
# UNIVERSIDADE FEDERAL DO MARANHÃO

**Programa de Pós-Graduação em Ciência da Computação**

**ANDRÉ FELIPE DA SILVA OLIVEIRA**

## *CASAMENTO DE MAPAS OBTIDOS POR VÁRIOS ROBÔS*

**São Luís**
**2018**

**ANDRÉ FELIPE DA SILVA OLIVEIRA**

# CASAMENTO DE MAPAS OBTIDOS POR VÁRIOS ROBÔS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Maranhão, como requisito necessário para obtenção do título de mestre em Ciência da Computação.

Orientador: Prof. Dr. Areolino de Almeida Neto

**São Luis - MA**
**2018**

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

Felipe da Silva Oliveira, André.
Casamento de Mapas Obtidos por Vários Robôs / André Felipe da Silva Oliveira. – 2018.
86 p.

Orientador(a): Areolino de Almeida Neto.
Dissertação (Mestrado) – Programa de Pós-graduação em Ciência da Computação/ccet, Universidade Federal do Maranhão, São Luís, 2018.

1. Casamento de mapas. 2. Encaixe de mapas. 3. Fusão de mapas. 4. Mapeamento. 5. Navegação de robôs. I. de Almeida Neto, Areolino. II. Título.

**André Felipe da Silva Oliveira**

# Casamento de Mapas Obtidos por Vários Robôs

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Maranhão, como requisito necessário para obtenção do título de mestre em Ciência da Computação.

Aprovado em 7 de maio de 2018

BANCA EXAMINADORA

Prof. Dr. Areolino de Almeida Neto (Orientador)
Universidade Federal do Maranhão - UFMA

Prof. Dr. Paulo Rogério de Almeida Ribeiro
Universidade Federal do Maranhão - UFMA

Prof. Dr. Sérgio Ronaldo Barros dos Santos
Universidade Federal de São Paulo - UNIFESP

Prof. Dr. Will Ribamar Mendes Almeida
Centro Universitário do Maranhão - Universidade CEUMA

São Luis - MA
2018

## AGRADECIMENTOS

Agradeço a minha mãe e meu pai, Maria do Socorro Alves da Silva e Evaldo Vieira Oliveira, por terem sido os principais responsáveis por eu ter chegado até aqui. Agradeço a minha futura esposa Thayane Soares da Silva, por sempre ter me dado forças até mesmo nos momentos mais difíceis. Agradeço as minhas irmãs, minhas tias e primos, que sempre estiveram ao meu lado durante todo esse tempo.

Agradeço ao meu orientador Areolino de Almeida Neto e ao Programa de Pós-Graduação em Ciência da Computação que me proporcionaram a chance de obter este título de mestre.

Agradecimentos também aos amigos que fiz ao longo dessa caminhada, especialmente a Diego de Oliveira Dantas e Moisés Laurence de Freitas Lima Junior, que também passaram pelo laboratório InovTec, sob a orientação de Areolino e que tanto que ajudaram para que esse desafio pudesse ser concluído com sucesso.

Agradeço também ao curso de Engenharia da Computação da UEMA e seu corpo docente, que nesse ultimo ano do mestrado acolheram-me como professor de robótica e em especial ao coordenador do curso, professor Dr. Rogério Moreira Lima, que me ensinou e mostrou o caminho de ser um bom profissional na docência.

Agradecimentos especiais à CAPES e FAPEMA pelo financiamento em forma de bolsa de estudos.

## RESUMO

Este trabalho propõe um método para casamento de mapas tipo *occupancy grid* em forma de imagem, no qual uma nova forma de busca de semelhanças é desenvolvida. Os mapas usados neste trabalho foram obtidos via SLAM realizado por diferentes robôs, individualmente, e com poses iniciais desconhecidas. O método concebido é baseado em técnicas de processamento de imagem, onde se extraem características dos mapas fornecidos pelos robôs e cria-se um alfabeto para cada mapa através das relações entre essas características. Candidatos a pontos de casamento são encontrados a partir da comparação entre os membros dos alfabetos gerados.

Após a etapa de comparação, realiza-se a verificação dos casamentos possíveis a partir dos pontos candidatos, onde se aplica uma métrica de verificação de similaridade. Desta forma, os pontos de casamento que proporcionarem uma maior similaridade são escolhidos como melhores pontos atuais. Estas operações são repetidas a cada vez que são fornecidas novas atualizações dos mapas ao método. Assim, atualizam-se as taxas de similaridades dos melhores pontos da iteração anterior e calculam-se novos melhores pontos de casamento para a iteração atual, desta forma os pontos de casamento que possuírem maior taxa de similaridade entre a iteração anterior e a atual são escolhidos como melhores pontos atuais.

Para realização dos testes, foram utilizados diversos ambientes gerados de forma aleatória e com quantidade variada de compartimentos. Foram também realizados diversos mapeamentos para cada ambiente. Os testes consideraram as poses iniciais dos robôs desconhecidas e a ausência de comunicação entre os robôs para compartilhamento de quaisquer informações que não fossem o estado atual de seus mapas.

Os resultados obtidos são promissores, visto que na maioria dos testes realizados se obteve sucesso ao encontrar o casamento correto entre os mapas. Em alguns desses casos, encontrou-se o casamento mesmo em situação de grande deformação no mapa, devido ao escorregamento das rodas dos robôs durante o mapeamento e ruídos nas medições.



**Palavras-chaves**: Mapeamento, Navegação de Robôs, Casamento de Mapas, Fusão de Mapas, Encaixe de Mapas.

## ABSTRACT

This work proposes a matching method of occupancy grid maps in image form, in which a new way of searching for similarities is developed.The maps used in this work were obtained via SLAM and were performed individually by different robots in unknown initial poses. The designed method is based on image processing techniques in which the maps characteristics are provided by robots and, through the relations between these characteristics, an alphabet is created to each map. The matching points candidates are found from the comparison between the members of the alphabet generated.

After the comparison, it is possible to verify possible matches from the candidate points, where a similarity verification metric is applied. In this way, matching points that provide greater similarity are chosen as best current points. These operations are repeated every time new map updates are provided to the method. Thus, the similarity rates of the best points of the previous iteration are updated and new best matching are calculated for the current iteration, so that the matching points with the highest similarity rate between the previous and the current iteration are chosen as best current points.

Several randomly generated environments with a varied amount of compartments were used to perform the tests. Several mappings were also performed to each environment. The tests considered the initial poses of unknown robots and the lack of communication between robots to share any information other than the current state of their maps.

The results obtained are promising, since in the majority of the tests performed, it was successful in finding the correct matching between the maps. In some of these cases, the matching was found even in a situation of great deformation on the map, which is due to the skidding of the robots and noises in the measurements.



**Key-words**: Mapping, Robot Navigation, Map Matching, Map Merging, Map Fitting.

## LISTA DE ILUSTRAÇÕES

## LISTA DE ALGORITMOS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| SLAM | *Simultanous Localization and Mapping* |
| EKF | *Extended Kalman Filter* |
| ROS | *Robot Operating System* |
| AMCL | *Adaptative Monte Carlo Localization* |

**SUMÁRIO**

# 1 Introdução

Sempre que se fala em robôs autônomos, pensa-se que ele deve ser capaz de reconhecer e atuar no ambiente ao seu redor, mesmo que esse ambiente seja inicialmente desconhecido. Quando se fala em robôs móveis, atuação do robô dá-se na forma de locomoção no ambiente e com isso a sua autonomia passa por reconhecer ou mapear o ambiente em que está inserido.

Para um robô móvel autônomo realizar uma tarefa, muitas vezes é necessário ter conhecimento sobre o ambiente em que está inserido, pois através deste conhecimento o robô poderá tomar decisões e locomover-se de forma mais eficiente. Entretanto, nem sempre um mapa do ambiente está disponível, tornando necessário que haja um processo de mapeamento do local, seja esse realizado pelo próprio robô ou não.

Os processos de mapeamentos podem ser realizados de várias formas, um exemplo bem comum na robótica é o uso de técnicas de SLAM (do inglês *Simultaneous Localization and Mapping* - Mapeamento e Localização Simultâneos), em que o robô mapeia o ambiente ao mesmo tempo em que se localiza. Processos de mapeamentos realizados por diferentes atores têm sido cada vez mais utilizados, a fim de melhorar a velocidade de mapeamento e tornar o processo mais confiável. Dessa forma, caso um dos atores fique indisponível, os outros dão continuidade ao processo.

Para que processos de mapeamentos realizados por vários atores (podendo ser estes robôs) diferentes sejam viáveis, é necessário que haja também métodos eficientes de conexão (casamento) entre os mapas gerados. Esses métodos devem ser robustos, de forma que o mapa global gerado seja navegável pelo robô que irá utilizá-lo para navegação.

O processo de casamento de mapas é complexo, visto que comumente os processos de mapeamento realizados por atores individuais contém incertezas. Essas incertezas tornam o processo de casamento difícil, pois as regiões em comum, utilizadas para comparação entre os mapas parciais, não são totalmente simétricas. A dificuldade está em tentar relacionar ambientes apenas parcialmente simétricos, pois todas as incertezas encontradas nos mapas gerados com apenas um ator são multiplicadas quando se tenta associar os dados de dois ou mais destes atores.

Diversos fatores geram essas incertezas, dentre os principais é possível citar a imprecisão dos sensores (pois, no mundo real, os sensores recebem diversos ruídos em suas leituras) e a imprecisão na execução de controle por parte dos robôs (no mundo real, uma roda escorrega, um motor não gira exatamente o quanto ele foi programado para girar, o terreno é irregular e diversos outros fatores). Apesar das incertezas não serem o foco deste trabalho o pré-processamento nele utilizado é capaz de suavizá-las.

## 1.1 Objetivo

Este trabalho consiste em realizar o casamento de mapas bidimensionais de um mesmo ambiente. Os mapas podem ser obtidos por fontes diversas: por múltiplos robôs; por um mesmo robô em momentos diferentes; ou mesmo por outros agentes. Porém é importante que os mapeamentos sejam realizados individualmente, como por exemplo, com um robô de cada vez, para assim gerar um mapa final mais preciso, a partir dos mapas individuais. Os mapas são tratados como imagens para fim de busca de características que são melhores visualizadas, detectadas e tratadas com técnicas de visão computacional. Além disso, as poses iniciais de cada agente são consideradas desconhecidas.

## 1.2 Justificativa

Muitos trabalhos para casamento de mapas usam mapas gerados por robôs, via SLAM. Assim, esses trabalhos costumam concentrar seus esforços em extrair características diretamente do processo de mapeamento, relacionando o casamento diretamente ao próprio processo de geração de mapa. Desta forma, para portabilizar o método de casamento dos mapas para outros robôs, seria necessário também portabilizar o método de SLAM.

Dentro da pesquisa bibliográfica realizada para este trabalho, não foram encontrados métodos genéricos de casamento de mapas *occupancy grid* com poses inciais desconhecidas e com as características utilizadas para o casamento de mapas sendo extraídas dos mapas, sem necessitar de características extras a serem extraídas do ambiente durante o processo de mapeamento. Portanto, a contribuição deste trabalho está em gerar um método genérico que abranja as características citadas.

## 1.3 Trabalhos relacionados

Na literatura, existem diversas técnicas de casamento de mapas provenientes de SLAM através de vários robôs. Essas técnicas costumam se utilizar de marcações feitas durante o SLAM individual de cada robô da equipe, de forma que através dessas marcações seja possível encontrar diversos tipos de associações. No trabalho de Bresson *et al.* (2013) é realizado casamento de mapas provenientes de SLAM com múltiplos robôs simultâneos. Nesse trabalho, a cada vez que uma certa quantidade de marcações é feita pelos robôs, tenta-se fazer associações entre essas marcações através da distância entre elas, admitindo variações nas posições dessas marcações, desde que dentro de uma margem de erro aceitável para o problema. Um problema desse método é em relação ao quanto e onde (ponto de rotação do mapa) o mapa de cada robô está rotacionado em relação a outro.

No trabalho de Thrun e Liu (2005), também foi realizado casamento de mapas provenientes de SLAM com múltiplos robôs simultâneos. Nesse trabalho, a cada nova marcação encontrada durante o processo de SLAM, são selecionadas as três marcações mais próximas dentro de um raio limite, memorizando distâncias e ângulos relativos às marcações e salvando dentro de uma árvore de possibilidades. Marcações com configurações semelhantes entre os mapas são consideradas para o casamento desses mapas. Um problema desse método consiste em necessitar de uma grande quantidade de marcações e que as três marcações mais próximas a cada novo ponto sejam as mesmas para todos os pontos equivalentes, em cada mapa de cada robô da equipe.

No trabalho de Cunningham *et al.* (2012), de três em três marcações extraídas dos mapas provenientes do processo de SLAM, são gerados triângulos através da triangulação de Delaunay. Características como perímetro e área dos triângulos são dadas a um algoritmo RANSAC, que tenta encontrar o melhor casamento entre os mapas. O problema identificado neste método é que ele tende a falhar caso determinem-se muitas marcações divergentes entre os mapas de cada robô.

No trabalho de Koch *et al.* (2016), descreve-se uma abordagem bidimensional de SLAM com múltiplos robôs. A estratégia desse trabalho constrói uma representação dinâmica baseada em Funções de Distância Assinada. O mapa global é construído em paralelo, em vez de casar periodicamente os mapas individuais de cada robô e a

partir disso definir suas poses. Uma vantagem desse trabalho é não requerer detecção de fechamento de ciclo. Uma arquitetura *multi-thread* registra e a integra os dados em paralelo, estimando a pose de vários robôs simultaneamente. Um problema desse trabalho está em precisar de poses inciais conhecidas ou alguma estratégia de reconhecimento de outros robôs (durante os processo de mapeamento e localização), visto que o mapa não é casado e sim construído em conjunto.

No trabalho de Jafri e Chellali (2013), apresenta-se um sistema de SLAM com múltiplos robôs para o aprendizado de ambientes baseados em recursos. As informações do ambiente são medidas através de uma rede de sensores dinâmicos caracterizada por robôs com poses iniciais desconhecidas. Cada robô possui um *scanner* a *laser* 2D e uma *webcam*, servindo como um nó em movimento (da rede de sensores) para detectar as linhas horizontais e verticais, respectivamente. Esses nós em movimento são responsáveis por construir um espaço estruturado informacional. O sistema usa uma estrutura SLAM baseada em Filtro de Kalman Estendido (EKF) em cada robô, que eventualmente constrói um modelo tridimensional parcial baseando-se em recursos de linhas do ambiente. Cada nó compartilha seu modelo de mapa com outros nós que estão dentro de um intervalo de comunicação. Um mapa global é então gerado depois de obter uma estimativa das poses dos robôs, através da combinação de características mútuas de mapas individuais, além de obter confirmação visual de outros robôs.

No trabalho de Jian *et al.* (2017), um algoritmo de extração de características ORB, para casamento de mapas *occupancy grid*, foi melhorado. Características ORB têm boa rotação e invariância de escala, enquanto mantém uma boa velocidade de cálculo para SLAM, em tempo real, em grande número de aplicativos. Baseado-se no *Robot Operating System* (ROS), esse trabalho utilizou dois robôs omnidirecionais equipados com sensores *laser* 2D RP LIDAR para construir mapas através de Hector SLAM. O casamento dos mapas é integrado como um problema de registro de imagem, extraindo as características ORB do mapa e usando um método de busca tipo força bruta para combinar, entre as melhores correspondências, o cálculo da transformação afim e o casamento.

No trabalho de Blanco *et al.* (2007), propõe-se um método de casamento de mapas *occupancy grid* sendo tratados como imagens. A abordagem utilizada é baseada em extração de descritores de características, o que é feito através de uma transformação

de coordenadas polares em torno de pontos altamente distintos. Esse trabalho encontra correspondências entre os mapas de forma confiável e independente da orientação entre eles.

No trabalho de Blanco *et al.* (2013), apresenta-se uma abordagem de casamento de mapas *occupancy grid*. Esse método utiliza uma metodologia de localização de correspondências entre conjuntos de características esparsas detectadas no mapa. Neste trabalho os mapas são tratados como imagens genéricas. Para lidar com as incertezas e ambiguidades provenientes dos mapas *occupancy grid*, utilizou-se um RANSAC modificado que procura um valor dinâmico de subconjuntos internamente consistentes, de forma a emparelhá-los para calcular hipóteses sobre a translação e rotação entre os mapas. Em suma, esse método fornece uma distribuição de probabilidade da posição relativa entre os mapas e pode ser integrado a estruturas de mapeamentos em grande escala para robôs móveis.

No trabalho de Birk e Carpin (2006), propõe-se um método que identifica regiões de sobreposição entre os mapas. Esse método utiliza uma métrica de similaridade heurística e um algoritmo de busca estocástica. Assim, utilizam-se dois mapas: m e m'; o algoritmo de busca transforma m' com rotações e translações para encontrar a sobreposição máxima entre m e m'. Durante este processo, a métrica de similaridade heurística guia o algoritmo de pesquisa para encontrar soluções ótimas.

No trabalho de Zhang *et al.* (1995), faz-se a correlação entre imagens não calibradas (a calibração das câmeras são desconhecidas) para casamento de imagem em uma cena única. Nesse trabalho utilizam-se relações geométricas (entre os cantos extraídos das imagens), chamadas de geometria epipolar, para procurar correspondências[1] que correlacionem as imagens. Assim, junto aos métodos de correlação e relaxamento[2] e, por fim utilizando a técnica de Mínima Média Quadrática, são então descartadas correspondências falsas entre os conjuntos de características detectadas. Dessa forma, reduziu-se a alta complexidade que seria a realização de uma busca exaustiva por correspondências entre os conjuntos de geometrias epipolares das imagens.

[1] Nesse trabalho foi utilizado uma estratégia própria para atualização de correspondências, que seleciona apenas as características com alta correspondências e baixa ambiguidade.

[2] Nesse mesmo trabalho foi utilizado uma métrica própria de suporte de correspondência, de forma que se obteve uma maior tolerância nas deformações das transformações rígidas no plano da imagem e menor impacto das correspondências distantes em relação as próximas

## 1.4 Divisão do trabalho

Além do presente capítulo, este trabalho está dividido da seguinte forma: no Capítulo 2, tem-se a fundamentação teórica, em que se aborda a lógica e o conhecimento base para SLAM com um e com múltiplos robôs. No Capítulo 3, detalha-se a proposta deste trabalho, informando o processo de extração de características do mapa, como são realizados os relacionamentos destas características e como, através destes relacionamentos, são realizadas as comparações necessárias para o casamento entre os mapas. No Capítulo 4, analisam-se os resultados de forma qualitativa, mostrando que as análises foram feitas em relação aos casamentos em que se obtiveram sucesso e também nos casos de falha. No Capítulo 5, têm-se as conclusões apresentadas sobre esta pesquisa.

# 2 Fundamentação teórica

Neste capítulo são abordados os conteúdos utilizados no desenvolvimento deste trabalho, sendo estes: técnicas pré-processamento de imagens, métodos de detecção de cantos, métrica de similaridade de Jaccard e vetores.

Como pré-processamento de imagens, são abordadas as técnicas: limiarização; dilatação; erosão; e filtros de suavização, com foco em filtro da mediana. Em detecção de cantos, discute-se sobre detectores de canto, concentrando-se porém nas técnicas *Corner Harris* e *Good Feature to Track*. Na métrica de similaridade de Jaccard, é definida sua aplicação. Em vetores, define-se um vetor e algumas de suas características, além disso, explana-se sobre como calcular o tamanho de um vetor e o ângulo entre vetores.

## 2.1 Pré-processamento de imagens

Para poder trabalhar com imagens, muitas vezes é necessário que se utilize primeiro um pré-processamento, pois é muito comum que existam ruídos nessas imagens. Esses ruídos podem provocar falhas na detenção de regiões de interesse da imagem, portanto devem ser tratados. O pré-processamento de imagens também é muito utilizado para realçar características específicas de uma imagem, tornando-as mais evidentes ao detector utilizado. Em suma, pré-processamento de imagens é utilizado para melhorar a qualidade da imagem dada, de forma a proporcionar melhores resultados ao processamento dessas imagens.

Existem diversos pré-processamentos na literatura, mas são descritos apenas os métodos utilizados neste trabalho.

### 2.1.1 Limiarização

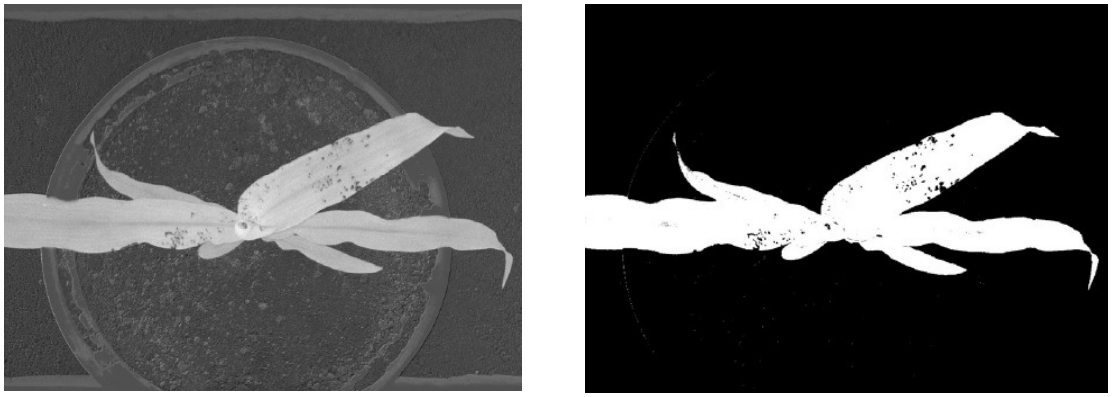
Segundo Lopes (2012), a limiarização é uma técnica de segmentação que se baseia na análise de similaridade dos níveis de cinza de uma imagem. Dessa forma, essa técnica é utilizada para extrair objetos de interesse da imagem. Funcionando mediante a definição de um limiar que separe os agrupamentos de níveis de cinza da imagem.

Este processo de segmentação é realizado percorrendo a imagem, pixel por pixel, associando cada pixel como parte do objeto de interesse ou como parte do fundo. Essa

operação é realizada comparando os valores dos pixels com o limiar. Para exemplificar o resultado desta técnica, observe a a Figura 1.

Figura 1 – Segmentação de uma planta de milho através de limiarização.

(a) Imagem da planta em escala de cinza. (b) Resultado da segmentação da planta.



Fonte: (JUNIOR *et al.*, 2003)

Conforme se observa na Figura 1, para a imagem que representa a planta em escala de cinza, é estabelecido um valor limiar em escala de cinza. Este limiar significa que qualquer pixel com valor (em escala de cinza), acima deste limiar, terá seu valor aumentado ao maior valor da escala (255, cor preta) e qualquer pixel que tenha um valor abaixo do limiar, terá seu valor reduzido ao menor valor (zero, cor branca). Desta forma, destaca-se o objeto de interesse na imagem, em que o projetista deverá definir como sendo relacionado aos pixels com valores acima do limiar ou aos pixels de valor abaixo deste limiar.

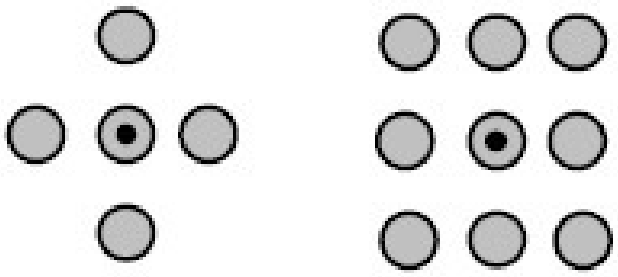
### 2.1.2 Dilatação

A dilatação é um operador morfológico aplicado em processamento digital de imagens, em situações como restauração e segmentação de imagens (MEDEIROS *et al.*, 2002).

A dilatação utiliza elementos estruturantes. Estes elementos são caracterizados como conjuntos definidos e conhecidos (tanto em forma, como em tamanho). Exemplos de elementos estruturantes podem ser observados na Figura 2. Estes conjuntos são comparados ao conjunto desconhecido da imagem.

Figura 2 – Elementos estruturantes.

(a) Forma de cruz. (b) Forma de quadrado.



Fonte: (MEDEIROS *et al.*, 2002)

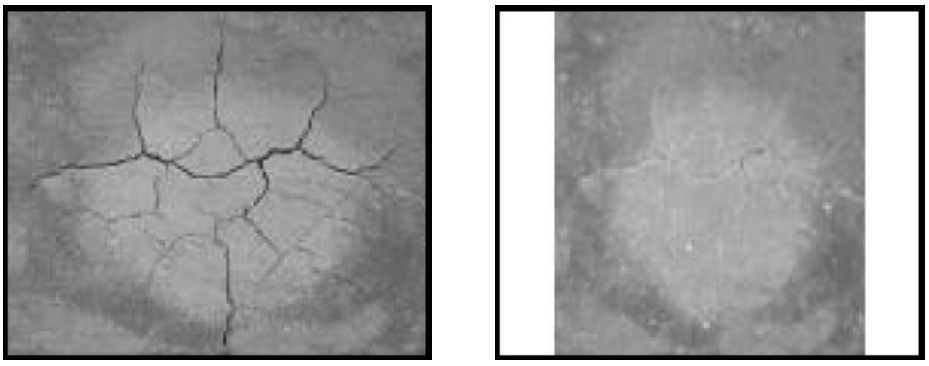
No trabalho de Medeiros *et al.* (2002): a dilatação de uma imagem (escala de cinza) através de um elemento estruturante é dada pela Equação 2.1.

$$[\delta_B(f)](x) = \max_{b \in B} f(x + b) \tag{2.1}$$

O resultado de como funciona o processo de dilatação em uma imagem em escala de cinza, pode ser observado na Figura 3, em que a imagem dilatada (utilizando um elemento estruturante 3x3) apresenta um alargamento nas regiões de cores claras (MEDEIROS *et al.*, 2002).

Figura 3 – Exemplo de dilatação.

(a) Imagem original. (b) Imagem dilatada.



Fonte: (MEDEIROS *et al.*, 2002)

Quando aplicado em imagens binárias, a dilatação "aumenta"ou "engrossa"os objetos na imagem (GONZALEZ; WOODS, 2010).

### 2.1.3 Erosão

A erosão, assim como a dilatação, é um operador morfológico que também é aplicado em processamento digital de imagens, da mesma forma, também necessita de elementos estruturantes.

Enquanto a dilatação "engrossa"os objetos, a erosão "afina"os objetos em imagens binárias (GONZALEZ; WOODS, 2010). No trabalho de Medeiros et al. (2002): a erosão de uma imagem (escala de cinza), através de um elemento estruturante, é dada pela Equação 2.2

$$[\varepsilon_B(f)](x) = \min_{b \in B} \{ f(x + b) - B(b) \} \tag{2.2}$$

O resultado de como funciona o processo de erosão em uma imagem, em escala de cinza, pode ser observado na Figura 4, em que a imagem erodida (utilizando um elemento estruturante 3x3) apresenta um alargamento nas regiões de cores escuras (MEDEIROS *et al.*, 2002).

Figura 4 – Exemplo de erosão.

(a) Imagem original. (b) Imagem erodida.

Fonte: (MEDEIROS *et al.*, 2002)

### 2.1.4 Filtros de suavização

Segundo Gonzalez e Woods (2010): filtros de suavização são usados para borrar e reduzir ruídos. O borramento é aplicado em pré-processamento de imagens, como uma forma de reduzir pequenos detalhes dessas imagens. A redução de ruído é realizada através de borramento utilizando um filtro linear ou por filtragem não linear.

#### 2.1.4.1 Filtro de mediana

Filtro de mediana é um tipo de filtro espacial não linear, que se baseia na ordenação de pixels localizados na área abrangida pelo filtro, desta forma, o pixel central é substituído pela mediana dos valores de intensidade de sua vizinhança (o valor do pixel é incluído no cálculo) (GONZALEZ; WOODS, 2010).

Os filtros de mediana são eficazes para redução de ruídos aleatórios, como nos casos de "ruído sal e pimenta"(caracterizados pela sua aparência, pontos brancos e pretos aleatoriamente sobrepostas em uma imagem) (GONZALEZ; WOODS, 2010).

O filtro da mediana pode ser definido da seguinte forma (SOUZA, 2018): $S_{xy}$ sendo um conjunto em uma janela retangular $m\ x\ n$ e com centro no ponto $(x,y)$, de forma que o filtro calcula a mediana da imagem $g(x,y)$, na área definida por $S_{xy}$. Assim, qualquer ponto $(x,y)$ da imagem $f$ obtida, é a mediana dos pixels na região abrangida por $S_{xy}$, conforme observa-se na Equação 2.3.

$$f(x,y) = \underset{(s,t) \in S_{xy}}{mediana}\{g(s,t)\} \tag{2.3}$$

A Figura 5 exemplifica o resultado da aplicação do filtro da mediana (tamanho 3$x$3) em uma imagem com 5% de ruído do tipo sal e pimenta.

Figura 5 – Exemplo de aplicação do filtro da mediana.

(a) Imagem original. (b) Imagem com ruído. (c) Resultado do filtro.

Fonte: (FARIA, 2005)

## 2.2 Detecção de cantos

O uso de detectores de cantos para encontrar pontos de interesse, sendo estes correspondentes entre várias imagens ou entre imagens e objetos reais, é um passo essencial para diversas aplicações utilizando processamento de imagens e/ou visão computacional (LIU *et al.*, 2009). Algumas das possíveis aplicações (LIU *et al.*, 2009) incluem correspondências entre imagens, reconhecimento de gestos, detecção de boca, detecção de canto de olho e rastreamento de movimento e navegação por robôs.

Um canto é um ponto determinado por duas bordas dominantes e com direções diferentes. Assim, um canto pode ser definido pela intersecção entre duas arestas. Em relação a imagens, uma aresta pode ser determinada por uma alteração brusca de brilho na imagem (PATTAR, 2015).

Detecção de cantos é normalmente chamada de detecção de pontos de interesse, sendo esta uma metodologia utilizada para extrair determinados tipos de características de uma imagem (PATTAR, 2015).

Uma grande quantidade de metodologias de detecção de pontos de interesse foram desenvolvidas nas últimas duas décadas. Algumas delas encontram pontos de alta simetria local, outras encontram áreas de textura com muita variação, existem ainda

aquelas que localizam pontos de esquina (LIU *et al.*, 2009). Estudos comparativos entre métodos de detecção de cantos podem ser encontrados nos trabalhos de Pattar (2015), Liu et al. (2009) e de Dey et al. (2012).

### 2.2.1 *Corner harris*

*Corner Harris* (HARRIS; STEPHENS, 1988) é um popular detector de cantos com forte invariância a rotação, escala, variação de iluminação e ruído de imagem (PATTAR, 2015). Este método baseia-se na função de auto-correlação local de um sinal, de forma que esta função mede as alterações locais do sinal com *patches* deslocados uma pequena quantidade em direções diferentes (DEY *et al.*, 2012).

Dados um deslocamento $(\Delta x, \Delta y)$ e um ponto $(x, y)$, a função de auto-correlação é definida de acordo com a Equação 2.4 (DEY *et al.*, 2012).

$$c_{x,y} \;=\; \Sigma_W [I(x_i, y_i) \;-\; I(x_i \;+\; \Delta x, y_i \;+\; \Delta y)]^2 \tag{2.4}$$

$I(x_i, y_i)$ representa a função imagem e $(x_i, y_i)$ são os pontos na janela $W$ centrada em $(x, y)$. A imagem deslocada é aproximada por uma expansão de Taylor truncada para os termos de primeira ordem, conforme observa-se na Equação 2.5 (DEY *et al.*, 2012).

$$I(x_i \;+\; \Delta x, y_i \;+\; \Delta y) \;\approx\; [I(x_i, y_i) \;+\; [I_x(x_i, y_i)]\; I_y(x_i, y_i)]] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{2.5}$$

$I_x(x_i, y_i)$ e $I_y(x_i, y_i)$ são as derivadas parciais em relação a $x$ e $y$, respectivamente. Com um filtro como $[-1, 0, 1]$ e $[-1, 0, 1]^T$, as derivadas parciais da imagem podem ser calculadas substituindo a Equação 2.5 na Equação 2.4. Assim, obtem-se a Equação 2.6 (DEY *et al.*, 2012).

$$\begin{aligned} c(x, y) \;&=\; [\Delta x, \Delta y] \begin{bmatrix} \sum_W (I_x(x_i, y_i))^2 & \sum_W (I_x(x_i, y_i) I_y(x_i, y_i)) \\ \sum_W (I_x(x_i, y_i) I_y(x_i, y_i)) & \sum_W (I_y(x_i, y_i))^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\ &=\; [\Delta x, \Delta y] C(x, y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \end{aligned} \tag{2.6}$$

$C(x, y)$ é a matriz de auto-correlação que captura a estrutura de intensidade da vizinhança local. Sejam $\alpha_1$ e $\alpha_w$ os autovalores de $C(x, y)$, então têm-se três casos a considerar (PATTAR, 2015):

- Ambos autovalores têm médias baixas e são regiões uniformes (intensidade constante).
- Um autovalor tem média alta, o outro tem média baixa e é um contorno (borda).
- Ambos autovalores têm médias altas e são pontos de interesse (canto).

Um exemplo da aplicação do método de detecção de cantos *Corner Harris* pode ser observado na Figura 6, em que os quadrados brancos na imagem da direita (b) são os cantos que foram detectados pelo método.

Figura 6 – Exemplo de detecção de cantos utilizando Corner Harris.

(a) Imagem original. (b) Imagem com cantos detectados.


Fonte: (PATTAR, 2015)

### 2.2.2 *Good feature to track*

*Good Feature to Track*[1] (SHI; TOMASI, 1994) é um método que busca em uma imagem os melhores cantos para serem utilizados em *tracking* de objetos, de forma que esses são utilizados para detectar e mensurar movimentos de objeto ou da câmera na imagem.

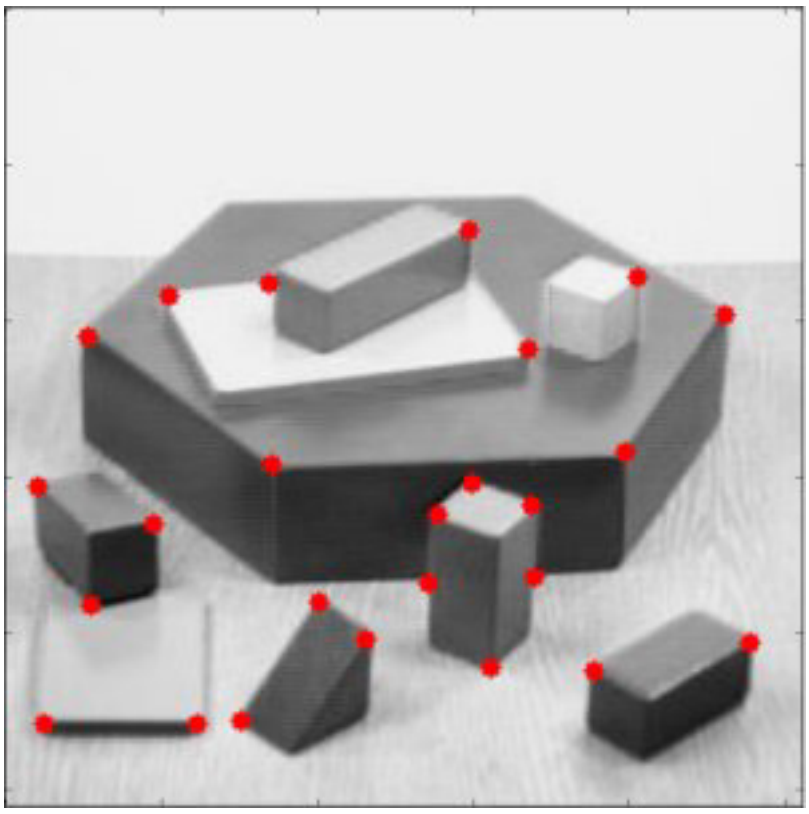
Este método é uma extensão do *Corner Harris* (HARRIS; STEPHENS, 1988). Ele busca os $N$ melhores cantos da imagem, os cantos são detectados utilizando um método de detecção de cantos (como o Corner Harris). A imagem precisa estar em escala de cinza e então o projetista precisa especificar o número máximo de cantos a ser selecionado (a partir dos cantos encontrados pelo método de detecção e cantos utilizado).

[1] Este método pode ser encontrado em OPENCV.

A seleção dos melhores cantos é baseada principalmente no número máximo $N$ de cantos e na qualidade mínima dos cantos, que também é definida pelo projetista e essa varia de 0 a 1. Os cantos, cuja qualidade mensurada for inferior a qualidade mínima definida, são simplesmente descartados. Além disso, é estabelecido também uma distância mínima entre os cantos.

Em suma, os cantos são ranqueados de acordo com a qualidade mensurada. Dentre esses, os que estão acima da qualidade mínima e que encontram-se à uma distância superior à distância mínima estabelecida, são escolhidos os melhores cantos. De forma que o número de cantos não ultrapasse o número máximo $N$ estabelecido. Um exemplo do resultado deste método pode ser observado na Figura 7, em que os pontos vermelhos representam os melhores cantos escolhidos pelo método.

Figura 7 – Exemplo de cantos selecionados.



Fonte: (OPENCV, 201?)

## 2.3 Métrica de similaridade de Jaccard

Também conhecida como coeficiente de Jaccard, a métrica de similaridade de Jaccard (JACCARD, 1912) mede o quão similar são duas imagens ou trechos segmentados de imagens (MILSZTAJN; GEUS, 2015). Essa métrica é calculada de acordo com a Equação 2.7 (MILSZTAJN; GEUS, 2015), em que $A_1$ representa uma imagem e $A_2$ representa uma segunda imagem, o resultado da equação define a semelhança entre $A_1$ e $A_2$.

$$Jaccard = \frac{|A_1 \cap A_2|}{|A_1 \cup A_2|} \tag{2.7}$$

## 2.4 Vetores

Segundo Venturi (2015), um vetor é definido por uma tripla composta de uma direção, um sentido e um número não negativo. Um vetor pode ser representado por uma letra latina minúscula com um seta em cima, exemplo: $\vec{v}$.

A soma de um ponto $A$ com um vetor $\vec{v}$ é um ponto $B$ (VENTURI, 2015), conforme observa-se na Equação 2.8 (VENTURI, 2015).

$$A + \vec{v} = B \tag{2.8}$$

Isolando o termo $\vec{v}$ na Equação 2.8, tem-se então a Equação 2.9 (VENTURI, 2015), que significa que um vetor é definido por dois pontos, em que $A$ é origem e $B$ a extremidade do vetor.

$$\vec{v} = B - A \tag{2.9}$$

Então, considerando um plano $\alpha$ e os pontos $A = (x_A, y_A)$ e $B = (x_B, y_B)$ pertencentes a este plano, um vetor $\vec{v}$ definido por $A$ (origem) e $B$ (extremidade), é expresso da forma $\vec{v} = (x_B - x_A, y_B - y_A)$.

### 2.4.1 Tamanho de vetores

Um vetor $\vec{v}$ definido pelos pontos $A$ (origem) e $B$ (extremidade) tem seu tamanho expresso pela notação $|\vec{v}|$ e essa é definida pela Equação 2.10 (VENTURI, 2015).

$$|\vec{v}| = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2} \tag{2.10}$$

### 2.4.2 Ângulo entre vetores

Segundo Venturi (2015), o ângulo entre dois vetores $\vec{u}$ e $\vec{v}$ é defino pela Equação 2.11 (VENTURI, 2015).

$$\cos\theta = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}|\,|\vec{v}|} \tag{2.11}$$

A expressão $\vec{u} \cdot \vec{v}$ é o produto interno (escalar) dos vetores $\vec{u} = (x_{\vec{u}}, y_{\vec{u}})$ e $\vec{v} = (x_{\vec{v}}, y_{\vec{v}})$, em que esse produto interno é defino pela Equação 2.12 (VENTURI, 2015).

$$\vec{u} \cdot \vec{v} = x_{\vec{u}} x_{\vec{v}} + y_{\vec{u}} y_{\vec{v}} \tag{2.12}$$

# 3 Casamento de mapas através de relacionamento de características

Este trabalho realiza o casamento de mapas provenientes de vários robôs. O casamento entre os mapeamentos fornecidos são realizados de dois em dois mapas, gerando assim um mapa global.

Os mapas fornecidos para a método são do tipo *occupancy grid* (ELFES, 1989), que é caracterizado por uma matriz de ocupação e quando tratado em forma de imagem é representado em escala de cinza, onde a cor preta representa colisões detectadas no ambiente, a cor branca representa regiões mapeadas e sem colisões, e a cor cinza representa regiões não mapeadas.

O formato de imagem utilizado no método foi o *Portable Graymap* (*pgm*) pois, por tratar-se de um formato portável, possui dimensões reduzidas e ao mesmo tempo uma boa relação de conversão de escala para com o ambiente original sendo mapeado. Este formato de imagem é importante pois reduz a complexidade da busca de semelhanças e características entre os mapas em comparação com formatos mais populares como *bitmap* e *jpeg*.

## 3.1 Casamento de mapas

Para realizar o casamento entre os mapas individuais dos robôs da equipe e assim gerar um mapa global, é necessário seguir uma sequência de etapas.

Como o método proposto é baseado no casamento de mapas através do processo de mapeamento, ele funciona de forma que o sistema receba os mapas de cada novo estado fornecido ou salvo durante o processo. No Algoritmo 1, pode-se observar o procedimento, em que o sistema recebe os estados de cada mapeamento e fornece-os para a função de casamento de mapas. Esta função retorna o casamento resultante e a taxa de casamento, que define o quão similar são os estados dos mapas (em relação aos pontos de casamento selecionados). Após esta etapa, é verificado se a taxa de casamento é maior que a melhor taxa de casamento até o momento (a melhor taxa é inicialmente zero), se a taxa de casamento for superior, então a melhor taxa de casamento é atualizada

juntamente com o melhor casamento.

**Algoritmo 1:** Casamento dos Mapas dos Processos de Mapeamento

```
1:  procedure CASAMENTOMAPASMAPEAMENTO(mapeamento_1, mapeamento_2)
2:      taxaCasamento = 0
3:      melhorTaxaCasamento = 0
4:      map_1 = vazio
5:      map_2 = vazio
6:      melhorCasamento = vazio
7:      Enquanto existir mapas em mapeamento_1 ou mapeamento_2
8:          se (existir mapas em mapeamento_1)
9:              map_1 = Próximo mapa do mapeamento_1
10:         fim se
11:         se (existir mapas em mapeamento_2)
12:             map_2 = Próximo mapa do mapeamento_2
13:         fim se
14:         casamento, taxaCasamento = CasamentoMapas(map_1, map_2)
15:         se (taxaCasamento > melhorTaxaCasamento) então
16:             melhorCasamento = casamento
17:             melhorTaxaCasamento = taxaCasamento
18:         fim se
19:     fim enquanto
20:     Retornar melhorCasamento
21: end procedure
```

O processo de casamento pode ser observado na função exposta no Algoritmo 2, em que o procedimento recebe dois mapas como entrada. Assim, é realizada uma sequência de procedimentos para a obtenção do casamento. Inicialmente, realiza-se o pré-processamento dos mapas, de forma que os ruídos mais evidentes são suavizados e as características principais dos mapas são ressaltadas. Após o pré-processamento, extraem-se as características dos mapas e então, a partir de relacionamentos geométricos entre elas, são gerados os alfabetos. Através da comparação entre as semelhanças dos membros dos alfabetos, define-se um conjunto de pontos que são considerados candidatos a pontos de casamento entre os mapas (pontos de encaixe dos mapas). A partir desses

pontos de casamento, são definidos os melhores pontos baseado em uma métrica de similaridade e então os casamentos gerados pelos pontos com maior similaridade são retornados com taxa de similaridade associada.

**Algoritmo 2:** Casamento de Mapas

1: **procedure** CASAMENTOMAPAS($map\_1, map\_2$)
2: $map\_cinza\_1 =$ **PreProcessar**($map\_1$)
3: $map\_cinza\_2 =$ **PreProcessar**($map\_2$)
4: $caracteristicas\_1 =$ **ExtrairCaracteristicas**($map\_cinza\_1$)
5: $caracteristicas\_2 =$ **ExtrairCaracteristicas**($map\_cinza\_2$)
6: $alfabeto\_1 =$ **GerarAlfabeto**($caracteristicas\_1$)
7: $alfabeto\_2 =$ **GerarAlfabeto**($caracteristicas\_2$)
8: $alfabetosComparados =$ **CompararAlfabetos**($alfabeto\_1, alfabeto\_2$)
9: $pontosCandidatos =$ **CandidatosPontosCasamento**($alfabetosComparados$)
10: $melhoresPontos, maiorJaccard =$ **MelhoresPontosCasamento**($pontosCandidatos, map\_cinza\_1, map\_cinza\_2$)
11: $casamento =$ **Casamento**($melhoresPontos, map\_1, map\_2$)
12: **Retornar** $casamento$, $melhorJaccard$
13: **end procedure**

As etapas dispostas no Algoritmo 2 são melhores detalhas nos tópicos a seguir.

### 3.1.1 Pré-processamento dos mapas

O pré-processamento dos mapas é uma etapa fundamental, principalmente para suavizar os ruídos (provenientes do sistema utilizado para mapear os ambientes) e para ressaltar as características dos mapas (deixar as colisões mais evidentes).

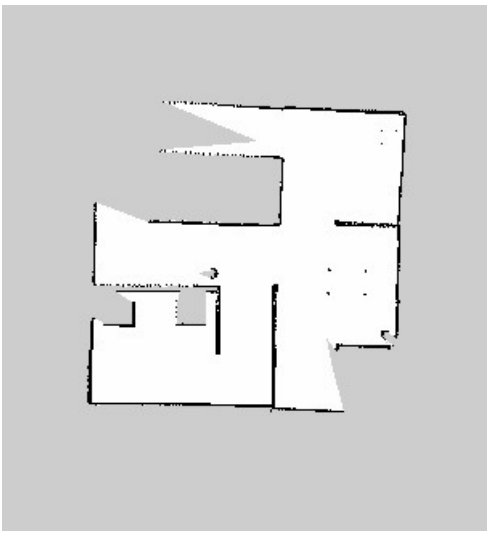
O Algoritmo 3 detalha bem as etapas necessárias para obter-se um bom nível de detalhamento do mapa. O mapa é inicialmente convertido para escala de cinza, o que é necessário para que posteriormente ele seja binarizado, de forma que apenas as cores branca e preta permaneçam no mapa. A exclusão da cor cinza é importante para evitar a detecção de cantos entre as cores pretas e cinza ou entre as cores cinza e branca, portanto, esta é convertida para branca, conforme observa-se na Figura 8, em que a letra (a) representa o mapa fornecido ao sistema, e a letra (b) representa o mapa após ser binarizado.

**Algoritmo 3:** Pré-processamento do mapa

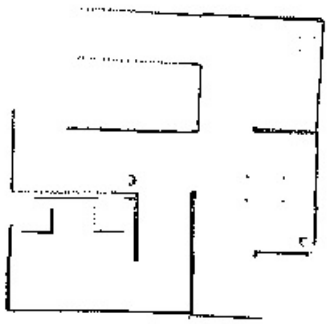
```
1:  procedure PREPROCESSAR(map)
2:      map_cinza = converterEscalaCinza(map)
3:      map_cinza = binarizar(map_cinza)
4:      map_cinza = Erodir(map_cinza)
5:      map_cinza = Erodir(map_cinza)
6:      map_cinza = Dilatar(map_cinza)
7:      map_cinza = Dilatar(map_cinza)
8:      map_cinza = Erodir(map_cinza)
9:      map_cinza = FiltroMediana(map_cinza)
10:     map_cinza = Dilatar(map_cinza)
11:     retornar map_cinza
12: end procedure
```

Figura 8 – Exemplo de binarização de um mapa.

(a) Mapa fornecido.



(b) Mapa binarizado.

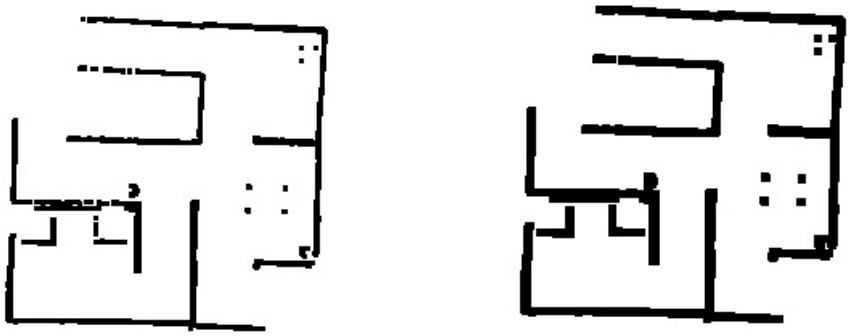


Nos mapas *occupancy grid*, uma parede é determinada por uma nuvem de pontos detectados como forma de colisão. Esta característica pode ocasionar alguns problemas, como a descontinuidade da parede, gerando pequenos espaçamentos. Esses problemas podem confundir o detector de cantos utilizado neste trabalho, provocando a detecção de cantos em paredes contínuas, portanto aplica-se uma sequência de duas erosões, para

"engrossar"as colisões do mapa e então unir os pontos pertencentes à parede, de forma a promover as continuidades destas. O resultado da aplicação de duas erosões na letra (a) da Figura 8 pode ser observado na Figura 9, em que verifica-se que as erosões promovem o fechamento dos espaçamentos indevidos nas paredes do mapa.

Figura 9 – Exemplo de erosão aplicado a um mapa binarizado.

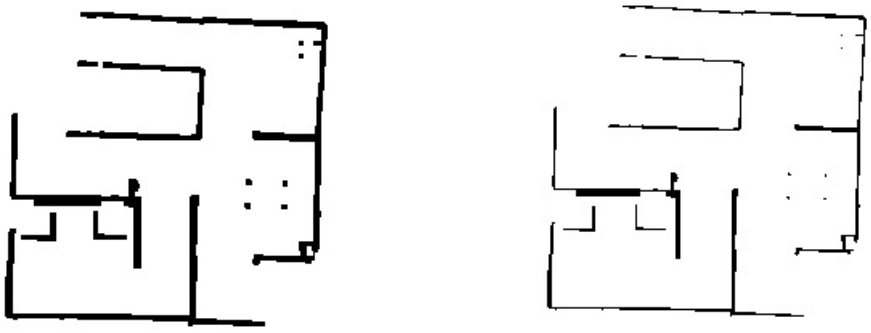
(a) Primeira erosão. (b) Segunda erosão.



Após o fechamento dos espaçamentos indevidos nas paredes do mapa, é necessário então voltar as paredes às suas espessuras originais, portanto aplica-se uma sequência de duas dilatações, para "afinar"as paredes do mapa. O resultado das dilatações podem ser observadas na Figura 10 e, ao comparar o resultado da letra (b) dessa figura com o resultado da letra (b) da Figura 8, pode-se observar que as descontinuidades foram corrigidas.

Figura 10 – Exemplo de dilatação aplicado a um mapa erodido.

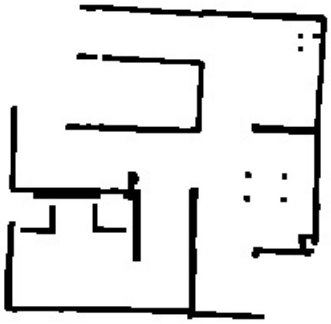
(a) Primeira dilatação. (b) Segunda dilatação.



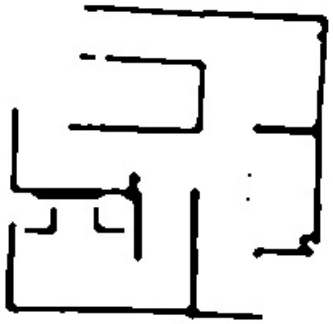
Nota-se que alguns objetos (pernas de cadeiras ou mesas, etc) nos mapeamentos podem provocar marcações que não são interessantes para o casamento, pois estas podem causar associações erradas no método. Essas marcações podem ser removidas através de três operações em sequência que são a erosão, o filtro da mediana e, por fim, uma dilatação. A erosão nesta sequência de operações serve para que quando o filtro da mediana for aplicado, as paredes finas do mapeamento não sejam excluídas, o filtro da mediana será responsável por excluir as marcações indesejadas e, ao final, a dilatação deverá retornar a espessura das paredes ao original, afinando dessa forma o que sobrou das marcações que continuaram na imagem, assim, essas marcações são excluídas. O resultado destas operações aplicadas à imagem da letra (b), da Figura 10, podem ser observadas na letra (c) da Figura 11, onde havia pequenas marcações que poderiam provocar extrações de características prejudiciais, e desta forma gerariam associações erradas entre elas, aumentando a complexidade da busca de semelhança entre os mapas.

Figura 11 – Exemplo de uma sequência de erosão, filtro da mediana e dilatação.

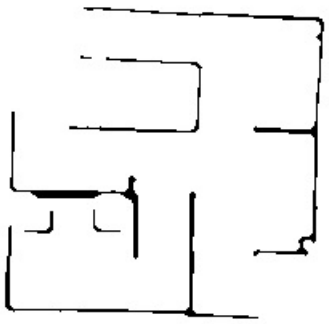
(a) Erosão.



(b) Filtro da mediana.



(c) Dilatação.



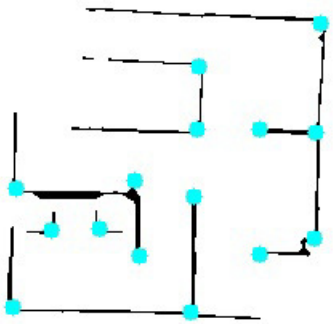
### 3.1.2 Extração de características

Em razão de os mapas SLAM serem, em sua essência, pobres de características, escolheu-se uma das característica mais evidentes nos mapas, que são os cantos. Como detector de cantos, decidiu-se usar a técnica *Good features to track* (SHI; TOMASI, 1994), pois essa é uma extensão no método de detecção de cantos *Harris corner* (HARRIS;

STEPHENS, 1988) que, além de ser um eficiente método de detecção de cantos, também possui uma forte invariância de rotação e ruído. A vantagem de utilizar o técnica *Good features to track* está em sua principal característica, que é selecionar cantos de forma distribuída ao longo de toda a imagem e limitar o número máximo de cantos a serem detectados, provocando uma redução de complexidade para o processo de geração e comparação de alfabetos.

A técnica *Good features to track* é aplicada logo após o pré-processamento dos mapeamentos. Na Figura 12 é possível observar os cantos que foram detectados no mapa da letra (c), da Figura 11.

Figura 12 – Detecção de cantos utilizando a técnica *Good features to track*.



Com a detecção destes cantos, gera-se uma lista $L_m$ para cada mapa de cada robô com $N_m$ cantos detectados em cada mapa, em que $m$ representa o mapa, $N$ o número de cantos e $L$ representa a lista conforme se observa na expressão 3.1.

$$L_m = (c_{1,m}, c_{2,m}, c_{3,m}, ..., c_{N_m,m}) \tag{3.1}$$

No Algoritmo 4, pode-se observar a sequência de ações necessárias para extração das características após o pré-processamento dos mapas. Neste algoritmo, a lista de cantos

$L_m$ é representada pelo vetor "características"e cada item dentro deste vetor representa um item $c$, da lista $L_m$.

**Algoritmo 4:** Extração de Características

```
1: procedure EXTRAIRCARACTERISTICAS(map)
2:     N = número de cantos
3:     q = qualidade mínima dos cantos
4:     caracteristicas = vazio
5:     caracteristicas = goodFeaturestoTrack(N,q)
6:     retornar caracteristicas
7: end procedure
```

Em um caso onde haveria dois mapeamentos, considerando que, no primeiro mapa foram detectados 100 cantos e no segundo mapa foram detectados 80 cantos, seriam geradas duas listas de cantos conforme expostas na expressão 3.2.

$$\begin{aligned} L_1 &= (c_{1,1}, c_{2,1}, c_{3,1}, ..., c_{100,1}), \\ L_2 &= (c_{1,2}, c_{2,2}, c_{3,2}, ..., c_{80,2}) \end{aligned} \tag{3.2}$$

### 3.1.3 Geração do alfabeto

Neste trabalho, alfabeto é tratado como um conjunto de símbolos com características próprias. Cada símbolo é definido por um determinado canto e as características do símbolo são determinadas pelo conjunto de relacionamentos geométricos com outros pertencentes ao mesmo mapa. O algoritmo 5 demonstra a sequência de ações a serem tomadas.

**Algoritmo 5:** Geração de alfabeto

```
1: procedure GERARALFABETO(L_m)
2:     para todo c_{i,m} ∈ L_m faça
3:         L_{c_{i,m}} = RelacionarCantos(c_{i,m}, L_m)
4:         alfabeto = CalcularAngulos(L_{c_{i,m}})
5:     fim para todo
6:     retornar alfabeto
7: end procedure
```

Para que um canto $c_{i,m}$ se relacione com outro $c_{j,m}$, foi definido que a distância euclidiana $d$ entre eles esteja sujeita a seguinte condição: $\forall d_{i,j} \mid D_{max} \geq d_{i,j} \wedge D_{min} \leq d_{i,j} \wedge D_{max} \geq D_{min} \wedge i \neq j$. Sendo que $D_{max}$ e $D_{min}$ são os parâmetros que definem as distâncias máxima e mínima, respectivamente, de relacionamento entre os cantos.

**Algoritmo 6:** Seleção dos cantos que serão relacionados para formação dos membros do alfabeto

```
1:  procedure RELACIONARCANTOS(c_{i,m}, L_m)
2:      para todo c_{j,m} ∈ L_m | i ≠ j faça
3:          d_{i,j} = distância euclidiana de c_{i,m} para c_{j,m}
4:          se (d ≤ D_max ∧ d ≥ D_min) então
5:              relacionar d_{i,j} a c_{j,m}
6:              adicionar c_{j,m} em L_{c_{i,m}}
7:          fim se
8:      fim para todo
9:      retornar L_{c_{i,m}}
10: end procedure
```

A função **RelacionarCantos** do Algoritmo 5 é expressa no Algoritmo 6 e expõe como o relacionamento é dado em que, para cada canto $c_{i,m}$ dentro da lista de cantos $L_m$, calcula-se a distância desse canto para todos os outros $c_{j,m}$ da mesma lista, sendo $j \neq i$. Os cantos dentro da margem de distâncias definida por $D_{max}$ e $D_{min}$ são adicionados a uma lista $L_{c_{i,m}}$ relacionada ao ponto $c_{i,m}$, conforme observa-se na expressão 3.3, em que o valor $T_{L_{c_{i,m}}}$ representa o tamanho da lista $L_{c_{i,m}}$, ou seja, o número de outros cantos com os quais $c_{i,m}$ se relaciona.

$$L_{c_{i,m}} = (c_{1,m}, c_{2,m}, c_{3,m}, ..., c_{T_{L_{c_{i,m}}},m}) \tag{3.3}$$

Após selecionar com quem cada canto será relacionado através de um critério de distância, é preciso calcular a segunda característica de cada membro do alfabeto. Calcula-se o ângulo formado pelos vetores definidos pelos cantos $c_{i,m}$ para os todos os membros $c_{j,m}$ de sua lista $L_{c_{i,m}}$, conforme observa-se no Algoritmo 7. Neste algoritmo faz-se o cálculo de todas as combinações possíveis dentro de cada lista e assim obtém-se

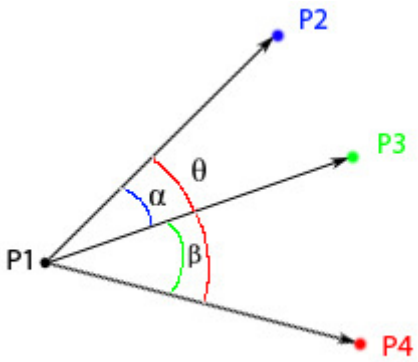
o alfabeto que será utilizado para comparação entre os mapas.

**Algoritmo 7:** Adição da característica ângulo nos membros do alfabeto

```
1:  procedure CALCULARANGULOS(c_{i,m}, L_{c_{i,m}})
2:      para todo c_{j,m} ∈ L_{c_{i,m}} | i ≠ j faça
3:          para todo c_{k,m} ∈ L_{c_{i,m}} | i ≠ k ∧ j ≠ k faça
4:              a_{i,j,k} = ângulos entre os vetores c_{i,m}c_{j,m} e c_{i,m}c_{k,m}
5:              relacionar a_{i,j,k} a c_{j,m} e c_{k,m}
6:          fim para todo
7:      fim para todo
8:      alfabeto = L_{c_{i,m}} com todos os relacionamentos de cada membro c_{j,m}
9:      retornar alfabeto
10: end procedure
```

Cada $c_{i,m}$ junto aos seus ângulos e distâncias calculadas em relação a todos os outros cantos dentro de um intervalo de distâncias delimitados por $D_{max}$ e $D_{min}$ são uma letra do alfabeto. Para melhor entender essa relação, observe a Figura 13, em que o ponto $P1$ representa cada canto $c_{i,m}$ e os pontos $P2$, $P3$ e $P4$ representam os outros cantos $c_{j,m}$ pertecentes a lista $L_{c_{i,m}}$ com os quais o canto $c_{i,m}$ se relaciona.

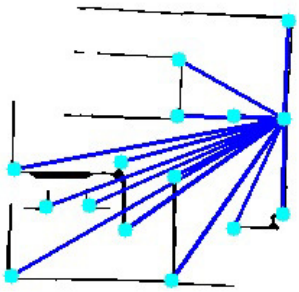
Figura 13 – Representação da construção do ângulo que compõe o alfabeto.



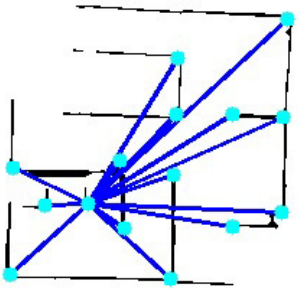

Na Figura 14, pode-se observar visualmente como fica o formato de 4 membros de um alfabeto gerado, estes membros foram gerados utilizando os cantos detectados na Figura 12.

Figura 14 – Membros de um alfabeto gerado.

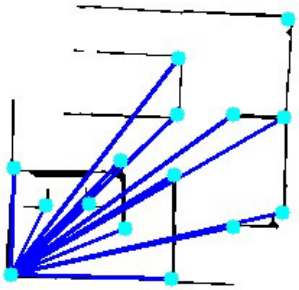
(a) Membro 1.



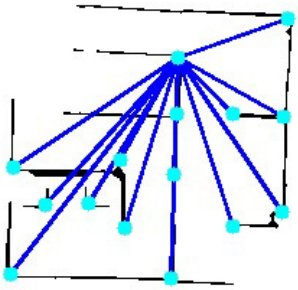
(b) Membro 2.



(c) Membro 3.

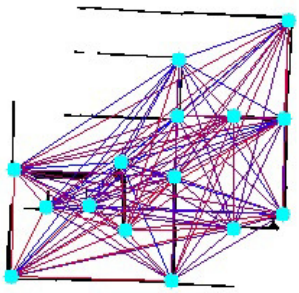


(d) Membro 4.



Expandindo essa formação para todo o contexto do mapa, levando em consideração todos os cantos, tem-se a formação de um grafo semelhante ao que pode ser observado da Figura 15. É possível observar que os cantos se relacionam apenas com outros cantos detectados dentro de um intervalo de distâncias mínima e máxima. O objetivo de delimitar essa distância é impedir que os erros acumulados com o distanciamento do canto interfiram no processo de casamento.

Figura 15 – Representação do alfabeto gerado em forma de grafo.



### 3.1.4 Comparação dos membros dos alfabetos

Nesta etapa, é realizada a comparação entre os membros dos alfabetos, em que realiza-se uma comparação baseada nas características de distâncias e ângulos relacionados a cada membro. Os cantos de cada mapa, possuem um membro de alfabeto associado.

Ao se observar o Algoritmo 8, pode-se verificar como se dá a comparação dos membros do alfabeto, com a finalidade de calcular as similaridade entre os cantos de cada mapa. Primeiramente compara-se cada distância $d_{i,j}$ e $d_{z,k}$ relacionada a cada alfabeto $L_{c_{i,1}}$ e $L_{c_{z,2}}$, se essas distâncias forem semelhantes (iguais ou dentro de uma margem de erro *erro_distancia*), procuram-se outras distâncias $d_{i,y}$ e $d_{z,q}$ também semelhantes e, se também forem encontradas distâncias semelhantes, procura-se relacionar os ângulos. Para o relacionamento entre os ângulos, verifica-se o ângulo formado entre as distâncias em cada mapa, ângulo $a_{i,j,y}$ para as distâncias $d_{i,j}$ e $d_{i,y}$ e o ângulo $a_{z,k,q}$ para as distâncias $d_{z,k}$ e $d_{z,q}$, se o ângulos $a_{i,j,y}$ e $a_{z,k,q}$ forem semelhantes (iguais ou dentro de uma margem de erro *erro_angulo*), então incrementa-se a matriz $alfabetosComparados[i][z]$ nas

posições referentes a cada membro do alfabeto $i$ e $z$.

**Algoritmo 8:** Comparação de similaridade entre os membros dos alfabetos

```
1: procedure COMPARARALFABETOS(alfabeto_1, alfabeto_2)
2:     para todo d_{i,j} ∈ L_{c_{i,1}} | i ≠ j faça
3:         para todo d_{z,k} ∈ L_{c_{z,2}} | z ≠ k faça
4:             se ((d_{i,j} ≤ d_{z,k} + erro_distancia) ∧ (d_{i,j} ≥ d_{z,k} − erro_distancia) ∧
   (d_{z,k} ≤ d_{i,j} + erro_distancia) ∧ (d_{z,k} ≥ d_{i,j} − erro_distancia)) então
5:                 para todo d_{i,y} ∈ L_{c_{i,1}} | i ≠ y land j ≠ y faça
6:                     para todo d_{z,q} ∈ L_{c_{z,2}} | z ≠ q land k ≠ q faça
7:                         se ((d_{i,y} ≤ d_{z,q} + erro_distancia) ∧
   (d_{i,y} ≥ d_{z,q} − erro_distancia) ∧ (d_{z,q} ≤ d_{i,y} + erro_distancia) ∧
   (d_{z,q} ≥ d_{i,y} − erro_distancia)) então
8:                             se ((a_{i,j,y} ≤ a_{z,k,q} + erro_angulo) ∧
   (a_{i,j,y} ≥ a_{z,k,q} − erro_angulo) ∧ (a_{z,k,q} ≤ a_{i,j,y} + erro_angulo) ∧
   (a_{z,k,q} ≥ a_{i,j,y} − erro_angulo)) então
9:                                 alfabetosComparados[i][z] += 1
10:                            fim se
11:                        fim se
12:                    fim para todo
13:                fim para todo
14:            fim se
15:        fim para todo
16:    fim para todo
17:    retornar alfabetosComparados
18: end procedure
```

### 3.1.5 Seleção dos candidatos a pontos de casamento

Após calcular a similaridade entre os cantos de cada mapa, deve-se selecionar os candidatos a pontos de casamento entre os mapas. Esta seleção é realizada através de uma busca de cantos com maiores similaridades na matriz $alfabetosComparados$.

Baseando-se no Algoritmo 9, estabelece-se uma quantidade máxima $numero_candidatos$ de candidatos a pontos de casamento $pontosCandidatos$. Para cada candidato $pontosCandidatos_{a,1}$,

associa-se uma posição de maior valor dentro de $alfabetosComparados$, de forma que as posições não se repetem entre os candidatos. Após definir o primeiro ponto de casamento para cada candidato, define-se o segundo ponto, em que $pontosCandidatos_{a,2}$ recebem a posição com maior valor dentro de $alfabetosComparados$ e que as distâncias associadas são semelhantes ($d_{i,j} \approx d_{z,k}$ são semelhantes, $i$ e $z$ são relacionados a $pontosCandidatos_{a,1}$ e $j$ e $k$ são relacionados a $pontosCandidatos_{a,2}$). Assim, seleciona-se os candidatos a pontos de casamentos a serem testados na próxima etapa.

**Algoritmo 9:** Seleção dos cantos candidatos a pontos de casamento

1: **procedure** CANDIDATOSPONTOSCASAMENTO($alfabetosComparados$)
2: **faça enquanto** ($a < numero\_candidatos$)
3: $pontosCandidatos_{a,1}$ = posição do $alfabetosComparados$ com maior valor e sem repetir posições para todos os $a$ em $pontosCandidatos_{a,1}$
4: $pontosCandidatos_{a,2}$ = posição do $alfabetosComparados$ com maior valor, sendo que $pontosCandidatos_{a,1} \neq pontosCandidatos_{a,2}$ e que $d_{i,j} \approx d_{z,k}$ sejam semelhantes, sendo $i$ e $z$ relacionados a $pontosCandidatos_{a,1}$ e $j$ e $k$ relacionados a $pontosCandidatos_{a,2}$
5: $a += 1$
6: **fim faça**
7: **retornar** $pontosCandidatos$
8: **end procedure**

### 3.1.6 Melhores pontos de casamento

Após a etapa de seleção de candidatos a pontos de casamento, é necessário escolher entre eles, aqueles que melhor se encaixam. Esta escolha é feita através de tentativas de encaixe entre os mapas, utilizando os pontos candidatos como pontos de encaixe.

Observando o Algoritmo 10, tem-se que, para cada conjunto $a$ de $pontosCandidatos$, os pontos (cantos) $c_{i,1}$ e $c_{z,2}$ referentes a $pontosCandidatos_{a,1}$, o mapa $map\_2$ a partir da posição do canto $c_{z,2}$ é transladado sobre a posição do canto $c_{i,1}$ pertencente ao mapa $map\_1$. Após se encaixar $c_{z,2}$ sobre $c_{i,1}$, faz-se necessário encaixar os pontos $c_{j,1}$ e $c_{k,2}$ referentes a $pontosCandidatos_{a,2}$, de forma que, a partir da posição $c_{i,1}$, rotaciona-se a o mapa $map\_2$ da posição do ponto $c_{k,2}$ sobre a posição do ponto $c_{j,1}$, realizando assim o

encaixe entre os pontos candidatos. A referência global do $map\_1$ continua a mesma e a do $map\_2$ é transladada e rotacionada na mesma proporção em que seus pontos foram transformados.

**Algoritmo 10:** Escolha dos melhores pontos de casamento

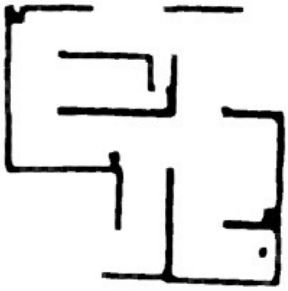
1: **procedure** MELHORESPONTOSCASAMENTO($pontosCandidatos, map\_1, map\_2$)
2: $melhorJaccard = 0$
3: **para todos** ($pontosCandidatos_{a,1}$) **faça**
4: $map\_2 = map\_2$ transladado da posição do ponto $c_{z,2}$ para posição do ponto $c_{i,1}$
5: $map\_2 = map\_2$ rotacionado em torno da posição do ponto $c_{i,1}$ pelo ângulo formado entre os vetores $\overrightarrow{c_{i,1}c_{j,1}}$ e $\overrightarrow{c_{i,1}c_{k,2}}$, sendo que $c_{j,1}$ e $c_{k,2}$ pertencem a $pontosCandidatos_{a,2}$
6: $inter =$ intersecção entre os mapas $map\_1$ e $map\_2$
7: $uni =$ união entre os mapas $map\_1$ e $map\_2$
8: $jaccard_a = \frac{inter}{uni}$
9: **se** ($jaccard_a > maiorJaccard$) **então**
10: $maiorJaccard = jaccard_a$
11: $melhoresPontos_1 = pontosCandidatos_{a,1}$
12: $melhoresPontos_2 = pontosCandidatos_{a,2}$
13: **fim se**
14: **fim para todos**
15: **retornar** $melhoresPontos, maiorJaccard$
16: **end procedure**

Após o encaixe dos pontos candidatos, é necessário medir qual dos encaixes possui maior grau de similaridade e, portanto, maior confiabilidade. Para isto, aplica-se a métrica de similaridade de Jaccard entre os mapas $map\_1$ e $map\_2$, exemplificados pela letra (a) e (b), da Figura 16. Esta metrificação necessita de que duas operações sejam realizadas entre os mapas, a primeira é uma intersecção *inter*, exemplificado pela letra (c), da Figura 16, e a segunda é uma união *uni*, exemplificado pela letra (d), da Figura 16. No resultado destas operações, conta-se o número de *pixels* pretos na intersecção *inter* e na união *uni* e então dividi-se a quantidade de *pixels* pretos de *inter* pela de *uni*. Desta forma, obtêm-se a métrica de similaridade referente a cada dupla de candidatos a pontos

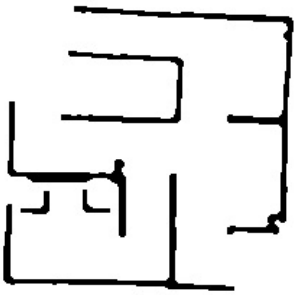
de casamento, a dupla que possuir maior similaridade é escolhida como melhores pontos de casamentos atuais.

Figura 16 – Exemplo de aplicação da união e intersecção sobre dois mapas.

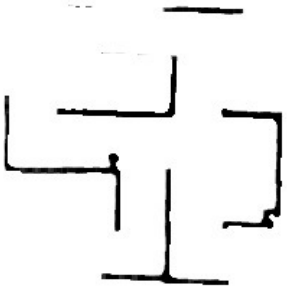
(a) Mapa 1.



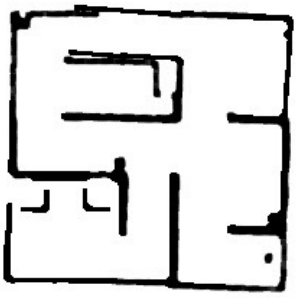
(b) Mapa 2.



(c) Intersecção.



(d) União.



### 3.1.7 Casamento entre os mapas

Após a definição dos melhores pontos de casamento, tem-se então que realizar de fato o casamento entre os mapas. Para isso, é realizada uma sequência de operações, conforme pode-se observar no Algoritmo 11. Estas operações são de simples encaixe entre os mapas, pois neste momento os melhores pontos já estão definidos. Realiza-se a translação

o mapa $map\_2$, a partir da posição do canto $c_{z,2}$ pertencente a $melhoresPontos_1$, sobre a posição de $c_{i,1}$, também pertencente a $melhoresPontos_1$. Após esta operação, realiza-se uma rotação tendo a posição do canto $c_{i,1}$ como centro, a rotação ocorre de forma a encaixar a posição do canto $c_{k,2}$ sobre a posição do canto $c_{j,2}$, ambos pertencentes à $melhoresPontos_2$.

**Algoritmo 11:** Casamento entre mapas

1: **procedure** CASAMENTO($melhoresPontos, map\_1, map\_2$)
2: $map\_2 =$ $map\_2$ transladado da posição do ponto $c_{z,2}$ para posição do ponto $c_{i,1}$
3: $map\_2 =$ $map\_2$ rotacionado em torno da posição do ponto $c_{i,1}$ pelo ângulo formado entre os vetores $\overrightarrow{c_{i,1}c_{j,1}}$ e $\overrightarrow{c_{i,1}c_{k,2}}$, sendo que $c_{j,1}$ e $c_{k,2}$ pertencem a $pontosCandidatos_{a,2}$
4: $referenciaGlobal\_map\_2 =$ a translação e a rotação que $map\_2$ recebeu nas linhas 2 e 3
5: $casamento =$ todas as cores cinza dos mapas $map\_1$ e $map\_2$
6: $casamento =$ todas as cores brancas dos mapas $map\_1$ e $map\_2$ e sobrepõe a cor cinza
7: $casamento =$ todas as cores pretas dos mapas $map\_1$ e $map\_2$ e sobrepõe a cor cinza e a cor branca
8: **retornar** $casamento$
9: **end procedure**

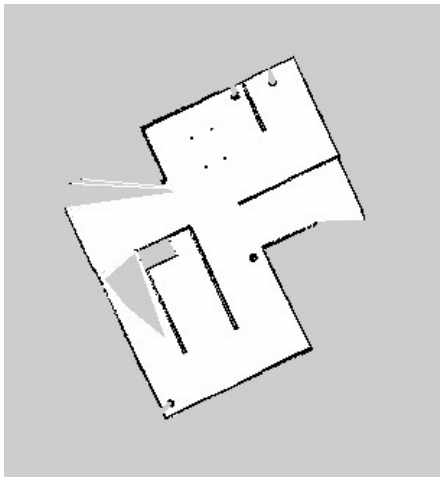
Após a translação e a rotação, gera-se uma nova imagem que irá receber as características de ambos os mapas $map\_1$ e $map\_2$, essa transferências de características seguem as seguintes regras:

- Cor cinza é a primeira a ser transferida para imagem;
- Cor branca é a segunda a ser transferida para imagem e cobre a cor cinza quando houver sobreposição de cores para a mesma posição;
- Cor preta é a terceira a ser transferida para imagem e cobre as cores cinza e branco, quando houver sobreposição de cores para a mesma posição.

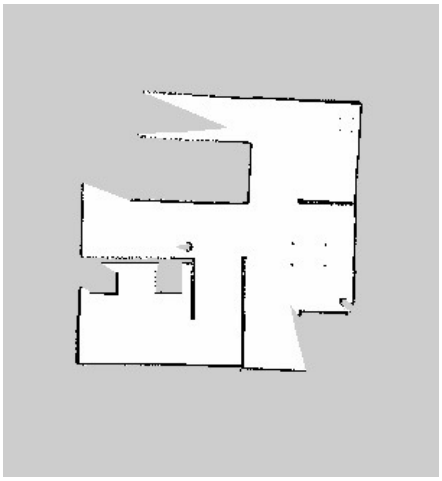
Colocando estas regras em prática, tem-se então o novo mapa "*casamento*"gerado. Uma exemplificação do resultado do casamento pode ser observada na letra (c), da Figura 17, em que há o resultado do casamento dos mapas representado pelas letras (a) e (b), da mesma figura.

Figura 17 – Exemplo de casamente entre dois mapas.

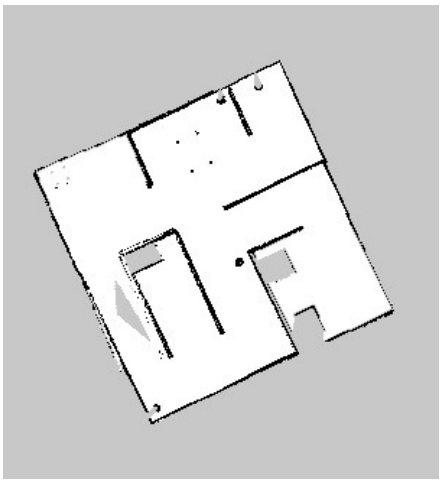
(a) Mapa 1.



(b) Mapa 2.



(c) Casamento entre os mapas 1 e 2.

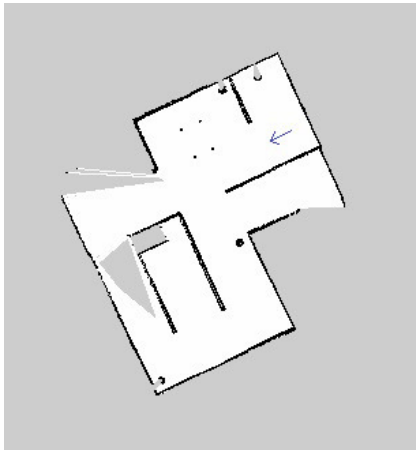


A referência global do mapa *map*_1 continua a mesma do processo de mapeamento. A referência global do mapa *map*_2 recebe as mesmas operações de translação e rotação que *map*_2 recebeu no Algoritmo 11. Na letra (a), da Figura 18, a referência

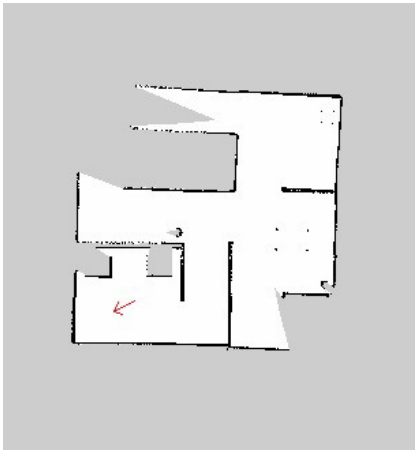
global do mapa é referenciada pela seta azul, em que a direção da seta representa a orientação inicial do Robô 1 e a base da seta é a posição inicial do Robô 1 que mapeou o $map\_1$. Na letra (b) da Figura 18, a referência global do mapa é referenciada pela seta vermelha, em que a direção da seta representa a orientação inicial do Robô 2 e a base da seta é a posição inicial do Robô 2 que mapeou o $map\_2$. Dessa forma, quando as operações de translação e rotação são aplicadas ao $map\_2$, essas mesmas operações são também aplicadas na referência global do Robô 2, de forma que o resultado no casamento pode ser observado na letra (c), da Figura 18, em que a referência global do Robô 2 é ajusta ao casamento dos mapas.

Figura 18 – Referência global no casamento entre dois mapas.

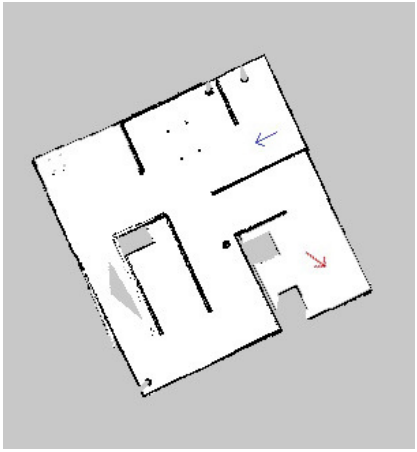
(a) Referência global do mapa 1.



(b) Referência global do mapa 2.



(c) Referência global no Casamento.



## 3.2 Metodologia

### 3.2.1 Materiais

Para a confecção deste trabalho, foi necessário um conjunto de materiais, como um computador com uma configuração robusta, um sistema operacional Ubuntu, um sistema para manipulação de robôs, um simulador robótico, um gerador de ambientes e um sistema para mapeamento destes ambientes.

Com relação ao computador, a configuração de máquina utilizada foi: processador Intel *Kaby Lake* I7-7700HQ (6 MB cache até 3.8 GHZ), memória RAM 32 GB DDR4 (2133 MHZ), placa de vídeo NVIDIA GEFORCE GTX 1050 GPU (4 GB GDDR5), um SSHD de 1 TB com 8 GB SSD, um SSD SATAE M.2 de 480 GB (500 MB/S) e sistema operacional Ubuntu 16.04 LTS 64 bits.

Como sistema para manipulação de robôs, foi utilizado o *Robot Operational System* (ROS) versão KINETIC. Como simulador robótico, foi utilizador o V-REP. Como gerador de ambientes, foi utilizado o V-REP *map generator*[1] (JELINEK, 2016). Como sistema para mapeamento, foi utilizado o *Toolkit* SLAM *Gmapping*. Utilizou-se também o *software Oracle VM VirtualBox* para virtualização do sistema operacional.

#### 3.2.1.1 ROS

O ROS (*Robot Operating System*) é um sistema com funcionalidades de um sistema operacional (muito semelhante a um *framework*), criado para manipulação de robôs. Esse sistema fornece bibliotecas e ferramentas para auxiliar no desenvolvimento de aplicativos para robôs. Além disso, "ele também fornece abstração de *hardware*, *drivers* de dispositivo, bibliotecas, visualizadores, passagem de mensagens, gerenciamento de pacotes e muito mais"(ROS, 201?).

Segundo Dantas (2017, p. 45), o ROS consegue atender à demanda de uma grande variedade de robôs de complexidade variada. O ROS é geralmente instalado através de um pacote Debian. Dessa forma, esses pacotes são apenas compatíveis com algumas versões do Ubuntu (especificadas na *home page* do ROS). Quando instalado e

[1] Disponível em: https://github.com/Lukx19/Vrep-map-generator

configurado, o ROS funciona junto com o Linux, de forma a se tornar uma extensão do próprio Linux.

Dantas (2017, p. 45-46) também descreve as características que tornam vantajoso o uso do ROS como ferramenta computacional para controle de robôs reais e simulados: "A vantagem do ROS é que ele de uma maneira simples consegue fazer com que aplicações se comuniquem entre si, mesmo quando executadas em máquinas diferentes e/ou usando diferentes linguagem de programação". Isso é possível porque no ROS, cada processo é executado sob um padrão, sendo este representado por um nó. Esses nós podem trocar informações na forma de envio ou recebimento de mensagens (Essas mensagens podem ser dos tipos strings, números flutuantes, inteiros, vetores e etc.). O meio para troca de mensagens é chamado de tópico e eles podem ser classificados como *publisher* ou *subscriber*. O *publisher* relaciona-se ao envio de mensagens, enquanto o *subscriber* refere-se ao recebimento de mensagens. Os nós de um sistema utilizando ROS podem estar distribuídos em várias máquinas diferentes, sendo que um deles deve ser atribuído como nó *Master*. Esse nó *Master* gerencia os registros de tópicos e serviços, ele também armazena as informações sobre os outros nós ativos. As máquinas envolvidas no sistema podem conectar-se ao nó *Master* através da rede TCP/IP Dantas (2017, p. 45-46).

Além disso, ROS possui ainda a vantagem de realizar a manipulação de diferentes tipos de coordenadas:

> As coordenadas são importantes para saber onde está localizado, por exemplo, o centro do robô, as rodas, os sensores, as juntas e os outros componentes. As transformações entre as coordenadas são publicadas usando o sistema de tópicos. A transformação atual entre duas coordenadas podem ser obtidas lendo as mensagens do tópico *tf* (DANTAS, 2017, p. 46).

#### 3.2.1.2 V-REP

Segundo Dantas (2017, p. 43), "o V-Rep é um simulador de robô, desenvolvido e mantido pela Coppelia Robotics"[2]. Esse autor afirma ainda que o V-REP permite escolher entre três tipos de *physics engine* (utilizados para simular processos dinâmicos), que são estes: ODE, *Bullet* e *Vortex*. O V-REP é descrito como:

[2] Disponível em: www.coppeliarobotics.com

> O ambiente de simulação é modelado usando o editor de cenas do programa. O V-Rep contém um grande número de modelos predefinidos para diferentes tipos de robôs. O comportamento do simulador é totalmente personalizável e suas funcionalidades podem ser estendida através de *plugins*. Um *plugin* é uma biblioteca de *software* que é escrita em C ++ e interage com o simulador usando a API de programação fornecida. O V-Rep também permite adicionar *scripts* embutidos para simulações específicas. Esses *scripts* precisam ser escritos na linguagem de programação conhecida como LUA. O V-Rep não é um *software* de código aberto, mas fornece uma licença livre para unidades educacionais e pode, portanto, ser usado para fins de pesquisa. As simulações no V-Rep são organizadas em cenas. Uma cena representa o mundo simulado que é composto de uma série de elementos, como os corpos tridimensionais, as articulações, os sensores e as câmeras, dentre outros. Esses elementos são combinados em uma estrutura de árvore e são dispostos dentro do ambiente. Essa estrutura de árvore forma a hierarquia de cena. Os elementos dentro dessa hierarquia são chamados de objetos de cena. O V-Rep fornece vários tipos de objetos de cena (DANTAS, 2017, p. 43).

Desta forma, dentre os vários tipos de objetos fornecidos pelo V-REP, os mais importantes no desenvolvimento deste trabalho são o *shape*, a junta e a visão, estes são definidos como (DANTAS, 2017, p. 43-44):

- *Shape*: Os *shapes* são objetos rígidos, sendo esses estruturas do robôs ou paredes. No V-Rep estão presentes cinco formas primitivas, sendo o cilindro, o cuboide, a esfera, o plano, o disco e também as formas complexas como malhas triangulares. Essas formas podem ser agrupadas e tratadas como um objeto único. Elas também podem ser definidas como estáticas ou não estáticas. Segundo Dantas (2017, p. 43-44): "a posição de um objeto estático é fixa em relação ao seu nó pai dentro da hierarquia da cena. Os objetos não estáticos são submetidos à gravidade e cairão se não estiverem conectados a uma junção ou um apoio rígido". Também é preciso diferenciar entre formas respondíveis e não respondíveis. Sendo que as formas respondíveis são manipuladas através do módulo *physics engine* e assim geram reações de colisão. Dessa forma, seus atributos físicos (como massa, momento de inércia e configurações de material) precisam ser claramente definidos. As formas não respondíveis são visíveis em cena, mas não geram reações de colisão (DANTAS, 2017, p. 43-44);

- Junta: é uma conexão entre duas partes rígidas de um robô, sendo essa conexão flexível. As juntas que foram utilizadas no modelo robótico neste projeto são articulações revolutas, que são caracterizadas por permitir a rotação em um único eixo. As juntas são atuadas por motores e às configurações do motor são incluídos limites máximos de torque e de velocidade (DANTAS, 2017, p. 43-44);

- Visão: funciona a partir da simulação de dispositivos de captura de imagens. Os sensores de visão capturam imagens de acordo com sua localização dentro da cena, do ângulo de visão e a resolução. "Os sensores de visão disponíveis são capazes de produzir imagens RGB e de profundidade"(DANTAS, 2017, p. 43-44).

Com esses objetos, é possível definir as estruturas necessárias para a simulação do robô, em que os modelos de robô são compostos por elos (corpos rígidos) e estes são ligados por juntas. Essas juntas proporcionam o acionamento das partes flexíveis do robô simulado. Os *links* são geralmente representados de duas formas diferentes. A primeira forma está relacionada com a geometria visual e é geralmente não respondível. A segunda forma está relacionada com a geometria de colisão e essa tem que ser uma aproximação simplificada da geometria visual, que normalmente é composta de grupos de formas primitivas. Essa segunda forma indica a parte respondível do *link* do robô e como ele é percebido pelo *physics engine*. Desta forma, é necessária uma configuração apropriada dos parâmetros dinâmicos. A parte respondível é normalmente invisível, porém muito importante, pois com ela são aplicados comportamentos realistas ao modelo, permitindo que ele interaja com outros objetos também respondíveis (dentro da cena). Não havendo peças respondíveis, o modelo iria apenas se movimentar através de outros corpos, visto que somente formas respondíveis podem produzir reações de colisão(DANTAS, 2017, p. 44).

É possível integrar o V-REP com o ROS, pois o V-Rep possui um *plugin* em C++. Esse *plugin* consegue fazer a comunicação entre os *scripts* que são escritos em LUA e os tópicos do ROS (DANTAS, 2017, p. 45). O ROS sendo executado no sistema embarcado no robô possui *scripts*. Esses *scripts* são utilizados para publicar mensagens de odometria (*odom*) e mensagens de imagens (RGB e profundidade)(DANTAS, 2017, p. 45).

#### 3.2.1.3 Gmapping

Gouveia (2013, p. 23) descreve o Gmapping, desenvolvido por Grisetti *et al.* (2007), como:

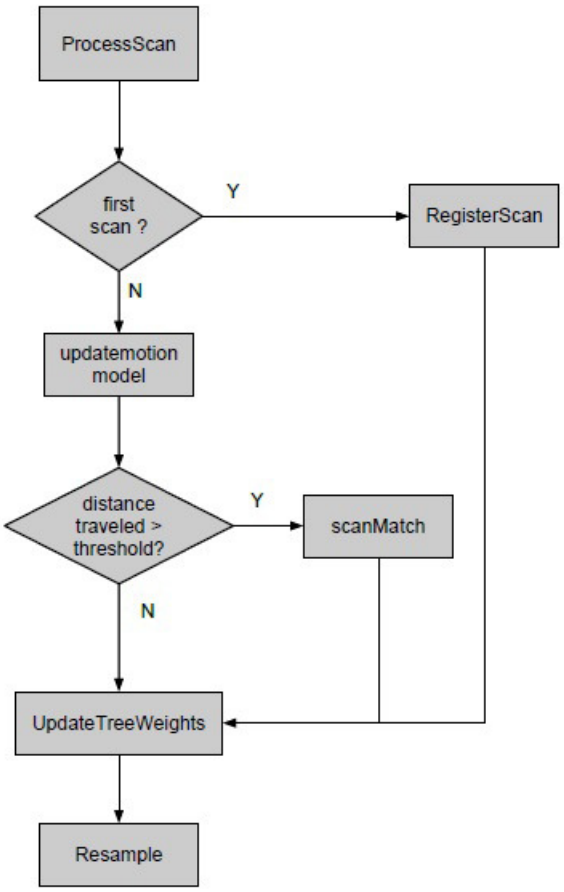
> Um filtro de partículas Rao BlackWellized em que cada partícula mantém o seu próprio mapa e a pose do robot. Como estrutura principal, usa uma árvore em que cada nó representa uma parte da trajetória do robô. Cada nó mantém a pose do robô, um ponteiro para a estrutura do *LaserScan* efetuado nesse momento, os pesos calculados a partir do *LaserScanMatch* (operação de otimização que tenta estimar o valor da rotação e translação entre dois vectores da pose do robô usando as características do mapa) e um ponteiro para o nó anterior. A árvore tem portanto o número de folhas igual ao número de partículas no algoritmo, são guardadas numa estrutura "Vetor"para serem mais facilmente processadas. Para obter um mapa atual a partir da partícula com melhor peso, basta atravessar a árvore da folha (partícula escolhida) até à raiz e em cada passo marcar o mapa com o dados do *LaserScan* ("*RegisterScan*"), marcando as células do mapa como vazio, ocupado ou desconhecido).

Esse autor detalha ainda a sequência de operações realizadas pelo Gmapping (observada no fluxograma da Figura 19) que, ao receber a odometria e os dados de leitura do *scanner laser*, realiza a sequência de passos:

- Caso os dados recebidos sejam os primeiros, registra-se os dados do *Laser* em cada partícula do mapa, neste caso, marca-se as células como ocupada, vazia ou desconhecida e então executa-se a função *resample*;
- Caso não sejam os primeiros dados, atualiza-se a pose do robô em cada partícula, baseando-se no modelo de movimento (usando odometria);
- Caso a distância percorrida seja maior que o limite (parâmetro do algoritmo), executa-se um *LaserScanMatch* utilizando os dados dos sensores e então corrige a pose de acordo com o mapa de cada partícula;
- Atualiza-se os pesos da arvore e então calcula-se o parâmetro $N_eff$ de acordo com a Equação 3.4, em que $\tilde{w}^i$ é o peso normalizado da partícula $i$;
- Executa-se a função *Resample*.

$$N_eff = \frac{1}{\sum_{i=1}^{N}(\tilde{w}^i)^2} \tag{3.4}$$

Figura 19 – Fluxograma de ações do gmapping




Fonte: (GOUVEIA, 2013)

Ainda segundo Gouveia (2013, p. 24), caso o parâmetro $N_eff$ seja inferior a um limite, a função "*Resample*"executa o *resampling*, selecionando aleatoriamente (com reposição) as partículas. Esta seleção se baseia nos pesos associados e dá prioridade às partículas que têm peso maior, podendo assim, no próximo passo, haver várias cópias da mesma partícula.

A operação "*Resample*"pode ser observada na Figura 20, em que o algoritmo escolhe várias vezes a partícula 0 e 2, descartando a 3 e a 4.

Figura 20 – Resample


fonte: (GOUVEIA, 2013)

Frese *et al.* (2010) ressalta que o Gmapping está presente no ROS e que este foi complementado por um módulo baseado no algoritmo *Adaptative Monte Carlo Localization* (AMCL) (THRUN *et al.*, 2005).

#### 3.2.1.4 Modelo robótico

Para a simulação no V-REP, utilizou-se o modelo de robô proposto no trabalho de Dantas (2017)[3]. O robô real desenvolvido por Dantas (2017, p. 37) (Figura 21) possui: chassi de alumínio, com motores padrão NEMA 17 acoplados. Esse chassi comporta as baterias, a eletrônica de alimentação, o modem wireless, os drives dos motores e a unidade de processamento de baixo nível (Arduino Mega). O chassi possui espaçadores para o encaixe de uma base de acrílico. Nessa base de acrílico são fixados os servos motores para a movimentação do braço, a unidade de processamento de alto nível (Raspberry Pi 2) e o sensor de visão (ASUS Xtion Pro) (DANTAS, 2017).

Dantas (2017, p. 40) descreve a forma de acionamento de motores de passo: "Para o acionamento dos motores foi utilizado o Arduino Mega em conjunto com um *shield* para motores de passo chamado de RAMPS 1.4".

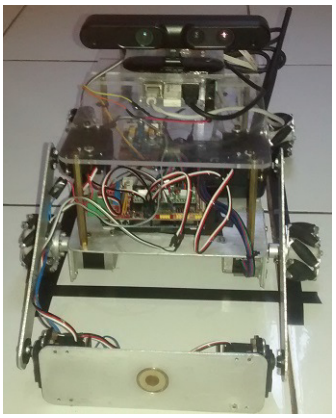
Para este trabalho, o modelo utilizado não continha o braço robótico e o sensor de visão foi substituído por um sensor *scanner laser fastHokuyo* disponível no V-REP. Nesse sensor foi introduzido, via *script*, um erro de 0.01 metros, também foi introduzido uma abertura de leitura de 240 graus em torno do robô. Foi configurado também, no sensor, através de um clique duplo no ícone de três barras paralelas, ao lado do sensor *fastHokuyo* no explorer do V-REP, selecionando a opção "*scanAngle*"e determinou-se

[3] Detelhes do modelo do robô disponível em: https://tede2.ufma.br/jspui/handle/tede/2062

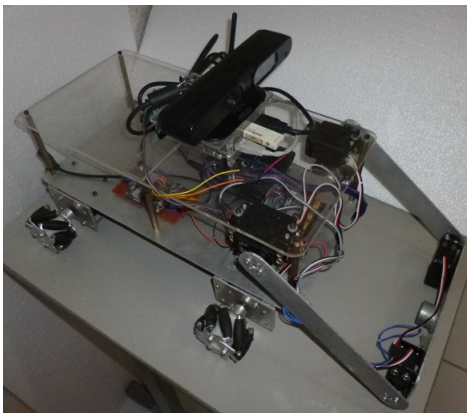
240 graus de abertura para leitura, selecionando a outra opção "*maxScanDistance*", determinou-se um raio de leitura de 5 metros. Esse robô possui uma configuração de roda do tipo quadriciclo, sendo que todas as rodas do tipo mecanum com roletes de 45 graus, conforme observa-se na letra (a) da Figura 22, proporcionando um modelo de movimento omnidirecional ao robô.

Figura 21 – Robô real

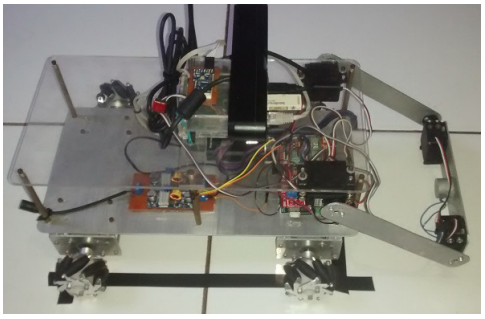
(a) Visão frontal.



(b) Visão transversal.



(c) Visão lateral.

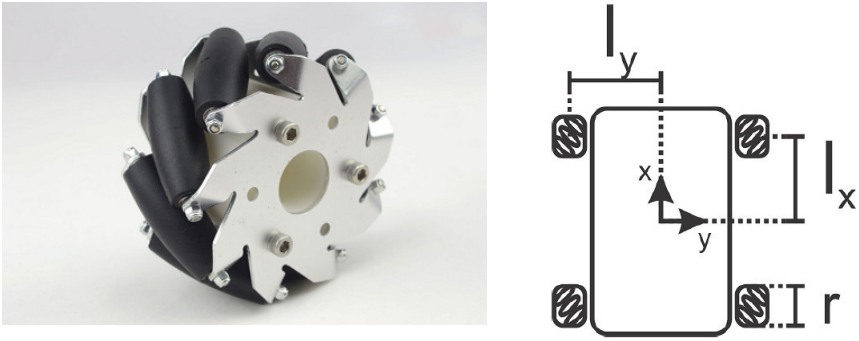


Fonte: (DANTAS, 2017)

A disposição das rodas está conforme demonstrado na letra (b) da Figura 22, em que o raio $r$ das rodas é $0,03$ metros. A distância $l_x$, que é a distância do centro do robô até o centro da roda em relação ao eixo $X$, é $0,133$ metros. A distância $l_y$, que é a distância do centro do robô até o centro da roda em relação ao eixo $Y$, é $0,1294$ metros. A altura do robô é $0,145$ metros, a largura é de $0,188$ metros e o comprimento é $0,320$ metros. O peso relativo ao robô era de aproximadamente $3,868$ quilos.

Figura 22 – Características do robô

(a) Roda mecanum. (b) Configuração do robô.

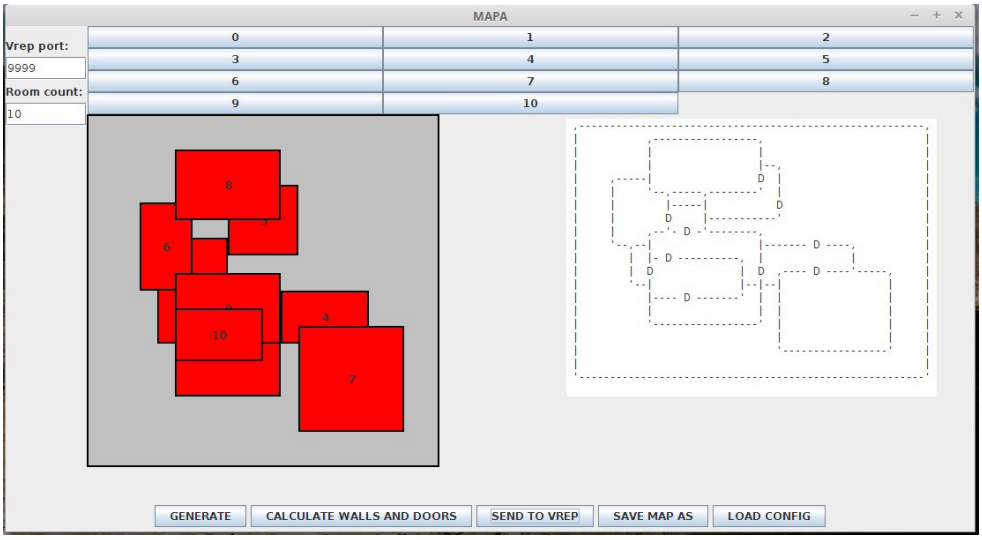



Fonte: (DANTAS, 2017)

#### 3.2.1.5 V-REP *map generator*

É um aplicativo implementado em JAVA por Jelinek (2016) que pode se comunicar diretamente com o V-REP em execução. Esse aplicativo gera um número específico de salas através de um botão *generate*, essas salas pode ser movidas selecionando-as e movendo-as, via setas do teclados. Após configuração de posição dessas salas, clica-se no botão *calculate walls and doors* para que o aplicativo gere aleatoriamente portas, de forma que de qualquer sala seja possível chegar em outra (JELINEK, 2016).

Figura 23 – Visão do V-REP *map generator*




fonte: (JELINEK, 2016)

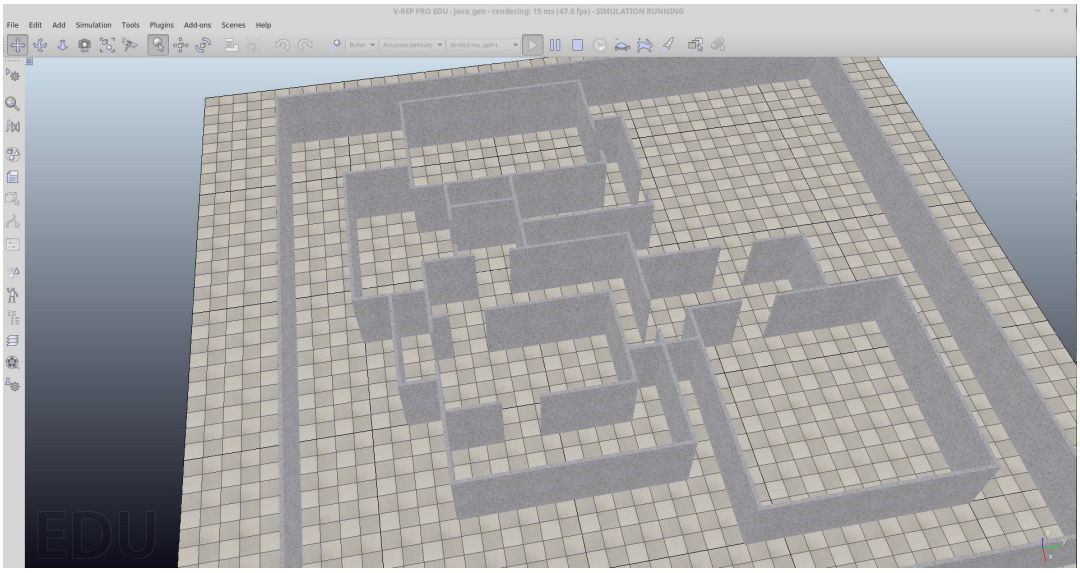
É possível gerar o mapa e transferi-lo diretamente para o V-REP, esta operação é realizada através da seguinte forma (JELINEK, 2016):

- No V-REP: carrega-se a cena "java_gen.ttt", localizada em "Vrep-map-generator/vrep/";
- No V-REP: realiza-se duplo clique no ícone de lista próximo ao *object renderer*, no *project explorer*, um editor irá abrir e então trocam-se os caminhos absolutos de *Walls* e *Maps*, baseando-se no caminho em que está localizado o V-REP *map generator*:

  SOURCE_WALLS="/home/andre/workspace/Vrep-map-generator/vrep/walls/",

  SOURCE_MAP="/home/andre/workspace/Vrep-map-generator/vrep/maps/";
- No V-REP: realiza-se clique duplo no ícone de lista azul, próximo ao objeto *Resizable_floor* no *explorer*, troca-se a porta de comunicação, conforme a descrição: *local server = assert(socket.bind("*", 9999))*. A porta utilizada deve ser a mesma que consta no V-REP *map generator*;
- No V-REP: pressiona-se *run*, na barra superior;
- No V-REP *map generator*: o mapa é gerado e pressiona-se *send to vrep* (conforme observa-se na Figura 23). Se as portas estiverem corretamente configuradas, o mapa irá aparecer no V-REP (conforme observa-se na Figura 24) e este será salvo na pasta "Vrep-map-generator/vrep/maps". Após o fim da simulação, o mapa irá desaparecer do V-REP, mas é possível carregar o mapa salvo de volta apenas carregando a nova cena salva.

Figura 24 – Visão do mapa enviado ao V-REP




fonte: (JELINEK, 2016)

### 3.2.2 Métodos

Para realização deste trabalho, foi necessário a utilização dos materiais descritos no Tópico 3.2.1 de materiais. De forma que os *softwares* descritos foram instalados em uma máquina virtual Ubuntu, com a configuração de quatro processadores e 16 GB de RAM.

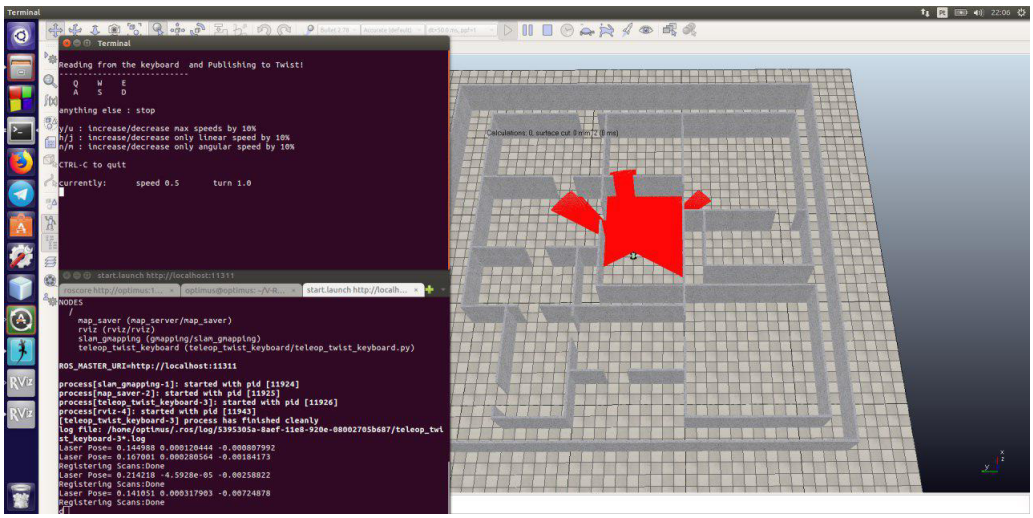
Utilizando o V-REP *map generator* conectado com o V-REP, pode-se gerar e salvar cenas de dez ambientes aleatórios e diferentes entre si. Esses ambientes foram gerados com o objetivo de serem mapeados, a fim de gerar uma base de dados para o testes de casamentos de mapas.

No V-REP, para cada cena de ambiente gerada, foi "importado", via menu "*file->load model*", o modelo de robô descrito no tópico 3.2.1.4, salvando a cena logo em seguida.

Para que a conexão da integração entre ROS e V-REP fosse aplicada, foi preciso primeiro executar, via terminal, o comando "roscore", para então abrir outro terminal e executar o comando de abertura do V-REP "./vrep.sh"(através do caminho de onde o V-REP está instalado). Executando os referidos comandos nesta sequência, o *plugin* de

integração do ROS com o V-REP foi inicializado corretamente e assim os aplicativos se comunicaram, via ROS, com as cenas no V-REP.

Figura 25 – Visão da tela de controle do robô




Com o V-REP aberto e se comunicando com o ROS, carregou-se uma cena de ambiente para mapeamento. Com a cena carregada, clicou-se no botão "*run*"na barra superior do V-REP. Com a cena em executação, via terminal e através do comando "*roslaunch*", executou-se: o pacote de SLAM gmapping, um *launch* de salvamento com o caminho em que o processo de mapeamento deva ser salvo e o aplicativo de controle do robô (*teleop twist keyboard*), via setas do teclado [4]. Desta forma, através do terminal do *teleop twist keyboard*, em primeiro plano, realiza-se o controle de movimento do robô, conforme observa-se na Figura 25. O controle do robô é dado da seguinte forma:

- A tecla "w" e "s" faz com que o robô se mova para frente e para trás, respectivamente;
- A tecla "a" e "d" faz com que o robô se mova para esquerda e para direita, respectivamente;
- A tecla "q" e "e" faz com que o robô rotacione no sentido anti-horário e horário, respectivamente;

[4] O *teleop twist keyboard* está presente no ROS e através deste envia sinais de velocidade ao V-REP, que por sua vez aplica esses sinais ao robô. Disponível em: http://wiki.ros.org/teleop_twist_keyboard

- A tecla "u" e "y" reduz e aumenta, respectivamente, em 10% as velocidades linear e angular do robô;
- A tecla "j" e "h" reduz e aumenta, respectivamente, em 10% a velocidade linear do robô;
- A tecla "m" e "n" reduz e aumenta, respectivamente, em 10% a velocidade angular do robô;
- Qualquer outra tecla faz com que o robô pare;

Desta forma, utilizando o aplicativo de controle, via teclado *teleop twist keyboard*, e aplicando a velocidade linear de 0.8 m/s e a velocidade angular de 1.61 rad/s, foram realizados, via controle humano, cinco mapeamentos de cada um dos dez ambientes gerados (para cada ambiente, cada mapeamento aconteceu a partir de poses iniciais diferentes), formando assim uma base de dados de 50 mapeamentos. Para cada ambiente foi criada uma possibilidade de dez casamentos utilizando dois mapeamentos referentes a este ambiente. A base de dados gerada pelo gmapping é composta dos históricos de mapeamentos, sendo que para cada mapeamento e a cada período de tempo de um segundo (caso haja mudança no estado do mapa), o estado do mapeamento naquele instante é salvo em forma de imagem.

Após a geração da base de dados, para cada ambiente, insere-se no sistema implementado com base na método proposto neste trabalho, dois processos de mapeamento de cada vez, de forma a nunca repetir uma combinação. A inserção é realizada, para cada ambiente, colocando cada um dos seus cinco processos de mapeamento em pastas diferentes, o caminho dos mapeamentos sendo casados deve ser fornecido ao sistema e cada mapa no processo de mapeamento deve estar numerado de acordo com a sequência em que foi gerado. Assim, há dez possíveis combinações de casamento para cada ambiente. Esta operação foi realizada com a seguinte configuração de parâmetros no sistema:

- $N = 30$, referente ao número de cantos a serem extraídos de cada mapa;
- $q = 0.2$, referente a qualidade dos cantos a serem extraídos de cada mapa;
- $D_{max}$ é metade da maior distâncias entre cantos, levando em consideração os dois mapas sendo casados;

- $D_{min} = 18$, sendo este valor em *pixels*;
- $erro\_distancia = 1.1$, sendo este valor me *pixels*;
- $erro\_angulo = 1.1$, sendo este valor em graus;

Com base no método proposto, o sistema retorna, em relação à base de dados geradas, uma quantia de 100 casamentos realizados. Inicialmente, os resultados foram avaliados de forma qualitativa, conforme observa-se no Capítulo 4 de resultados. Observando uma alta taxa de sucesso nos casamentos, realizou-se então uma análise quantitativa dos casamentos em relação aos mapeamentos da base de dados, analisando a precisão dos mapas de cada ambiente na base em comparação com os casamentos gerados. A avaliação quantitativa é descrita no Tópico 4.1 do capítulo de resultados. Além disso, para melhor entendimento dos motivos de falhas no método, foi realizada uma análise qualitativa nos casos de falha nos Tópico 4.2, também no capítulo de resultados, discorrendo sobre os possíveis motivos observados que levaram o método a falhar, mesmo que em pequena quantidade.

# 4 Resultados dos experimentos

Para os experimentos foram gerados dez ambientes *in-door*, aleatoriamente, com diferentes números de salas. Para cada ambiente foram realizados cinco mapeamentos, sendo que as poses de partida do robô, para cada mapeamento, foram diferentes. Para fim de tentativa de casamento entre os mapas, explorou-se sempre uma área em comum entre eles.

Em geral os casamentos foram realizados a cada dois mapeamentos, gerando um total de dez mapas gerados pelos casamentos para cada ambiente, sendo um total de 100 casamentos. Os mapeamentos foram feitos um robô por vez e os casamentos são efetuados utilizando combinações entre esses mapeamentos, simulando como se dois robôs estivesse mapeando ao mesmo tempo. Alguns casamentos com mais de dois mapeamentos foram realizados para testar a limitação do método, esses casamentos foram realizados em dois ambientes, sem aprofundamento, mas apontando uma direção para a continuidade da pesquisa como trabalhos futuros.

Foi realizada uma análise qualitativa de resultados, em que a confirmação do sucesso do casamento é dada analisando os mapas casados de forma visual.

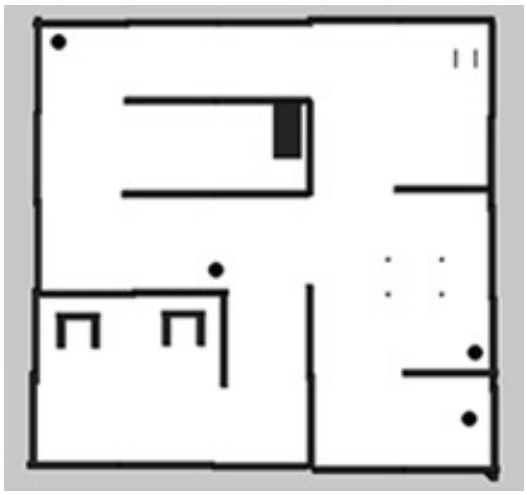
## 4.1 Casos de sucesso no casamento

De 100 casamentos com dois mapeamentos, 79 foram realizados com sucesso, conforme observa-se nas Figuras 26, 27, 28, 29, 30, 31, 32, 33, 34 e 35. A cor vermelha representa o mapeamento feito pelo robô 1, a cor azul representa o mapeamento feito pelo robô 2 e a cor verde representa os locais mapeados em comum.

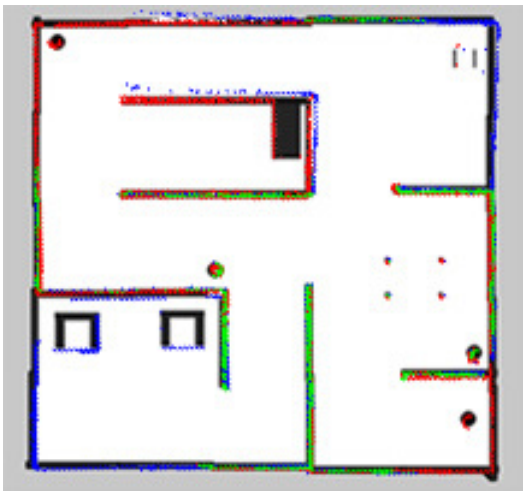
Na letra (a) da Figura 26, pode-se observar o formato do ambiente real. Assim, na letra (b), observa-se que a áreas em comum, demarcadas pela cor verde, costumam ter uma concentração maior em regiões próximas, enquanto à medida em que se afasta da cor verde, as cores azul e vermelha tornam-se mais evidentes, demonstrando que conforme se afasta da zona de casamento (regiões verdes), o erro de cada mapeamento utilizado no processo provoca um perda de precisão no casamento. O mesmo fenômeno pode ser observado nas letras (c) e (d) da mesma figura.

Figura 26 – Representação do ambiente 1 e seus casamentos.

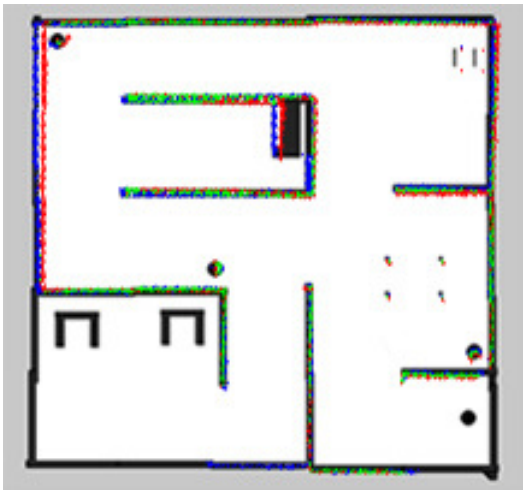
(a) Mapa do ambiente 1.



(b) Casamento dos mapeamentos 1 e 2.



(c) Casamento dos mapeamentos 3 e 4.

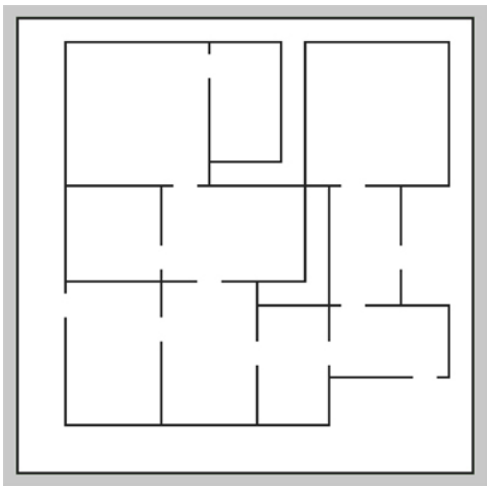


(d) Casamento dos mapeamentos 4 e 5.

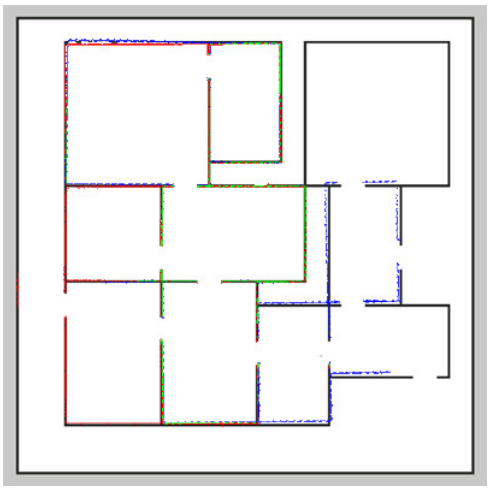
Por a Figura 27 representar um ambiente maior que o ambiente da Figura 26, o fenômeno de perda de precisão, conforme afasta-se da zona verde, fica mais evidente. Assim, pode-se observar tanto a perda de precisão gerada pelo mapeamento representado pela cor azul, quanto pelo mapeamento representado pela cor vermelha.

Figura 27 – Representação do ambiente 2 e seus casamentos.

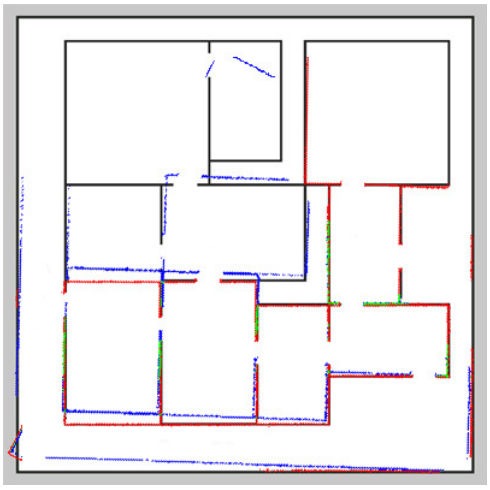
(a) Mapa do ambiente 2.



(b) Casamento dos mapeamentos 1 e 3.



(c) Casamento dos mapeamentos 2 e 4.



(d) Casamento dos mapeamentos 3 e 4.

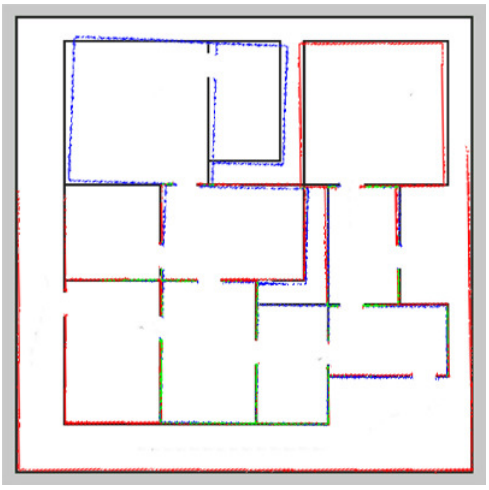
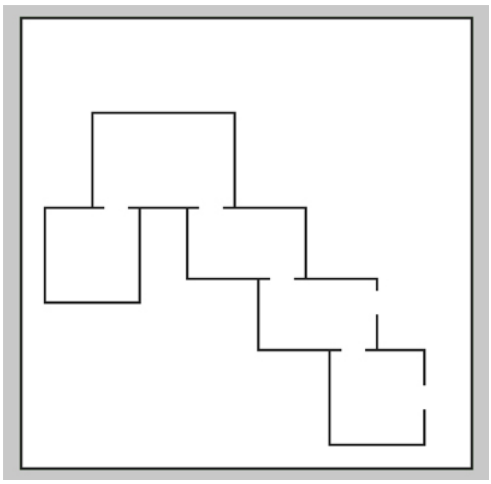


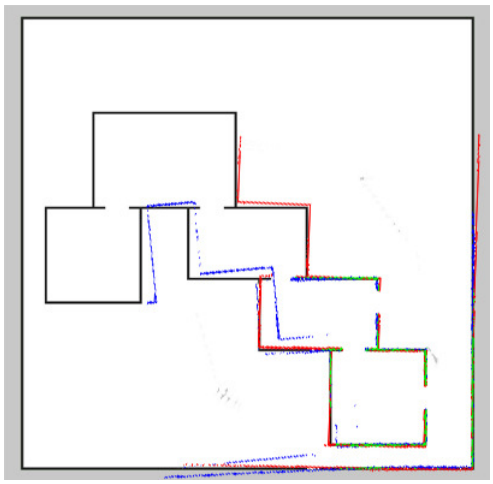
Figura 28 – Representação do ambiente 3 e seus casamentos.

(a) Mapa do ambiente 3.



(b) Casamento dos mapeamentos 1 e 2.

(c) Casamento dos mapeamentos 2 e 3.



(d) Casamento dos mapeamentos 2 e 5.

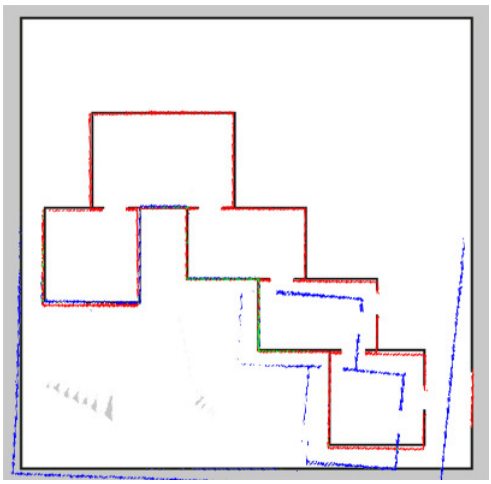
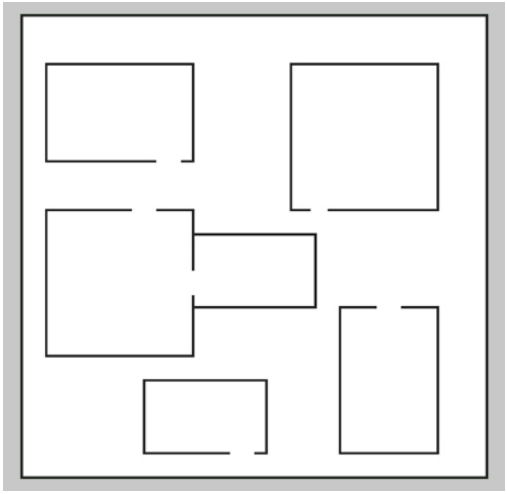
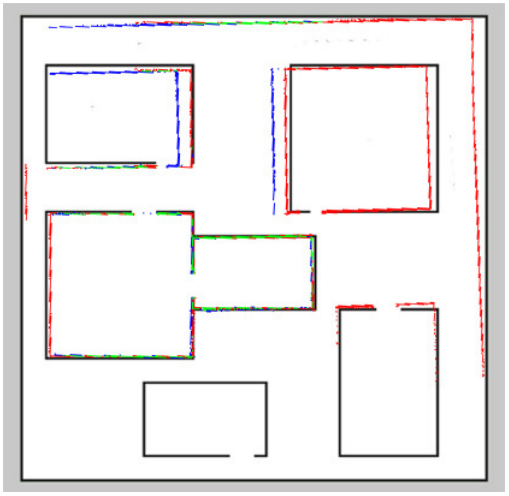


Figura 29 – Representação do ambiente 4 e seus casamentos.

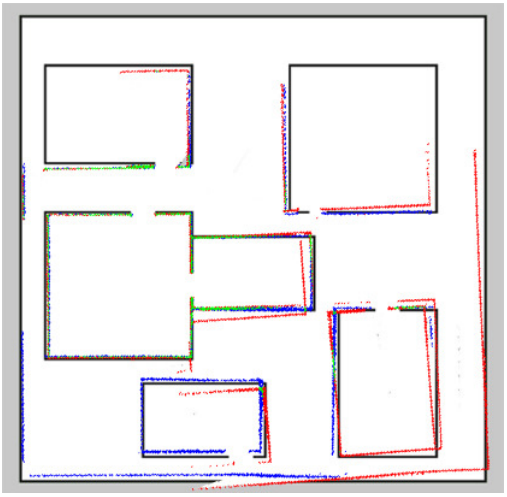
(a) Mapa do ambiente 4.



(b) Casamento dos mapeamentos 1 e 2.



(c) Casamento dos mapeamentos 3 e 4.



(d) Casamento dos mapeamentos 4 e 5.

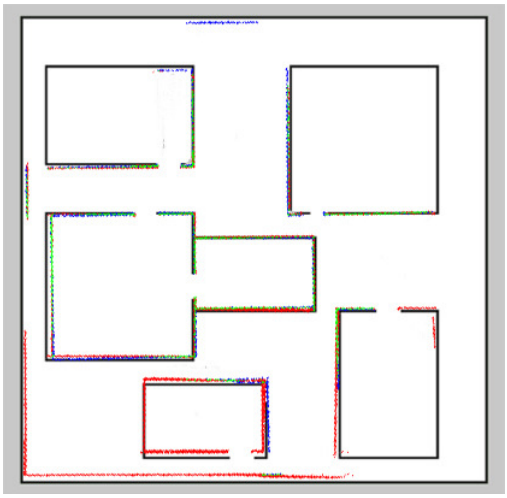
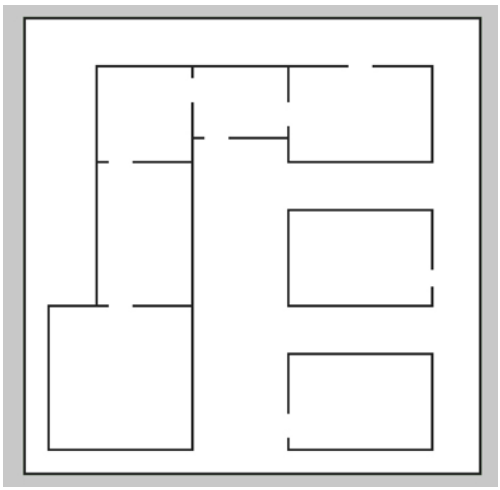
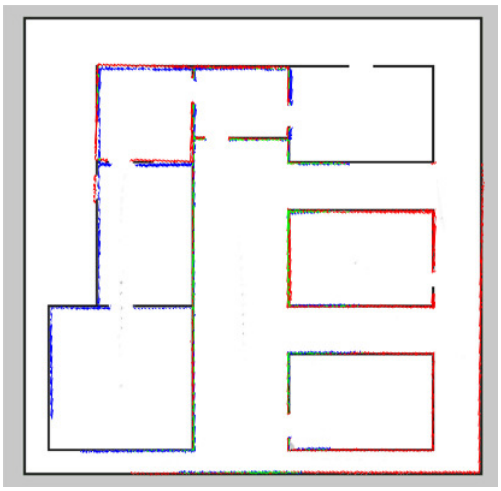


Figura 30 – Representação do ambiente 5 e seus casamentos.

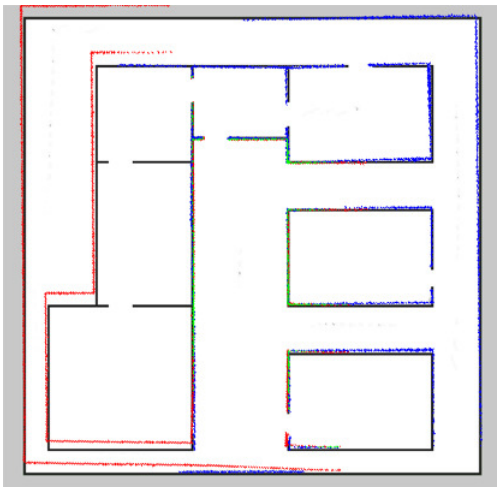
(a) Mapa do ambiente 5.



(b) Casamento dos mapeamentos 1 e 4.



(c) Casamento dos mapeamentos 3 e 5.



(d) Casamento dos mapeamentos 4 e 5.

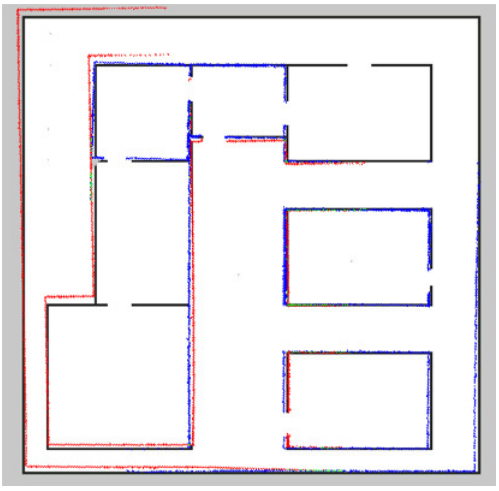
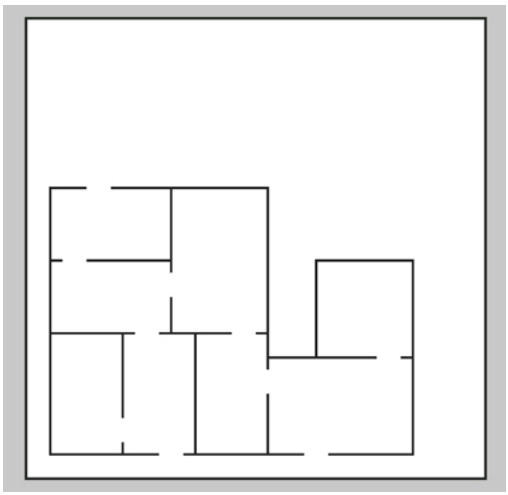
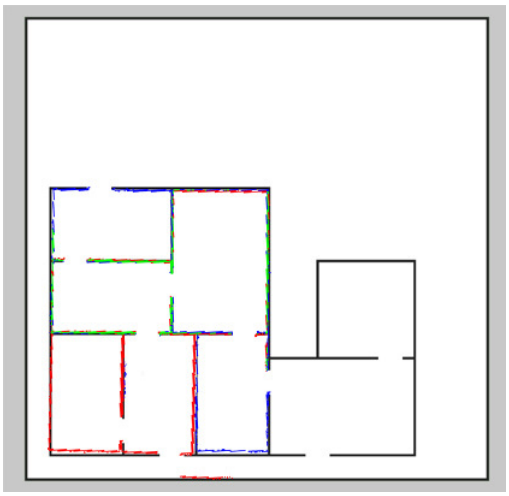


Figura 31 – Representação do ambiente 6 e seus casamentos.

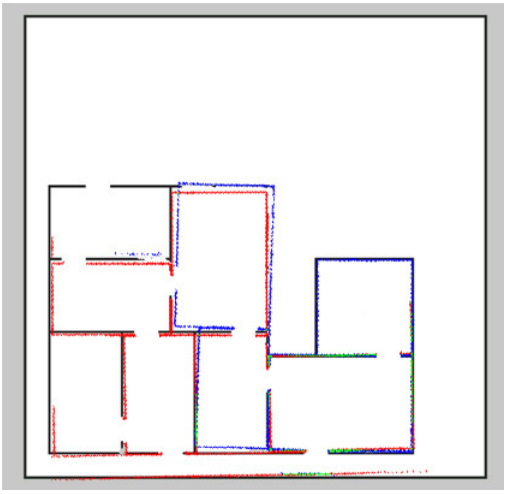
(a) Mapa do ambiente 6.



(b) Casamento dos mapeamentos 1 e 2.



(c) Casamento dos mapeamentos 3 e 4.



(d) Casamento dos mapeamentos 4 e 5.

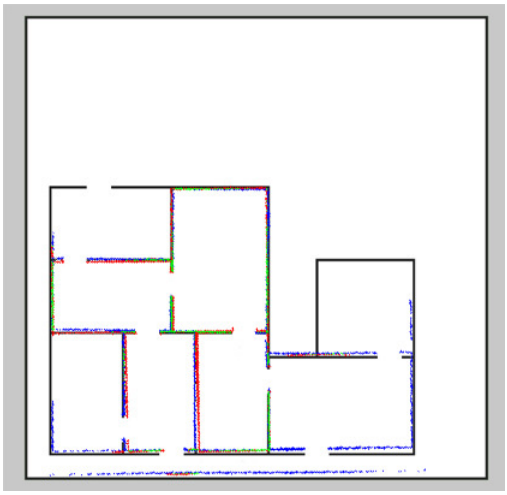
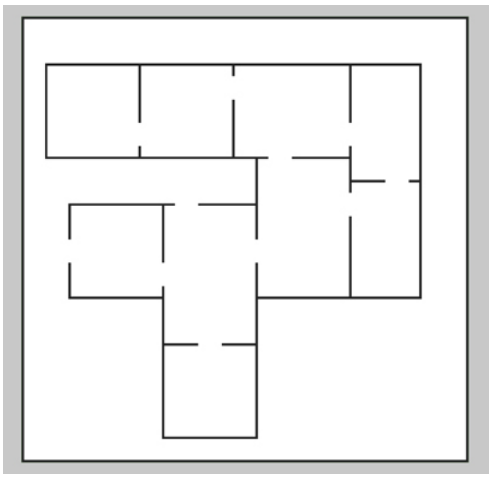
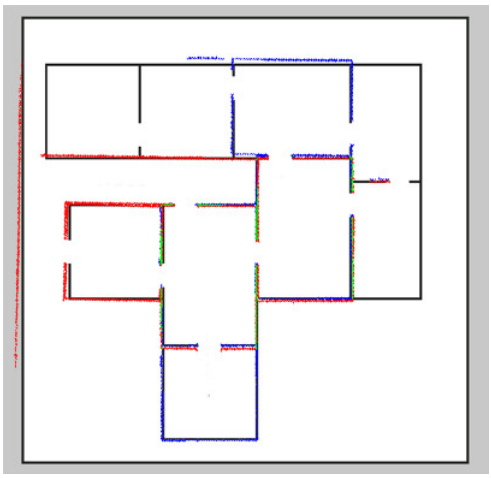


Figura 32 – Representação do ambiente 7 e seus casamentos.

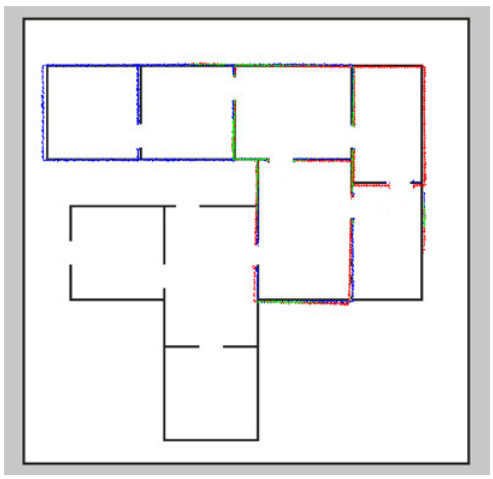
(a) Mapa do ambiente 7.



(b) Casamento dos mapeamentos 1 e 2.



(c) Casamento dos mapeamentos 3 e 4.



(d) Casamento dos mapeamentos 4 e 5.

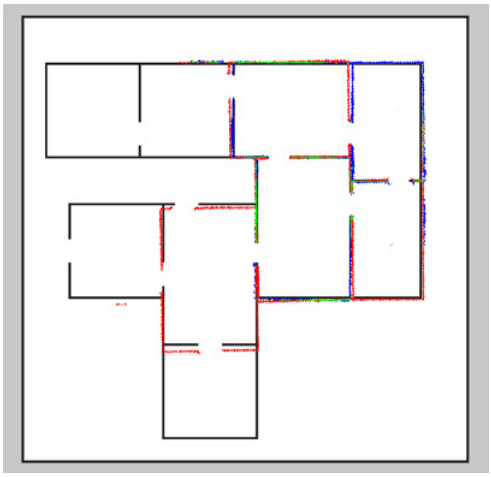
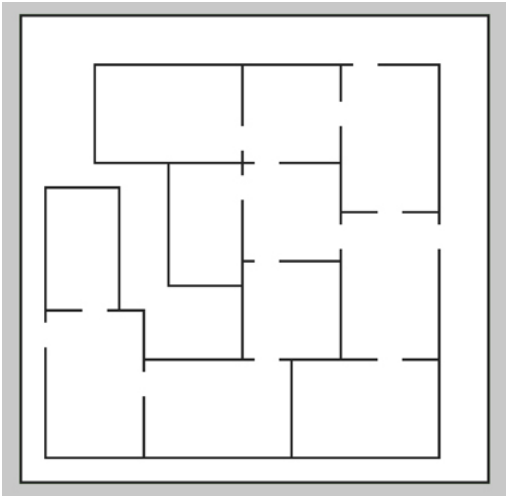
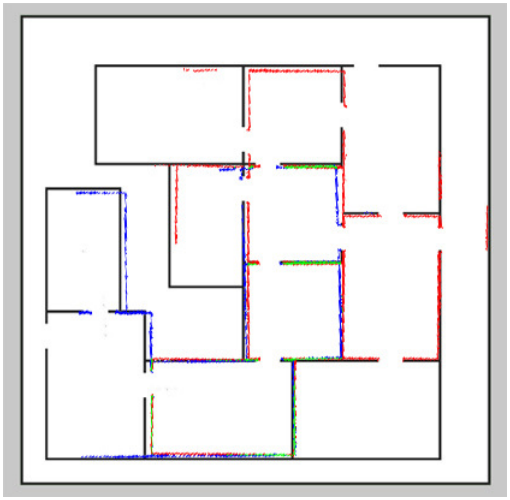


Figura 33 – Representação do ambiente 8 e seus casamentos.

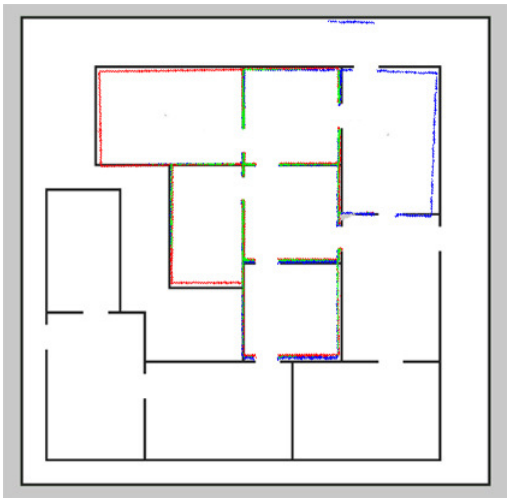
(a) Mapa do ambiente 8.



(b) Casamento dos mapeamentos 1 e 5.



(c) Casamento dos mapeamentos 2 e 4.



(d) Casamento dos mapeamentos 3 e 5.

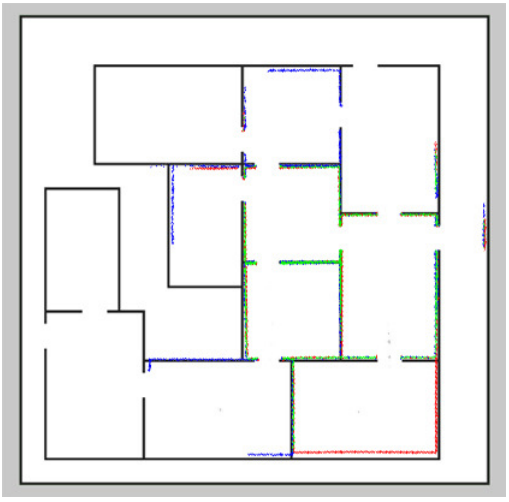
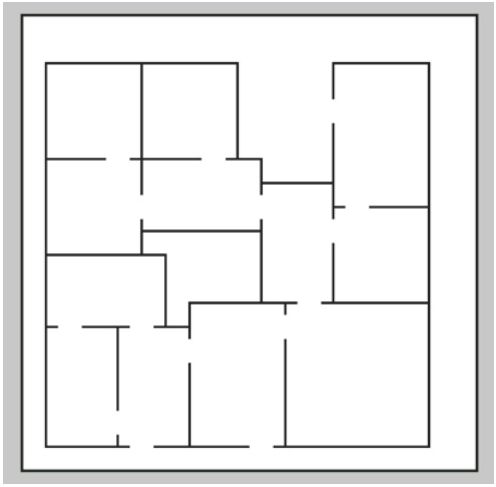
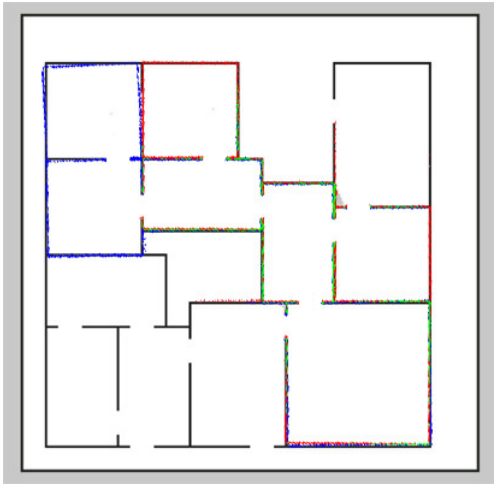


Figura 34 – Representação do ambiente 9 e seus casamentos.

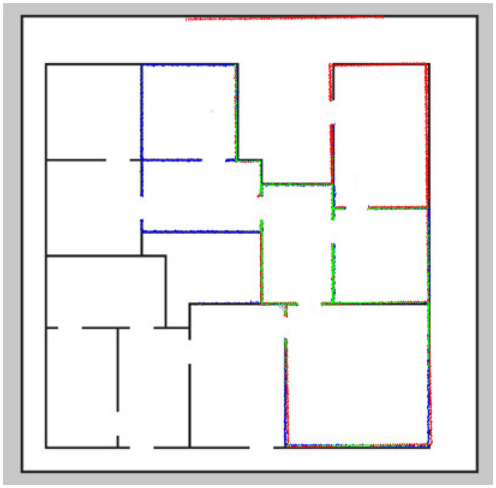
(a) Mapa do ambiente 9.



(b) Casamento dos mapeamentos 1 e 2.



(c) Casamento dos mapeamentos 2 e 5.



(d) Casamento dos mapeamentos 3 e 4.

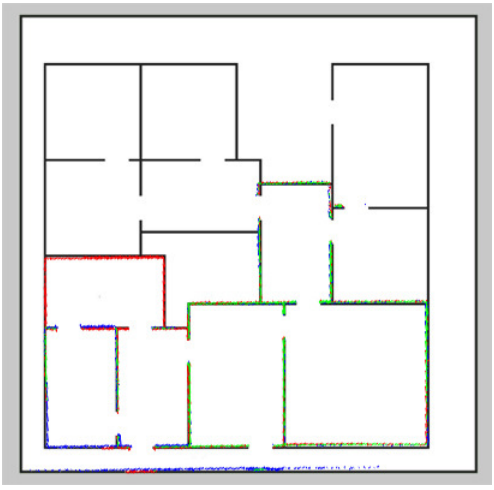
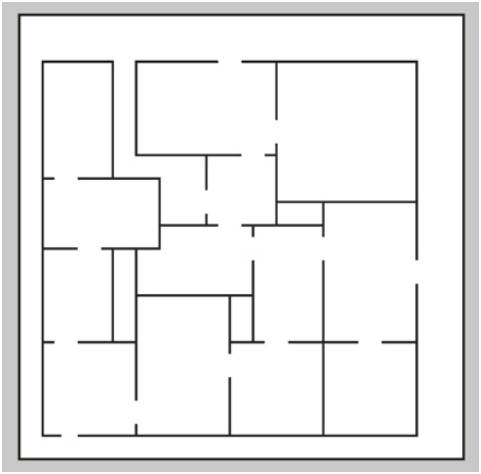
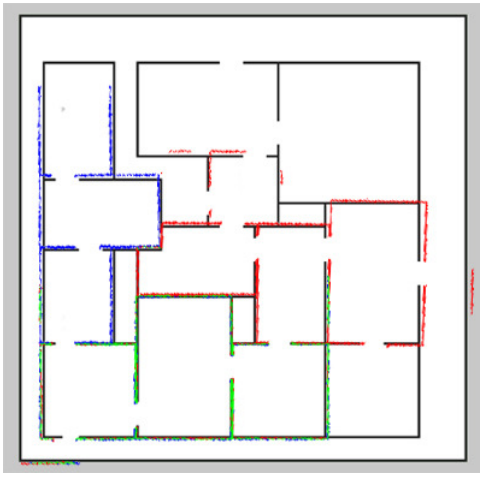


Figura 35 – Representação do ambiente 10 e seus casamentos.

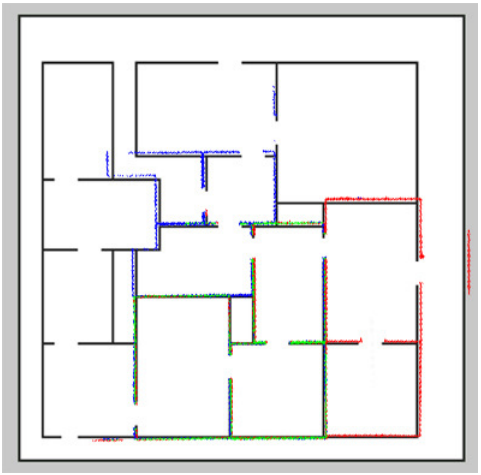
(a) Mapa do ambiente 10.



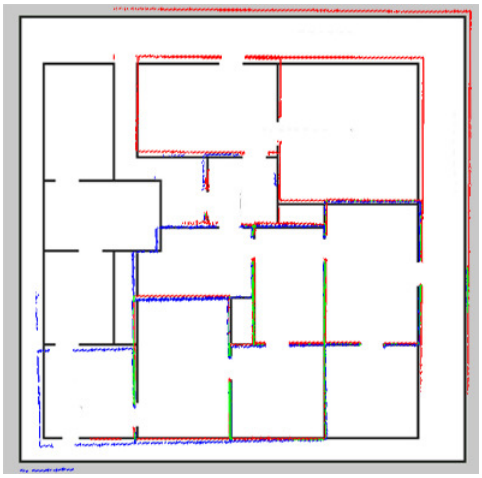
(b) Casamento dos mapeamentos 1 e 5.



(c) Casamento dos mapeamentos 2 e 3.



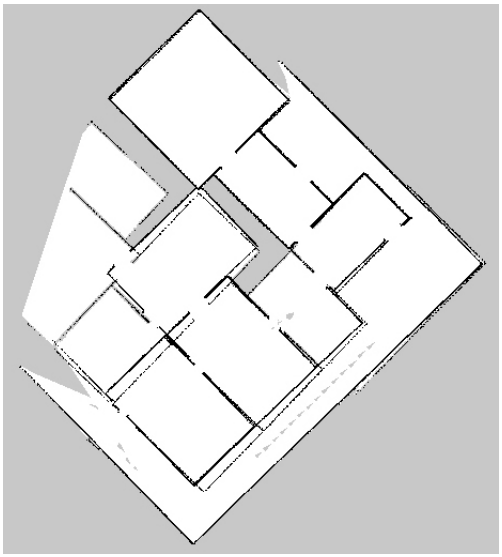
(d) Casamento dos mapeamentos 4 e 5.



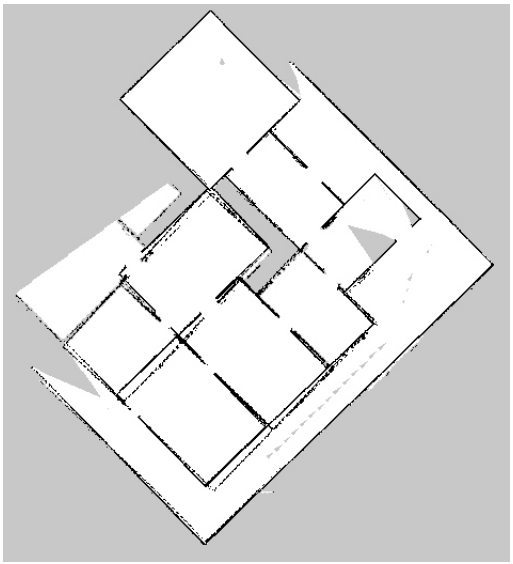
Observa-se nos locais de casamento (cor verde), que a precisão nessas regiões é maior e também existe uma grande quantidade de similaridades entre os mapas (distâncias entre cantos e linhas sobrepostas de um mapa para o outro). Essa é uma característica predominante nos casos de sucesso, pois quanto mais próximas as regiões, menor as imprecisões, enquanto à medida que uma região distancia-se de outra, as imprecisões do processo de mapeamento acentuam-se. Por este motivo, quanto mais densa de informações for uma determinada região, cujo tamanho permita limites aceitáveis de imprecisões, maiores as chances de casamento naquela região.

Figura 36 – Casamentos com mais de dois mapeamentos referente ao ambiente 2.

(a) Casamento dos mapeamentos 1, 2 e 4.



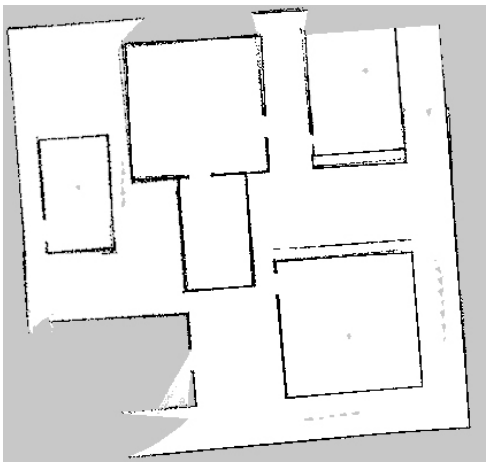
(b) Casamento dos mapeamentos 1, 2, 3, 5 e 4.



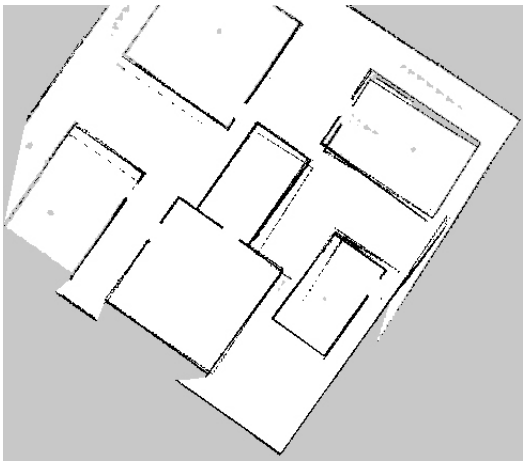
O mesmo processo foi aplicado a resultados bem sucedidos de casamento (entre dois mapeamentos diferentes do mesmo ambiente), de forma que esses resultados foram casados com outros mapeamentos, sempre realizando o casamento dois a dois e assim proporcionando um casamento entre três ou mais mapeamentos. Assim, alguns casos de sucesso podem ser observados nas Figuras 36 e 37. Além disso, exemplos de falhas no casamento com múltiplos robôs são apresentados na Seção 4.2.

Figura 37 – Casamentos com mais de dois mapeamentos referente ao ambiente 4.

(a) Casamento dos mapeamentos 1, 2, 4 e 5.



(b) Casamento dos mapeamentos 1, 2, 3, 4 e 5.



Foi realizada uma análise quantitativa com os casos de sucesso. Nessa análise foi feita a comparação dos ambientes originais com os resultados dos casamentos das quais extraiu-se o número $P_p$ de *pixels* pretos e o número $P_b$ de *pixels* brancos, que coincidiram (preto com preto, branco com branco) entre o ambiente original e o resultado de casamento. Contabilizou-se também o número $P_{pc}$ de *pixels* pretos e o número $P_{bc}$ de *pixels* brancos presentes no resultados dos casamentos. Dessa forma, para cada resultado de casamento, dividiu-se a soma dos *pixels* coincidentes $P_b$ e $P_p$ pela soma dos *pixels* brancos $P_{bc}$ e pretos $P_{pc}$ no resultado do casamento, conforme expresso na Equação 4.1. A partir desta fórmula, obtém-se a taxa de precisão entre o resultado do casamento e o ambiente original.

$$\frac{(Pp + P_b)}{(P_{pc} + P_{bc})} \tag{4.1}$$

Foi calculada a precisão de cada mapeamento individual. Este cálculo foi realizado utilizando a Equação 4.1, com a pequena alteração de que os números $P_p$ e $P_b$ representaram os valores em relação ao mapeamento individual, em vez de resultados de casamento. Desta forma, utilizaram-se ambas precisões de mapeamentos individuais e resultados de casamentos para calcular a médias, mediana e desvio padrão. Estes cálculos foram realizados para duas classificações, a primeira representando apenas os

mapeamentos individuais (portanto representando os mapas gerados via gmapping) e a segunda representando os resultados dos casamentos.

Tabela 1 – Resultado quantitativo

| Análises | Mapeamentos via Gmapping | Resultados dos Casamentos |
|---|---|---|
| Média | 0,929006 | 0,927027 |
| Mediana | 0,926777 | 0,931214 |
| Desvio Padrão | 0,020205 | 0,021912 |

Os resultados quantitativos dos casamentos e dos mapeamentos podem ser observados na Tabela 1, em que na primeira coluna, explicitam-se os tipos de análises utilizadas, na linha dois da tabela, têm-se os resultados das médias das precisões, na linha três, têm-se os resultados das medianas das precisões e, na linha quatro, os resultados dos desvios padrão das precisões, onde a coluna dois representa os resultados para os mapeamentos individuais via gmapping e a coluna três representa os resultados de casamento do método proposto neste trabalho. Nestes resultados, observa-se que as diferenças de média, mediana e desvio padrão foram muito pequenos, sendo estas inferior a $0,01$, comprovando que o método de casamento proporciona resultados aproximados ao método de mapeamento individual gmapping. Além disso, em razão das médias e medianas de precisões serem consideravelmente altas (acima de 92%), os mapas gerados são muito semelhantes ao ambiente original e, portanto, os mapas podem ser considerados navegáveis.

## 4.2 Casos de falhas no casamento

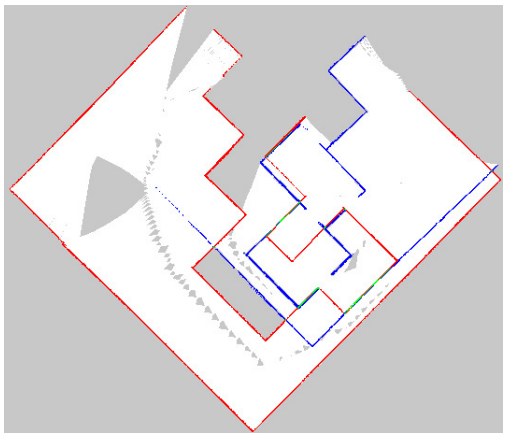
Nos casos onde os casamentos falharam, observou-se uma grande quantidade de erros nos mapas, erros esses decorrentes da derrapagem das rodas do robô e do erro de medição do sensor (foi atribuído o erro de 1 (um) centímetro no processo de mapeamento). Além disso, observou-se também o excesso de simetrias[1] indevidas entre regiões onde não deveria ocorrer o casamento. Outra causa da falha no casamento dos mapas é um tamanho pequeno na área de casamento e/ou baixa densidade de cantos concentrados nessa área.

[1] Objetos com características semelhantes, distância entre cantos e abertura dos ângulos, normalmente caracterizados por salas de tamanho semelhante.

As causas dos erros podem ser observadas claramente nas Figuras 38, 39, 40 e 41, nas quais as linhas verdes, que representam os mapeamentos semelhantes entre os mapeamentos vermelho e o azul, estão dispostas exatamente nos locais onde ocorrem excesso de simetrias entre regiões diferentes e não há uma intensificação na quantidade de cantos, de forma a superar o erro proporcionado por essas simetrias.

Figura 38 – Representação de dois casos de falha de casamentos para o ambiente 3.

(a) Falha com os mapeamentos 3 e 4.



(b) Falha com os mapeamentos 3 e 5.

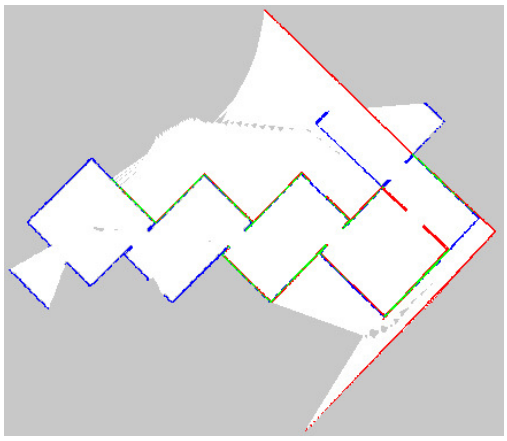
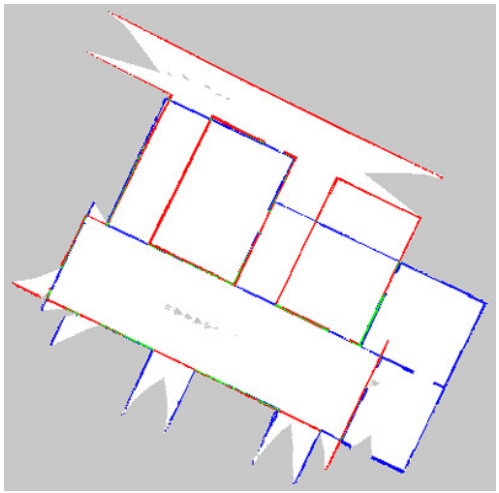


Figura 39 – Representação de dois casos de falha de casamentos para o ambiente 5.

(a) Falha com os mapeamentos 1 e 2.



(b) Falha com os mapeamentos 1 e 3.

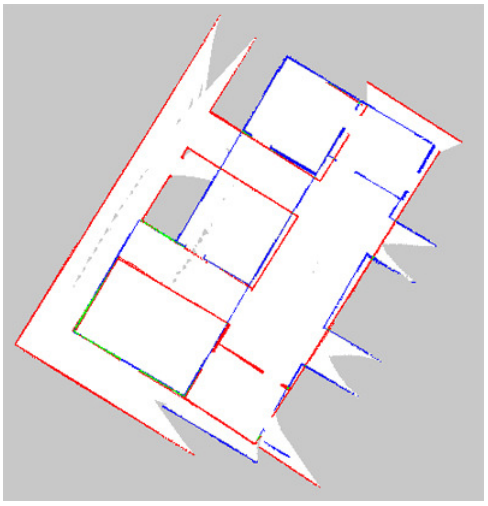
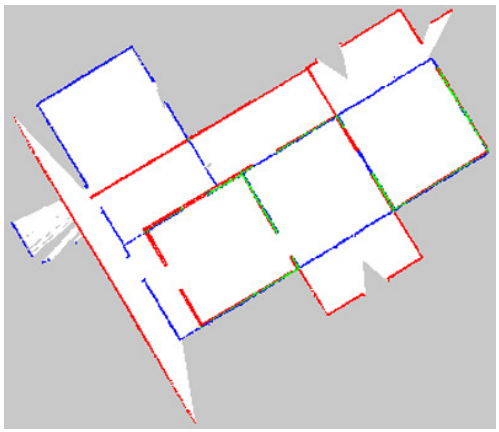


Figura 40 – Representação de dois casos de falha de casamentos para o ambiente 7.

(a) Falha com os mapeamentos 2 e 3.



(b) Falha com os mapeamentos 2 e 4.

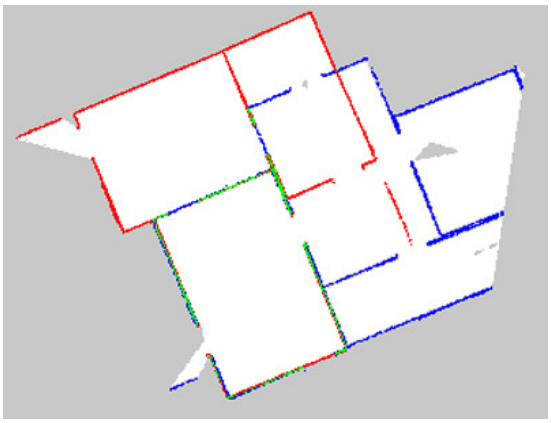
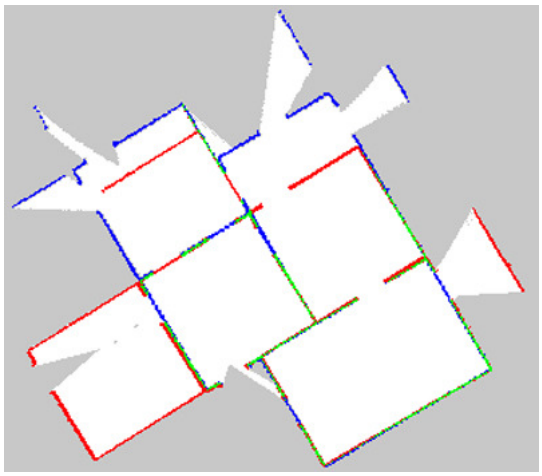


Figura 41 – Representação de dois casos de falha de casamentos para o ambiente 8.

(a) Falha com os mapeamentos 1 e 2.

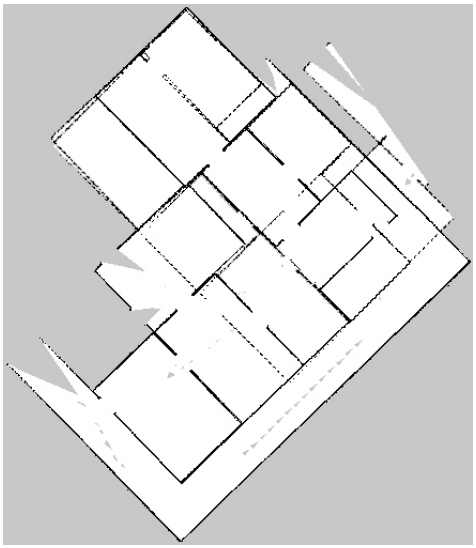
(b) Falha com os mapeamentos 3 e 4.



Visto que ocorreram falhas nos casamentos entre dois mapeamentos, esperava-se que falhas também ocorressem com casamentos com três ou mais mapeamentos e estes podem ser observados nas Figuras 42 e 43 . Porém esses erros concentraram-se em combinações onde os mapeamentos envolvidos no casamento já haviam gerado falhas para casamento com dois mapas.

Figura 42 – Casamentos com mais de dois mapeamentos com falhas para o ambiente 2.

(a) Falha com os mapeamentos 1, 3 e 4.



(b) Falha com os mapeamentos 1, 4 e 5.

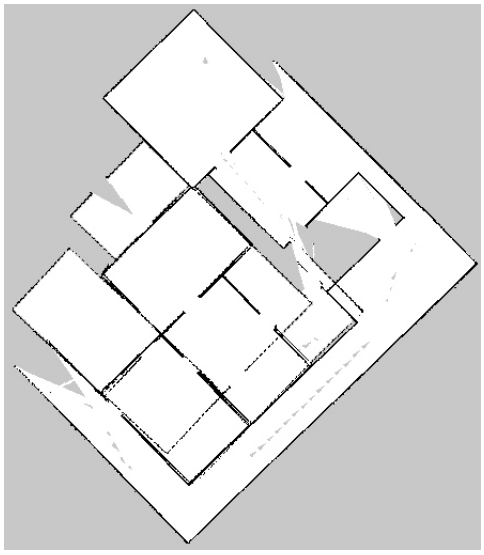
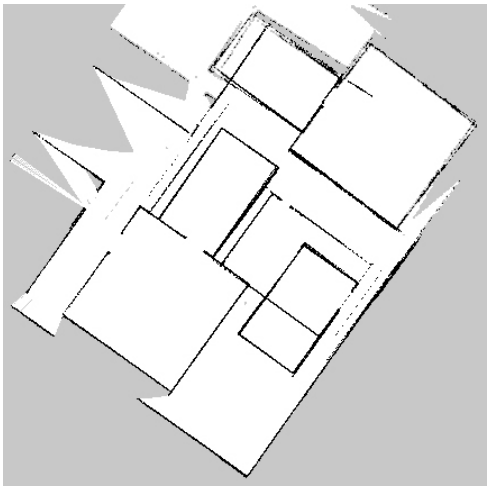
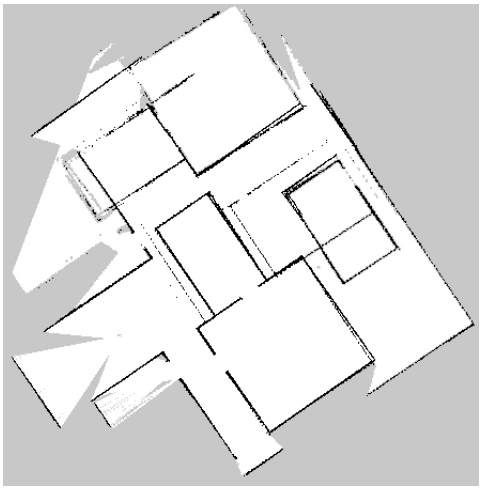


Figura 43 – Casamentos com mais de dois mapeamentos com falhas para o ambiente 4.

(a) Falha com os mapeamentos 1, 3 e 4.



(b) Falha com os mapeamentos 1, 3, 4 e 5.



## 4.3 Análise de sucessos e falhas

Através da análise dos mapas casados, tanto no casos de sucesso quanto nos erros, os principais fatores observados que influenciaram no resultado foram:

- Tamanho das áreas em comum mapeada entre os robôs: implica uma maior quantidade de características a serem exploradas e comparadas;
- Concentração de cantos nas regiões em comum: permite uma geração de alfabetos mais detalhados e ricos em relacionamentos, dessa forma a quantidade de casamentos entre os membros do alfabeto que estão nesta região será mais elevada;
- Simetrias indevidas entre regiões não correspondentes entre os mapas: fazem com que regiões diferentes entre os mapas possam também obter casamentos, possivelmente provocando falhas no processo;
- Erros nas distâncias e/ou nos ângulos das regiões superiores aos parâmetros permitidos: fazem com que as características não sejam relacionadas entre si no processo de geração do alfabeto, fazendo com que seus membros sejam pobres em características e, portanto, proporcionando casamentos mais fracos;
- Parametrização do método: com uma boa parametrização, é possível suprir uma parte dos problemas causados pelos itens acima.

# 5 Conclusão

Neste trabalho foi realizado casamento de mapas para SLAM com múltiplos robôs. Esses casamentos foram realizados de dois em dois mapas. Para possibilitar os casamentos, foi desenvolvida uma técnica de relacionamento geométrico entre os cantos que foram detectados nos mapeamentos. Essa técnica detecta três candidatos para cada um dos dois pontos candidatos a encaixe entre os mapas, após essa detecção realiza-se a tentativa de casamento entre esses candidatos e então os mapas gerados são avaliados através de uma métrica de similaridade. Através deste método, obtiveram-se bons resultados, de forma a viabilizar futuros testes em robôs reais e a continuidade da pesquisa.

O processo de relacionamento geométrico entre os cantos, estabelecendo uma distância limite para que esse relacionamento seja realizado, é interessante pois geram elementos que individualizam e caracterizam esses cantos como únicos, desta forma, pode-se compará-los através dessas individualidades geradas. Essas individualidades podem ser semelhantes entre regiões similares, porém não correspondentes. Dessa maneira, a fim de gerar diferenças entre as características, uma exploração mais completa entre as áreas em comum dos mapas sendo casadas é necessária, tanto para gerar semelhantes entre cantos equivalentes, quanto para gerar diferenças entre cantos não equivalentes.

Comparar o conjunto das individualidades de um canto pode não ser suficiente para um determinado instante dos mapeamentos, pois não há garantias de que áreas em comum entre os mapas já tenham sido mapeadas ou, caso haja essas áreas em comum, se elas possuem quantidade de cantos suficientes que permita que tanto as características de semelhança, quanto as características que diferenciam as similaridades tenham sido destacadas. Dessa forma, fez-se necessária a adição de uma verificação de encaixe entre os cantos com melhores casamentos no dado instante do mapeamento, que foi a aplicação da métrica de similaridade de Jaccard, o que acaba suprindo parte dos problemas que surgem em casos que a quantidade de cantos nas áreas em comum é pouca ou muito similar a áreas não equivalentes.

Pela composição do método através de duas etapas, uma que sugere os cantos promissores para casamento e uma de confirmação, tornou o método em questão bastante promissor, promovendo o casamento de mapas até mesmo em situações de intenso erros

oriundos do processo de mapeamento dos ambientes. Também foi comprovado que o método é capaz de realizar o casamento com mais de dois mapas, obtendo casos de sucesso em casamento com cinco mapas. O tempo de processamento com dois mapas foi eficiente, gerando a possibilidade de aplicação do método em *online* SLAM em tempo real, porém acima de dois mapas é possível que o tempo de processamento aumente de forma a inviabilizar sua aplicação em tempo real e sua aplicação seja possível apenas após o processo de mapeamento e dessa forma seria melhor aplicado nos mapeamentos resultados de *full* SLAM.

A análise quantitativa dos casamentos mostrou que a diferença de precisão entre o mapeamento apenas com um robô e o resultado do casamento não possuem diferenças consideráveis. Assim, o casamento pode ser considerado uma ferramenta viável para processos de mapeamentos com mais de um robô.

A qualidade do casamento foi altamente influenciada pela quantidade de áreas em comum mapeadas pelos robôs envolvidos e pela quantidade de erros nessas áreas especificas dos mapas.

A parametrização do método mostrou-se de grande importância, merecendo um estudo paralelo para melhoria na eficiência do método. Os principais parâmetros que influenciaram no problema foram os erros aceitáveis entre distâncias e ângulos, a quantidade máxima de cantos a serem detectados (a quantidade influencia diretamente no tempo de processamento), a qualidade dos cantos detectados e distâncias mínima e máxima para os cantos serem relacionados.

## 5.1 Trabalhos futuros

Como continuidade do trabalho, há a possibilidade de mesclar o método proposto com técnicas que extraem padrões de características além dos cantos, durante o processo de mapeamento, como cores e textura, para isso seria melhor aplicada com Visual SLAM, que utiliza processamento de imagens adquiridas por vídeo para gerar os mapas.

Um estudo mais apurado na parametrização do método poderia melhorar o desempenho deste, de forma que aplicação de técnicas que pudessem gerar melhores parâmetros seria de grande relevância para o casamento dos mapas.

O casamento dos mapas costuma somar as deformações sem suavizá-las ou

corrigi-las. Desta forma, encontrar um modo de reduzir as deformações entre os mapeamentos ao casar os mapas, tornaria o método ainda mais robusto e confiável. E assim, suavizariam as deformações sem perder a navegabilidade do mapa.

## Referências

BIRK, A.; CARPIN, S. Merging occupancy grid maps from multiple robots. *Proceedings of the IEEE*, v. 94, n. 7, p. 1384–1397, July 2006. ISSN 0018-9219. 18

BLANCO, J.-L.; GONZALEZ, J.; FERNANDEZ-MADRIGAL, J.-A. A new method for robust and efficient occupancy grid-map matching. In: MARTÍ, J.; BENEDÍ, J. M.; MENDONÇA, A. M.; SERRAT, J. (Ed.). *Pattern Recognition and Image Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 194–201. ISBN 978-3-540-72849-8. 17

BLANCO, J.-L.; GONZáLEZ-JIMéNEZ, J.; FERNáNDEZ-MADRIGAL, J.-A. A robust, multi-hypothesis approach to matching occupancy grid maps. *Robotica*, Cambridge University Press, v. 31, n. 5, p. 687–701, 2013. 18

BRESSON, G.; AUFRERE, R.; CHAPUIS, R. Consistent multi-robot decentralized SLAM with unknown initial positions. *Information Fusion (FUSION), 2013*, p. 372–379, 2013. 16

CUNNINGHAM, A.; WURM, K. M.; BURGARD, W.; DELLAERT, F. Fully distributed scalable smoothing and mapping with robust multi-robot data association. *Proceedings - IEEE International Conference on Robotics and Automation*, p. 1093–1100, 2012. ISSN 10504729. 16

DANTAS, D. d. O. *Implementação de um Sistema Autônomo de Construção de Estrutura usando Aprendizado por Reforço*. Dissertação (Mestrado), 2017. DEPARTAMENTO DE INFORMÁTICA/CCET. Disponível em: <https://tedebc.ufma.br/jspui/handle/tede/tede/2062>. 51, 52, 53, 54, 57, 58, 59

DEY, N.; NANDI, P.; BARMAN, N.; DAS, D.; CHAKRABORTY, S. A comparative study between moravec and harris corner detection of noisy images using adaptive wavelet thresholding technique. *arXiv preprint arXiv:1209.1558*, 2012. 26

ELFES, A. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. Tese (Doutorado), Pittsburgh, PA, USA, 1989. AAI9006205. 31

FARIA, D. R. *Reconhecimento de impressões digitais com baixo custo computacional para um sistema de controle de acesso*. Dissertação (Mestrado), 2005. 25

FRESE, U.; WAGNER, R.; RÖFER, T. A slam overview from a user's perspective. *KI - Künstliche Intelligenz*, v. 24, n. 3, p. 191–198, Sep 2010. ISSN 1610-1987. Disponível em: <https://doi.org/10.1007/s13218-010-0040-4>. 57

GONZALEZ, R. C.; WOODS, R. E. *Processamento digital de imagens* . [S.l.]: Pearson Prentice Hall, 2010. 23, 24

GOUVEIA, B. D. *Exploração de ambientes desconhecidos com Clusters Robóticos*. Dissertação (Mestrado), 2013. Disponível em: <https://estudogeral.sib.uc.pt/handle/10316/35582>. 55, 56, 57

GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans Robot 23*, 2007. Disponível em: <https://openslam-org.github.io/gmapping.html>. 55

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: CITESEER. *Alvey vision conference*. [S.l.], 1988. v. 15, p. 50. 26, 27, 38

JACCARD, P. The distribution of the flora in the alpine zone. *New phytologist*, Wiley Online Library, v. 11, n. 2, p. 37–50, 1912. 29

JAFRI, S. R. u. N.; CHELLALI, R. A distributed multi robot slam system for environment learning. In: *2013 IEEE Workshop on Robotic Intelligence in Informationally Structured Space (RiiSS)*. [S.l.: s.n.], 2013. p. 82–88. 17

JELINEK, L. *Vrep map generator*. 2016. Accesso: 15-03-2018. Disponível em: <https://github.com/Lukx19/Vrep-map-generator>. 51, 59, 60, 61

JIAN, L.; CHEN, Z.; QIANG, I.; HENG, W.; MANZHEN, M. Vision feature extraction algorithm for occupancy grid maps merging. In: *Proceedings of the 2017 2Nd International Conference on Communication and Information Systems*. New York, NY, USA: ACM, 2017. (ICCIS 2017), p. 290–293. ISBN 978-1-4503-5348-9. Disponível em: <http://doi.acm.org/10.1145/3158233.3159372>. 17

JUNIOR, D. S.; PINTO, F. d. A.; GOMIDE, R. L.; TEIXEIRA, M. M. Avaliação de métodos automáticos de limiarização para imagens de plantas de milho atacadas por spodoptera frugiperda. *Embrapa Milho e Sorgo-Artigo em periódico indexado (ALICE)*, SciELO Brasil, 2003. 21

KOCH, P.; MAY, S.; SCHMIDPETER, M.; KÜHN, M.; PFITZNER, C.; MERKL, C.; KOCH, R.; FEES, M.; MARTIN, J.; AMMON, D.; NÜCHTER, A. Multi-robot localization and mapping based on signed distance functions. *Journal of Intelligent & Robotic Systems*, v. 83, n. 3, p. 409–428, Sep 2016. ISSN 1573-0409. Disponível em: <https://doi.org/10.1007/s10846-016-0375-7>. 16

LIU, J.; JAKAS, A.; AL-OBAIDI, A.; LIU, Y. A comparative study of different corner detection methods. In: IEEE. *Computational Intelligence in Robotics and Automation (CIRA), 2009 IEEE International Symposium on*. [S.l.], 2009. p. 509–514. 25, 26

LOPES, F. M. Um modelo perceptivo de limiarização de imagens digitais. 2012. 20

MEDEIROS, N. das G.; SILVA, E. A. da; NOGUEIRA, J. R. Segmentação morfológica de imagens utilizando o gradiente morfológico multi-escala. *Revista Brasileira de Cartografia*, n. 54, 2002. 21, 22, 23

MILSZTAJN, F.; GEUS, K. de. Segmentação de tecidos cerebrais em imagens tridimensionais do cérebro. 2015. Disponível em: <https://www.researchgate.net/publication/255662964_Segmentacao_de_Tecidos_Cerebrais_em_Imagens_Tridimensionais_do_Cerebro>. Acesso em: 2018-05-14. 29

OPENCV. *Shi-Tomasi Corner Detector & Good Features to Track*. 201? Disponível em: <https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_shi_tomasi/py_shi_tomasi.html>. Acesso em: 2018-05-14. 27, 28

PATTAR, S. Study of corner detection algorithms and evaluation methods. *Int. J. Innov. Res. Sci*, v. 4, n. 5, 2015. 25, 26, 27

ROS. *Documentation = <http://wiki.ros.org/>, note = Accessed: 2017-04-09,*. 201? 51

SHI, J.; TOMASI, C. Good features to track. In: *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 1994. p. 593–600. ISSN 1063-6919. 27, 37

SOUZA, J. C. *Diagnóstico de câncer de mama a partir de imagens de mamografia 2d utilizando descritores de forma 3d*. Dissertação (Mestrado), 2018. Disponível em: <https://tedebc.ufma.br/jspui/handle/tede/tede/2140>. 24

THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic robotics*. [S.l.]: MIT press, 2005. 57

THRUN, S.; LIU, Y. Multi-robot SLAM with Sparse Extended Information Filers. *The International Journal of Robotics Research*, v. 15, p. 254–266, 2005. ISSN 16107438. 16

VENTURI, J. J. *Álgebra Vetorial e Geometria Analítica*. [S.l.]: Livrarias Curitiba, 2015. 29, 30

ZHANG, Z.; DERICHE, R.; FAUGERAS, O.; LUONG, Q.-T. *A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry*. Elsevier, 1995. v. 78. 87-119 p. Disponível em: <https://www.microsoft.com/en-us/research/publication/robust-technique-matching-two-uncalibrated-images-recovery-unknown-epipolar-geometry/>. 18