MO810 - Trabalho 2

MATEUS CORADINI SANTOS *

*Prof (a): Profa. Dra. Esther Luna Colombini
Aluno especial - Mestrado
E-mail: mateuscoradini@gmail.com

***Resumo* – O objetivo inicial deste trabalho sï¿½o os testes das teorias aplicadas em sala de aula, colocando em pratica a implementaï¿½ï¿½o do sistema de controle diferencial para um robo movel executado em um ambiente simulado, no caso, usamos o software V-REP. Foram implementados dois comportamentos de controle *Avoid Obstacle* e *Wall Follow* utilizando sistemas Fuzzy em Java e estimativa de Pose baseada em odometria. As classes de controle para os algoritmos em Fuzzy, foram implementadas a modo de fazer somentes testes bï¿½sicos de implementaï¿½ï¿½o e nï¿½o houve um trabalho de modelagem especifica para obter melhores resultados, pois o objetivo do trabalho era entender e aprender a utilizar a lï¿½gica em funï¿½ï¿½o das tarefas executadas anteriormente como malha aberta. A odometria nï¿½o estï¿½ correta, e ainda necessita de tratamento, pois nï¿½o esta limpando os erros acumulados. Os algoritmos de fuzzy para WallFollow e AvoidObstacle se encontram sem otimizaï¿½ï¿½o no momento.**

***Palavras-chave* – V-REP Pioneer AvoidObstacleColision WallFollow Fuzzy**

I. INTRODUï¿½ï¿½O

Este projeto consiste no desenvolvimento de um sistema com inteligencia necessï¿½ria para o controle do modelo de robo *Pioneer P3-DX* no simulador *V-REP*. A implementaï¿½ï¿½o foi realizada em Java versï¿½o 1.7, com a utilizaï¿½ï¿½o de algumas bibliotecas como JFuzzyLite. Os ciclos de leitura e atuaï¿½ï¿½o dos sensores sï¿½o controlados por threads na qual utilizamos um tempo de 300ms para atuaï¿½ï¿½o. O cï¿½digo fonte pode ser obtido em https://github.com/mateuscoradini/VRepSimulator/V2. As instruï¿½ï¿½es de instalaï¿½ï¿½o se encontram na pagina principal, mapeados para MAC-OS e Windows.

Este artigo estï¿½ dividido em trï¿½s sessï¿½es principais, cada uma com sua apresentaï¿½ï¿½o e discussï¿½o dos resultados.

- Odometria - Estimativa de Pose
- Controle: Avoid Obstaclea (Evitar obstï¿½culo)
- Controle: Wall Follow (Seguir parede)

II. ODOMETRIA

A estimativa de pose, foi comparada inicialmente ao GroundThruth (posiï¿½ï¿½o real do robï¿½), para que medisse as falhas em acumulo, ocorridas durante o tempo de execuï¿½ï¿½o do trajeto do robï¿½. Para isso, a implementaï¿½ï¿½o foi realizada na classe *PositionLocator*. Conforme velocidade das rodas, calculados pela formula citada em sala de aula, estima-se a localizaï¿½ï¿½o aproximada no vetor x,y,z, a localizaï¿½ï¿½o do robï¿½.

A. Componente: Rodas

Cada roda, possui um enconder, que nos informa o valor da sua posiï¿½ï¿½o angular. Com as informaï¿½ï¿½es desses enconders, ï¿½ possivel determinar a velocidade do robï¿½ atrï¿½s da formula:

$$V = \frac{\Delta\theta}{\Delta time} R$$

Em que $\Delta\theta$ representa a diferenï¿½a angular entre posiï¿½ï¿½es do *encoder* durante um intervalo de tempo $\Delta time$ e R ï¿½ o raio da roda. ï¿½ importante destacar que o cï¿½lculo da diferenï¿½a angular deve levar em conta a orientaï¿½ï¿½o do giro e o universo em que estï¿½o sendo determinados os ï¿½ngulos.

A implementaï¿½ï¿½o do mï¿½todos de cï¿½lculo de velocidade da roda encontra-se na classe *RobotWheelHandler*. A orientaï¿½ï¿½o do giro ï¿½ obtida utilizando a hipï¿½tese que a diferenï¿½a angular deve ser sempre menor do que π.

B. Velocidade do Robï¿½ diferencial

Dada a velocidade de cada roda, pode-se calcular a velocidade linear e angular do robï¿½ atravï¿½s da fï¿½rmula:

$$V = \frac{V_r + V_l}{2} \qquad \omega = \frac{V_r - V_l}{2L}$$

Figura 1. Formula: Velocidade Linear (V), (W) Velocidade Rotacional ao longo do eixo

Em que V_r ï¿½ a velocidade da roda direita, V_l ï¿½ a roda esquerda, D ï¿½ a distï¿½ncia entre as rodas, V ï¿½ a velocidade linear e ω ï¿½ a velocidade angular.

C. Pose

A pose do robï¿½ ï¿½ entï¿½o calculada

```
float x = (float) (robotLastPosition.getX()
        + (deltaSpace * Math.cos(addDelta(robotLastPosition.getOrientation(), deltaTheta / 2))));
float y =  (float) (robotLastPosition.getX()
        + (deltaSpace * Math.sin(addDelta(robotLastPosition.getOrientation(), deltaTheta / 2))));
float orientation = addDelta(robotLastPosition.getOrientation(), deltaTheta);
```

Figura 2. Classe PositionLocator: odometry

D. Resultados dos testes

Nï¿½o foi possivel gerar o grafico de coordenadas das linhas de comparaï¿½ï¿½o, entï¿½o determinei uma saida de console comparativa, somente para verificar a implementaï¿½ï¿½o e efetuar teste de que se havia erro acumulado. A saida abaixo foi gerada ao usar o algoritmo para controle de Wall Follow, apï¿½s alguns algum tempo de execuï¿½ï¿½o:

```
GroundThruth {0.49356467 X, -1.8204131 Y, -1.986935 O}
Odometry {0.46356467 X, -1.7204131 Y, -1.986935 O}
GroundThruth {0.48098224 X, -1.7404131 Y, -2.0268958 O}
Odometry {0.45356467 X, -1.7404131 Y, -2.0268958 O}
GroundThruth {0.46728283 X, -1.7625132 Y, -2.0673726 O}
Odometry {0.43356467 X, -1.7104131 Y, -2.0673726 O}
```

Figura 3. Saida do console em vetor x,y, orientaï¿½ï¿½o, da PoseLocator

Nï¿½o consegui concluir com uma dimensï¿½o exata o acumulo erro crï¿½tico. Gostaria de gerar esses dados a partir de um grafico e conseguir tabelar o acumulo de erro. Para a prï¿½xima melhoria, irei implementar alguma alternativa para ter visualmente o caminho de cada posiï¿½ï¿½o percorrida e estimada do robï¿½. Logo se nota uma diferenï¿½a apï¿½s a execuï¿½ï¿½o por um minuto a minuto de execuï¿½ï¿½o, apï¿½s a rotaï¿½ï¿½o do robï¿½, o acumulo de diferenï¿½a da odometria e do GroundThruth da posiï¿½ï¿½o real do robo. Para a implementaï¿½ï¿½o ser finalizada, necessita de uma correï¿½ï¿½o de orientaï¿½ï¿½o usando-se algum outro enconder anexado ao robï¿½ Pioneer.

III. Robot Fuzzy Control: Wall Follow - Seguir Parede

A implementaï¿½ï¿½o da lï¿½gica fuzzy e do comportamento de seguir parede foi implementado na classe *FuzzyWallFollower*. O controle tem por objetivo manter uma distï¿½ncia de 30cm da parede direita. A entrada do sistemas sï¿½o apenas dois sensores do *Pionner* modelados de acordo com a figura 4. A figura abaixo representa uma das modelagens usadas no sensor para o controle de wall following. Pode-se ver todas as regras na classe (*FuzzyWallFollower*). As figuras abaixo somente exemplificam a modelagem implementadas nas regras de entrada e saï¿½da e os exemplos de cada ativaï¿½ï¿½o.

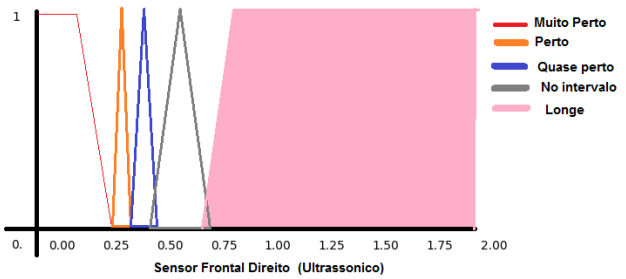

Figura 4. Modelagem sensor frontal

As saï¿½das do sistema sï¿½o a velocidade angular - figura 5 - e a velocidade linear - figura 6.

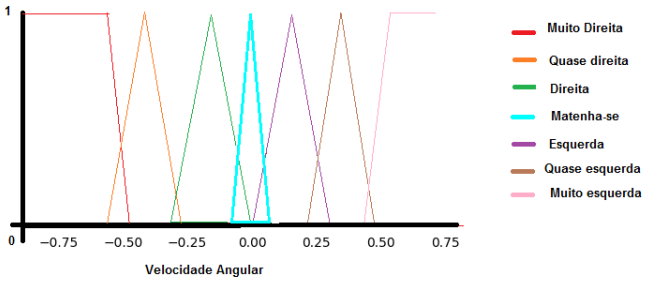

Figura 5. Modelagem Fuzzy da velocidade angular

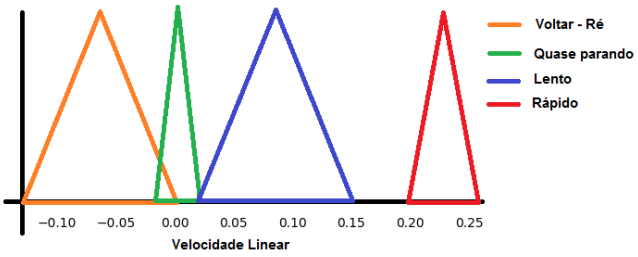

Figura 6. Modelagem velocidade linear

A. Resultados

Os resultados se mostraram se promissores quando-se usado o fuzzy com uma modelagem correta, pode-se obter precisï¿½o nos comportamentos de execuï¿½ï¿½o dos algoritmos mencionados acima. Os resultados descritos nesse trabalho apenas foram exemplificaï¿½ï¿½es das implementaï¿½ï¿½es sugeridas ao decorrer do trabalho. Para cada implementaï¿½ï¿½o, obtive um teste bï¿½sico para avaliar o aprendizado da orientaï¿½ï¿½o passada em sala de aula. Porï¿½m ainda hï¿½ alguns comportamentos a serem implementados, e o ajuste fino dessa modelagem necessita ser realizado caso queira que o comportamento do robï¿½, se adapte-se ao ambiente sugerido. Nï¿½o consegui evitar alguns problemas como as passagens pelos obstaculos do cenï¿½rio, porï¿½m consegui entender a lï¿½gica de implementaï¿½ï¿½o dos algoritmos citados e consegui exemplifica-los na linguagem de programaï¿½ï¿½o Java. Foram usados alguns artigos como leitura base de conhecimento das implementaï¿½ï¿½es realizadas ao longo da ultima dï¿½cada.Artigos [1], [2].

+————————+

Referências

[1] H. Omrane, M. S. Masmoudi, and M. Masmoudi, "Fuzzy logic based control for autonomous mobile robot navigation," *Computational Intelligence and Neuroscience*, vol. 2016, 2016. 2

[2] H. R. Beom and H. S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," *IEEE transactions on Systems, Man, and Cybernetics*, vol. 25, no. 3, pp. 464–477, 1995. 2