Universidade Federal de Santa Catarina
Departamento de Informática e Estatística
Curso de Ciências da Computação
INE5430 - Inteligência Artificial
Professor Dr.: Mauro Roisenberg

Relatório do Trabalho V

Autor: Décio Moritz Júnior

Florianópolis
16 de junho de 2015

O sistema Fuzzy criado

O sistema desenvolvido foi baseado numa aplicação já pronta, consistindo de um servidor que implementa dois modelos de carrinho, que ao receber comandos via socket, efetua os movimentos dos carrinhos. Ao final dos movimentos, avaliar a "qualidade" com que os carrinhos foram estacionados e calcular pontos a fim de determinar o melhor motorista.

O papel do aluno consistiu em criar um arquivo .fcl (Fuzzy Control Language) com as entradas e saídas corretas, determinar os conjuntos fuzzy(fuzzyficar as entradas), as regras utilizadas e a defuzzificação da variável de saída, a fim de estacionar o carrinho corretamente. O arquivo .fcl criado está anexado ao final deste relatório.

O sistema fuzzy utilizado conta com 3 entradas:

- x: real, variando de 0 a 1, representa a posição do caminhão no eixo x
- y: real,variando de 0 a 1, representa a posição do caminhão no eixo y
- actualAngle: real,variando de 0 a 360, representa o angulo em que o volante de encontra

Em linguagem .fcl:

```
VAR_INPUT
        x : REAL;
        y : REAL;
        actualAngle : REAL;
END_VAR
```

Conta com 1 saída:

- turnAmount: real, variando de -1 a 1, representa o tanto que o volante deverá ser virado e em que sentido no proximo passo.

Em linguagem .fcl

```
VAR_OUTPUT
        turnAmount : REAL;
END_VAR
```

Os conjuntos fuzzy criados são:

- Para variável x:

```
FUZZIFY x
        TERM left := (-0.5, 0) (-0.25, 1) (0.25, 1) (0.5,0);
        TERM middle := (0.25, 0) (0.45, 1) (0.55, 1) (0.75,0);
        TERM right := (0.5,0) (0.75,1) (1.25,1) (1.5,0);
END_FUZZIFY
```

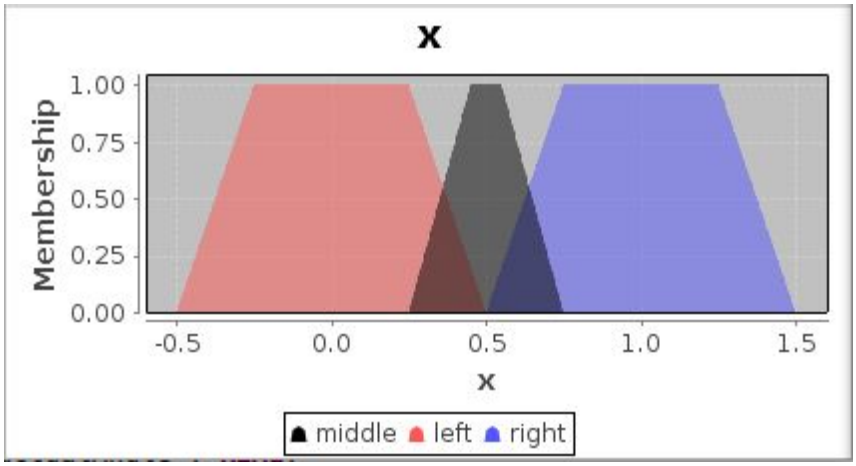


Gráfico da variável x, com os conjuntos fuzzy identificados pelas cores e a legenda. Pode-se observar que foi utilizada uma função trapezoidal. Aqui é interessante observar que o intervalo mostrado vai de -0.5 até 1.5. Como foi utilizado o método de defuzzificação do centro de gravidade (que será mencionado posteriormente) , para tratar problemas inerentes deste método o intervalo foi acrescido de 0.5 unidade para cada extremo.

- Para variável y

```
FUZZIFY y
        TERM tooSouth := (0.5,0) (0.8,1) (1,1) (1.5,0);
        TERM notTooSouth := (0,1) (0.8,0);
END_FUZZIFY
```

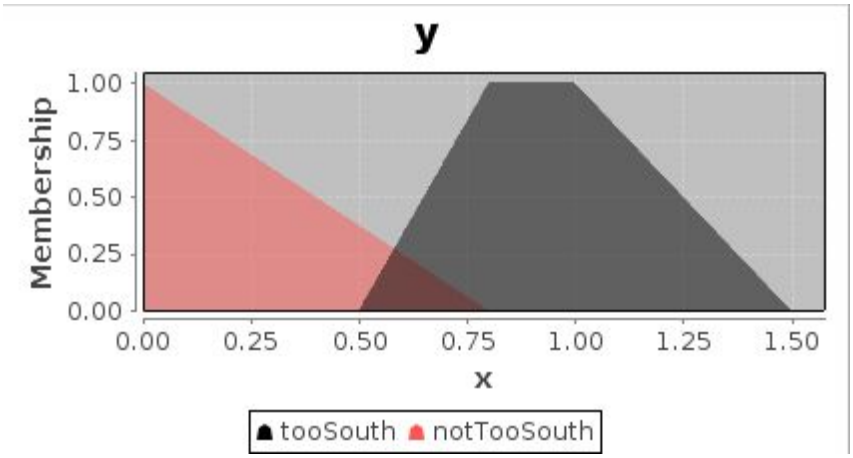


Gráfico da variável y, com os conjuntos fuzzy identificados pelas cores e a legenda. Pode-se observar que foi utilizada uma função trapezoidal para o conjunto tooSouth e triangular para notTooSouth. A razão disso é dar ênfase à parte do problema onde o carrinho encontra-se muito ao sul do mapa, onde recebe um tratamento diferente: em vez de virar o volante em direção à vaga, vira ao contrário, para distanciar-se da vaga e criar um ângulo que possibilite estacionar corretamente.

Aqui também o intervalo foi acrescido de 0.5 unidade à direita.

- Para variável actualAngle

```
FUZZIFY actualAngle
        TERM south_west := (180, 0) (225, 1) (270, 0);
        TERM west := (135, 0) (180, 1) (225, 0);
        TERM north_west := (90, 0) (135, 1) (180, 0);
        TERM south := (225, 0) (245, 1) (295, 1) (325, 0);
        TERM south_east :=  (0, 0) (270, 0) (315, 1) (360, 0);
        TERM east := (0, 1) (45, 0) (325, 0) (360, 1);
        TERM north_east:= (0, 0) (45, 1) (90, 0) (360, 0);
        TERM north := (45, 0) (90, 1) (135, 0);
END_FUZZIFY
```

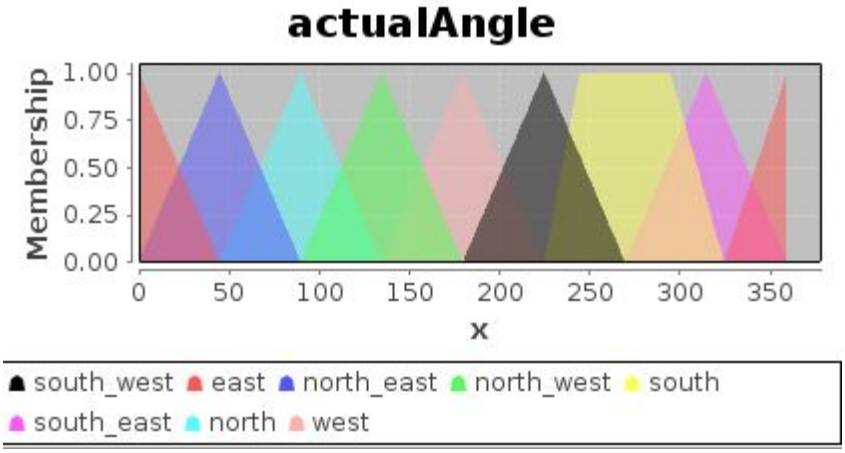


Gráfico da variável actualAngle. Aqui não há nada muito interessante para comentar. São apenas os sentidos básicos da rosa dos ventos definidos com funções triangulares. Com exceção do sentido sul, em que foi usado uma função trapezoidal maior que as outras.

A razão disso é : como a vaga é um pouco maior que o caminhão, dá-se uma “folga” maior e faz o caminhão ser mais propenso a se deslocar em direção à vaga, numa tentativa de melhorar a solução.

Regras:

Seguem as regras definidas em linguagem FCL:

RULEBLOCK Block1
AND : MIN;
ACT : MIN;
ACCU : MAX;
OR : MAX;

RULE 1 : IF x IS left AND actualAngle IS north THEN turnAmount IS right;

RULE 2 : IF x IS left AND actualAngle IS north_east THEN turnAmount IS right;

RULE 3 : IF x IS left AND actualAngle IS north_west THEN turnAmount IS center;

RULE 4 : IF x IS left AND actualAngle IS west AND y IS notTooSouth THEN turnAmount IS center;

RULE 5 : IF x IS left AND actualAngle IS east AND y IS notTooSouth THEN turnAmount IS right;

RULE 6 : IF x IS left AND actualAngle IS south THEN turnAmount IS left;

RULE 7 : IF x IS middle AND actualAngle IS north THEN turnAmount IS center;

RULE 8 : IF x IS middle AND actualAngle IS north_west THEN turnAmount IS left;

RULE 9 : IF x IS middle AND actualAngle IS north_east THEN turnAmount IS right;

RULE 10 : IF x IS middle AND actualAngle IS west THEN turnAmount IS left;

RULE 11 : IF x IS middle AND actualAngle IS east THEN turnAmount IS right;

RULE 18 : IF x IS middle AND actualAngle IS south THEN turnAmount IS center;

RULE 12 : IF x IS right AND actualAngle IS north THEN turnAmount IS left;

RULE 13 : IF x IS right AND actualAngle IS north_east THEN turnAmount IS center;

RULE 14 : IF x IS right AND actualAngle IS north_west THEN turnAmount IS left;

RULE 15 : IF x IS right AND actualAngle IS west AND y IS notTooSouth THEN turnAmount IS right;

RULE 16 : IF x IS right AND actualAngle IS east AND y IS notTooSouth THEN turnAmount IS right;

RULE 17 : IF x IS right AND actualAngle IS south THEN turnAmount IS right;

RULE 19 : IF y IS tooSouth AND actualAngle IS east THEN turnAmount IS left;

RULE 20 : IF y IS tooSouth AND actualAngle IS west THEN turnAmount IS right;

END_RULEBLOCK

Essas regras podem ser divididas em 4 blocos:

- Tratamento com o caminhão à direita do mapa
- Tratamento com o caminhão à esquerda do mapa
- Tratamento com o caminhão no centro do mapa

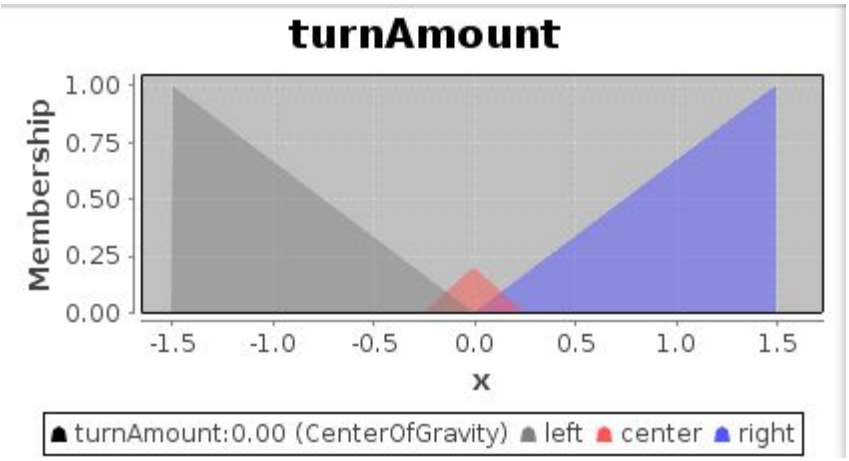
Nos três casos acima são definidas as ações esperadas para quando se encontra nesta posição e nos sentidos: norte,nordeste,noroeste,leste,oeste e sul. A ideia básica é: quando “mais de costas” para a vaga dar ré em linha reta, e quanto “mais de frente“ virar para alinhar-se à vaga.

- Tratamento com o caminhão muito ao sul do mapa

Aqui é dado um tratamento especial. Quando o caminhão encontra-se muito ao sul, a estratégia é se afastar do sul para liberar espaço e ângulo para manobrar. Por isso, caso esteja virado para leste, vira volante à esquerda, caso virado a oeste, vira à direita.

Defuzzificação:

```
DEFUZZIFY turnAmount
        TERM left := (-1.5, 0) (-1.5, 1) (0, 0);
        TERM right := (0, 0) (1.5, 1) (1.5, 0);
        TERM center := (-0.25, 0) (0, 0.2) (0.25, 0);
        METHOD : COG;
        DEFAULT := 0.1;
END_DEFUZZIFY
```



Como pode-se observar no arquivo fcl, foi utilizado o método do centro de gravidade(COG). Os conjuntos definidos podem ser vistos no gráfico. Foram usadas funções triangulares desta forma numa tentativa de suavizar, procurando evitar virar o volante demais.

Arquivo .fcl na íntegra:

```
FUNCTION_BLOCK truck

VAR_INPUT
	x : REAL;
	y : REAL;
	actualAngle : REAL;
END_VAR

VAR_OUTPUT
	turnAmount : REAL;
END_VAR

FUZZIFY x
	TERM left := (-0.5, 0) (-0.25, 1) (0.25, 1) (0.5,0);
	TERM middle := (0.25, 0) (0.45, 1) (0.55, 1) (0.75,0);
	TERM right := (0.5,0) (0.75,1) (1.25,1) (1.5,0);
END_FUZZIFY

FUZZIFY y
```

```
        TERM tooSouth := (0.5,0) (0.8,1) (1,1) (1.5,0);
        TERM notTooSouth := (0,1) (0.8,0);
END_FUZZIFY

FUZZIFY actualAngle
   TERM south_west := (180, 0) (225, 1) (270, 0);
   TERM west := (135, 0) (180, 1) (225, 0);
   TERM north_west := (90, 0) (135, 1) (180, 0);
   TERM south := (225, 0) (245, 1) (295, 1) (325, 0);
   TERM south_east :=  (0, 0) (270, 0) (315, 1) (360, 0);
   TERM east := (0, 1) (45, 0) (325, 0) (360, 1);
   TERM north_east:= (0, 0) (45, 1) (90, 0) (360, 0);
   TERM north := (45, 0) (90, 1) (135, 0);
END_FUZZIFY

DEFUZZIFY turnAmount
        TERM left := (-1.5, 0) (-1.5, 1) (0, 0);
        TERM right := (0, 0) (1.5, 1) (1.5, 0);
        TERM center := (-0.25, 0) (0, 0.2) (0.25, 0);
        METHOD : COG;
        DEFAULT := 0.1;
END_DEFUZZIFY

RULEBLOCK Block1
        AND : MIN;
        ACT : MIN;
        ACCU : MAX;
        OR : MAX;

        RULE 1 : IF x IS left AND actualAngle IS north THEN
turnAmount IS right;

        RULE 2 : IF x IS left AND actualAngle IS north_east THEN
turnAmount IS right;
```

RULE 3 : IF x IS left AND actualAngle IS north_west THEN turnAmount IS center;

RULE 4 : IF x IS left AND actualAngle IS west AND y IS notTooSouth THEN turnAmount IS center;

RULE 5 : IF x IS left AND actualAngle IS east AND y IS notTooSouth THEN turnAmount IS right;

RULE 6 : IF x IS left AND actualAngle IS south THEN turnAmount IS left;

RULE 7 : IF x IS middle AND actualAngle IS north THEN turnAmount IS center;

RULE 8 : IF x IS middle AND actualAngle IS north_west THEN turnAmount IS left;

RULE 9 : IF x IS middle AND actualAngle IS north_east THEN turnAmount IS right;

RULE 10 : IF x IS middle AND actualAngle IS west THEN turnAmount IS left;

RULE 11 : IF x IS middle AND actualAngle IS east THEN turnAmount IS right;

RULE 18 : IF x IS middle AND actualAngle IS south THEN turnAmount IS center;

RULE 12 : IF x IS right AND actualAngle IS north THEN turnAmount IS left;

RULE 13 : IF x IS right AND actualAngle IS north_east THEN turnAmount IS center;

RULE 14 : IF x IS right AND actualAngle IS north_west THEN turnAmount IS left;

RULE 15 : IF x IS right AND actualAngle IS west AND y IS notTooSouth THEN turnAmount IS right;

RULE 16 : IF x IS right AND actualAngle IS east AND y IS notTooSouth THEN turnAmount IS right;

RULE 17 : IF x IS right AND actualAngle IS south THEN turnAmount IS right;

RULE 19 : IF y IS tooSouth AND actualAngle IS east THEN turnAmount IS left;

RULE 20 : IF y IS tooSouth AND actualAngle IS west THEN turnAmount IS right;

END_RULEBLOCK

END_FUNCTION_BLOCK