Seguidor de linha utilizando controlador fuzzy baseado em visão computacional

Vinícius Costa de Almeida Dias[1], **Luiz Carlos Simões Soares Junior**[1]

[1]Centro de Ciências Exatas e Tecnológicas – Universidade Federal do Recôncavo da Bahia (UFRB)
Caixa Postal 44380-000 – Cruz das Almas – BA – Brazil

viny.diaas@gmail.com, lcsimoes@ufrb.edu.br

Abstract. *War, automation of industrial processes and space exploration motivated the development of technological tools aimed at solving these problems. From this perspective it was proposed a mobile robotic system capable of taking user built paths based on computational vision. This work uses images to identify the path and provide inputs to the fuzzy controller. Tests were performed comparing the movement of the robot with the ideal path in order to evaluate controller behavior and validate the system created besides bringing improvements to the efficiency of the same.*

Resumo. *A guerra, a automação de processos industriais e a exploração espacial motivou o desenvolvimento de ferramentas tecnológicas voltadas para a solução destes problemas. Sob essa perspectiva foi proposto um sistema robótico móvel capaz de cursar caminhos construídos pelo usuário baseado em visão computacional. Este trabalho utiliza imagens para identificar o trajeto e fornecer entradas para o controlador fuzzy. Foram realizados testes comparando o movimento do robô com o trajeto ideal com o intuito de avaliar comportamento do controlador e validar o sistema criado além de trazer melhorias à eficiência do mesmo.*

1. Introdução

A intensa modernização da sociedade, competitividade, rotatividade de produtos gera um crescimento vertiginoso nos sistemas robóticos, impulsionado principalmente pela indústria que utiliza essa tecnologia como substituta de mão de obra humana [Silva et al. 2003]. Deste modo, pesquisadores têm aplicado esforços na construção de robôs moveis "inteligentes" introduzindo a capacidade de realizar ações adequadas de acordo com a situação do ambiente, sem a necessidade de intervenção humana [Howard and Seraji 2001].

O primeiro robô móvel surge durante as últimas fases da Segunda Guerra Mundial, com a criação dos misseis balísticos alemães V-1 e V-2 que possuíam um sistema de localização e controle por radar [Keane and Carr 2013]. Posteriormente à Segunda Guerra houveram diversos avanços técnicos em campos de pesquisa relacionados a robótica móvel e desde então a humanidade faz uso da tecnologia desenvolvida em diversas áreas do mercado mundial nos setores industriais, de serviço, pesquisa e entretenimento [Pereira et al. 2003].

Seguir trajetórias é uma ação difundida entre os engenheiros e pesquisadores da área, principalmente por ser empregada em ambientes estruturados, como no setor industrial [Costa et al. 2003, Pereira et al. 2003]. A utilização do sistema de visão computacional para obter esse tipo de comportamento tem sido satisfatória [Ismail et al. 2009, Elhady et al. 2014, Ramírez-Cortés et al. 2011].

O objetivo deste trabalho é a implementação de um sistema robótico capaz de percorrer caminhos construídos pelo usuário com base em sinais de entrada obtidos a partir de imagens, corrigindo sua trajetória a cada imagem, baseado em um controlador fuzzy. Esse tipo de controlador foi escolhido por apresentar uma maneira simples e eficaz na modelagem de problemas não-lineares [Zadeh 1965]. Para a construção desse sistema, será utilizado um robô educacional e programas com pacotes complementares que serão citados no decorrer do texto.

A construção dos trajetos teve como base o trabalho de [Reuss and Lee 2001]. Em seguida, foram plotados gráficos das posições percorridas pelo robô ao longo do tempo, com objetivo de avaliar o comportamento do controlador. Por fim, a validação e melhoramento do controlador fuzzy foi baseado em experiências adquiridas com a cinemática do robô no percurso [Niku 2013].

2. Materiais e Métodos

2.1. Plataforma de testes

Os testes foram realizados em um robô educacional não-holonômico, modelo QBot 2 fabricado pela Quanser®. Este robô é composto por uma plataforma de movimentação modelo Kobuki fabricada pela Yujin Robot®, um sistema de visão Kinect Xbox 360 fabricado pela Microsoft® e uma placa controladora fabricada pela Quanser® baseada no *Gumstix DuoVero*. A Quanser® disponibiliza um conjunto de ferramentas chamada *QUARC*® que possibilitam a comunicação entre o ambiente *MATLAB®/Simulink®* e o robô.

2.2. Captura e processamento das imagens

O sensor *Microsoft® Kinect* dispõe de uma câmera RGB e sensor de profundidade 3D. A câmera RGB do *Microsoft® Kinect* fornece uma imagem (matriz) RGB de *640x480 a 30fps (frames per second)*. Neste trabalho foi utilizada apenas a câmera RGB que serviu como sistema para detecção da trajetória a ser seguida.

A primeira etapa do processamento das imagens foi a segmentação do trajeto. De acordo com [Gonzalez and Woods 2009] na segmentação os melhores resultados são obtidos quando empregados os vetores de cores RGB, o artigo [Freitas et al. 2007] mostra resultados positivos dessa afirmação, por este motivo neste artigo foram utilizadas as imagens RGB.

Com base em [Gonzalez and Woods 2009], se o processo de segmentação é bem sucedido em uma imagem digital com poucos pontos é improprio usar a resolução máxima da câmera RGB, Figura 1.a. Durante percurso que o robô segue, este realizará o processamento das imagens capturadas sendo o tempo de execução desta tarefa um fator crítico para o seu sucesso [Reuss and Lee 2001, Costa et al. 2003]. Visando o menor custo computacional e maior velocidade no processamento das imagens, foi utilizado

10% da resolução da imagem original, resultando em uma matriz *68x48x3*, Figura 1.b. O que para o contexto de trajetória não foi um problema.

(a) Imagem original. (b) 10% da resolução total. (c) Imagem limiarizada.

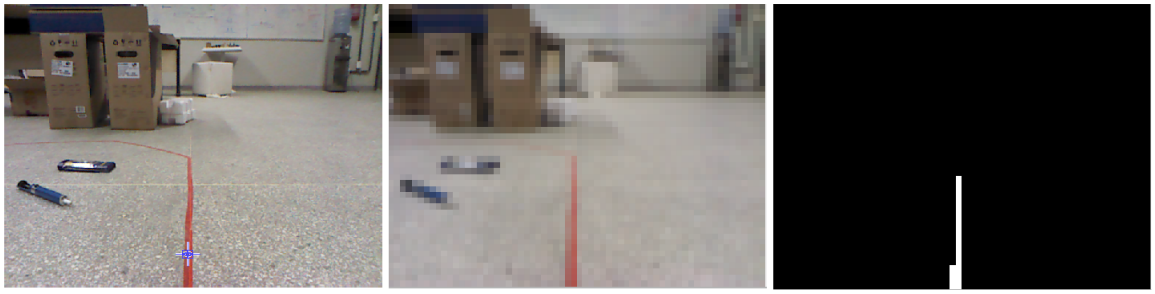


Figura 1. Imagem original, com a qualidade reduzida e após ser limiarizada.

Foi desenvolvido um algoritmo no *MATLAB*® que realiza a segmentação da imagem aplicando a técnica da limiarização, com base no padrão de cores do sensor. O algoritmo de limiarização segue as regras descritas abaixo:

$$g(x,y) = \begin{cases} 1, & \textit{se } (f_R - f_B) > RB_{min} \cap (f_R - f_G) > RG_{min} \cap |f_G - f_B| < GB_{max} \\ 0, & \textit{caso contrário} \end{cases}$$

Onde $f_{RGB}(x,y)$ é o vetor de uma imagem RGB, tal que f_R, f_B, f_G as intensidades das cores nos canal vermelho, azul e verde, respectivamente, no ponto (x,y), já RB_{min}, RG_{min}, GB_{max} são os valores de limiar, $\cap$ é a interseção que representa o operador lógico e, em que $g(x,y)$ é a imagem de saída binarizada. Buscando segmentar a imagem de acordo com as características do trajeto, de forma que a comparação é feita com três limiares (RB_{min}, RG_{min}, GB_{max}) em que f_R deverá ser maior que f_B e f_G e o módulo a subtração entre f_B e f_G, afins de garantir todas as diferenças positivas. Após a aplicação do algoritmo de limiarização, obtêm-se uma imagem binária, Figura 1.c, onde os pixels em branco representam a localização da trajetória.

2.3. Extração das características do caminho

Nesta seção serão discutidas as estratégias abordadas para extração de características do caminho que serão utilizadas como parâmetros de entrada para o controlador Fuzzy.

Foi empregando um artifício semelhante ao usado por [Costa et al. 2003] para identificar o erro em relação ao caminho a ser seguindo. Para avaliar este erro foi criada uma "*linha imaginária*" com origem no centro do robô e perpendicular ao eixo das suas rodas. Ao comparar a distância em pixels do caminho da linha de centro é obtido um erro que pode ser nulo, positivo ou negativo.

A distância entre pixels brancos na linha inferior da imagem limiarizada e a linha de centro (centro da imagem) foi chamada de "*VariaçãoErro*", esta linha representa o trecho do percurso mais próximo do robô.

Objetivando-se que o robô mantenha seu curso independe do grau de complexidade do caminho é preciso reconhecer as curvas e quinas. Observando a Figura 2, é possível notar que os pixels extremos ao longo de um trecho de reta do caminho variam pouco em relação à linha de centro, em contrapartida, nas curvas e quinas a variação é grande [Reuss and Lee 2001]. Compreendendo isso, foi desenvolvido um algoritmo no *MATLAB*® que retorna a variação dos pixels da trajetória, esta característica do caminho foi chamada de "*VariaçãoPixelExtremo*".

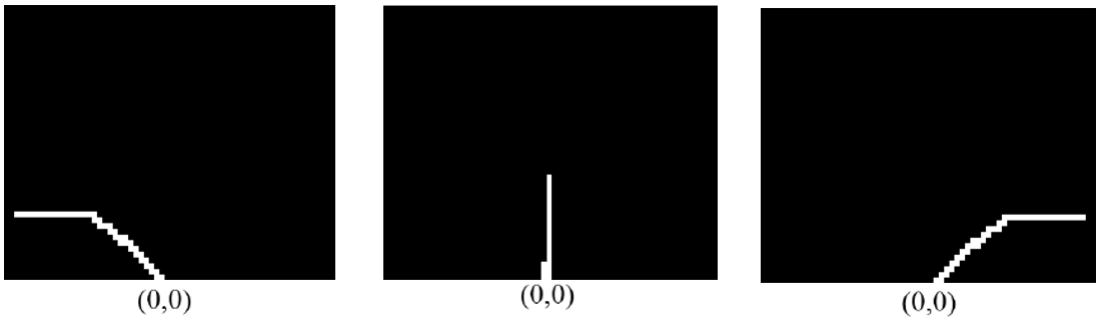


Figura 2. Três situações de variação dos pixels extremos da perspectiva da câmera.

Os parâmetros "*VariaçãoErro*" e "*VariaçãoPixelExtremo*" foram utilizados como entradas para o controlador Fuzzy.

2.4. Controle fuzzy

Foram utilizadas as entradas discutidas na Seção 2.3, que passaram por um fuzzyficador, sistema de inferência do estilo Mamdani, e as saídas processadas por um defuzzyficador usando o método do centro de gravidade.

A fuzzyficação pode ser descrita como a etapa em que são criadas variáveis linguísticas, funções de pertinência e a região do universo de discurso onde elas são definidas. Esse procedimento é realizado de forma subjetiva a partir da experiência do especialista [Niku 2013]. Deste modo, durante o processo físico de movimentação do robô, percebe-se a necessidade de utilizar duas variáveis numéricas de entrada "*VariaçãoErro*" e "*VariaçãoPixelExtremo*" e a partir da experiência adquirida foram criadas para as entradas funções de pertinência como na Figura 3.

(a) Funções da variação do erro. (b) Funções da variação dos pixels.

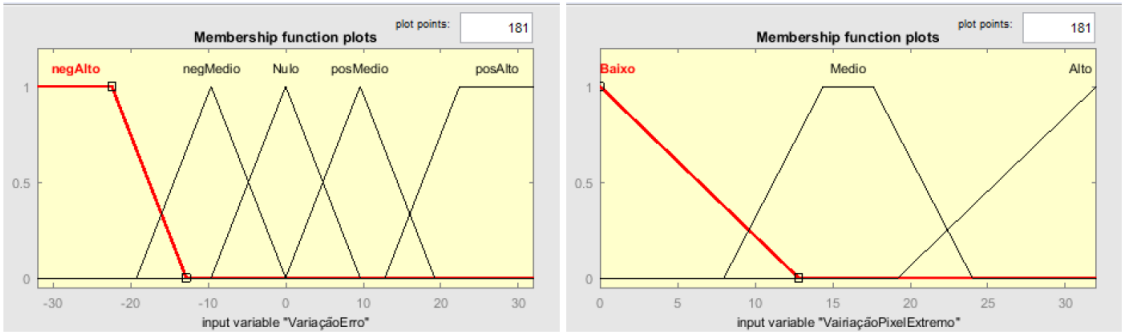
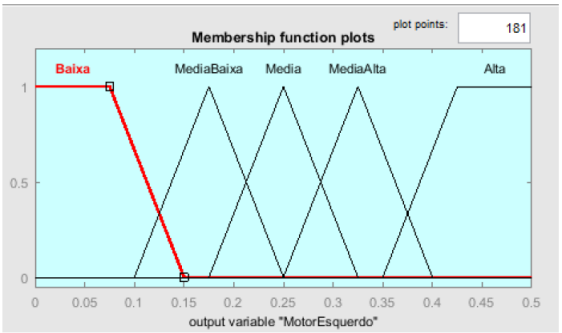


Figura 3. Funções de pertinência criadas para as entradas.

Para a ação de controle também foram criadas para as saídas funções de pertinência, como observa-se na Figura 4, associadas ao movimento das rodas esquerda e direita.

(a) Funções da resposta do motor esquerdo.



(b) Funções da resposta do motor direito.

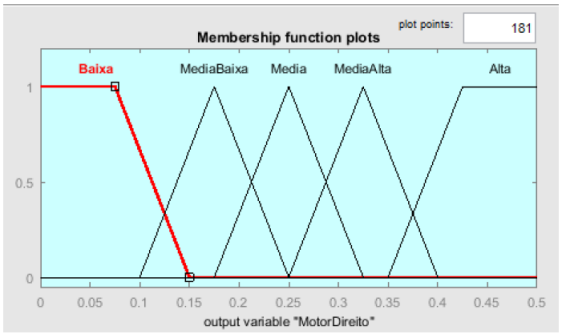


Figura 4. Funções de pertinência criadas para as saídas.

Objetivando-se simplicidade e desempenho foram criadas para as entradas 8 funções pertinência Figura 3, e 10 funções para as saídas, Figura 4. Vale ressaltar que, não existe limite para criação dessas funções, quanto maior for o número de funções criadas, mais preciso é o controlador fuzzy, em compensação o custo computacional também será elevado [Shaw 1999].

Na etapa seguinte é criado o bloco de inferência fuzzy, observe a Figura 5, no qual as proposições (regras) são definidas, que no modelo de Mamdani usa preposições condicionais, de forma que a análise do antecedente resulta no consequente. As regras foram construídas com base no conhecimento do especialista (autor do presente trabalho) sobre o problema.

1. If (VariaçãoErro is posAlto) and (VairiaçãoPixelExtremo is Baixo) then (MotorDireito is Baixa)(MotorEsquerdo is Media) (1)
2. If (VariaçãoErro is posAlto) and (VairiaçãoPixelExtremo is Medio) then (MotorDireito is Baixa)(MotorEsquerdo is Media) (1)
3. If (VariaçãoErro is posAlto) and (VairiaçãoPixelExtremo is Alto) then (MotorDireito is Baixa)(MotorEsquerdo is MediaBaixa) (1)
4. If (VariaçãoErro is posMedio) and (VairiaçãoPixelExtremo is Baixo) then (MotorDireito is Media)(MotorEsquerdo is MediaAlta) (1)
5. If (VariaçãoErro is posMedio) and (VairiaçãoPixelExtremo is Medio) then (MotorDireito is MediaBaixa)(MotorEsquerdo is Media) (1)
6. If (VariaçãoErro is posMedio) and (VairiaçãoPixelExtremo is Alto) then (MotorDireito is Baixa)(MotorEsquerdo is MediaBaixa) (1)
7. If (VariaçãoErro is Nulo) and (VairiaçãoPixelExtremo is Baixo) then (MotorDireito is Alta)(MotorEsquerdo is Alta) (1)
8. If (VariaçãoErro is Nulo) and (VairiaçãoPixelExtremo is Medio) then (MotorDireito is Media)(MotorEsquerdo is Media) (1)
9. If (VariaçãoErro is Nulo) and (VairiaçãoPixelExtremo is Alto) then (MotorDireito is Baixa)(MotorEsquerdo is Baixa) (1)
10. If (VariaçãoErro is negMedio) and (VairiaçãoPixelExtremo is Baixo) then (MotorDireito is MediaAlta)(MotorEsquerdo is Media) (1)
11. If (VariaçãoErro is negMedio) and (VairiaçãoPixelExtremo is Medio) then (MotorDireito is Media)(MotorEsquerdo is MediaBaixa) (1)
12. If (VariaçãoErro is negMedio) and (VairiaçãoPixelExtremo is Alto) then (MotorDireito is MediaBaixa)(MotorEsquerdo is Baixa) (1)
13. If (VariaçãoErro is negAlto) and (VairiaçãoPixelExtremo is Baixo) then (MotorDireito is Media)(MotorEsquerdo is Baixa) (1)
14. If (VariaçãoErro is negAlto) and (VairiaçãoPixelExtremo is Medio) then (MotorDireito is Media)(MotorEsquerdo is Baixa) (1)
15. If (VariaçãoErro is negAlto) and (VairiaçãoPixelExtremo is Alto) then (MotorDireito is MediaBaixa)(MotorEsquerdo is Baixa) (1)

Figura 5. Conjunto de regras baseados no estilo Mamdani.

O *MATLAB*® permite que as regras da Figura 5 sejam simuladas, de forma que, o processo de defuzzyficação é mostrado através de gráficos assim como com a resposta defuzzyficada correspondente ao método escolhido (centroide) [Lee 1990, Mamdani 1976].

2.5. Criação do sistema de controle no *Simulink*

Os blocos do modelo proposto neste trabalho podem ser visualizados na Figura 6.

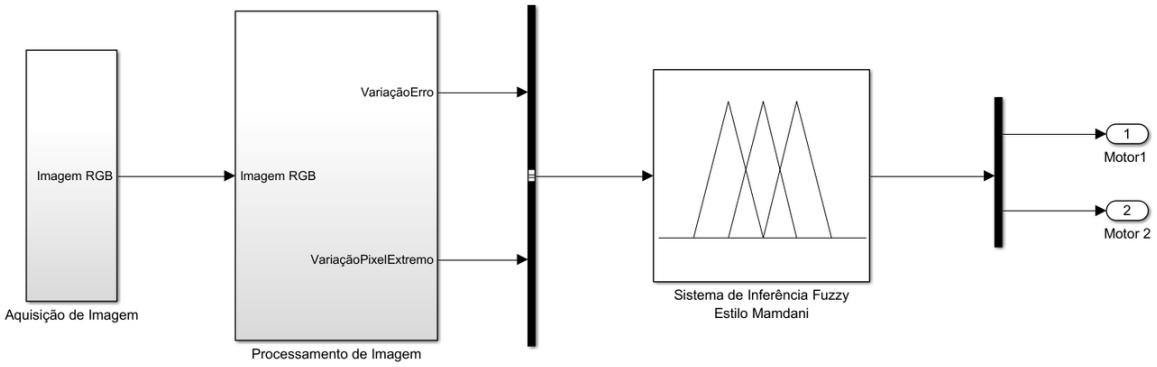


Figura 6. Blocos do modelo criado no simulink para movimentação do robô Qbot 2.

De forma que, subsistema de *aquisição de imagem* é responsável por adquirir a imagem do sensor, sendo a entrada do próximo subsistema, *processamento de imagem*, em que a imagem será segmentada e analisada para gerar saídas físicas que são utilizadas pelo *sistema de inferência fuzzy* no bloco posterior e por fim fornecer a resposta para os dois motores.

2.6. Trajetória

O método para construção das trajetórias teve como base o trabalho de [Reuss and Lee 2001] em que foram construídos trajetos com variadas angulações. Buscando um resultado semelhante, padrões geométricos foram construídos no computador e projetados no chão a partir de um projetor de vídeo acoplado na extremidade livre de uma viga fixada em balanço, essa montagem pode ser vista na Figura 7.

Figura 7. Montagem realizada para projetar padrões geométricos.

É importante notar que, todos os trajetos foram construídos contendo uma cor primária com base no cubo do modelo RGB. O proposito principal desse método é facilitar a identificação da trajetória simplificando a segmentação da imagem.

Em contraposição, não é uma tarefa trivial reproduzir de forma fiel, um pixel colorido de imagem digital, sem que haja um controle rigoroso do ambiente [Marques Filho and Neto 1999]. Mesmo assim, a construção dos caminhos sempre alvejava a cor vermelho puro (totalmente intenso) Figura 8.a, idealmente representado pelo vetor RGB da forma [255, 0, 0], em que só existe intensidade no canal 1, mas como mostra a Figura 8.b isso não acontece, o que já era esperado, devido às condições do ambiente, cor inferior da fita e das ferramentas usadas para construir o caminho.

(a) Imagem de parte trajeto construído com fita. (b) Região do trajeto ampliada (quadro azul Figura 19.a).

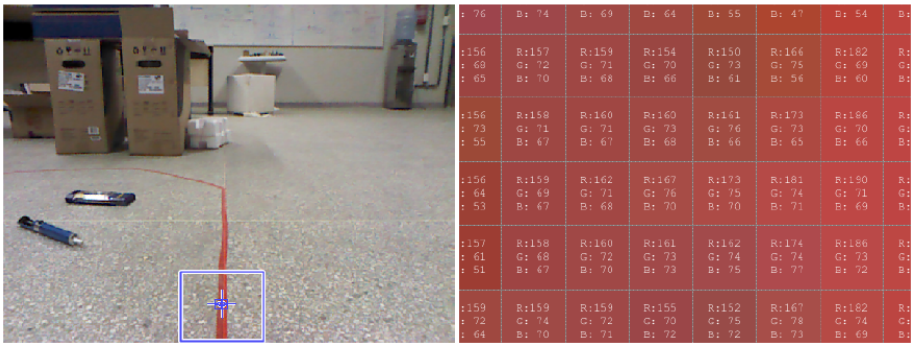
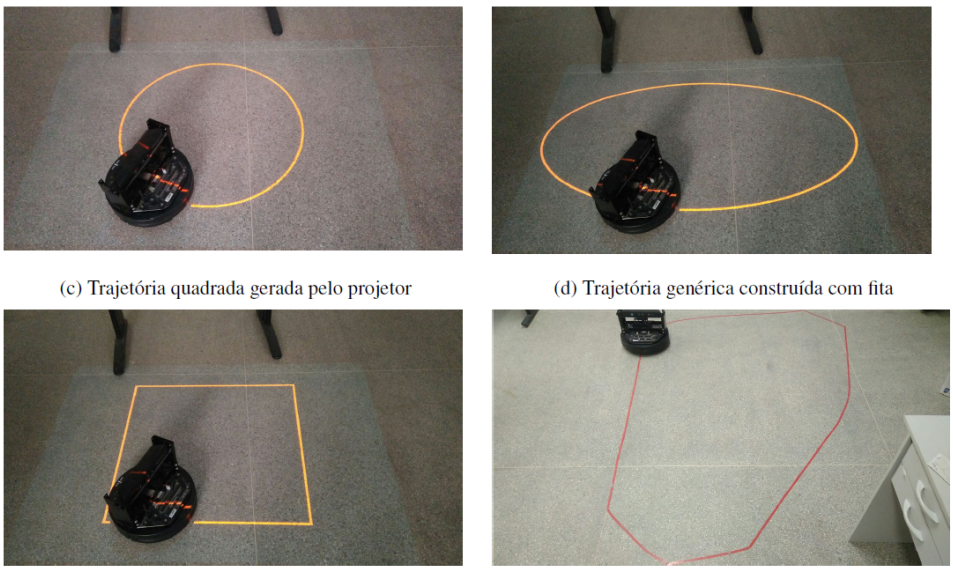


Figura 8. Imagem do trajeto sendo ampliada em uma região.

3. Resultados e Discussões

Para avaliar a eficiência do sistema implementado foi definido um tempo de 120 segundos para a simulação de cada percurso, avaliando diferentes tipos de trajetórias citadas na Seção 2.6. Tais trajetos são observados na Figura 9 logo a seguir:

(a) Trajetória circular gerada pelo projetor (b) Trajetória elíptica gerada pelo projetor



(c) Trajetória quadrada gerada pelo projetor (d) Trajetória genérica construída com fita

Figura 9. Exemplo de trajetórias construídas.

Durante as simulações dois parâmetros foram analisados: (1) a eficiência do robô em seguir o trajeto, (2) a semelhança da trajetória real, com os gráficos das posições X e Y em relação ao tempo. Foi utilizado um método odométrico para localizar as posições do robô, mas esse não é o foco deste trabalho. A partir das posições X e Y foi possível plotar gráficos dos caminhos construídos de acordo com a Figura 9 e da posição do robô em relação ao tempo, conforme a Figura 10.

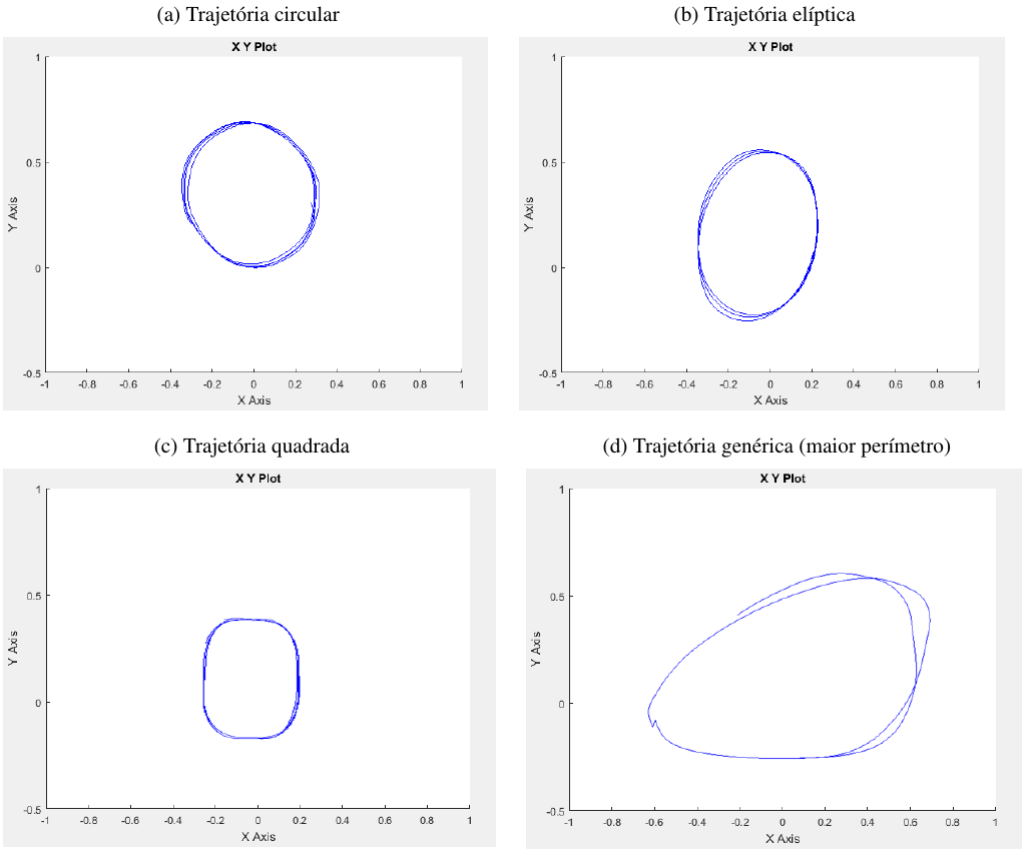


Figura 10. Gráficos das posições do robô X,Y em relação ao tempo.

No decorrer dos testes, o sistema implementado apresentou respostas satisfatórias, pois, durante o tempo de simulação o robô se manteve no percurso, sendo capaz de seguir qualquer caminho construído pelo usuário. Os gráficos da Figura 10 possuem forte semelhança com seus respectivos trajetos criados na Figura 9, esses dois fatores evidenciam a eficiência do algoritmo implementado. Isso se deve ao fato de uma segmentação bem sucedida que gera entradas precisas da posição do robô e variações do percurso para o controlador fuzzy.

Segundo [Marques Filho and Neto 1999, Costa et al. 2003] a iluminação é um fator que deve receber atenção, e mesmo não sendo foco deste trabalho, o método de segmentação apresentou resultados positivos quando houvera variações na intensidade da iluminação no local de trabalho do robô, quando utilizava o caminho feito com a fita. Entretanto, nos trajetos gerados a partir do projetor, em condições de alta luminosidade, o método de segmentação deixou a desejar, sendo necessário uma recalibração do sistema para que o mesmo voltasse a funcionar.

Por outro lado, em conformidade com os trabalhos de [Reuss and Lee 2001, Costa et al. 2003], o aumento da velocidade e baixo poder de processamento são fatores que geram entradas espúrias para o controlador fuzzy, fazendo com que o robô responda de forma desatualizada e brusca as variações do trajeto. Esse problema foi resolvido reduzindo a velocidade do robô, dado que o objetivo é seguir uma trajetória com eficiência, sendo o aumento da velocidade um possível aprimoramento para trabalhos futuros.

4. Conclusão

O desenvolvimento do presente trabalho possibilitou implementar, em uma plataforma de testes móvel, um controle fuzzy capaz de manter o percurso em trajetos que não são previamente conhecidos, com base no processamento de sinais obtidos a partir de imagens coletadas pelo sensor. Dentro deste contexto, constata-se a grande importância do estudo da robótica móvel, sendo esta uma das áreas capaz de proporcionar conforto e desenvolvimento a sociedade, desde a criação de aspiradores de pó robóticos até robôs com sistemas mais robustos utilizados em missões planetárias.

O algoritmo empregado na identificação de linha se mostrou poderoso devido aos resultados positivos na segmentação da imagem, exceto nos trajetos projetados, uma vez que os projetores utilizam uma luz brilhante para projetar o caminho, o que conflita com o algoritmo que foi desenvolvido visando extrair o máximo do padrão do sensor do robô. O comportamento descrito pelo controlador fuzzy foi satisfatório, pois, o robô se manteve no percurso, sendo capaz de seguir qualquer caminho construído, todavia por possuir um hardware limitado não foi possível operar o robô em velocidade máxima.

Pode-se, futuramente, buscar métodos mais sofisticados de segmentação, assim como o melhoramento da velocidade do sistema, através de novas técnicas e hardware mais específico para o trabalho, uma consequência disso será possibilidade da ampliação das regras fuzzy, resultando em um sistema com maior precisão. O estudo do papel da luminosidade no processamento digital de imagens poderia trazer melhorias nos resultados.

Referências

Costa, E. R., Gomes, M. L., and Bianchi, R. A. (2003). Um mini robô móvel seguidor de pistas guiado por visao local. *VI Simpósio Brasileiro de Automação Inteligente, Bauru, Brasil*, pages 175–186.

Elhady, W. E., Elnemr, H. A., and Selim, G. (2014). Implementation and evaluation of image processing techniques on a vision navigation line follower robot.

Freitas, R. F., Costa, R. C. S., da Silva Barros, A. C., da Silva Siqueira, R., Cortez, P. C., and Soares, J. M. (2007). Algoritmos para segmentaçao da pele utilizando modelos de cores rgb em ambiente matlab/simulink. *Conexões-Ciência e Tecnologia*, 1(1):65–71.

Gonzalez, R. C. and Woods, R. C. (2009). *Processamento digital de imagens*. Pearson Educación.

Howard, A. and Seraji, H. (2001). An intelligent terrain-based navigation system for planetary rovers. *IEEE Robotics & Automation Magazine*, 8(4):9–17.

Ismail, A., Ramli, H., Ahmad, M., and Marhaban, M. (2009). Vision-based system for line following mobile robot. In *Industrial Electronics & Applications, 2009. ISIEA 2009. IEEE Symposium on*, volume 2, pages 642–645. IEEE.

Keane, J. F. and Carr, S. S. (2013). A brief history of early unmanned aircraft. *Johns Hopkins APL Technical Digest*, 32(3):558–571.

Lee, C.-C. (1990). Fuzzy logic in control systems: fuzzy logic controller. i e ii. *IEEE Transactions on systems, man, and cybernetics*, 20(2):404–418.

Mamdani, E. H. (1976). Application of fuzzy logic to approximate reasoning using linguistic synthesis. In *Proceedings of the sixth international symposium on Multiple-valued logic*, pages 196–202. IEEE Computer Society Press.

Marques Filho, O. and Neto, H. V. (1999). *Processamento digital de imagens*. Brasport.

Niku, S. B. (2013). *Introdução a robótica*. Grupo Gen-LTC.

Pereira, J. et al. (2003). Avaliação e correção do modelo cinemáticio de robôs móveis visando a redução de erros no seguimento de trajetórias.

Ramírez-Cortés, J., Gómez-Gil, P., Martínez-Carballido, J., and López-Larios, F. (2011). Sistema de navegación autónoma de un vehículo usando visión robótica y control difuso en labview. *Ingeniería, investigación y tecnología*, 12(2):129–136.

Reuss, R. F. and Lee, T.-M. (2001). Fuzzy logic control in a line following robot. *Class Notes of University of Nevada at Reno*.

Shaw, I. S. (1999). *Controle e modelagem fuzzy*. Edgard Blucher.

Silva, L. R. d. et al. (2003). Análise e programação de robôs móveis autônomos da plataforma eyebot.

Zadeh, L. A. (1965). Information and control. *Fuzzy sets*, 8(3):338–353.