UFSC – CTC – INE
INE5430 – Inteligência Artificial

FUZZY TRUCK DRIVER

Lucas Finger Roman - 13103523
Luiz Gustavo Lorgus Decker - 12100759
Renato Matta Machado Pereira da Silva - 14100869

Intrudução

A lógica difusa é a forma de lógica multivalorada na qual os valores lógicos das variáveis podem ser qualquer número real entre 0 (FALSO) e 1 (VERDADEIRO). Assim, uma pertinência de 0.5 pode representar meio verdade, logo 0.9 e 0.1, representam quase verdade e quase falso, respectivamente. Diferentemente, na lógica booleana, os valores lógicos das variáveis podem ser apenas 0 e 1. A lógica difusa foi estendida para lidar com o conceito de verdade parcial, onde o valor verdade pode compreender entre completamente verdadeiro e completamente falso. Além disso, quando variáveis linguísticas são usadas, esses graus podem ser manipulados por funções especificas.

Diferente da Lógica Booleana que admite apenas valores booleanos, ou seja, verdadeiro ou falso, a lógica difusa ou fuzzy, trata de valores que variam entre 0 e 1. Assim, uma pertinência de 0.5 pode representar meio verdade, logo 0.9 e 0.1, representam quase verdade e quase falso, respectivamente

O termo lógica difusa foi introduzido em 1965 com a proposta da teoria de conjuntos difusos por Lotfi A. Zadeh. A lógica difusa tem sido aplicada em várias áreas, desde a teoria do controle à inteligência artificial. A lógica difusa tem sido no entanto estudada desde meados da década de 1920, como lógica infinito-valorada, por Łukasiewicz e Tarski.

As implementações da lógica difusa permitem que estados indeterminados possam ser tratados por dispositivos de controle. Desse modo, é possível avaliar conceitos não-quantificáveis. Casos práticos: avaliar a temperatura (quente, morno, médio, etc.), o sentimento de felicidade (radiante, feliz, apático, triste, etc.), a veracidade de um argumento (corretíssimo, correto, contra-argumentativo, incoerente, falso, totalmente errôneo, etc.)

Um exemplo disso era considerar o período meia-idade que começa em 35 anos e termina em 55 anos.

Utilizando a lógica tradicional, uma pessoa com 34 anos só iria pertencer a esse grupo após completar seu 35º aniversário. Desse modo uma pessoa que tenha 56 anos não faria parte de tal grupo. A figura 1 mostra a definição de meia idade segundo a teoria de conjuntos convencional

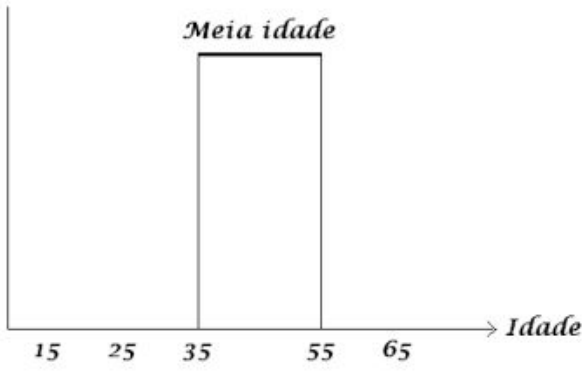

Figura 1 - Definição de meia idade em conjuntos convencionais

Já na figura 2, é apresentada a definição de meia idade segundo a teoria fuzzy. Nota-se que o grau de pertinência que uma pessoa de 25 anos pertença a tal grupo é muito menor em relação a uma pessoa de 45 anos

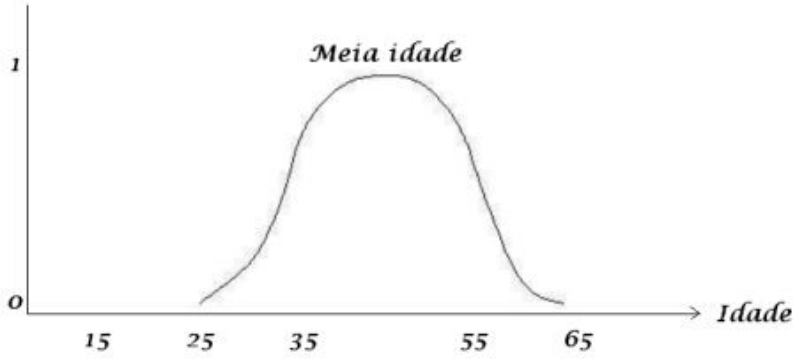

Figura 2 - definição de meia idade em conjuntos fuzzy -

Ao trabalhar com a lógica difusa é comum chamar a lógica booleana de lógica nítida.

Muitos pesquisadores de versões booleanas de lógica não aceitam a lógica difusa como uma verdadeira lógica, no sentido em que aceitam, por exemplo, a lógica modal. Isso pode ser associado a diferentes fatos, entre eles o fato de muitos modelos permitirem soluções aproximadas que não correspondem a uma "verdade" lógica.

Desenvolvimento

Para desenvolver este trabalho, utilizamos a biblioteca JFuzzyLogic. Um objeto desta linguagem foi incluído na área de controle do caminhão, recebendo a posição do caminhão na tela (x,y) e o ângulo que este caminhão está olhando. Estas variáveis alimentam o sistema fuzzy implementado pela biblioteca, são fuzzificadas de acordo com as classes fuzzy estipuladas no arquivo de configuração da biblioteca (um arquivo Fuzzy Controller Logic), e estas variáveis fuzzificadas são então defuzzificadas em classes que representam um esterçamento do volante do caminhão (de -1/30º
a 1/30º), e os pessos dessas classes defuzzificadas são setadas de acordo com as regras definidas neste mesmo arquivo de configuração.

Código Fonte

O unico código fonte desenvolvido foi o arquivo Fuzzy Control Logic, utilizado pela biblioteca. O sistema fuzzy gerado pelas regras deste arquivo já retornam uma resposta para ser enviada ao servidor.

```
//Voce define um FUNCTION_BLOCK (pode ter mais que um em cada arquivo)
FUNCTION_BLOCK caminhao
//agora, as variaveis de input e output são definidas
//Só tem variavel REAL
VAR_INPUT
```

```
   posicaoX : REAL;// Variaveis lidas do ambiente
   posicaoY : REAL;
   anguloOlhando : REAL;
END_VAR

VAR_OUTPUT
   volante : REAL;//Variaveis de controle enviadas para o ambiente
END_VAR
//A maneira em que cada variavel é fuzzificada é definida no bloco FUZZIFY.
//Em cada bloco definimos um ou mais termos. Cada termo é composto de um nome
//e uma função de pertinência

//Cada valor é um ponto (x,y) em que esta funcao será definida.
//No eixo x temos o valor da variavel, e no eixo y o grau de pertinencia [0,1]

FUZZIFY posicaoX
   TERM muitoEsquerda := (0,1) (0.3,1) (0.31,0);
   TERM aEsquerda := (0.25,0) (0.35,1) (0.5,0);
   TERM centro := (0.3,0) (0.45,1) (0.55,1) (0.7,0);
   TERM aDireita := (0.5,0) (0.65,1) (0.75,0);
   TERM muitoDireita := (0.5,0) (0.69,0) (0.7,1) (1,1);
END_FUZZIFY


FUZZIFY posicaoY
   TERM acima := (0,1) (1,0);
   TERM abaixo := (0,0) (1,1);
END_FUZZIFY

FUZZIFY anguloOlhando
   TERM norte := (0,0) (180,0) (270,1) (360,0);
   TERM sul := (0,0) (90,1) (180,0) (360,0);
   TERM leste := (0,0)(90,0)(180,1) (270,0);
   TERM oeste := (0,1)(90,0)(270,0)(360,1);
END_FUZZIFY

//  As variaveis de saida são desfuzzificadas para um valor de saida REAL.
//Isto é definido em um bloco DEFUZZIFY. Como o FUZZIFY, TERMs são definidos

DEFUZZIFY volante
   TERM muitoDireita := (-1,1) (-0.51,1) (-0.5,0);
   TERM direita := (-0.5,0) (-0.25,1)(0,0);
   TERM centro := (-0.05,0) (0,1) (0.05,0);
   TERM esquerda := (0,0) (0.25,1) (0.5,0);
   TERM muitoEsquerda := (0.5,0) (0.51,1) (1,1);
```

//"Center of Gravity" sera o metodo de defuzificação
METHOD : COG;
//Valor defaul (caso ninguém atue)
DEFAULT := 0;
END_DEFUZZIFY
//Agora definimos as regras. Fazemos isso usando um RULEBLOCK.

RULEBLOCK No1
//Primeiro definimos os parametros
//Usar MIN para AND (E MAX para OR, seguindo a lei de DeMorgan)
AND : MIN;
//Usar MIN como método de ativacao
ACT : MIN;
//Usar MAX como método de acumulação
ACCU : MAX;

//Agora definimos as regras
//Regras para quando estiver a esquerda do centro mas nao muito
RULE 1: IF posicaoX IS aEsquerda AND anguloOlhando IS leste THEN volante IS direita;

RULE 2: IF posicaoX IS aEsquerda AND anguloOlhando IS norte THEN volante IS muitoDireita;

RULE 3: IF posicaoX IS aEsquerda AND anguloOlhando IS oeste THEN volante IS muitoEsquerda;

RULE 4: IF posicaoX IS aEsquerda AND anguloOlhando IS sul THEN volante IS esquerda;
//Regras para quando estiver a direita do centro mas nao muito

RULE 5: IF posicaoX IS aDireita AND anguloOlhando IS leste THEN volante IS muitoDireita;

RULE 6: IF posicaoX IS aDireita AND anguloOlhando IS norte THEN volante IS muitoEsquerda;

RULE 7: IF posicaoX IS aDireita AND anguloOlhando IS oeste THEN volante IS esquerda;

RULE 8: IF posicaoX IS aDireita AND anguloOlhando IS sul THEN volante IS direita;

//Regra para quando estiver muito a esquerda do centro
RULE 9: IF posicaoX IS muitoEsquerda AND anguloOlhando IS leste THEN volante IS muitoDireita;

RULE 10: IF posicaoX IS muitoEsquerda AND anguloOlhando IS norte THEN volante IS muitoDireita;

RULE 11: IF posicaoX IS muitoEsquerda AND anguloOlhando IS oeste THEN volante IS muitoEsquerda;

RULE 12: IF posicaoX IS muitoEsquerda AND anguloOlhando IS sul THEN volante IS muitoEsquerda;

//Regra para quando estiver muito a direita do centro
RULE 13: IF posicaoX IS muitoDireita AND anguloOlhando IS leste THEN volante IS muitoDireita;

RULE 14: IF posicaoX IS muitoDireita AND anguloOlhando IS norte THEN volante IS muitoEsquerda;

RULE 15: IF posicaoX IS muitoDireita AND anguloOlhando IS oeste THEN volante IS muitoEsquerda;

RULE 16: IF posicaoX IS muitoDireita AND anguloOlhando IS sul THEN volante IS muitoDireita;
//Regra para quando estiver ao centro
RULE 17: IF posicaoX IS centro AND anguloOlhando IS leste THEN volante IS muitoDireita;

RULE 18: IF posicaoX IS centro AND anguloOlhando IS norte THEN volante IS muitoEsquerda;

RULE 19: IF posicaoX IS centro AND anguloOlhando IS oeste THEN volante IS muitoEsquerda;

RULE 20: IF posicaoX IS centro AND anguloOlhando IS sul THEN volante IS centro;
END_RULEBLOCK
END_FUNCTION_BLOCK

Dificuldades

Como parte do código estava pronto as principais dificuldades foram entender como utilizar a biblioteca JFuzzyLogic. Depois de compreender o seu funcionamento, encontramos outras dificuldades, como definir as forças para os motores e verificar como essas forças agiam no carrinho, além de claro compreender onde seria o output para o motor no código em java.

Achar esse output não foi difícil. Para verificar como as forças agiam no carrinho fomos fazendo testes e a partir deles definindo as forças e suas direções, o que tomou um certo

tempo. A maior dificuldade nisso, foi definir forças que levassem o carrinho até o local onde deve estacionar, independente do ângulo inicial e de sua posição.

Referências

https://en.wikipedia.org/wiki/Fuzzy_logic
https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_fuzzy_logic_systems.htm
http://www.logicafuzzy.com.br/wp-content/uploads/2013/04/uma_introducao_a_logica_fuzzy.pdf