VF)

α = {L, SL, ZE, SR, R}

turn left (L)

turn somewhat left (SL)

don’t turn (ZE)

turn somewhat right (SR)

turn right (R)

a = {ZE, S, SS, SF, F}

stop (ZE)

go slow (S)

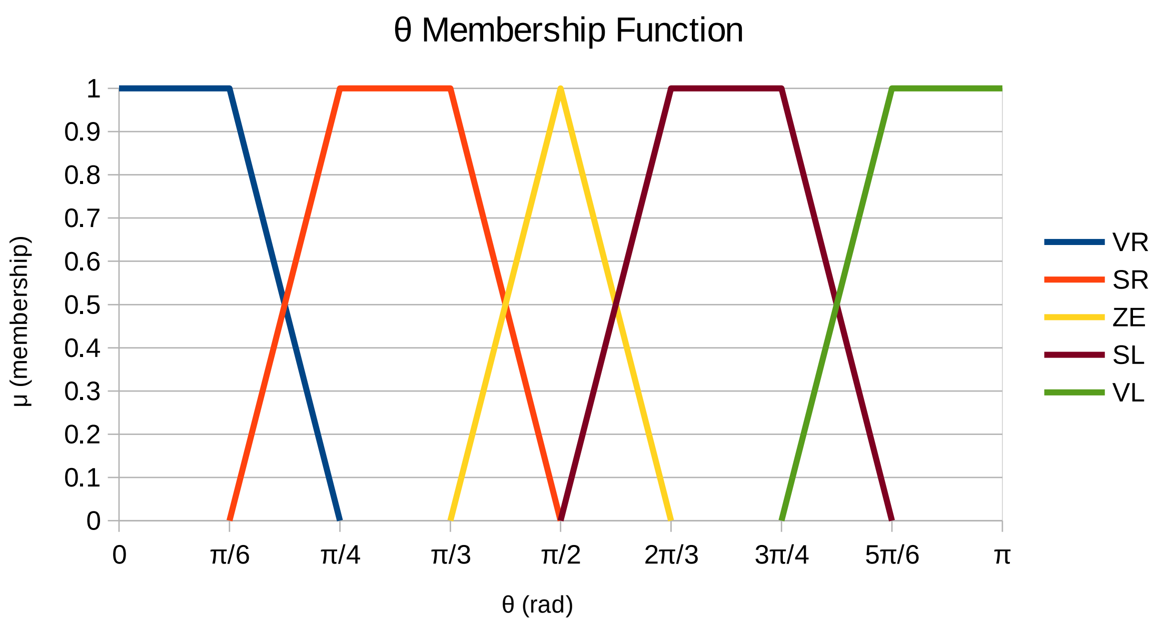
go somewhat slow (SS)

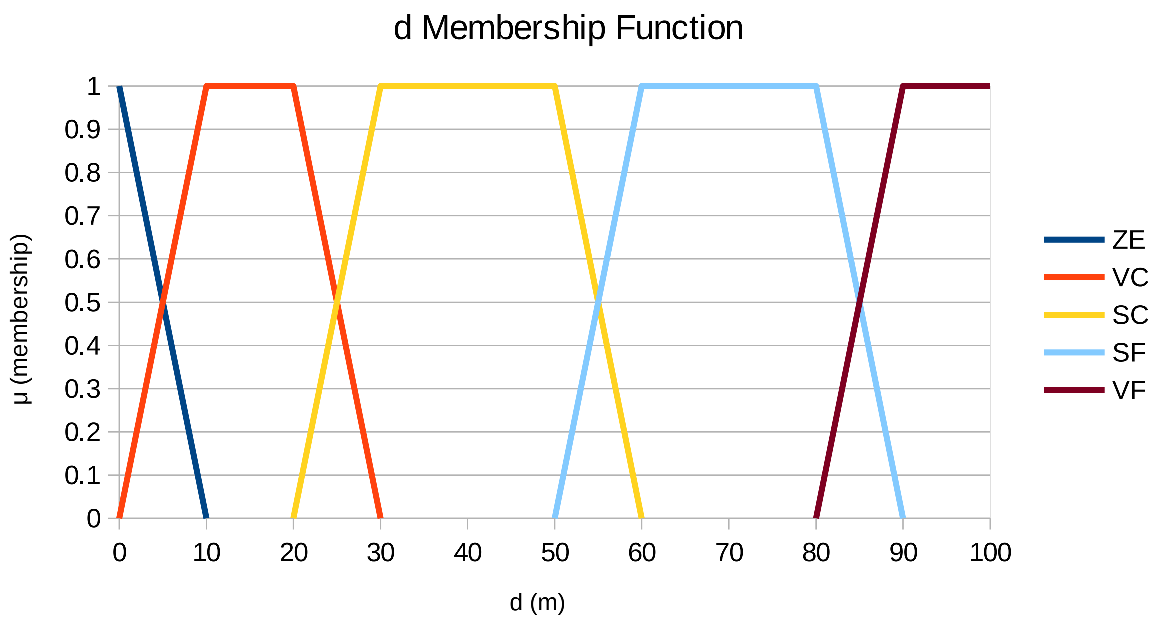
go somewhat fast (SF)

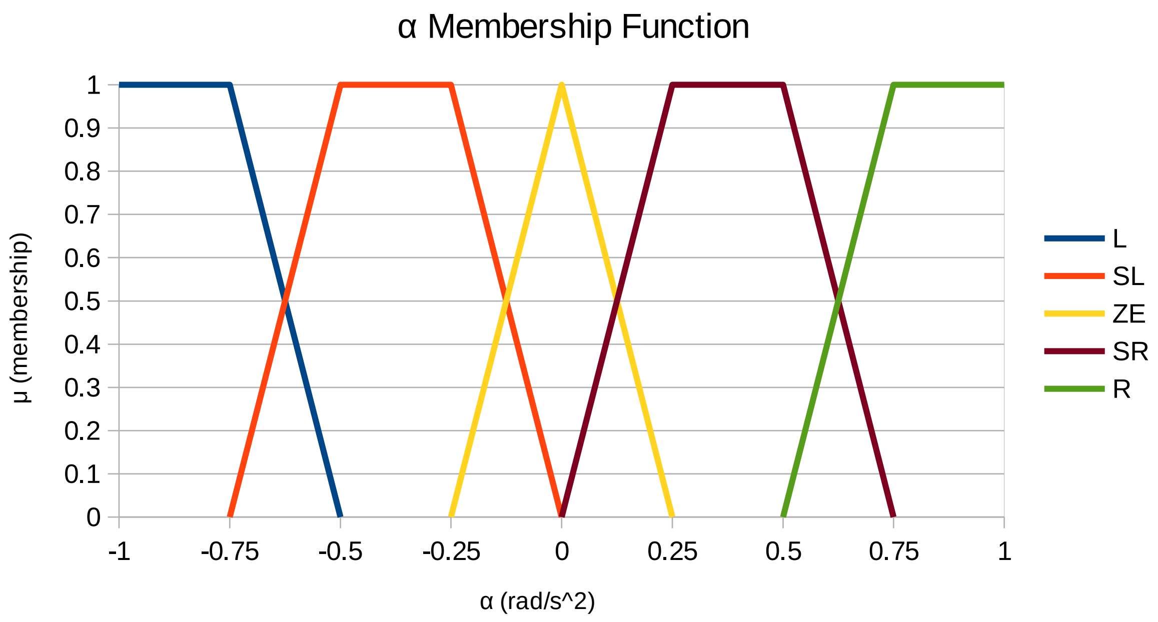
go fast (F)

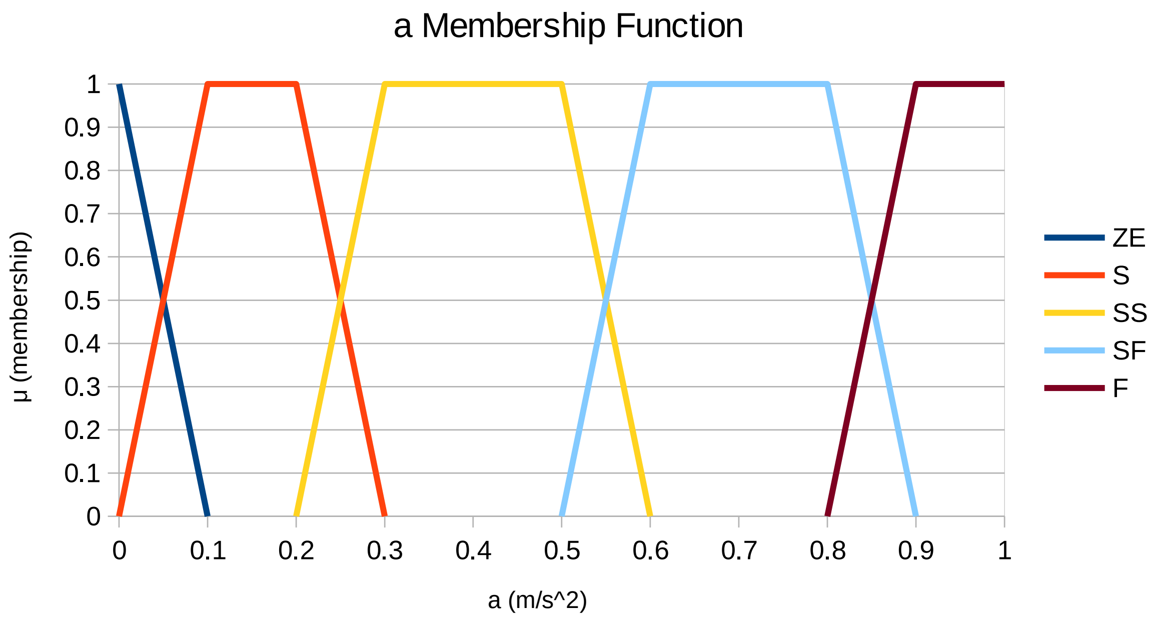
Membership Function Graphs:

Based on the fuzzy sets, membership functions can be graphically represented as follows. These graphs are made with linguistic reasoning and with the predefined range for each input and output.

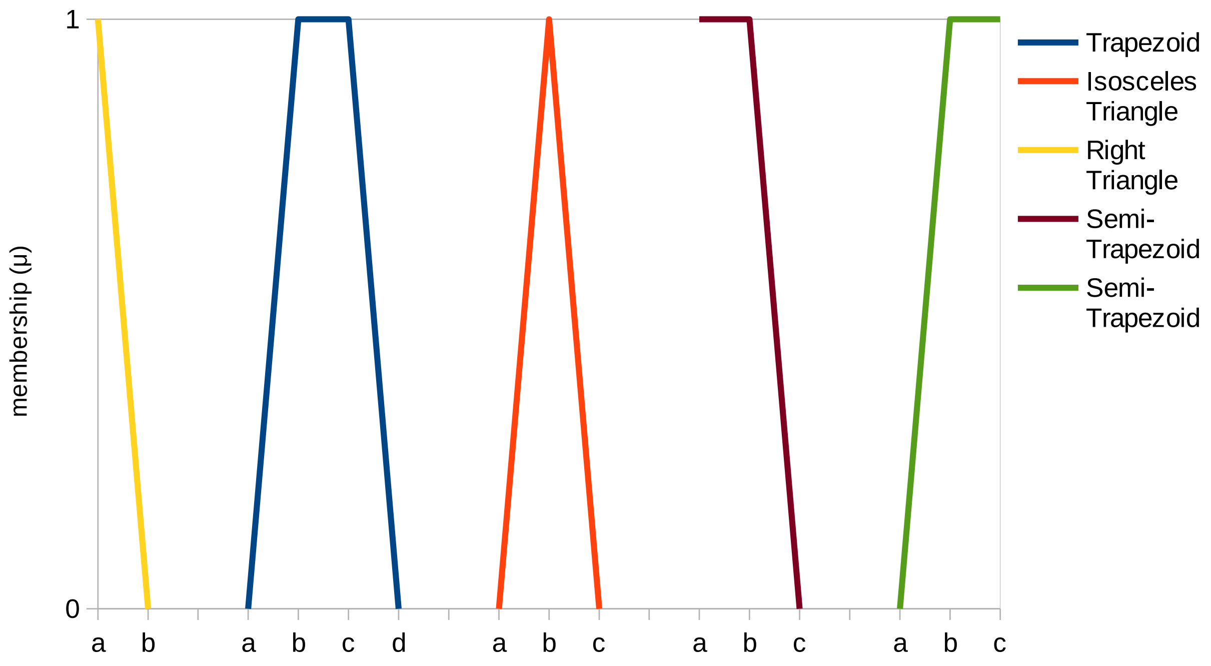





α Membership Function
μ (membership)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.75
-0.5
-0.25
0
0.25
0.5
0.75
1
α (rad/s^2)
L
SL
ZE
SR
R


a Membership Function
μ (membership)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
a (m/s^2)
ZE
S
SS
SF
F

Based on these graphical representations of the fuzzy sets, the fuzzy sets can be evaluated as piece-wise function corresponding to the shape of the membership function. The shapes used are trapezoids, semi-trapezoids, isosceles triangles, and right triangles. Note that semi-trapezoids and right triangles are at the edges of membership function graphs.



Their piece-wise membership functions for said shapes are defined as such:

$$\mu_{\text{Trap}}(x) = \begin{array}{ll} 0 & x \leq a \\ (x-a)/(b-a) & a < x < b \\ 1 & b \leq x \leq c \\ (x-d)/(c-d) & c < x < d \\ 0 & x \geq d \end{array}$$

$$\mu_{\text{Iso Tri}}(x) = \begin{array}{ll} 0 & x \leq a \\ (x-a)/(b-a) & a < x < b \\ 1 & x = b \\ (x-c)/(b-c) & b < x < c \\ 0 & x \geq c \end{array}$$

$$\mu_{\text{S. Trap}}(x) = \begin{array}{ll} 1 & a \leq x \leq b \\ (x-c)/(b-c) & b < x < c \\ 0 & x \geq c \end{array}$$

$$\mu_{\text{R. Tri}}(x) = \begin{array}{ll} 1 & x = a \\ (x-b)/(a-b) & a < x < b \\ 0 & x \geq b \end{array}$$

Twenty piece-wise membership functions can be created using these formulas, one for each member of each fuzzy set. These functions will be used to fuzzify a value in the predefined range of an input or output into it’s respective memberships.

θ Membership Functions:

$$\mu_{VR}(x) = \begin{cases} 1 & 0 \le x \le \pi/6 \\ (x - \pi/6) / (-\pi/12) & \pi/6 < x < \pi/4 \\ 0 & x \ge \pi/4 \end{cases}$$

$$\mu_{SR}(x) = \begin{cases} 0 & x \le \pi/6 \\ (x - \pi/6) / (\pi/12) & \pi/6 < x < \pi/4 \\ 1 & \pi/4 \le x \le \pi/3 \\ (x - \pi/2) / (-\pi/6) & \pi/3 < x < \pi/2 \\ 0 & x \ge \pi/2 \end{cases}$$

$$\mu_{ZE}(x) = \begin{cases} 0 & x \le \pi/3 \\ (x - \pi/3) / (\pi/6) & \pi/3 < x < \pi/2 \\ 1 & x = \pi/2 \\ (x - 2\pi/3) / (-\pi/6) & \pi/2 < x < 2\pi/3 \\ 0 & x \ge 2\pi/3 \end{cases}$$

$$\mu_{SL}(x) = \begin{cases} 0 & x \le \pi/2 \\ (x - \pi/2) / (\pi/6) & \pi/2 < x < 2\pi/3 \\ 1 & 2\pi/3 \le x \le 3\pi/4 \\ (x - 5\pi/6) / (-\pi/12) & 3\pi/4 < x < 5\pi/6 \\ 0 & x \ge 5\pi/6 \end{cases}$$

$$\mu_{VL}(x) = \begin{cases} 0 & x \le 3\pi/4 \\ (x - \pi/4) / (\pi/12) & 3\pi/4 < x < 5\pi/6 \\ 1 & 5\pi/6 \le x \le \pi \end{cases}$$

d Membership Functions:

$$\mu_{ZE}(x) = \begin{cases} 1 & x = 0 \\ (x - 10) / -10 & 0 < x < 10 \\ 0 & x \geq 10 \end{cases}$$

$$\mu_{VC}(x) = \begin{cases} 0 & x \leq 0 \\ x / 10 & 0 < x < 10 \\ 1 & 10 \leq x \leq 20 \\ (x - 30) / -10 & 20 < x < 30 \\ 0 & x \geq 30 \end{cases}$$

$$\mu_{SC}(x) = \begin{cases} 0 & x \leq 20 \\ (x - 20) / 10 & 20 < x < 30 \\ 1 & 30 \leq x \leq 50 \\ (x - 60) / -10 & 50 < x < 60 \\ 0 & x \geq 60 \end{cases}$$

$$\mu_{SF}(x) = \begin{cases} 0 & x \leq 50 \\ (x - 50) / 10 & 50 < x < 60 \\ 1 & 60 \leq x \leq 80 \\ (x - 90) / -10 & 80 < x < 90 \\ 0 & x \geq 90 \end{cases}$$

$$\mu_{VF}(x) = \begin{cases} 0 & x \leq 80 \\ (x - 80) / 10 & 80 < x < 90 \\ 1 & x \geq 90 \end{cases}$$

α Membership Functions:

$$\mu_{L}(x) = \begin{cases} 1 & -1 \leq x \leq -0.75 \\ (x + 0.5) / -0.25 & -0.75 < x < -0.5 \\ 0 & x \geq -0.5 \end{cases}$$

$$\mu_{SL}(x) = \begin{cases} 0 & x \leq -0.75 \\ (x + 0.75) / 0.25 & -0.75 < x < -0.5 \\ 1 & -0.5 \leq x \leq -0.25 \\ x / -0.25 & -0.25 < x < 0 \\ 0 & x \geq 0 \end{cases}$$

$$\mu_{ZE}(x) = \begin{cases} 0 & x \leq -0.25 \\ (x + 0.25) / 0.25 & -0.25 < x < 0 \\ 1 & x = 0 \\ (x - 0.25) / -0.25 & 0 < x < 0.25 \\ 0 & x \geq 0.25 \end{cases}$$

$$\mu_{SR}(x) = \begin{cases} 0 & x \leq 0 \\ x / 0.25 & 0 < x < 0.25 \\ 1 & 0.25 \leq x \leq 0.5 \\ (x - 0.75) / -0.25 & 0.5 < x < 0.75 \\ 0 & x \geq 0.75 \end{cases}$$

$$\mu_{R}(x) = \begin{cases} 0 & x \leq 0.5 \\ (x - 0.5) / 0.25 & 0.5 < x < 0.75 \\ 1 & 0.75 \leq x \leq 1 \end{cases}$$

a Membership Functions:

$$\mu_{ZE}(x) = \begin{cases} 1 & x = 0 \\ (x - 0.1) / -0.1 & 0 < x < 0.1 \\ 0 & x \geq 0.1 \end{cases}$$

$$\mu_{S}(x) = \begin{cases} 0 & x \leq 0 \\ x / 0.1 & 0 < x < 0.1 \\ 1 & 0.1 \leq x \leq 0.2 \\ (x - 0.3) / -0.1 & 0.2 < x < 0.3 \\ 0 & x \geq 0.3 \end{cases}$$

$$\mu_{SS}(x) = \begin{cases} 0 & x \leq 0.2 \\ (x - 0.2) / 0.1 & 0.2 < x < 0.3 \\ 1 & 0.3 \leq x \leq 0.5 \\ (x - 0.6) / -0.1 & 0.5 < x < 0.6 \\ 0 & x \geq 0.6 \end{cases}$$

$$\mu_{SF}(x) = \begin{cases} 0 & x \leq 0.5 \\ (x - 0.5) / 0.1 & 0.5 < x < 0.6 \\ 1 & 0.6 \leq x \leq 0.8 \\ (x - 0.9) / -0.1 & 0.8 < x < 0.9 \\ 0 & x \geq 0.9 \end{cases}$$

$$\mu_{F}(x) = \begin{cases} 0 & x \leq 0.8 \\ (x - 0.8) / 0.1 & 0.8 < x < 0.9 \\ 1 & x \geq 0.9 \end{cases}$$

Fuzzy Inference Engine

FIE Matrices:

Fuzzy rules can be created using a matrix. For this project the two FIE matrices were used. One comparing distance and acceleration to determine new acceleration, and one comparing angle and angular acceleration to determine new angular acceleration. The cells of this matrix each represent a fuzzy rule which is linguistically inferred. For example: if distance is somewhat close and acceleration is fast, go slow.

d

		ZE	VC	SC	SF	VF
	ZE	ZE	S	SS	SF	F
	S	ZE	S	SS	SF	F
a	SS	ZE	S	SS	SF	F
	SF	ZE	ZE	S	SF	F
	F	ZE	ZE	S	SF	F

θ

		VL	SL	ZE	SR	VR
	L	L	SL	ZE	R	R
	SL	L	SL	ZE	SR	R
α	ZE	L	SL	ZE	SR	R
	SR	L	SL	ZE	SR	R
	R	L	L	ZE	SR	R

From each matrix twenty-five fuzzy rules can be generated from the fuzzy sets. These rules follow the conventions:

If d and a, then a If θ and α, then α (i.e. if d=VF and a=SS then a=F)

Fuzzy Rules:

The fifty fuzzy rules produced by the FIE matrices are as follows

a Rules:

if d=ZE and a=ZE then a=ZE

if d=VC and a=ZE then a=S

if d=SC and a=ZE then a=SS

if d=SF and a=ZE then a=SF

if d=VF and a=ZE then a=F

if d=ZE and a=S then a=ZE

if d=VC and a=S then a=S

if d=SC and a=S then a=SS

if d=SF and a=S then a=SF

if d=VF and a=S then a=F

if d=ZE and a=SS then a=ZE

if d=VC and a=SS then a=S

if d=SC and a=SS then a=SS

if d=SF and a=SS then a=SF

if d=VF and a=SS then a=F

if d=ZE and a=SF then a=ZE

if d=VC and a=SF then a=ZE

if d=SC and a=SF then a=S

if d=SF and a=SF then a=SF

if d=VF and a=SF then a=F

if d=ZE and a=F then a=ZE

if d=VC and a=F then a=ZE

if d=SC and a=F then a=S

if d=SF and a=F then a=SF

if d=VF and a=F then a=F

α Rules:

if θ=VL and α=L then α=L

if θ=SL and α=L then α=SL

if θ=ZE and α=L then α=ZE

if θ=SR and α=L then α=R

if θ=VR and α=L then α=R

if θ=VL and α=SL then α=L

if θ=SL and α=SL then α=SL

if θ=ZE and α=SL then α=ZE

if θ=SR and α=SL then α=SR

if θ=VR and α=SL then α=R

if θ=VL and α=ZE then α=L

if θ=SL and α=ZE then α=SL

if θ=ZE and α=ZE then α=ZE

if θ=SR and α=ZE then α=SR

if θ=VR and α=ZE then α=R

if θ=VL and α=SR then α=L

if θ=SL and α=SR then α=SL

if θ=ZE and α=SR then α=ZE

if θ=SR and α=SR then α=SR

if θ=VR and α=SR then α=R

if θ=VL and α=R then α=L

if θ=SL and α=R then α=L

if θ=ZE and α=R then α=ZE

if θ=SR and α=R then α=SR

if θ=VR and α=R then α=R

Defuzzifier Design Process

Lukasiewicz t-norm:

Now with membership functions, fuzzy sets and fuzzy rules defined, fuzzy values can be defuzzified into real values. This involves a methods known as Lukasiewicz t-norms. These t-norms give the membership value of a combination of membership functions and are defined as such.

Lukasiewicz t-norm: max(min(membership functions), 0)

This t-norm method can be used with the FIE matrices in order to determine the membership of each output cell as a weight. The matrix output cells have the corresponding weights.

	W_1	W_2	W_3	W_4	W_5
	W_6	W_7	W_8	W_9	W_{10}
	W_{11}	W_{12}	W_{13}	W_{14}	W_{15}
	W_{16}	W_{17}	W_{18}	W_{19}	W_{20}
	W_{21}	W_{22}	W_{23}	W_{24}	W_{25}

For example:

$W_1(a) = \max(\min(d\mu_{ZE}(x), a\mu_{ZE}(x)), 0)$ based on the FIE matrix

Sugeno s-norm:

Then using the results of the t-norms as weights, Sugeno s-norms can be used to get a real value from the fuzzy memberships. These s-norms are defined as such.

Sugeno s-norm: Σ (weights * output lvl) / Σ (weights)

where output levels are typical values corresponding to a membership of 1 for the respective weight.

System Results

The Matlab script corresponding to this project has an issue with looping and printing. The script takes in the proper input, and outputs membership functions, but due to the looping issue, output is only displayed for the initial timestamp and is missing forward force and angular velocity. New forward acceleration and new angular acceleration are also printed.

The following input cases were used:

Case 1: current location = (0,0), destination = (100,100), timestamps=.1s

Case 2: current location = (0,0), destination = (-50,50), timestamps=.1s

Case 3: current location = (0,0), destination = (0,5), timestamps=.1s

The system results are attached to this project report in pdf format.

Ethical Concerns

With any real world control system there are a variety of ethical concerns. For example, if this robot was out in the real world in an uncontrolled environment, then it could be very dangerous. It could collide with obstacles in it's path which can include people, animals, and property. To avoid this ethical concern, obstacle avoidance could be implemented into the control system. This could be done in a very similar manner to movement, following the fuzzy design process.