Sistema de navegação para um robô móvel usando um sensor kinect e um controlador Fuzzy

Marcelo Mafalda[1], Nicoli Vieira Rodrigues[1], Rodrigo Mattos da Silva[1], Thiago Rodrigues Garcia[1], Marco Antonio de Souza Leite Cuadros[2], Rodrigo da Silva Guerra[1,] Andrei Piccinini Legg[1], Daniel Fernando Tello Gamarra[1]

[1] Universidade Federal de Santa Maria (UFSM)
GARRA – Grupo de Automação e Robótica Aplicada
97.105-900 – Santa Maria – RS – Brasil
[2] Instituto Federal do Espírito Santo 2, IFES-Serra
29173-087 - Serra, Espírito Santo Brasil
rodrigomattos@hotmail.com.br, thiago.rgarcia@hotmail.com,
marcoantonio@ifes.edu.br, andrei.legg@gmail.com, daniel.gamarra@ufsm.br,

Resumo. Este artigo apresenta o desenvolvimento de um sistema de navegação em ambientes internos para um robô móvel de tração diferencial. O sistema de navegação permite ao robô deslocar-se de uma pose inicial até uma pose final dentro de um ambiente *indoor* previamente mapeado, com o auxílio das imagens obtidas com um sensor Kinect, do *software* ROS e da biblioteca Rtab-Map. O sistema utiliza um controlador *Fuzzy* para o rastreamento da trajetória. Experimentos são ilustrados para validar o sistema proposto.

Keywords: *kinect*, *rgbd-slam*, *fuzzy*, *robotics*.

1 Introdução

Para realização de tarefas de forma autônoma, ou seja, sem a intervenção humana, os robôs precisam, dentre outras coisas, de uma capacidade de navegação. Conceitualmente, o termo navegar, na robótica, consiste em guiar um robô em seu espaço de trabalho por um caminho, de forma a levá-lo de uma posição e orientação iniciais até uma posição e orientação finais [1]. De maneira geral, a navegação é uma das tarefas mais complexas na problemática da locomoção dos robôs, pois deve integrar subtarefas fundamentais como o mapeamento e a localização.

O crescimento explosivo de sensores de imageamento e de navegação tornou o mapeamento tridimensional (3D) de ambientes internos uma tarefa essencial em inúmeras aplicações de engenharia, tais como: sistemas de localização e mapeamento simultâneo, conhecidos como *SLAM* (*Simultaneous Localization and Mapping*); vigilância e gerenciamento de emergências; navegação e posicionamento; inspeção de infraestruturas, modelagem e reconstrução 3D; cartografia, etc. Em novembro de 2011 o Google Maps lançou seu projeto de mapeamento 3D de ambientes internos que permite qualquer usuário de smartphone acessar mapas on-line de ambientes públicos. Esses mapas podem ser facilmente acessados por tecnologia *Wi-Fi*, *Bluetooth*, *RFID (Radio-Frequency Identification)*, sinais de televisão, etc. Recentemente, o Google lançou seu projeto mais arrojado em mapeamento 3D de ambientes internos denominado *Projeto Tango* [11]. Um smartphone equipado com sensores de imageamento *RGB* e de profundidade (D) é a base do segredo da empresa. O equipamento permite que usuários detentores da tecnologia possam construir seus próprios mapas tridimensionais em qualquer ambiente interno [7].

Neste artigo é apresentada a estrutura de um robô móvel e um sistema de navegação para o robô. O sistema é composto por um sensor kinect, juntamente com o sistema operativo ROS (*Robotic Operating System*), abordagens do SLAM (*Simultaneous Localization and Mapping*) e um controlador *fuzzy* para rastreamento da trajetória. O sistema de navegação permite ao robô deslocar-se de uma pose inicial até uma pose final, dentro de um ambiente interno. Enquanto o robô se locomove, realiza simultaneamente, através de um sensor kinect, a representação do ambiente em um mapa de grade de ocupação 2D. Com esses dados o controlador *fuzzy* faz com que o robô execute a trajetória.

Entre os trabalhos anteriores relacionados ao modelo de aplicação utilizado neste trabalho, podemos mencionar o trabalho de Pegas [6], que controla um robô móvel com rodas utilizando informações visuais captadas pelo Microsoft Kinect. A tarefa de controle desse projeto consiste em fazer com que o robô mantenha uma distância constante a um determinado alvo móvel. Em [7] é apresentado o desenvolvimento de um sistema de navegação em ambientes internos para o robô pessoal chamado MARIA. O sistema de

navegação permite ao robô, ao receber instruções de voz, deslocar-se de uma pose inicial até uma pose final dentro de um ambiente interno previamente mapeado.

Trabalhos relacionados ao assunto foram tomados como referência para o desenvolvimento deste artigo como o [5], no qual, utilizando visão computacional obtida pelo sensor kinect, um robô é capaz de mapear e analisar a trajetória de uma escada, para assim movimentar-se nela de maneira correta e manter o equilíbrio com o uso do controlador fuzzy. Blakouti [2] utiliza o sensor kinect e o controlador fuzzy na abordagem de cadeiras de rodas autônomas, usando a visão do sensor para evitar obstáculos e o controlador para traçar a trajetória em tempo real. Por fim, o trabalho [3] implementa um sistema de rastreamento para um robô móvel utilizando o sensor Kinect e o controlador para lógica difusa e redes neurais, com o objetivo de obter um robô multiagente trabalhando de forma autônoma ao ar livre.

Assim, este artigo está divido em 6 seções. A primeira seção é uma breve introdução, a segunda seção descreve os fundamentos teóricos, a terceira seção apresenta o controlador *fuzzy*, a quarta seção aborda os dispositivos e softwares usados neste experimento, a quinta seção apresenta graficamente os resultados obtidos e a seção 6 explicita as conclusões do artigo

2 Cinemática do Robô Móvel

A configuração do robô móvel utilizado neste trabalho é mostrada na Fig. 1. O robô possui duas rodas paralelas e uma roda livre para equilibrar a estrutura do robô. As rodas utilizadas neste projeto são definidas como rodas convencionais, cuja velocidade no ponto de contato da roda com o solo é zero. Elas são classificadas quanto à orientação, podendo ser fixas, centradas orientáveis e centradas não orientáveis, estas últimas também são conhecidas como castor.

O robô possui duas rodas independentes de tração diferencial e uma roda de movimento livre (Castor) para manter o equilíbrio da estrutura.

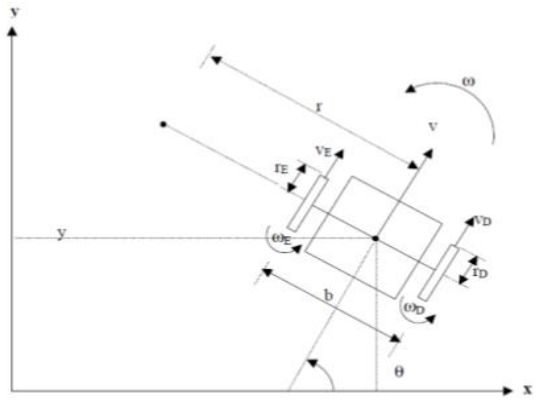

Fig. 1. Modelo cinemático de um robô móvel com tração diferencial.

Podemos definir as seguintes variáveis da figura:
(x, y) = Posição do referencial fixo no robô em relação ao referencial fixo no espaço de trabalho.
θ = Ângulo de orientação do robô em relação ao referencial fixo no espaço de trabalho.
b = Comprimento do eixo.
r = Raio de giro do robô.
r_d (r_e) = Raio da roda direita (e esquerda).
ω = Velocidade angular do robô.
ω_d (ω_e) = Velocidade angular da roda direita (e esquerda).
v = Velocidade linear do robô.
v_d (v_e) = Velocidade linear da borda da roda direita (e esquerda).

$$\left[\frac{\omega}{v}\right] = \left[\frac{(r_d\ /\ 2)\ (r_e\ /\ 2)}{(r_d\ /\ b) - (r_e\ /\ b)}\right] \star \left[\frac{\omega_d}{\omega_e}\right] \qquad (1)$$

As coordenadas (x, y) e θ dão a posição do robô no plano cartesiano. É relevante observar que, caso as velocidades das rodas forem iguais ($v_d = v_e$), o robô percorrerá uma linha reta. Se as velocidades forem diferentes e a velocidade da roda direita for maior que a da roda esquerda ($v_d > v_e$), o robô descreverá uma trajetória circular no sentido anti-horário. De outro modo, quando a velocidade da roda esquerda for maior que a da roda direita ($v_e > v_d$), o robô percorrerá uma trajetória circular no sentido horário [4].

3 Controlador Fuzzy

O controlador fuzzy permite ao robô mover-se de um ponto a outro. O controlador fuzzy tem como objetivo minimizar o ângulo ɣ e a distância entre as duas retas e as regras fuzzy são representadas por duas entradas e duas saídas. O modelo básico do controlador conhecido na literatura e as regras fuzzy foram fundamentadas em Motta [9].

O controlador recebe a distância (D) e a diferença de ângulo (ɣ) entre o robô e o ponto de destino e retorna à velocidade linear e à velocidade angular necessárias para minimizar, respectivamente, a distância e a diferença de ângulo. Na Fig. 2, pode-se observar o gráfico representando o centro de massa do robô móvel (x,y) e o ponto que deseja-se alcançar (x_r , y_r). Onde ɣ = α – θ [4].

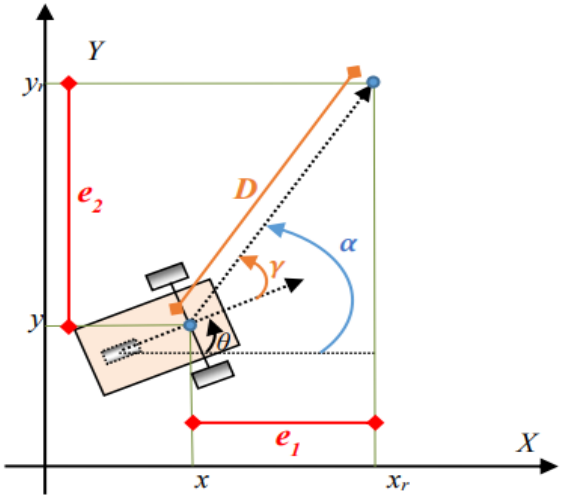

Fig. 2. Distância entre o centro de massa do robô e o ponto de destino. Fonte: Motta [9].

As regras fuzzy abordadas fundamentam-se totalmente nesse gráfico. Se ɣ for positivo, ou seja, α > θ, o robô precisará virar para à esquerda para aumentar θ, minimizando a diferença entre os ângulos e vice-versa. Se o ângulo ɣ for nulo, o robô deverá percorrer uma reta.

Tabela 1. Regras *Fuzzy*.

		ɣ				
		NB	**NS**	**Z**	**PS**	**PB**
D	**VS**	VS (RF)	VS (RS)	VS (Z)	VS (LS)	VS (LF)
	S	S (RF)	S (RS)	S (Z)	S (LS)	S (LF)
	M	M (RF)	M (RS)	M (Z)	M (LS)	M (LF)
	B	B (RF)	B (RS)	B (Z)	B (LS)	B (LF)
	VB	VB (RF)	VB (RS)	VB (Z)	VB (LS)	VB (LF)

Tabela 2. Variáveis *Fuzzy.*

Variável	Valores	Descrição
ɣ	NB	Negative Big
	NS	Negative Small
	ZE	Zero
	PS	Positive Small
	PB	Positive Big
D e v	VS	Very Small
	S	Small
	M	Medium
	B	Big
	VB	Very big
ω	LF	Left Fast
	LS	Left Small
	Z	Zero
	RS	Right Slow
	RF	Right Fast

A partir da velocidade máxima que cada roda pode assumir, é possível calcular a velocidade angular (ω) e linear (v) do robô. Desse modo, vel_e e vel_d são, respectivamente, a velocidade da roda esquerda e da roda direita, da seguinte forma:

$$v = \frac{r}{2} \times (vel_d + vel_e) \quad (2)$$

$$\omega = \frac{r}{b} \times (vel_d - vel_e) \quad (3)$$

Os intervalos de valores que as variáveis de entrada e saída podem assumir são descritos na Tabela 3 (distância de 0 a 2 metros).

Tabela 3. Intervalo de Valores das Variáveis *Fuzzy.*

Distância (D)	0 ~ 2
Distância de ângulo (ɣ)	-3.14 ~ 3.14
Velocidade Linear do Robô (v)	0 ~ 7.6
Velocidade Angular do Robô (ω)	-15.6 ~ 15.6

4 Setup Experimental

4.1 Sensor kinect

O sensor Kinect é uma câmera do tipo RGB-D utilizada como controlador para jogos. Esse sensor foi desenvolvido pela Microsoft para ser utilizado no vídeo game Xbox

360. O Kinect tem cerca de 23 cm de comprimento – como visto na Fig. 3 – e é composto pelos seguintes recursos:

• Uma câmera RGB (Resolução 640x480, 30fps VGA) para codificar vídeo e imagens digitais.

• Um sensor de profundidade 3D que permite que o acessório escaneie o ambiente a sua volta em três dimensões, composto de um projetor de luz infravermelha e uma câmera infravermelha com por um sensor CMOS monocromático de luz infravermelha (Resolução 640x480, 30fps).

• Quatro Microfones (16bit taxa de amostragem: 16KHz), utilizados para reconhecimento de voz.

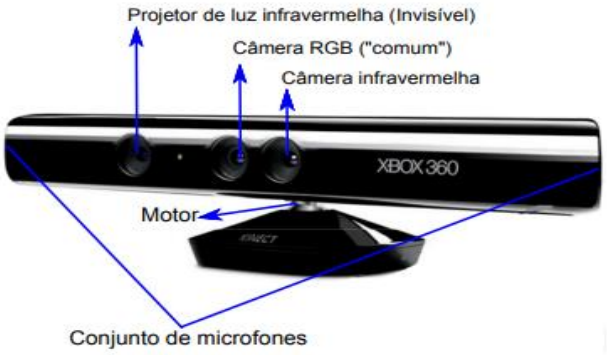

Fig. 3. Componentes de um sensor Kinect. Fonte: http://nti.ceavi.udesc.br/.

O funcionamento desse dispositivo está restrito ao alcance da visão do sensor de profundidade. Contudo, existem faixas nas quais o sensor não consegue registrar as imagens em 3D adequadamente. A seguir é apresentada a tabela com as principais características do equipamento [7].

Tabela 4. Tabela de características técnicas do sensor Kinect.

Kinect	**Especificações**
Ângulo de Visão	43º vertical e 57 º horizontal
Faixa de Inclinação Vertical	$\pm$ 27º
Taxa de Frames	30 ftps (frames por segundo)

Para melhor aproveitamento das imagens feitas pelo sensor, a distância do kinect do solo deve ser igual a 60 cm como é representado na Fig. 4.

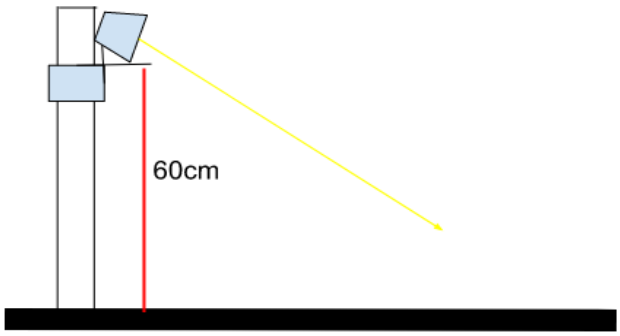

Fig. 4. Posição do Kinect e orientação.

4.2 Robô móvel

O robô móvel utilizado nos experimentos foi projetado mecanicamente no laboratório de robótica GARRA. O mesmo possui 2 motores de corrente contínua capazes de suportar toda a estrutura e executar trajetórias pré-programadas. Dois motores CC têm acoplados seus respectivos encoders do tipo incremental para leituras de distância e posição e cada encoder gera 980 pulsos elétricos por rotação.

O elemento responsável por abrigar toda a programação, ler os dados dos sensores e controlar os demais dispositivos do robô é o microcontrolador Arduino Mega 2560. O microcontrolador obtém as leituras dos encoders do driver MD49 e também a leitura dos sensores de ultrassom, entre outras tarefas de baixo nível. O robô é mostrado na Fig. 5.

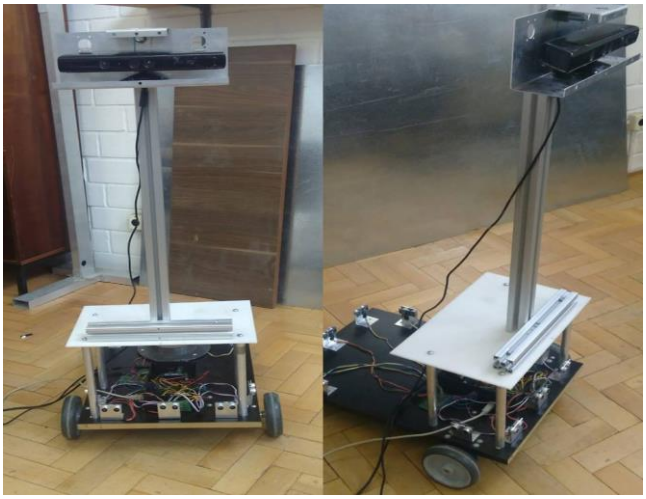

Fig. 5. Robô Poter

4.3 ROS (Robotic Operating System)

O ROS é um metassistema operacional que fornece bibliotecas e ferramentas para ajudar os desenvolvedores de software a criarem aplicações para robôs. Ele fornece abstrações para diversos recursos, tais como: controle de acesso a dispositivos; implementação de funcionalidades genéricas de controle; trocas de mensagens entre processos; sincronismo e gerenciamento de pacotes, etc. O ROS é livre e de código aberto tendo sua distribuição regida pela licença BSD (*Berkeley Software Distribution*). Dentre os pacotes que compõem a plataforma ROS, os que estão sendo utilizados neste trabalho são Freenect, um pacote que contém arquivos de inicialização para usar com o Microsoft Kinect usando a biblioteca libfreenect. Esta pasta replica a API oferecida pelo openni_launch em um esforço para manter a máxima compatibilidade com o driver OpenNI [8]. Optamos por usar esta biblioteca, pois o modelo kinect usado é o 1473, não é compatível com o pacote OpenNi segundo testes realizados neste trabalho e em pesquisas relacionadas.

4.4 RTAB-MAP

O RTAB-Map – Mapeamento baseado em aparência em tempo real – é uma abordagem SLAM RGB-D baseada em um detector de fechamento de loop bayesiano global. O detector de fechamento de loop usa uma abordagem *bag-of-words* [10] para determinar a probabilidade de uma nova imagem vir de um local anterior ou de uma nova localização. Quando uma hipótese de encerramento de loop é aceita, uma nova restrição é adicionada ao gráfico do mapa, então um otimizador de gráfico minimiza os erros no mapa. Uma abordagem de gerenciamento de memória é usada para limitar o número de locais utilizados para detecção de fechamento de loop e otimização de gráfico, de modo que as restrições em tempo real em ambientes em larga escala são sempre respeitadas.

O RTAB-Map pode ser utilizado sozinho com uma câmera Kinect, estéreo de mão para o mapeamento RGBD de 6DoF ou em um robô equipado com um *range finder* a laser para o mapeamento 3D [8]. Para essa aplicação foi instalado o Sistema Operacional para Robôs (ROS) no Ubuntu 16.04, juntamente com mais alguns pacotes para rodar as aplicações em um computador acoplado ao robô. Essa seção do trabalho tem a finalidade de, a partir da análise das imagens, determinar a localização do robô móvel no espaço e a localização do destino no qual deseja-se que o robô se encontre. A captura das imagens ocorre através do sensor kinect trabalhando juntamente com ROS e RTAB-Map. O processamento e análise das imagens é feita com a biblioteca de visão computacional OpenCV em um computador acoplado ao robô. Para que o algoritmo possa identificar a posição do robô, os dados de odometria são lidos de um arquivo da aplicação RTAB-Map em funcionamento e enviados ao programa do sistema de navegação desenvolvido na linguagem python.

5 Metodologia

A partir da análise obtida pelo sensor Kinect, é determinada a localização do robô móvel no espaço e a localização do destino no qual deseja-se que o robô se encontre. A captura das imagens ocorre através do sensor kinect trabalhando com ROS e RTAB-Map. Para

que o algoritmo possa identificar a posição do robô, os dados de odometria visual são lidos de um arquivo da aplicação RTAB-Map em funcionamento e enviados ao sistema de navegação na linguagem python. Para o processamento e análise das imagens é usada a biblioteca de visão computacional freenect em um computador acoplado ao robô. O sistema funciona de maneira que os dados obtidos pelo sensor kinect são processados pelo software Rtab-Map e pela biblioteca ROS citada. Desta maneira é realizado o mapeamento da nuvem de pontos das imagens (figuras 6 e 7) para assim construir o mapa 2D do ambiente (figura 9). Para determinar e controlar a trajetória, usa-se o controlador fuzzy desenvolvido e descrito no item 3 deste artigo, no qual são informados os dados da distância a ser percorrida de um ponto a outro.

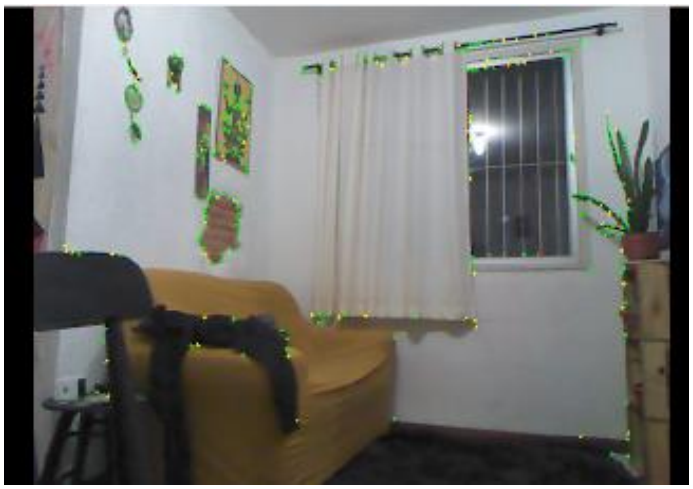

Fig. 6. Imagem de odometria (nuvem de pontos) Rtab-Map.

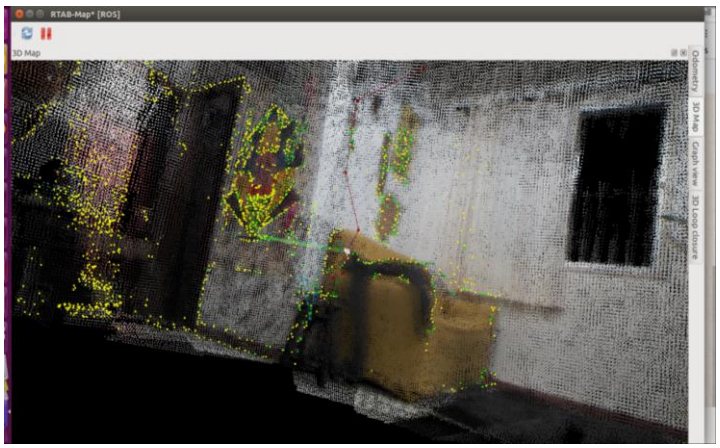

Fig. 7. Mapa 3D Rtab-Map.

O programa principal se comunica com o kinect ao receber distâncias dos obstáculos mais próximos e mostra a posição e a orientação do kinect. A distância do kinect e do solo deve ser igual a 60 cm. O sistema é representado em três partes principais: o notebook, o kinect e o controlador fuzzy. Na imagem da Fig. 8, é representado o funcionamento do sistema.

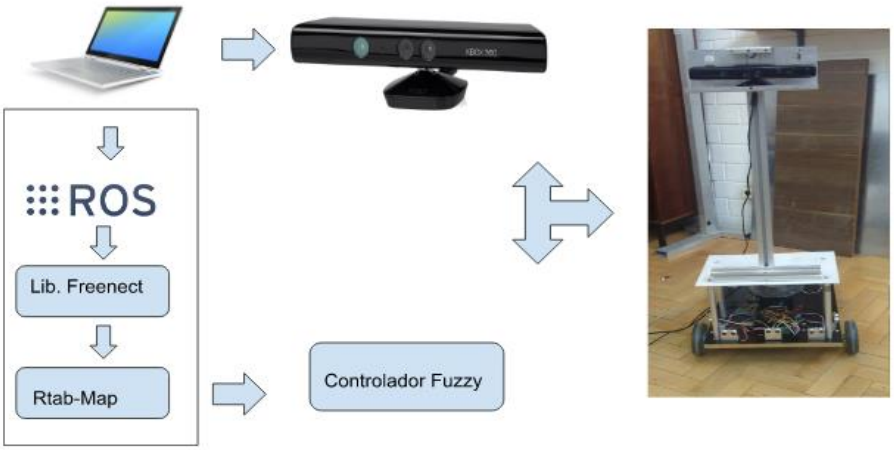

Fig. 8. Esquema de Funcionamento do sistema de navegação para robô móvel.

6 Resultados

A partir da estrutura do robô previamente montada, juntamente com o controlador fuzzy descrito no item 3 e as aplicações computacionais acima citadas, foi possível cumprir o objetivo principal deste trabalho que é fazer o robô locomover-se sozinho dentro de uma trajetória proposta de um ponto a outro.

Foram feitos diversos experimentos com a finalidade de testar a arquitetura de controle proposta neste trabalho. Foi determinada como posição final a posição de X_g = 0.5 e Y_g = 0.5. Na Fig. 9, é possível perceber que o robô realiza o percurso com precisão e armazena os dados do caminho percorrido construindo também um mapa 2D do ambiente com as informações derivadas do sensor kinect.

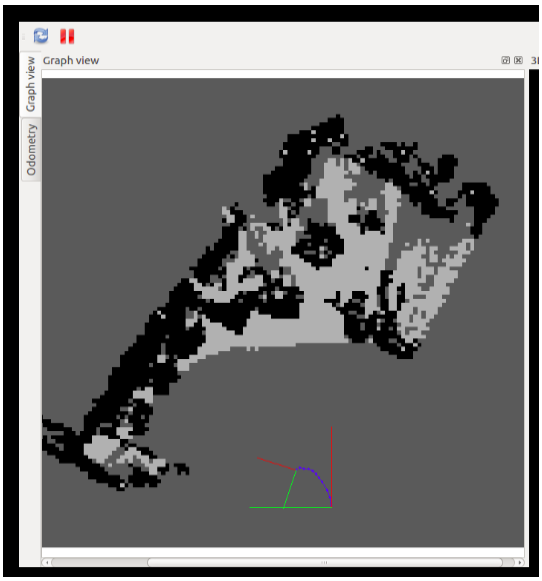

Fig. 9. Imagem RTAB-Map com mapa 2D do ambiente e visão da trajetória do robô

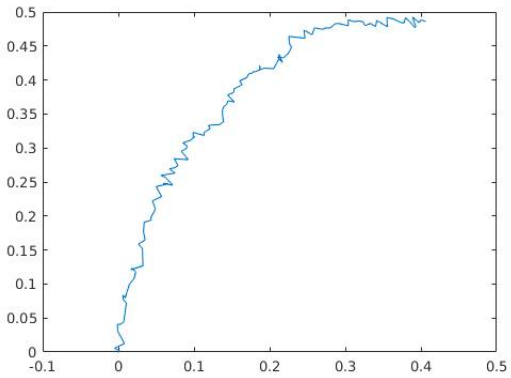

Fig. 10. Gráfico da trajetória do robô ponto a ponto construído no MATLAB

Com os dados do controlador podemos observar no gráfico gerado pelo MATLAB o percurso que o robô percorreu desde seu ponto inicial até sua posição final nas coordenadas $X_g = 0.5$ e $Y_g = 0.5$.

6 Conclusões

Este do trabalho apresentou uma arquitetura de controle baseada em um sistema de localização de um robô móvel a partir das informações da odometria visual geradas pelo sensor Kinect, utilizando o ROS e a libraria RTAB. Para isso, foi aplicado um controlador baseado em um sistema Fuzzy, que é o encarregado de levar o robô de uma

posição inicial a uma posição final. A opção pela lógica fuzzy é das mais convenientes, uma vez que direciona o foco para a resolução do problema, facilitando e otimizando a implementação do controlador, pois dispensa a necessidade de custosas modelagens matemáticas. Experimentos para validar a arquitetura de controle foram apresentados, validando a proposta. Futuramente, espera-se implementar um sistema com a funcionalidade de fazer com que o robô também possa desviar de obstáculos dentro do ambiente experimental.

Referências

1. Barrera, A.: Mobile Robots Navigation. [S.l.]: In-Teh (2010).
2. Blakouti, E., Amor, B. N.: Jallouti Autonomous wheelchair navigation with real time obstacle detection using 3D sensor. M AUTOMATIKA 57, 3, 761–773 (2016).
3. Benavidez, P. Mo Jamshidi, Ph.D.: Lutcher Brown Endowed Chair Mobile Robot Navigation and Target Tracking System International Conference on System of Systems Engineering. Albuquerque, New Mexico, USA - June 27-30 (2011).
4. Silva, R. M., Garcia, T. R., Cuadros, M. A. S. L., Gamarra. D. T. F.: Controle da Trajetória de um Robô Móvel com Tração Diferencial Utilizando Processamento Digital de Imagens e Lógica Fuzzy. Santa Maria, RS (2017).
5. Li, I-H., Wang, W-Y. Chien, Y-H.: A Kinect-sensor-based Tracked Robot for Exploring and Climbing Stairs. International Journal of Advanced Robotic Systems, 11:80 (2014)
6. Pegas, G. L., Inoue, R. S.: Rastreamento Visual para Robôs usando Microsoft Kinect. 2014, 94f, Trabalho de Graduação (Graduação Engenharia Elétrica) - Universidade Federal de São Carlos,SP (2014).
7. Peñafiel A. S. D, Pereira S. A. Guilherme.: Desenvolvimento de um Sistema de Navegação em Ambientes Internos para um Robô Pessoal. 2014, 87f, Dissertação (Dissertação de Mestrado Engenharia Elétrica), Universidade Federal de Minas Gerais Laboratório de Sistemas de Computação e Robótica, MG (2014).
8. ROS.org, http://wiki.ros.org/, acesso em 05/01/2018.
9. Motta, D.R.V.; Salarolli, P. F.; Almeida, G.M.; Gamarra, D.F.T.; Cuadros, M.A.D.L. Comparative Trajectory Controllers: Fuzzy, Fixed Gains and Backstepping for Robots with Differential Traction Using Image Processing, Simposium Brasileiro de Automação Inteligente (SBAI), Porto Alegre, 2017.
10. Sivic, Josef (April 2009). "Efficient visual search of videos cast as text retrieval" (PDF). *ieee transactions on pattern analysis and machine intelligence, vol. 31, no. 4*. ieee. pp. 591–605.
11. Tango Project, https://developers.google.com/tango/, acesso em 05/01/2018.