Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Ciência da Computação

CHRYSTIAN GUSTAVO MARTINS NASCIMENTO

MAPEAMENTO E LOCALIZAÇÃO PARA O KIT ROBÓTICO ROBODECK UTILIZANDO VISÃO COMPUTACIONAL

São Luís
2017

CHRYSTIAN GUSTAVO MARTINS NASCIMENTO

MAPEAMENTO E LOCALIZAÇÃO PARA O KIT ROBÓTICO ROBODECK UTILIZANDO VISÃO COMPUTACIONAL

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Profº Dr. Alexandre César Muniz de Oliveira

São Luís

2017

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Nascimento, Chrystian Gustavo Martins.
Mapeamento e Localização para o Kit Robotico RoboDeck Utilizando Visão Computacional / Chrystian Gustavo Martins Nascimento. – 2017.
43 p.

Orientador(a): Alexandre César Muniz de Oliveira.
Monografia (Graduação) – Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2017.

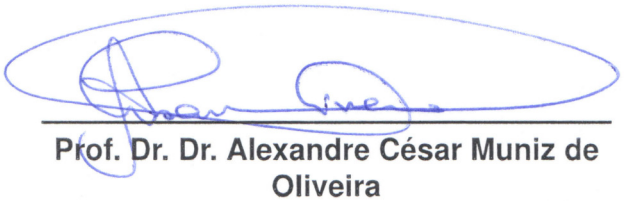
1. Localização. 2. Mapeamento. 3. RoboDeck. 4. Visão Computacional. I. de Oliveira, Alexandre César Muniz. II. Título.

Chrystian Gustavo Martins Nascimento

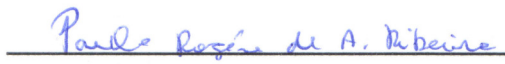
Mapeamento e Localização para o Kit Robótico RoboDeck Utilizando Visão Computacional

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

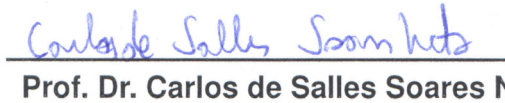
Trabalho aprovado. São Luís, 24 de janeiro de 2017:



Prof. Dr. Dr. Alexandre César Muniz de Oliveira
Orientador
Universidade Federal do Maranhão



Prof. Dr. Paulo Rogerio de Almeida Ribeiro
Universidade Federal do Maranhão



Prof. Dr. Carlos de Salles Soares Neto
Universidade Federal do Maranhão

São Luís
2017

Aos meus pais, José e Maria, que sempre me apoiaram.

Agradecimentos

Agradeço primeiramente à Deus por todas as suas bênçãos.

Aos meus pais, José e Maria, por todo apoio e dedicação. São meus maiores exemplos de amor, dedicação, e honestidade.

Aos meus irmãos, Guilherme e Gabriel, por toda ajuda e suporte.

À todos os amigos e colegas de curso que se fizeram importantes, especialmente a Thacyla e Matheus pelo auxilio prestado na realização deste trabalho. Aos meus companheiros e amigos do LACMOR.

Ao meu orientador, Alexandre, por todas as oportunidades e apoio dado.

Aos demais professores do curso, sempre dispostos a ensinar e proporcionar uma educação de qualidade.

À todos que contribuíram direta ou indiretamente na minha formação acadêmica.

“Toda verdade é fácil de ser compreendida depois de serem descobertas. O problema é descobrí-las!” - Galileu Galilei

RESUMO

Em robótica móvel, mapeamento e localização são tarefas essenciais, permitem mover-se a partir de uma posição desconhecida em um ambiente desconhecido de forma segura, usando sensores para obter informações sobre o ambiente a fim de detectar e evitar obstáculos.

Este trabalho descreve uma ferramenta para mapeamento e localização de robôs móveis que utiliza visão computacional com o objetivo de mapear as coordenadas do ambiente e marcações numeradas a fim ter referências no ambiente para se localizar. O mapeamento de coordenada é realizado transformando as coordenadas de pixel obtidas nas imagens em coordenadas de distância reais com a finalidade de construir o mapa do ambiente. O mapa é construído iterativamente, durante o processo as imagens e a leitura dos sensores de orientação são empregados na construção do mapa global. A localização é realizada utilizando o reconhecimento de marcações numeradas que funcionam como pontos de referência no mapa do ambiente. O reconhecimento das marcações é feito com a aplicação de uma Rede Neural Convolucional que tem capacidade de classificar números. Para validar a ferramente é usada a plataforma robótica RoboDeck. Os testes de validação são feitos em ambiente estruturado, com paredes e terreno plano, e os resultados experimentais demonstram que a proposta é adequada para mapeamento e localização de robôs móveis.

Palavras-chaves: Mapeamento. Localização. Robodeck. Visão Computacional.

ABSTRACT

In mobile robotics, mapping and localization are essential tasks, they allow to move from an unknown location in an unknown environment safely, using sensors to obtain information about the environment in order to detect and avoid obstacles.

This work describes a framework for mapping and localization of mobile robots. The approach uses computational vision to map the coordinates of the environment and numbered markings to have references in the environment to be located. The coordinate mapping is performed by transforming the pixel coordinates obtained in the images into real distance coordinates to construct the map of the environment. The mapping is an iterative process, during this process the images and the readings of the orientation sensors are used to construct the global map. For the localization task is numbered markings are used as reference point for the robot. These numbered markings are recognized qith a Convolutional Neural Network. The robotic plataform RoboDeck is used for tool validation. The approach was tested in a controlled environment, with walls and flat terrain, and the experimental results demonstrate that the proposal is suitable for mapping and localization mobile robots.

Keywords: Mapping. Localization. Robodeck. Computer Vision.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

EM	Expectation Maximization
CPU	Central Processing Unit
GPS	Global Positioning System
GPU	Graphics Processing Unit
MAP	Módulo de Alta Performance
MCC	Módulo de Controle de Comunicação
MCR	Módulo de Controle Robótico
MCS	Módulo de Controle de Sessão
MLP	Multi Layer Perceptron
RNC	Rede Neural Convolucional
RELU	Rectified Linear Unit
ROI	Region of Interest
ROS	Robot Operating System
SDK	Software Development kit

Sumário

1 Introdução

Segundo (WOLF et al., 2009) o estudo da robótica móvel é um tema bastante relevante e atual, tendo apresentado extraordinário desenvolvimento nas últimas duas décadas. Além disso, a aplicação prática em diversas atividades em nossa sociedade vem demonstrando o quão promissor é o futuro dessa área. Como exemplo tem seu uso em aplicações domésticas (e.g. aspiradores de pó e cortadores de grama robóticos), industriais (e.g. transporte automatizado e veículos de carga autônomos), urbanas (e.g. transporte público, cadeiras de rodas robotizadas) e de segurança e defesa civil e militar (e.g. controle e patrulhamento de ambientes, resgate e exploração em ambientes hostis). Por isso se faz necessário que o robô móvel tenha capacidade de explorar o ambiente.

Em Robótica Móvel, a exploração autônoma passa pela construção de um mapa progressivamente e auto localização nesse mapa, à medida que o robô movimenta-se pelo ambiente. O mapeamento e localização são tarefas essenciais, pois permitem mover-se a partir de uma posição desconhecida em um ambiente desconhecido de forma segura, usando seus sensores para obter informações sobre o ambiente a fim de detectar e evitar obstáculos.

O problema do mapeamento se relaciona com a capacidade de extrair dados do ambiente e colocar no sistema de coordenadas(GUTMANN; KONOLIGE, 1999). No âmbito da robótica móvel, a utilização de mapas é necessária para que o robô consiga construir um mapa do ambiente que vai explorar e utiliza-lo para conseguir cumprir seu objetivo. A extração de informação do ambiente -marcações- pelo robô móvel é feita através de sensores que interagem com o espaço, como, por exemplo, laser, infravermelho, sonar e câmera.

Por sua vez, o problema da localização na navegação de robôs moveis se relaciona com a capacidade que um robô deve ter de conhecer sua posição e orientação a cada instante em um sistema global de coordenadas (FOX; BURGARD; THRUN, 1999). Durante a exploração o robô deve se movimentar a partir de um ponto inicial e chegar ao objetivo, dessa forma ele deve ser capaz de perceber seu descolamento. Para isso, ele tem que combinar os sensores iterativamente a fim de estimar sua localização global.

A exploração do ambiente do robô pode ser feita de varias maneiras. Isso depende de alguns fatores como o ambiente que o robô vai explorar, os sensores utilizados pelo robô móvel , se o ambiente é estático ou dinâmico entre outros.

Em (BURGARD et al., 1999) é apresentado um algoritmo utilizando *Expectation*

Maximization (EM) para resolver o problema de mapeamento e localização. EM é um método iterativo no espaço de verossimilhança, que alterna em dois passos, um *expectation step(E-step)* e um passo de maximização(M-step). Os mapas globais são compostos por pequenos mapas locais, explorando o fato que na odometria em curto prazo os erros são tipicamente pequenos e podem ser relaxados. Essa abordagem é adequada para construção de grandes mapas de ambiente internos usando sensores inexatos como sonares.

Em (SE; LOWE; LITTLE, 2001) é descrito um algoritmo de localização e mapeamento de robôs moveis baseado em visão que utiliza características de imagem invariantes em escala como pontos de referencia em ambientes dinâmicos não modificado. Essas marcações 3d são localizadas e a movimentação do robô é estimada por *matching*, levando em conta a características do ponto de vista. O sistema usa visão estéreo.

Neste trabalho, é apresentado um estudo visando o emprego da câmera da plataforma robótica RoboDeck para auxiliar na construção de mapas a partir de visão computacional em ambientes fechados (*indoor*). Com relação a técnicas de visão computacional, a ideia central é permitir que o robô detecte obstáculos, paredes e também reconheça as marcações do ambiente para que possa se localizar no mapa. Por isso, foram usadas redes neurais do tipo *deep learning* para reconhecimento de marcações numeradas, com suporte do Sistema Operacional Robótico (Robot Operationg System – ROS2).

1.1 Objetivos

O objetivo geral deste trabalho é apresentar um estudo de técnicas de visão computacional que possibilitem o mapeamento e localização em ambientes internos de um robô móvel equipado com câmera. Especificamente, pretende-se

- Adaptar a construção de mapas à câmera disponível no kit RoboDeck;
- Treinar uma rede neural do tipo *deep learning* para reconhecimento de marcações numeradas;
- Analisar o desempenho da proposta em ambiente real.

1.2 Organização do Trabalho

Os demais capítulos desse trabalho estão organizados da seguinte forma: O capítulo 2 apresenta a fundamentação teórica com a base teórica onde se aborda

sobre Visão Computacional com alguns conceitos de processamento de imagem como: Filtro, limiarização e detecção de bordas e Rede Neurais Convolucionais. No capítulo 3, encontra-se as ferramentas utilizados no trabalho proposto. No capítulo 4, a proposta de mapeamento e localização baseada em visão é apresentada. No capítulo 5, relatam-se os testes realizados e discute-se os resultados experimentais. Finalmente, no capítulo 6, tem-se a conclusão do trabalho.

2 Fundamentação Teórica

Este capítulo apresenta os fundamentos teóricos utilizados no desenvolvimento deste trabalho.

2.1 Visão Computacional

Utilizar imagens afim de obter informações do ambiente tem se tornado tarefa cada vez mais comum em robótica móvel, devido a fatores como baixo custo e facilidade de implementação (SUJAN; MEGGIOLARO; BELO, 2008). A maior vantagem da utilização de sensores visuais está na riqueza de informações que eles são capazes de fornecer. Como por exemplo: descobrir se a alguma placa na imagem, se uma parede está próxima, se tem algum objeto na imagem entre outros. Mas para isso é necessário fazer o processamento de imagem a fim de extrair a informação desejada.

A seguir são apresentados alguns dos métodos usados para processamento de imagem.

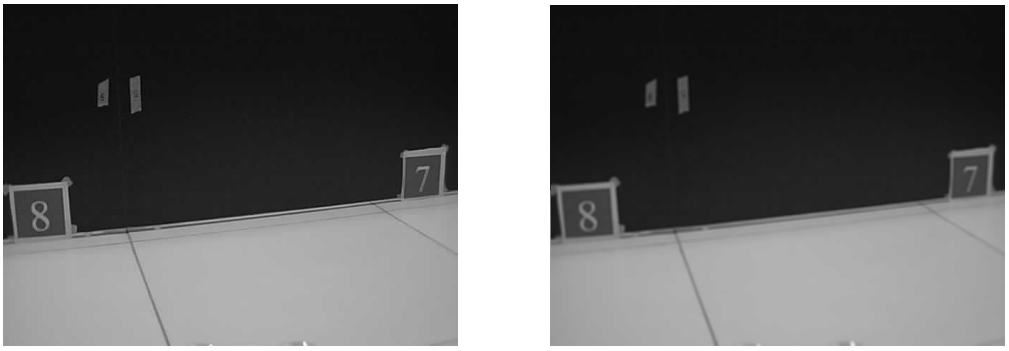
2.1.1 Filtragem

Filtragem consiste na aplicação de técnicas de transformação (operadores-máscaras) com o objetivo de corrigir, suavizar ou realçar determinadas características de uma imagem dentro de uma aplicação especifica. A filtragem é realizada pixel a pixel, e o valor do pixel atual depende do seu valor de nível de cinza original e do valor de nível de cinza de seus vizinhos.

A classificação da filtragem quanto ao domínio de frequência são dois tipos principais: filtragem passa baixa e filtragem passa alta. Os filtros são denominados passa-baixas quando atenuam ou eliminam as componentes de alta frequência no domínio das transformadas de *Fourier*. Como as componentes de alta frequência correspondem a regiões de bordas e/ou detalhes finos na imagem, o efeito da filtragem passa-baixas é a suavização da imagem, provocando um leve borramento na mesma. Já os filtros passa altas atenuam ou eliminam os componentes de baixa frequência e, em função disto, realçam as bordas e regiões de alto contraste da imagem (FILHO; NETO, 1999).

Um exemplo de filtro passa-baixa é o filtro da média. Na Figura 1 tem um exemplo de utilização do filtro da média para suavizar a imagem.

Figura 1 – Aplicação do filtro da média numa imagem em escala de cinza. A imagem original (esquerda) e a imagem filtrada(direita).



Fonte: Acervo do autor

2.1.2 Morfologia Binaria

A morfologia binária é uma ferramenta utilizada geralmente para extrair componentes de uma imagem que sejam úteis na representação e descrição de uma região, tais como contornos, esqueletos, etc. Além de ser úteis para a extração de características, permitem a eliminação do ruído produzido em todo processo de segmentação.As técnicas morfológicas também são de interesse para processos como filtragem morfológica, redução e recortado.

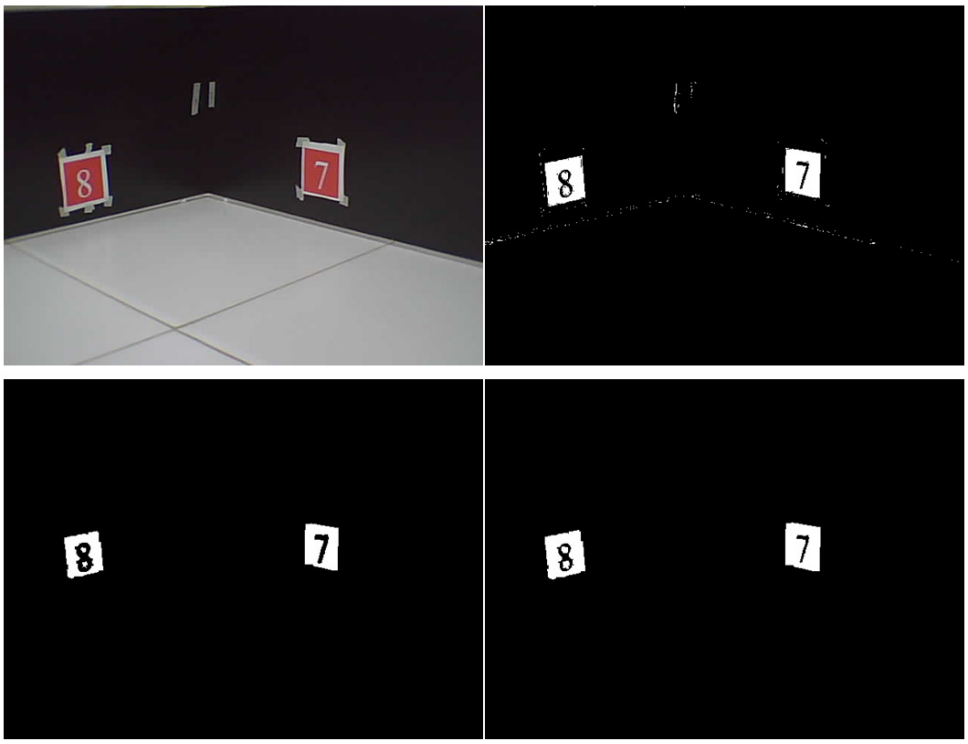
Na morfologia binária existem dois operadores básicos, chamados de erosão e dilatação. A erosão atua no sentido de remover pixeis das bordas externas dos objetos, enquanto que a dilatação acrescenta pixeis a essas bordas(FERREIRA, 2012). Na figura 2,a imagem superior esquerda é a original, a imagem superior direita é a imagem binarizada, a imagem inferior esquerda é a imagem após a erosão, utilizada para eliminar ruídos, a imagem inferior direita é o resultado da dilatação, usada para recuperar o tamanho original do objeto.

2.1.3 Limiarização (*Thresholding*)

Limiarização é o mais simples método de segmentação de imagem. O princípio da limiarização consiste em separar as regiões de uma imagem quando esta apresenta duas classes (o fundo e o objeto). Neste trabalho são utilizado dois tipos de limiarização: o método de *Otsu* e o método de limiarização multi banda.

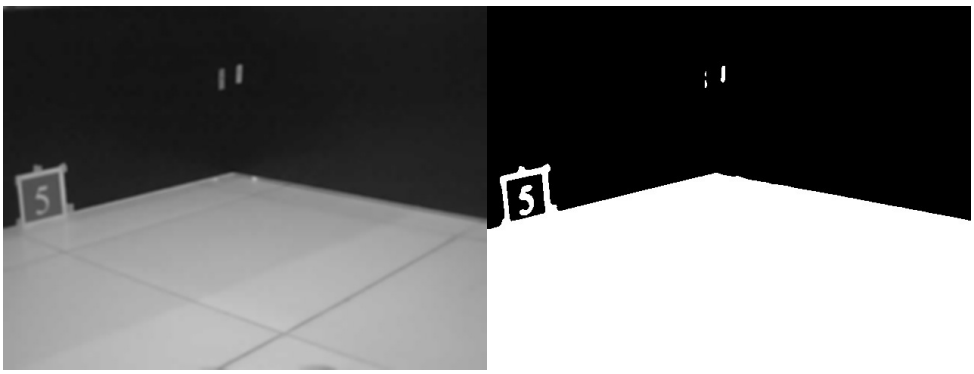
O método de *Otsu* é um algoritmo de limiarização, proposto por Nobuyuki *Otsu* . Seu objetivo é, a partir de uma imagem em escala de cinza, encontrar o melhor valor de *threshold* que separa frente de fundo em dois *clusters*, binarizando a imagem (atribuindo cor branca ou preta a cada um deles), como mostra a Figura 3.

Figura 2 – Aplicação de erosão e dilatação para retirar ruídos da imagem.



Fonte: Acervo do autor

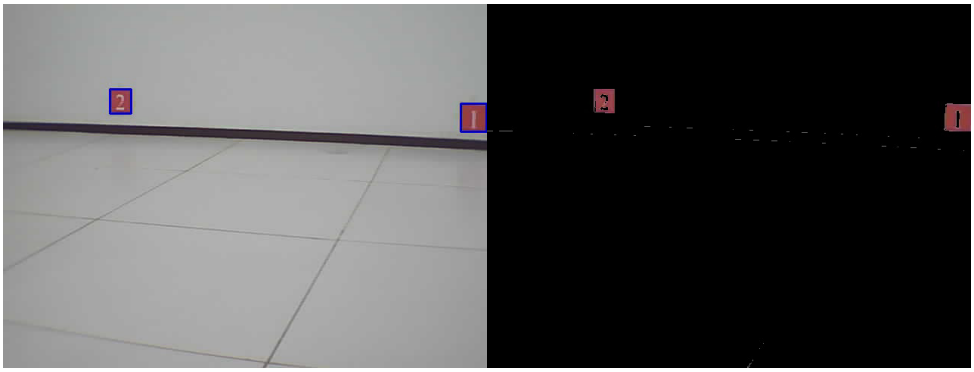
Figura 3 – Aplicação do método otsu. A imagem original em escala de cinza (esquerda) e a imagem binarizada pelo método Otsu (direita).



Fonte: Acervo do autor

O método de limiarização multi banda consiste em aplicar a limiarização em uma imagem colorida. Isso é feito aplicando um limiar para cada canal da imagem e depois combina-los com uma operação AND. Na Figura 4 é mostrado a limiarização para encontrar a marcação de cor vermelha na imagem.

Figura 4 – Limiarização multi banda. A imagem original (esquerda) e a imagem resultante da limiarização multi banda (direita).

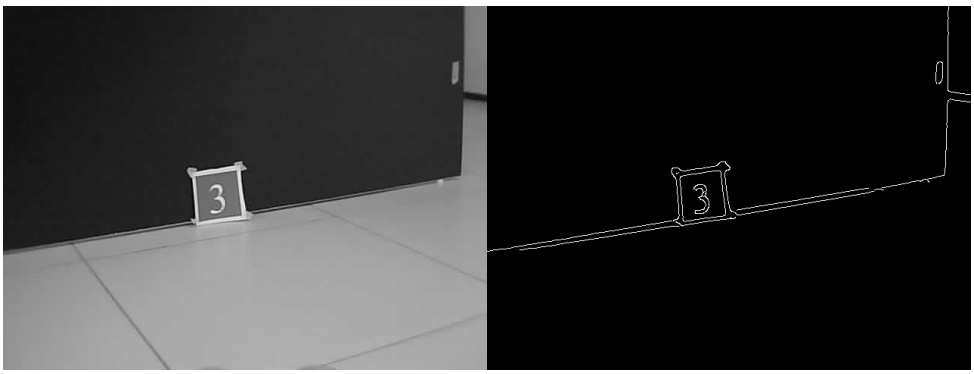


Fonte: Acervo do autor

2.1.4 Detector de Bordas *Canny*

Uma borda é a fronteira entre duas regiões da imagem com intensidades distintas de níveis de cinza. O operador *Canny* encontra bordas procurando por um máximo local do gradiente da imagem.O gradiente é calculado a partir da derivada de um filtro gaussiano. O método usa dois limiares para incluir as bordas fortes e fracas. Supondo que as bordas são linhas continuas, mesmo linhas de pouca intensidade são investigadas e podem ser inclusas como bordas se estiverem conectadas a bordas fortes. É o método menos sensível a ruído do que os demais e mais provável de detectar bordas fracas. Exemplo na Figura 5.

Figura 5 – Detecção de bordas. A imagem original em escala de cinza (esquerda) e a imagem resultante da detecção de bordas *Canny* (direita).



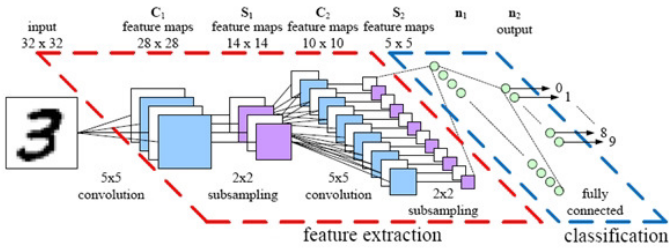
Fonte: Acervo do autor

2.2 Rede Neural Convolucional

Aprendizagem profunda é o termo usado para denotar o problema de treinar redes neurais artificiais que realizam o aprendizado de características de forma hierárquica, de tal forma que características nos níveis mais altos da hierarquia sejam formadas pela combinação de características de mais baixo nível. Um exemplo de rede neural de aprendizagem profunda com aplicação em imagem é a Rede Neural Convolucional.

Redes Neurais Convolucionais (RNC) são redes neurais biologicamente inspiradas que podem aprender características invariantes(LECUN; KAVUKCUOGLU; FARABET, 2010). São formadas por um conjunto de camadas que extrai características da imagem de entrada utilizando convoluções e subamostragens para que no final a rede possa inferir a qual classe a imagem pertence, como mostra a Figura 6.

Figura 6 – Modelo de arquitetura de RNC para classificar dígitos em imagens 32x32.



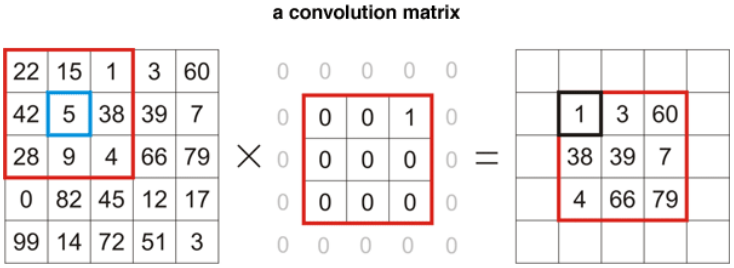
Fonte: (WARD et al., 2011)

A camada de convolução é o núcleo de construção de uma RNC. Os parâmetros da camada consistem de um conjunto de filtros que podem ser aprendidos. Durante a etapa da convolução, cada filtro realiza uma operação de convolução (Figura 7),fazendo um deslocamento sobre a entrada calculando o produto escalar em cada local como mostra a Equação 2.1, onde $f(xi, yi)$ é matriz de entrada, $g(xi, yi)$ é a matriz de saída e h é um filtro nxn. Durante o treinamento, a rede aprende filtros que ativam quando detecta algum tipo especifico de característica em alguma posição espacial da imagem.

$$g(x, y) = \sum_{i=1}^{n} \sum_{j=1}^{n} f(x - i, y - j)h(i, j) \tag{2.1}$$

A camada de subamostragem, também chamada de *pooling*, tem como objetivo criar uma representação mais compacta das características mantendo as informações mais relevantes. Isso é feito dividindo a imagem em conjuntos de retângulos que não se

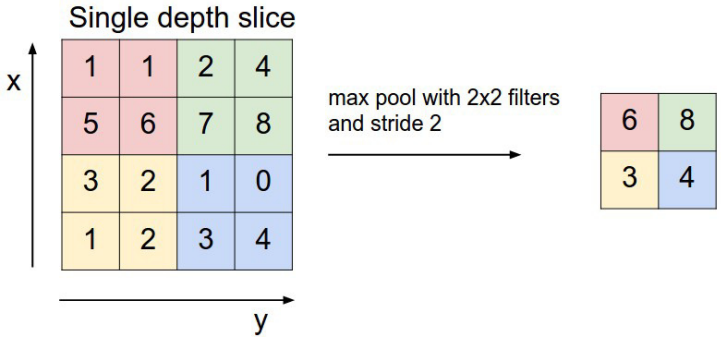
Figura 7 – Exemplo de convolução. A primeira matriz é a matriz de entrada, a segunda é a matriz *kernel* e a terceira a saída da convolução.



Fonte: (WANG; RAJ, 2015)

sobrepõem e, para cada sub-região é escolhido um valor que a representa, geralmente é escolhido o valor máximo da região,Figura 8. Essa operação reduz a quantidade de dados sem perdas significativas, o que é responsável por reduzir a complexidade computacional.

Figura 8 – Exemplo de *Pooling*.



Fonte: (CS231N.STANFORD.EDU, 2016)/

Por fim na camada totalmente conectada as características são classificadas em classes. Nessa camada também se encontra as funções de custo cujo objetivo é representar o erro entre o valor esperado e o encontrado pela rede neural e usa-lo na retropropagação para ajustar os parâmetros da rede. As funções de ativação usadas nas RNC são a *sigmoid*, Equação 4.1, tangente hiperbólica, Equação 4.2, e a ReLU(*Retified Linear Unit*), Equação 4.3. Sendo a ReLU a mais usada porque resulta no treinamento da rede neural várias vezes mais rápido sem fazer uma diferença significativa na precisão da generalização (KRIZHEVSKY; SUTSKEVER; HINTON,

2012).

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

$$f(x) = \frac{e^{x} + e^{-x}}{e^{x} + e^{-x}} \tag{2.3}$$

$$f(x) = max(0, x) \tag{2.4}$$

3 Materiais

Este capítulo apresenta os materiais utilizados no desenvolvimento deste trabalho.

3.1 RoboDeck

O RoboDeck é uma plataforma robótica desenvolvida com intuito de promover o desenvolvimento educacional e o aprendizado de conceitos gerais sobre sistemas embarcados, sensoriamento, programação, entre outros, por intermédio de um robô. O pacote educacional conta com a plataforma robótica móvel, o robô, um SDK (*Software Development kit*) para desenvolvimentos de aplicações, um ambiente de programação baseado na linguagem C/C++ e um software para testes. O robô, ilustrado na Figura 9, possui uma câmera localizada no topo do robô. Além disso, o robô possui infravermelho, sensores ultrassônicos, acelerômetro, bússola, *encoders* nos motores de tração e GPS. A comunicação com o RoboDeck pode ser feita por Wi Fi ou ZigBee.

Figura 9 – Visão geral do Robodeck.

Fonte: (XBOT, 2011)

3.1.1 Visão Geral da Arquitetura do RoboDeck

O RoboDeck foi concebido para ser modular e permitir que o usuário expanda tanto os seus comandos quanto seu aspecto físico, adicionando sensores ou outros periféricos robóticos que se queira. Partindo desse conceito de modularidade, o hardware básico de controle do robô é dividido em duas partes: o essencial e o opcional. O hardware essencial é composto por dois módulos microcontrolados: um ARM9 - para o controle das funções básicas do robô como movimentação e coleta de dados dos sensores - e um microcontrolador Jennic, próprio para o gerenciamento da comunicação com interface para redes ZigBee. Além disso, o robô conta com uma placa NanoITX com um módulo de alta performance integrado ao sistema operacional Debian Squeeze.

O hardware opcional tem o objetivo de agregar autonomia e comunicação banda larga ao robô. O software que executa nesse hardware é chamado de Módulo de Alta Performance (MAP). Para agregar autonomia ao robô, o MAP permite que os controladores carreguem aplicativos robóticos. Estes aplicativos podem, então, controlar o hardware essencial do robô através de comandos enviados diretamente ao MCS (Módulo de Controle de Sessão).

O Módulo de Controle de Sessão (MCS) representa a identidade do robô. Ele é responsável por armazenar as informações sobre o robô, tais como identificador único, nome, versão do robô, versão do protocolo de comunicação e etc. Também, este módulo, se incumbe da autenticação dos controladores que queiram comandar o robô e pelo estabelecimento das sessões de controle. Os comandos provenientes de um controlador só serão aceitos caso estejam dentro de uma sessão de controle previamente estabelecida entre o controlador e o robô. O MCS pode receber comandos provenientes do MCC (Módulo de Controle de Comunicação) ou do MAP. Os comandos recebidos podem ser delegados ao MCR ou ao MAP.

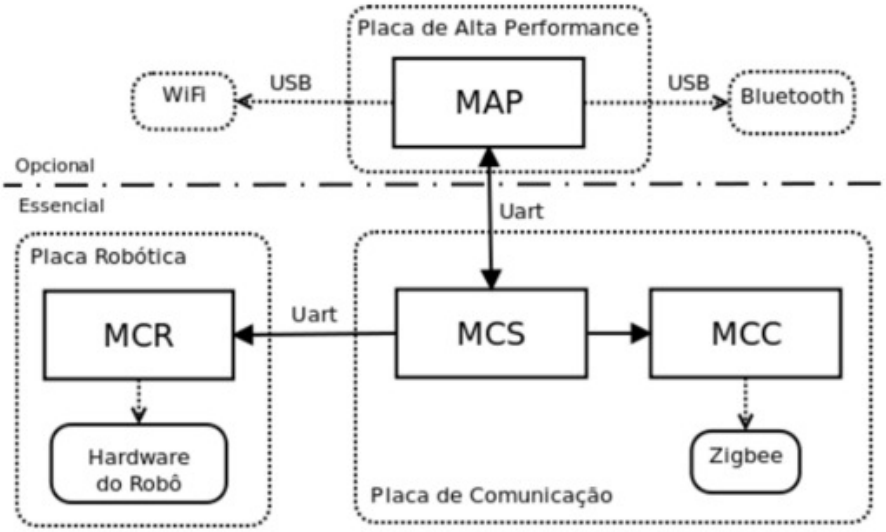
O software do RoboDeck é composto pelos *firmwares* em C/C++ dos microcontroladores no nível mais baixo. Superior a eles encontra-se o conjunto de instruções do módulo de alta performance, capaz de interagir com sistemas autônomos ou controladores externos (aplicativos), além dos SDKs, um em C e outro em Java ME, para o desenvolvimento de aplicações de controle do robô através de periféricos (XBOT, 2011).

A Figura 10 mostra as principais divisões de hardware e software do RoboDeck.

3.1.2 Sensores Utilizados

Para este trabalho foi utilizado os seguintes sensores: bússola e câmera. A câmera do RoboDeck é capaz de transmitir fotografias e vídeos para outros dispositivos.

Figura 10 – Visão geral da arquitetura do RoboDeck.



Fonte:(MUñOZ, 2011)

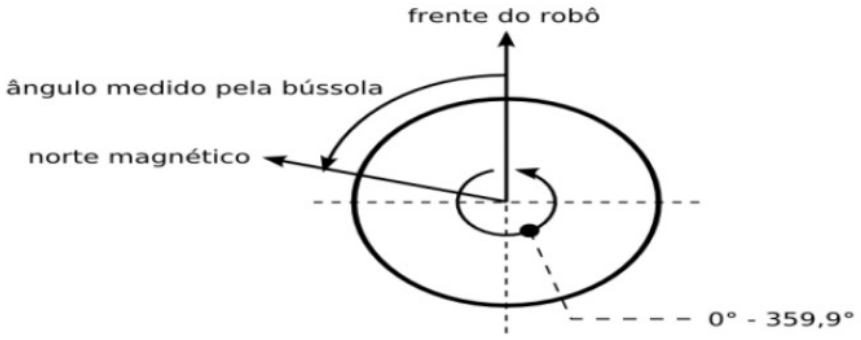
Porem para acessa-la é necessário a presença da placa de alta performance que vai transmitir as imagens geradas ao controlador por um canal Wi fi. A comunicação da câmera é intermediada pelo giga de teste, um aplicativo robótico que faz intermédio entre o computador e o RoboDeck.

Para auxiliar no sistema de navegação do RoboDeck, o primeiro sensor utilizado é uma bússola eletrônica Devantech® CMPS03, que retorna o ângulo do robô. O ângulo é medido entre a semirreta que sai do centro do robô e segue em direção à frente do robô, e a semirreta que sai do centro do robô e segue em direção ao norte magnético da terra e varia de 0º até 359,9º possuindo exatidão de 0,1º, a Figura 11 mostra o esquema de funcionamento da bússola.

3.2 ROS (*Robot Operating System*)

ROS é um meta sistema operacional de codigo aberto com bibliotecas e ferramentas para desenvolvimento de software robóticos. Ele fornece os serviços de um sistema operacional como abstração de hardware, controle de dispositivos de baixo nível, implementação de funcionalidades usualmente utilizadas, troca de mensagens entre processos e gerenciamento de pacotes(ROS.ORG, 2014b). É importante destacar que o ROS não substitui, mas funciona ao lado de um sistema operacional tradicional (O'KANE, 2014). O ROS em tempo de execução é uma rede *peer-to-peer* de

Figura 11 – Esquema de funcionamento da bússola.



Fonte:(XBOT, 2011)

processos que são fracamente acoplados usando a infra-estrutura de comunicação ROS(ROS.ORG, 2014b). A seguir são apresentados alguns conceitos do nível de grafo de computação (ROS.ORG, 2014a):

- Nós: são processos que realizam computação. O ROS é projetado para ser modular em uma escala de granularidade fina; um sistema de controle de robô geralmente compreende vários nós, como por exemplo um nó controla a rotação do robô, um nó executa a localização. Um nó ROS é escrito utilizando a biblioteca cliente do ROS, como o *roscpp* ou *rospy*.

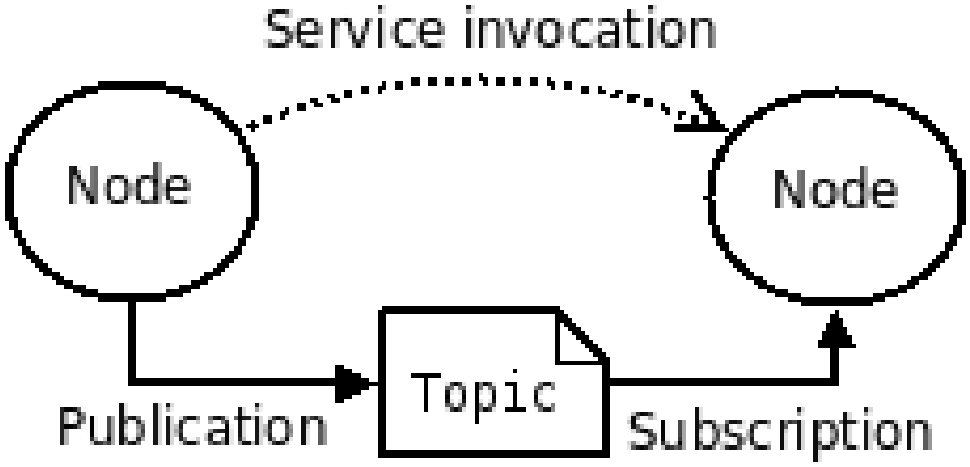
- Mensagens: Nós se comunicam uns com ou outros através da troca de mensagens. Uma mensagem é uma estrutura de dados simples que possui campos digitáveis. Tipos primitivos padrão (*integer, floating point, boolean,* etc.) são suportados, assim como *arrays* de tipos primitivos. As mensagens podem incluir estruturas arbitrariamente aninhadas e *arrays*.

- Tópicos: Mensagens são encaminhadas através de um sistema de transporte com semântica de publicação/subscrição. Um nó envia uma mensagem publicando-a para determinado tópico. O tópico é um nome que é usado para identificar o conteúdo da mensagem.Um nó que está interessado em determinado tipo de dado irá subscrever o tópico apropriado. Pode haver vários editores e assinantes simultâneos para um único tópico e um único nó pode publicar e / ou assinar vários tópicos. Em geral, editores e assinantes não estão cientes da existência uns dos outros. A ideia é dissociar a produção de informação do seu consumo. Logicamente, pode-se pensar em um tópico como um barramento de mensagens fortemente digitado. Cada barramento tem um nome e qualquer pessoa pode

se conectar ao barramento para enviar ou receber mensagens, desde que elas sejam do tipo certo.

- Serviços: O modelo de publicação / subscrição é um paradigma de comunicação muito flexível, mas o seu transporte muitos-para-muitos, unidirecional não é apropriado para interações de solicitação / resposta, que são frequentemente requeridas em um sistema distribuído. A solicitação / resposta é feita através de serviços, que são definidos por um par de estruturas de mensagem: uma para a solicitação e outra para a resposta. Um nó de fornecimento oferece um serviço sob um nome e um cliente usa o serviço enviando a mensagem de solicitação e aguardando a resposta, Figura 12. As bibliotecas de cliente ROS geralmente apresentam essa interação ao programador como se fosse uma chamada de procedimento remoto.

Figura 12 – Diagrama do conceito básico de ROS.



Fonte:(ROS.ORG, 2014a)

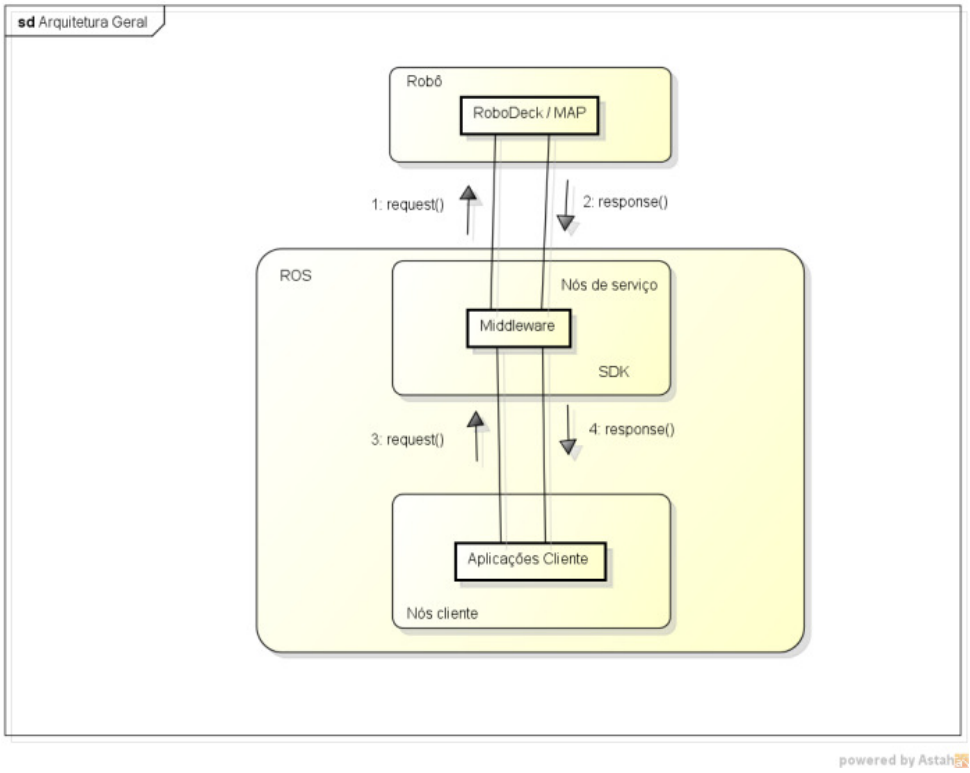
3.3 *Middleware* de Integração ROS-RoboDeck

O *middleware* desenvolvido por (GASHU, 2014) funciona como um intermediador da comunicação dos programas clientes escritos no ROS com o RoboDeck. Ele faz a tradução tanto da requisição do cliente para o robô quanto da resposta do robô para o cliente.

Tanto o *middleware* quanto as aplicações clientes utilizam o *framework* ROS para permitir a comunicação. Assim, eles devem ser executados como nós da rede do ROS, para que o processo *core* do ROS possa gerenciar a comunicação entre o nó do

middleware e os nós clientes. Além disso, o *middleware* pode ser executado tanto na placa em que é executado o MAP quanto em um computador externo. A comunicação entre o robô e o ROS funciona através de requisição/resposta. O cliente que quer se comunicar com o robô deve solicitar uma requisição através de um nó do ROS. Ele faz a tradução tanto da requisição do cliente para o robô quanto da resposta do robô para o cliente. Tanto o *middleware* quanto as aplicações clientes utilizam o ROS para permitir a comunicação, assim, eles devem ser executados como nós de rede do ROS. (GASHU, 2014). A Figura 13 mostra o diagrama de comunicação do *middleware*.

Figura 13 – Visão geral do *middleware* de integração ROS-Robodeck.



Fonte: Gashu, 2014

3.4 Keras

Keras é uma biblioteca de redes neurais de alto-nível, escrita em Python e capaz de executar em cima do *Theano*, uma biblioteca de computação numérica para Python. Desenvolvida para permitir a escrita rápida e fácil através da modularidade total, minimalista e extensível. Tem suporte para redes convolucionais e redes recorrentes, assim com a combinação dos dois, funcionando tanto na CPU quanto na GPU. Os princípios orientadores do Keras são:

- Modularidade: Um modelo é entendido como uma sequencia ou um grafo de módulos autônomos, totalmente configuráveis, que podem ser conectados com o menor número possível de restrições. Assim é possível combinar camadas neurais, funções de ativação, funções de custo, entre outros módulos autônomos, para criar novos modelos.
- Minimalismo: Cada modulo deve ser mantido curto e simples. Cada pedaço de código deve ser transparente na primeira leitura.
- Extensibilidade fácil: Novos módulos são simples de adicionar, e os módulos existente fornecem amplos exemplos, tornando Keras adequado para pesquisas avançadas.
- Trabalha com Python: Os modelos são descritos em código Python, que é compacto, mais fácil de depurar e permite fácil extensibilidade;

4 Mapeamento e Localização Baseado em Visão

Neste capitulo é descrito o desenvolvimento da implementação para mapeamento e localização utilizando a visão computacional.

4.1 Visão Geral

A proposta emprega visão computacional para mapeamento e localização em ambientes internos. É utilizado visão computacional para mapear coordenadas do ambiente e recuperar marcações numeradas a fim de ter referencias no ambiente para se localizar.

O mapeamento de coordenadas é realizado transformando coordenadas de pixel, obtida por processamento digital da imagem, em coordenadas de distancia real. O mapa é construído iterativamente, usando informações sensoriais do robô para obter sua orientação no ambiente e construir o mapa global.

Para localização, as marcações numeradas que são reconhecidas e tem suas posições armazenadas para utiliza-las como ponto de referencia no mapa. Para o reconhecimento, a placas são extraídas com processamento de imagem e classificadas com uma Rede Neural Convolucional, treinada previamente.

O SDK e o *middleware* de integração Ros-RoboDeck servem como intermediário para a comunicação entre o RoboDeck e o computador, sendo responsável por enviar os comandos de movimentação e receber dados dos sensores. O Giga de Teste do SDK é utilizado para obter as imagens da câmera do robô e o *middleware* é usado para fazer a movimentação geral do RoboDeck.

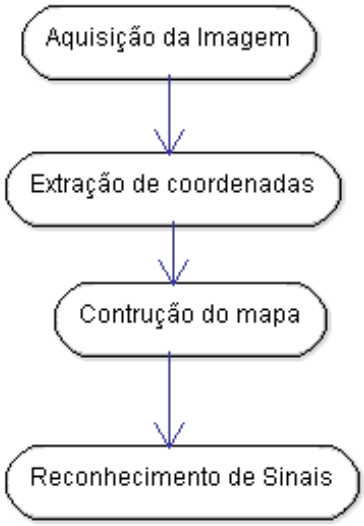
No computador ocorre a fase de processamento de dados, como mostra a Figura 14. Primeiramente, é adquirida a imagem câmera do robô. Em seguida ocorre o processamento da imagem para extrair as coordenadas e construir o mapa.E então é obtida os sinais numeradas para auxiliar na localização do robô.

4.2 Extração de Coordenadas

4.2.1 Segmentação

A segmentação da imagem é feita para que seja possível adquirir as características necessárias para a a construção do mapa. O algoritmo de extração é realizado

Figura 14 – Diagrama de visão computacional.



Fonte: Elaborado pelo autor

em etapas como mostra a Figura 15. Inicialmente é aplicado um filtro *Blur* na imagem para borrá-la afim de reduzir possíveis ruídos para a etapa de detecção de bordas. Na detecção de bordas é usado o método *Canny*, calculando o limiar máximo com o método *Otsu* e usando metade desse valor como limiar mínimo. Isso é feito visando destacar a fronteira entre o terreno e as paredes.

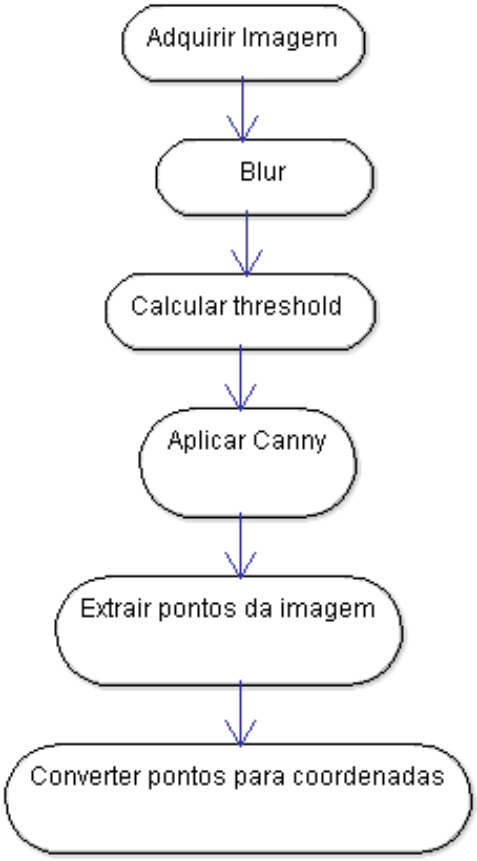
Apos a detecção de borda é utilizado um algoritmo para obter os pontos da borda entre o chão e a parede/obstaculo.Essa abordagem consiste em varrer a imagem da esquerda para a direita e de baixo para cima procurando pixel que façam fronteira entre o chão e a parede/obstaculo, obtendo um conjunto de pontos como mostra a Figura 16. Em seguida, com a lista do pontos de coordenadas em pixel, é feita o mapeamento de coordenadas.

4.2.2 Mapeamento de Coordenada

O mapeamento de coordenada é responsável por transformar as coordenadas da imagem, obtida na extração, para coordenadas reais de distancia. A transformação de coordenadas de pixel para coordenadas de espaço é feita usando trigonometria(Figura 17). As equações usadas são as 4.1 e 4.2.

$$Y = H\tan(\alpha + \frac{y}{k}a) \tag{4.1}$$

Figura 15 – Diagrama de extração de pontos de coordenada



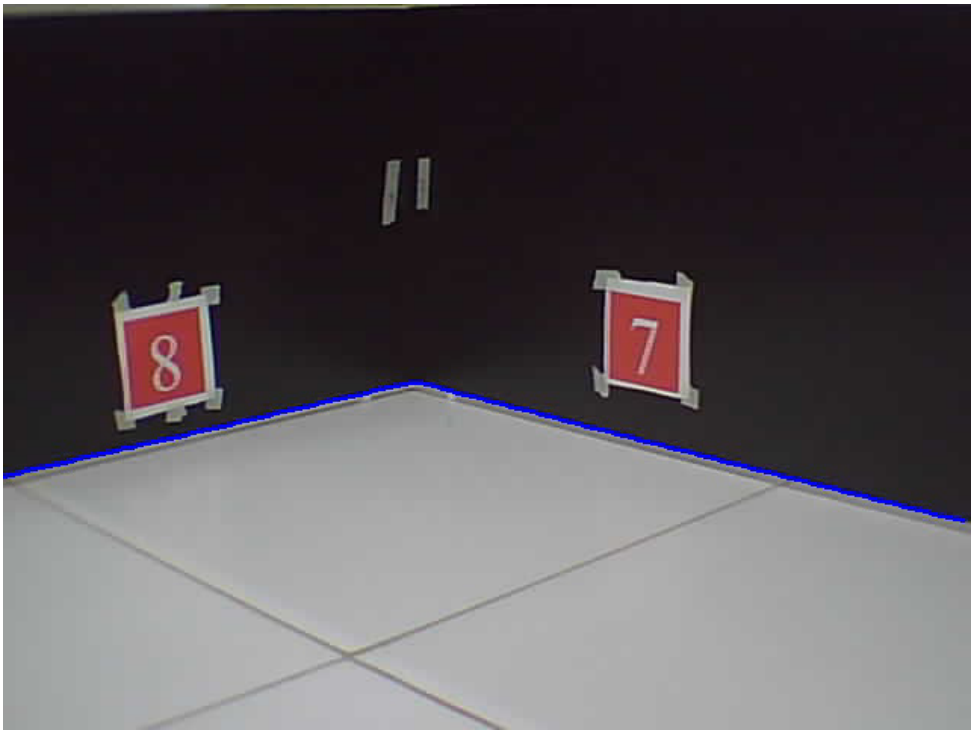
Fonte: Elaborado pelo autor

$$X = \sqrt[2]{Y^2 + H^2}\tan(\beta(\frac{x}{C} - 1)) \tag{4.2}$$

onde H é a altura da câmera até o terreno, α é o ângulo calculado usando a tangente da distancia do robô até o fundo da imagem e a altura do câmera, y é o valor de pixel de y contando a partir do fundo da imagem e k é uma constante que indica a razão de pixel por ângulo. Na 4.2 o β é o ângulo de abertura da câmera, tendo o valor de 20º para a câmera do RoboDeck, x é o valor de pixel contando da esquerda para direita e C é a metade da largura da imagem.

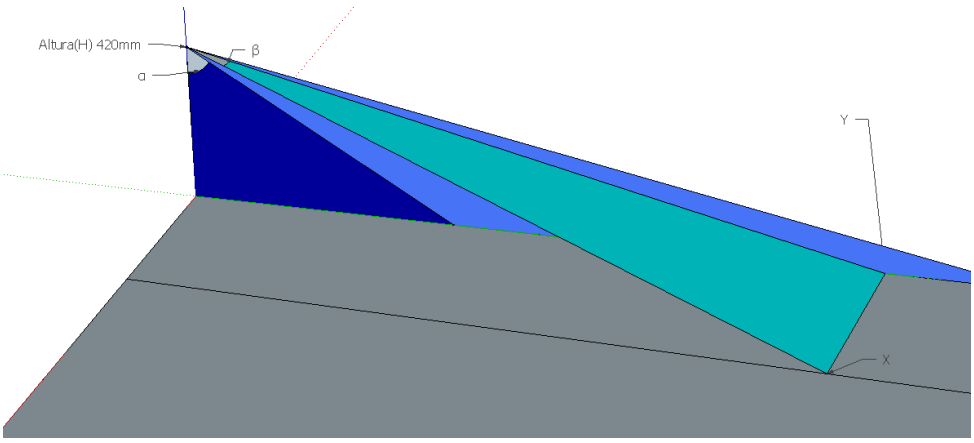
Como resultado do mapeamento de coordenadas tem-se o conjunto de pontos em coordenadas do espaço, como mostra a Figura 18. Esse pontos são como mini mapas, criado a partir da imagem do robô, e são usados como base para a representação no mapa global.

Figura 16 – Bordas detectadas



Fonte: Acervo do autor

Figura 17 – Projeção do mapeamento de coordenadas

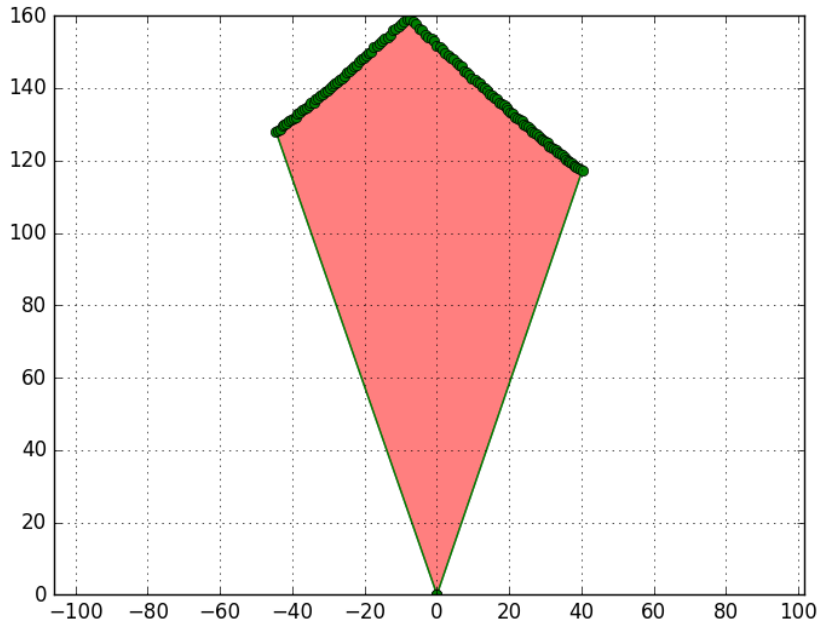


Fonte: Elaborado pelo autor

4.3 Construção do Mapa

Para representação do mapa é empregada grade de ocupação(*occupancy grid*). Na grade de ocupação, o mapa é representado por uma matriz de duas dimensões, como um campo uniformemente espaçado, onde cada célula da matriz contem um valor

Figura 18 – Coordenadas resultantes do mapeamento



Fonte: Acervo do autor

de ocupação, definido para representar se a célula está livre ou ocupada. A atualização de cada célula é feita usando os pontos de coordenadas de distancia encontrados pela câmera e o ângulo de orientação do robô obtido por meio da bússola, γ, como visto nas equações 4.3 e 4.4. X e Y são calculados usando as equações 4.1 e 4.2 e D é a distancia da câmera para o centro do robô. Os valores encontrados são discretizados para encontrar o valor correspondente na grade de ocupação.

$$Xmapa = X\cos(\gamma) - (Y + D)\sin(\gamma) \tag{4.3}$$

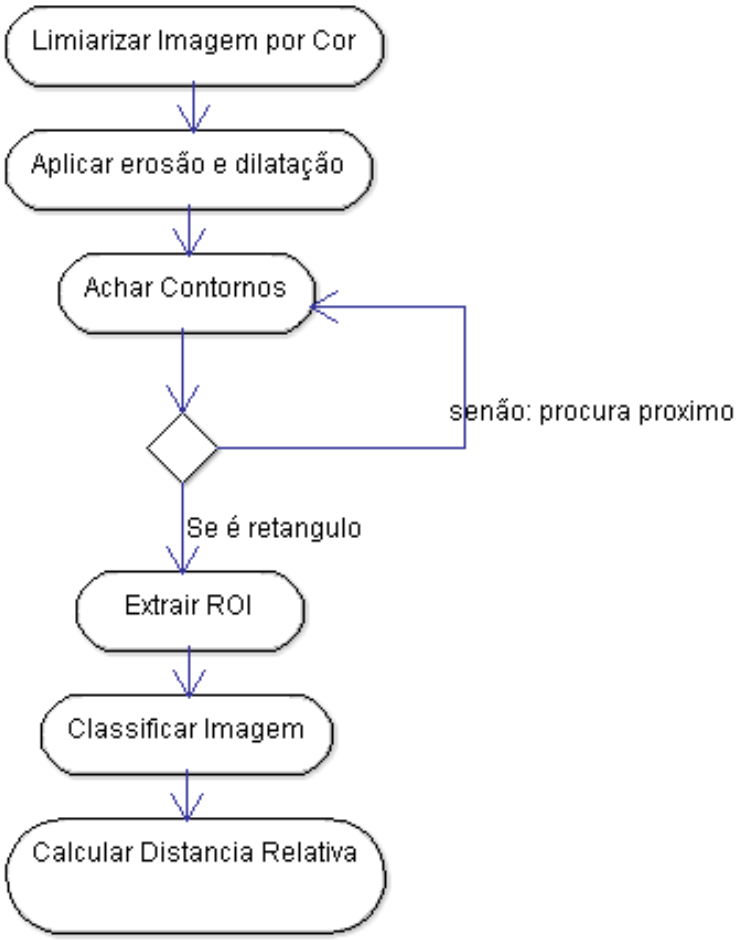
$$Ymapa = X\sin(\gamma) + (Y + D)\cos(\gamma) \tag{4.4}$$

4.4 Reconhecimento das Marcações Numeradas

Após a extração de características da imagem, é realizada a extração dos sinais existentes na imagem. O algoritmo de reconhecimento de sinais é processado em etapas, conforme a Figura 19. Na primeira etapa, é feita uma limiarização por cor na imagem, que evidência as marcações numeradas, seguido de uma dilatação e erosão

na imagem para retirar possíveis ruídos. O próximo passo é procurar por contornos na imagem, delimitar o contorno em um retângulo e extrair a região de interesse (ROI). Por fim a marcação numerada é classificado na RNC.

Figura 19 – Diagrama do processo de reconhecimento



Fonte: Elaborado pelo autor

Para fazer o classificador, foi empregada a biblioteca de aprendizado profundo Keras. A arquitetura de rede utilizada foi a *Lenet* , que utiliza as seguintes camadas: INPUT => CONV => RELU => POOL => CONV => RELU => POOL => FC => RELU => FC. O treinamento foi realizado usando 11 classes, 10 de dígitos e uma de erro, com 100 exemplos de cada classe, totalizando 1100 imagens. Cada exemplo é representado por uma imagem em escala de cinza de dimensão 28x28.

O treinamento é processado em 100 iterações ou épocas, onde todo o conjunto de treinamento é apresentado a rede. O aprendizado da rede foi efetuado por atualização em lote, os pesos são ajustado somente após a apresentação de todos os exemplos que constituem o lote. O tamanho do lote escolhido foi de 20 exemplos,

devido ao tamanho reduzido do conjunto de treinamento. O teste da rede foi executado utilizando 49 imagens de cada classe e o resultado do teste foi de 98,0% de acurácia. Na Figura 20 é apresentado um exemplo da utilização do classificador, na imagem da direita todos foram classificados corretamente e na imagem da esquerda o 8 não foi detectado e o 9 foi classificado com 0.

Figura 20 – Reconhecimento da marcação numerada.

Fonte: Acervo do autor

4.5 Recuperando a Posição pela Marcação

Para recuperar a posição do robô em relação ao mapa, é empregado a posição relativa do sinal numerado, sua posição absoluta no mapa global e a orientação do robô, obtida por meio da bússola interna. Após reconhecer, o sinal numerado é verificado se é a primeira vez que o sinal foi encontrado, se for, é calculada a posição absoluta da marcação no mapa global e armazenada para posteriormente utilizar a marcação como ponto de referência.

Se o robô for colocado em um ponto qualquer do ambiente e reconhecer um sinal já visto, ele recuperará sua posição no mapa global. Para isso é efetuado o calculo da posição em relação ao sinal. Com o seu ângulo de orientação é calculado a coordenada do robô em relação ao marco, usando rotação (equações 4.3 e 4.4). Assim utilizando a posição global do sinal é feita a localização do RoboDeck (equações 4.5 e 4.6).

$$X_{robo} = X_{placa} - X_{mapa} \tag{4.5}$$

$$Y_{robo} = Y_{placa} - Y_{mapa} \tag{4.6}$$

5 Resultados Experimentais

Visando testar a metodologia proposta foram realizados alguns experimentos em ambiente interno. O objetivo específico dos testes é verificar a abordagem do mapeamento em uma posição, o resultado do mapeamento quando o robô tem que mapear duas regiões vizinhas e a abordagem da localização.

5.1 Descrição do Ambiente

O trabalho foi desenvolvido em um ambiente interno. O ambiente possui terreno plano e foi delimitado por placas e uma parede. Possui aberturas laterais entre as placas e a parede, com tamanho de 140cm. As placas laterais possuem dimensões internas de 264x45cm e a placa frontal tem dimensão interna de 270x45 cm.

Para o segundo e terceiro teste foram utilizadas caixas que representam o obstaculo para o robô. A caixa da esquerda tem dimensão de 60x35cm e a caixa da direita tem dimensão de 48x33 cm. A Figura 21 mostra as duas configurações de ambiente utilizadas nos testes.

Figura 21 – Configuração do ambiente de teste. Sem as caixas (esquerda) e com as caixas (direita)

Fonte: Acervo do autor

5.2 Movimentação no Ambiente

Para a movimentação do RoboDeck foi utilizado os comandos do *middleware* de integração Ros-Robodeck por oferecer maior liberdade de movimentação ao Robodeck. Os comandos de movimento no *middleware* permitem que escolha a quantidade de

movimento feita pelo robô e o comando de rotação possibilita a escolha do ângulo de rotação sobre o eixo.

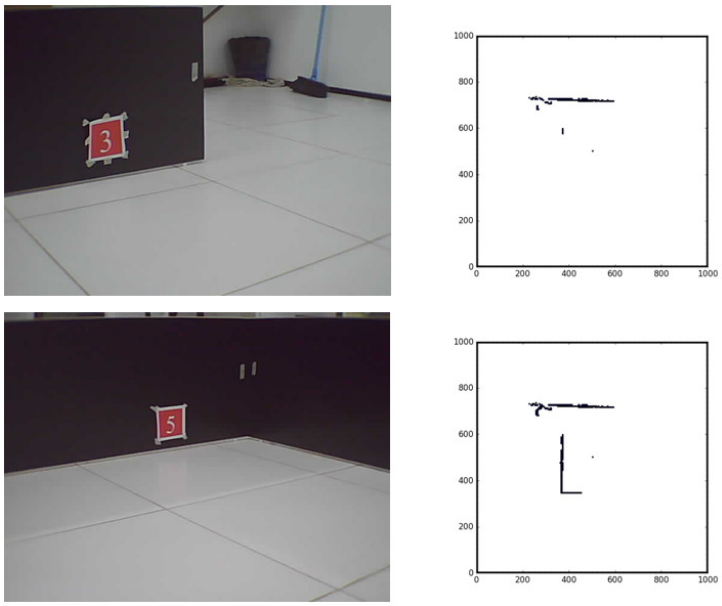
O ângulo de leitura da bussola é convertido para que a orientação para a parede seja igual ao ângulo zero. Isso é feito armazenando o angulo do robô obtido quando o mesmo está virado para a parede e usando esse valor para subtrair pelas próximas leituras.

5.3 Procedimentos Experimentais

5.3.1 Teste 1 - Mapeamento na Área de Teste

O primeiro teste tem como objetivo verificar a aplicação da proposta para o mapeamento. O mapeamento ocorreu em 15 etapas, onde em cada etapa o robô realiza um movimento de rotação em torno do seu eixo e ler o angulo da bússola. O robô se encontra no mapa global na posição (500,500) e realiza rotação tendo como ângulo de orientação em cada etapa da rotação realizada pelo robô os seguintes valores: [-4, 14, 40, 53, 75, 95, 115, 144, 175, 204, 233, 255, 311, 294, 342. A imagem adquirida em cada etapa é processada e o mapa é criado incrementalmente. Na Figura 22 é mostrado a etapa 3 e 8 da construção do mapa.

Figura 22 – Mapeamento incremental: etapa 3 e 8 do mapeamento

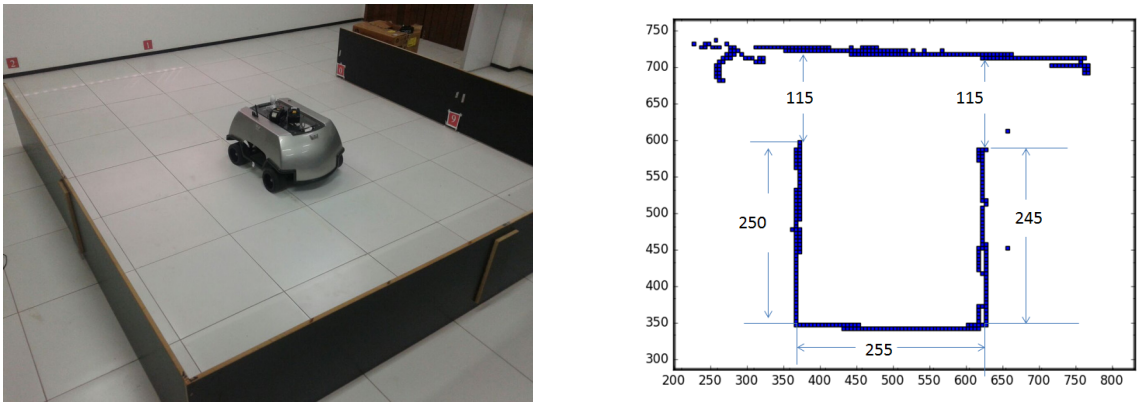


Fonte: Acervo do autor

O mapa resultante desse mapeamento é mostrado na Figura 23. O resultado mostra que o robô pode fazer o mapeamento do ambiente de teste. Embora haja erro de

dimensionamento, tendo a parede da lateral direita apresentada a menor variação com diferença de 14 cm e a maior diferença foi notada na distância entre a parede lateral e a parede de cima, sendo de 21cm. Isso ocorre devido aos parâmetros usados no mapeamento de coordenada serem imprecisos e essa imprecisão aumenta a medida que aumenta a distância do robô para os obstáculos mapeados.

Figura 23 – Mapeamento completo.



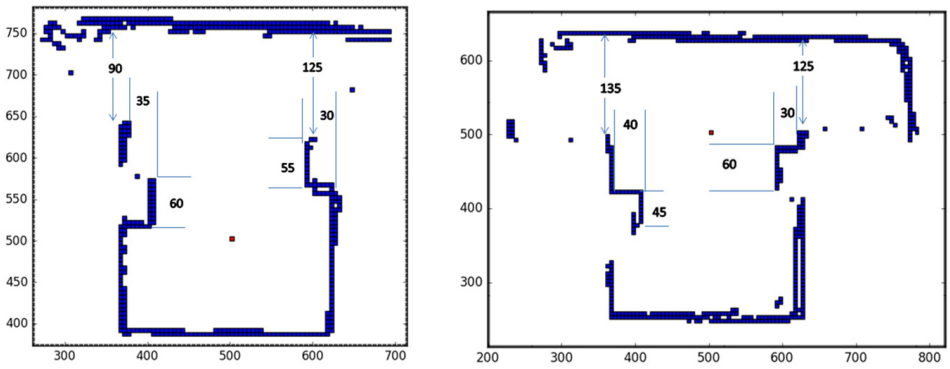
Fonte: Acervo do autor

5.3.2 Teste 2 - Mapeamento com Obstáculos na Lateral

O segundo teste tem como objetivo verificar o resultado do mapeamento quando tem objetos na lateral do ambiente, simulando uma parede, o que impede que o robô tenha uma visão completa do ambiente. Nesse experimento o robô faz o mapeamento a partir de duas posições (Figura 24) e para criar o mapa foi necessário juntar as informações dos dois mapeamentos. Isso foi feito com o robô obtendo as informações do mapeamento na posição 1, em seguida movimentando-se 150cm a frente e fazendo a leitura das informações do mapeamento na posição dois. Os ângulos lidos pelo robô na posição 1 é: [357, 14, 33, 46, 60, 71, 92, 113, 133, 160, 180, 200, 224, 247, 273, 287, 310, 331, 350, 357]. E na posição 2 é: [0, 20, 30, 49, 105, 123, 156, 170, 185, 204, 223, 240, 256, 278, 287, 315, 333, 353, 0]).

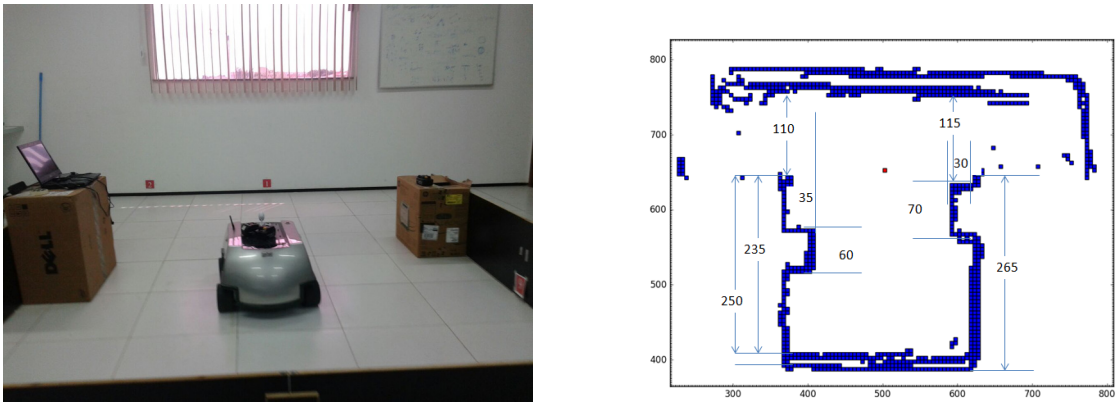
O resultado do mapa completo é mostrado na Figura 25. Pode se percebe claramente a influencia dos erros do mapeamento de coordenada sobre o resultado final do mapa. As paredes de cima e de baixo mostram uma variação grande da posição, variando em 15 cm, enquanto as paredes laterais tem uma variação menor, isso ocorre devido ao deslocamento do robô de uma posição a outra ser apenas no eixo y.

Figura 24 – Mapeamento da posição 1(esquerda) e posição 2(direita)



Fonte: Acervo do autor

Figura 25 – Mapeamento completo teste 2



Fonte: Acervo do autor

5.3.3 Teste 3 - Localização na Área de Teste

O terceiro teste tem como objetivo verificar o resultado da localização. O teste foi dividido em duas fases.

Na primeira fase foi feita uma varredura utilizando as imagens do mapa 1 para que fosse validado o processo de localização. O Robô foi posicionado no mapa na posição (500,500) e foi feita a estimativa durante a construção do mapa, para verificar a acurácia do sistema. Nessa primeira fase as marcações encontradas pelo são armazenadas e utilizadas para estimar a sua coordenada no mapa. Na tabela 1 tem os valores estimado para a coordenada do robô juntamente com o ângulo de orientação do robô.

A segunda fase foi testada num ponto diferente do inicial para que o robô

Tabela 1 – Localização estimada

rótulo	**Ângulo do Robô**	**Posição X**	**Posição Y**
1	14	499,67	497,33
2	40	495,38	495,86
3	53	503,64	502,05
0	294	498,43	502,57
1	342	509,72	503,28
Ponto médio	-	501,37	500,29

Fonte: Elaborado pelo autor

pudesse se localizar usando as informações das placas reconhecidas na primeira fase. O robô se movimentou 100cm a frente (deslocamento no eixo y do mapa global) do ponto inicial e rotacionou em busca das placas numeradas. Na tabela 2 mostra-se o resultado da localização na nova posição, cada coordenada é estimada usando apenas o conhecimento prévio que o robô adquiriu do ambiente.

Tabela 2 – Localização estimada na posição dois

rótulo	**Ângulo do Robô**	**Posição X**	**Posição Y**
1	0	513,20	585,5
2	49	506,15	584,18
3	105	506,85	602,33
6	170	503,72	590,7
7	204	500,74	593,06
9	225	489,44	582,6
Ponto Médio	-	503,35	589,73

Fonte: Elaborado pelo autor

Pode-se perceber que a estimativa de localização utilizando as placas numeradas manteve-se estável com resultados próximos. Isso mostra que uma placa é suficiente para recuperar sua posição, pois com somente a posição calculada previamente das marcações numeradas foi-se capaz de estimar sua localização no mapa do ambiente.

6 Conclusão

A exploração autônoma em robótica móvel requer a construção progressiva de mapas e auto localização nesse mapa, a partir de dados obtidos durante a movimentação do robô em um dado ambiente. O mapeamento e localização é uma tarefa essencial, pois permite mover-se a partir de uma posição desconhecida em um ambiente desconhecido de forma segura, usando seus sensores para obter informações sobre o ambiente a fim de detectar e evitar obstáculo.

Neste trabalho, foi prototipado um sistema de mapeamento e localização para o kit robótico Robodeck usando técnicas de visão computacional. Para extrair informações sobre o ambiente, imagens obtidas a partir da câmera do equipamento foram digitalmente processadas usando técnicas como filtros, limiarização, *Canny* e segmentação.

Para o mapeamento foi utilizado as razões trigonométricas para mapear as informações do ambiente adquiridas pelas imagens em coordenadas de distancia. O mapa foi construído iterativamente utilizando as coordenadas de distancia e a orientação do robô.

Para obter a localização no ambiente foi utilizado marcações numeradas como pontos de referencia. A classificação dos marcadores foi realizado usando-se uma Rede Neural Convolucional, treinada para reconhecer os números. A rede treinada obteve acurácia de 98% durante o treino. O reconhecimento das marcações no ambiente apresentou algumas falhas, como a classificação errada de um marcador e o não reconhecimento do marcado no ambiente, como visto na seção 4.3.

Em ambiente controlado, foram realizados experimentos, em que foi verificado a performance do método. No mapeamento, o mapa construído apresentou dimensão diferente do ambiente de teste, tendo um erro de 15cm no eixo x e 35cm no eixo y em um ambiente de 270x400 cm. A localização apresentou performance satisfatória, tendo conseguido estimar sua posição de forma estável usando as marcações como referência.

Para trabalho futuros os seguintes pontos podem ser explorados: desenvolver estudos para o problema de SLAM(*Simultaneous Localization and Mapping*) para implementar um modelo de navegação autônoma utilizando visão computacional; Estudar métodos que possam permitir uma melhor calibração da visão do ambiente para minimizar os erros obtidos no mapeamento de coordenada; investigar a abordagem usando fusão de dois mapas, com usos de filtros, como o *Expectation Maximization* , para minimizar os erros na construção do mapa quando o robô se desloca no ambiente.

Referências

BURGARD, W.; FOX, D.; JANS, H.; MATENAR, C.; THRUN, S. Sonar-based mapping with mobile robots using em. In: MORGAN KAUFMANN PUBLISHERS, INC. *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*. [S.l.], 1999. p. 67–76. Citado na página 14.

CS231N.STANFORD.EDU. *Convolutional Neural Networks (CNNs / ConvNets)*. 2016. <http://cs231n.github.io/convolutional-networks/>. [Acessado em 01/01/2017]. Citado na página 22.

FERREIRA, C. S. Implemantação do algoritmo de subtração de fundo para detecção de objetos em movimento, usando sistemas reconfiguráveis. 2012. Citado na página 18.

FILHO, O. M.; NETO, H. V. *Processamento digital de imagens*. [S.l.]: Brasport, 1999. Citado na página 17.

FOX, D.; BURGARD, W.; THRUN, S. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, v. 11, p. 391–427, 1999. Citado na página 14.

GASHU, T. Y. *Integração do sistema operacional ROS com o módulo de alta performance Robodeck*. São José dos Campos, SP: [s.n.], 2014. Citado 2 vezes nas páginas 28 e 29.

GUTMANN, J.-S.; KONOLIGE, K. Incremental mapping of large cyclic environments. In: IEEE. *Computational Intelligence in Robotics and Automation, 1999. CIRA'99. Proceedings. 1999 IEEE International Symposium on*. [S.l.], 1999. p. 318–325. Citado na página 14.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105. Citado na página 23.

LECUN, Y.; KAVUKCUOGLU, K.; FARABET, C. Convolutional networks and applications in vision. In: IEEE. *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. [S.l.], 2010. p. 253–256. Citado na página 21.

MUñOZ, M. E. de S. *Projeto de Software RobotDeck versão 0.2*. [S.l.], 2011. Citado na página 26.

O'KANE, J. M. A gentle introduction to ros. 2014. Citado na página 26.

ROS.ORG. *ROS Concepts*. 2014. <http://wiki.ros.org/ROS/Concepts>. [Acessado em 01/01/2017]. Citado 2 vezes nas páginas 27 e 28.

ROS.ORG. *ROS Introduction*. 2014. <http://wiki.ros.org/ROS/Introduction>. [Acessado em 01/01/2017]. Citado 2 vezes nas páginas 26 e 27.

SE, S.; LOWE, D.; LITTLE, J. Vision-based mobile robot localization and mapping using scale-invariant features. In: IEEE. *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*. [S.l.], 2001. v. 2, p. 2051–2058. Citado na página 15.

SUJAN, V. A.; MEGGIOLARO, M. A.; BELO, F. A. Mobile robot simultaneous localization and mapping using low cost vision sensors. In: SPRINGER. *Experimental Robotics*. [S.l.], 2008. p. 259–266. Citado na página 17.

WANG, H.; RAJ, B. A survey: Time travel in deep learning space: An introduction to deep learning models and how deep learning models evolved from the initial ideas. *arXiv preprint arXiv:1510.04781*, 2015. Citado na página 22.

WARD, J.; ANDREEV, S.; HEREDIA, F.; LAZAR, B.; MANEVSKA, Z. Efficient mapping of the training of convolutional neural networks to a cuda-based cluster. Citeseer, 2011. Citado na página 21.

WOLF, D. F.; SIMÕES, E.; OSÓRIO, F. S.; JUNIOR, O. T. Robótica móvel inteligente: Da simulação às aplicações no mundo real. In: *Mini-Curso: Jornada de Atualização em Informática (JAI), Congresso da SBC*. [S.l.: s.n.], 2009. p. 13. Citado na página 14.

XBOT. Apostila de software do robodeck versão 1.1. [S.l.], 2011. Citado 3 vezes nas páginas 24, 25 e 27.