PLATAFORMA ROBÓTICA MÓVEL PARA ESTUDOS DE LOCALIZAÇÃO, MAPEAMENTO, NAVEGAÇÃO E PERCEPÇÃO 3D BASEADOS EM UM SENSOR RGB-D

ANA CLÁUDIA R. LEMOS, KENNYA R. MENDONÇA

Coordenação do Curso de Engenharia Elétrica, Instituto Federal de Goiás – Campus Jataí
Alameda Flamboyant, Q.35, Lt. 01, nº 130, Área Institucional, Residencial Flamboyant, Jataí, GO, BRASIL
E-mails: rl.anaclaudia@gmail.com, kennyaresende@gmail.com

Abstract— Following the thinking of innovation in mobile robotics aimed at the use of RGB-D cameras, this work presents the development of a mobile robotic platform with an embedded RGB-D sensor based on Microsoft Kinect for mobile robotics studies related the location, mapping and autonomous navigation using visual information, and the 3D representation of the environment for general purposes. Initiating such studies, this paper also presents a proposed solution to the SLAM problem using the RGB-D data provided by Kinect. This proposal was implemented in the ROS, using the RTAB-Map package. For the locomotion of the platform during the SLAM process, it was decided to use a gamepad, and the microcontroller board Arduino Mega 2560. The system is designed to map internal environments, such as offices, homes and some industries.

Keywords— Mobile robotic platform, RGB-D sensor, SLAM visual. RTAB-Map. ROS.

Resumo— Seguindo o pensamento de inovação em robótica móvel, o presente artigo apresenta o projeto e o desenvolvimento de uma plataforma robótica móvel com um sistema sensor RGB-D embarcado baseado no Microsoft Kinect, para estudos de robótica móvel relacionados a localização, mapeamento e navegação autônoma de robôs móveis utilizando informações visuais, e a representação 3D do ambiente de atuação do robô para propósitos gerais. Iniciando tais estudos, este artigo apresenta também uma proposta de solução do problema de SLAM, usando os dados RGB-D fornecidos pelo Kinect. Tal proposta foi implementada no ROS, utilizando o pacote RTAB-Map. Para a locomoção da plataforma durante o processo de SLAM, optou-se por teleoperação utilizando um *gamepad*, e a placa microcontrolada Arduino Mega 2560. O sistema foi projetado para mapear ambientes internos, como por exemplo escritórios, residências e algumas indústrias.

Palavras-chave— Plataforma robótica móvel, sensor RGB-D, SLAM visual, RTAB-Map, ROS.

1 Introdução

O desenvolvimento de robôs que possam operar autonomamente é um dos temas mais explorados nas pesquisas em robótica móvel. Em se tratando de robôs móveis autônomos a tarefa mais básica que estes devem realizar é a navegação, ou seja, o robô deve ser capaz de se locomover com segurança de um local para outro, desviando de possíveis obstáculos sem a necessidade de auxílio externo (Jácobo, 2001).

Vários métodos de localização dependem de um mapa prévio do ambiente para estimar sua posição e orientação de forma precisa. Em contrapartida, a construção do mapa do ambiente depende da localização do robô móvel (Romero *et al*., 2014). Assim, fica evidente que há uma interdependência entre localização e mapeamento, sendo essa interdependência o maior problema para a navegação autônoma de robôs em ambientes desconhecidos. Para tratar destes problemas de forma simultânea surgiram as técnicas de Localização e Mapeamento Simultâneos, do inglês, *Simultaneous Localization and Mapping* (SLAM), que permitem ao robô se localizar no ambiente e ao mesmo tempo construir um mapa deste ambiente.

Para a solução dos problemas de SLAM, as técnicas mais atuais têm como base a utilização de informações fornecidas por câmeras, surgindo o SLAM Visual, ou *Visual Simultaneous Localization and Mapping* (VSLAM) (Karlsson *et al*., 2005).

Dentre as câmeras utilizadas em robótica móvel, as que mais tem se destacado, principalmente para a solução do problema de VSLAM, são as de profundidade, também chamadas de câmeras RGB-D. Além das tarefas relacionadas a navegação, as câmeras também têm sido muito utilizadas na robótica móvel para se obter a percepção tridimensional (3D) do ambiente em tarefas de teleoperação (Azevedo, 2017), identificação e manipulação de objetos (Dantas, 2017) e identificação de pessoas (Sales, 2014).

O Microsoft Kinect, que foi desenvolvido pela empresa PrimeSense em colaboração com a Microsoft, foi um dos primeiros dispositivos de fácil acesso a fornecer imagens RGB-D, isto porque os sensores anteriores ao Kinect tinham um custo extremamente elevado, o que os tornavam inviáveis a projetos de baixo orçamento (Cruz *et al*., 2012; Viecili, 2014).

Seguindo o pensamento de inovação em robótica móvel voltado ao uso de câmeras RGB-D com a tecnologia desenvolvida pela PrimeSense e de compartilhamento desses conhecimentos, uma ferramenta que tem chamado a atenção é o *Robot Operating System* (ROS). O ROS é um *framework* de código aberto que veio auxiliar o desenvolvimento de aplicações para robôs, sendo difundido mundialmente tanto na comunidade acadêmica quanto na indústria. O ROS é citado em trabalhos de Hjelmare and Rangsjo (2012) e Sales (2014), por exemplo, como ferramenta utilizada para solucionar problemas da robótica móvel como VSLAM usando dados RGB-D.

Com a apresentação deste cenário, constata-se a grande relevância do aprimoramento de pesquisas em

robótica móvel voltadas para a utilização de informações visuais em tarefas como manipulação, teleoperação e navegação autônoma de robôs móveis.

Como base para o estudo e ensino da interação entre robôs e ambientes, muitas instituições de ensino fazem uso de simulações. As simulações são importantes ferramentas para se obter resultados preliminares e aproximar estudantes de conceitos da robótica. No entanto, para ser útil, um sistema robótico tem que ser testado em um robô real (Dorigo, 1996). Como plataformas robóticas móveis comerciais têm, em geral, um custo elevado, decorre então, a importância de se implementar uma plataforma robótica móvel que permita a realização de tais estudos, aproximando pesquisadores e estudantes de tais conceitos.

Sendo assim, este trabalho tem como objetivo o desenvolvimento de uma plataforma robótica móvel com um sistema sensor RGB-D embarcado baseado no Microsoft Kinect, e a utilização desta plataforma na solução do problema de VSLAM usando dados RGB-D por meio do ROS.

O desenvolvimento desta plataforma tem o intuito de aproximar os estudantes de conceitos relacionados a localização, mapeamento e navegação autônoma de robôs móveis utilizando informações visuais, e a representação 3D do ambiente de atuação do robô usando dados fornecidos por sensores RGB-D, visando uma melhor aprendizagem e uma interação entre teoria e prática.

2 Localização e Mapeamento Simultâneos baseado em Câmeras

Efetivamente é muito difícil solucionar os problemas de localização e mapeamento de maneira independente, isto porque, para que um robô possa construir um mapa do ambiente ele deve saber de onde foi feita a observação que será acrescentada ao mapa, e para se estimar a localização precisa de um robô é necessário um mapa (Romero *et al*., 2014; Vilas Boas, 2011).

O processo de SLAM, onde o mapa é formado por um conjunto de marcos, consiste em cinco fases, conforme apresentado na Figura 1: extração de pontos de referência ou características, associação de dados, estimação do estado, atualização do estado, e atualização dos pontos de referência (Gaspar, 2017; Santana, 2011).

Os pontos de referências são características facilmente encontradas no ambiente de operação do robô e correspondem às informações do ambiente que o robô usa para se localizar. Bons pontos de referência além de serem facilmente identificáveis, podem ser facilmente reobservados; ou seja, podem ser detectados de posições diferentes; abundantes, para que o robô não fique sem referência e se perca no ambiente e estacionários, permitindo que o robô possa mapeá-lo correntemente e consequentemente se localizar em relação a estes pontos. Após escolha dos pontos de referência, deve-se extrair estes de maneira segura, sendo que esse processo depende do tipo de ponto de referência bem como dos sensores utilizados (lasers, sonares, visuais) (Gaspar, 2017; Santana, 2011).

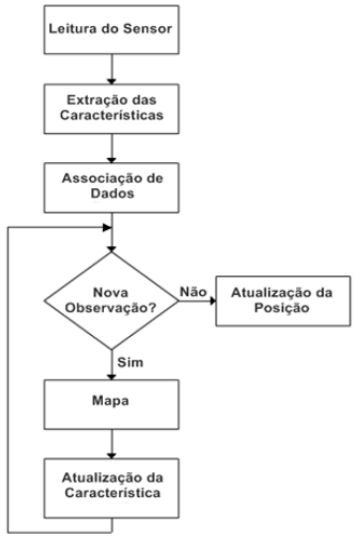


Figura 1. Etapas do processo SLAM. Adaptado de (Gaspar, 2017).

No SLAM, sempre que uma nova observação é feita, tenta-se associá-la a um marcador (ponto de referência) já existente, em um processo chamado associação de dados. Se um ponto de referência da nova observação corresponder a um marcador já existente, este é usado para refinar a pose do robô, tornando-o a localização do robô mais precisa; se corresponder a um novo marco, este é adicionado ao mapa; e caso seja um "falso" marcador este é descartado (Santana, 2011; Sumei PI *et al.*, 2014 apud Gaspar, 2017).

As etapas de estimação do estado, atualização do estado e atualização dos pontos de referência dependem do método aplicado, mas em geral são feitas utilizando estimadores estatísticos (Santana, 2011).

2.1 SLAM Visual

VSLAM consiste na abordagem de SLAM utilizando sensores visuais, tendo como objetivo estimar a trajetória da câmera, e, portanto, do robô, enquanto constrói uma representação do ambiente. Assim, nas técnicas de VSLAM, tem-se como pontos de referências características visuais, onde estas características são extraídas por processamento de imagem (Gaspar, 2017).

No VSLAM a complexidade e dimensão do ambiente são fatores importantes. Quanto maior e mais complexo o ambiente, maior a probabilidade de ocorrer problemas relacionados ao movimento do robô. Fatores como a dimensão do mapa (2D ou 3D), poder de processamento, quantidade de memória computacional disponível, também devem ser levados em consideração. Um dos maiores desafios do VSLAM é a associação de dados (*matching*). Pode ocorrer de um objeto ser visualizado mais de uma vez, em diferentes posições e este não ser reconhecido, ou um objeto

pode ser associado a outro indevidamente. Outro problema está relacionado à dinamicidade do ambiente (Souza, 2012; Thrun, 2002).

Outro desafio do VSLAM são as incertezas inerentes aos movimentos da câmera/robô. Uma vez que os robôs se movimentam em superfícies não uniformes pode ocorrer de a câmera capturar imagens embassadas ou com baixa textura, e como consequência obter a localização errada do robô e um mapa inconsistente (Davison, 2003).

Atualmente muitos algoritmos de SLAM e VSLAM são encontrados na literatura. Em sua pesquisa, Torres (2016), faz uma análise de alguns dos principais algoritmos de SLAM e VSLAM. Segundo Torres (2016), em se tratando de VSLAM usando dados RGB-D, um algoritmo que tem apresentado resultados satisfatórios é o *Real-Time Appearance-Based Mapping* (RTAB-Map), que é baseado em três trabalhos de Labbé and Michaud (2011, 2013, 2014). O RTAB-Map surgiu com o objetivo de fornecer uma ferramenta para mapeamento em tempo real de ambientes de qualquer dimensão. Para isso o RTAB-Map utiliza um sistema de gerenciamento de memória com componentes inspirados em trabalhos de psicologia (Labbé and Michaud, 2011, 2013, 2014).

Por suas características e por poder ser integrado ao ROS, o RTAM-Map foi o algoritmo escolhido neste trabalho para a solução do problema de VSLAM usando dos dados RGB-D fornecidos pelo Kinect.

2.2 Câmeras RGB-D: Kinect

Uma câmera RGB-D tem a capacidade de obter imagens RGB em conjunto com dados de profundidade da cena (D) (Dantas, 2017). Para ser capaz de fornecer imagens RGB-D, o Kinect possui duas câmeras CMOS, uma infravermelho (IR) e uma que fornece imagens em cores RGB; além de um emissor laser IR, onde os dados de profundidade são fornecidos pelo sensor IR (emissor e câmera) (Oliveira Junior, 2015). Na Figura 2, é mostrado o Kinect sem o seu envoltório, o que permite visualizar a câmera RGB, a câmera IR e o emissor IR.

Figura 2. Sensor RGB-D Microsoft Kinect. Adaptado de (Open Source Robotics Foudation, 2012).

Para obter a profundidade da cena o Kinect relaciona o emissor de IR e a câmera de IR, por estereoscopia por luz estruturada. O emissor de IR projeta um único feixe que é dividido por uma grade para criar o padrão de luz estruturada. Esse padrão é então capturado pela câmera IR, e a correspondência estéreo é então obtida por triangulação entre o padrão de luz estruturada capturado pela câmera IR, e uma imagem de referência armazenada no Kinect (Khoshelham and Elbernik, 2012).

2.3 Robot Operating System

Ao longo dos anos, muitos *frameworks* foram criados para facilitar a prototipagem de softwares para robôs. Neste contexto, em meados dos anos 2000, o ROS surgiu da necessidade da universidade de *Standford* em ter uma ferramenta de suporte no desenvolvimento dos seus projetos de robótica (Open Source Robotics Foundation, 2018).

Por ter foco no desenvolvimento de códigos específicos de hardware e algoritmos em bibliotecas independentes; por estimular a formação de um ecossistema de colaboração e compartilhamento de código; por ser gratuito, de código aberto, e ser distribuído com licença de software permissiva que permite utilizá-lo em projetos comerciais, o ROS passou a ser adotado por inúmeros laboratórios de pesquisa, o que resultou em inúmeros produtos comerciais com *drives* e ferramentas dentro dos padrões ROS (Araújo, 2017; Klaser, 2014; Sales, 2014).

O ROS possui três níveis de conceitos, *Filesystem* (sistema de arquivos), *Computational Graph* (gráfico computacional) e *Community* (comunidade). O nível *Filesystem* diz respeito a como o ROS é organizado, em termos de disco. Neste nível o destaque é para os *packages* (pacotes), que é a estrutura principal de organização do ROS. O nível *Computational Graph* diz respeito a estrutura de comunicação em ROS. No nível *Community* encontram-se recursos ROS que permitem que comunidades troquem software e conhecimento (Araujo, 2017; Open Source Robotics Foundation, 2014).

3 Descrição da Plataforma Robótica Móvel

O sistema desenvolvido pode ser dividido em dois módulos: o embarcado na plataforma robótica móvel, e o exterior à plataforma. A Figura 3 apresenta o diagrama geral do sistema.

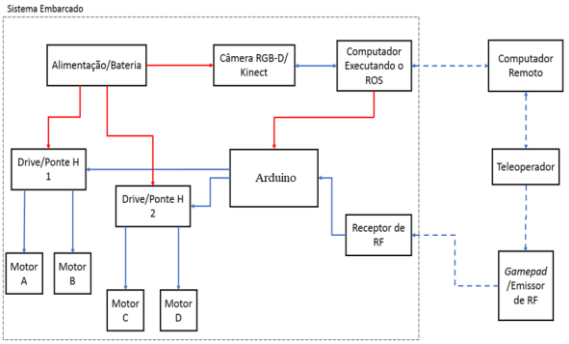


Figura 3. Diagrama geral do sistema desenvolvido.

A partir do computador remoto, que funciona como um espelho do computador embarcado, os pacotes e ferramentas ROS instalados no sistema embarcado necessários ao VSLAM são executados, e o teleope-

rador tem acesso em tempo real à câmera RGB do Kinect e ao processo de VSLAM. Os comandos de locomoção são recebidos pelo Arduino por meio do módulo receptor de RF, processados e analisados, e dependendo das instruções, nos quatro motores do sistema de locomoção, serão operadas funções apropriadas.

Ao finalizar o processo de VSLAM, os dados gerados são automaticamente salvos, com o nome "rtabmap.db", no diretório do usuário, especificadamente no diretório "ros". É importante renomear o arquivo, pois em caso de um novo mapeamento, este será salvo no lugar do anterior. Os dados gerados no processo de VSLAM, podem ser visualizados pela ferramenta de análise de *database* do RTAB-Map, o *RTABMap database Viewer*. Esta ferramenta permite gerar mapas, em conjunto com os dados de odometria, que se referem as poses da plataforma durante o processo de VSLAM, dados estatísticos, bem como fazer a otimização dos dados.

Os mapas gerados, juntamente com os dados de odometria e estatísticos podem ser usados futuramente em processos de navegação autônoma.

3.1 Desenvolvimento do Hardware

Tendo em vista que robôs com deslocamento por rodas são mais simples do que outros meios, como pernas e esteiras, e que o robô foi idealizado para ambientes relativamente planos, optou-se por desenvolver uma plataforma robótica móvel com locomoção por rodas, facilitando assim sua construção e controlabilidade.

Levado em consideração a dimensão mínima do *chassi*, definida a partir do tamanho do maior sistema computacional que poderia ser embarcado na plataforma (um notebook), foi escolhido um sistema com quatro rodas. Levando em consideração também a simplificação do sistema de tração e a direção, foi escolhido o sistema diferencial com as quatro rodas tracionadas.

O projeto do sistema eletroeletrônico engloba:

- dispositivos computacionais: que neste projeto são placa microcontrolada Arduino, responsável pelo controle dos motores do sistema de locomoção, um computador embarcado na plataforma, responsável pelo processo de VSLAM, e um computador remoto, que funciona como um espelho do computador embarcado, permitindo controle e monitoramento do processo de VSLAM de maneira remota;
- drive dos motores, que é responsável pelo interfaceamento entre o Arduino e os motores;
- sensores, que neste projeto são o Kinect, e os sensores de comunicação que fazem parte do módulo de teleoperação; e,
- baterias usadas na alimentação do sistema.

O sistema computacional não embarcado pode ser qualquer computador com os sistemas operacionais Windows, Linux ou Mac, assim não há especificações para este.

O computador escolhido para ser usado no processo de VSLAM é o Notebook Samsung modelo NP-RV415I. Este computador foi escolhido pela sua disponibilidade, não sendo necessário o investimento em um primeiro momento em outro sistema computacional.

Para controle da locomoção da plataforma durante os testes de VSLAM, optou-se por um sistema teleoperado. Neste controle é usado um *gamepad*, ou controlador de jogos, com conexão padrão *wireless*. A conexão *wireless* é feita por rádio frequência (RF), sendo composta por dois módulos, o módulo *gamepad* com emissor de RF, e o módulo receptor de RF.

Neste sistema, quando um operador apertar algum botão predefinido do *gamepad* um sinal é gerado e enviado ao módulo receptor, que está conectado ao Arduino. Os dados recebidos são então processados e analisados pela placa, e dependendo das instruções, nos quatro motores do sistema de locomoção, serão operadas funções apropriadas.

Para controle da locomoção por teleoperação foi escolhida a placa microcontrolada Arduino Mega 2560.

Os motores são do modelo AK555/11.1PF12R83CE da *Akiyama Motors*, sendo motores de corrente contínua equipados de uma caixa de redução.

Para transformar os sinais eletrônicos de comando, de baixa potência, para comando dos motores, de alta potência, foi escolhido o drive *Monster Motor Shield* da *SparkFun Electronics* que é baseado no circuito integrado VNH2SP30, e que possui ponte H dupla, e é capaz de controlar dois motores.

Para alimentação dos motores de corrente contínua e do Kinect, optou-se pela bateria do tipo Chumbo-Ácido selada, 12VDC/7A.h. O fator que levou a escolha desta foi a disponibilidade. De modo a oferecer facilidade de uso da plataforma robótica móvel foi feito um sistema de recarga desta bateria, que se resume a um simples circuito de recarga e a uma fonte de tensão contínua externa.

Em relação ao computador, este possui bateria própria (Lítio-Íon 11,1V – 48Wh – 44000mAh) sendo esta, portanto, utilizada para a alimentação deste. Já para o Arduino, escolheu-se o meio mais simples de fornecer a energia necessária para o seu funcionamento, a conexão USB, usada na transferência de dados da placa para o computador e/ou transferir o *sketch* do computador para a placa. Cada porta USB de um computador é capaz de suprir até 500mA de corrente, sendo suficiente para os *drives* conectados aos pinos da placa.

3.2 Desenvolvimento do Software

O software para controle da locomoção por teleoperação foi desenvolvido na ferramenta Arduino IDE. O código estruturado é em sua essência um *loop* infinito onde é realizada a leitura dos comandos recebidos do *gamepad*, e, com base nessas informações, são acionados os motores responsáveis pela locomoção da plataforma robótica móvel. Os coman-

dos de locomoção implementados para a plataforma robótica móvel e a localização dos botões referentes a cada comando são mostrados na Figura 4.

Comandos	*Botões do Gamepad*
Mover para frente	À cima
Mover para trás	À baixo
Girar para à direita	Direita
Girar para à esquerda	Esquerda
Parado	Nem um botão pressionado

Figura 4. Comandos implementados para a plataforma e a localização dos botões referentes a cada comando no *gamepad*.

No fluxograma da Figura 5 é apresentada a lógica de programação para a locomoção por teleoperação (função *void loop*). O "X" representa um dos possíveis comandos quando um botão de comando é apertado, como mostra a Figura 4.

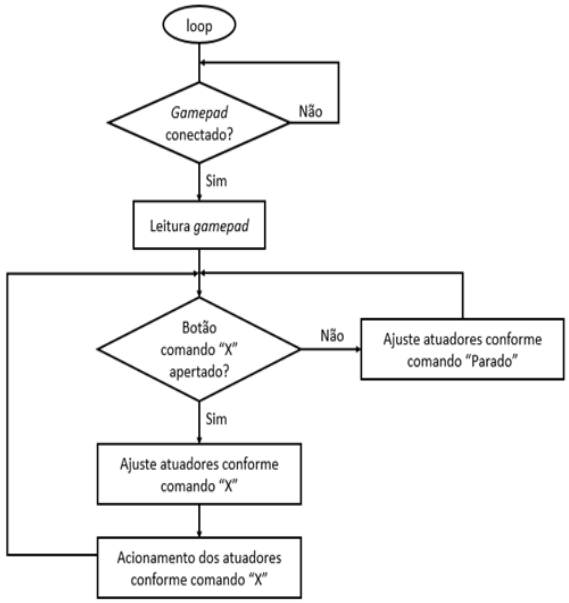


Figura 5. Fluxograma que representa o *loop* da lógica de controle do microcontrolador.

Para o desenvolvimento do sistema de VSLAM do robô escolheu-se o *framework ROS Kinetic Kame* (versão lançada em maio de 2016), sobre o sistema operacional *Linux Ubuntu 16.04*. Existem muitos métodos de VSLAM que podem ser executados por meio do ROS. Escolheu-se o RTAB-Map, porque este é capaz de prover SLAM Visual em tempo real, independentemente do tamanho do ambiente.

Dentre as diversas ferramentas e pacotes disponibilizados pelo ROS, os utilizados neste trabalho são: *Freenect* para ROS; RTAB-Map para ROS; ROS *Camera Calibration*; *Transform* (TF); *Robot Description Format* (URDF); e ROS *Visualization* (RViz).

Após a instalação do ROS e de pacotes adicionais necessários ao projeto, bem como a configuração do ambiente de trabalho *Catkin* ("*Catkin Workspace*"), criou-se o pacote referente a plataforma robótica móvel desenvolvida, onde se definiu o modelo URDF do robô e os arquivos (*.launch*) de configuração dos parâmetros e execução dos *nodes* do modelo do robô e do pacote RTAB-Map para ROS (rtabmap_ros).

A Figura 6 ilustra a configuração do RTAB-Map usada neste trabalho de acordo com plataforma robótica móvel desenvolvida (em termos de sensores usados no processo de VSLAM).

No caso da configuração mostrada na Figura 6, os *nodes* rtabmap e rgbd_odometry se inscrevem nos tópicos sensor_msgs/Image (rgb), sensor_msgs/CameraInfo e sensor_msgs/Image (*deph_registered*) para receber as imagens RGB, os metadados da câmera RGB e as imagens de profundidade publicadas pelo Kinect. O *node* rtabmap se inscreve também no tópico nav_msgs/Odometry publicado pelo *node* rgbd_odometry. Os clientes rviz e o rtabmapviz (ferramentas de visualização do ROS e do RTAB-Map, respectivamente), se inscrevem no tópico rtabmap/MapData para receber, o mapa 3D, em nuvem de pontos gerado no processo de VSLAM publicada pelo *node* rtabmap.

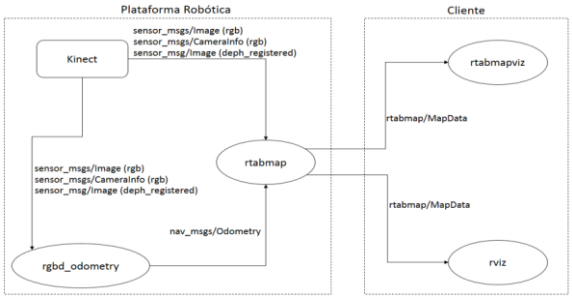


Figura 6. Configuração do RTAB-Map usada. Adaptado de (Open Source Robotics Foundation, 2016).

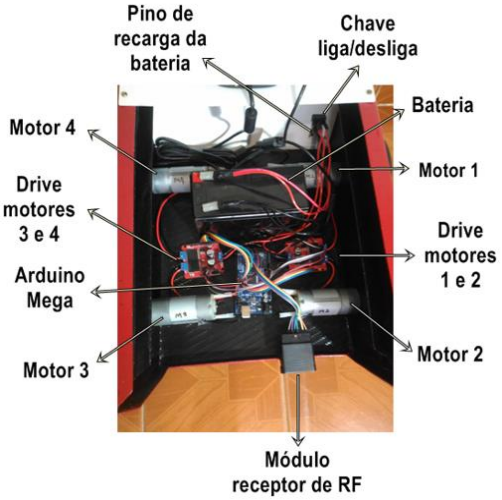
O primeiro passo na criação da descrição URDF da plataforma robótica móvel foi definir o "base_link" que é o *link* básico que todos os modelos devem ter, pois a partir desse *link* é que se define a relação de transformação TF (rotação e translação) entre todos os *links* de um robô. Nesse caso o próprio modelo do *chassi* foi escolhido como "base_link". No caso deste projeto, outro *link* necessário foi o "camera_link", que diz respeito ao Kinect, e é usado para integrar o modelo da plataforma e o processo de VSLAM. A partir do "camera_link" ao se executar o pacote *Freenect* (freenect_camera), tem-se a relação de transformação entre o "camera_link" e os *frames* obtidos pelas câmeras do Kinect (RGB e de profundidade), e consequentemente do restante da plataforma robótica móvel em relação a estes. Assim, durante o processo de VSLAM, quando o Kinect se move é realizada a estimação das poses do Kinect, e, portanto, da plataforma robótica.

Após a finalização da descrição do robô, criou-se um arquivo *.launch* do modelo do robô que tem como função iniciar os *nodes* que permitem a visualização do modelo 3D no RViz e de publicar os estados e relações de transformação da plataforma.

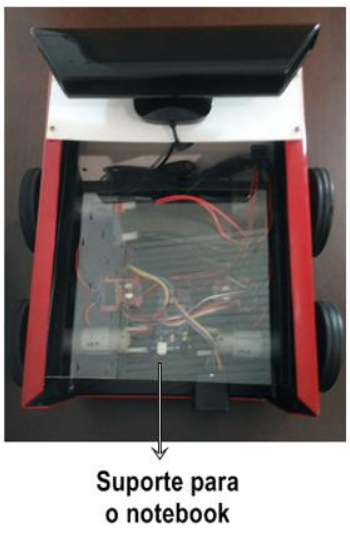
4 Resultados e Discussões

A Figura 7 apresenta o sistema embarcado que compõe a plataforma robótica móvel desenvolvida neste trabalho, conforme o diagrama apresentado na Figura 3. A placa microcontrolada Arduino Mega, os *drives* de controle dos motores, o módulo receptor de RF e a bateria foram acomodados na base do *chassi*. As ligações eletroeletrônicas foram feitas e o suporte para o notebook foi colocado.

A Figura 8 ilustra as vistas frontal e superior/lateral da plataforma. Os motores foram fixados na plataforma robótica móvel, e as rodas fixadas aos motores. O fechamento frontal foi feito com uma chapa de policarbonato, onde a chave liga/desliga do sistema (exceto Arduino e o computador do processo de VSLAM) e o pino de recarga da bateria foram fixados. O suporte para o Kinect foi feito de chapa de policarbonato e foi colocado na parte superior frontal da plataforma. O Kinect foi fixado ao suporte com fita dupla face de alta aderência.



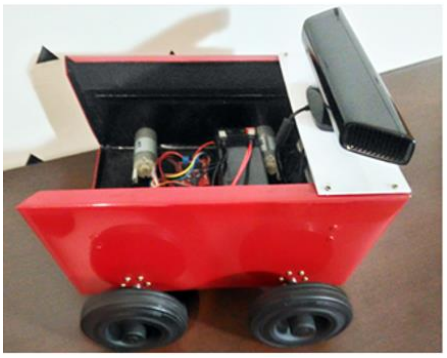
(a)



(b)

Figura 7. Resultado da disposição dos elementos e ligações da plataforma desenvolvida (a) e montagem do suporte para o notebook (b).

(a)



(b)

Figura 8. Resultado da fixação dos motores, rodas, chave liga liga/desliga, pino de recarga da bateria e Kinect. Vista frontal (a) e vista lateral (b).

No processo de VSLAM, uma solução ótima permite computar as poses da plataforma robótica móvel por meio da câmera Kinect e produzir os mapas 3D do ambiente onde está inserida a plataforma. Para computar as poses da plataforma robótica foi necessário obter a relação de transformação entre o centro da plataforma robótica e o Kinect.

Para tornar o sistema mais atraente como ferramenta educacional, optou-se por obter a relação da transformação pela descrição URDF da plataforma robótica móvel, permitindo durante o processo de VSLAM a visualização em tempo real do modelo 3D da plataforma robótica a medida que a plataforma robótica se locomove pelo ambiente, e o mapa é adicionado ao RViz. A Figura 9 apresenta o resultado da descrição URDF da plataforma robótica móvel desenvolvida, visualizado no RViz.

Já a Figura 10 apresenta um *print* do processo de VSLAM, onde foi utilizado o RViz como ferramenta de visualização, permitindo acompanhar o modelo 3D da plataforma se locomovendo no ambiente de visualização a medida que a plataforma robótica se locomovia pelo ambiente real e com o mapa sendo adicionado no ambiente de visualização.

É importante destacar que a visualização do modelo 3D da plataforma só é possível no RViz. Porém, para se obter a transformação do Kinect em relação as outras partes da plataforma robótica móvel, o arquivo *.launch* da descrição URDF da plataforma deve ser

iniciado mesmo que se tenha preferência pela visualização no RTAB-Map. Se o usuário preferir sempre visualizar o processo de VSLAM no RTAB-Map, pode-se excluir do arquivo *.launch* da descrição URDF da plataforma robótica, a linha que define a visualização pelo RViz.

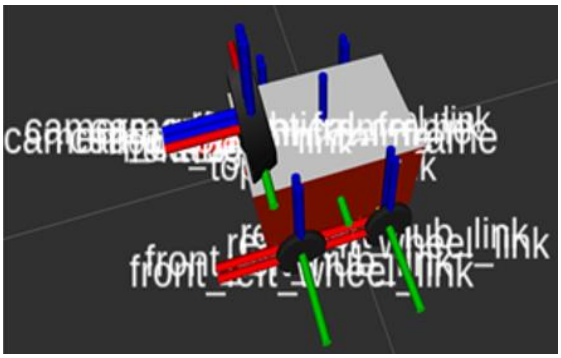


Figura 9. Visualização no RViz: modelo 3D da plataforma.

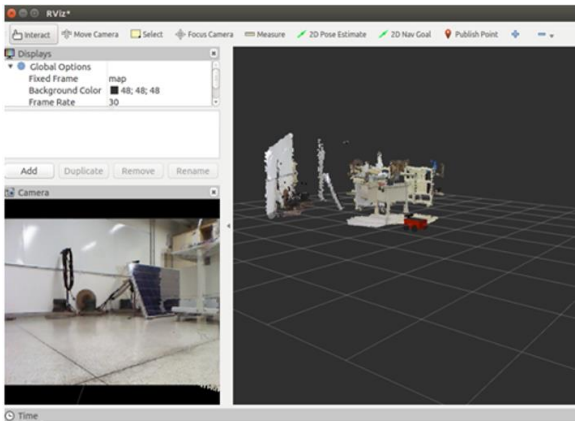


Figura 10. Visualização no RViz: execução do VSLAM.

4.1 Experimentos de VSLAM

Para a análise do processo de VSLAM foram realizados dois experimentos, cujo cenário é mostrado na Figura 11, e corresponde a um laboratório de estudos.

Figura 11. Cenário dos experimentos de VSLAM.

O primeiro experimento, teve como objetivo acompanhar o mapeamento, e a localização da plataforma robótica móvel no mapa, pela ferramenta RViz. Neste experimento a plataforma robótica foi posicionada no piso próximo ao centro do cenário, fazendo uma varredura de forma circular. O resultado do experimento é mostrado na Figura 12.

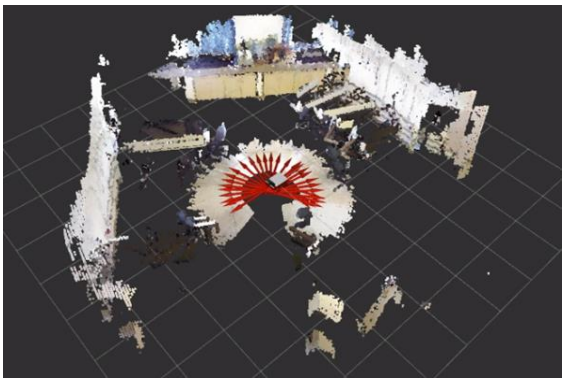


Figura 12. Experimento 1: uso do RViz para vizualização do processo de VSLAM.

Apesar de a plataforma robótica ter feito a varredura por completo, o mapa gerado apresentou muitas zonas sem a obtenção de dados, além disso ocorreu a perda de informações da odométricas (sequência das poses da plataforma representadas pelas setas).

Uma solução para a perda da odometria visual, seria a fusão sensorial com *encoders*, obtendo assim duas fontes de dados que permitem determinar a pose da plataforma robótica, onde em caso de perda de uma fonte, a localização é garantida pela outra. Com relação as falhas no mapeamento, o segundo experimento realizado teve como objetivo principal averiguar uma hipótese sobre o porquê isto ocorreu.

No primeiro experimento identificou-se também, que em alguns momentos, durante a locomoção da plataforma, o Kinect se movimentava, o que prejudica o mapeamento. Assim, para os outros experimentos, utilizou-se fita adesiva para fixar melhor o Kinect.

O segundo experimento teve como objetivo, analisar a influência da localização do Kinect em relação à altura do pé direito do ambiente no processo de VSLAM. Para isto foram realizados dois testes:

- Teste 1: a plataforma robótica foi posicionada no meio de uma mesa, como mostra a Figura 13 (a), onde Kinect ficou a uma altura aproximada de 104cm em relação ao piso, e foi feita uma varredura de forma circular, com a plataforma robótica locomovendo-se em torno do seu eixo.
- Teste 2: a plataforma robótica foi posicionada no piso, como mostra a Figura 13 (b), onde o Kinect ficou a uma altura aproximada de 26cm em relação ao piso, e foi feita uma varredura de forma circular, com a plataforma robótica locomovendo-se em torno do seu eixo.

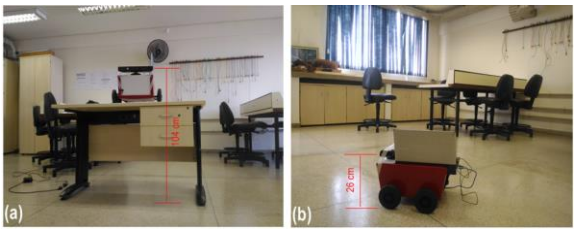


Figura 13. Posições da plataforma para segundo experimento: (a) Teste 1 e (b) Teste 2.

No Teste 1 obteve-se o mapa 3D colorido, em perspectiva, e perfil superior, mostrado na Figura 14 (a) e (b), e duas vistas do interior do mapa, apresentadas na Figura 15 (a) e (b).

No mapa 3D resultante do Teste 1 observa-se a visão quase que completa do ambiente capturado pelo sensor Kinect, fornecendo um mapa compreensível, porém com erros de correspondências de características, e como consequência erros na localização da plataforma.

No Teste 2 obteve-se o mapa 3D colorido, em perspectiva, e perfil superior, mostrado na Figura 16 (a) e (b), e duas vistas do interior do mapa, apresentadas na Figura 17 (a) e (b).

Ao se comparar o mapa gerado no Teste 1 com o mapa gerado no Teste 2, constata-se que no Teste 2 a localização do Kinect, em relação à altura do pé direito do ambiente, influenciou muito a qualidade do mapa gerado. No teste 2, o campo de visão da plataforma robótica ficou limitado, formando muitas zonas cegas, e como resultado um mapa com muitas regiões sem obtenção de dados.

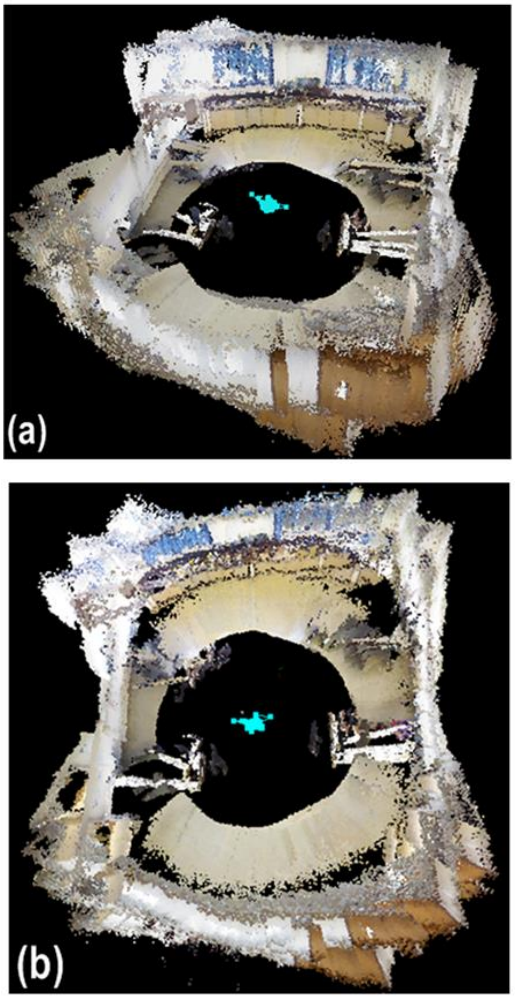


Figura 14. Mapa obtido do teste 1: (a) em perspectiva e (b) perfil superior.

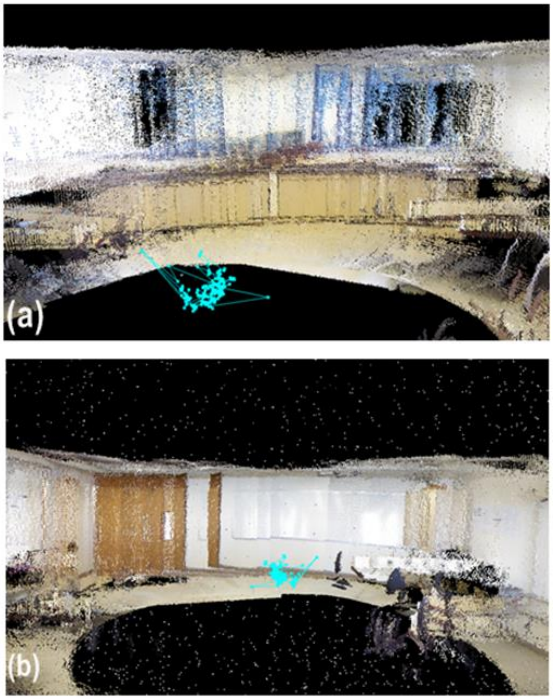


Figura 15. Mapa obtido do teste 1: (a) e (b) vistas do interior do mapa.

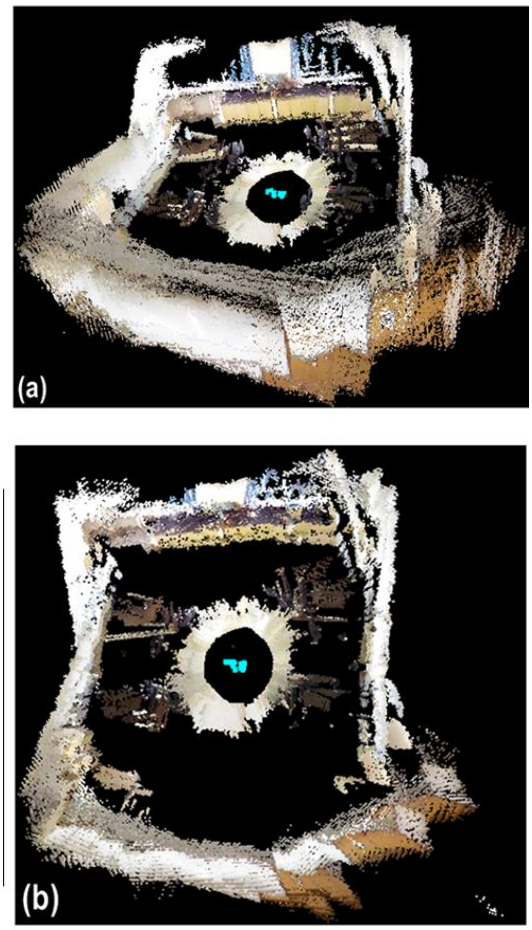


Figura 16. Mapa obtido do teste 2: (a) em perspectiva e (b) perfil superior.

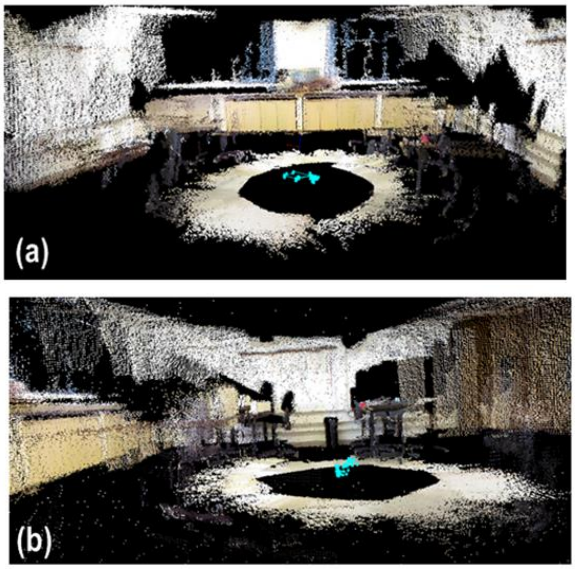


Figura 17. Mapa obtido do teste 2: (a) e (b) vistas do interior do mapa.

Para os experimentos de VSLAM o processamento foi realizado no computador apresentado na subseção 3.1, que possui processador dual-core ADM E-300, CPU de 1,3 Ghz e RAM de 2 GB. Devido ao poder computacional limitado, o processo de VSLAM com o computador escolhido não foi o ideal.

5 Conclusão

O desenvolvimento da plataforma robótica móvel foi um grande desafio, porém, dentro das limitações de projeto do hardware, apresentou bons resultados. Com a metodologia de projeto aplicada, foi possível construir uma plataforma robótica móvel, com requisitos mínimos para as propostas de utilização, de custo bastante reduzido em relação as plataformas robóticas disponíveis no mercado. Vale ressaltar, que esta plataforma não se compara às comercias, uma vez que para o desenvolvimento de um produto comercializável há muita tecnologia envolvida. Entretanto para a realização de estudos com orçamento reduzido e como ferramenta educacional, a plataforma desenvolvida é muito útil. Vale ressaltar também que as plataformas comercias apresentam no mínimo além de uma câmera RGB-D, sensores do tipo *encoders* para fusão odométrica. Todavia, a implementação futura deste tipo de sensor na plataforma robótica é completamente viável.

Por não ter sido possível a realização de experimentos detalhados do processo de VSLAM, não foi possível concluir, se a reconstrução 3D do ambiente e a localização da plataforma robótica no ambiente utilizando técnicas exclusivamente baseadas em visão computacional com os dados fornecidos pelo Kinect, utilizando o ROS e o RTAB-Map é viável.

Referências Bibliográficas

Araújo, M.A (2017). “Aplicação do ROS no Controle de Movimento de Robôs Equipados com Motores Dynamixel em Linux de Tempo Real”. UFU, Uberlândia, MG.

Azevedo, T.P (2017). “Locomoção de um robô móvel com esteiras em escadas”. UFRJ, Rio de Janeiro, RJ.

Cruz, L.; Lucio, D. and Velho, L (2012). “Kinect and RGB-D images: Challenges and applications”. SIBGRAPI Tutorials, p. 36-49.

Dantas, D.O (2017). “Implementação de um Sistema Autônomo de Construção de Estrutura usando Aprendizado por Reforço”. UFMA, São Luís.

Davison, A.J (2003). “Real-time simultaneous localisation and mapping with a single camera”. In: IEEE International Conference on Computer Vision (ICCV), v. 2, p. 1403-1410, Piscataway.

Dorigo, M (1996). “Introduction to the Special Issue on Learning Autonomous Robots”. IEEE Trans. On Systems, Man, and Cybernetics, v. 26, n. 3, p. 361-364.

Gaspar, A.R.S (2017). “Navegação e mapeamento subaquáticos simultâneos”. Universidade do Porto, Portugal.

Hjelmare, F. and Rangsjo, J (2012). “Simultaneous Localization and Mapping Using a Kinect™ In a Sparse Feature Indoor Environment”. Department of Electrical Engineering, Linkopings Universitet, Sweden.

Jácobo, J.E.A (2001). “Desenvolvimento de um robô Autônomo móvel versátil utilizando arquitetura subsumption”. Unicamp, Campinas, SP.

Karlsson, N.; Di Bernardo, E.; Ostrowsk, J.; Goncalves, L; Pirjanian, P. and Munich, M.E (2005). “The VSLAM algorithm for robust localization and mapping”. In: IEEE ICRA, p.24–29, Pasadena, California, USA.

Khoshelham, K. and Elbernik, S.O (2012). “Accuracy and resolution of Kinect depth data for indoor mapping applications”. Sensors, v. 12, n. 12, p. 1437-1454. doi:10.3390/s120201437.

Klaser, R.L (2014). “Navegação de veículos autônomos em ambientes externos não estruturados baseada em visão computacional”. USP, São Carlos, SP.

Labbé, M. and Michaud, F (2011). “Memory Management for Real-Time Appearance-Based Loop Closure Detection”. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco.

Labbé, M. and Michaud, F (2013). “Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation”. IEEE Transactions on Robotics, v.29, n.3, p. 734-745.

Labbé, M. and Michaud, F (2014). “Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM”. In: IEEE/RSJ

International Conference on Intelligent Robots and Systems, Chicago.

Oliveira Junior, E.M (2015). “Proposta de um método adaptativo para registro de dados RGB-D”. UFPR, Curitiba, PR.

Open Source Robotics Foudation (2012). “Technical description of Kinect calibration”. Disponível em:<http://wiki.ros.org/kinect_calibration/technical>. Acesso em: 13 out. 2017.

Open Source Robotics Foudation (2014). “ROS Concepts”. Disponível em: <http://wiki.ros.org/ROS/Concepts>. Acesso em: 13 out. 2017.

Open Source Robotics Foudation (2018). “History”. Disponível em: <http://www.ros.org/history/>. Acesso em: 06 mar. 2018.

Romero, R.A.F.; Silva Júnior E.P.; Osório. F.S. and Wolf. D.F. (Org.) (2014). “Robótica Móvel”. 1. Ed. LTC, Rio de Janeiro, RJ.

Santana, A.M (2011). “Localização e mapeamento simultâneos de ambientes planos usando visão monocular e representação híbrida do ambiente”. UFRN, Natal, RN.

Sales, F.F (2014). “SLAM and Localization of people with a mobile robot using a RGB-D sensor”. University of Coimbra, Coimbra.

Souza, A.A.S (2012). “Mapeamento robótico 2,5-D com representação em grade de ocupação-elevação”. UFRN, Natal, RN.

Thrun, S (2002). “Robotic Mapping: A Survey”. School of Computer Science Carnegie Mellon University, Pittsburgh.

Torres, P.L (2016). “Análisis de algoritmos para localización y mapeado simultáneo de objetos”. Dissertação (Mestrado), Ingeniería de Sistemas y Automática, Universidad de Sevilla, Sevilla.

Viecili, E.B (2014). “Exploração robótica ativa usando câmera de Profundidade”. UDESC, Joinville, SC.

Vilas Boas, E.R (2011). “Mapeamento e localização simultânea de ambientes dinâmicos aplicados na navegação de veículo autônomo inteligente”. UNIFEI, Itajubá, MG.