

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO

DCA0121 - INTELIGÊNCIA ARTIFICIAL APLICADA - PROJETO UNIDADE II

Componentes: **José Genilson da Silva Filho**

**Renato Maia Chacon**

**RESUMO**

Este projeto consiste na implementação de duas técnicas para o controle de um robô diferencial: lógica binária e um método de inteligência artificial chamado de lógica fuzzy ou lógica nebulosa. Foi utilizada a ferramenta V-Rep para que um determinado robô (Pioneer P3) seja capaz de percorrer um espaço e desviar de forma autônoma de obstáculos colocados em uma pista virtual de treinamento. Este artigo irá primeiramente apresentar o problema em questão, a solução usando lógica binária e lógica fuzzy assim como os resultados obtidos.

**DESCRIÇÃO DO PROBLEMA A SER RESOLVIDO**

A dupla precisou pensar em uma maneira de implementar as técnicas de lógica fuzzy e lógica binária para que seja possível locomover um determinado robô em um ambiente desconhecido, evitando assim que haja colisão do mesmo com obstáculos. Os métodos deverão retornar a velocidade angular das rodas do robô para que ele desvie das barreiras. O robô deve seguir o modelo diferencial. Além disso a dupla fez uso de três sensores do tipo sonar com range de 2 metros e raio de 30°.

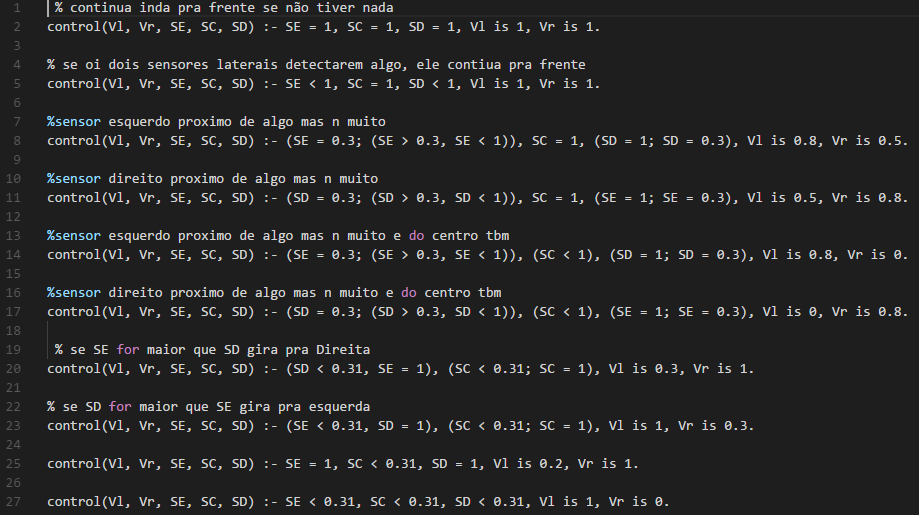
**ESTRATÉGIA UTILIZADA**

Para implementarmos a lógica binária para o controle do robô no V-Rep a dupla escolheu fazer uso da linguagem Python em conjunto com a Prolog. Foi escolhido pelo grupo trabalhar com o robô Pioneer do V-Rep, de forma que sejam utilizados apenas 3 dos 16 sensores ultrassônicos frontais para desviar de obstáculos através do controle da velocidade angular das duas rodas. Para a lógica fuzzy, foi usado o Matlab com conjunto com a ToolBox fuzzy.

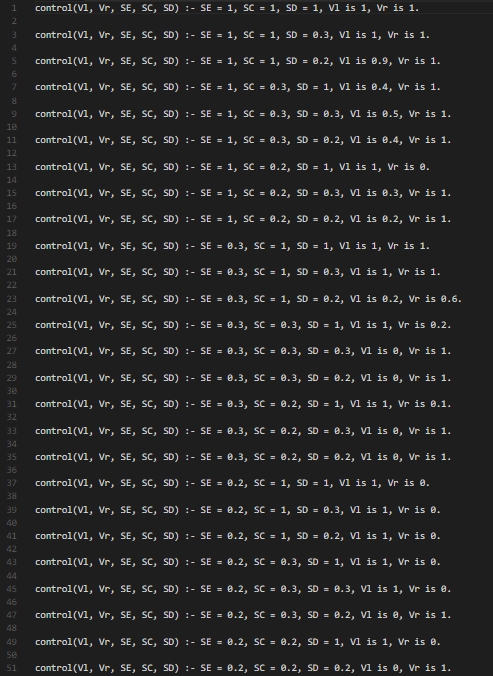
**ESQUEMA DE TREINAMENTO**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | |  | | | |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

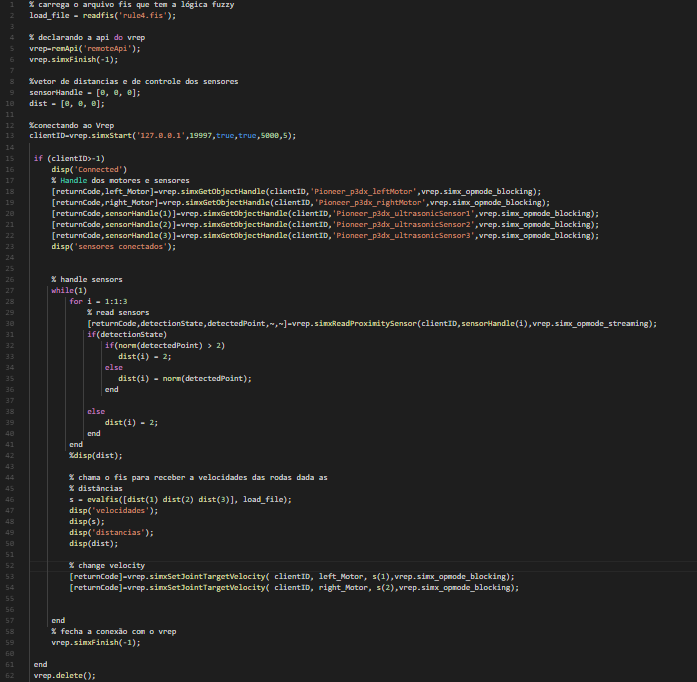
**DESCRIÇÃO DETALHADA DO ALGORITMO**

****

**Regra 1**

****

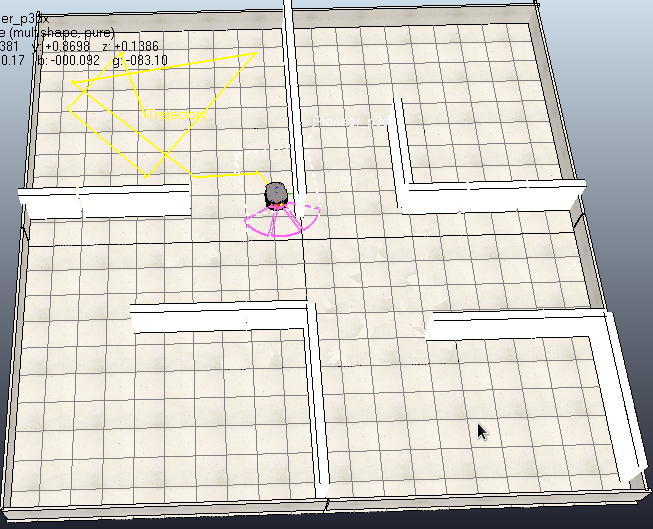
**Regra 2**

****

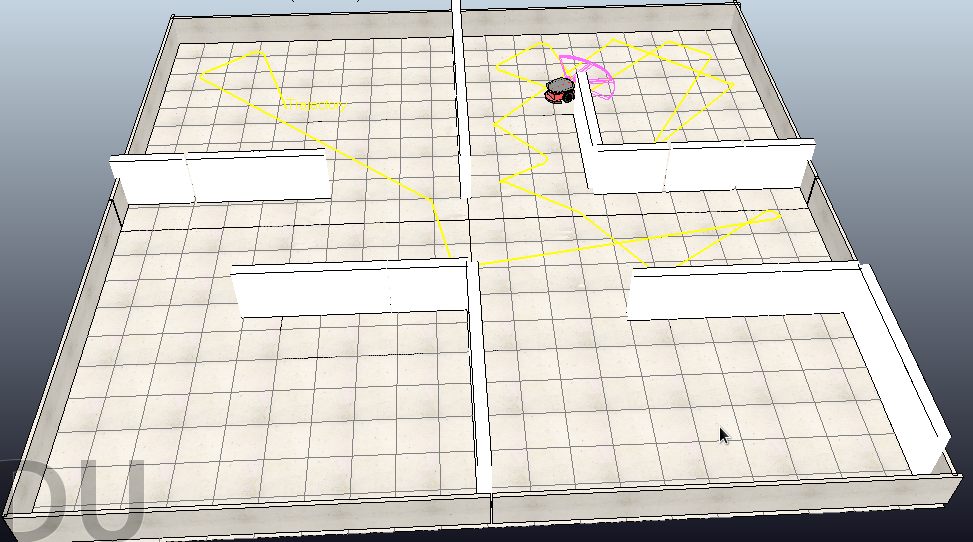
**Código em Matlab**

**RESULTADOS E TESTES**

Tomando primeiramente como análise a lógica binária, foi determinado que os dois testes fossem feitos em Prolog, graças à facilidade com que são introduzidas regras ao sistema. Foram então desenvolvidos dois códigos de regras: a primeira com poucas regras, em que o robô possui um resultado insatisfatório em comparação com a segunda, que apresenta mais regras e um melhor rendimento do robô ao navegar.   
  
 Pelo fato de se tratar de uma lógica binária, fica evidentemente provado que o robô precisará de muitas regras para se locomover em ambientes fechados de forma satisfatória e eficiente. Seguem abaixo as plotagens de cada conjunto de regras:

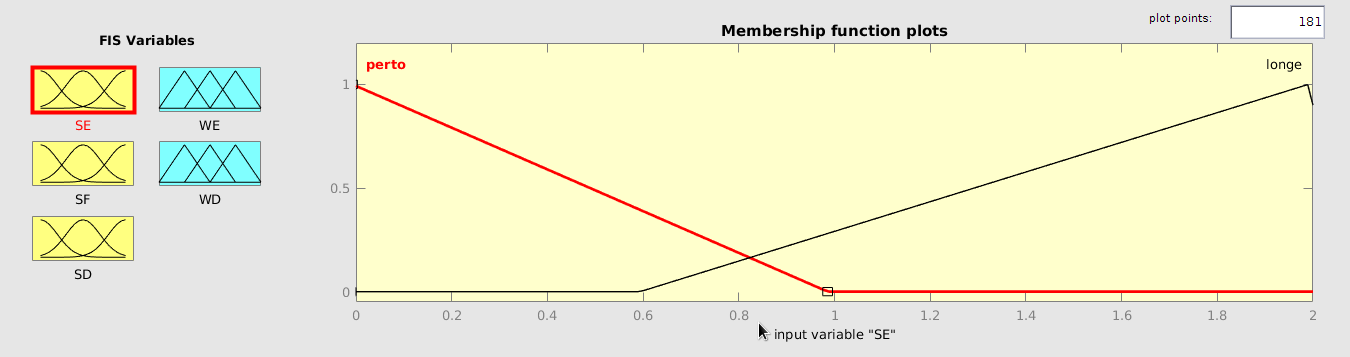


**Plotagem 1:** *Deslocamento do robô (Regra 1)*

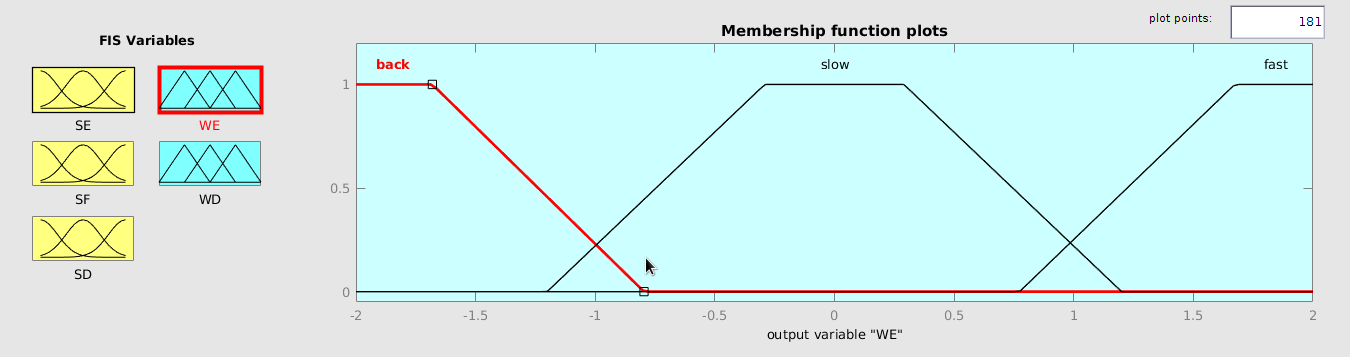


**Plotagem 2:** *Deslocamento do robô (Regra 2)*

Na lógica fuzzy, são consideradas informações dadas como incertas ou vagas para que, ao obter de forma incerta a informação, seja feita a fuzzificação dos dados de entrada por meio das funções de pertinência. Foi escolhido pela dupla trabalhar com 2 saídas e 3 entradas, sendo elas com cinco funções de pertinência. Enquanto nas variáveis de entrada foi usada a função Gaussiana, na de saída foi usada a função trapézio. Foram associados os seguintes termos às funções de pertinência para a entrada: ‘perto’ e ‘longe’. Seguem abaixo as imagens ilustrando as funções de pertinência de entrada e saída:

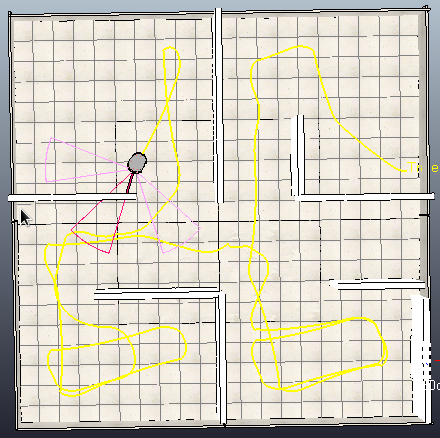


**Imagem 1:** *Funções de pertinência de entrada*

**

**Imagem 2:** *Funções de pertinência de saída*

Na plotagem a seguir, é mostrada a rota percorrida pelo robô quando submetido à lógica fuzzy:



**REQUISITOS FUNCIONAIS DA IMPLEMENTAÇÃO**

Neste projeto, todos as etapas até aqui citadas foram de total importância para chegarmos até o resultado final. A começar pela implementação do algoritmo de treinamento. A MLP faz o cálculo da saída da camada escondida fazendo o somatório das entradas multiplicadas por seus respectivos pesos, encontrando um resultado que passa pela função de ativação tanh, que dá a entrada da camada de saída. No caso desse trabalho, temos seis entradas externas além do bias, onde as seis entradas definem a distância​ ​ do​ ​ robô​ ​ a ​ ​ um​ ​ possível​ ​ obstáculo​ ​ à ​ ​ sua​ ​ frente​ ​ ou​ ​ lados.

Na camada de saída temos um número de neurônios igual ao número de saídas desejadas (duas), onde o mesmo processo da camada anterior é repetido, após isso, teremos a saída da primeira iteração do sistema. Caso o erro não seja satisfatório, acontecerá a atualização de pesos, até que se chegue em um erro aceitável para o problema.​ ​ A ​ ​ atualização​ ​ dos​ ​ pesos​ ​ é ​ ​ feita​ ​ pelo​ ​ algoritmo​ backpropagation.

Começamos o treinamento do robô utilizando um dataset muito grande, e com valores que faziam com que a rede divergisse. Após um pouco estudo, decidimos utilizar um robô que já andava só por ambientes sem colidir, e usá-lo para obter o nosso conjunto de treinamento.

Com esses dados em mãos, fizemos um filtro manual do que seriam os melhores pesos, e utilizamos-os para treinar a rede.

**CONCLUSÃO**

Foi percebido pela dupla que apesar de lidar com maiores margens incertezas no uso da lógica fuzzy, o resultado quando submetido a teste é ainda melhor do que quando é usada a lógica binária. Notou-se que que o número de colisões do robô com objetos foi reduzido quando o mesmo esteve submetido à lógica fuzzy.

**LINKS**

* <http://www.coppeliarobotics.com/downloads.html> (site do VRep)
* <https://drive.google.com/drive/folders/0B66BO9eGtLpGcVpSTUFLTXNJczg> (códigos e vídeo do projeto)