6 de Dezembro 2013

CONTROLE AUTÔNOMO DE ROBÔ MÓVEL BASEADO EM LÓGICA FUZZY

Daniel de Sousa Leite

CONTROLE AUTÔNOMO DE ROBÔ MÓVEL BASEADO EM LÓGICA FUZZY

Aluno: Daniel de Sousa Leite

Orientadora: Karla Tereza Figueiredo Leite

Trabalho apresentado com requisito parcial à conclusão do curso de Engenharia de Controle e Automação na Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

PUC RIO

Agradecimentos

Agradeço primeiramente a Deus, pelas oportunidades que me foram concedidas.

Aos meus familiares por sempre me darem apoio.

A professora Karla Tereza Figueiredo Leite, pela orientação e a poio, na realização desse trabalho e de outros, como a minha iniciação científica.

Aos meus colegas de trabalho, que sempre se mostraram dispostos a trocas de horário em épocas de prova.

E aos amigos que fiz durante o curso, que sempre me ajudaram.

Resumo

Esse trabalho a apresenta a implementação do controle de um robô móvel baseado em Lógica Fuzzy. O robô foi configurado com sensores tipo ultrassom, GPS e bússola e a partir destes, deverá "perceber" o ambiente no qual está inserido e, por intermédio do controle Fuzzy, conjugar um comportamento reativo com um deliberativo para que ele possa executar uma rota, desviando-se de possíveis obstáculos, dinâmicos e estáticos, que venha a encontrar, independente do ambiente que estiver navegando. Devido à localização feita por GPS, o ambiente proposto para navegação deve ser externo.

O projeto foi desenvolvido em ambiente simulado por intermédio do simulador de robôs STAGE em conjunto com a interface Player e os códigos fuzzy serão escritos em C++.

Palavras-chave: robô móvel; Lógica Fuzzy; controle.

AUTONOMOUS CONTROL OF MOBILE ROBOT BASED ON FUZZY LOGIC

Abstract

This work presents the implementation of the control of a mobile robot based on Fuzzy Logic. The robot was configured with ultrasound sensors, GPS and compass type and from these, should "realize" the environment in which it is inserted and through the Fuzzy control, conjugating a reactive behavior with a deliberative so it can run a route, bypassing possible obstacles, dynamic and static, it will find, regardless of the environment it is browsing. Because of the location taken by GPS, the environment proposed for navigation should be external.

The project was developed in simulated environment through the robot simulator STAGE together with the Player interface and fuzzy codes are written in C++.

Keywords: mobile robot; Fuzzy Logic; control

Sumário

Lista de Figuras

Introdução

Com o avanço da robótica surgiram diversos tipos de robôs com formas, funções e ambientes de atuação distintos. Essa diversidade dificulta uma definição universalmente aceita do que é o robô. Contudo, segundo a ROBOTIC INDUSTRIES ASSOCIATION [1] o robô é "um manipulador programável multi-função projetado para mover materiais, peças ou dispositivos especializados através de movimentos variados programados para o desempenho de uma variedade de tarefas". Além dessa definição formal, os robôs são classificados em dois grupos principais:

- Robô Manipulador: Sistema semelhante a um "braço mecânico" com o objetivo de manipular e posicionar peças ou ferramentas. Ele possui dois ou mais graus de liberdade controláveis por computador, combina a tecnologia de computação com a de servo-controle de cadeias articuladas, é reprogramável para executar uma variedade de tarefas e possui sensores que permitam reagir e se adaptar a diferentes condições [2].

– Robô móvel: Dispositivos robóticos dotados de um sistema de locomoção capazes de navegar através de um determinado ambiente de trabalho e com certo nível de autonomia para sua locomoção [2].

Atualmente as aplicações em robótica são as mais variadas possíveis, sendo os robôs do tipo manipulador largamente usados na indústria em atividades como inserção de peças, soldagem e pintura. Já os robôs móveis, além das aplicações industriais como plataformas para movimentação de carga, possui grande atuação na área de petróleo, para inspeção de dutos e atividades relacionadas a lançamento de linhas. Outra aplicação da robótica móvel que vem se popularizando é no auxílio nas tarefas domésticas com robôs que são capazes de exercer atividades como limpeza de piscinas, chão e cortar grama de maneira autônoma.

Esse trabalho de conclusão de curso teve por objetivo desenvolver um Sistema de Controle Fuzzy para robô móvel em ambiente virtual, cuja aplicação será o auxílio de locomoção para pessoas idosas. Esse trabalho consiste na etapa inicial do projeto "Bengala Robótica" sobre edital FAPERJ – N^0 06/2013. Esse robô móvel será a base da bengala que irá guiar o idoso ao seu objetivo, desviando de situações de risco que venha a encontrar em seu caminho.

A partir dessa motivação, foi criado um robô em ambiente virtual, controlado por modelo baseado em técnica inteligente, de forma a navegar em ambiente externo, partindo de um determinado local e tendo um ponto identificado como destino, desviando de obstáculos fixos ou móveis. Esse desenvolvimento foi realizado por intermédio das ferramentas PlayerStage, simulador *opensource* para robôs, que permite uma interface com o usuário nas linguagens de programação C, C++ ou Phyton. Para avaliar o robô, foi construído um "circuito de provas" que simulou um ambiente urbano com obstáculos comuns que possam ser encontrados em uma calçada, tais como pedras, postes, bancos e até outras pessoas. Nessas condições, o robô deverá ser capaz de desviar para evitar colisões, mas sem se desfocar do objetivo final, que é chegar á uma coordenada desejada no ambiente. Para tal, serão utilizados sensores de distância, para detecção de obstáculos, GPS e bússola, para localizar o robô em relação ao seu objetivo. Todo processo de evolução do robô no ambiente irá ocorrer sem que ele tenha um conhecimento prévio sobre o local onde está inserido, como um mapa, por exemplo.

1. Bengala Robótica

O projeto Bengala Robótica visa à criação de um dispositivo robótico que irá auxiliar o idoso no seu deslocamento em centros urbanos. Ele deverá ser capaz de dar apoio à sustentação do idoso e de guia-lo baseado na coordenada GPS do objetivo. Essas coordenadas serão enviadas por um aplicativo de celular que irá extrair as rotas do Google Maps@ para percursos a pé, figura 1.

A base da bengala robótica será um robô móvel que, dotado de sensores, deverá identificar obstáculos que possam levar o idoso a queda e, baseado em sua lógica de controle, realizar os desvios necessários. O robô não terá informações prévias, como mapas, da estrutura do ambiente que irá navegar, mas apenas sensores que irão detectar a presença de obstáculos.

Figura 1 - Figura representativa da atuação da Bengala Robótica

a. Estrutura Física

A equipe desenvolvimento do projeto estabeleceu as características físicas iniciais do robô móvel, base da bengala, como sendo um robô diferencial com uma roda castor para apoio, cujas dimensões podem ser observadas na figura 2.

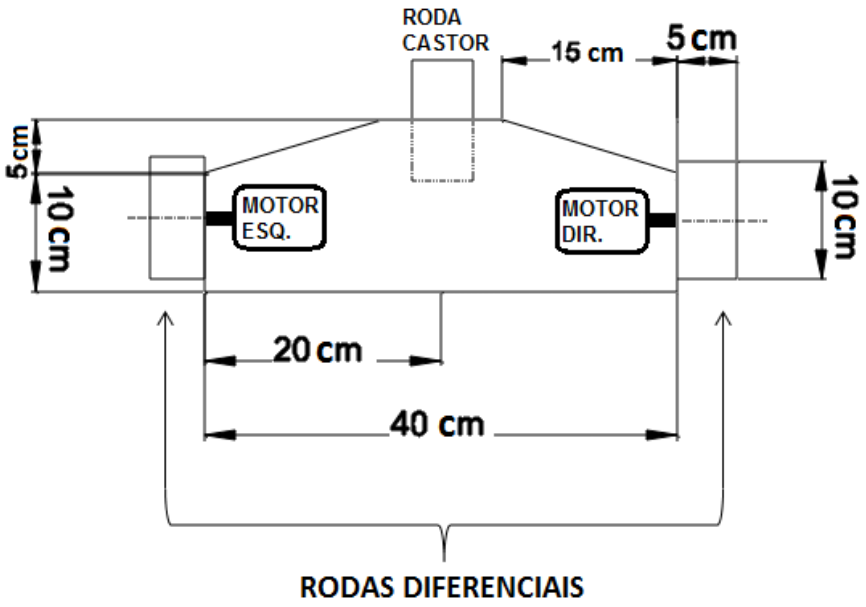

Figura 2 - Características físicas do robô móvel

2. Robô Móvel

a. Sensores e navegação

Segundo Leonard e Durrant-Whyte [3], o problema de robótica móvel pode ser resumido em três perguntas básicas: "Onde estou?", "Aonde vou?" e "Como devo chagar lá?". O grau de dificuldade para responder cada uma dessas perguntas varia de acordo com os tipos de sensores, a morfologia escolhida para o agente e as características do ambiente.

1. Onde estou?

Dependendo da técnica utilizada para medir a localização do robô ela pode ser classificada como posição relativa ou posição absoluta e alguns dos principias métodos presentes nessas duas categorias são:

Medição Relativa

- ✓ **Odometria:** Técnica que utiliza sensores para medir a rotação e/ou o ângulo de orientação das rodas para estimar o deslocamento e a posição do robô [4]. É uma técnica com grande aplicação devido ao baixo custo, mas tem a desvantagem de gerar erros acumulados que crescem indefinidamente, a menos que outra referência independente seja utilizada periodicamente para correção do erro.

- ✓ **Navegação Inercial:** Método que utiliza giroscópios e acelerômetros para medir a taxa de rotação e aceleração que são integradas uma vez (ou duas vezes) para se obter a posição do robô[4].Tem a vantagem de não depender de variáveis externas. Contudo, devido à necessidade de integrar o valor de leitura, qualquer pequeno erro constante aumenta indefinidamente. Sendo, portanto, inadequados para calcular o posicionamento preciso durante um período de tempo prolongado.

Medição Absoluta

- ✓ **Sinalização ativa:** Este método calcula a posição absoluta do robô a partir da medição da incidência direta de três ou mais sinais de transmissão ativos. Esses transmissores podem ser de luz ou radio frequência e suas posições no ambiente devem ser bem definidas. Um exemplo de aplicação dessa técnica é o sensor GPS (*Global Positioning System*) que se utiliza de satélites na órbita terrestre para fazer a triangulação.

- ✓ **Marcação artificial:** Nesse método marcações distintas são colocadas em posições conhecidas do ambiente para que sejam usadas como referência na localização do robô. Assim como no método da sinalização ativa, são necessários três ou mais marcações para que o robô possa ser localizado. Esse método tem a vantagem das marcações poderem ser feitas de maneira a aperfeiçoar seu reconhecimento, contudo é necessário que no mínimo três marcações estejam no "campo de visão" do robô para que a posição possa ser estimada.

- ✓ **Marcação Natural:** Método semelhante à marcação artificial, contudo os marcos são naturais do ambiente. Esse método tem a desvantagem de necessitar conhecer o ambiente muito bem para evitar erros como duplicidades de marcações ou imprecisão na localização dos marcos.

- ✓ **Modelo correspondente:** Nesse método o robô compara os dados das leituras dos seus sensores embarcados com o mapa ou modelo do ambiente que ele carrega. Se houver uma correspondência entre esses dados, a posição do robô pode ser estimada. Os mapas utilizados na navegação incluem dois tipos principais: mapas geométricos e mapas topológicos. Mapas geométricos representam o mundo em um sistema de coordenadas global, enquanto os mapas topológicos representam o mundo como uma rede de nós e arcos. Uma vantagem desse método é que os mapas podem ser atualizados a partir da interação do robô com o ambiente, fazendo com que regiões a principio desconhecidas sejam mapeadas.

2. Aonde vou e como deve chegar lá?

Para que o robô possa adotar um conjunto de ações, é necessário que primeiro ele perceba o ambiente no qual está inserido. Essa percepção é estabelecida por meio de sensores, em geral embarcados, cuja natureza irá depender do tipo de ação e ambiente que se deseja trabalhar. Na robótica móvel, é comum o uso de sensores para detecção de objetos a fim de evitar a colisão do robô e para o reconhecimento de padrões. Alguns dos sensores mais utilizados são:

- ✓ **Sensores Ultrassom:** Sensor no qual uma onda sonora de alta frequência é emitida e a partir do tempo que ela demora a ecoar é determinada a distância ao obstáculo. Tem a vantagem de ser um sensor de baixo custo, possuir uma rápida resposta e poder ser usado em movimento. Contudo, é sensível a ruídos gerados pela utilização de outros sensores semelhantes (*crosstalk*), e é impreciso para determinar a posição do objeto, dentro do raio de atuação da onda sonora.

- ✓ **Sensores Infra-vermelho:** Sensor fotoelétrico no qual um LED infravermelho emiti um sinal luminoso e um foto transistor recebe esse sinal refletido por um objeto. A partir do tempo do deslocamento dessa onda é obtida a distância ao objeto. Tem a vantagem de conseguir medir pequenas distâncias e a desvantagem de ser um facho pontual.

- ✓ **Sensor Laser-scan:** Possui o mesmo princípio de funcionamento do infra-vermelho, contudo a fonte de sinal luminoso está presa a uma base móvel que gira em um eixo perpendicular a sua emissão. Como o ângulo da base é conhecido, é possível especificar, além da distância ao objeto, o tamanho e a posição que se encontra dentro do raio de atuação do laser. Tem como principal desvantagem o preço.

- ✓ **Câmeras:** Com o auxílio de algorítmicos de visão computacional, as câmeras podem ser utilizadas para detecção de diferentes padrões, como formas, tamanhos e cores, além da própria distância do objeto. Esse tipo de técnica tem como desvantagem o custo computacional.

- ✓ **Sensor de luz:** Sensores de luz são sensores que variam propriedades físicas, em geral elétricas, devido à incidência de luz.

- ✓ **Sensor de Cor:** Sensor fotoelétrico composto por três LEDs, vermelho, verde e azul, e um foto transistor. De acordo com a intensidade do sinal luminoso refletido, é calculada a cor do objeto.

Uma vez que o robô esteja localizado no ambiente e que ele consiga extrair informações dele, é necessário estabelecer quais conjuntos de ações ele deve tomar. A estratégia a ser adotada está diretamente relacionada às características do ambiente que foram percebidas. Por exemplo, em um ambiente estruturado no qual o robô possua um mapa ou marcos que ele possa utilizar para obter sua localização global, pode-se estabelecer uma rota ótima a qual o robô deva seguir. Essa rota pode está embarcada do próprio mapa que o robô possui do ambiente, ou por marcações no ambiente, como acontece em um robô seguidor de linhas. Nessa configuração, para que o robô permaneça na rota estabelecida, basta um simples controle PID que irá corrigir qualquer erro que ocorra devido a imprecisões no ambiente ou modelo do robô.

Outras configurações, com ambientes não estruturados, o estabelecimento de uma rota torna-se uma tarefa complexa, pois o agente robótico deverá conjugar um comportamento reativo, para que desvie dos obstáculos que não poderão ser previsto, com um comportamento deliberativo, para que ele siga o objetivo principal. Em situações como essas, técnicas de inteligência computacional vêm se mostrando bem adequadas, pois permitem que o agente ganhe maior flexibilidade para lidar com diferentes obstáculos, estáticos e dinâmicos, que venha a encontrar, sem que seja necessária uma modelagem matemática complexa do ambiente ao qual está inserido.

A técnica emprega nesse trabalho será a Lógica Fuzzy, pois ela consegue imitar a tomada de decisão de um operador humano porque tem capacidade de modelar modos imprecisos e vagos, que são a base para o raciocínio.

3. Lógica Fuzzy

A teoria dos conjuntos fuzzy foi introduzia por L. A. Zadeh em meados dos anos 60 com objetivo fornecer um ferramental matemático para tratamento de dados com caráter impreciso ou vago [5]. Ela surgiu inspirada na lógica tradicional, conduto, diferente da característica binária da lógica clássica, na qual um valor pertence ou não a um conjunto, na Lógica Fuzzy, o valor passa a ter um grau de compatibilidade que varia de 0 à 1. Com isso, uma variável pode pertencer a mais de um conjunto fuzzy com diferentes graus de compatibilidade, mais conhecido como grau de pertinência.

As variáveis cujos valores são nomes de conjuntos fuzzy, recebem o nome de variável linguística [5]. Por exemplo, a velocidade de rotação de um motor em um determinado processo pode ser uma variável linguística assumindo valores como baixo, médio e alto. Esses valores descrevem conjuntos fuzzy representados por funções de pertinência, conforme figura 3.

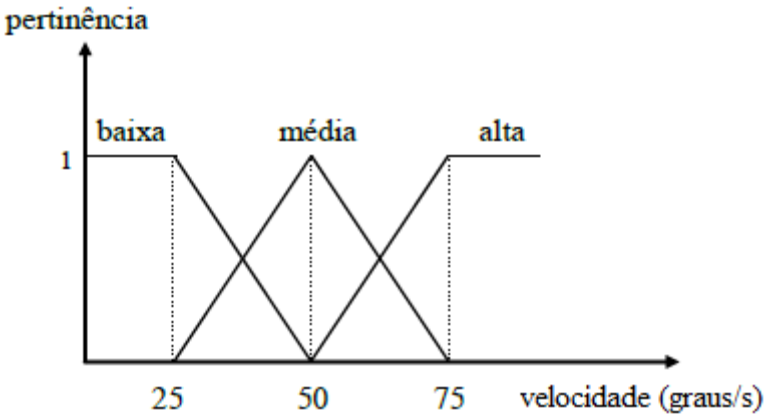

Figura 3 - Funções de pertinência

As funções de pertinência são aquelas que descrevem a distribuição dos valores pertencentes a um dado conjunto fuzzy. Elas podem ter diferentes formas dependendo do conceito e das características do problema que se deseja retratar. Algumas das formas mais utilizadas para a área de controle são triangulares, trapezoidais e gaussianas.

Ainda com respeito às variáveis linguísticas, elas têm como principal objetivo fornecer de maneira sistemática uma caracterização aproximada de fenômenos complexos ou mal definidos [5]. Com isso, o tratamento de problemas por meio dessas variáveis, permite modelar de maneira simples problemas complexos de serem modelados por meio de técnicas matemáticas convencionais. Portanto, a Lógica Fuzzy se mostra extremamente útil em aplicações que visão emular a tomada de decisão de operadores humanos, que são capazes de tomar decisões complexas baseados em informações de caráter impreciso e vago, pois ela consegue tratar os modos imprecisos do raciocínio.

A Lógica Fuzzy possui aplicações em diversas áreas do conhecimento, como previsão de séries, classificação e controle, e elas ocorrem por meio de um sistema de inferência fuzzy, figura 4. O sistema de inferência será mostrado baseado em uma aplicação de controle, pois foi a utilizada nesse projeto.

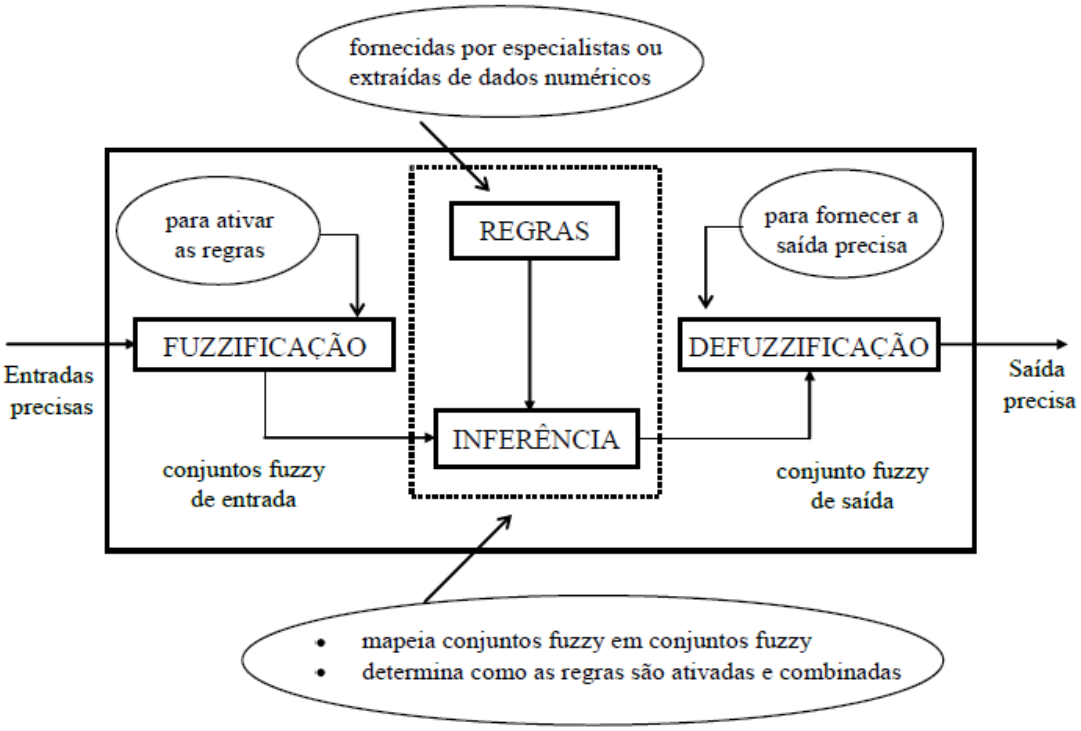

Figura 4 - Sistema de Inferência Fuzzy

Em aplicações relacionadas a controle, as variáveis de entrada são em geral precisas, pois são fornecidas a partir da leitura de sensores. Para que essas variáveis possam ser tratadas no Sistema Fuzzy, elas devem passar por um processo conhecido como fuzzificação.

A fuzzificação consiste em transformar uma variável precisa em uma variável fuzzy. Isso ocorre verificando-se o grau de pertinência do valor preciso aos conjuntos fuzzy que representam as variáveis de entrada. Conjuntos estes, modelados por suas respectivas funções de pertinência, figura 5.

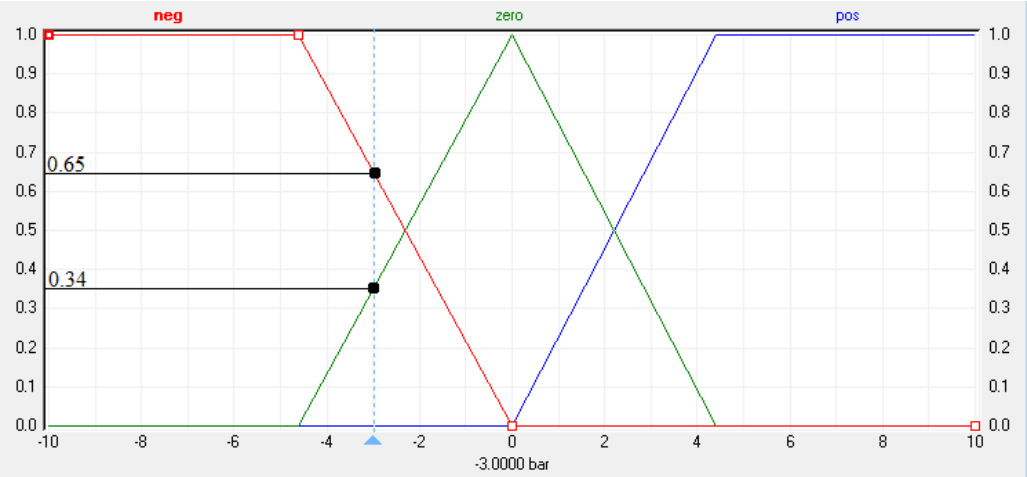

Figura 5 - Exemplo de fuzzificação

No exemplo da figura 5, o valor preciso de pressão -3 bar passa a ter os valores fuzzy ***negativo***, com pertinência de 0.65, e ***zero***, com pertinência de 0.34, pertencentes a variável linguística pressão.

As regras do sistema de inferência fuzzy são fornecidas por um especialista ou extraídas de um conjunto de dados e são do tipo "se ... então". Assim como na lógica clássica, a Lógica Fuzzy possui operados para união, intercessão e negação dos conjuntos que compões as regras.

Os operadores de conjunção utilizados na Lógica Fuzzy são aquelas que se enquadram na norma triangular, sendo os mais utilizados o mínimo e o produto. Para representar a disjunção, utilizam-se operadores de co-norma triangular, sendo o operador máximo o mais utilizado.

A negação é obtida pelo complemento da função de pertinência que representa o valor do conjunto fuzzy. Por exemplo, seja μ(x) a função de pertinência do valor ***médio*** da variável ***temperatura***, o valor ***não médio*** será 1- μ(x), figura 6.

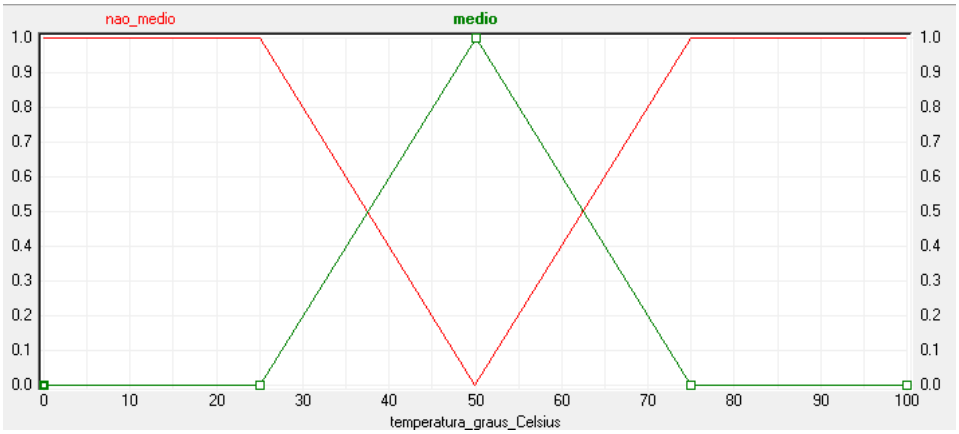

Figura 6 - Negação de conjuntos Fuzzy

A inferência das regras fuzzy, assim como na lógica clássica, é feita por um operador de implicação. Contudo, em fuzzy, esse operador, em geral, é um operador de norma triangular, sendo os mais utilizados o operador de mínimo e o de produto. Com isso, o resultado de um processo de inferência fuzzy é uma saída fuzzy, cujos conjuntos que compõe seus valores serão ativados de acordo com a ativação dos antecedentes de cada regra.

Após a inferência dos dados fuzzy, normalmente necessita-se de uma resposta precisa para ser utilizada como dado de entrada de um atuador, num exemplo de processo de controle. Para que isso ocorra, a saída do processo de inferência deve ser defuzzificada, ou seja, transformada de fuzzy para precisa. Existem diversas técnicas para de defuzzificação, sendo algumas das mais empregadas:

- ✓ **Centroide:** A saída precisa é o valor no universo de discurso da variável de saída fuzzy onde se localiza seu centro de gravidade, figura 7. Esse método tem a característica o fato de que todos os conjuntos ativos das variáveis de saída possam contribuir tanto na forma quando no grau de ativação para o valor final.

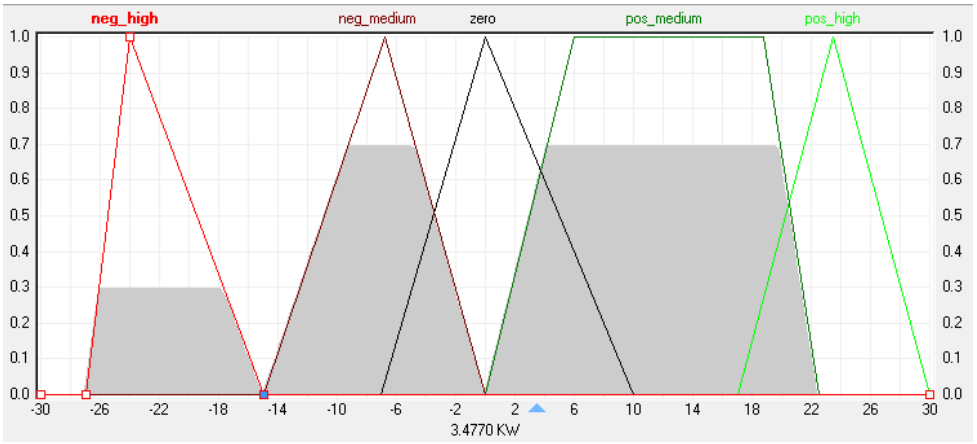

Figura 7 - Exemplo do método do Centróide

- ✓ **Máximo:** A saída precisa é o valor no universo de discurso da variável de saída fuzzy, onde se encontra a altura do conjunto com maior grau de ativação, figura 8. Esse método tem como desvantagem não utilizar informações provenientes de conjuntos com ativações menores.

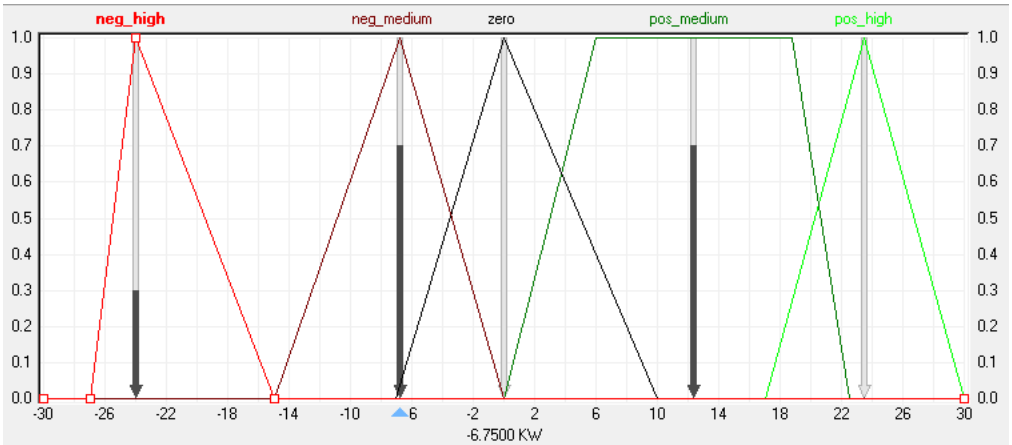

Figura 8 – Exemplo do método do Máximo

- ✓ **Centro dos Máximos:** A saída precisa é o valor no universo de discurso da variável de saída fuzzy, onde se encontra a média ponderada pela pertinência dos valores máximos de ativação de cada conjunto, figura 9. Ela tem a vantagem de todos os conjuntos contribuírem para a resposta final.

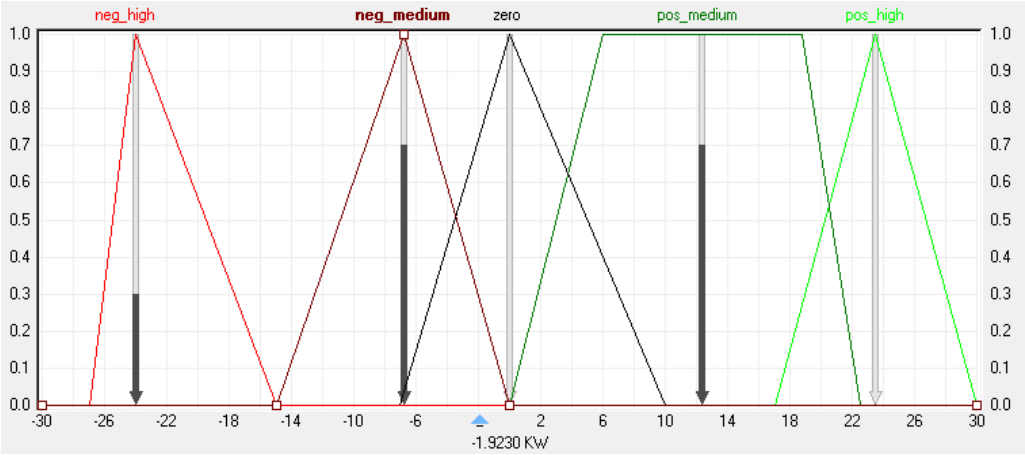

Figura 9 - Exemplo do método Centro dos Máximos

Conforme pôde ser observado nos exemplos acima, o valor da saída precisa varia de acordo com o método escolhido. Contudo, cabe ao especialista avaliar qual método representa melhor a natureza do problema que está sendo avaliado.

4. Ferramentas de Simulação

Para verificar e validar as configurações do robô móvel e de seu controle fuzzy, foi utilizado a ferramenta de simulação Stage [7] em conjunto com a interface Player [6]. Ambos os softwares são *OpenSource*, sendo o Player desenvolvido por Brian Gerkey e o Stage por Richard Vaughan, com contribuições de colaboradores de diversas parte de mundo. Essas ferramentas de simulação foram escolhidas devido a sua grande flexibilidade ao ligar com diversos tipos de robôs, sensores e atuadores, sendo ainda ideal para aplicações que envolvem sistemas multiagentes.

a. Player

O Player é um software que fornece uma camada de abstração de hardware para sistemas robóticos [6]. Ele funciona com uma espécie de sistema operacional para robôs, no qual possui um conjunto de interfaces padrões, onde estão descritas as maneiras como se pode interagir com diferentes classes de dispositivos, definindo a sintaxe e a semântica de todas as mensagens que podem ser trocados com entidades da mesma classe. Dentre as interfaces disponíveis, utilizaram-se as seguintes no projeto:

position2d: Utilizada para controlar a base de robôs móveis em 2D, ela fornece ferramentas para extrair informações como velocidade linear e angular, posição (relativa ou absoluta) e orientação da base do robô. Além de fornecer essas informações, é possível especificar a velocidade angular e linear da base do robô.

ranger: Utilizada para receber dados de sensores de distância, como laser scanner, sonar e infravermelho.

simulator: Utilizados para receber e enviar dados de dispositivos virtuais que estejam rodando em simuladores, como o Stage.

Para adequar as características dos dispositivos aos protocolos padrões fornecidos pelas classes ao qual o Player fornece suporte, é necessária a utilização de um drive. O drive é um pequeno programa (geralmente escrito em C++) que se comunica com um determinado sensor, atuador ou algoritmo e traduz suas entradas e saídas para se adequar com uma ou mais interfaces. O trabalho do driver é esconder os detalhes de qualquer entidade, fazendo com que ela pareça ser a mesmo que qualquer outra na sua classe [6]. Existem diversos drives disponíveis para diferentes dispositivos como, o ***roomba*** (utilizado para o robô aspirador da Irobot Roomba), o ***garminnmea*** (utilizados para GPS da Garmin) e o ***sicklms400*** (utilizados para o SICK LMS 400 laser range-finder). Contudo, como esse projeto foi realizado em ambiente virtual, o drive utilizado foi o ***libstageplugin*** que irá adequar as características do simulador Stage à classe ***Simulator***.

A comunicação entre o programa cliente, programa que irá gerenciar o controle do seu dispositivo, ou seja, o seu código, e o Player é feita através do protocolo TCP [8]. O Player instancia um servidor TCP no qual ele fornece portas de entrada e saída para todos os dispositivos que foram declarados no arquivo de configuração, figura 10. O arquivo de configuração é um arquivo de extensão ".cfg" não qual são especificados os drivers, portas e dispositivos que serão utilizados.

```
world
(
   name              "cidade3.world"
   interval_real     100
   interval_sim      100
   gui_interval      100
   resolution        0.01
)

driver
(
        name "stage"
        plugin "stageplugin"
        provides ["simulation:0"]
        # load the named file into the simulator
        worldfile "cidade3.world"
)

# the main idoso1 robot
driver
(
        name "stage"
        provides ["6665:position2d:0" "6665:ranger:0"]
        model "idoso"
)
```

Figura 10 - Exemplo de arquivo.cfg do Player

O acesso do programa cliente ao servidor Player é feito por funções específicas para classe que se deseja trabalhar. Essas funções estão presentes na biblioteca referente a linguagem de programação do código cliente utilizado. Por exemplo, ***libplayerc***, para cliente em ***C***, ***libplayerc_py***, para clientes em ***Python***, e ***libplayerc++***, para clientes em ***C++***. Nesse projeto em específico, as funções do programa cliente foram criadas em ***C++***, conforme pode ser vista no anexo 2.

Em resumo, a interação do programa cliente, servidor Player e dispositivo pode ser exemplificada na figura 11.

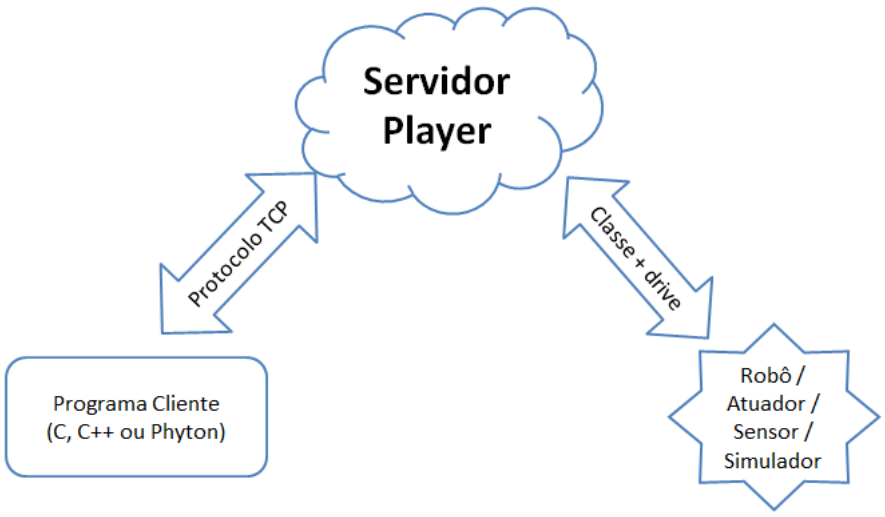

Figura 11 - Digramas da interação do programa Cliente/ Player/ Dispositivo

b. Stage

O Stage é um simulador 2D no qual é possível criar um ambiente virtual com diversos robôs, sensores e objetos que os agentes robóticos poderão detectar e manipular. Existem três maneiras de se utilizar o Stage:

1 – Como um simulador de robôs autônomos, no qual o programa de controle é carregado por uma biblioteca previamente fornecida pelo usuário, não necessitando do Player.

2 – Como um plugin para o Player (libstageplugin), onde os dados da simulação são fornecidos para o Servidor Player que irá fazer a interface com o programa cliente.

3 – Como um simulador personalizado, no qual, por meio da biblioteca **libstage**, o usuário pode rodar alterações do simulador Stage dentro de seu próprio código.

O Stage disponibiliza uma série de módulos de sensores e atuadores, incluindo sonar, laser infravermelho, laser scanner, garras e bases de robôs móveis, figura 12.

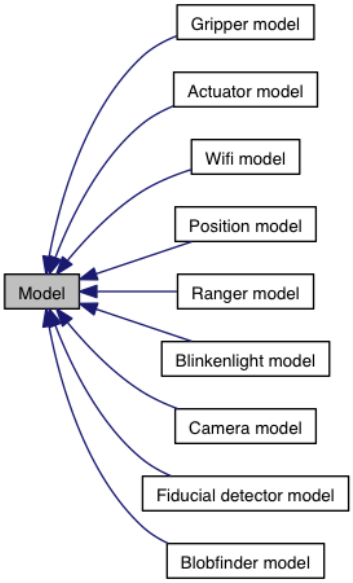

Figura 12 - Digramas dos módulos fornecidos pelo Stage

Os módulos utilizados no projeto foram:

- ✓ **Position**: Módulo que simula a base do robô móvel, podendo ela ser do tipo diferencial, omnidirecional ou quaridiclo. Por meio desse módulo o simulador fornece dados da base do robô, como sua localização global, simulando um sensor tipo GPS, e a localização relativa, simulando a odometria. Também é possível emular erros de leitura provenientes da odometria e fornecer as velocidades, linear e angular, desejadas para a movimentação da base.

- ✓ **Ranger**: Módulo que simula um array leituras de sensores do tipo sonar ou infravermelho. Ele permite editar características como campo de visão e máxima e mínima distância de leitura do objeto, fazendo com que o sensor se torne o mais próximo possível do sensor real desejado.

Todas as características que se desejam no ambiente de simulação como, paredes, robôs, sensores e atuadores, são especificadas num arquivo de extensão ".world", figura 13.

```
position
(
# position properties
drive "diff"

velocity [ 0.0 0.0 0.0 0.0 ]

localization "gps"
localization_origin [ <defaults to model's start pose>

# odometry error model parameters,
# only used if localization is set to "odom"
odom_error [0.03 0.03 0.00 0.05]

# only used if drive is set to "car"
wheelbase 1.0

# [ xmin xmax ymin ymax zmin zmax amin amax ]
velocity_bounds [-1 1 -1 1 -1 1 -90 90 ]
acceleration_bounds [-1 1 -1 1 -1 1 -90 90]

# model properties

)
```

```
ranger
(
# ranger-specific properties

sensor (
pose [ x y z a ]
size [  x y z ]
fov a
range [min max]
)

# generic model properties with non-default values
watts 2.0
color_rgba [ 0 1 0 0.15 ]
)
```

Figura 13 - Exemplo de arquivo. world do Stage para configurar os módulos position e ranger

5. Metodologia

a. Configuração do robô móvel

Para que o robô possa ter a percepção do ambiente no qual está inserido, foram utilizados sensores tipos Ultra Som. Esse modelo de sensor foi escolhido principalmente pelo seu baixo preço, quando comparado a sensores do tipo laser scan e câmeras, e da sua fácil utilização, em relação principalmente ao sensoriamento por câmeras que precisam de algoritmos de visão computacional.

Ao todo foram utilizados sete sensores ultrassom que foram distribuídos conforme a figura 14.

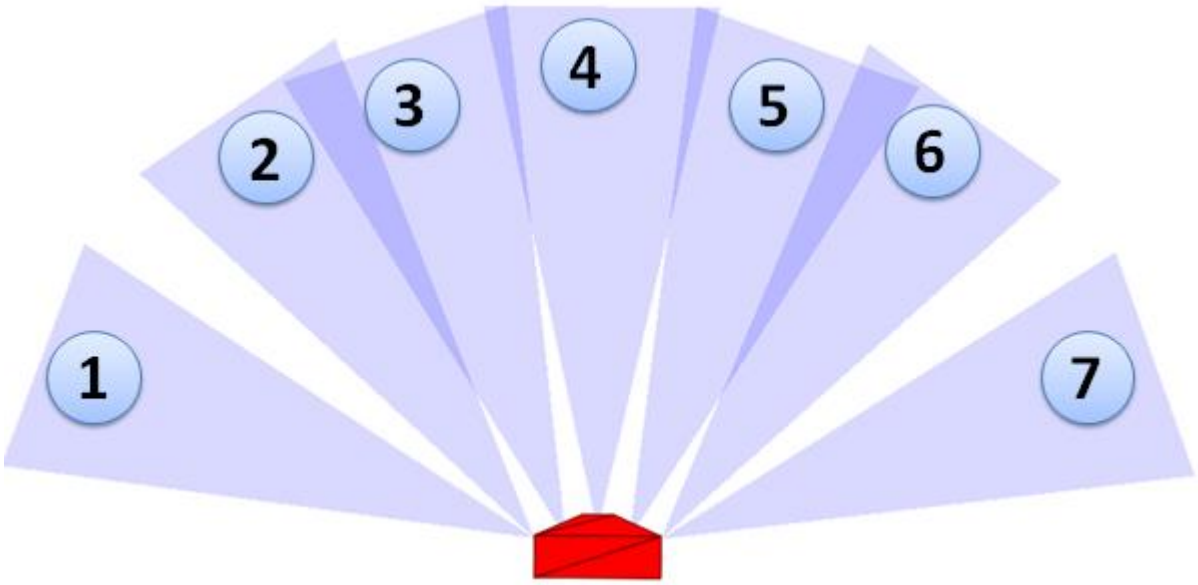

Figura 14 - Robô virtual com seus sensores de distância

- ✓ **Sensor 1** – Identificado como ***Sensor Lateral Esquerdo***
- ✓ **Sensores 2 e 3** – Combinados para aumentar a área de cobertura, compõe o ***Sensor Frente Esquerdo***
- ✓ **Sensor 4** – Identificado como ***Sensor Frontal***
- ✓ **Sensores 5 e 6** – Assim como os sensores 2 e 3, são combinados para aumentar a área de cobertura e compõe o ***Sensor Frente Direito***
- ✓ **Sensor 7** – Identificado como ***Sensor Lateral Direito***.

Além dos sensores de ultrassom, o agente foi dotado de um sensor GPS e uma bússola para que ele possa se localizar no ambiente e em relação ao objetivo. É de boa prática combinar sensores de medida absoluta, no caso o GPS, com relativa, odometria, por exemplo, para que se possam diminuir os erros de leituras, contudo, devido à característica do robô de ser utilizado como a base de uma bengala, a odometria se mostra pouco eficaz, já que o usuário poderia levantar o robô a qualquer momento.

b. Controle Fuzzy

Para que fosse possível aplicar a metodologia fuzzy que será apresentada a seguir, foi criada uma biblioteca de funções que irão realizar todas as etapas do processo de inferência fuzzy. Essas funções foram escritas em C++, devido a linguagem utilizada para o programa cliente, e podem ser encontradas nos anexos 4, 5, 6, 7, 8 e 9.

1. Fuzzificação

Com as características físicas do agente definidas, o primeiro passo para a implementação do controle fuzzy foi definir as variáveis linguísticas com seus possíveis valores e as funções de pertinência associados a cada um deles.

Os dados que serão mostrados nessa etapa são o resultado de vários ajustes feitos durante a fase de testes, sendo está configuração, a que demonstrou melhor resposta.

Os sete sensores de distância, que estão distribuídos conforme figura 14, foram nomeados como: ***Sensor Lateral Esquerdo*** (sensor 1), ***Sensor Frente Esquerdo*** (sensores 2 e 3), ***Sensor Frente*** (sensor 4), ***Sensor Frente Direito*** (sensores 5 e 6) e ***Sensor Lateral Direito*** (sensor 7). Seus valores possíveis são: ***MP*** (Muito Perto), ***P*** (Perto), ***L*** (Longe) e ***ML*** (Muito Longe) e suas funções de pertinência, com formas trapezoidais e triangulares, estão distribuídas conforme a figura 15.

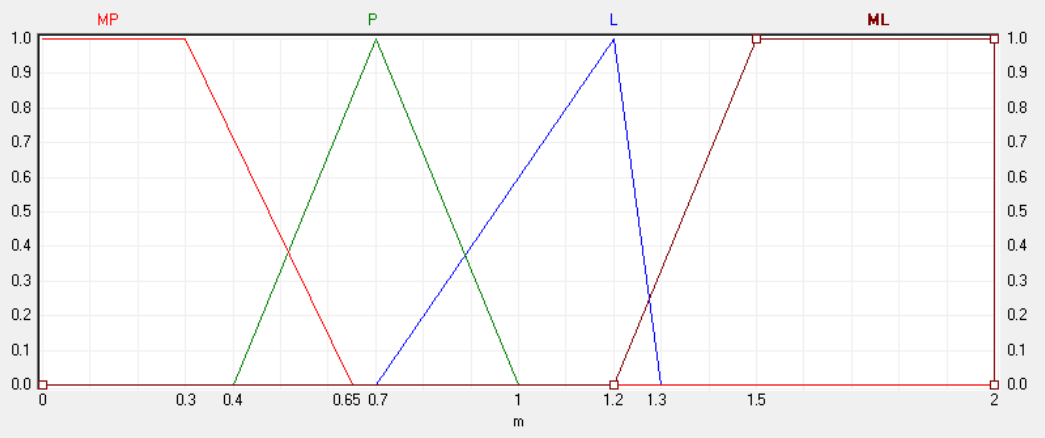

Figura 15 - Funções de pertinência da variável Sensor (...)

Com base no ângulo fornecido pela bussola e a posição do robô dado pelo GPS, foi criada a variável ***Ângulo***, cujo valor representa a orientação do agente em relação ao objetivo. Essa variável pode assumir os seguintes valores: ***NG*** (Negativo Grande), ***NM*** (Negativo Médio), ***Z*** (Zero), ***PM*** (Positivo Médio) e ***PG*** (Positivo Grande), onde, partindo do eixo de rotação, os ângulos positivos são aqueles formados entre a frente do robô e o objetivo à esquerda e os negativos entre a frente e o objetivo à direita, figura 16.

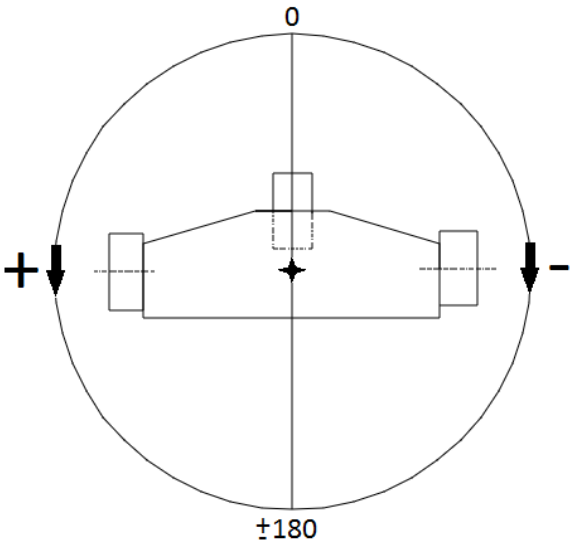

Figura 16 - Referência para o ângulo do robô

As funções de pertinência dessas variáveis linguísticas possuem formas triangulares e estão distribuídas conforme a figura 17.

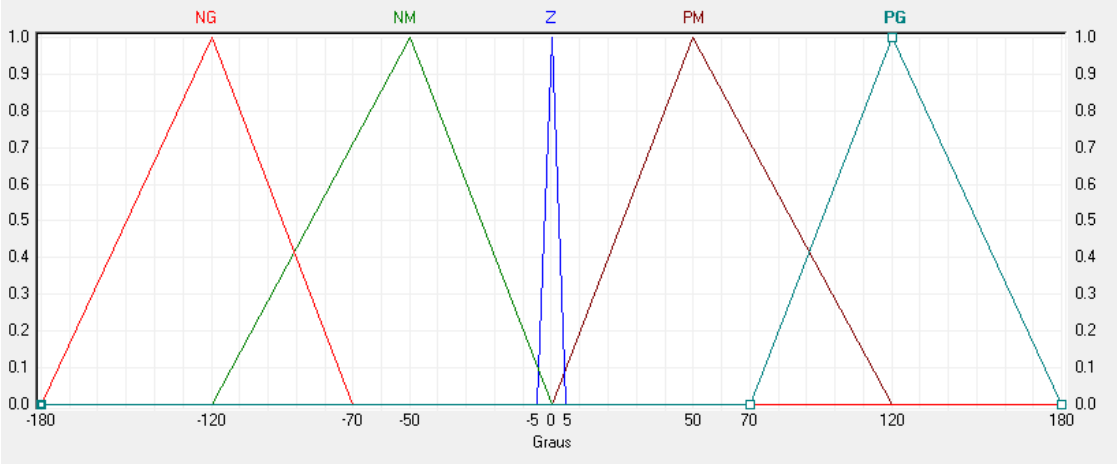

Figura 17 - Funções de pertinência da variável Ângulo

Além de ser utilizado na composição da variável ***Ângulo***, a posição do robô fornecida pelo GPS é emprega no cálculo da distância entre o agente e o ponto objetivo. Esse valor crisp, que representa o menor percurso que, ignorando obstáculos, o robô poderia estabelecer para chegar ao objetivo é representado pela variável fuzzy ***Distância***. Na aplicação real esse termo terá que ser revisado, pois poderá levar o robô a caminhos onde não há saída, o que aumentará o tempo de percurso.

A variável ***Distância***, assim como as variáveis que representam os sensores ultrassom, pode assumir os valores ***NG*** (Negativo Grande), ***NM*** (Negativo Médio), ***Z*** (Zero), ***PM*** (Positivo Médio) e ***PG*** (Positivo Grande) e suas funções de pertinência também possuem forma trapezoidais e triangulares, contudo sua distribuição é diferente, conforme pode ser visto na figura 18.

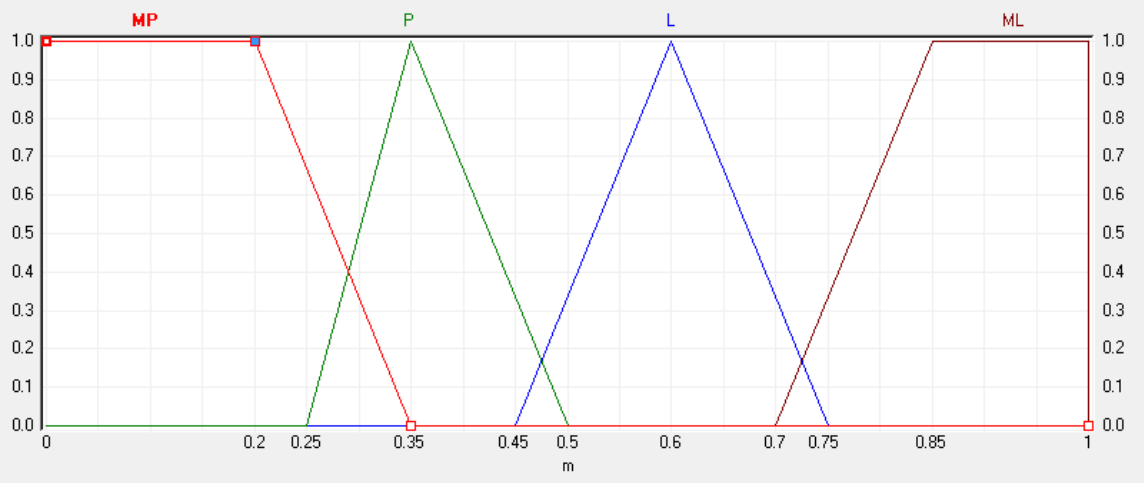

Figura 18 - Funções de pertinência da variável Distância

As ultimas variáveis fuzzy a serem especificadas são as variáveis de saída ***Motor Esquerdo*** e ***Motor Direito***, que representam as potências a serem aplicadas em cada motor, dado o processo de inferência. Essas variáveis podem assumir os valores: ***NA*** (Negativo Alto), ***NM*** (Negativo Médio), ***Z*** (Zero), ***PA*** (Positivo Alto) e ***PM*** (Positivo Médio), cujas funções de pertinência possuem formas trapezoidais e triangulares e estão distribuídas conforme a figura 19.

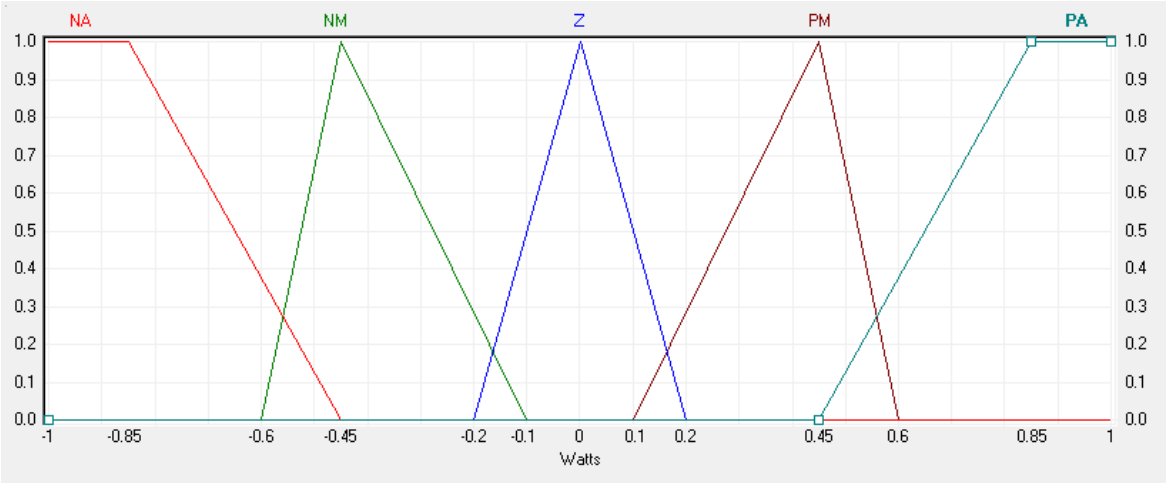

Figura 19 - Funções de pertinência das variáveis Motor Esquerdo e Motor Direito

2. Inferência

A obtenção das regras para o processo de inferência fuzzy foi feita a partir de um especialista e elas foram divididas em dois grupos: regras reativas e regras deliberativas.

As regras reativas representam a maior parte das regras do sistema de inferência e foram obtidas por meio do seguinte raciocínio: para cada valor da variável ***Sensor Frente***, ou seja, de distância ao obstáculo medida pelo sensor ultrassom 4, figura 14, é feita uma comparação entre os valores das variáveis ***Sensor Frente Esquerdo*** (sensores 2 e 3) e ***Sensor Frente Direito*** (sensores 5 e 6). A variável que apresentar o conjunto cujo valor represente a maior distância ao obstáculo será a direção para a qual o robô irá desviar. No caso das ativações das variáveis ***Sensor Frente Direito*** e ***Sensor Frente Esquerdo*** forem iguais, é feita uma comparação entre as variáveis ***Sensor Lateral Esquerdo*** e ***Sensor Lateral Direito*** e a que possuir o maior valor de distância, será a direção escolhida.

Na especificação do grupo de regras deliberativas foi incluído em seus antecedentes a variável ***Ângulo***. Outra característica desse conjunto de regras é que, dependendo do valor de ***Ângulo***, foi possível ignorar diversos valores para as variáveis do tipo ***Sensor***. Por exemplo, se o valor de ***Ângulo*** é ***PM***, objetivo à esquerda do agente, o robô deve virar para a esquerda e, portanto, as leituras dos sensores localizados à direita podem ser ignoradas, figura 20.

p_Angulo[	p_Sensor_Frente_Esq[	p_Sensor_Frente[	p_Sensor_Frente_Dir[	p_Sensor_Lateral_Esq[	p_Sensor_Lateral_Dir[	p_Dist_Objetivo[	].motorE	].motorD
PM						MP	Z	Z
PM	ML	nMP				nMP	NM	PA
PM	L	nMP		nL		nMP	NM	PM
PM	P	nMP		nML		nMP	Z	PM
PM		nMP		ML		nMP	NA	PA
PM		nMP		L		nMP	NA	PA

Figura 20 - Trecho da tabela de regras

Ao todo foram especificadas 165 regras para o controle reativo e 53 regras para o controle deliberativo, totalizando 218 regras, e todas possuem dois consequentes, que são as saídas para cada motor. Dada a quantidade de variáveis fuzzy e de valores que elas poderiam obter, o número de regras possíveis de serem especificadas é de 512000, 5 (valores de ***Ângulo***) X 4x4x4x4x4 (valores de ***Sensor***) X 4 (valores de ***Distância***) X 5x5 (valores de ***Motor***), sendo portando, a quantidade especificada, cerca de 4,26% das regras possíveis.

Esse excelente resultado é fruto da avaliação do especialista, que soube dosar quais regras seriam as mais importantes para resolução do problema.

Como podem ocorrer ativações simultâneas de regras de grupos diferentes, para que o robô priorize o desvio ao obstáculo, as regras referentes ao processo deliberativo tiveram o peso diminuído para 0.7, valor obtido experimentalmente.

No anexo 1, encontra-se uma tabela com todas as 218 regras que foram especificadas para esse trabalho.

3. Defuzzificação

Uma vez obtidos os valores fuzzy das saídas dos motores, é necessário calcular o valor crisp que represente a ativação de seus conjuntos. Para tal, o método de defuzzificação escolhido foi a média ponderada dos máximos dos conjuntos. Esse método foi adotado por permitir que cada conjunto contribua proporcionalmente a sua forma e ativação para a saída do sistema. Por exemplo, dado que variável fuzzy ***Motor Esquerdo*** possua uma ativação de 0.4 no valor ***NM*** e uma ativação de 0.3 no valor ***Z*** (figura 21) o ponto de máximo médio de ***PN*** estará em -0.39 e o de ***Z*** em 0.0 e, portanto:

$$Power = \frac{\sum p_i * \mu_i}{\sum \mu_i} = \frac{-0.39 * 0.4 + 0 * 0.3}{0.4 + 0.3} = -0.223\ watts$$

Onde p_i representa o ponto em que se encontra a média dos máximos do conjunto i e μ_i é a pertinência desse conjunto.

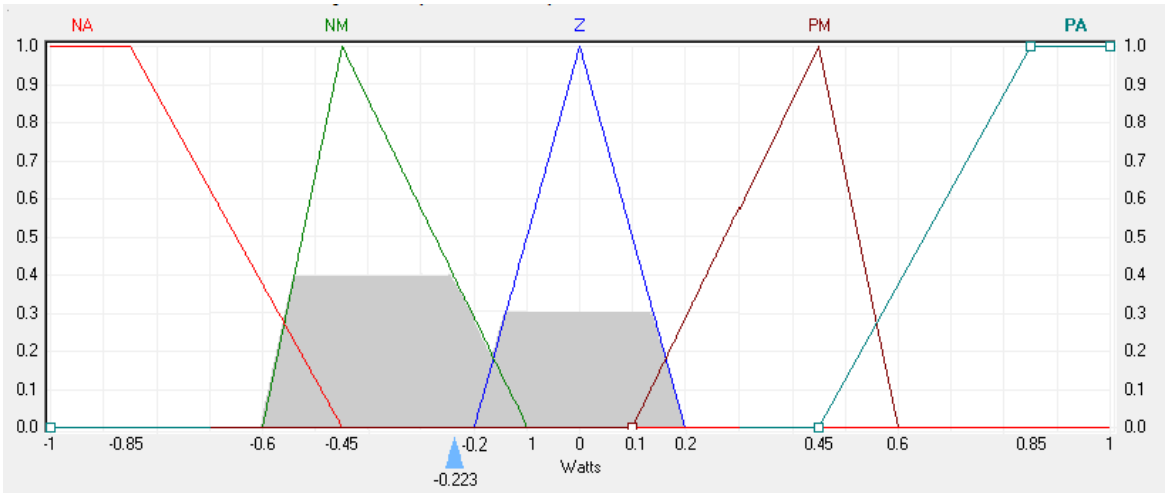

Figura 21 - Exemplo de defuzzificação da variável Motor pelo método da Média Ponderado dos Máximos dos Conjuntos

Como a entrada de dados que o simulador fornece para atuação dos motores é o valor desejado de velocidade angular e linear, foi feita um condicionamento na saída crisp, fornecida pelo processo de defuzzificação, para que o robô respondesse com as velocidades desejadas.

As velocidades lineares e angulares de um robô diferencial podem ser calculadas a partir da velocidade de cada roda pelas seguintes equações:

$$(1)\ v = \frac{v_d + v_e}{2}$$

$$(2)\ \omega = \frac{v_d - v_e}{2 * l}$$

Onde l é a distância entre as rodas.

Baseado as equações 1 e 2, as velocidades do robô foram obtidas substituindo v_e e v_d pelas saídas do processo de defuzzificação das variáveis Motor Esquerdo e Motor Direito, respectivamente. Além dessa substituição, as equações foram multiplicadas por constantes para adequar as velocidades as dimensões do problema.

$$(3)\ v = K_1 * \left(\frac{defuzzy_{Motor\ Direito} + defuzzy_{Motor\ Esquerdo}}{2}\right)$$

$$(4)\ \omega = K_2 * \left(\frac{defuzzy_{Motor\ Direito} - defuzzy_{Motor\ Esquerdo}}{2 * l}\right)$$

Onde $K_1 = 2$ e $K_2 = 64$.

Com isso, o robô poderá atingir a velocidade linear máxima de 2 m/s e a angular máxima de 160 graus/s, já que no caso $l = 0.4m$. Apesar da alta velocidade angular empregada, o deslocamento angular realizado pelo robô a cada ciclo de controle é de no máximo 16 graus, pois a atuação de cada inferência ocorre por 100ms.

Para evitar que o robô tenha um movimento de oscilação frente/traz ao se aproximar de uma parede, por exemplo, foi tomado o cuidado na etapa de criação das regras para que o maior valor em módulo dos consequentes da mesma regra fosse uma potência positiva. Portanto, o equacionamento após a defuzzificação não gera saídas negativas.

No caso em o robô colida a inferência fuzzy dificilmente poderá movê-lo, pois como não se executa movimento de ré, o agente não terá espaço para contornar o obstáculo. Para esse layout específico, foi criada uma rotina fora do fuzzy que verifica se o robô está parado por mais de 3 segundos, e caso esteja, ele executa uma ré seguida de uma rotação para direita. Essa contagem foi estabelecida, pois o simulador não contempla sensores de contato.

6. Testes e resultados

Para validar o conjunto de regras e a disposição dos sensores adotadas para o robô, foram realizados diversos testes com diferentes percursos, onde era avaliada a maneira como o robô contornava e se batia no obstáculo, se conseguia chegar ao objetivo e se a trajetória executada não apresentava desvios desnecessários.

Durante os testes realizados, foi verificada uma latência na execução do simulador que gerava alterações no tempo de atuação das saídas do processo de inferência fuzzy. Essa variação expunha o agente em diferentes situações, o que podia gerar novos percursos a cada vez que o mesmo trajeto era realizado. Apesar no caráter não ideal, essa variação ajudou no processo de desenvolvimento, uma vez que exigia um controle mais robusto.

Devido a latência do processo de simulação, para obtenção de resultados mais concretos, o mesmo percurso foi repetido mais de uma vez em cada teste.

a. Teste percurso 1

O primeiro teste realizado foi com um circuito com poucos obstáculos, figura 22, onde o objetivo estava alinhado com a posição inicial do robô.

O processo foi repetido dez vezes consecutivas, das quais, nove, o robô realizou todo trajeto sem colidir e na única vez que colidiu, trajeto 1.d, o controle externo ao fuzzy atuou, recuando o robô do obstáculo, e em seguida o fuzzy conseguiu realizar o restante do trajeto sem problemas.

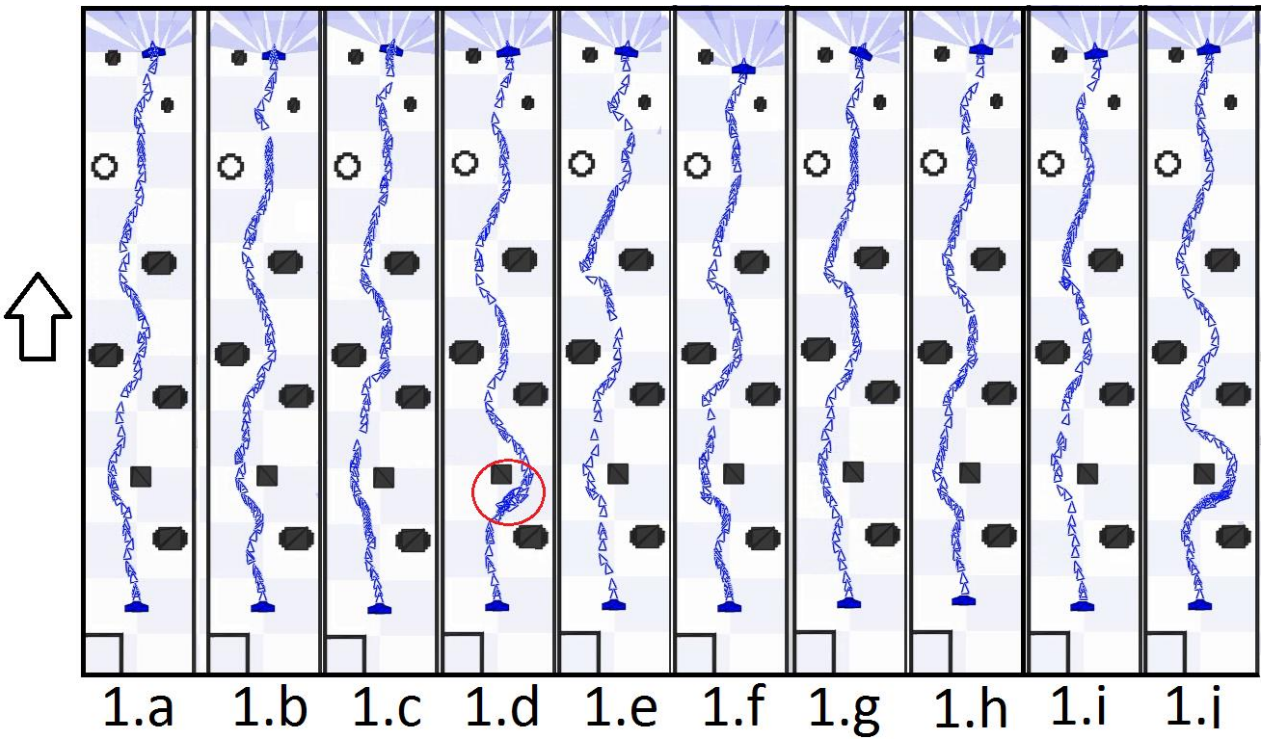

Figura 22 - Percursos realizados no trajeto 1

b. Teste percurso 2

No segundo trajeto o robô inicia no sentido oposto ao objetivo, figura 23. Esse circuito possui um grau de dificuldade bem maior que o anterior devido ao grande número de obstáculos.

Para essa etapa, também foram executadas dez tentativa consecutivas, das quais, cinco (2.a, 2.b, 2.c, 2.e e 2.h), o robô conseguiu realizar o trajeto sem colisões, três (2.d, 2.g e 2.h), realizou o trajeto com uma colisão, uma (2.i), com duas colisões e uma (2.f) ele apresentou desvios desnecessários para execução de seu trajeto.

Assim como no teste anterior, nos trajetos que ocorreram colisões, a rotina externa ao fuzzy atuou para a solução do problema.

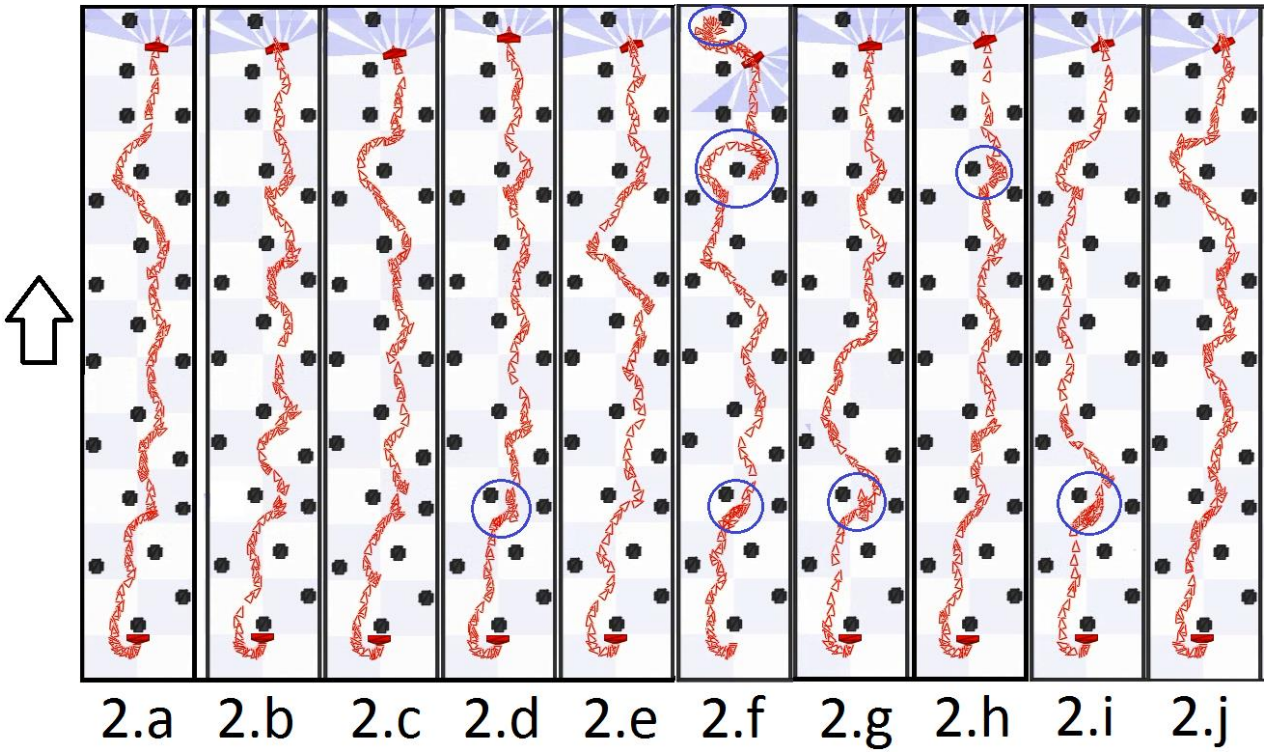

Figura 23 - Percursos realizados no trajeto 2

c. Teste percurso 3

A terceira bateria de testes visava observar o comportamento do robô ao deparar com um obstáculo dinâmico e em rota de colisão, figura 24. Também foram executadas dez tentativas consecutivas, nas quais, três (3.b, 3.e e 3.j), os robôs conseguiram executar a trajetória sem colisões, três (3.c, 3.d e 3.f), executaram a trajetória com apenas uma colisão, três (3.a, 3.g e 3.h), apenas um dos robôs conseguiu executar a trajetória e uma (3.i) nenhum robô conseguiu chegar ao objetivo .

Apesar do pior desempenho apresentada nessa configuração, ele pode ser justificada pelo fato dos dois robôs estarem executando o mesmo código e, portanto, ambos tentam desviar para o mesmo ponto, o que gera a colisão. Esse comportamento o é similar ao que pode ocorre quando duas pessoas de deparam, onde ambas tentam desviar para a direção da região mais livre.

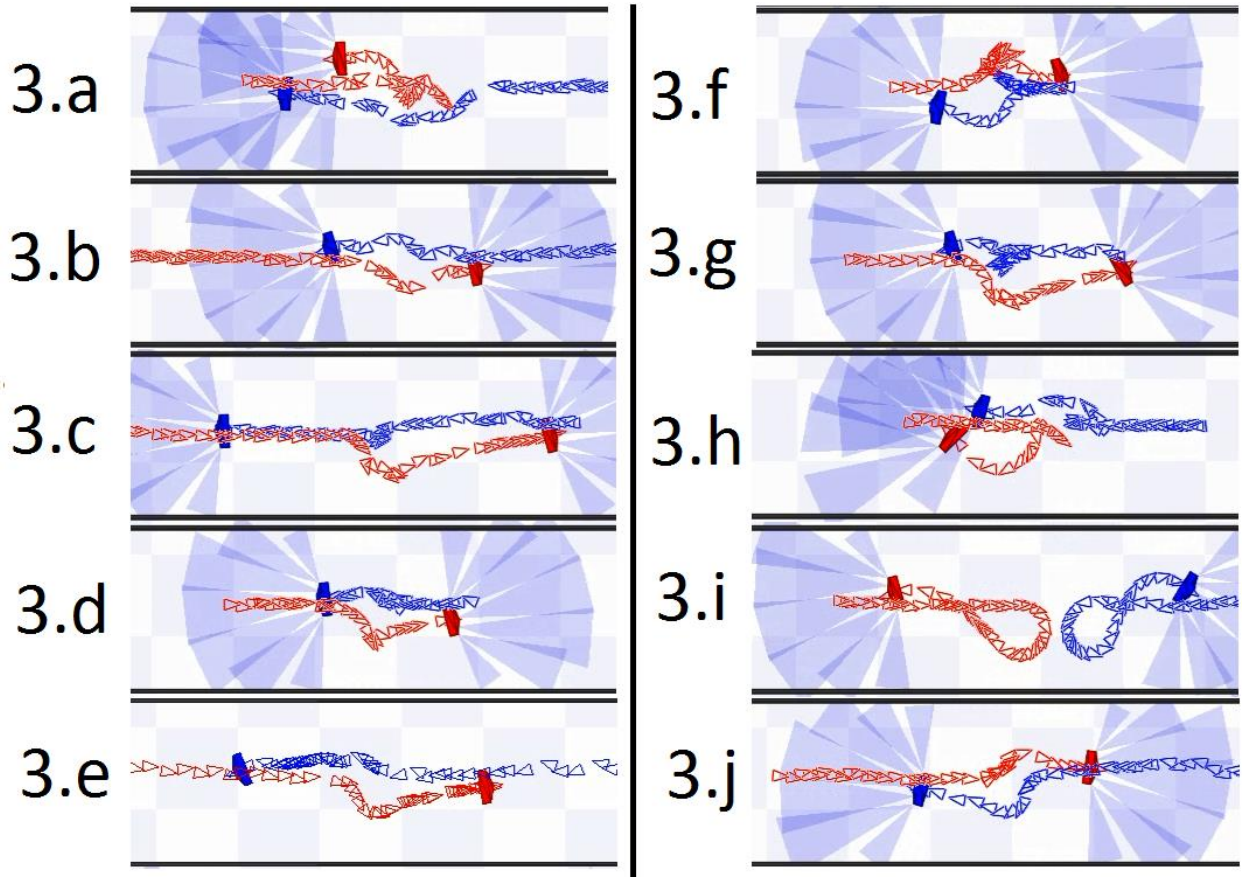

Figura 24 - Percursos realizados no trajeto 3

d. Outras configurações

Além das três configurações de circuitos de testes apresentadas, também foi verificada a atuação em situações mais específicas, como a de um corredor fechado, figura 25. Nesse tipo de percurso o robô apresentou excelente desempenho, conseguindo contornar o problema sem apresentar nenhuma colisão todas às fezes que o teste foi feito.

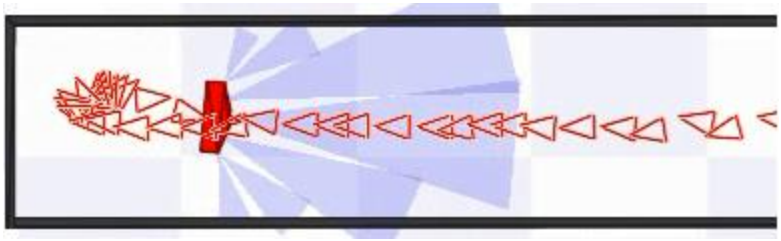

Figura 25 - Percurso em corredor fechado

Outra configuração na qual o algoritmo mostrou um bom desempenho foi em percursos onde o robô possui pouco espaço de manobra, mas sem obstáculos a frente, como o circuito da figura 26. Nesse layout específico o robô conseguiu executar sua trajetória sem colisões até atingir o ponto "B". Onde o controle externo ao fuzzy atuou para a correção.

Nesse circuito também foi possível notar uma pequena limitação na configuração das regras, pois apesar de seu objetivo final ser no ponto "A", o robô sempre desviava para esquerda ao passar pelo ponto "B". Esse comportamento se deve pelo maior peso das regras reativas, pois como a parede que está à direita do robô se encontra muito próxima, as regras reativas ficam grande ativação e conseguem suprimir as regras deliberativas, que possuem menor peso, levando o robô para região que oferece menor risco de colisões.

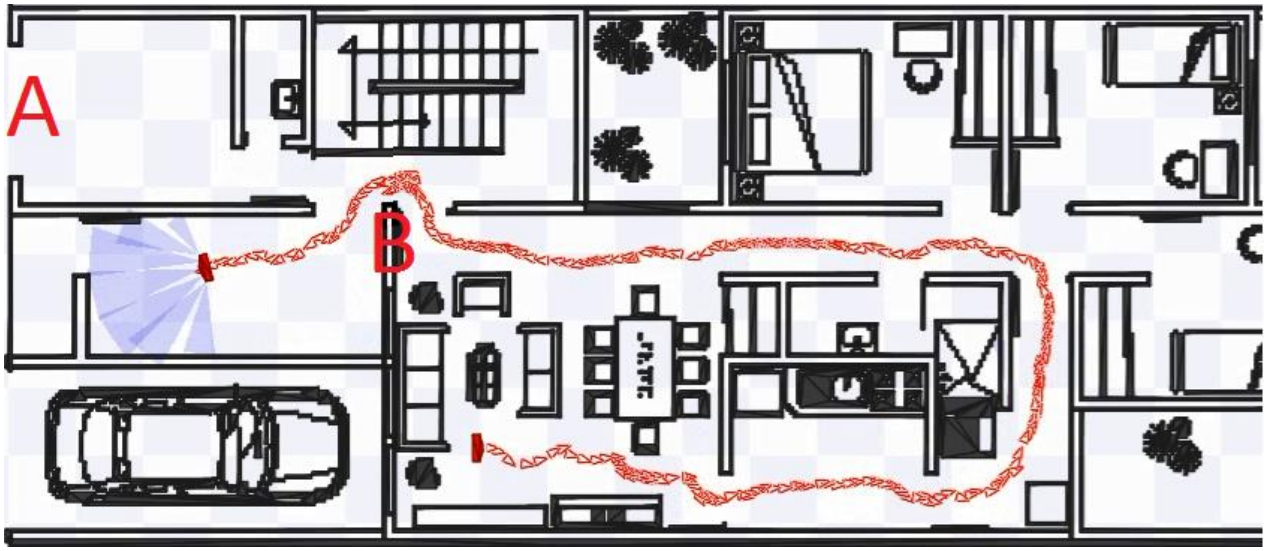

Figura 26 - Percurso dentro de uma casa

7. Conclusão

Com base nos testes realizados, é possível comprovar que o controle fuzzy se mostrou uma excelente alternativa para dotar o robô móvel de autonomia, pois sem que fosse necessária uma modelagem matemática do comportamento dinâmico do robô, sem um mapa para auxiliar sua orientação e com apenas poucas regras linguísticas, foi possível realizar diferentes configurações de percurso e sempre com ótima resposta, na qual, os obstáculos sempre eram superados de maneira suave.

Com relação às limitações observadas no conjunto de regras especificadas, ao priorizar o desvio em relação ao deslocamento para o objetivo, em algumas configurações, o robô não conseguia acessar um caminho estreito, pois priorizava uma região de menor risco. Contudo, esse maior peso dado às regras reativas, permitiu que o robô conseguisse sair de situações onde o obstáculo estivesse entre ele e o ponto objetivo.

Outro ponto a ser observado, foram as diferenças de percursos realizadas num mesmo trajetos com as mesmas condições inicias. Essas diferenças, conforme apresentado anteriormente, foram causadas pela latência presente na execução das simulações. Essa latência acabou por gerar um comportamento não ideal, que estará presente em maior escala numa aplicação real.

Apesar de possuir alguns pontos para serem melhorados, acredita-se que a proposta desse projeto, de servir como guia para uma pessoa, tenha sido alcança em parte, pois o robô consegue realizar trajetórias complexas desviando-se de maneira suave e sem grades oscilações em sua velocidade, de posse somente da coordenada GPS do ponto objetivo.

8. Bibliografia

[1] ROBOTIC INDUSTRIES ASSOCIATION – Disponível em: <http://www.robotics.org/content-detail.cfm/Industrial-Robotics-Editorials/Robotics-for-Senior-Management/content_id/992 - Acesso em 15/11/2013

[2] Meggiolaro, M.A. Notas de Aula: Introdução a Sistemas Robóticos. Departamento de Engenharia Mecânica – PUC, Rio de Janeiro, RJ.

[3] Leonard, J.J. ; NEC Res. Inst., Princeton, NJ, USA ; Durrant-Whyte, H.F. "Simultaneous map building and localization for an autonomous mobile robot" - Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on - pp 1442 - 1447 vol.3

[4] J. Borenstein , H. R. Everett , and L. Feng Contributing authors: S. W. Lee and R. H. Byrne - "Where am I? Sensors and Methods for Mobile Robot Positioning" - Edited and compiled by J. Borenstein - April 1996

[5] R. Tanscheit. Notas de Aula: Sistemas Fuzzy. Departamento de Engenharia Elétrica – PUC, Rio de Janeiro, RJ.

[6] Brian Gerkey and contributors - Player Manual - http://playerstage.sourceforge.net/doc/Player-3.0.2/player/index.html - Acesso em 15/11/13

[7] Richard Vaughan and contributors - Stage Manual - http://rtv.github.io/Stage/ - Acesso em 15/11/13

[8] Kioskea Brasil – "O protocolo TCP". Disponível em: http://pt.kioskea.net/contents/284-o-protocolo-tcp . Acessado 17/11/2013.

[9] Maria Isabel Ribeiro – "Uma Viagem ao Mundo dos Robots" - Instituto de Sistemas e Robótica Instituto Superior Técnico Lisboa, Portugal. Disponível em:< http://users.isr.ist.utl.pt/~mir/pub/ViagemRobots-IsabelRibeiro05.pdf>. Acessado 16/11/2013.

[10] Dr. Humberto Secchi – "Uma Introdução aos Robôs Móveis" Instituto de Automática – INAUT Universidade Nacional de San Juan – UNSJ – Argentina. Disponível em: < http://www.obr.org.br/wp-content/uploads/2013/04/Uma_Introducao_aos_Robos_Moveis.pdf> Acessado em: 15/11/2013

[11] Huosheng Hu and Dongbing Gu – "Landmark-based Navigation of Industrial Mobile Robots" - International Journal of Industry Robot, Vol. 27, No. 6, 2000, pages 458 - 467

[12] Roland Siegwart, Margarita Chli, Martin Rufli e Davide Scaramuzza, Notas de Aula: Autonomous Mobile Robotics - ETH Zurich, Zurich, Alemanha. Disponível em: <http://www.asl.ethz.ch/education/master/mobile_robotics> Acessado em: 20/11/2013.

ANEXOS

1. Tabela de Regras

numero da regra	p_Angulo[	p_Sensor_Frente_Esq[	p_Sensor_Frente[	p_Sensor_Frente_Dir[	p_Sensor_Lateral_Esq[	p_Sensor_Lateral_Dir[	p_Dist_Objetivo[	].motorE	].motorD
0		MP	MP	nMP			nMP	PM	NM
1		P	MP	L			nMP	PM	NM
2		P	MP	ML			nMP	PM	NM
3		L	MP	ML			nMP	PM	NM
4		nMP	MP	MP			nMP	NM	PM
5		L	MP	P			nMP	NM	PM
6		ML	MP	P			nMP	NM	PM
7		ML	MP	L			nMP	NM	PM
8		MP	P	nMP			nMP	PA	NM
9		P	P	L			nMP	PA	NM
10		P	P	ML			nMP	PA	NM
11		L	P	ML			nMP	PA	NM
12		nMP	P	MP			nMP	NM	PA
13		L	P	P			nMP	NM	PA
14		ML	P	P			nMP	NM	PA
15		ML	P	L			nMP	NM	PA
16		MP	L	nMP			nMP	PA	NM
17		P	L	L			nMP	PA	Z
18		P	L	ML			nMP	PA	Z
19		L	L	ML			nMP	PA	Z
20		nMP	L	MP			nMP	NM	PA
21		L	L	P			nMP	Z	PA
22		ML	L	P			nMP	Z	PA
23		ML	L	L			nMP	Z	PA
24		MP	ML	nMP			nMP	PA	NM
25		P	ML	L			nMP	PA	Z
26		P	ML	ML			nMP	PA	Z
27		L	ML	ML			nMP	PM	Z
28		nMP	ML	MP			nMP	NM	PA
29		L	ML	P			nMP	Z	PA
30		ML	ML	P			nMP	Z	PA
31		ML	ML	L			nMP	Z	PM
32		L	MP	L	MP	nMP	nMP	PM	NM
33		L	MP	L	P	L	nMP	PM	NM
34		L	MP	L	P	ML	nMP	PM	NM
35		L	MP	L	L	ML	nMP	PM	NM
36		L	MP	L	nMP	MP	nMP	NM	PM
37		L	MP	L	L	P	nMP	NM	PM
38		L	MP	L	ML	P	nMP	NM	PM
39		L	MP	L	ML	L	nMP	NM	PM
40		L	MP	L	MP	MP	nMP	PA	NA
41		L	MP	L	P	P	nMP	PA	NA

numero da regra	p_Angulo[	p_Sensor_Frente_Esq[	p_Sensor_Frente[	p_Sensor_Frente_Dir[	p_Sensor_Lateral_Esq[	p_Sensor_Lateral_Dir[	p_Dist_Objetivo[	].motorE	].motorD
42		L	MP	L	L	L	nMP	PA	NA
43		L	MP	L	ML	ML	nMP	PA	NA
44		L	P	L	MP	nMP	nMP	PM	Z
45		L	P	L	P	L	nMP	PA	NM
46		L	P	L	P	ML	nMP	PA	NM
47		L	P	L	L	ML	nMP	PA	NM
48		L	P	L	nMP	MP	nMP	Z	PM
49		L	P	L	L	P	nMP	NM	PA
50		L	P	L	ML	P	nMP	NM	PA
51		L	P	L	ML	L	nMP	NM	PA
52		L	P	L	MP	MP	nMP	PA	NA
53		L	P	L	P	P	nMP	PA	NA
54		L	P	L	L	L	nMP	PA	NA
55		L	P	L	ML	ML	nMP	PA	NA
56		L	L	L	MP	nMP	nMP	PM	Z
57		L	L	L	P	L	nMP	PM	Z
58		L	L	L	P	ML	nMP	PM	Z
59		L	L	L	L	ML	nMP	PM	Z
60		L	L	L	nMP	MP	nMP	Z	PM
61		L	L	L	L	P	nMP	Z	PM
62		L	L	L	ML	P	nMP	Z	PM
63		L	L	L	ML	L	nMP	Z	PM
64		L	L	L	MP	MP	nMP	PM	PM
65		L	L	L	P	P	nMP	PM	PM
66		L	L	L	L	L	nMP	PM	PM
67		L	L	L	ML	ML	nMP	PM	PM
68	Z	L	ML	L			nMP	PM	PM
69		P	MP	P	MP	nMP	nMP	PM	NM
70		P	MP	P	P	L	nMP	PM	NM
71		P	MP	P	P	ML	nMP	PM	NM
72		P	MP	P	L	ML	nMP	PM	NM
73		P	MP	P	nMP	MP	nMP	NM	PM
74		P	MP	P	L	P	nMP	NM	PM
75		P	MP	P	ML	P	nMP	NM	PM
76		P	MP	P	ML	L	nMP	NM	PM
77		P	MP	P	MP	MP	nMP	PA	NA
78		P	MP	P	P	P	nMP	PA	NA
79		P	MP	P	L	L	nMP	PA	NA
80		P	MP	P	ML	ML	nMP	PA	NA
81		P	P	P	MP	nMP	nMP	PM	Z
82		P	P	P	P	L	nMP	PM	Z
83		P	P	P	P	ML	nMP	PM	Z
84		P	P	P	L	ML	nMP	PM	Z
85		P	P	P	nMP	MP	MP	Z	PM
86		P	P	P	L	P	nMP	Z	PM
87		P	P	P	ML	P	nMP	Z	PM
88		P	P	P	ML	L	nMP	Z	PM
89		P	P	P	MP	MP	nMP	PA	NM
90		P	P	P	P	P	nMP	PA	NM

numero da regra	p_Angulo[	p_Sensor_Frente_Esq[	p_Sensor_Frente[	p_Sensor_Frente_Dir[	p_Sensor_Lateral_Esq[	p_Sensor_Lateral_Dir[	p_Dist_Objetivo[	].motorE	].motorD
91		P	P	P	L	L	nMP	PA	NM
92		P	P	P	ML	ML	nMP	PA	NM
93		P	L	P	MP	nMP	nMP	PM	Z
94		P	L	P	P	L	nMP	PA	PM
95		P	L	P	P	ML	nMP	PA	PM
96		P	L	P	L	ML	nMP	PA	PM
97		P	L	P	nMP	MP	nMP	Z	PM
98		P	L	P	L	P	nMP	PM	PA
99		P	L	P	ML	P	nMP	PM	PA
100		P	L	P	ML	L	nMP	PM	PA
101		P	L	P	MP	MP	nMP	PM	PM
102		P	L	P	P	P	nMP	PM	PM
103		P	L	P	L	L	nMP	PA	PM
104		P	L	P	ML	ML	nMP	PA	PM
105		P	ML	P			nMP	PM	PM
106		MP	MP	MP	MP	nMP	nMP	PM	NM
107		MP	MP	MP	P	L	nMP	PM	NM
108		MP	MP	MP	P	ML	nMP	PM	NM
109		MP	MP	MP	L	ML	nMP	PM	NM
110		MP	MP	MP	nMP	MP	nMP	NM	PM
111		MP	MP	MP	L	P	nMP	NM	PM
112		MP	MP	MP	ML	P	nMP	NM	PM
113		MP	MP	MP	ML	L	nMP	NM	PM
114		MP	MP	MP	MP	MP	nMP	PA	NA
115		MP	MP	MP	P	P	nMP	PA	NA
116		MP	MP	MP	L	L	nMP	PA	NA
117		MP	MP	MP	ML	ML	nMP	PA	NA
118		MP	P	MP	MP	nMP	nMP	PM	NM
119		MP	P	MP	P	L	nMP	PM	NM
120		MP	P	MP	P	ML	nMP	PM	NM
121		MP	P	MP	L	ML	nMP	PM	NM
122		MP	P	MP	nMP	MP	MP	NM	PM
123		MP	P	MP	L	P	nMP	NM	PM
124		MP	P	MP	ML	P	nMP	NM	PM
125		MP	P	MP	ML	L	nMP	NM	PM
126		MP	P	MP	MP	MP	nMP	PM	NA
127		MP	P	MP	P	P	nMP	PM	NA
128		MP	P	MP	L	L	nMP	PM	NA
129		MP	P	MP	ML	ML	nMP	PM	NA
130		MP	L	MP	MP	nMP	nMP	PM	Z
131		MP	L	MP	P	L	nMP	PM	Z
132		MP	L	MP	P	ML	nMP	PM	Z
133		MP	L	MP	L	ML	nMP	PM	Z
134		MP	L	MP	nMP	MP	nMP	Z	PM
135		MP	L	MP	L	P	nMP	Z	PM
136		MP	L	MP	ML	P	nMP	Z	PM

numero da regra	p_Angulo[	p_Sensor_Frente_Esq[	p_Sensor_Frente[	p_Sensor_Frente_Dir[	p_Sensor_Lateral_Esq[	p_Sensor_Lateral_Dir[	p_Dist_Objetivo[	].motorE	].motorD
137		MP	L	MP	ML	L	nMP	Z	PM
138		MP	L	MP	MP	MP	nMP	PA	NA
139		MP	L	MP	P	P	nMP	PA	NA
140		MP	L	MP	L	L	nMP	PA	NA
141		MP	L	MP	ML	ML	nMP	PA	NA
142		MP	ML	MP	MP	nMP	nMP	PM	Z
143		MP	ML	MP	P	L	nMP	PM	Z
144		MP	ML	MP	P	ML	nMP	PM	Z
145		MP	ML	MP	L	ML	nMP	PM	Z
146		MP	ML	MP	nMP	MP	nMP	Z	PM
147		MP	ML	MP	L	P	nMP	Z	PM
148		MP	ML	MP	ML	P	nMP	Z	PM
149		MP	ML	MP	ML	L	nMP	Z	PM
150		MP	ML	MP	MP	MP	nMP	PM	NM
151		MP	ML	MP	P	P	nMP	PM	NM
152		MP	ML	MP	L	L	nMP	PM	NM
153		MP	ML	MP	ML	ML	nMP	PM	NM
154		ML	MP	ML	MP	nMP	nMP	PM	NM
155		ML	MP	ML	P	L	nMP	PM	NM
156		ML	MP	ML	P	ML	nMP	PM	NM
157		ML	MP	ML	L	ML	nMP	PM	NM
158		ML	MP	ML	nMP	MP	nMP	NM	PM
159		ML	MP	ML	L	P	nMP	NM	PM
160		ML	MP	ML	ML	P	nMP	NM	PM
161		ML	MP	ML	ML	L	nMP	NM	PM
162		ML	MP	ML	MP	MP	nMP	PA	NA
163		ML	MP	ML	P	P	nMP	PA	NA
164		ML	MP	ML	L	L	nMP	PA	NA
165		ML	MP	ML	ML	ML	nMP	PA	NA
166	Z	ML	P	ML	MP	nMP	nMP	PA	NM
167	Z	ML	P	ML	P	L	nMP	PA	NM
168	Z	ML	P	ML	P	ML	nMP	PA	NM
169	Z	ML	P	ML	L	ML	nMP	PA	NM
170	Z	ML	P	ML	nMP	MP	nMP	NM	PA
171	Z	ML	P	ML	L	P	nMP	NM	PA
172	Z	ML	P	ML	ML	P	nMP	NM	PA
173	Z	ML	P	ML	ML	L	nMP	NM	PA
174	Z	ML	P	ML	MP	MP	nMP	PA	NA
175	Z	ML	P	ML	P	P	nMP	PA	NA
176	Z	ML	P	ML	L	L	nMP	PA	NA
177	Z	ML	P	ML	ML	ML	nMP	PA	NA
178	Z	ML	L	ML	MP	nMP	nMP	PM	Z
179	Z	ML	L	ML	P	L	nMP	PM	Z
180	Z	ML	L	ML	P	ML	nMP	PM	Z
181	Z	ML	L	ML	L	ML	nMP	PM	Z
182	Z	ML	L	ML	nMP	MP	nMP	Z	PM
183	Z	ML	L	ML	L	P	nMP	Z	PM

numero da regra	p_Angulo[	p_Sensor_Frente_Esq[	p_Sensor_Frente[	p_Sensor_Frente_Dir[	p_Sensor_Lateral_Esq[	p_Sensor_Lateral_Dir[	p_Dist_Objetivo[	].motorE	].motorD
184	Z	ML	L	ML	ML	P	nMP	Z	PM
185	Z	ML	L	ML	ML	L	nMP	Z	PM
186	Z	ML	L	ML	MP	MP	nMP	PM	PM
187	Z	ML	L	ML	P	P	nMP	PM	PM
188	Z	ML	L	ML	L	L	nMP	PM	PM
189	Z	ML	L	ML	ML	ML	nMP	PM	PM
190	Z						MP	Z	Z
191	Z	ML	ML	ML			nMP	PA	PA
192	Z	L	ML	L			nMP	PA	PA
193	Z	P	ML	P			nMP	PM	PM
194	NM						MP	Z	Z
195	NM		nMP	ML			nMP	PA	NM
196	NM		nMP	L			nMP	PM	NM
197	NM		nMP	P		nML	nMP	PM	Z
198	NM		nMP			ML	nMP	PA	NA
199	NM		nMP			L	nMP	PA	NA
200	NG						MP	Z	Z
201	NG		nMP	ML			nMP	PA	NM
202	NG		nMP	L			nMP	PA	NM
203	NG		nMP	P		nML	nMP	PA	NM
204	NG		nMP			ML	nMP	PA	NA
205	NG		nMP			L	nMP	PA	NA
206	PM						MP	Z	Z
207	PM	ML	nMP				nMP	NM	PA
208	PM	L	nMP		nL		nMP	NM	PM
209	PM	P	nMP		nML		nMP	Z	PM
210	PM		nMP		ML		nMP	NA	PA
211	PM		nMP		L		nMP	NA	PA
212	PG						MP	Z	Z
213	PG	ML	nMP				nMP	NM	PA
214	PG	L	nMP		nL		nMP	NM	PA
215	PG	P	nMP		nML		nMP	NM	PA
216	PG		nMP		ML		nMP	NA	PA
217	PG		nMP		L		nMP	NA	PA

2. Código Principal

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <math.h>
#include <iostream>
#include <fstream>
#include <time.h>
#include <libplayerc++/playerc++.h>
#include <Stage-4.1/stage.hh>

#include "pertinencia.h"
#include "pertinencia_palavra.h"
#include "pertinencia_palavra_inv.h"
#include "exec_antecedente.h"
#include "calcula_angulo_and_dist.h"
#include "defuzzy.h"
#include "defuzzy_palavra.h"

#define PI 3.14159265
// CONJUNTOS PARA A POTENCIA DO MOTOR
#define NA 0
#define NM 1
#define Z 2
#define PM 3
#define PA 4

// NEGAÇÃO DOS CONJUNTOS PARA A POTENCIA DO MOTO
#define nNA 5
#define nNM 6
#define nZ 7
#define nPM 8
#define nPA 9
//*****************************

// CONJUNTOS PARA OS ANGULOS
#define NG 0
#define NM 1
#define Z 2
#define PM 3
#define PG 4


// NEGAÇÃO CONJUNTOS PARA OS ANGULOS
#define nNG 5
#define nNM 6
#define nZ 7
#define nPM 8
#define nPG 9
//*****************************

// CONJUNTOS PARA DISTANCIA AO OBSTACULO E AO OBJETIVO
#define ML 0
#define L 1
#define P 2
#define MP 3

// NEGAÇÃO CONJUNTOS PARA DISTANCIA AO OBSTACULO E AO OBJETIVO
#define nML 4
#define nL 5
#define nP 6
#define nMP 7
//*****************************
```

```
// ESTRUTURA DA REGRA
struct regra_fuzzy
{
    int n_conjuntos; // NÚMERO DE ANTECEDENTES DA REGRA
    double motorE;   // CONJUNTO DE SAÍDA PARA VARIÁVEL motorE
    double motorD;   // CONJUNTO DE SAÍDA PARA VARIÁVEL motorD
    double  **pertinencias; // PERTINÊNCIA DOS ANTECEDENTES DA REGRA
    double result_antecedente; // ATIVAÇÃO DA REGRA

};

// Macro que retorna o menor entre 2 valores
#define Min(x, y) (((x) < (y)) ? (x): (y))

// Macro que retorna o maior entre 2 valores
#define Max(x, y) (((x) > (y)) ? (x): (y))
// Para usar operador Min poerador[i] == 1; Max opeardor[i] == 0


using namespace std;
using namespace PlayerCc;

// Conjuntos fuzzy para as variaveis Sensro (...)
double ** sensor;
double sensor_ML[] = {1.2,1.50,1.50};           //MUITO LONGE
double sensor_L[]  = {0.70,1.2,1.30};           //LONGE
double sensor_P[]  = {0.40,0.70,1.0};           //PERTO
double sensor_MP[] = {0.30, 0.30,0.65};         //MUITO PERTO

// Conjuntos fuzzy para variavel Ângulo
double ** angulo;

double ang_NG[] = {-180*(PI/180), -120*(PI/180), -70*(PI/180)};//NEGATIVO GRANDE
double ang_NM[] = {-120*(PI/180), -50*(PI/180),  0*(PI/180)};  //NEGATIVO MEDIO
double ang_Z[]  = {-5*(PI/180),    0*(PI/180),   5*(PI/180)};   //ZERO
double ang_PM[] = { 0*(PI/180),   50*(PI/180), 122*(PI/180)};  //POSITIVO MEDIO
double ang_PG[] = { 70*(PI/180),   120*(PI/180), 180*(PI/180)};//POSITIVO GRANDE

// Conjuntos fuzzy para variriavel Distância

double ** dist_objetivo;

double dist_ML[] = {0.70,0.85,0.85};        //MUITO LONGE
double dist_L[]  = {0.45,0.60,0.75};        //LONGE
double dist_P[]  = {0.25,0.35,0.50};        //PERTO
double dist_MP[] = {0.20, 0.20,0.35};       //MUITO PERTO

// Conjuntos fuzzy paras variaveis Motor (...)
double ** motor;
double motor_NA[] = {-0.85,-0.85,-0.45};    //NEGATIVO ALTO
double motor_NM[] = {-0.60,-0.45,-0.1};     //MEGATOVO MEDIO
double motor_Z[]  = {-0.2,0,0.2};           //ZERO
double motor_PM[] = {0.1,0.45,0.60};        //POSITIVO MEIDO
double motor_PA[] = {0.45,0.85,0.85};       //POSITIVO ALTO

// Pertinencias as variaveis dos conjuntos fuzzy
double p_Sensor_Frente_Esq[8];
double p_Sensor_Frente[8];
double p_Sensor_Frente_Dir[8];

double p_Sensor_Lateral_Esq[8];
//double p_Sensor_Esq_Frente[8];

double p_Sensor_Lateral_Dir[8];
```

```
//double p_Sensor_Dir_Frente[8];

double p_Angulo[10];
double p_Dist_Objetivo[8];

int N_REGRAS = 218;
regra_fuzzy regra[218];
double ang_and_dist[2];
double motorE[5];
double motorD[5];
double ranger[7];
double objetivo_x = -33.7;//-7;
double objetivo_y = -21.43;//104;
double limit_dir = 1;       // limite direito conjunto motor
double limit_esq = -1;      // limite esquerdo conjunto motor
double v = 0;
double w = 0;
double PowerE = 0;
double PowerD = 0;
double direita = 0;
double esquerda = 0;
double frente = 0;
int media;
int passo = -1;
double ti,tf,tempo; // ti = tempo inicial // tf = tempo final
ofstream myfile;
double tempo_acc = 0;
double temp1,temp2, px1,py1,teta1,esp;
double espera = 3000.0;  // 3000 ms
double peso = 0.7;
int cont1 = 0;
int cont2 = 0;
int casa = 1;


int main(int argc, char *argv[])
{
regra[0].n_conjuntos = 4;
double * perti_r0[4] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[MP],
                        &p_Sensor_Frente_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[0].motorE = PM;   regra[0].motorD = NM;   regra[0].pertinencias = perti_r0;

regra[1].n_conjuntos = 4;
double * perti_r1[4] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                        &p_Sensor_Frente_Dir[L],&p_Dist_Objetivo[nMP],};
regra[1].motorE = PM;   regra[1].motorD = NM;   regra[1].pertinencias = perti_r1;

regra[2].n_conjuntos = 4;
double * perti_r2[4] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                        &p_Sensor_Frente_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[2].motorE = PM;   regra[2].motorD = NM;   regra[2].pertinencias = perti_r2;

regra[3].n_conjuntos = 4;
double * perti_r3[4] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
                       &p_Sensor_Frente_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[3].motorE = PM;   regra[3].motorD = NM;   regra[3].pertinencias = perti_r3;

regra[4].n_conjuntos = 4;
double * perti_r4[4] = {&p_Sensor_Frente_Esq[nMP],&p_Sensor_Frente[MP],
                        &p_Sensor_Frente_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[4].motorE = NM;   regra[4].motorD = PM;   regra[4].pertinencias = perti_r4;

regra[5].n_conjuntos = 4;
double * perti_r5[4] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
```

```
                        &p_Sensor_Frente_Dir[P],&p_Dist_Objetivo[nMP],};
regra[5].motorE = NM;    regra[5].motorD = PM;    regra[5].pertinencias = perti_r5;

regra[6].n_conjuntos = 4;
double * perti_r6[4] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                        &p_Sensor_Frente_Dir[P],&p_Dist_Objetivo[nMP],};
regra[6].motorE = NM;    regra[6].motorD = PM;    regra[6].pertinencias = perti_r6;

regra[7].n_conjuntos = 4;
double * perti_r7[4] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                        &p_Sensor_Frente_Dir[L],&p_Dist_Objetivo[nMP],};
regra[7].motorE = NM;    regra[7].motorD = PM;    regra[7].pertinencias = perti_r7;

regra[8].n_conjuntos = 4;
double * perti_r8[4] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[P],
                        &p_Sensor_Frente_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[8].motorE = PA;    regra[8].motorD = NM;    regra[8].pertinencias = perti_r8;

regra[9].n_conjuntos = 4;
double * perti_r9[4] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                        &p_Sensor_Frente_Dir[L],&p_Dist_Objetivo[nMP],};
regra[9].motorE = PA;    regra[9].motorD = NM;    regra[9].pertinencias = perti_r9;

regra[10].n_conjuntos = 4;
double * perti_r10[4] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[10].motorE = PA;   regra[10].motorD = NM;   regra[10].pertinencias = perti_r10;

regra[11].n_conjuntos = 4;
double * perti_r11[4] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[11].motorE = PA;   regra[11].motorD = NM;   regra[11].pertinencias = perti_r11;

regra[12].n_conjuntos = 4;
double * perti_r12[4] = {&p_Sensor_Frente_Esq[nMP],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[12].motorE = NM;   regra[12].motorD = PA;   regra[12].pertinencias = perti_r12;

regra[13].n_conjuntos = 4;
double * perti_r13[4] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Dist_Objetivo[nMP],};
regra[13].motorE = NM;   regra[13].motorD = PA;   regra[13].pertinencias = perti_r13;

regra[14].n_conjuntos = 4;
double * perti_r14[4] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Dist_Objetivo[nMP],};
regra[14].motorE = NM;   regra[14].motorD = PA;   regra[14].pertinencias = perti_r14;

regra[15].n_conjuntos = 4;
double * perti_r15[4] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[L],&p_Dist_Objetivo[nMP],};
regra[15].motorE = NM;   regra[15].motorD = PA;   regra[15].pertinencias = perti_r15;

regra[16].n_conjuntos = 4;
double * perti_r16[4] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[16].motorE = PA;   regra[16].motorD = NM;   regra[16].pertinencias = perti_r16;

regra[17].n_conjuntos = 4;
double * perti_r17[4] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[L],&p_Dist_Objetivo[nMP],};
regra[17].motorE = PA;   regra[17].motorD = Z;    regra[17].pertinencias = perti_r17;

regra[18].n_conjuntos = 4;
```

```
double * perti_r18[4] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[18].motorE = PA;  regra[18].motorD = Z;   regra[18].pertinencias = perti_r18;

regra[19].n_conjuntos = 4;
double * perti_r19[4] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[19].motorE = PA;  regra[19].motorD = Z;   regra[19].pertinencias = perti_r19;

regra[20].n_conjuntos = 4;
double * perti_r20[4] = {&p_Sensor_Frente_Esq[nMP],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[20].motorE = NM;  regra[20].motorD = PA;  regra[20].pertinencias = perti_r20;

regra[21].n_conjuntos = 4;
double * perti_r21[4] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[P],&p_Dist_Objetivo[nMP],};
regra[21].motorE = Z;   regra[21].motorD = PA;  regra[21].pertinencias = perti_r21;

regra[22].n_conjuntos = 4;
double * perti_r22[4] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[P],&p_Dist_Objetivo[nMP],};
regra[22].motorE = Z;   regra[22].motorD = PA;  regra[22].pertinencias = perti_r22;

regra[23].n_conjuntos = 4;
double * perti_r23[4] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[L],&p_Dist_Objetivo[nMP],};
regra[23].motorE = Z;   regra[23].motorD = PA;  regra[23].pertinencias = perti_r23;

regra[24].n_conjuntos = 4;
double * perti_r24[4] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[ML],
                         &p_Sensor_Frente_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[24].motorE = PA;  regra[24].motorD = NM;  regra[24].pertinencias = perti_r24;

regra[25].n_conjuntos = 4;
double * perti_r25[4] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[ML],
                         &p_Sensor_Frente_Dir[L],&p_Dist_Objetivo[nMP],};
regra[25].motorE = PA;  regra[25].motorD = Z;   regra[25].pertinencias = perti_r25;

regra[26].n_conjuntos = 4;
double * perti_r26[4] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[ML],
                         &p_Sensor_Frente_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[26].motorE = PA;  regra[26].motorD = Z;   regra[26].pertinencias = perti_r26;

regra[27].n_conjuntos = 4;
double * perti_r27[4] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[ML],
                         &p_Sensor_Frente_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[27].motorE = PM;  regra[27].motorD = Z;   regra[27].pertinencias = perti_r27;

regra[28].n_conjuntos = 4;
double * perti_r28[4] = {&p_Sensor_Frente_Esq[nMP],&p_Sensor_Frente[ML],
                         &p_Sensor_Frente_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[28].motorE = NM;  regra[28].motorD = PA;  regra[28].pertinencias = perti_r28;

regra[29].n_conjuntos = 4;
double * perti_r29[4] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[ML],
                         &p_Sensor_Frente_Dir[P],&p_Dist_Objetivo[nMP],};
regra[29].motorE = Z;   regra[29].motorD = PA;  regra[29].pertinencias = perti_r29;

regra[30].n_conjuntos = 4;
double * perti_r30[4] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[ML],
                         &p_Sensor_Frente_Dir[P],&p_Dist_Objetivo[nMP],};
regra[30].motorE = Z;   regra[30].motorD = PA;  regra[30].pertinencias = perti_r30;
```

```
regra[31].n_conjuntos = 4;
double * perti_r31[4] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[ML],
                         &p_Sensor_Frente_Dir[L],&p_Dist_Objetivo[nMP],};
regra[31].motorE = Z;   regra[31].motorD = PM;  regra[31].pertinencias = perti_r31;

regra[32].n_conjuntos = 6;
double * perti_r32[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[MP],
                         &p_Sensor_Lateral_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[32].motorE = PM;  regra[32].motorD = NM;  regra[32].pertinencias = perti_r32;

regra[33].n_conjuntos = 6;
double * perti_r33[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[33].motorE = PM;  regra[33].motorD = NM;  regra[33].pertinencias = perti_r33;

regra[34].n_conjuntos = 6;
double * perti_r34[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[34].motorE = PM;  regra[34].motorD = NM;  regra[34].pertinencias = perti_r34;

regra[35].n_conjuntos = 6;
double * perti_r35[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[L],
                         &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[35].motorE = PM;  regra[35].motorD = NM;  regra[35].pertinencias = perti_r35;

regra[36].n_conjuntos = 6;
double * perti_r36[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[nMP],
                         &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[36].motorE = NM;  regra[36].motorD = PM;  regra[36].pertinencias = perti_r36;

regra[37].n_conjuntos = 6;
double * perti_r37[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[L],
                         &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[37].motorE = NM;  regra[37].motorD = PM;  regra[37].pertinencias = perti_r37;

regra[38].n_conjuntos = 6;
double * perti_r38[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[38].motorE = NM;  regra[38].motorD = PM;  regra[38].pertinencias = perti_r38;

regra[39].n_conjuntos = 6;
double * perti_r39[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[39].motorE = NM;  regra[39].motorD = PM;  regra[39].pertinencias = perti_r39;

regra[40].n_conjuntos = 6;
double * perti_r40[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[MP],
                         &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[40].motorE = PA;  regra[40].motorD = NA;  regra[40].pertinencias = perti_r40;

regra[41].n_conjuntos = 6;
double * perti_r41[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[41].motorE = PA;  regra[41].motorD = NA;  regra[41].pertinencias = perti_r41;
```

```
regra[42].n_conjuntos = 6;
double * perti_r42[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[L],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[42].motorE = PA;  regra[42].motorD = NA;  regra[42].pertinencias = perti_r42;

regra[43].n_conjuntos = 6;
double * perti_r43[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[43].motorE = PA;  regra[43].motorD = NA;  regra[43].pertinencias = perti_r43;

regra[44].n_conjuntos = 6;
double * perti_r44[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[MP],
                         &p_Sensor_Lateral_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[44].motorE = PM;  regra[44].motorD = Z;   regra[44].pertinencias = perti_r44;

regra[45].n_conjuntos = 6;
double * perti_r45[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[45].motorE = PA;  regra[45].motorD = NM;  regra[45].pertinencias = perti_r45;

regra[46].n_conjuntos = 6;
double * perti_r46[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[46].motorE = PA;  regra[46].motorD = NM;  regra[46].pertinencias = perti_r46;

regra[47].n_conjuntos = 6;
double * perti_r47[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[L],
                         &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[47].motorE = PA;  regra[47].motorD = NM;  regra[47].pertinencias = perti_r47;

regra[48].n_conjuntos = 6;
double * perti_r48[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[nMP],
                         &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[48].motorE = Z;   regra[48].motorD = PM;  regra[48].pertinencias = perti_r48;

regra[49].n_conjuntos = 6;
double * perti_r49[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[L],
                         &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[49].motorE = NM;  regra[49].motorD = PA;  regra[49].pertinencias = perti_r49;

regra[50].n_conjuntos = 6;
double * perti_r50[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[50].motorE = NM;  regra[50].motorD = PA;  regra[50].pertinencias = perti_r50;

regra[51].n_conjuntos = 6;
double * perti_r51[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[51].motorE = NM;  regra[51].motorD = PA;  regra[51].pertinencias = perti_r51;

regra[52].n_conjuntos = 6;
double * perti_r52[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[MP],
```

```
                        &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[52].motorE = PA;  regra[52].motorD = NA; regra[52].pertinencias = perti_r52;

regra[53].n_conjuntos = 6;
double * perti_r53[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                        &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[P],
                        &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[53].motorE = PA;  regra[53].motorD = NA;  regra[53].pertinencias = perti_r53;

regra[54].n_conjuntos = 6;
double * perti_r54[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                        &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[L],
                        &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[54].motorE = PA;  regra[54].motorD = NA;  regra[54].pertinencias = perti_r54;

regra[55].n_conjuntos = 6;
double * perti_r55[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[P],
                        &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[ML],
                        &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[55].motorE = PA;  regra[55].motorD = NA;  regra[55].pertinencias = perti_r55;

regra[56].n_conjuntos = 6;
double * perti_r56[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[MP],
                        &p_Sensor_Lateral_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[56].motorE = PM;  regra[56].motorD = Z;   regra[56].pertinencias = perti_r56;

regra[57].n_conjuntos = 6;
double * perti_r57[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[P],
                        &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[57].motorE = PM;  regra[57].motorD = Z;   regra[57].pertinencias = perti_r57;

regra[58].n_conjuntos = 6;
double * perti_r58[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[P],
                        &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[58].motorE = PM;  regra[58].motorD = Z;   regra[58].pertinencias = perti_r58;

regra[59].n_conjuntos = 6;
double * perti_r59[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[L],
                        &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[59].motorE = PM;  regra[59].motorD = Z;   regra[59].pertinencias = perti_r59;

regra[60].n_conjuntos = 6;
double * perti_r60[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[nMP],
                        &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[60].motorE = Z;   regra[60].motorD = PM;  regra[60].pertinencias = perti_r60;

regra[61].n_conjuntos = 6;
double * perti_r61[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[L],
                        &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[61].motorE = Z;   regra[61].motorD = PM;  regra[61].pertinencias = perti_r61;

regra[62].n_conjuntos = 6;
double * perti_r62[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[ML],
                        &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[62].motorE = Z;   regra[62].motorD = PM;  regra[62].pertinencias = perti_r62;

regra[63].n_conjuntos = 6;
```

```
double * perti_r63[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[63].motorE = Z;    regra[63].motorD = PM;  regra[63].pertinencias = perti_r63;

regra[64].n_conjuntos = 6;
double * perti_r64[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[MP],
                         &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[64].motorE = PM;   regra[64].motorD = PM;  regra[64].pertinencias = perti_r64;

regra[65].n_conjuntos = 6;
double * perti_r65[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[65].motorE = PM;   regra[65].motorD = PM;  regra[65].pertinencias = perti_r65;

regra[66].n_conjuntos = 6;
double * perti_r66[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[L],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[66].motorE = PM;   regra[66].motorD = PM;  regra[66].pertinencias = perti_r66;

regra[67].n_conjuntos = 6;
double * perti_r67[6] = {&p_Sensor_Frente_Esq[L],&p_Sensor_Frente[L]
                         ,&p_Sensor_Frente_Dir[L],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[67].motorE = PM;   regra[67].motorD = PM;  regra[67].pertinencias = perti_r67;

regra[68].n_conjuntos = 5;
double * perti_r68[5] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[L],
                         &p_Sensor_Frente[ML],&p_Sensor_Frente_Dir[L],&p_Dist_Objetivo[nMP],};
regra[68].motorE = PM;   regra[68].motorD = PM;  regra[68].pertinencias = perti_r68;

regra[69].n_conjuntos = 6;
double * perti_r69[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[MP],
                         &p_Sensor_Lateral_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[69].motorE = PM;   regra[69].motorD = NM;  regra[69].pertinencias = perti_r69;

regra[70].n_conjuntos = 6;
double * perti_r70[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[70].motorE = PM;   regra[70].motorD = NM;  regra[70].pertinencias = perti_r70;

regra[71].n_conjuntos = 6;
double * perti_r71[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[71].motorE = PM;   regra[71].motorD = NM;  regra[71].pertinencias = perti_r71;

regra[72].n_conjuntos = 6;
double * perti_r72[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[L],
                         &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[72].motorE = PM;   regra[72].motorD = NM;  regra[72].pertinencias = perti_r72;

regra[73].n_conjuntos = 6;
double * perti_r73[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[nMP],
                         &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[73].motorE = NM;   regra[73].motorD = PM;  regra[73].pertinencias = perti_r73;
```

```
regra[74].n_conjuntos = 6;
double * perti_r74[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[L],
                         &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[74].motorE = NM;  regra[74].motorD = PM;  regra[74].pertinencias = perti_r74;

regra[75].n_conjuntos = 6;
double * perti_r75[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[75].motorE = NM;  regra[75].motorD = PM;  regra[75].pertinencias = perti_r75;

regra[76].n_conjuntos = 6;
double * perti_r76[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[76].motorE = NM;  regra[76].motorD = PM;  regra[76].pertinencias = perti_r76;

regra[77].n_conjuntos = 6;
double * perti_r77[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[MP],
                         &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[77].motorE = PA;  regra[77].motorD = NA;  regra[77].pertinencias = perti_r77;

regra[78].n_conjuntos = 6;
double * perti_r78[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[78].motorE = PA;  regra[78].motorD = NA;  regra[78].pertinencias = perti_r78;

regra[79].n_conjuntos = 6;
double * perti_r79[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[L],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[79].motorE = PA;  regra[79].motorD = NA;  regra[79].pertinencias = perti_r79;

regra[80].n_conjuntos = 6;
double * perti_r80[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[MP],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[80].motorE = PA;  regra[80].motorD = NA;  regra[80].pertinencias = perti_r80;

regra[81].n_conjuntos = 6;
double * perti_r81[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[MP],
                         &p_Sensor_Lateral_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[81].motorE = PM;  regra[81].motorD = Z;   regra[81].pertinencias = perti_r81;

regra[82].n_conjuntos = 6;
double * perti_r82[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[82].motorE = PM;  regra[82].motorD = Z;   regra[82].pertinencias = perti_r82;

regra[83].n_conjuntos = 6;
double * perti_r83[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[83].motorE = PM;  regra[83].motorD = Z;   regra[83].pertinencias = perti_r83;

regra[84].n_conjuntos = 6;
double * perti_r84[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[L],
                         &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
```

```
regra[84].motorE = PM;  regra[84].motorD = Z;   regra[84].pertinencias = perti_r84;

regra[85].n_conjuntos = 6;
double * perti_r85[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[nMP],
                         &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[MP],};
regra[85].motorE = Z;   regra[85].motorD = PM;  regra[85].pertinencias = perti_r85;

regra[86].n_conjuntos = 6;
double * perti_r86[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[L],
                         &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[86].motorE = Z;   regra[86].motorD = PM;  regra[86].pertinencias = perti_r86;

regra[87].n_conjuntos = 6;
double * perti_r87[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[87].motorE = Z;   regra[87].motorD = PM;  regra[87].pertinencias = perti_r87;

regra[88].n_conjuntos = 6;
double * perti_r88[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[88].motorE = Z;   regra[88].motorD = PM;  regra[88].pertinencias = perti_r88;

regra[89].n_conjuntos = 6;
double * perti_r89[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[MP],
                         &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[89].motorE = PA;  regra[89].motorD = NM;  regra[89].pertinencias = perti_r89;

regra[90].n_conjuntos = 6;
double * perti_r90[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[90].motorE = PA;  regra[90].motorD = NM;  regra[90].pertinencias = perti_r90;

regra[91].n_conjuntos = 6;
double * perti_r91[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[L],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[91].motorE = PA;  regra[91].motorD = NM;  regra[91].pertinencias = perti_r91;

regra[92].n_conjuntos = 6;
double * perti_r92[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[P],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[92].motorE = PA;  regra[92].motorD = NM;  regra[92].pertinencias = perti_r92;

regra[93].n_conjuntos = 6;
double * perti_r93[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[MP],
                         &p_Sensor_Lateral_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[93].motorE = PM;  regra[93].motorD = Z;   regra[93].pertinencias = perti_r93;

regra[94].n_conjuntos = 6;
double * perti_r94[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[94].motorE = PA;  regra[94].motorD = PM;  regra[94].pertinencias = perti_r94;

regra[95].n_conjuntos = 6;
double * perti_r95[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
```

```
                        &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[P],
                        &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[95].motorE = PA;  regra[95].motorD = PM;  regra[95].pertinencias = perti_r95;

regra[96].n_conjuntos = 6;
double * perti_r96[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[L],
                        &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[96].motorE = PA;  regra[96].motorD = PM;  regra[96].pertinencias = perti_r96;

regra[97].n_conjuntos = 6;
double * perti_r97[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[nMP],
                        &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[97].motorE = Z;   regra[97].motorD = PM;  regra[97].pertinencias = perti_r97;

regra[98].n_conjuntos = 6;
double * perti_r98[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[L],
                        &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[98].motorE = PM;  regra[98].motorD = PA;  regra[98].pertinencias = perti_r98;

regra[99].n_conjuntos = 6;
double * perti_r99[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[ML],
                        &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[99].motorE = PM;  regra[99].motorD = PA;  regra[99].pertinencias = perti_r99;

regra[100].n_conjuntos = 6;
double * perti_r100[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[100].motorE = PM;     regra[100].motorD = PA;     regra[100].pertinencias = perti_r100;

regra[101].n_conjuntos = 6;
double * perti_r101[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[MP],
                         &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[101].motorE = PM;     regra[101].motorD = PM;     regra[101].pertinencias = perti_r101;

regra[102].n_conjuntos = 6;
double * perti_r102[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[P],
                         &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[102].motorE = PM;     regra[102].motorD = PM;     regra[102].pertinencias = perti_r102;

regra[103].n_conjuntos = 6;
double * perti_r103[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[L],
                         &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[103].motorE = PA;     regra[103].motorD = PM;     regra[103].pertinencias = perti_r103;

regra[104].n_conjuntos = 6;
double * perti_r104[6] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[L],
                         &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Esq[ML],
                         &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[104].motorE = PA;     regra[104].motorD = PM;     regra[104].pertinencias = perti_r104;

regra[105].n_conjuntos = 4;
double * perti_r105[4] = {&p_Sensor_Frente_Esq[P],&p_Sensor_Frente[ML],
                         &p_Sensor_Frente_Dir[P],&p_Dist_Objetivo[nMP],};
regra[105].motorE = PM;     regra[105].motorD = PM;     regra[105].pertinencias = perti_r105;

regra[106].n_conjuntos = 6;
```

```
double * perti_r106[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[MP],
                          &p_Sensor_Lateral_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[106].motorE = PM;     regra[106].motorD = NM;     regra[106].pertinencias = perti_r106;

regra[107].n_conjuntos = 6;
double * perti_r107[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[P],
                          &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[107].motorE = PM;     regra[107].motorD = NM;     regra[107].pertinencias = perti_r107;

regra[108].n_conjuntos = 6;
double * perti_r108[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[P],
                          &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[108].motorE = PM;     regra[108].motorD = NM;     regra[108].pertinencias = perti_r108;

regra[109].n_conjuntos = 6;
double * perti_r109[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[L],
                          &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[109].motorE = PM;     regra[109].motorD = NM;     regra[109].pertinencias = perti_r109;

regra[110].n_conjuntos = 6;
double * perti_r110[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[nMP],
                          &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[110].motorE = NM;     regra[110].motorD = PM;     regra[110].pertinencias = perti_r110;

regra[111].n_conjuntos = 6;
double * perti_r111[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[L],
                          &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[111].motorE = NM;     regra[111].motorD = PM;     regra[111].pertinencias = perti_r111;

regra[112].n_conjuntos = 6;
double * perti_r112[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[ML],
                          &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[112].motorE = NM;     regra[112].motorD = PM;     regra[112].pertinencias = perti_r112;

regra[113].n_conjuntos = 6;
double * perti_r113[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[ML],
                          &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[113].motorE = NM;     regra[113].motorD = PM;     regra[113].pertinencias = perti_r113;

regra[114].n_conjuntos = 6;
double * perti_r114[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[MP],
                          &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[114].motorE = PA;     regra[114].motorD = NA;     regra[114].pertinencias = perti_r114;

regra[115].n_conjuntos = 6;
double * perti_r115[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[P],
                          &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[115].motorE = PA;     regra[115].motorD = NA;     regra[115].pertinencias = perti_r115;

regra[116].n_conjuntos = 6;
double * perti_r116[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[L],
                          &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[116].motorE = PA;     regra[116].motorD = NA;     regra[116].pertinencias = perti_r116;
```

```
regra[117].n_conjuntos = 6;
double * perti_r117[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[ML],
                          &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[117].motorE = PA;     regra[117].motorD = NA;     regra[117].pertinencias = perti_r117;

regra[118].n_conjuntos = 6;
double * perti_r118[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[P],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[MP],
                          &p_Sensor_Lateral_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[118].motorE = PM;     regra[118].motorD = NM;     regra[118].pertinencias = perti_r118;

regra[119].n_conjuntos = 6;
double * perti_r119[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[P],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[P],
                          &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[119].motorE = PM;     regra[119].motorD = NM;     regra[119].pertinencias = perti_r119;

regra[120].n_conjuntos = 6;
double * perti_r120[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[P],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[P],
                          &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[120].motorE = PM;     regra[120].motorD = NM;     regra[120].pertinencias = perti_r120;

regra[121].n_conjuntos = 6;
double * perti_r121[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[P],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[L],
                          &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[121].motorE = PM;     regra[121].motorD = NM;     regra[121].pertinencias = perti_r121;

regra[122].n_conjuntos = 6;
double * perti_r122[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[P],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[nMP],
                          &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[MP],};
regra[122].motorE = NM;     regra[122].motorD = PM;     regra[122].pertinencias = perti_r122;

regra[123].n_conjuntos = 6;
double * perti_r123[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[P],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[L],
                          &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[123].motorE = NM;     regra[123].motorD = PM;     regra[123].pertinencias = perti_r123;

regra[124].n_conjuntos = 6;
double * perti_r124[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[P],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[ML],
                          &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[124].motorE = NM;     regra[124].motorD = PM;     regra[124].pertinencias = perti_r124;

regra[125].n_conjuntos = 6;
double * perti_r125[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[P],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[ML],
                          &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[125].motorE = NM;     regra[125].motorD = PM;     regra[125].pertinencias = perti_r125;

regra[126].n_conjuntos = 6;
double * perti_r126[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[P],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[MP],
                          &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[126].motorE = PM;     regra[126].motorD = NA;     regra[126].pertinencias = perti_r126;

regra[127].n_conjuntos = 6;
double * perti_r127[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[P],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[P],
```

```
                        &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[127].motorE = PM;     regra[127].motorD = NA;     regra[127].pertinencias = perti_r127;

regra[128].n_conjuntos = 6;
double * perti_r128[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[P],
                        &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[L],
                        &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[128].motorE = PM;     regra[128].motorD = NA;     regra[128].pertinencias = perti_r128;

regra[129].n_conjuntos = 6;
double * perti_r129[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[P],
                        &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[ML],
                        &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[129].motorE = PM;     regra[129].motorD = NA;     regra[129].pertinencias = perti_r129;

regra[130].n_conjuntos = 6;
double * perti_r130[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[MP],
                        &p_Sensor_Lateral_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[130].motorE = PM;     regra[130].motorD = Z;  regra[130].pertinencias = perti_r130;

regra[131].n_conjuntos = 6;
double * perti_r131[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[P],
                        &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[131].motorE = PM;     regra[131].motorD = Z;  regra[131].pertinencias = perti_r131;

regra[132].n_conjuntos = 6;
double * perti_r132[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[P],
                        &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[132].motorE = PM;     regra[132].motorD = Z;  regra[132].pertinencias = perti_r132;

regra[133].n_conjuntos = 6;
double * perti_r133[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[L],
                        &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[133].motorE = PM;     regra[133].motorD = Z;  regra[133].pertinencias = perti_r133;

regra[134].n_conjuntos = 6;
double * perti_r134[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[nMP],
                        &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[134].motorE = Z;  regra[134].motorD = PM;     regra[134].pertinencias = perti_r134;

regra[135].n_conjuntos = 6;
double * perti_r135[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[L],
                        &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};

regra[135].motorE = Z;  regra[135].motorD = PM;     regra[135].pertinencias = perti_r135;

regra[136].n_conjuntos = 6;
double * perti_r136[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[ML],
                        &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[136].motorE = Z;  regra[136].motorD = PM;     regra[136].pertinencias = perti_r136;

regra[137].n_conjuntos = 6;
double * perti_r137[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[L],
                        &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[ML],
                        &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[137].motorE = Z;  regra[137].motorD = PM;     regra[137].pertinencias = perti_r137;
```

```
regra[138].n_conjuntos = 6;
double * perti_r138[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[L],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[MP],
                          &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[138].motorE = PA;     regra[138].motorD = NA;     regra[138].pertinencias = perti_r138;

regra[139].n_conjuntos = 6;
double * perti_r139[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[L],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[P],
                          &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[139].motorE = PA;     regra[139].motorD = NA;     regra[139].pertinencias = perti_r139;

regra[140].n_conjuntos = 6;
double * perti_r140[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[L],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[L],
                          &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[140].motorE = PA;     regra[140].motorD = NA;     regra[140].pertinencias = perti_r140;

regra[141].n_conjuntos = 6;
double * perti_r141[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[L],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[ML],
                          &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[141].motorE = PA;     regra[141].motorD = NA;     regra[141].pertinencias = perti_r141;

regra[142].n_conjuntos = 6;
double * perti_r142[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[ML],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[MP],
                          &p_Sensor_Lateral_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[142].motorE = PM;     regra[142].motorD = Z;  regra[142].pertinencias = perti_r142;

regra[143].n_conjuntos = 6;
double * perti_r143[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[ML],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[P],
                          &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[143].motorE = PM;     regra[143].motorD = Z;  regra[143].pertinencias = perti_r143;

regra[144].n_conjuntos = 6;
double * perti_r144[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[ML],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[P],
                          &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[144].motorE = PM;     regra[144].motorD = Z;  regra[144].pertinencias = perti_r144;

regra[145].n_conjuntos = 6;
double * perti_r145[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[ML],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[L],
                          &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[145].motorE = PM;     regra[145].motorD = Z;  regra[145].pertinencias = perti_r145;

regra[146].n_conjuntos = 6;
double * perti_r146[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[ML],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[nMP],
                          &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[146].motorE = Z;  regra[146].motorD = PM;     regra[146].pertinencias = perti_r146;

regra[147].n_conjuntos = 6;
double * perti_r147[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[ML],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[L],
regra[147].motorE = Z;  regra[147].motorD = PM;     regra[147].pertinencias = perti_r147;

regra[148].n_conjuntos = 6;
double * perti_r148[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[ML],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[ML],
                          &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[148].motorE = Z;  regra[148].motorD = PM;     regra[148].pertinencias = perti_r148;
```

```
regra[149].n_conjuntos = 6;
double * perti_r149[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[ML],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[ML],
                          &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[149].motorE = Z;  regra[149].motorD = PM;     regra[149].pertinencias = perti_r149;

regra[150].n_conjuntos = 6;
double * perti_r150[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[ML],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[MP],
                          &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[150].motorE = PM;     regra[150].motorD = NM;     regra[150].pertinencias = perti_r150;

regra[151].n_conjuntos = 6;
double * perti_r151[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[ML],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[P],
                          &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[151].motorE = PM;     regra[151].motorD = NM;     regra[151].pertinencias = perti_r151;

regra[152].n_conjuntos = 6;
double * perti_r152[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[ML],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[L],
                          &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[152].motorE = PM;     regra[152].motorD = NM;     regra[152].pertinencias = perti_r152;

regra[153].n_conjuntos = 6;
double * perti_r153[6] = {&p_Sensor_Frente_Esq[MP],&p_Sensor_Frente[ML],
                          &p_Sensor_Frente_Dir[MP],&p_Sensor_Lateral_Esq[ML],
                          &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[153].motorE = PM;     regra[153].motorD = NM;     regra[153].pertinencias = perti_r153;

regra[154].n_conjuntos = 6;
double * perti_r154[6] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[ML],&p_Sensor_Lateral_Esq[MP],
                          &p_Sensor_Lateral_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[154].motorE = PM;     regra[154].motorD = NM;     regra[154].pertinencias = perti_r154;

regra[155].n_conjuntos = 6;
double * perti_r155[6] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[ML],&p_Sensor_Lateral_Esq[P],
                          &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[155].motorE = PM;     regra[155].motorD = NM;     regra[155].pertinencias = perti_r155;

regra[156].n_conjuntos = 6;
double * perti_r156[6] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[ML],&p_Sensor_Lateral_Esq[P],
                          &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[156].motorE = PM;     regra[156].motorD = NM;     regra[156].pertinencias = perti_r156;

regra[157].n_conjuntos = 6;
double * perti_r157[6] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[ML],&p_Sensor_Lateral_Esq[L],
                          &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[157].motorE = PM;     regra[157].motorD = NM;     regra[157].pertinencias = perti_r157;

regra[158].n_conjuntos = 6;
double * perti_r158[6] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[ML],&p_Sensor_Lateral_Esq[nMP],
                          &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[158].motorE = NM;     regra[158].motorD = PM;     regra[158].pertinencias = perti_r158;

regra[159].n_conjuntos = 6;
double * perti_r159[6] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                          &p_Sensor_Frente_Dir[ML],&p_Sensor_Lateral_Esq[L],
```

```
                        &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[159].motorE = NM;     regra[159].motorD = PM;     regra[159].pertinencias = perti_r159;

regra[160].n_conjuntos = 6;
double * perti_r160[6] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                        &p_Sensor_Frente_Dir[ML],&p_Sensor_Lateral_Esq[ML],
                        &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[160].motorE = NM;     regra[160].motorD = PM;     regra[160].pertinencias = perti_r160;

regra[161].n_conjuntos = 6;
double * perti_r161[6] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                        &p_Sensor_Frente_Dir[ML],&p_Sensor_Lateral_Esq[ML],
                        &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[161].motorE = NM;     regra[161].motorD = PM;     regra[161].pertinencias = perti_r161;

regra[162].n_conjuntos = 6;
double * perti_r162[6] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                        &p_Sensor_Frente_Dir[ML],&p_Sensor_Lateral_Esq[MP],
                        &p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[162].motorE = PA;     regra[162].motorD = NA;     regra[162].pertinencias = perti_r162;

regra[163].n_conjuntos = 6;
double * perti_r163[6] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                        &p_Sensor_Frente_Dir[ML],&p_Sensor_Lateral_Esq[P],
                        &p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[163].motorE = PA;     regra[163].motorD = NA;     regra[163].pertinencias = perti_r163;

regra[164].n_conjuntos = 6;
double * perti_r164[6] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                        &p_Sensor_Frente_Dir[ML],&p_Sensor_Lateral_Esq[L],
                        &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[164].motorE = PA;     regra[164].motorD = NA;     regra[164].pertinencias = perti_r164;

regra[165].n_conjuntos = 6;
double * perti_r165[6] = {&p_Sensor_Frente_Esq[ML],&p_Sensor_Frente[MP],
                        &p_Sensor_Frente_Dir[ML],&p_Sensor_Lateral_Esq[ML],
                        &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[165].motorE = PA;     regra[165].motorD = NA;     regra[165].pertinencias = perti_r165;

regra[166].n_conjuntos = 7;
double * perti_r166[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                        &p_Sensor_Frente[P],&p_Sensor_Frente_Dir[ML],
                        &p_Sensor_Lateral_Esq[MP],&p_Sensor_Lateral_Dir[nMP],
                        &p_Dist_Objetivo[nMP],};
regra[166].motorE = PA;     regra[166].motorD = NM;     regra[166].pertinencias = perti_r166;

regra[167].n_conjuntos = 7;
double * perti_r167[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                        &p_Sensor_Frente[P],&p_Sensor_Frente_Dir[ML],
                        &p_Sensor_Lateral_Esq[P],&p_Sensor_Lateral_Dir[L],
                        &p_Dist_Objetivo[nMP],};
regra[167].motorE = PA;     regra[167].motorD = NM;     regra[167].pertinencias = perti_r167;

regra[168].n_conjuntos = 7;
double * perti_r168[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                        &p_Sensor_Frente[P],&p_Sensor_Frente_Dir[ML],
                        &p_Sensor_Lateral_Esq[P],&p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[168].motorE = PA;     regra[168].motorD = NM;     regra[168].pertinencias = perti_r168;

regra[169].n_conjuntos = 7;
double * perti_r169[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                        &p_Sensor_Frente[P],&p_Sensor_Frente_Dir[ML],
                    &p_Sensor_Lateral_Esq[L],&p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[169].motorE = PA;     regra[169].motorD = NM;     regra[169].pertinencias = perti_r169;
```

```
regra[170].n_conjuntos = 7;
double * perti_r170[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                          &p_Sensor_Frente[P],&p_Sensor_Frente_Dir[ML],
                      &p_Sensor_Lateral_Esq[nMP],&p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[170].motorE = NM;     regra[170].motorD = PA;     regra[170].pertinencias = perti_r170;

regra[171].n_conjuntos = 7;
double * perti_r171[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                          &p_Sensor_Frente[P],&p_Sensor_Frente_Dir[ML],
                        &p_Sensor_Lateral_Esq[L],&p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[171].motorE = NM;     regra[171].motorD = PA;     regra[171].pertinencias = perti_r171;

regra[172].n_conjuntos = 7;
double * perti_r172[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                          &p_Sensor_Frente[P],&p_Sensor_Frente_Dir[ML],
                       &p_Sensor_Lateral_Esq[ML],&p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[172].motorE = NM;     regra[172].motorD = PA;     regra[172].pertinencias = perti_r172;

regra[173].n_conjuntos = 7;
double * perti_r173[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                          &p_Sensor_Frente[P],&p_Sensor_Frente_Dir[ML],
                     &p_Sensor_Lateral_Esq[ML],&p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[173].motorE = NM;     regra[173].motorD = PA;     regra[173].pertinencias = perti_r173;

regra[174].n_conjuntos = 7;
double * perti_r174[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                          &p_Sensor_Frente[P],&p_Sensor_Frente_Dir[ML],
                      &p_Sensor_Lateral_Esq[MP],&p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[174].motorE = PA;     regra[174].motorD = NA;     regra[174].pertinencias = perti_r174;

regra[175].n_conjuntos = 7;
double * perti_r175[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                          &p_Sensor_Frente[P],&p_Sensor_Frente_Dir[ML],
                          &p_Sensor_Lateral_Esq[P],&p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[175].motorE = PA;     regra[175].motorD = NA;     regra[175].pertinencias = perti_r175;

regra[176].n_conjuntos = 7;
double * perti_r176[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                          &p_Sensor_Frente[P],&p_Sensor_Frente_Dir[ML],
                          &p_Sensor_Lateral_Esq[L],&p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[176].motorE = PA;     regra[176].motorD = NA;     regra[176].pertinencias = perti_r176;

regra[177].n_conjuntos = 7;
double * perti_r177[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                          &p_Sensor_Frente[P],&p_Sensor_Frente_Dir[ML],
                          &p_Sensor_Lateral_Esq[ML],&p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],
regra[177].motorE = PA;     regra[177].motorD = NA;     regra[177].pertinencias = perti_r177;

regra[178].n_conjuntos = 7;
double * perti_r178[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                          &p_Sensor_Frente[L],&p_Sensor_Frente_Dir[ML],
                        &p_Sensor_Lateral_Esq[MP],&p_Sensor_Lateral_Dir[nMP],&p_Dist_Objetivo[nMP],};
regra[178].motorE = PM;     regra[178].motorD = Z;  regra[178].pertinencias = perti_r178;

regra[179].n_conjuntos = 7;
double * perti_r179[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                          &p_Sensor_Frente[L],&p_Sensor_Frente_Dir[ML],
                          &p_Sensor_Lateral_Esq[P],&p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[179].motorE = PM;     regra[179].motorD = Z;  regra[179].pertinencias = perti_r179;

regra[180].n_conjuntos = 7;
double * perti_r180[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                          &p_Sensor_Frente[L],&p_Sensor_Frente_Dir[ML],
```

```
                        &p_Sensor_Lateral_Esq[P],&p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[180].motorE = PM;     regra[180].motorD = Z;  regra[180].pertinencias = perti_r180;

regra[181].n_conjuntos = 7;
double * perti_r181[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                        &p_Sensor_Frente[L],&p_Sensor_Frente_Dir[ML],
                        &p_Sensor_Lateral_Esq[L],&p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[181].motorE = PM;     regra[181].motorD = Z;  regra[181].pertinencias = perti_r181;

regra[182].n_conjuntos = 7;
double * perti_r182[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                        &p_Sensor_Frente[L],&p_Sensor_Frente_Dir[ML],
                      &p_Sensor_Lateral_Esq[nMP],&p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[182].motorE = Z;  regra[182].motorD = PM;     regra[182].pertinencias = perti_r182;

regra[183].n_conjuntos = 7;
double * perti_r183[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                        &p_Sensor_Frente[L],&p_Sensor_Frente_Dir[ML],
                        &p_Sensor_Lateral_Esq[L],&p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[183].motorE = Z;  regra[183].motorD = PM;     regra[183].pertinencias = perti_r183;

regra[184].n_conjuntos = 7;
double * perti_r184[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                        &p_Sensor_Frente[L],&p_Sensor_Frente_Dir[ML],
                        &p_Sensor_Lateral_Esq[ML],&p_Sensor_Lateral_Dir[P],
                        &p_Dist_Objetivo[nMP],};
regra[184].motorE = Z;  regra[184].motorD = PM;     regra[184].pertinencias = perti_r184;

regra[185].n_conjuntos = 7;
double * perti_r185[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                        &p_Sensor_Frente[L],&p_Sensor_Frente_Dir[ML],
                       &p_Sensor_Lateral_Esq[ML],&p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[185].motorE = Z;  regra[185].motorD = PM;     regra[185].pertinencias = perti_r185;

regra[186].n_conjuntos = 7;
double * perti_r186[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                        &p_Sensor_Frente[L],&p_Sensor_Frente_Dir[ML],
                      &p_Sensor_Lateral_Esq[MP],&p_Sensor_Lateral_Dir[MP],&p_Dist_Objetivo[nMP],};
regra[186].motorE = PM;     regra[186].motorD = PM;     regra[186].pertinencias = perti_r186;

regra[187].n_conjuntos = 7;
double * perti_r187[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                        &p_Sensor_Frente[L],&p_Sensor_Frente_Dir[ML],
                        &p_Sensor_Lateral_Esq[P],&p_Sensor_Lateral_Dir[P],&p_Dist_Objetivo[nMP],};
regra[187].motorE = PM;     regra[187].motorD = PM;     regra[187].pertinencias = perti_r187;

regra[188].n_conjuntos = 7;
double * perti_r188[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                        &p_Sensor_Frente[L],&p_Sensor_Frente_Dir[ML],
                      &p_Sensor_Lateral_Esq[L],&p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[188].motorE = PM;     regra[188].motorD = PM;     regra[188].pertinencias = perti_r188;

regra[189].n_conjuntos = 7;
double * perti_r189[7] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
                        &p_Sensor_Frente[L],&p_Sensor_Frente_Dir[ML],
                    &p_Sensor_Lateral_Esq[ML],&p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[189].motorE = PM;     regra[189].motorD = PM;     regra[189].pertinencias = perti_r189;

regra[190].n_conjuntos = 2;
double * perti_r190[2] = {&p_Angulo[Z],&p_Dist_Objetivo[MP],};
regra[190].motorE = Z;  regra[190].motorD = Z;  regra[190].pertinencias = perti_r190;

regra[191].n_conjuntos = 5;
double * perti_r191[5] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[ML],
```

```
                        &p_Sensor_Frente[ML],&p_Sensor_Frente_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[191].motorE = PA;     regra[191].motorD = PA;     regra[191].pertinencias = perti_r191;

regra[192].n_conjuntos = 5;
double * perti_r192[5] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[L],
                        &p_Sensor_Frente[ML],&p_Sensor_Frente_Dir[L],&p_Dist_Objetivo[nMP],};
regra[192].motorE = PA;     regra[192].motorD = PA;     regra[192].pertinencias = perti_r192;

regra[193].n_conjuntos = 5;
double * perti_r193[5] = {&p_Angulo[Z],&p_Sensor_Frente_Esq[P],
                        &p_Sensor_Frente[ML],&p_Sensor_Frente_Dir[P],&p_Dist_Objetivo[nMP],};
regra[193].motorE = PM;     regra[193].motorD = PM;     regra[193].pertinencias = perti_r193;

regra[194].n_conjuntos = 2;
double * perti_r194[2] = {&p_Angulo[NM],&p_Dist_Objetivo[MP],};
regra[194].motorE = Z;  regra[194].motorD = Z;  regra[194].pertinencias = perti_r194;

regra[195].n_conjuntos = 4;
double * perti_r195[4] = {&p_Angulo[NM],&p_Sensor_Frente[nMP],
                        &p_Sensor_Frente_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[195].motorE = PA; regra[195].motorD = NM;     regra[195].pertinencias = perti_r195;

regra[196].n_conjuntos = 4;
double * perti_r196[4] = {&p_Angulo[NM],&p_Sensor_Frente[nMP],
                        &p_Sensor_Frente_Dir[L],&p_Dist_Objetivo[nMP],};
regra[196].motorE = PM;     regra[196].motorD = NM;     regra[196].pertinencias = perti_r196;

regra[197].n_conjuntos = 5;
double * perti_r197[5] = {&p_Angulo[NM],&p_Sensor_Frente[nMP],
               &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Dir[nML],&p_Dist_Objetivo[nMP],};
regra[197].motorE = PM;     regra[197].motorD = Z;  regra[197].pertinencias = perti_r197;

regra[198].n_conjuntos = 4;
double * perti_r198[4] = {&p_Angulo[NM],&p_Sensor_Frente[nMP],
                        &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[198].motorE = PA;     regra[198].motorD = NA;     regra[198].pertinencias = perti_r198;

regra[199].n_conjuntos = 4;
double * perti_r199[4] = {&p_Angulo[NM],&p_Sensor_Frente[nMP],
                        &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[199].motorE = PA;     regra[199].motorD = NA;     regra[199].pertinencias = perti_r199;
regra[200].n_conjuntos = 2;

double * perti_r200[2] = {&p_Angulo[NG],&p_Dist_Objetivo[MP],};
regra[200].motorE = Z;  regra[200].motorD = Z;  regra[200].pertinencias = perti_r200;

regra[201].n_conjuntos = 4;
double * perti_r201[4] = {&p_Angulo[NG],&p_Sensor_Frente[nMP],
                        &p_Sensor_Frente_Dir[ML],&p_Dist_Objetivo[nMP],};
regra[201].motorE = PA;     regra[201].motorD = NM;     regra[201].pertinencias = perti_r201;

regra[202].n_conjuntos = 4;
double * perti_r202[4] = {&p_Angulo[NG],&p_Sensor_Frente[nMP],
                        &p_Sensor_Frente_Dir[L],&p_Dist_Objetivo[nMP],};
regra[202].motorE = PA;     regra[202].motorD = NM;     regra[202].pertinencias = perti_r202;

regra[203].n_conjuntos = 5;
double * perti_r203[5] = {&p_Angulo[NG],&p_Sensor_Frente[nMP],
                      &p_Sensor_Frente_Dir[P],&p_Sensor_Lateral_Dir[nML],&p_Dist_Objetivo[nMP],};
regra[203].motorE = PA;     regra[203].motorD = NM;     regra[203].pertinencias = perti_r203;

regra[204].n_conjuntos = 4;
double * perti_r204[4] = {&p_Angulo[NG],&p_Sensor_Frente[nMP],
                        &p_Sensor_Lateral_Dir[ML],&p_Dist_Objetivo[nMP],};
```

```
regra[204].motorE = PA;     regra[204].motorD = NA;     regra[204].pertinencias = perti_r204;

regra[205].n_conjuntos = 4;
double * perti_r205[4] = {&p_Angulo[NG],&p_Sensor_Frente[nMP],
                          &p_Sensor_Lateral_Dir[L],&p_Dist_Objetivo[nMP],};
regra[205].motorE = PA;     regra[205].motorD = NA;     regra[205].pertinencias = perti_r205;

regra[206].n_conjuntos = 2;
double * perti_r206[2] = {&p_Angulo[PM],&p_Dist_Objetivo[MP],};
regra[206].motorE = Z;  regra[206].motorD = Z;  regra[206].pertinencias = perti_r206;

regra[207].n_conjuntos = 4;
double * perti_r207[4] = {&p_Angulo[PM],&p_Sensor_Frente_Esq[ML],
                          &p_Sensor_Frente[nMP],&p_Dist_Objetivo[nMP],};
regra[207].motorE = NM;     regra[207].motorD = PA;     regra[207].pertinencias = perti_r207;

regra[208].n_conjuntos = 5;
double * perti_r208[5] = {&p_Angulo[PM],&p_Sensor_Frente_Esq[L],
                         &p_Sensor_Frente[nMP],&p_Sensor_Lateral_Esq[nL],&p_Dist_Objetivo[nMP],};
regra[208].motorE = NM;     regra[208].motorD = PM;     regra[208].pertinencias = perti_r208;

regra[209].n_conjuntos = 5;
double * perti_r209[5] = {&p_Angulo[PM],&p_Sensor_Frente_Esq[P],
                         &p_Sensor_Frente[nMP],&p_Sensor_Lateral_Esq[nML],&p_Dist_Objetivo[nMP],};
regra[209].motorE = Z;  regra[209].motorD = PM;     regra[209].pertinencias = perti_r209;

regra[210].n_conjuntos = 4;
double * perti_r210[4] = {&p_Angulo[PM],&p_Sensor_Frente[nMP],
                          &p_Sensor_Lateral_Esq[ML],&p_Dist_Objetivo[nMP],};
regra[210].motorE = NA;     regra[210].motorD = PA;     regra[210].pertinencias = perti_r210;

regra[211].n_conjuntos = 4;
double * perti_r211[4] = {&p_Angulo[PM],&p_Sensor_Frente[nMP],
                          &p_Sensor_Lateral_Esq[L],&p_Dist_Objetivo[nMP],};
regra[211].motorE = NA;     regra[211].motorD = PA;     regra[211].pertinencias = perti_r211;

regra[212].n_conjuntos = 2;
double * perti_r212[2] = {&p_Angulo[PG],&p_Dist_Objetivo[MP],};
regra[212].motorE = Z;  regra[212].motorD = Z;  regra[212].pertinencias = perti_r212;

regra[213].n_conjuntos = 4;
double * perti_r213[4] = {&p_Angulo[PG],&p_Sensor_Frente_Esq[ML],
                          &p_Sensor_Frente[nMP],&p_Dist_Objetivo[nMP],};
regra[213].motorE = NM;     regra[213].motorD = PA;     regra[213].pertinencias = perti_r213;

regra[214].n_conjuntos = 5;
double * perti_r214[5] = {&p_Angulo[PG],&p_Sensor_Frente_Esq[L],
                          &p_Sensor_Frente[nMP],&p_Sensor_Lateral_Esq[nL],&p_Dist_Objetivo[nMP],};
regra[214].motorE = NM;     regra[214].motorD = PA;     regra[214].pertinencias = perti_r214;

regra[215].n_conjuntos = 5;
double * perti_r215[5] = {&p_Angulo[PG],&p_Sensor_Frente_Esq[P],
                          &p_Sensor_Frente[nMP],&p_Sensor_Lateral_Esq[nML],&p_Dist_Objetivo[nMP],};
regra[215].motorE = NM;     regra[215].motorD = PA;     regra[215].pertinencias = perti_r215;

regra[216].n_conjuntos = 4;
double * perti_r216[4] = {&p_Angulo[PG],&p_Sensor_Frente[nMP],
                          &p_Sensor_Lateral_Esq[ML],&p_Dist_Objetivo[nMP],};
regra[216].motorE = NA;     regra[216].motorD = PA;     regra[216].pertinencias = perti_r216;

regra[217].n_conjuntos = 4;
double * perti_r217[4] = {&p_Angulo[PG],&p_Sensor_Frente[nMP],
                          &p_Sensor_Lateral_Esq[L],&p_Dist_Objetivo[nMP],};
regra[217].motorE = NA;     regra[217].motorD = PA;     regra[217].pertinencias = perti_r217;
```

```
sensor = new (nothrow) double*[4];
angulo = new (nothrow) double*[4];
dist_objetivo = new (nothrow) double*[4];
motor = new (nothrow) double*[5];


if (sensor == 0)
{
    cout << "Error: memory could not be allocated";
    return 0;
}
for (int i =0;i< 4;i++)
{
    sensor[i]= new (nothrow) double[3];

    if (sensor[i] == 0)
    {
        cout << "Error: memory could not be allocated";
        return 0;
    }
}

if (angulo == 0)
{
    cout << "Error: memory could not be allocated";
    return 0;
}
for (int i =0;i< 4;i++)
{
    angulo[i]= new (nothrow) double[3];

    if (angulo[i] == 0)
    {
        cout << "Error: memory could not be allocated";
        return 0;
    }
}

if (dist_objetivo == 0)
{
    cout << "Error: memory could not be allocated";
    return 0;
}
for (int i =0;i< 4;i++)
{
    dist_objetivo[i]= new (nothrow) double[3];

    if (dist_objetivo[i] == 0)
    {
        cout << "Error: memory could not be allocated";
        return 0;
    }
}

if (motor == 0)
{
    cout << "Error: memory could not be allocated";
    return 0;
}
for (int i =0;i< 4;i++)
{
    motor[i]= new (nothrow) double[3];
```

```
        if (motor[i] == 0)
        {
            cout << "Error: memory could not be allocated";
            return 0;
        }
    }

    sensor[0] = sensor_ML;       // CONJUNTO MUITO LONGE      --> ML
    sensor[1] = sensor_L;        // CONJUNTO LONGE            --> L
    sensor[2] = sensor_P;        // CONJUNTO PERTO            --> P
    sensor[3] = sensor_MP;       // CONJUNTO MUITO PERTO      --> MP

    angulo[0] = ang_NG;          // CONJUNTO NEGATIVO GRANDE --> MG
    angulo[1] = ang_NM;          // CONJUNTO NEGATIVO MEIDIO --> NM
    angulo[2] = ang_Z;           // CONJUNTO ZERO             --> Z
    angulo[3] = ang_PM;          // CONJUNTO POSITIVO MEDIO   --> PM
    angulo[4] = ang_PG;          // CONJUNTO POSITIVO GRANDE --> PG

    dist_objetivo[0] = dist_ML; // CONJUNTO MUITO LONGE      --> ML
    dist_objetivo[1] = dist_L;  // CONJUNTO LONGE            --> L
    dist_objetivo[2] = dist_P;  // CONJUNTO PERTO            --> P
    dist_objetivo[3] = dist_MP; // CONJUNTO MUITO PERTO      --> MP

    motor[0] = motor_NA;         // CONJUNTO NEGATIVO ALTO    --> NA
    motor[1] = motor_NM;         // CONJUNTO NEGATIVO MEDIO   --> NM
    motor[2] = motor_Z;          // CONJUNTO ZERO             --> Z
    motor[3] = motor_PM;         // CONJUNTO POSITIVO MEIDO   --> PM
    motor[4] = motor_PA;         // CONJUNTO POSITIVO ALTO    --> PA


// CONEXÃO COM O ROBÔ E SEUS DISPOSITIVOS
    PlayerClient *robot = new PlayerClient("127.0.0.1");
    Position2dProxy *p2dProxy = new Position2dProxy(robot, 0);
    RangerProxy *rangerProxy = new RangerProxy(robot, 0);
    p2dProxy->SetMotorEnable(true);
    p2dProxy->RequestGeom();
//*****************************************************************

    time_t begin, end;
    time(&begin);
    sleep(5);
    time(&end);
    printf("Time elapsed: %f seconds",difftime(end, begin));
    myfile.open ("LOG_ROBO.txt");
    myfile << "Tempo; ";
    myfile << "Sensor Lateral direito; ";
    myfile << "Sensor Frente direita; ";
    myfile << "Sensor Frente; ";
    myfile << "Sensor Esquerda Frente; ";
    myfile << "Sensor Lateral Esquerda; ";
    myfile << "Distancia ao objetivo; ";
    myfile << "Angulo ao objetivo; ";
    myfile << "Posissão x; ";
    myfile << "Posissão y; ";
    myfile << "Teta; ";
    myfile << "Motor_E NA; ";
    myfile << "Motor_E NM; ";
    myfile << "Motor_E Z; ";
    myfile << "Motor_E PM; ";
    myfile << "Motor_E PA; ";
    myfile << "Motor_E NA; ";
    myfile << "Motor_D NM; ";
```

```
    myfile << "Motor_D Z; ";
    myfile << "Motor_D PM; ";
    myfile << "Motor_D PA; ";
    myfile << "Defuzzy_E; ";
    myfile << "Defuzzy_D; ";
    myfile << "v; ";
    myfile << "w; ";
    myfile << "objetivo_x; ";
    myfile << "objetivo_y; ";
    myfile << endl << "0; ";
// -------------------LOOP DE CONTROLE ----------------------------------
    p2dProxy->SetMotorEnable(true);
    while (true)
    {

        time(&begin);
        ti = tf = tempo = 0;
        timeval tempo_inicio,tempo_fim;
        gettimeofday(&tempo_inicio,NULL);
        robot->ReadIfWaiting();

        media = 0;
        ranger[0] = 0;
        ranger[1] = 0;
        ranger[2] = 0;
        ranger[3] = 0;
        ranger[4] = 0;
        ranger[5] = 0;
        ranger[6] = 0;
        for (int i =0; i < 5; i++)
        {
            // LEITURA DOS SENSORES DE DISTÂNCIA
            ranger[0]= ranger[0] + (i+1)*rangerProxy->GetRange(0);  // lateral direita
            ranger[1]= ranger[1] + (i+1)*rangerProxy->GetRange(1);  // direita frente
            ranger[2]= ranger[2] + (i+1)*rangerProxy->GetRange(2);  // frente direita
            ranger[3]= ranger[3] + (i+1)*rangerProxy->GetRange(3);  // frente
            ranger[4]= ranger[4] + (i+1)*rangerProxy->GetRange(4);  // frente esquerda
            ranger[5]= ranger[5] + (i+1)*rangerProxy->GetRange(5);  // esquerda frente
            ranger[6]= ranger[6] + (i+1)*rangerProxy->GetRange(6);  // lateral esquerda
            media = media + (i+1);
        }
// ---------------------------------------------------------------------------------------
        // LOG DAS LEITURAS
        myfile << ranger[0]/media << "; ";
        myfile << Min(ranger[1],ranger[2])/media << "; ";
        myfile << ranger[3]/media << "; ";
        myfile << Min(ranger[4],ranger[5])/media << "; ";
        myfile << ranger[6]/media << "; ";
// ---------------------------------------------------------------------------------------
//------ FUZZIFICAÇÃO DAS LEITURAS DOS SENORES DE DISTÂNCIA---------------------------------

        pertinencia_palavra(sensor, 4, ranger[0]/media, p_Sensor_Lateral_Dir);
        pertinencia_palavra(sensor, 4, Min(ranger[1],ranger[2])/media, p_Sensor_Frente_Dir);
        pertinencia_palavra(sensor, 4, ranger[3]/media, p_Sensor_Frente);
        pertinencia_palavra(sensor, 4, Min(ranger[4],ranger[5])/media, p_Sensor_Frente_Esq);
        pertinencia_palavra(sensor, 4, ranger[6]/media, p_Sensor_Lateral_Esq);

        pertinencia_palavra_inv(sensor, 4, ranger[0]/media, p_Sensor_Lateral_Dir);
        pertinencia_palavra_inv(sensor, 4, Min(ranger[1],ranger[2])/media, p_Sensor_Frente_Dir);
        pertinencia_palavra_inv(sensor, 4, ranger[3]/media, p_Sensor_Frente);
        pertinencia_palavra_inv(sensor, 4, Min(ranger[4],ranger[5])/media, p_Sensor_Frente_Esq);
        pertinencia_palavra_inv(sensor, 4, ranger[6]/media, p_Sensor_Lateral_Esq);
// ---------------------------------------------------------------------------------------
        // CALUCULA DISTÂCIA E ORIENTAÇÃO DO ROBÔ EM RELAÇÃO AO OBJETIVO
```

```
        calcula_angulo_and_dist(p2dProxy->GetXPos(), p2dProxy->GetYPos(),
                                p2dProxy->GetYaw(), objetivo_x, objetivo_y,ang_and_dist);

//-------------------------------------------------------------------------------------------
        // LOG DOS VALORES DE DISTÂNCIA, POSIÇÃO E ORIENTAÇÃO
        myfile << ang_and_dist[1] << "; ";
        myfile << ang_and_dist[0] << "; ";
        myfile << p2dProxy->GetXPos() << "; ";
        myfile << p2dProxy->GetYPos() << "; ";
        myfile << p2dProxy->GetYaw()*(PI/180) << "; ";
//-------------------------------------------------------------------------------------------
//------ FUZZIFICAÇÃO DA DISTÂNCIA E DE ORIENTAÇÃO
        pertinencia_palavra(angulo, 5, ang_and_dist[0], p_Angulo);
        pertinencia_palavra_inv(angulo, 5, ang_and_dist[0], p_Angulo);
        pertinencia_palavra(dist_objetivo, 4, ang_and_dist[1], p_Dist_Objetivo);
        pertinencia_palavra_inv(dist_objetivo, 4, ang_and_dist[1], p_Dist_Objetivo);
//-------------------------------------------------------------------------------------------

        motorE[NA] = 0;
        motorE[NM] = 0;
        motorE[Z] = 0;
        motorE[PM] = 0;
        motorE[PA] = 0;

        motorD[NA] = 0;
        motorD[NM] = 0;
        motorD[Z] = 0;
        motorD[PM] = 0;
        motorD[PA] = 0;

// --------------   CALCULA O ANTECEDENTE DE CADA REGRA ------------------------
        for(int i = 0;i<N_REGRAS;i++)
        {
            regra[i].result_antecedente = exec_antecedente(regra[i].pertinencias,
                                                           regra[i].n_conjuntos);
            if(i>187) //  VERIFICA SE A REGRA É DE ORIENTAÇÃO
            {
                regra[i].result_antecedente = regra[i].result_antecedente*peso;
                // Menor peso para regras de direcionamento ao objetivo
            }

            // EXECUTA A COMBINAÇÃO DOS CONSEQUENTES COMO O MAXIMO ENTRE ELES
            if(regra[i].motorE == NA)
            {
                motorE[NA] = Max(motorE[NA],regra[i].result_antecedente);
            }
            else if(regra[i].motorE == NM)
            {
                motorE[NM] = Max(motorE[NM],regra[i].result_antecedente);
            }
            else if(regra[i].motorE == Z)
            {
                motorE[Z] = Max(motorE[Z],regra[i].result_antecedente);
            }
            else if(regra[i].motorE == PM)
            {
                motorE[PM] = Max(motorE[PM],regra[i].result_antecedente);
            }
            else if(regra[i].motorE == PA)
            {
                motorE[PA] = Max(motorE[PA],regra[i].result_antecedente);
            }

            // EXECUTA A COMBINAÇÃO DOS CONSEQUENTES COMO O MAXIMO ENTRE ELES
```

```
            if(regra[i].motorD == NA)
            {
                motorD[NA] = Max(motorD[NA],regra[i].result_antecedente);
            }
            else if(regra[i].motorD == NM)
            {
                motorD[NM] = Max(motorD[NM],regra[i].result_antecedente);
            }
            else if(regra[i].motorD == Z)
            {
                motorD[Z] = Max(motorD[Z],regra[i].result_antecedente);
            }
            else if(regra[i].motorD == PM)
            {
                motorD[PM] = Max(motorD[PM],regra[i].result_antecedente);
            }
            else if(regra[i].motorD == PA)
            {
                motorD[PA] = Max(motorD[PA],regra[i].result_antecedente);
            }

        }
//      LOG DAS ATIVAÇÕES DOS CONJUNTOS DOS MOTOERES ESQUERDO E DIREITO
        myfile <<  motorE[0] << "; " << motorE[1] << "; " << motorE[2] << "; ";
        myfile << motorE[3] << "; " << motorE[4] << "; ";
        myfile <<  motorD[0] << "; " << motorD[1] << "; " << motorD[2] << "; ";
        myfile << motorD[3] << "; " << motorD[4] << "; ";
// ------------------------------------------------------------------------

//------DEFUZZIFICAÇÃO DO MOTOR ESQUERDO
        PowerE = defuzzy_palavra(motor, 5, motorE, limit_esq, limit_dir);

//------DEFUZZIFICAÇÃO DO MOTOR ESQUERDO
        PowerD = defuzzy_palavra(motor, 5, motorD, limit_esq, limit_dir);
//------------------------------------------------------------------------

//------LOG DOS VALORES DA DEFUZZIFICAÇÃO ------------------------------------
        myfile << PowerE << "; ";
        myfile << PowerD << "; ";
//------------------------------------------------------------------------

        v  = 2*((PowerD + PowerE)/2); // VELOCIDADE LINEAR MÁXIMA DE 2m/S

        w = 64*((PowerD-PowerE)/(2*0.4))*(PI/180); // VELOCIDADE ANGULAR MAX = 160 graus/s
/*      if(v > 0.5)
            v = 0.5;
        else if(v < -0.5)
            v = -0.5;
*/

        p2dProxy->SetSpeed(v,w);     // ATUAÇÃO NOS MOTORES
        usleep(100000);              // TEMPO DE ATUAÇÃO
        p2dProxy->SetSpeed(0,0);     // TÉRMINO DA ATUAÇÃO

        w = w*(180/PI);


        cout << endl << "dist_obj = " << ang_and_dist[1] << ", ang_obj_graus = ";
        cout << ang_and_dist[0]*(180/PI) << ", ang_obj_rad = " << ang_and_dist[0] << endl;
        cout << "Velocidade v = " << v << endl;
        cout << "Velocidade w = " << w << endl;
        cout << "Objetivo x = " << objetivo_x << ", Objetivo y = " << objetivo_y << endl;
//------LOG DAS VELOCIDADES ---------------------------------------------------
        myfile << v << "; ";
```

```
        myfile << w << "; ";
        myfile << objetivo_x << "; " << objetivo_y << "; ";
//-------------------------------------------------------------------------------
//------CALCULO DO TEMPO DO LOOP ----------------------------------------------
        time(&end);
        cout << "Time elapsed: "<< difftime(end, begin) <<" seconds"  << endl;
        gettimeofday(&tempo_fim,NULL);
        tf = (double)tempo_fim.tv_usec + ((double)tempo_fim.tv_sec * (1000000.0));
        ti = (double)tempo_inicio.tv_usec + ((double)tempo_inicio.tv_sec * (1000000.0));
        tempo = (tf - ti) / 1000;
        printf("Tempo gasto em milissegundos %.3f\n",tempo);
        tempo_acc =  tempo_acc + tempo;
        cout << "tempo acumulado = " << tempo_acc << endl;
        cout << " cont = " << cont1 << endl;
//-------------------------------------------------------------------------------
//----- ROTINA DE EMERGÊNCIA, PARA CONTROLE DE COLISÃO ------------------------
        if(cont1 == 0)
        {
            cout << " reset timer para espera " << endl << endl;
            temp1 = tempo_acc;
            px1 = p2dProxy->GetXPos();
            py1 = p2dProxy->GetYPos();
            teta1 = p2dProxy->GetYaw()*(180/PI);
            espera = 3000;                      // 3 segundos
            cont1 = 1;
            cont2 = 0;
            esp = 0;
        }

        ti = tf = tempo = 0;
        gettimeofday(&tempo_inicio,NULL);

        if(tempo_acc - temp1 > espera) // 3 verifica se passaram 3 segundos
        {
            cout << " Verifica se está parado " <<  endl << endl << endl;
            // VERIFICA SE DESLOCOU MAIS QUE 20 cm EM X E Y
            if(abs(p2dProxy->GetXPos()-px1) < 0.20 && abs(p2dProxy->GetYPos()-py1 < 0.20))
            {
                // VERIFICA SE VARIOU A ORIENTAÇÃO
                if(abs(p2dProxy->GetYaw()*(180/PI) - teta1) < 0.5 )
                {
                    // SEM VARIAÇÃO NO ANGULO, ROBÔ COLIDIU
                    cout << " recuo " << endl;
                    p2dProxy->SetSpeed(0,0);
                    usleep(300000);
                    p2dProxy->SetSpeed(-0.35,0);     // recua para se posicionar melhor
                    sleep(1);
                    p2dProxy->SetSpeed(0,-1);     // recua para se posicionar melhor
                    usleep(400000);
                    cont1 = 0;
                    cont2 = 1;
                    peso = 0.2;
                    temp2 = tempo_acc;
                    espera = 3000;
                    goto L1;
                }
                else if(esp == 0)
                {
                    // CASO EM QUE O ÂNGULO VARIOU, ROBÔ TENTANDO SAIR DE UMA
                    // CONFIGURAÇÃO DE PERIGO. ESPERA MAIS....
                    espera = 2*espera;
                    esp = 1;
                    goto L1;
                }
```

```
                    // EXECUTA RECUO PARA SAIR
                    cout << " recuo " << endl;
                    p2dProxy->SetSpeed(0,0);
                    usleep(300000);
                    p2dProxy->SetSpeed(-0.35,0);      // recua para se posicionar melhor
                    sleep(1);
                    p2dProxy->SetSpeed(0,-1);      // recua para se posicionar melhor
                    usleep(400000);
                    cont1 = 0;
                    cont2 = 1;
                    peso = 0.2;         // DIMINIU PESO DAS REGRAS DE ORIENTAÇÃO
                    espera = 4000;
L1:                 cout << " jump ";

            }
            else
            {
                cont1 = 0; // reset timer do recuo
            }
            gettimeofday(&tempo_fim,NULL);
            tf = (double)tempo_fim.tv_usec + ((double)tempo_fim.tv_sec * (1000000.0));
            ti = (double)tempo_inicio.tv_usec + ((double)tempo_inicio.tv_sec * (1000000.0));
            tempo = (tf - ti) / 1000;
            tempo_acc =  tempo_acc + tempo; // atualiza tempo apÃ³s o recuo
            temp2 = tempo_acc;              // tempo com o peso das resgras de direÃ§Ã£o menor
            cout << "tempo acumulado = " << tempo_acc << endl;

        }
        // 3 segundos com as regras de orientaçao com peso reduzido
        if(tempo_acc - temp2 > 3000 && cont2)
        {
            peso = 0.7;
        }

        myfile << endl << tempo_acc << "; ";

    }


}
```

3. Função auxiliar calcula_angulo_and_dist

```
/*
 * calcula_angulo_and_dist.h
 *
 *  Created on: Oct 6, 2013
 *      Author: Daniel de Sousa Leite
 */

#include <stdio.h>      /* printf */
#include <math.h>       /* atan2 */
#include <iostream>
#define PI 3.14159265

// Calcula o ângulo e a distância do robô ao objetivo
using namespace std;

void calcula_angulo_and_dist (double robo_x, double robo_y,double robo_teta,
                              double objetivo_x, double objetivo_y, double* ang_and_dist )
{
    double angulo;
    ang_and_dist[1] =(sqrt(pow(robo_x - objetivo_x, 2) + pow(robo_y - objetivo_y, 2)));
    ang_and_dist[0] = atan2((objetivo_y - robo_y),(objetivo_x - robo_x));
    angulo = ang_and_dist[0] - robo_teta;
    if(angulo > PI)
    {
        angulo = angulo - 2*PI;
    }
    else if(angulo < -PI)
    {
        angulo = angulo + 2*PI;
    }
    ang_and_dist[0] = angulo;
}
```

4. Função auxiliar defuzzy

```
/* defuzzy.h
 *  Created on: Oct 6, 2013
 *      Author: Daniel de Sousa Leite
 */
// Caculoa o ponto de maximo do conjunto fuzzy ativado
// conj = vetor de 3 elementos com as estremidades do triangulo que representa
// o conjunto fuzzy [esqueda, meio e direita], para reprensetar a extremidade
// um trapezio, basta repetir dos dois primeiros ou os dois ultimos elementos do
// vetor conj. Exemplo lateral esquerda: conj =[1,1,3]
// valor = valor ao que deseja-se saber a pertinencia ao conjunto em questï¿½o
# include <stdio.h>
#include <iostream>
using namespace std;

double defuzzy(double * conj, double valor, double limit_esq, double limit_dir)
{
    double saida = 0;
    double alpha_up = 1/(conj[1]-conj[0]);
    double beta_up = -alpha_up*conj[0];
    double alpha_down = 1/(conj[1]-conj[2]);
    double beta_down = -alpha_down*conj[2];
    if((alpha_up < 0 || alpha_down > 0 || conj[0] == conj[2]) &&
                                    conj[0] != conj[1] && conj[1] != conj[2])
    {
        cout << "Valores invalidos para limites do conjunto  Fuzzy!!!!!!!!" << endl;
        return 9999;
    }
    else if(conj[0] == conj[1]) // CONJUNTO FOR  TRAPEZIO COM INCLINACAO DEGATIVA
    {
        if(valor == 1)
        {
            saida = limit_esq;
            // valor dentro do base menor do trapezio
        }
        else if(valor < 1 && valor > 0)
        {
            saida =  (((valor - beta_down)/alpha_down) + limit_esq)/2;
            // valor na reta de descida do trapezio
        }
    }
    else if(conj[1] == conj[2]) // CONJUNTO FOR TRAPEZIO COM INCLINACAO POSITIVA
    {
        if(valor == 1)
        {
            saida = limit_dir;
            // valor dentro do base menor do trapezio
        }
        if(valor < 1 && valor > 0)
        {
            saida = (((valor - beta_up)/alpha_up) + limit_dir)/2;
            // retorna valor correspendente a reta de subida do trapezio
        }
    }
    else if(valor > 0)
    {
        saida = (((valor - beta_up)/alpha_up) + ((valor - beta_down)/alpha_down))/2;
        // retorna valor correspendente a reta de subida
    }
    if( valor <= 0)
    {
        saida = 9999; // codigo para função nao ativada
    }
    return saida;
}
```

5. Função auxiliar defuzzy_palavra

```
/*
 * defuzzy_palavra.h
 *
 *  Created on: Oct 6, 2013
 *      Author: Daniel de Sousa Leite
 */
// Realiza a media ponderada dos maximo dos conjuntos ativados

# include <stdio.h>
# include <iostream>
using namespace std;

double defuzzy_palavra(double ** variavel_fuzzy, int n_conj,
                       double* valor,double limit_esq, double limit_dir)
{
    double potencia = 0;
    double def = 0;
    double div = 0;
    int i;
    for(i = 0;i < n_conj;i++)
    {
        def = defuzzy(variavel_fuzzy[i], valor[i], limit_esq, limit_dir);
        if(def != 9999) //codigo para funcao nao ativada
        {
            potencia = potencia + def*valor[i];
            div = div + valor[i];
        }
    }
    if(div == 0)
    {
        return 0;
    }
    else
    {
        return potencia/div; // Media ponderada dos mÃ¡ximos de cada conjunto
    }
}
```

6. Função auxiliar exec_antecedente

```
/*
 * exec_antecedente.h
 *
 *  Created on: Oct 6, 2013
 *      Author: Daniel de Sousa Leite
 */

// Calcula a pertinencia do antecedente da regra

# include <stdio.h>
# include <iostream>

/* Macro que retorna o menor entre 2 valores */
#define Min(x, y) (((x) < (y)) ? (x): (y))

/* Macro que retorna o maior entre 2 valores */
#define Max(x, y) (((x) > (y)) ? (x): (y))
// Para usar operador Min poerador[i] == 1; Max opeardor[i] == 0

using namespace std;

double exec_antecedente(double ** pertinencias, int n_antecedentes)
{
    double aux = 1;
    for(int i = 0;i<n_antecedentes;i++)
    {
        aux = Min(aux,*pertinencias[i]); // operacao AND
    }
    return  aux;
}
```

7. Função auxiliar pertinência

```
/*
 * pertinencia.h
 *
 *  Created on: Oct 6, 2013
 *      Author: Daniel de Sousa Leite
 */

// conj - vetor de 3 elementos com as estremidades do triangulo que representa
// o conjunto fuzzy [esqueda, meio e direita], para reprensetar a extremidade
// de um trapezio, basta repetir dos dois primeiros ou os dois ultimos elementos do
// vetor conj. Exemplo lateral esquerda: conj -[1,1,3]
// valor - valor ao que deseja-se saber a pertinencia ao conjunto em questão
# include <stdio.h>
#include <iostream>
using namespace std;

double pertinencia(double * conj, double valor)
{
    double alpha_up = 1/(conj[1]-conj[0]);
    double beta_up = -alpha_up*conj[0];
    double alpha_down = 1/(conj[1]-conj[2]);
    double beta_down = -alpha_down*conj[2];
    if((alpha_up < 0 || alpha_down > 0 || conj[0] == conj[2])
    && conj[0] != conj[1] && conj[1] != conj[2])
    {
        cout << "Valores invalidos para limites do conjunto  Fuzzy!!!!!!!!" << endl;
        return 999;
    }
    else if(conj[0] == conj[1])
    {
        if(valor <= conj[1])
        {
            return 1;
            // valor dentro do base menor do trapezio
        }
        else if(valor > conj[1] && valor <= conj[2])
        {
            return alpha_down*valor + beta_down;
            // valor na reta de descida do trapezio
        }
        else
        {
            return 0;
        }
    }
    else if(conj[1] == conj[2])
    {
        if(valor >= conj[1])
        {
            return 1;
            // valor dentro do base menor do trapezio
        }
        else if(valor < conj[1] && valor >= conj[0])
        {
            return alpha_up*valor + beta_up;
            // retorna valor correspendente a reta de subida do trapezio
        }
        else
        {
            return 0;
        }
    }
    else if(valor <= conj[1] && valor > conj[0])
    {
```

```
            return alpha_up*valor + beta_up;
            // retorna valor correspendente a reta de subida
        }
        else if(valor > conj[1] && valor < conj[2])
        {
            return alpha_down*valor + beta_down;
            // retorna valor correspendente a reta de descida
        }
        else
        {
            return 0;
        }
}
```

8. Função auxiliar pertinência_palavra

```
/*
 * pertinencia_palavra.h
 *
 *  Created on: Oct 6, 2013
 *      Author: Daniel de Sousa Leite
 */

// regra => matriz cujas linhas sao os conjuntos fuzzy e cada valor de coluna
// representa as extremicdades do conjunto fuzzy.
// retorna um vetor com a pertinencia do valor para cada conjunto da regra.

# include <stdio.h>
#include <iostream>
using namespace std;

void pertinencia_palavra(double ** variavel_fuzzy, int n_conj,double valor,double * ativa)
{
    for(int i = 0;i < n_conj;i++)
    {
         ativa[i] = pertinencia(variavel_fuzzy[i],valor);
    }
    return;
}
```

9. Função auxiliar petinencia_palavra_inv

```
/*
 * pertinencia_palavra_inv.h
 *
 *  Created on: Oct 6, 2013
 *      Author: Daniel de Sousa Leite
 */

/* Macro que retorna o maior entre 2 valores */
#define Max(x, y) (((x) > (y)) ? (x): (y))
// Para usar operador Min poerador[i] == 1; Max opeardor[i] == 0

#include <stdio.h>
#include <iostream>
using namespace std;

void pertinencia_palavra_inv(double ** variavel_fuzzy, int n_conj,double valor,double * ativa)
{
    double aux[3];
    double aux2;
    double aux3;
    for(int i = 0;i < n_conj;i++)
    {
        // verifica se o conjunto é trepeziodal com inclinação na esquerda
        if(variavel_fuzzy[i][0] == variavel_fuzzy[i][1])
        {
            aux[0] = variavel_fuzzy[i][1];
            aux[1] = variavel_fuzzy[i][2];
            aux[2] = variavel_fuzzy[i][2];
            ativa[i + n_conj] = pertinencia(aux,valor);
        }
        // verifica se o conjunto é trepeziodal com incliançao na direita
        else if(variavel_fuzzy[i][1] == variavel_fuzzy[i][2])
        {
            aux[0] = variavel_fuzzy[i][0];
            aux[1] = variavel_fuzzy[i][0];
            aux[2] = variavel_fuzzy[i][1];
            ativa[i + n_conj] = pertinencia(aux,valor);
        }
        // Caso de conjuntos triangulares
        else
        {
            aux[0] = variavel_fuzzy[i][0];
            aux[1] = variavel_fuzzy[i][0];
            aux[2] = variavel_fuzzy[i][1];
            aux2 = pertinencia(aux,valor);

            aux[0] = variavel_fuzzy[i][1];
            aux[1] = variavel_fuzzy[i][2];
            aux[2] = variavel_fuzzy[i][2];
            aux3 = pertinencia(aux,valor);
            ativa[i + n_conj] = Max(aux2,aux3);
        }
    }
    return;
}
```

10. Arquivo de configuração do robô para Stage

```
# bengala.inc

#  prototype for bengala sonar sensors
define b_sonars sensor
(

  size [0.043 0.020 0.015 ]     # define the size of each transducer [xsize ysize zsize] in meters
  range [0 1.70]                # define the range bounds [min max]
  fov 25.5                      # define the angular field of view in degrees
  samples 1                     # define the number of samples spread over the fov, the sensor

                                # resolution (read/degrees)
  # generic model properties with non-default values
  # watts 2.0
  # color_rgba [ 0 1 0 0.15 ]

)

define bengalas_sonars ranger
(
  # define the pose of each transducer [xpos ypos heading]
  b_sonars( pose [ 0.025 -0.2 0 -70] ) #sensor lateral direita
  b_sonars( pose [ 0.025 -0.2 0 -35 ] ) #sensor frente direita 1
  b_sonars( pose [ 0.025 -0.10 0 -19.5 ] ) #sensor frente direita 2
  b_sonars( pose [ 0.05 0 0 0] ) # sensor frente
  b_sonars( pose [ 0.025 0.10 0 19.5 ] ) # sensor frente esquerda 1
  b_sonars( pose [ 0.025 0.2 0 35 ] ) #sensor frente esquerda  2
  b_sonars( pose [ 0.025 0.2 0 70] ) # sensor lateral esquerda
)


 # bengala's body
 define bengala position
 (
   size [0.15 0.40 0.20]         # actual size
   origin [0.075 0 0 0]  # bengala's centre of rotation is offset from its centre of area

   # the shape of bengala
   block
   (
     points 6
     point[0] [0 0 0]
     point[1] [0.1 0 0]
     point[2] [0.15 0.15 0]
     point[3] [0.15 0.25 0]
     point[4] [0.1 0.4 0]
     point[5] [0 0.4 0]
     z [0.05 0.2]  # distancia do chão e altura
   )
   mass 7.0                             # estimated mass in KG
   #friction 1000.2
   drive "diff"                         #differential steering model
   velocity [ 0.0 0.0 0.0 0.0 ]         # velocidade incial
   #odom_error [0.03 0.03 0.00 0.05]    # odometry error model parameters,
                                        # only used if localization is set to "odom"
   #[xmin xmax ymin ymax zmin zmax amin amax]
   # velocity_bounds [-1 1 -1 1 -1 1 -90 90 ]  # x,y,z in m/s, a in d/s
   # acceleration_bounds [-1 1 -1 1 -1 1 -90 90] # x,y,z in m/s, a in d/s²

   gui_nose 1     # if 1, draw a nose on the model showing its heading (positive X axis)
```

```
  bengalas_sonars(pose [0.075 0 0 0])  # sensors attached to bengala

  # determine how the model appears in various sensors
  obstacle_return 1     # Can hit things.
  ranger_return 1       # reflects sonar beams
  blobfinder_return 1
  fiducial_return 1
  gripper_return 1
)
```

11. Arquivo .world para o Stage

```
include "map.inc"
include "agente.inc"
include "sick.inc"
include "uoa_robotics_lab_models.inc"


#speedup 1                      # Stage will attempt to run at this multiple of real time. If -1,

                                # Stage will run as fast as it can go, and not attempt to track real

                                # time at all
paused 1

#resolution 0.02                # specifies the resolution of the underlying bitmap model

# configure the GUI window
window
(
  size [ 2400.000 2400.000 ]    # size of the window in pixels

  # camera options
  scale 45   # pixels per meter # ratio of world to pixel coordinates (window zoom)
  center [-40.71 1.23]           # location of the center of the window in world coordinates (meters)
  rotate [0 0]                  # angle relative to straight up, angle of rotation (degrees)

  # perspective camera options
  #pcam_loc [ 0 -4 2 ]          # location of the perspective camera (meters)
  #pcam_angle [ 70 0 ]          # verticle and horizontal angle of the perspective camera


  # GUI options
  show_data 1
  show_flags 1
  show_blocks 1
  show_clock 1
  show_footprints 0
  show_grid 1
  show_trailarrows 1
  show_trailrise 0
  show_trailfast 0
  show_occupancy 0
  show_tree 0
  pcam_on 0
  screenshots 0
)

# load an environment bitmap
floorplan
(
  size [120.000 120.000 0.800]
  pose [0 0 0 0]
  bitmap "bitmaps/qt4.png"
  obstacle_return 1 # Can hit things.
  ranger_return 1   # reflects sonar beams
  fiducial_return 1

)
```

```
bengala
(
  name "idoso"          # can refer to the robot by this name
  pose [ -40.71 1.23 0 90.000 ]
  color "red"
  # report error-free position in world coordinates
  localization "gps"
  localization_origin [ 0 0 0 0 ]
  obstacle_return 1 # Can hit things.
  ranger_return 1   # reflects sonar beams
  fiducial_return 1
)
bengala
(
  name "p1"     # can refer to the robot by this name
  pose [ -449 10 0 90.000 ]
  color "blue"
  sicklaser( pose [0.075 0 0.08 0])
  # demonstrate a plugin controller, implemented in examples/ctrl/wander.cc
  # you probably should comment this out when using simple.cfg with Player
  #ctrl "wander"
  # report error-free position in world coordinates
  localization "gps"
  localization_origin [ 0 0 0 0 ]
  obstacle_return 1 # Can hit things.
  ranger_return 1   # reflects sonar beams
  fiducial_return 1
)
```

12. Arquivo .cfg para o Player

```
world
(
   name     "simul.world"       # the name of the world, as displayed in the window title bar
   interval_real   100          # the amount of real-world (wall-clock) time the siulator will attempt
                                # to spend on each simulation cycle.
   interval_sim    100          # the length of each simulation update cycle in milliseconds
   gui_interval    100          # the amount of real-world time between GUI updates
   resolution      0.001        # specifies the resolution of the underlying bitmap model
)
driver
(
   name "stage"                 # Name of the driver to instantiate, as it was provided
                                # toDriverTable::AddDriver()
   plugin "stageplugin"         # Name of a shared library (i.e., a "plugin") that contains the driver
   provides ["simulation:0"]    # The device address(es) through which the driver can be accessed.
   worldfile "simul.world"      # where <string> is the filename of a Stage worldfile.

)
# the main idoso1 robot
driver
(
   name "stage"
   provides ["6665:position2d:0" "6665:ranger:0"]
   model "idoso"
)

driver
(
   name "stage"
   provides ["6665:position2d:1" "6665:ranger:1"]
   model "p1"
)
```