Victor Terra Ferrão

# Novo algoritmo para fusão de mapas invariantes a escala, rotação e translação.

Brasil

2018

sistema de bibliotecas ufg

PRPG
PRÓ-REITORIA DE PÓS-GRADUAÇÃO

UFG

# TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

**1. Identificação do material bibliográfico:** **[X] Dissertação** **[ ] Tese**

**2. Identificação da Tese ou Dissertação:**

Nome completo do autor: Victor Terra Ferrão

Título do trabalho: NOVO ALGORITMO PARA FUSÃO DE MAPAS INVARIANTES A ESCALA, ROTAÇÃO E TRANSLAÇÃO

**3. Informações de acesso ao documento:**

Concorda com a liberação total do documento [ X ] SIM [ ] NÃO[1]

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF da tese ou dissertação.

Victor Terra Ferrão

Assinatura do(a) autor(a)[2]

Ciente e de acordo:

Assinatura do(a) orientador(a)[2]

Data: 19 / 10 / 18

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Os dados do documento não serão disponibilizados durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

[2] A assinatura deve ser escaneada.

Victor Terra Ferrão

# Novo algoritmo para fusão de mapas invariantes a escala, rotação e translação.

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia de Computação.

Universidade Federal de Goiás – UFG-GO

Escola de Engenharia Elétrica, Mecânica e Computação

Programa de Pós-Graduação

Orientador: Gelson da Cruz Júnior

Coorientador: Cássio Dener Noronha Vinhal

Brasil

2018

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Terra Ferrão, Victor
Novo Algoritmo para fusão de mapas invariantes a escala, rotação e translação [manuscrito] / Victor Terra Ferrão. - 2018.
63 f.: il.

Orientador: Prof. Gelson da Cruz Junior; co-orientador Cássio Dener Noronha Vinhal.
Dissertação (Mestrado) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Goiânia, 2018.
Bibliografia.
Inclui siglas, gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. Fusão de Mapas. 2. Mapeamento cooperativo. 3. ROS. 4. OpenCV. 5. Robôs autônomos. I. da Cruz Junior, Gelson, orient. II. Título.

CDU 004.92

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO
COORDENAÇÃO DE PÓS-GRADUAÇÃO

# Ata de Dissertação de Mestrado

Ata da sessão de julgamento da Dissertação de Mestrado em Engenharia Elétrica e de Computação, área de concentração Engenharia de Computação, do candidato **Victor Terra Ferrão,** realizada em 28 de setembro de 2018.

Aos vinte e oito dias do mês de setembro de dois mil e dezoito, às 14:00 horas, na na Sala Caryocar Brasiliensis, Bloco A da Escola de Engenharia Elétrica e de Computação (EMC), Universidade Federal de Goiás (UFG), reuniram-se os seguintes membros da Comissão Examinadora designada pela Coordenadoria do Programa de Pós-graduação em Engenharia Elétrica e de Computação: Os Doutores , Gelson da Cruz Junior – Orientador (EMC/UFG), Fabrizzio Alphonsus Alves de Melo Nunes Soares – INF/UFG, Leonardo da Cunha Brito – EMC/UFG e Cássio Dener Noronha Vinhal – EMC/UFG, para julgar a Dissertação de Mestrado de **Victor Terra Ferrão,** intitulada **"Novo algoritmo para fusão de mapas invariantes a escala, rotação e translação."**, apresentada pelo Candidato como parte dos requisitos necessários à obtenção do grau de Mestre, em conformidade com a regulamentação em vigor. O Professor Doutor Gelson da Cruz Junior da Comissão, abriu a sessão e apresentou o candidato que discorreu sobre seu trabalho, após o que, foi arguido pelos membros da Comissão na seguinte ordem: Fabrizzio Alphonsus Alves de Melo Nunes Soares, Leonardo da Cunha Brito e Cássio Dener Noronha Vinhal. A parte pública da sessão foi então encerrada e a Comissão Examinadora reuniu-se em sessão reservada para deliberar. A Comissão julgou então que o Candidato, tendo demonstrado conhecimento suficiente, capacidade de sistematização e argumentação sobre o tema de sua Dissertação, foi considerado **aprovado** e deve satisfazer as exigências listadas na Folha de Modificação de Dissertação de Mestrado, em anexo a esta Ata, no prazo máximo de 60 dias, ficando o professor orientador responsável por atestar o cumprimento dessas exigências. Os membros da Comissão Examinadora descreveram as justificativas para tal avaliação em suas respectivas Folhas de Avaliação, anexas a esta Ata. Nada mais havendo a tratar, o presidente da Comissão declarou encerrada a sessão. Nos termos do Regulamento Geral dos Cursos de Pós-graduação desta Universidade, a presente Ata foi lavrada, lida e, julgada conforme, segue assinada pelos membros da Comissão supracitados e pelo Candidato. Goiânia, 28 de setembro de 2018.

**Comissão Examinadora Designada:**

Gelson da Cruz Junior – Orientador (EMC/UFG) (Avaliação: APROVADO)

Fabrizzio Alphonsus Alves de Melo Nunes Soares – INF/UFG (Avaliação: Aprovado)

Leonardo da Cunha Brito – EMC/UFG (Avaliação: Aprovado)

Cássio Dener Noronha Vinhal – EMC/UFG (Avaliação: APROVADO)

Candidato:

**Victor Terra Ferrão**

*Este trabalho é dedicado a todas as pessoas que*
*me apoiaram durante essa grande jornada que chamo de vida.*

*"As pessoas comumente têm medo*
*da mudança porque temem o desconhecido.*
*Mas a única grande constante*
*da história é que tudo muda."*
*(Harari, Y.N. in Homo Deus)*

# Resumo

É apresentado um novo algoritmo que trabalha em cima do problema de fusão de mapas de ocupação obtidos por diferentes agentes autônomos e todos os passos realizados para o desenvolvimento deste. Este novo método consegue combinar mapas, invariantemente à escala, rotação e translação.

Durante cada etapa do desenvolvimento, ele foi comparado diretamente com agorítmos pré-existentes utilizando ambos conjunto de dados e ambiente simulado.

São mostradas tabelas e imagens comparativas entre os algoritmos e versões do algoritmo proposto em cada um dos testes realizados. O novo algoritmo se mostrou mais rápido e também mais robusto, o que é mostrado de forma evolutiva de modo a melhor compreender as modificações introduzidas.

**Palavras-chaves**: Fusão de Mapas, Mapeamento cooperativo, ROS, OpenCV, Robôs autônomos.

# Abstract

This work presents a new algorithm that works on the problem of merging occupation maps obtained by different autonomous agents and all the steps taken to develop it. This new method can combine maps, invariant to scale, rotation and translation.

During each stage of development, it was compared directly with pre-existing agorithms using both data set and simulated environment.

Comparative tables and images are shown between algorithms and versions of the proposed algorithm in each of the tests performed. The new algorithm demonstrates to be faster and also more robust. Finally, the present text tries to show this evolutionarily so as to understand better the modifications introduced.

**Key-words**: Map Merging, Cooperative mapping, ROS, OpenCV, autonomous robots.

# Lista de ilustrações

# Lista de tabelas

# Lista de Algoritmos

# Lista de abreviaturas e siglas

| | |
|---|---|
| SIFT | *Scale-Invariant Feature Transform* |
| SLAM | *Simultaneous Localization and Mapping* |
| LiDaR | *Light Detection and Ranging* |
| ROS | *Robot Operating System* |
| OpenCV | *Open Source Computer Vision Library* |
| DNIFM | *Deterministic Non-Iterative Fast Method* |
| SM | *Sift Merge* |
| $SM^2$ | *Sift Merge* Versão 2 |
| IMU | Unidade de Medição Inercial |
| GPS | *Global Positioning System* |
| BSD | *Berkeley Software Distribution* |

# Sumário

# Introdução

A robótica móvel está se tornando cada vez mais acessível e robusta com a evolução da tecnologia. O aumento em poder de processamento e novos sensores permitem maior quantidade de projetos e aplicações em robótica móvel (SICILIANO; KHATIB, 2016), (PARKER, 2000). Devido a redução de custo, tarefas que antes só eram feitas por apenas um robô, podem ser divididas entre vários para reduzir o tempo da tarefa e aumentar a confiabilidade (MARJOVI; MARQUES, 2013). No entanto, vários desafios precisam ser contornados para que a coordenação entre vários robôs seja feita (FOX et al., 2006). Um dos desafios é integrar a informação coletada entre os robôs em um mapa global.

Existem diferentes modos de modelar o ambiente do o robô (THRUN, 1998). Mapas de grade de ocupação é uma das escolhas mais comuns pela facilidade de correspondência entre ambiente real e o mapa gerado.

SLAM é uma área de pesquisa já bem estabelecida, mas a maioria dos esforços de pesquisa se concentrou na construção de um mapa único, mesmo quando vários robôs são considerados (ERINC; CARPIN, 2014). Quando se usa robôs heterogêneos, a combinação de mapas se torna um problema maior, já que estes podem ter diferentes restrições, isto é, capacidade de processamento limitada, memória, precisão e tipo do sensor ou capacidade de comunicação. Em um mesmo ambiente podem existir robôs caros com alta capacidade de processamento utilizando LiDaR de alta resolução (BATALIN; SUKHATME; HATTIG, 2004), e ao mesmo tempo alternativas de baixo custo podem usar câmeras estéreo (MURRAY; LITTLE, 2000) ou sonares (ELFES, 1987) para produzir um mapa.

Aqui presentamos a idéia e o desenvolvimento por trás do algoritmo para mesclar mapas de grade de ocupação produzidos por vários agentes robóticos que exploram diferentes partes de um ambiente. O algoritmo usa uma transformação conhecida como SIFT (*Scale Invariant Feature Transform*) (LINDEBERG, 2012) para detectar pontos de interesse (*key-points*) entre dois mapas e depois calcular as transformações para mesclá-los. O algoritmo proposto não faz uso da posição e orientação inicial de cada robô. É possível combinar mapas com diferentes rotações, translações e escalas sem saber qualquer uma dessas informações de nenhum dos robôs. O resultado é um mapa global com informações dos outros mapas combinadas e a transformação necessária para passar de um para outro.

O algoritmo foi implementado e testado em mapas de diferentes conjuntos de dados públicos relacionados a ambientes internos e externos, e produz consistentemente excelentes resultados. Esse algoritmo e os testes feitos no conjunto de dados foram apresentados em (FERRAO; VINHAL; JR, 2017), onde foram feitas comparações com o método desenvolvido por Carpin (CARPIN, 2008). Aqui também será apresentada uma nova versão mais robusta

e mais rápida do nosso algoritmo. Nessa nova versão, foram feitos testes e comparações em ambos conjunto de dados e ambiente simulado.

# 1 Preparação da pesquisa

## 1.1 Tipos de Mapas

Quando se está desenvolvendo robôs autônomos, deve-se preocupar com qual a representação do ambiente que vai ser utilizada. As representações mais utilizadas são *Grid Maps* (Mapas de grade) e *Topological Maps* (Mapas topológicos) (THRUN, 1998).

### 1.1.1 *Grid Maps*

São representados por meio de células, em que cada célula representa uma área física, e o seu valor nos diz a probabilidade de existir um obstáculo nesta região correspondente. A Figura 1 ilustra este tipo de mapa.

Como um *Grid Map* é uma representação física de um ambiente, este pode ser gerado diretamente por meio de sensores. Os sensores mais comuns a se utilizarem são câmeras (MURRAY; LITTLE, 2000), LiDaR (*Light Detection and Ranging* (BATALIN; SUKHATME; HATTIG, 2004), ou mesmo ultrassons (ELFES, 1987).

Os mapas utilizados aqui fazem uso deste tipo de mapa, já que esses mapas podem ser facilmente traduzidos para imagens.

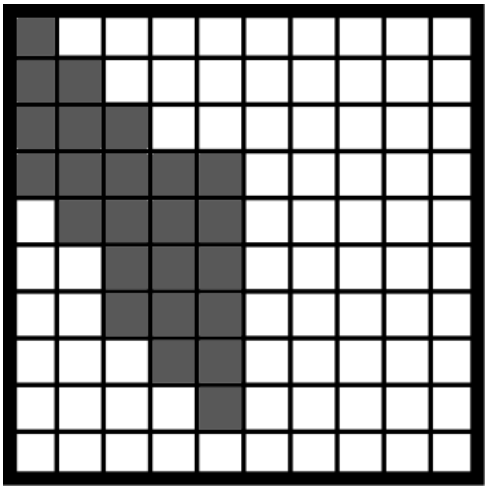

Figura 1 – Exemplo de *Grid Map*. As células em cinza indicam espaços onde o ambiente está ocupado por, e as em branco indicam os espaços livres.

### 1.1.2 Mapas topológicos

Geralmente são representados por meio de linhas e vértices. Neste tipo de mapa, a escala, distância e direção do ambiente é abstraído, e pode não representar os valores

corretos do ambiente, no entanto, o relacionamento entre os vértices são mantidos. Com esse tipo de mapa, fazer planejamentos em alto nível fica mais fácil, porque somente informações essenciais são informadas. A Figura 2 ilustra um exemplo de mapa topológico.

Como os mapas topológicos não são uma representação física do ambiente, mas sim uma abstração, este não pode ser gerado diretamente por meio de sensores. Processamento e detecção de características são necessárias para a geração do mapa.


Figura 2 – Linhas de metrô são exemplos de Mapas topológicos.

Disponível em <http://mapa-metro.com/en/egypt/cairo/cairo-metro-map.htm>, acessado em 1 de Setembro de 2018.

## 1.2 Localização e Mapeamento Simultâneos (SLAM)

Localização e Mapeamento Simuntâneos, normalmente conhecido como SLAM, consiste em construir um mapa enquanto estima sua posição enquanto está se movimentando por esse mapa (CADENA et al., 2016). Métodos que tentam resolver esse problema incluem filtro de partículas (ARULAMPALAM et al., 2002), filtro de Kalman (BROWN; HWANG et al., 1992), e GraphSLAM (THRUN; MONTEMERLO, 2006).

A precisão do processo de mapeamento é muito dependente dos sensores disponíveis. Geralmente quanto maior a quantidade e qualidade dos sensores, maior é a acurácia. Os sensores que geralmente são utilizados são IMU (Unidade de Medição Inercial), GPS (*global positioning system*), ultrassom e câmeras.

## 1.3 *Robot Operating System* (ROS)

O ROS é um sistema de gerenciamento de código aberto utilizado em robótica móvel. Ele fornece serviços de abstração de hardware, controle de dispositivo de baixo nível, mensageria entre processos e gerenciamento de pacotes, o que lembra os serviços disponíveis em um sistema operacional. Muitos trabalhos são feitos utilizando o ROS como plataforma no desenvolvimento de robôs por sua facildade de uso e extensibilidade, por exemplo, (MATÉS et al., 2010), (RUIZ et al., 2013) e (QUIGLEY; GERKEY; SMART, 2015).

Aqui o ROS foi utilizado na criação, gerenciamento de sensores, e movimentação dos robôs em um ambiente virtual. Nesse trabalho usamos Stage e o mesmo é descrito a seguir juntamente com a motivação para o seu uso.

## 1.4 Stage

Stage é um simulador de robôs. Ele fornece um mundo virtual preenchido por robôs e sensores móveis, juntamente com vários objetos para os robôs detectarem e manipularem. Seu projeto considera o paradigma de programação de sistemas multiagentes. Por isso, oferece modelos bastante simples e computacionalmente baratos de muitos dispositivos, em vez de tentar emular qualquer dispositivo com grande fidelidade (GERKEY; VAUGHAN; HOWARD, 2003).

Ao contrário de outros simuladores como o Gazebo (KOENIG; HOWARD, 2004), o Stage não simula as características físicas de um robô em um ambiente tridimensional, e.g., a simulação de corpo rígido. Em vez disso o robô é representado como um polígono em um ambiente bidimensional. Essa simplificação reduz o custo computacional, além de facilitar a prototipagem, já que não é necessário saber todas as características físicas do robô, apenas

as dimensões aproximadas da base, distância entre rodas, aceleração máxima nos eixos $X$ e $Y$, se é omnidirecional ou se possui eixo de Ackermann (MITCHELL; STANIFORTH; SCOTT, 2006). Sensores disponíveis para o robô simulado incluem IMU, sonar e LiDaR.

Devido a estas características, e pela facilidade de integração com o ROS, o Stage foi escolhido para ser utilizado nas simulações que são aqui apresentadas. As figura 3 ilustra um robô que utiliza LiDaR em um ambiente gerado pelo Stage, e a figura 4 ilustra um *swarm* de robôs cooperando em um ambiente compartilhado (VAUGHAN, 2008).

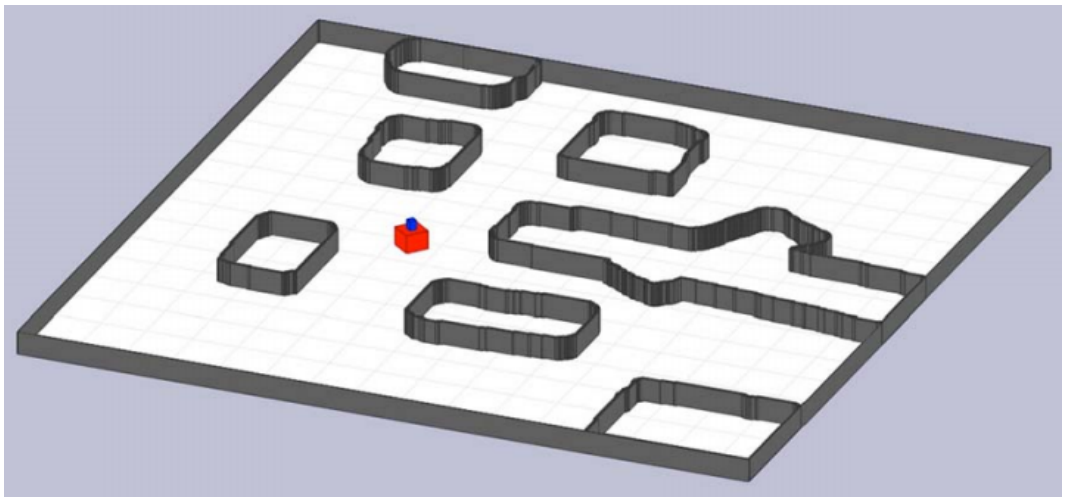

Figura 3 – Exemplo de robô e mapa gerado pelo Stage.

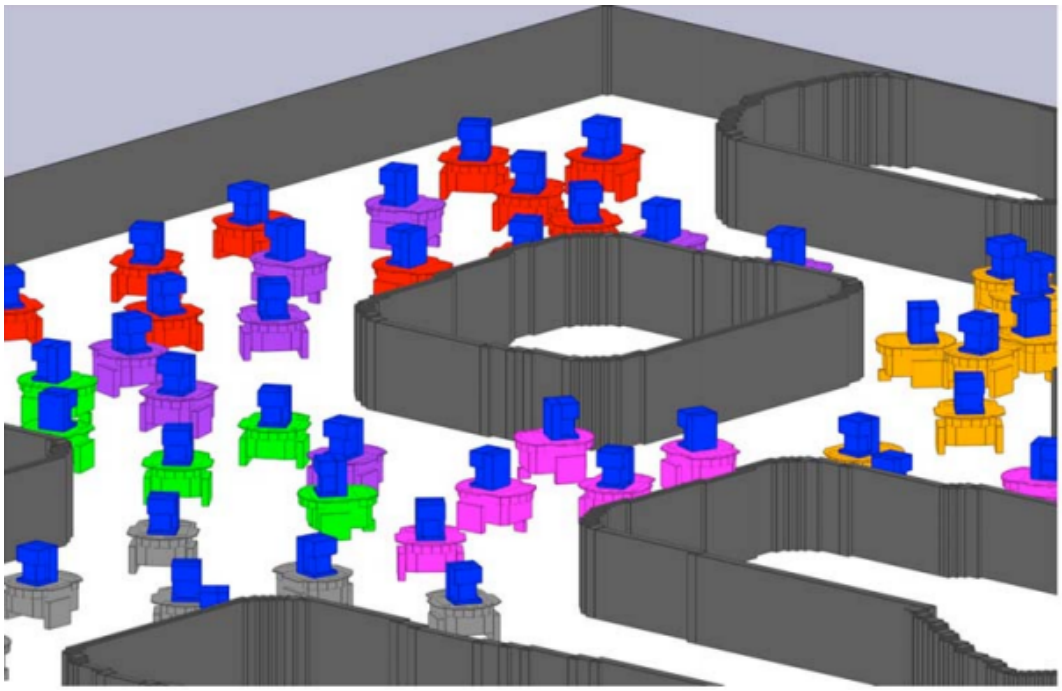

Figura 4 – Exemplo de um *swarm* de robôs gerados pelo Stage.

## 1.5 OpenCV

OpenCV é uma biblioteca de visão computacional e aprendizado de máquina. Ele é muito utilizado como *framework* base em *software* que faz uso de processamento de imagens. Devido a sua licença BSD (*Berkeley Software Distribution*) (LEHEY, 2003), ele é utilizado tanto em uso educacional quanto comercial. Os algoritmos utilizados (e.g., SIFT) e implementados nessa dissertação fazem uso dessa plataforma, já que essa já possui uma vasta quantidade de algoritmos otimizados.

## 1.6 Fusão de mapas

Existem muitas abordagens que assumem que robôs sabem sua posição em um mapa global (COOPER; DURRANT-WHYTE, 1994), (SUKKARIEH; NEBOT; DURRANT-WHYTE, 1999), (KIM et al., 2007). Com essa suposição e dado um objetivo de explorar a maior parte do mapa possível, a tarefa de coordenação é direta, ou seja, se cada robô tiver um *GPS*, o que talvez não é possível para ambientes internos, e assim os robôs ficam sem uma maneira direta de mesclar seus mapas.

Alguns trabalhos abordam a falta de um GPS verificando a posição relativa entre os robôs, confiando em encontros esporádicos e exigindo que os robôs estejam em seus campos de visão (FOX et al., 2006), (HOWARD; PARKER; SUKHATME, 2006), (ÖZKUCUR; AKIN, 2009). No entanto, isso é um empecilho, porque se os robôs podem se comunicar, mas não podem se ver, não serão capaz de fundir seus mapas. Outros trabalhos abordam esta tarefa de mesclagem sem precisar da posição relativa, realizando uma transformação que otimiza o número de células nas quais ambos os mapas se sobrepõem (CARPIN; BIRK; JUCIKAS, 2005), (MA et al., 2008).

O algoritmo apresentado originalmente por Carpin (CARPIN; BIRK; JUCIKAS, 2005) promete uma solução ótima quando o número de iterações tende ao infinito, mas torna-se inadequado para aplicações em tempo real. Um método mais rápido usa Transformada Hough e aborda o problema como um caso particular de *registro de imagem* ao extrair informações espectrais de mapas ao representá-los como imagens (CARPIN, 2008). No entanto, os mapas que serão fundidos devem estar na mesma escala. Este método é chamado aqui de DNIFM (*Deterministic Non-Iterative Fast Method*).

# 2 Problema e Abordagem

## 2.1 Definição do problema de fusão de mapas

Podemos definir o problema como em (CARPIN, 2008). Assumindo que um mapa de ocupação $M$ é representado como uma matriz de $r$ linhas e $c$ colunas. $M_i$ é o mapa do robô $i$.

Cada célula $M_i(j, k)$ pode conter três valores codificados distintos, indicando se a célula está livre, ocupada ou desconhecida; Como as células geralmente são codificadas como probabilidades de ocupação, essas podem ser facilmente convertidas para esta representação.

Dado dois mapas, $M_1$ e $M_2$, o objetivo da fusão de mapa é encontrar uma transformação rígida (escala, rotação e translação) em que os dois mapas se sobreponham.

## 2.2 Abordagem do problema

A idéia básica por trás da heurística proposta aqui é resolver o problema de mesclagem do mapa como se quiséssemos sobrepô-los manualmente, utilizando um papel transparente. Primeiro, tentamos encontrar lugares semelhantes nos mapas e colocamos o lugar mais parecido em um mapa em cima do outro. Depois disso, um dos mapas é girado até que a maioria dos outros locais semelhantes sejam sobrepostos. Se os mapas estiverem na mesma escala, a fusão está pronta. No entanto, se não estiverem, um mapa precisa ser esticado para que lugares semelhantes se sobreponham. Caso não seja esticado, apenas o primeiro ponto é sobreposto. O processo de combinação do mapa é detalhado abaixo.

### 2.2.1 Detecção e correspondência de *key-points*

Dados os mapas $M_1$ e $M_2$, esses são passados por um filtro gaussiando $5 \times 5$ para diminuir o ruído criado na geração dos mapas. Após, *key-points* são detectados no mapa utilizando SIFT. E finalmente, é feita a correspondência entre os *key-points* utilizando o algoritmo do vizinho mais próximo. Como o número de correspondências geralmente não passa das centenas, pode ser utilizado um método de força bruta sem muitos danos ao tempo de processamento. A figura 5 ilustra os passos.

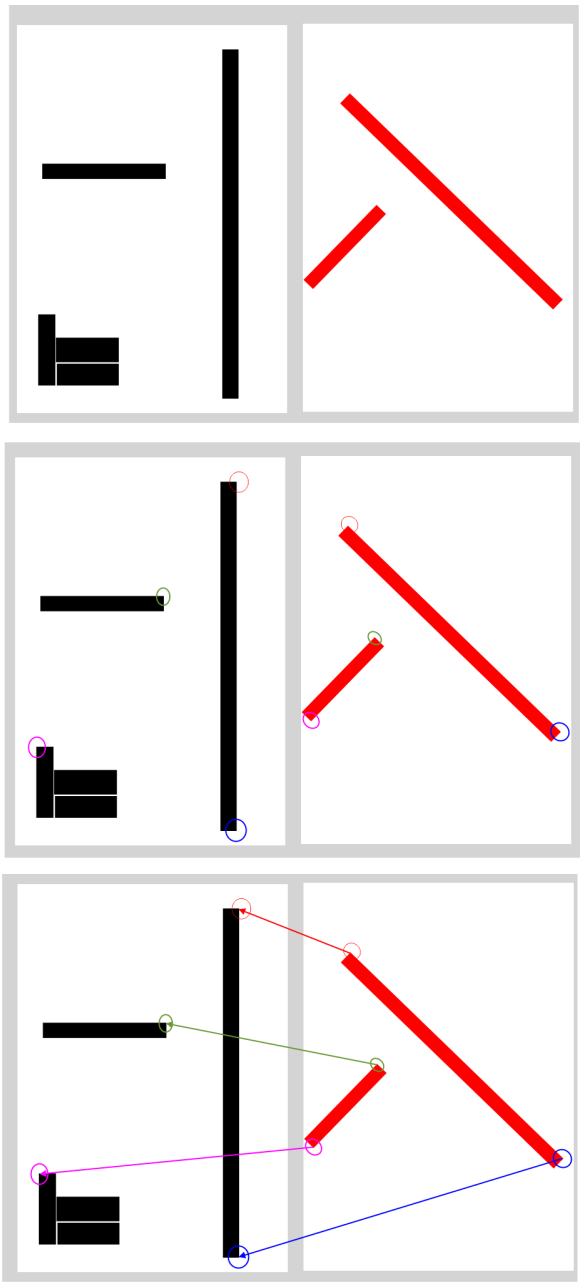

Figura 5 – A imagem de cima ilustra os dois mapas a serem combinados, $M_1$ e $M_2$; A imagem do meio mostra os *key-points* detectados após a execução do SIFT; A imagem de baixo representa a corerespondência de cada par de *key-point*

### 2.2.2 Alinhamento do pivô

O par de *key-points* que possuem a maior similaridade, são utilizado como pivô $(p_{v_1}, p_{v_2})$. Os descritores de cada *key-point* é um vetor, então estes podem ser comparados utilizando algo simples como a distância euclidiana. Podemos então dizer que os descritores que possuem a menor distância entre eles, são os que possuem maior similaridade.

Estes estão ilustrados como círculos verdes nas Figuras 5 e 6.

Ambos os mapas são sobrepostos, e $M_2$ é transladado de forma que ambos pivôs fiquem na mesma posição $(p_{v_1} = p_{v_2} = p_v)$ como mostrado na primeira imagem da Figura 6.

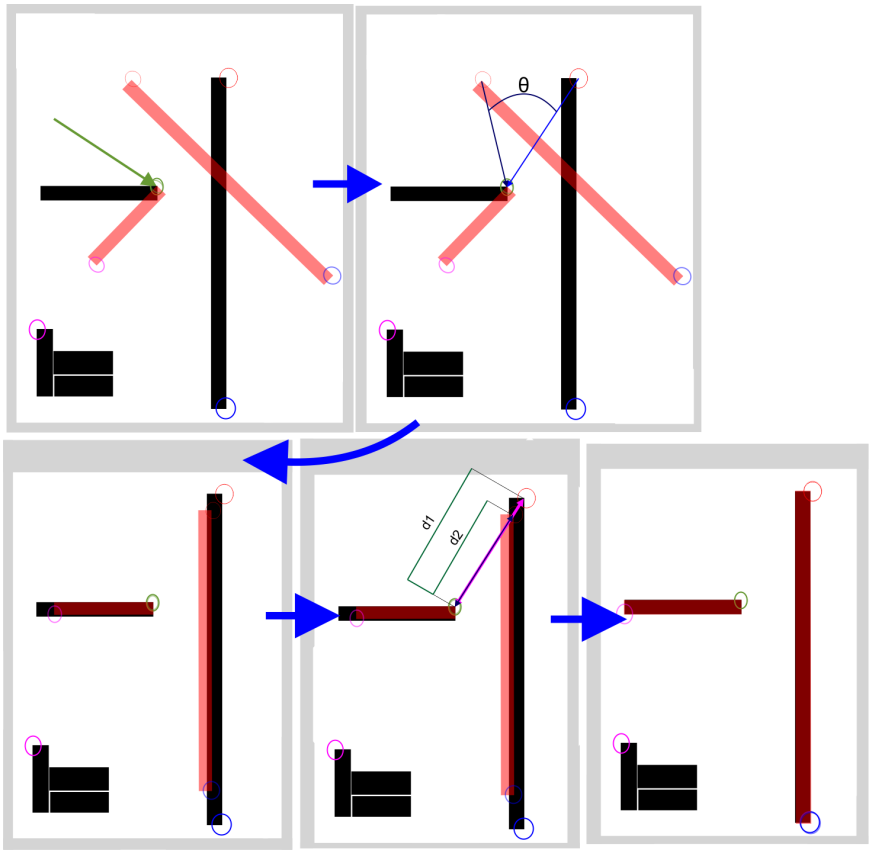


Figura 6 – Passos para combinação dos mapas após a seleção do pivô. Da esquerda para direita, cima para baixo: Alinhamento do pivô, cálculo do ângulo, rotação, cálculo da escala, e mudança de escala.

### 2.2.3 Rotação entre os mapas

Mais pares de *key-points* são selecionados no mapa. Seja $p_1$ e $p_2$ pontos pareados dos mapas $M_1$ e $M_2$ respectivamente. Podemos calcular o ângulo $\theta$ entre eles em relação a $p_v$, e assim rotacionar $M_2$. O valor de $\theta$ pode ser calculado da seguinte forma:

$$d_1 = \|\vec{v_1}\| \tag{2.1}$$

$$d_2 = \|\vec{v_2}\| \tag{2.2}$$

$$s = \frac{\vec{d_1}}{\vec{d_2}} \tag{2.3}$$

$$\vec{v_1} = p_1 - p_v \tag{2.4}$$

$$\vec{v_2} = p_2 - p_v \tag{2.5}$$

$$\theta = cos^{-1}(\frac{\vec{v_1} \cdot \vec{v_2}}{\|\vec{v_1}\| \cdot \|\vec{v_2}\|}) \tag{2.6}$$

### 2.2.4 Transformação de escala entre os mapas

Se os mapas não estiverem na mesma escala, $p_2$ e $p_1$ não irão se coincidir após a rotação de $M_2$. Sendo assim, uma transformação de escala é necessária, este valor $s$ pode ser calculado pela diferença entre $\vec{v_2}$ and $\vec{v_1}$. A proporção entre $\vec{v_2}$ e $\vec{v_1}$ também nos diz qual mapa possui a maior resolução. Podemos escalar o mapa de menor resolução para o de maior resolução para que não ocorra perda de informação. Após este passo, o processo de fusão dos mapas está completo.

### 2.2.5 Calculando a matriz de transformação e evitando fusões incorretas

Como mostrada, a fusão dos mapas ocorre após uma translação (alinhamento do pivô), uma rotação, e uma mudança de escala. Todas essas transformação podem ser incorporadas em uma matriz de transformação $R$.

A matriz $R$ (2.11) pode ser descrita como sendo uma matriz de translação $T$ (equação 2.7) multiplicado por uma matriz de transformação afim $A_{2\times3}$ (equação 2.10), para rotacionar e reescalar em torno do pivô.

$$T = \begin{bmatrix} 1 & 0 & p_{v_{1_x}} - p_{v_{2_x}} \\ 0 & 1 & p_{v_{1_y}} - p_{v_{2_y}} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

$$\alpha = s \cdot cos(\theta) \tag{2.8}$$

$$\beta = s \cdot sin(\theta) \tag{2.9}$$

$$A = \begin{bmatrix} \alpha & \beta & (1-\alpha) \cdot p_{v_x} - \beta \cdot p_{v_y} \\ -\beta & \alpha & \beta \cdot p_{v_x} + (1-\alpha) \cdot p_{v_y} \end{bmatrix} \tag{2.10}$$

$$R = A \times T \tag{2.11}$$

O valor de cada célula de $M_2$ após a transformação é dado na equação 2.12 e o valor de cada célula no mapa final $M_{f_{x,y}}$ é demonstrado na equação 2.13.

$$M_{t_{x,y}} = R \times M_{1_{x,y,1}} \tag{2.12}$$

$$M_{f_{x,y}} = 0.5 \cdot (M_{1_{x,y}} + A \times T \times M_{2_{x,y,1}}) \tag{2.13}$$

Se todos os pares *key-points* corresponderem, então teremos uma sobreposição perfeita após a transformação. No entanto, na maioria dos casos, más correspondências devem ocorrer, (como os *key-points* roxos na Figura 5. Pra evitar fusões incompatíveis como mostrado na Figura 7, os passos descritos em 6 são repetidos para cada par de *key-points* não pivô. A cada iteração, após a transformação, a mediana da distâncias entre cada par é calculada, e a transformação com a menor é selecionada. Caso a maior parte dos *key-points* estejam relacionados corretamente, a mediana irá nos dar uma boa indicação o quão longe, a maioria dos *key-points* estão do local desejado.

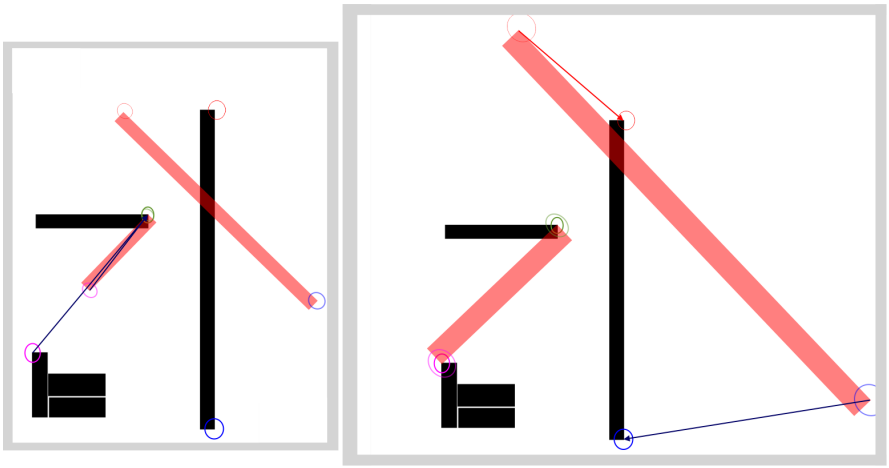

Figura 7 – Seleção de *key-points* incorreta: Após as transformação, os mapas estão combinados de forma errada.

O algoritmo 1 descreve o processo de fusão dos mapas. Iremos chamar esse algorítmo de SM (*Sift Merge*).

**Algoritmo 1** SM

**Input:** $\text{Mapa}_1$, $\text{Mapa}_2$
**Output:** Matriz de transformação $R_m$ de $\text{Mapa}_1$ para $\text{Mapa}_2$

1: $\text{Blur}(\text{Map}_1)$
2: $\text{Blur}(\text{Map}_2)$
3: $\text{keypoints}_1 \leftarrow \text{SIFT}(\text{Map}_1)$
4: $\text{keypoints}_2 \leftarrow \text{SIFT}(\text{Map}_2)$
5: matches $\leftarrow$ FLANN($\text{keypoints}_1$, $\text{keypoints}_2$)
6: bestMatch $\leftarrow$ SelectBestMatch(matches)
7: $\text{pivot}_1$, $\text{pivot}_2 \leftarrow$ bestMatch.pointsPair
8: translationToPivot1 $\leftarrow \text{pivot}_1 - \text{pivot}_2$
9: bestMedian $\leftarrow \infty$
10: **for** match in matches **do**
11: **if** (match $\neq$ bestMatch) **then**
12: $p_1$, $p_2 \leftarrow$ match.pointsPair
13: $v_1 \leftarrow p_1 - \text{pivot}_1$
14: $v_2 \leftarrow (p_2 + \text{translationToPivot1}) - \text{pivot}_1$
15: $\theta \leftarrow$ calculateAngle($v_1$, $v_2$)
16: ratio $\leftarrow v_1$.length/$v_2$.length
17: distances $\leftarrow$ []
18: **for** match_2 in matches **do**
19: $R \leftarrow$ createAffineMatrix($\text{pivot}_1$, ratio, $\theta$)
20: $p_3$, $p_4 \leftarrow$ match_2.pointsPair
21: $p_4 \leftarrow p_4 + \text{translationToPivot1}$
22: $p_4 \leftarrow R \times p_4$
23: distances.insert($p_3 - p_4$)
24: **end for**
25: median $\leftarrow$ calculateMedian(distances)
26: **if** (median $<$ bestMedian) **then**
27: bestMedian $\leftarrow$ median
28: selectedAngle $\leftarrow \theta$
29: selectedRatio $\leftarrow$ ratio
30: **end if**
31: **end if**
32: **end for**
33: $T \leftarrow$ createTranslationMatrix($\text{pivot}_1$, $\text{pivot}_2$)
34: $A \leftarrow$ createAffineMatrix($\text{pivot}_1$, selectedRatio, selectedAngle)
35: $R_m \leftarrow A \times T$
**return** $R_m$

# 3 SM: Primeiros experimentos e resultados

Dados dois mapas $M_1$ e $M_2$, há diferentes transformações que podemos aplicar para que possamos fundi-los. O *índice de aceitação*, $\omega(M_1, M_2)$, introduzido em (BIRK; CARPIN, 2006), será a métrica para estabelecer a qualidade da combinação dos mapas. Podemos defini-lo como

$$\omega(M_1, M_2) = \begin{cases} 0 & \text{se } agr(M_1, M_2) = 0 \\ \frac{agr(M_1,M_2)}{agr(M_1,M_2)+dis(M_1,M_2)} & \text{se } agr(M_1, M_2) \neq 0 \end{cases} \tag{3.1}$$

onde $agr(M_1, M_2)$ é o número de células em $M_1$ e $M_2$ em que ambas estão livres, ou ocupadas. E $dis(M_1, M_2)$ é o número de células em $M_1$ que estão livres, e em $M_2$ ocupadas, e vice-versa.

## 3.1 Fusão de dois mapas idênticos.

### 3.1.1 Teste de robustez

Se $M_1 = M_2$, então $\omega(M_1, M_2) = 1$. Podemos verificar o quão bom o algoritmo é se, ao aplicar uma *transformação aleatória* em um mapa, o algoritmo poder encontrar uma transformação que a reverte. Seja $M$ o mapa, e $M' = RM$, onde $R$ é a *transformação aleatória*. Seja $R_m$ a transformação produzida pelo algoritmo de fusão de mapas. Se o algoritmo conseguir encontrar a transformação inversa, o valor de $\omega(M, R_m M')$ deve ser igual a 1.

Estes testes foram realizados utilizando mapas 530x530 [1] como mostrado na Figura 8.[1] O teste é composto de 1000 execuções. Durante cada execução, o algorítmo escolhe um mapa aleatoreamente e uma transformação aleatória é aplicada.

[1] Estes mapas estão disponíveis em <http://robotics.ucmerced.edu/software>

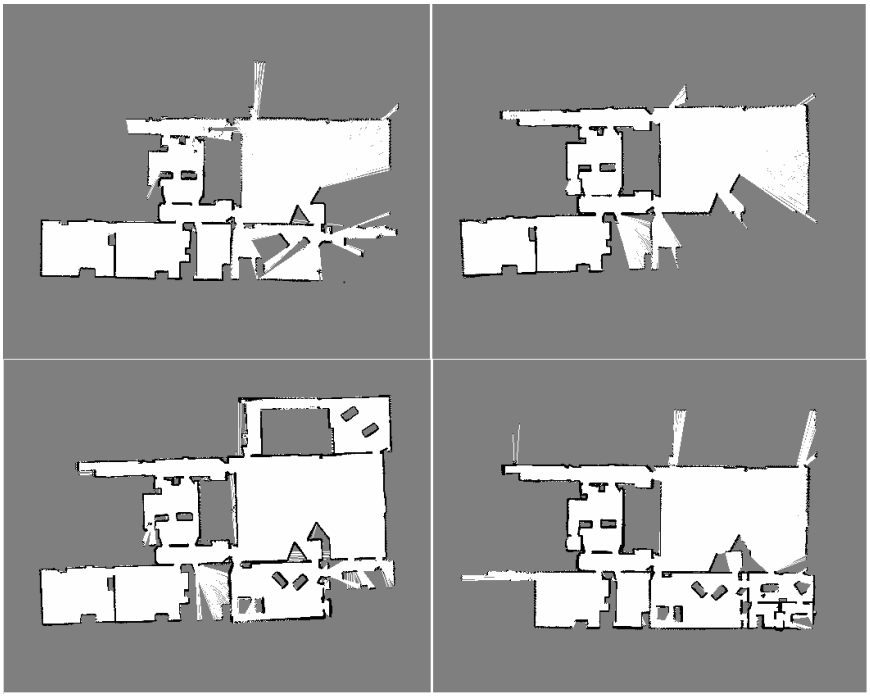

Figura 8 – Mapas utilizados nos testes.

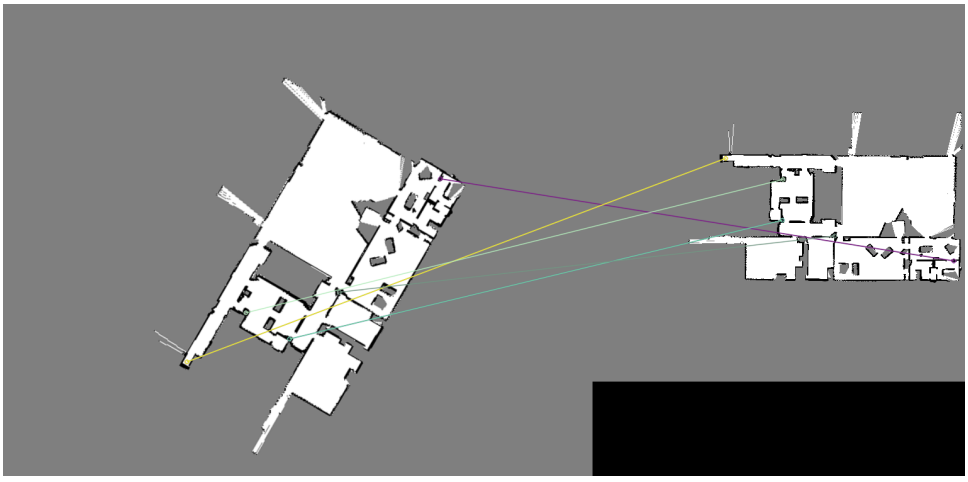

Figura 9 – Combinação de dois mapas idênticos com diferentes rotação, translação e escala. As linhas e circulos mostram as correspondência dos *key-points* de cada mapa.

As Figuras 9 e 10 mostram, respectivamente, o par de mapas com os *key-points* correspondentes e o resultado da combinação durante uma das iterações. O *índice de aceitação* nesta iteração foi de 0.9817. As diferenças entre os mapas inicial e o resultante após as transformações são quase imperceptíveis, mas tornam-se claras na Figura 10b quando destacadas. A figura 11 representa o índice de aceitação de cada execução. O valor médio é de 0.9768 e desvio padrão de 0.0199.

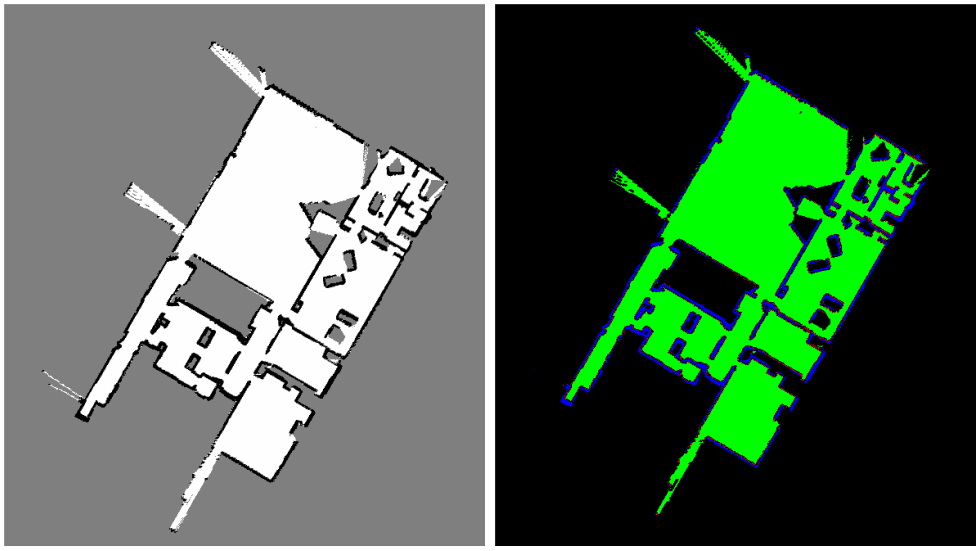

Figura 10 – a) Mapa final após a fusão; b) O mapa de concordância, após a fusão. Os píxeis verdes e azuis são células que, respectivamente, estão ocupadas ou livres em ambos os mapas. Os píxeis vermelhos são células em que eles estão diferentes.

### 3.1.2 Primeiras comparações entre SM e DNIFM

A implementação de DNIFM fornecida por (CARPIN, 2008) não mescla mapas de escala diferentes. O método que apresentamos permite a combinação de mapas com diferentes escalas, como já mencionado. No entanto, para fazer uma comparação honesta com o nosso método, o teste anterior foi repetido usando os mesmos mapas de escala para ambas as abordagens.

A implementação do DNIFM possui um parâmetro ajustável para aumentar a precisão enquanto sacrifica a velocidade de execução. Este parâmetro é o número de hipóteses $n$. O algoritmo calcula o espectros circular entre $M_1$ e $M_2$ e extrai os $n$ máximos locais associados aos valores mais altos. Cada um desses valores $n$ corresponde a uma rotação de candidatos. Para cada rotação, o algoritmo calcula uma translação correspondente. O número do parâmetro de hipótese foi definido como seis.

*Resultados do DNIFM* - após a execução, a média do índice de aceitação foi de 0.9943, e desvio padrão foi de 0.0172, corroborando os dados mostrados em (CARPIN, 2008). O tempo médio de cada iteração foi de $877.4061 \pm 58.5413$ $ms$.

*Resultados do SM* - após a execução, a média do índice de aceitação foi de 0.9837, e desvio padrão foi de 0.0129 (Figura 12). O tempo médio de cada iteração foi de $400.0374 \pm 51.7839$ $ms$.

Esses resultados mostram que nosso algoritmo apresenta resultados parecidos independente da escala, e ainda foi mais rápido do que a implementação *robusta* do DNIFM usando seis hipóteses como parâmetro.

## 3.2 Fusão de mapas em escalas diferentes

*Mesmo ambiente e escalas diferentes* – Os pares de mapas utilizados nessa parte estão ilustrados nas Figuras 13 e 14. Eles estão disponíveis no mesmo repositório que os mostrados anteriormente. Esses mapas originalmente têm tamanho $790 \times 790$. No entanto eles foram transformados para uma *escala diferente* para que o teste podermos testar o que queremos demonstrar. Para facilitar a visualização, as Figuras 15 e 16 fornecem sobreposições dos mapas logo após a conclusão das transformações. O resultado final da fusão é excelente, apesar de alguns problemas que apareceram.

Primeiro, dependendo do mapa, com o conjunto correto de transformações, o SIFT não conseguiu encontrar a correspondência certa de pontos-chave. Segundo, às vezes a mesclagem não alinha corretamente os mapas, na presença de longos corredores, o desalinhamento é perceptível, e isso aparece na parte inferior da Figura 15.

## 3.3 Problemas na implementação

Após os testes feitos e durante a refatoração do código do algoritmo SM, foram detectados problemas que fizeram tanto o tempo de execução aumentar, quanto a precisão diminuir.

### 3.3.1 Problemas que fizeram o tempo aumentar

Cada vez que a matriz de translação $T$ ou a matriz de transformação afim $A$ eram calculadas, a transformação era aplicada imediatamente ao mapa em vez de aplicar somente a matriz resultante final. Para fim de visualização outra transformação era feita com as bordas do mapa aumentadas em 150 píxeis, já que haviam casos em que dependendo da escala, as bordas do mapa não cabiam na imagem, ou seja, a imagem era modificada desnecessariamente quatro vezes ao calcular as matrizes de transformação.

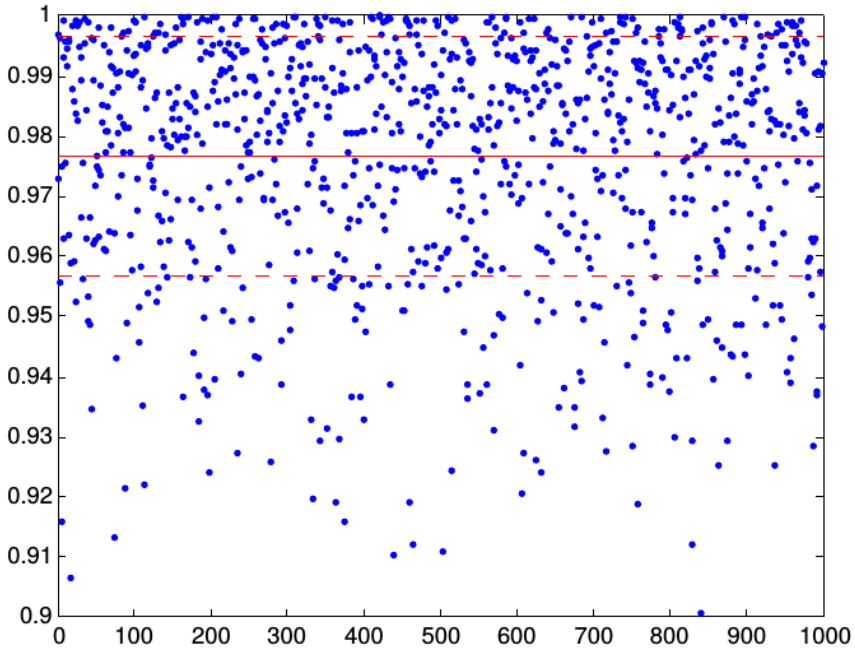


Figura 11 – Índice de aceitação após 1000 execuções com transformações aleatórias: a linha vermelha mostra o valor médio e as linhas tracejadas mostram o desvio padrão em torno do valor médio.

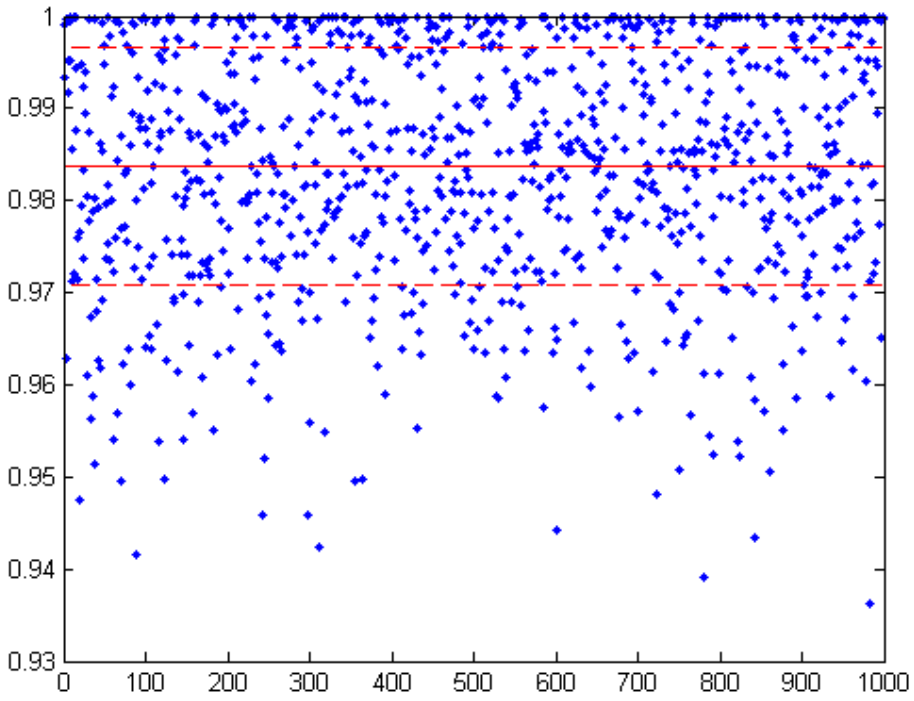


Figura 12 – Índice de aceitação após 1000 execuções com transformações aleatórias que não incluem escala: a linha vermelha mostra o valor médio e as linhas tracejadas mostram o desvio padrão em torno do valor médio.

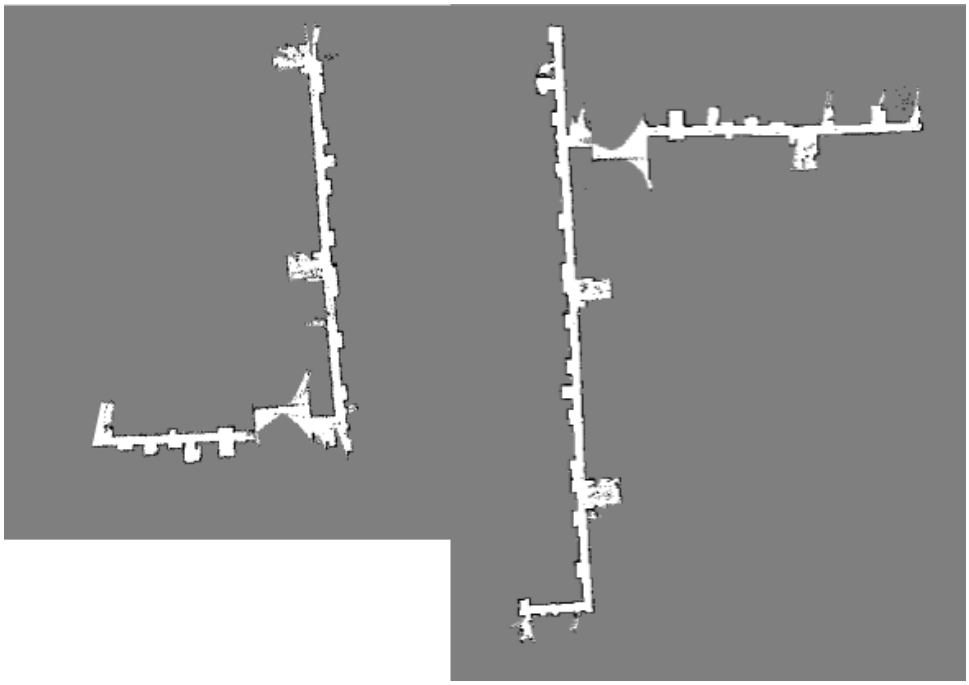

Figura 13 – Par 1 de mapas de mesmo ambiente mas em escalas diferentes.

### 3.3.2 Problemas que fizeram a precisão diminuir

Como a transformação $T$ e $M$ eram aplicadas diretamente ao mapa imediatamente após o cálculo de cada e não apenas a matriz resultante final, os erros eram acumulados devido a interpolação dos píxeis após cada transformação.

Ao corrigir tais problemas, tanto o tempo quanto o índice de aceitação tiveram melhores resultados, como será mostrado na seção 4.1.

## 3.4 SM: limitações e melhorias

O algoritmo SM produz um conjunto de transformações afins (rotações, traduções e escalas) que podem ser usadas para sobrepor dois mapas, e mesmo com algumas falhas de implementação, este se mostrou mais rápido do que o DNIFM. No entando, ele possui algumas limitações:

– A fusão de mapas no momento não é precisa a nível de pixel. Um refinamento portanto é necessário para que uma fusão perfeita ocorra.

– É muito dependente do ponto-chave do pivô imutável. Se não representar a mesma posição nos dois mapas, a fusão não funcionará.

Para resolver essas limitações, foi proposto uma melhoria desse algoritmo, que foi chamado de SM$^2$.

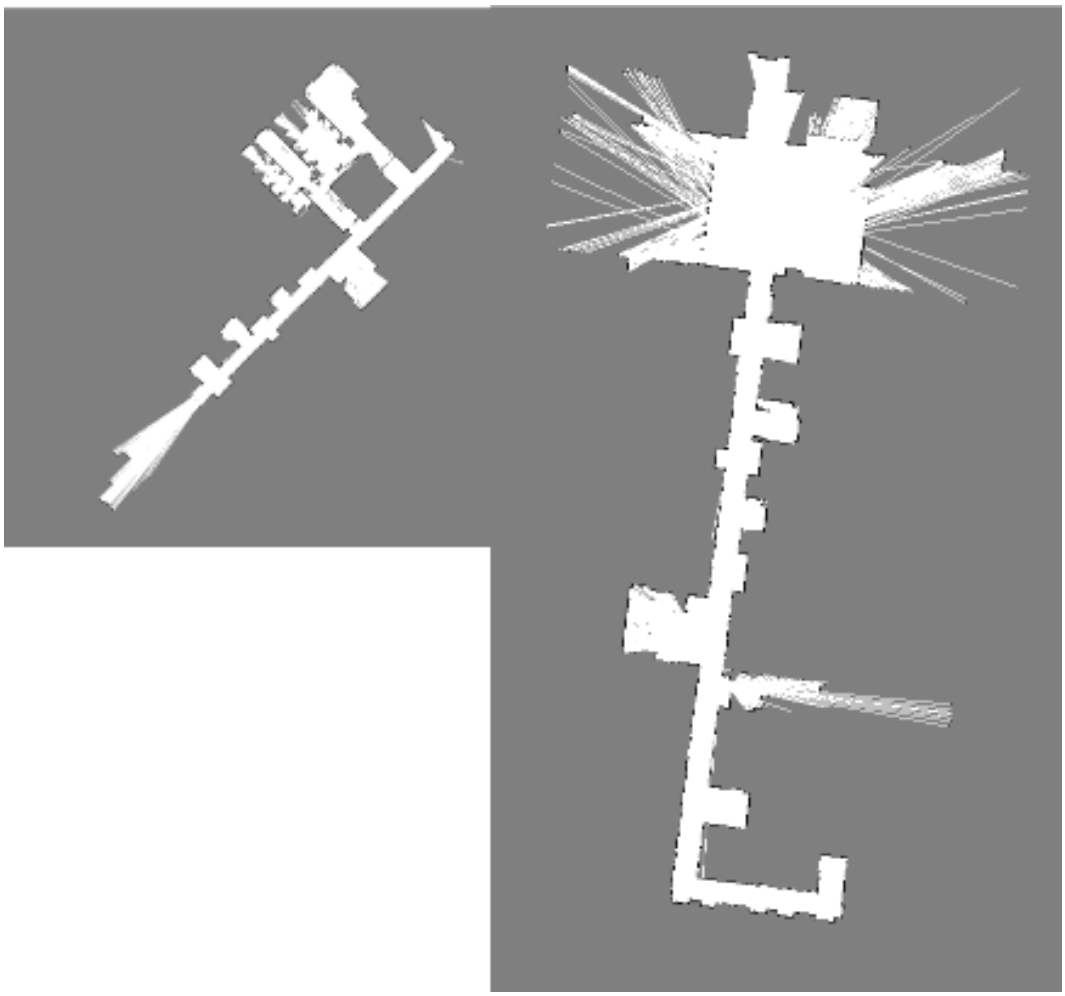

Figura 14 – Par 2 de Mapas de mesmo ambiente mas em escalas diferentes.

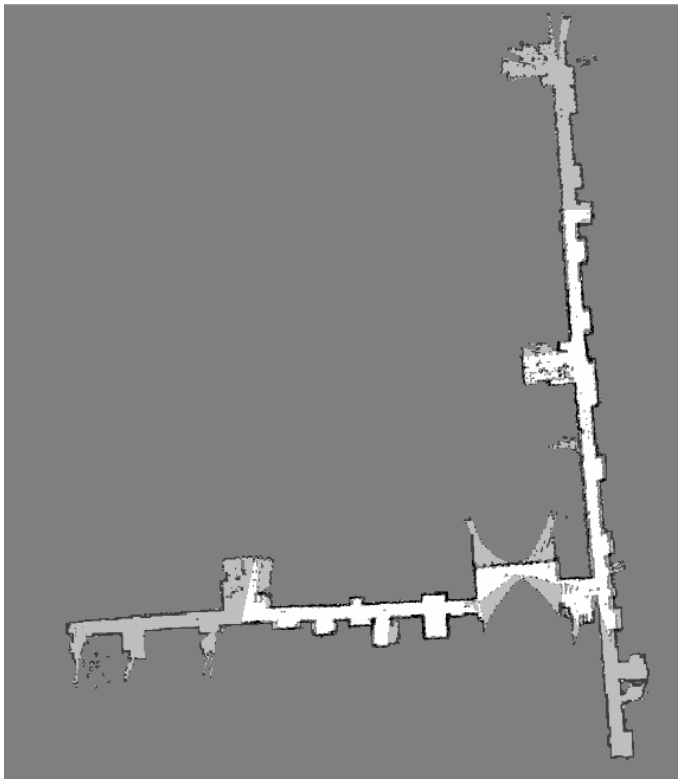

Figura 15 – Sobreposição do par de mapas da Figura 13 após a fusão.

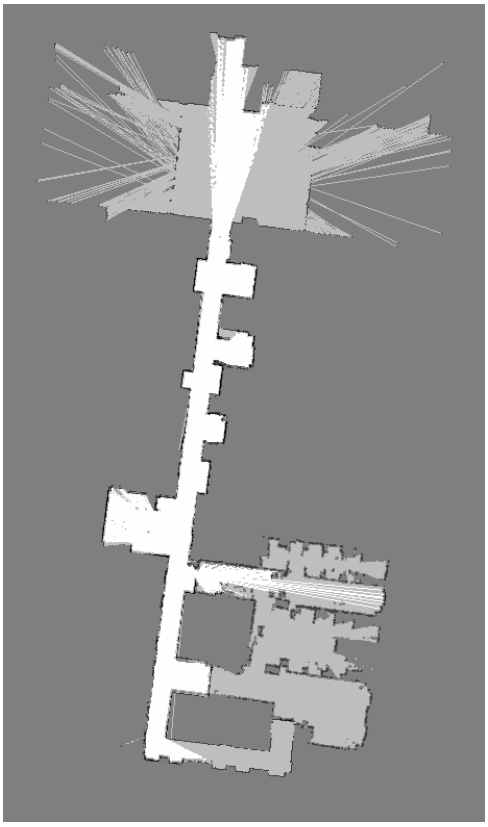

Figura 16 – Sobreposição do par de mapas da Figura 14 após a fusão.

# 4 SM$^2$: refinamentos na abordagem proposta

A abordagem é similar a mostrada no SM, mas com uma iteração a mais para refinar o alinhamento entre os mapas e assim aumentar a qualidade de sobreposição entre eles.

Como mostrado na equação 2.10, $A$ é uma matriz $2 \times 3$, e a transformação que cada célula que $M_i$ recebe é calculada na equação 2.12

$$T = \begin{bmatrix} 1 & 0 & p_{v2}.x - p_{v1}.x \\ 0 & 1 & p_{v2}.y - p_{v1}.y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

$$A = \begin{bmatrix} \alpha & \beta & (1-\alpha) \cdot p_{v2}.x - \beta \cdot p_{v2}.y \\ -\beta & \alpha & \beta \cdot p_{v2}.x + (1-\alpha) \cdot p_{v2}.y \end{bmatrix} \tag{4.2}$$

$$R = A \times T \tag{4.3}$$

Mas para que seja possível encadear a multiplicação das matrizes de transformação, $R_f = R_t \times R_i$, elas não podem ser $2 \times 3$; É necessário transformá-las em $3 \times 3$. Isso pode ser feito mudando como a Matriz de Transformação Afim ($A$) é calculada na função. Ao mudar o cálculo da Matriz de Transformação Afim da equação 4.2 para a equação 4.4, a transformação final $R$ passa a ser $3 \times 3$ e o cálculo de cada célula $M_f$ é calculada passa a ser descrita na equação 4.8.

$$A = \begin{bmatrix} \alpha & \beta & (1-\alpha) \cdot p_{v2}.x - \beta \cdot p_{v2}.y \\ -\beta & \alpha & \beta \cdot p_{v2}.x + (1-\alpha) \cdot p_{v2}.y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

Ao passarmos $M_i$ pelas duas transformações, teremos $M_f$ como ilustrado abaixo:

$$R_i = f(M_i, M_j) \tag{4.5}$$

$$R_t = f(R_i \times M_i, M_j) \tag{4.6}$$

$$R_f = R_t \times R_i \tag{4.7}$$

$$M_{f_{x,y,1}} = R_f \times M_{i_{x,y,1}} \tag{4.8}$$

Devido ao algoritmo possuir a mesma abordagem que o algoritmo SM e possuir duas iterações, iremos chamá-lo aqui de SM$^2$. O algoritmo final do método está detalhado no algoritmo 2 a seguir.

**Algoritmo 2** SM²

**Input:** $\text{Mapa}_1$, $\text{Mapa}_2$
**Output:** Matriz de transformação $R_m$ de $\text{Mapa}_1$ para $\text{Mapa}_2$

$R_m \leftarrow \text{Identidade}_{3\times3}$
**for** $i \in \{1, 2\}$ **do**
    BLUR($\text{Map}_1$)
    BLUR($\text{Map}_2$)
    $\text{keypoints}_1 \leftarrow \text{SIFT}(\text{Map}_1)$
    $\text{keypoints}_2 \leftarrow \text{SIFT}(\text{Map}_2)$
    matches $\leftarrow$ FLANN($\text{keypoints}_1$, $\text{keypoints}_2$)
    best_match $\leftarrow$ SELECT_BEST_MATCH(matches)
    $\text{pivot}_1$, $\text{pivot}_2 \leftarrow$ best_match.pair
    $t_{12} \leftarrow \text{pivot}_2 - \text{pivot}_1$
    best_median $\leftarrow \infty$
    **for** match in matches **do**
        **if** (match $\neq$ best_match) **then**
            $p_1$, $p_2 \leftarrow$ match.pair
            $v_1 \leftarrow p_1 - \text{pivot}_1$
            $v_2 \leftarrow p_2 - \text{pivot}_2$
            $\theta \leftarrow$ CALCULATE_ANGLE($v_1$, $v_2$)
            ratio $\leftarrow v_1$.length/$v_2$.length
            distances $\leftarrow [\emptyset]$
            **for** match_2 in matches **do**
                $R \leftarrow$ AFFINE_MATRIX($\text{pivot}_2$, ratio, $\theta$)
                $p_3$, $p_4 \leftarrow$ match_2.pair
                $p_3 \leftarrow p_3 + t_{12}$
                $p_3 \leftarrow R \times p_3$
                distances.insert($||p_3 - p_4||$)
            **end for**
            median $\leftarrow$ MEDIAN(distances)
            **if** (median $<$ best_median) **then**
                best_median $\leftarrow$ median
                selected_angle $\leftarrow \theta$
                selected_ratio $\leftarrow$ ratio
            **end if**
        **end if**
    **end for**
    $T \leftarrow$ TRANSLATION_MATRIX($\text{pivot}_1$, $\text{pivot}_2$)
    $A \leftarrow$ AFFINE_MATRIX($\text{pivot}_2$, selected_ratio, selected_angle)
    $R_m \leftarrow R_m \times A \times T$
**end for**
**return** $R_m$

## 4.1 Comparativos entre - SM², SM e DNIFM

Para uma comparação direta com o SM, serão feitos os mesmo testes utilizando os algoritmos SM² e DNIFM. Os testes do SM também serão refeitos, e será utilizada a nova implementação ,i.e., retirados os problemas de performance.

## 4.2 Teste de robustez - SM²

### 4.2.1 Primeiro teste utilizando o dataset

A figura 17 plota os índices de aceitação do SM² no primeiro teste. A tabela 1 faz o comparativo entre o SM e o SM². Os valores alcançados pelo SM, tanto o índice de aceitação quanto o tempo, foram melhores do que os obtidos nos testes anteriores. O nosso método no entanto se mostrou mais preciso do que o SM, mostrando que realmente está acontecendo um ajuste fino, como explicado em 2.2.

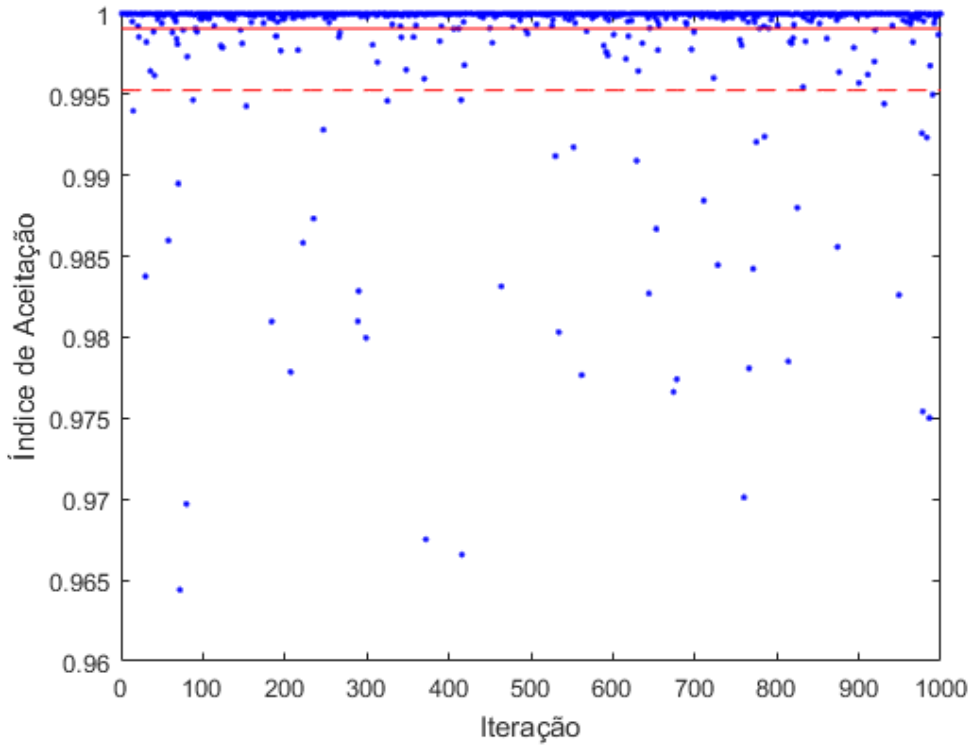


Figura 17 – Índices de aceitação do primeiro teste.

Tabela 1 – Resultados do primeiro teste

| | SM | SM² |
|---|---|---|
| $\omega$ | $0.9970 \pm 0.0048$ | $0.9987 \pm 0.0045$ |
| Tempo | $273.3052 \pm 113.9305$ | $608.8074 \pm 328.6446$ |

### 4.2.2 Segundo Teste utilizando o dataset

A figura 18 plota os índices de aceitação do nosso método no segundo teste. A tabela 2 faz o comparativo entre o SM², o DNIFM e o SM.

Diferente dos resultados em 3.1.2, e devido a correção dos erros na implementação explicados anteriormente, o SM se mostrou melhor que o DNIFM tanto no tempo, quanto no índice de aceitação.

Parecido com o primeiro teste, o SM² se mostrou mais preciso do que o SM, mas leva em média o dobro do tempo para ser executado, o que era de se esperar, mas ainda é mais rápido que o DNIFM.

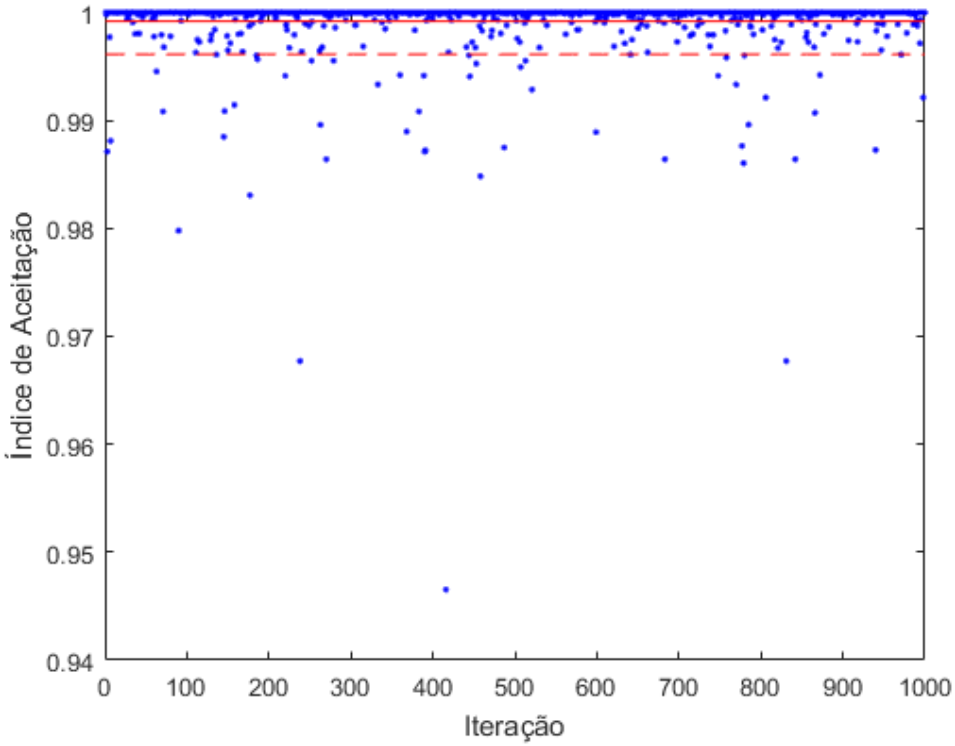


Figura 18 – Índices de aceitação do segundo teste.

Tabela 2 – Resultados do segundo teste

| | SM | DNIFM | SM² |
|---|---|---|---|
| $\omega$ | $0.9972 \pm 0.0039$ | $0.9943 \pm 0.0172$ | $0.9993 \pm 0.0026$ |
| Tempo | $226.3995 \pm 33.7901$ | $877.4061 \pm 58.5413$ | $448.6665 \pm 59.9823$ |

# 5 Experimentos e Resultados em um Ambiente simulado

Os algoritmos DNIFM, SM, e SM$^2$ foram comparados em um ambiente dinâmico simulado. O ambiente simulado foi criado utilizando o Stage (GERKEY; VAUGHAN; HOWARD, 2003) e o ROS (QUIGLEY et al., 2009). A imagem 19 ilustra o ambiente utilizado.

Dois robôs foram utilizados, ilustrados como os quadrados vermelho ($r_v$) e azul ($r_a$), e cada um percorre um mapa por meio de *waypoints* pré-definidos ilustrados pela Tabela 3. Se algum robô encontrar algum obstáculo durante o caminho até o *waypoint*, este irá tentar desviar do obstáculo. Cada robô tem acesso apenas a sua própria localização, e ao mapa que ele mesmo gerou. O mapa gerado por um dos robôs está rotacionado 30° em relação ao mapa global, para que seja analisado a capacidade de fundir mapas que possuem diferentes origens.

Durante a simulação, enquanto os robôs percorrem o ambiente, os mapas gerados por estes vão adquirindo mais detalhes, e representam uma maior parte do ambiente percorrido.

Será analisada a área do mapa que precisa ser revelada para que uma fusão bem sucedida seja possível.

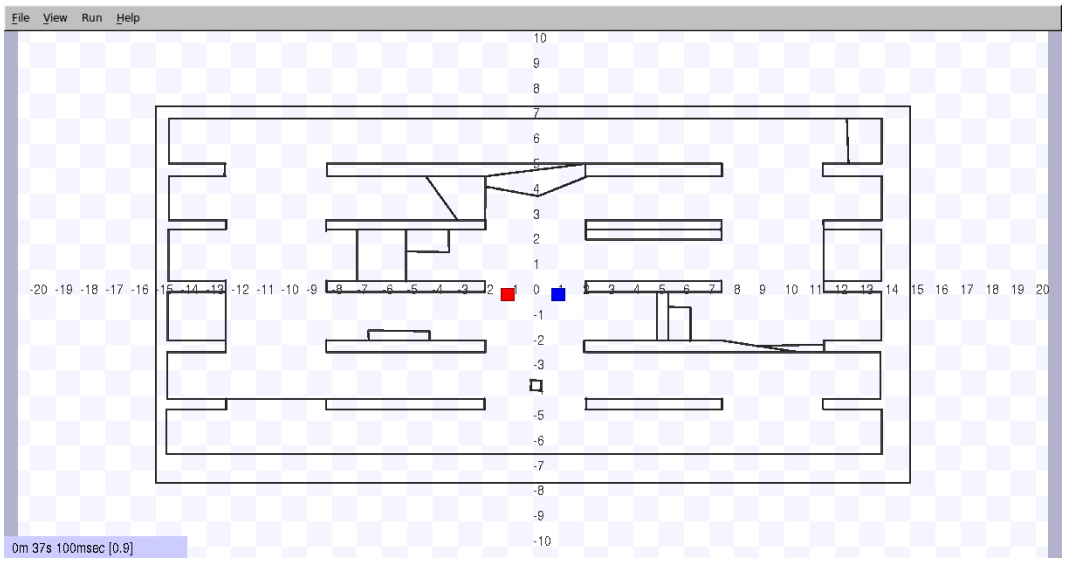


Figura 19 – Mapa simulado no Stage

Tabela 3 – Posição inicial e caminho dos robôs em cada um dos testes

| Primeiro Teste | | Segundo Teste | | Terceiro Teste | |
|---|---|---|---|---|---|
| $r_v$ | $r_a$ | $r_v$ | $r_a$l | $r_v$ | $r_a$ |
| $-1,0$ | $1,0$ | $-1,0$ | $10,6$ | $-11,2$ | $10,6$ |
| $0,3$ | $0,-5.5$ | $0,3$ | $10,2$ | $-11,-3.5$ | $10,2$ |
| $0,-1$ | $10,-3$ | $0,-1$ | $0,0$ | $-1,-3$ | $0,0$ |
| $-10,-1$ | $-11,-5.5$ | $-10,-1$ | $0,-5$ | $-1,-5$ | $0,-5$ |
| $-10,6$ | $-8,-1$ | $-10,6$ | $10,-6$ | $8,-5$ | $10,-6$ |
| $10,1$ | $0,3$ | $10,1$ | $0,3$ | $10,-3$ | $0,3$ |

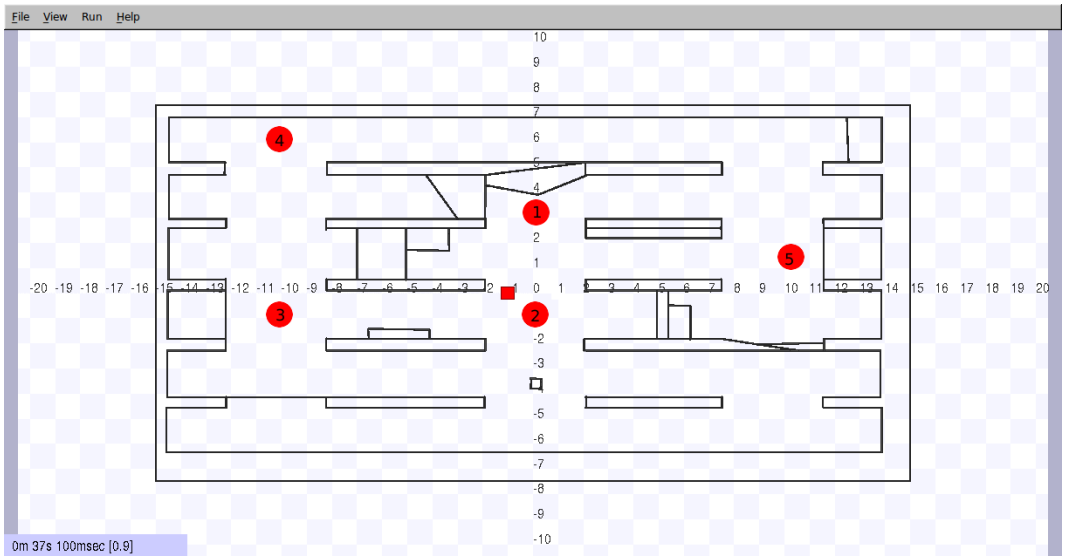


Figura 20 – Posição inicial e *waypoints* do robô vermelho no primeiro e segundo testes

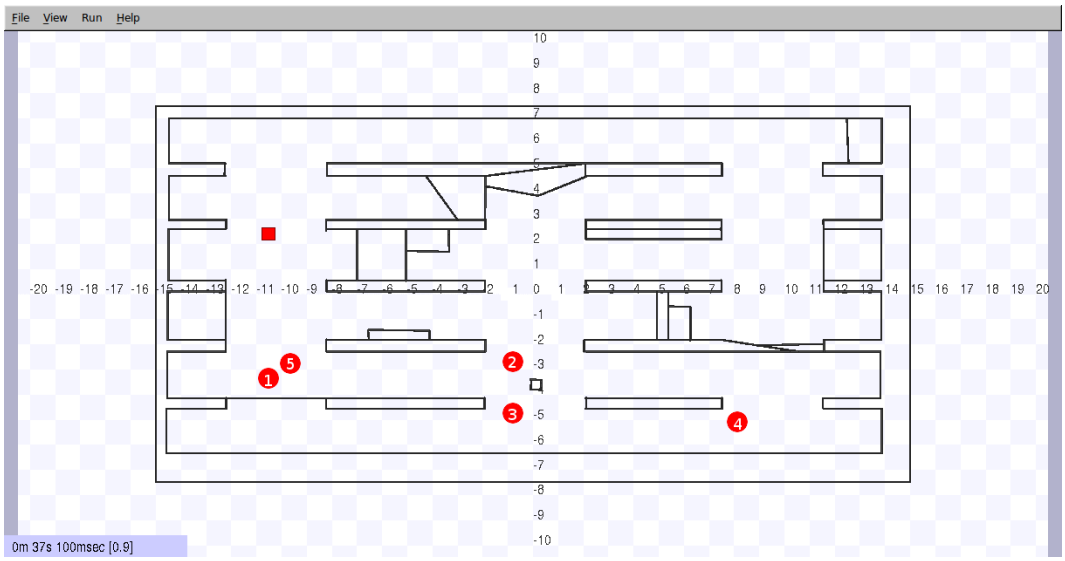


Figura 21 – Posição inicial e *waypoints* do robô vermelho no terceiro teste

Foram feitos 3 testes neste ambiente simulado. Cada um dos testes foi executado 10 vezes.

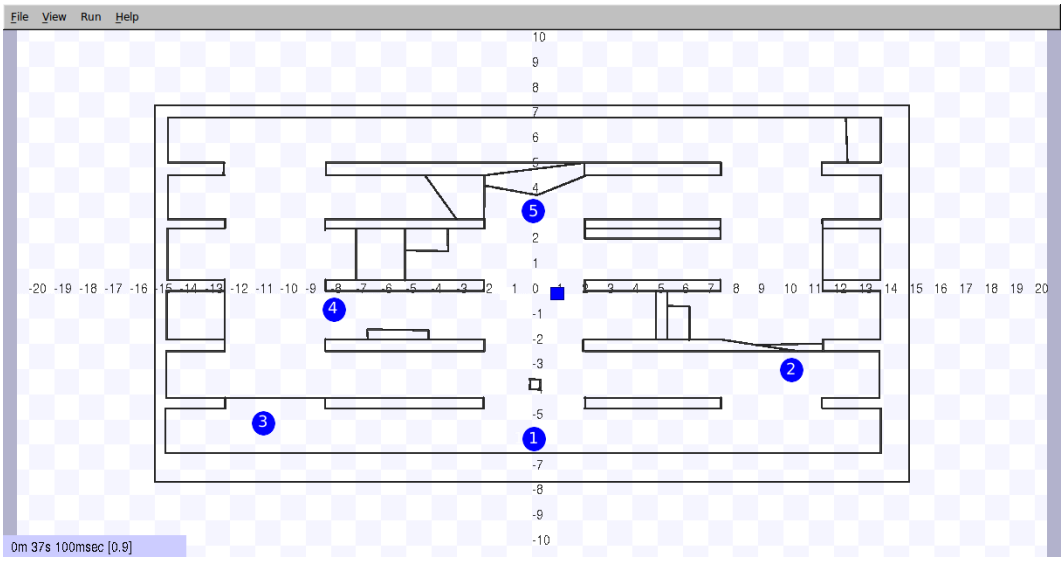


Figura 22 – Posição inicial e *waypoints* do robô azul no primeiro teste

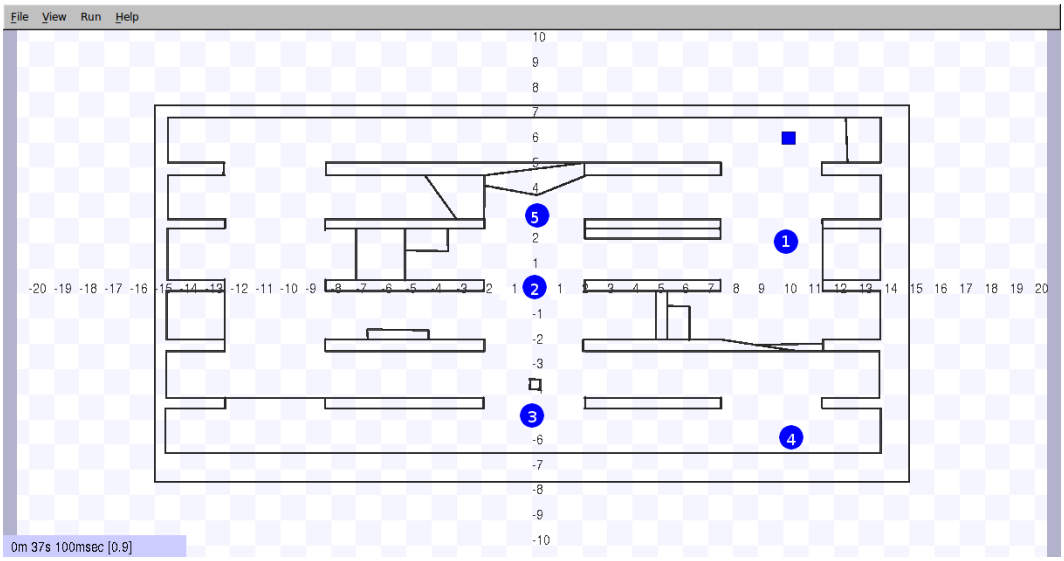


Figura 23 – Posição inicial e *waypoints* do robô azul no segundo e terceiro testes

## 5.1 Resultados do primeiro teste em ambiente simulado

As Figuras 20 e 22 ilustram a posição inicial e os pontos que o robô vermelho ($r_v$) e azul ($r_a$) devem percorrer respectivamente.

As Figuras 24, 25 e 26 mostram os mapas dos robôs no momento em que conseguiram obter uma fusão de mapas bem sucedida utilizando os método SM, SM$^2$ e DNIFM respectivamente.

Tanto o SM quanto no SM$^2$, as áreas necessárias são bem similares, o que era de se esperar, já que o SM$^2$ é o SM com um passo de refinamento a mais. No entanto, o método que necessitou de menor área foi o DNIFM.

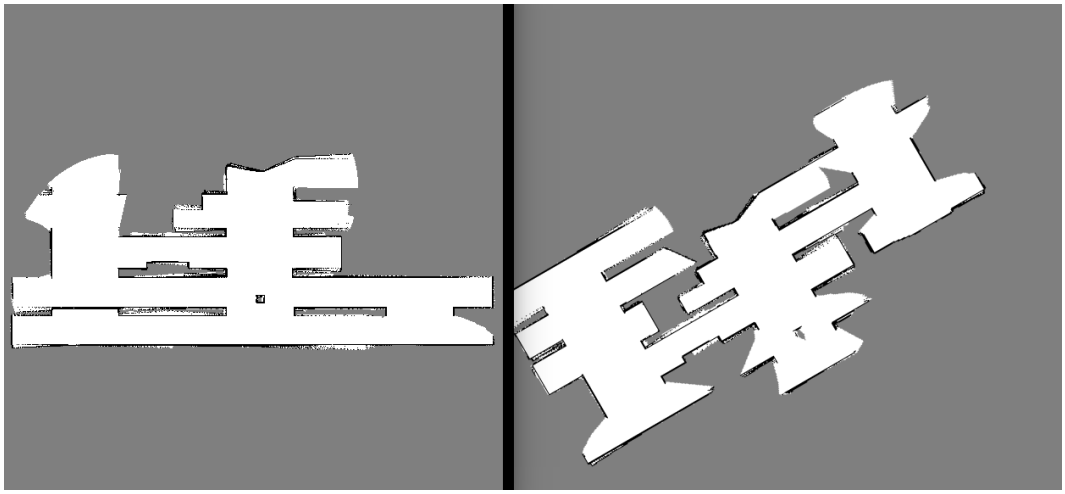

Figura 24 – Área mínima necessária a ser explorada pelos robôs azul e vermelho respectivamente utilizando o SM no primeiro teste.

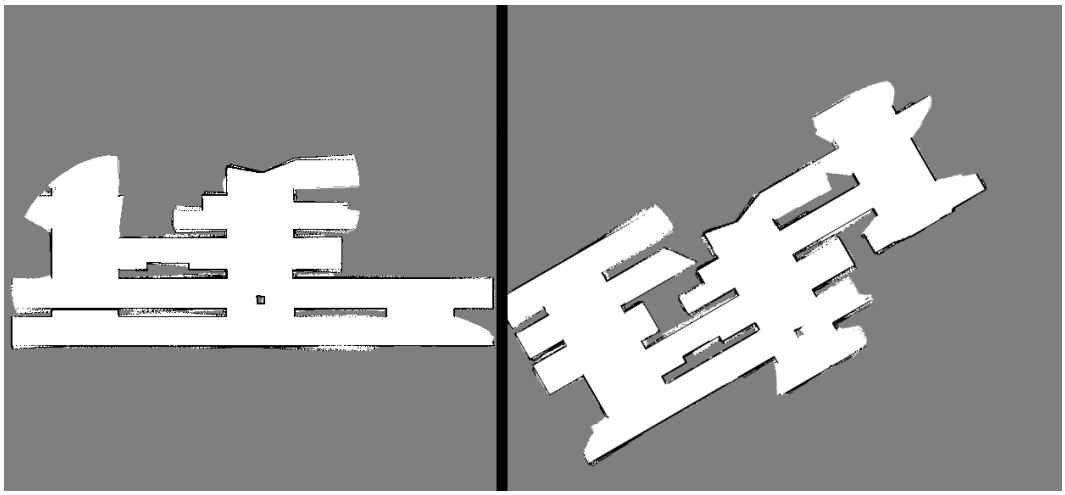

Figura 25 – Área mínima necessária a ser explorada pelos robôs azul e vermelho respectivamente utilizando o SM$^2$ no primeiro teste.

## 5.2 Resultados do segundo teste em ambiente simulado

A posição inicial e os pontos que o robô vermelho ($r_v$) e azul ($r_a$) devem percorrer estão ilustrados nas Figuras 20 e 23, na devida ordem. As áreas necessárias para que a fusão fosse bem sucedida neste segundo teste utilizando os métodos SM e SM$^2$ estão demonstradas nas Figuras 27 e 28.

Neste teste, o método DNIFM não foi capaz de gerar uma fusão de mapas bem sucedida. A Figura 29 mostra os mapas de ambos os robôs após terem percorrido todos os *waypoints*, e a Figura 30 ilustra o resultado de sua tentativa de combinar os mapas.

O método SM foi capaz de obter a fusão correta rapidamente, no entanto o SM$^2$ não foi tão rápido dessa vez, sendo capaz de obter a transformação correta apenas após os robôs terem passado por todos os *waypoints*. O que pôde ser observado durante os testes

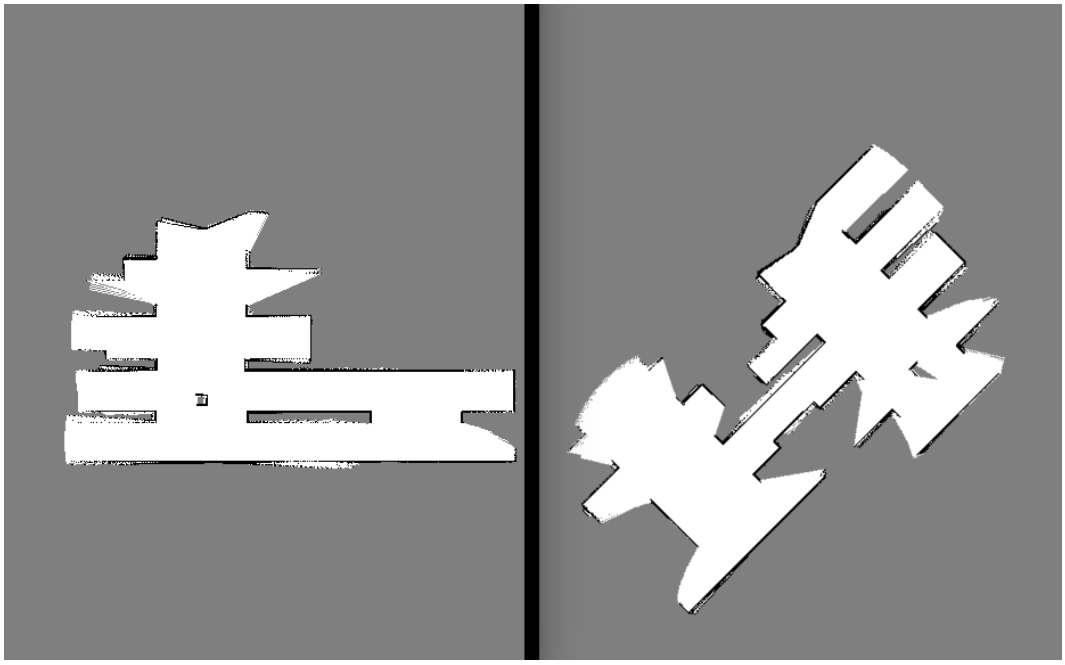

Figura 26 – Área mínima necessária a ser explorada pelos robôs azul e vermelho respectivamente utilizando o DNIFM no primeiro teste.

foi que na primeira iteração do algoritmo $SM^2$, ele encontrava a transformação correta, no entanto, ao passar pela segunda iteração, a transformação com a menor mediana não era a correta.

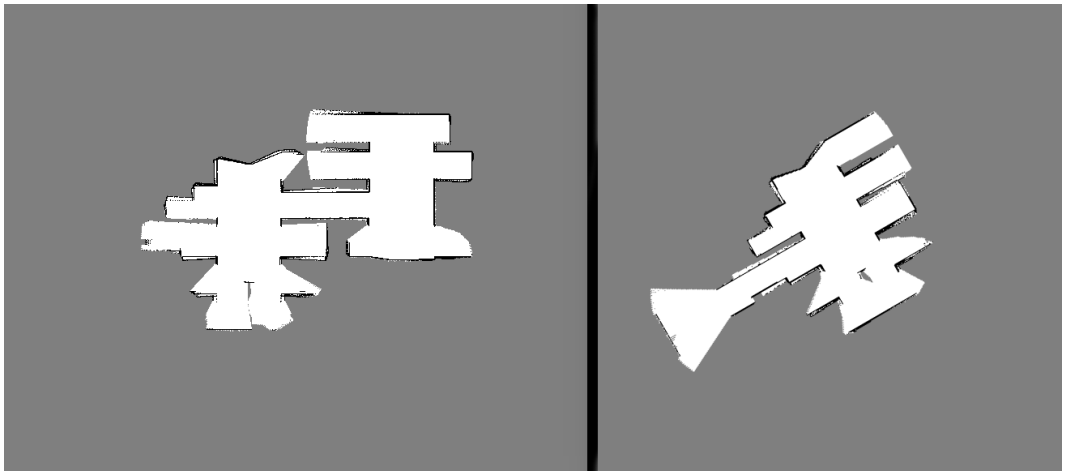

Figura 27 – Área mínima necessária a ser explorada pelos robôs azul e vermelho respectivamente utilizando o SM no segundo teste.

## 5.3 Resultados do terceiro teste em ambiente simulado

Novamente o DNIFM não foi capaz de gerar a fusão de mapas correta, como é possível ver nas Figuras 33 e 36 onde mostram, nessa ordem, os mapas de cada robô após terem percorrido os *waypoints* e o resultado de sua tentativa de combinar os mapas.

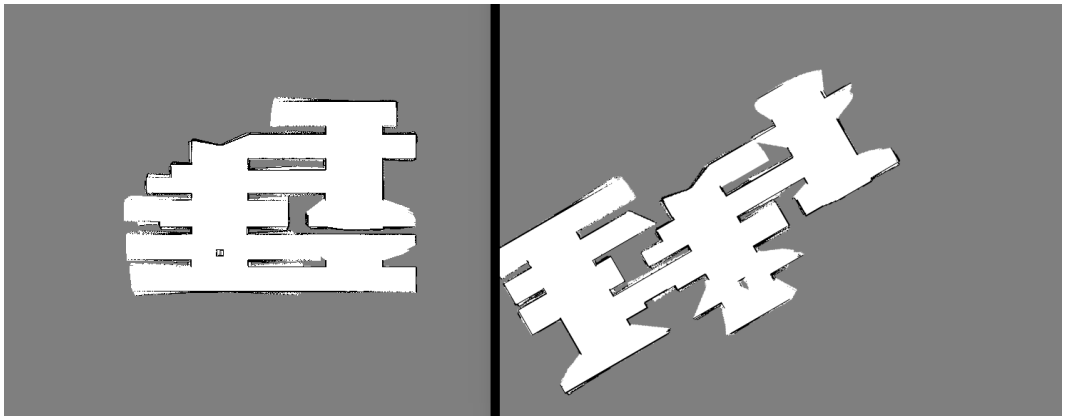

Figura 28 – Área mínima necessária a ser explorada pelos robôs azul e vermelho respectivamente utilizando o $SM^2$ no segundo teste.

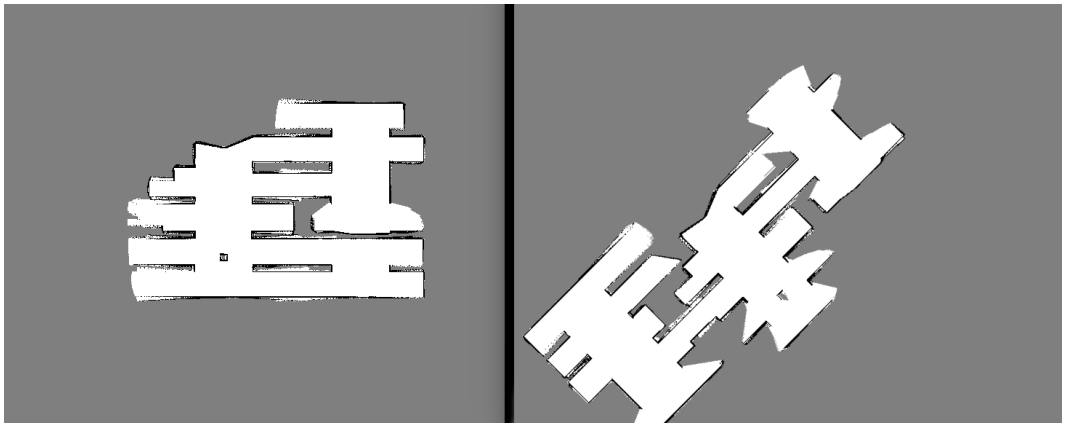

Figura 29 – Mapas de ambos os robôs após terem percorridos todos os *waypoints* no segundo teste.

As Figuras 31 e 32 mostram os mapas obtidos pelos robôs no momento em que foi obtida a fusão de mapas correta. Esses testes nos mostram que o SM foi o que obteve melhores resultados em encontrar a fusão de mapas correta com menor quantidade de informação.

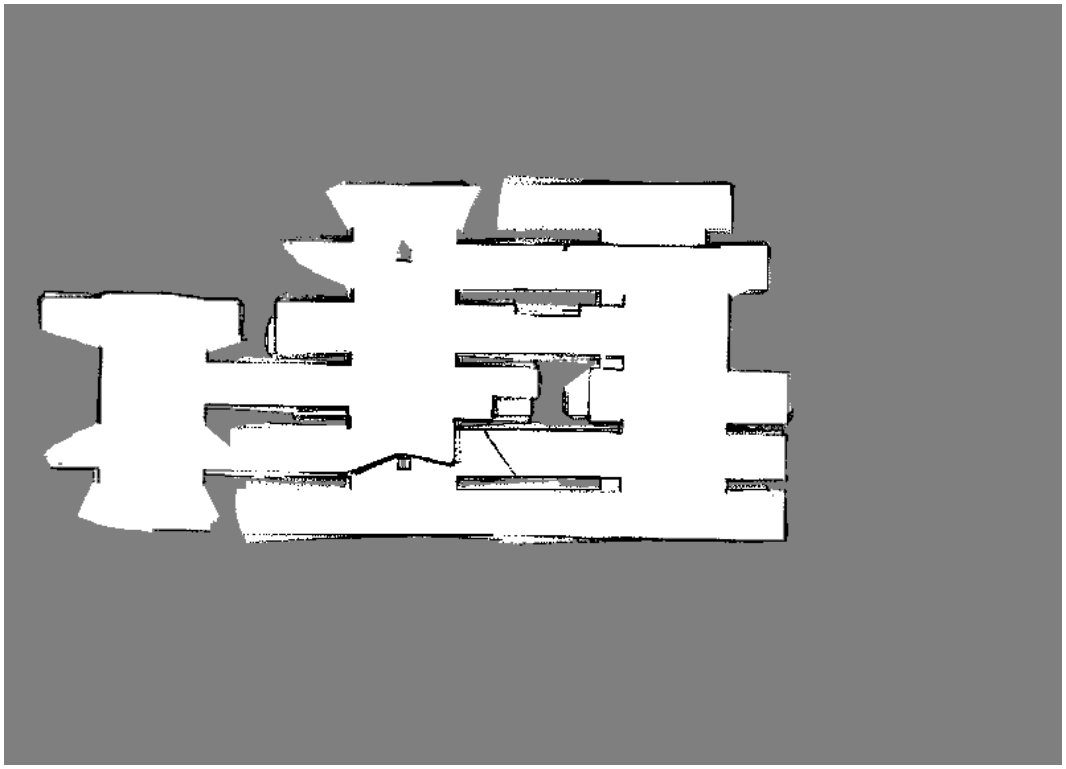

Figura 30 – Fusão de mapas incorreta gerada pelo DNIFM após os robôs terem percorridos todos os *waypoints* no segundo teste.

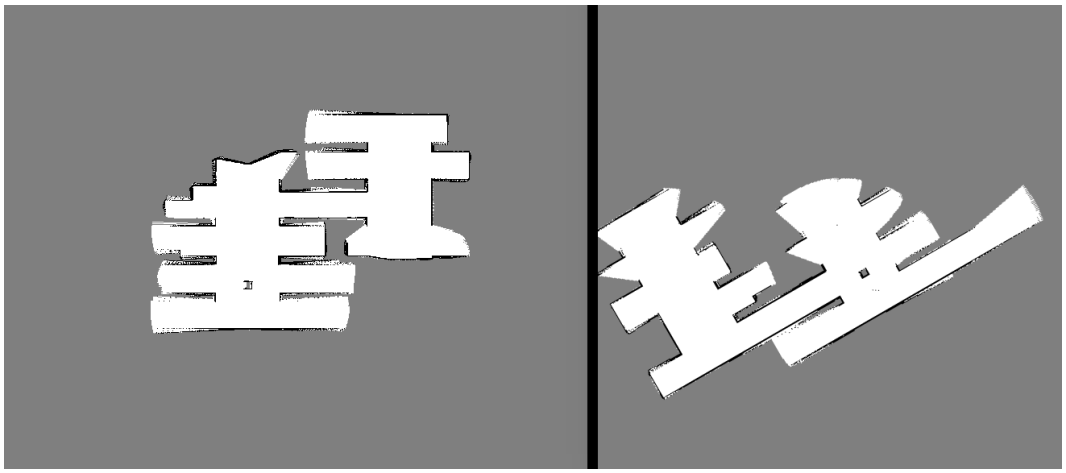

Figura 31 – Área mínima necessária a ser explorada pelos robôs azul e vermelho respectivamente utilizando o SM no terceiro teste.

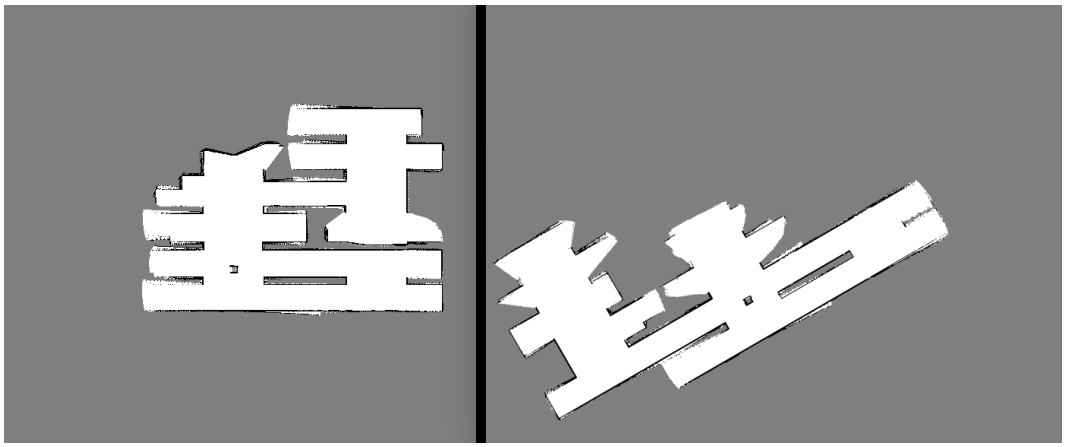

Figura 32 – Área mínima necessária a ser explorada pelos robôs azul e vermelho respectivamente utilizando o $SM^2$ no terceiro teste.

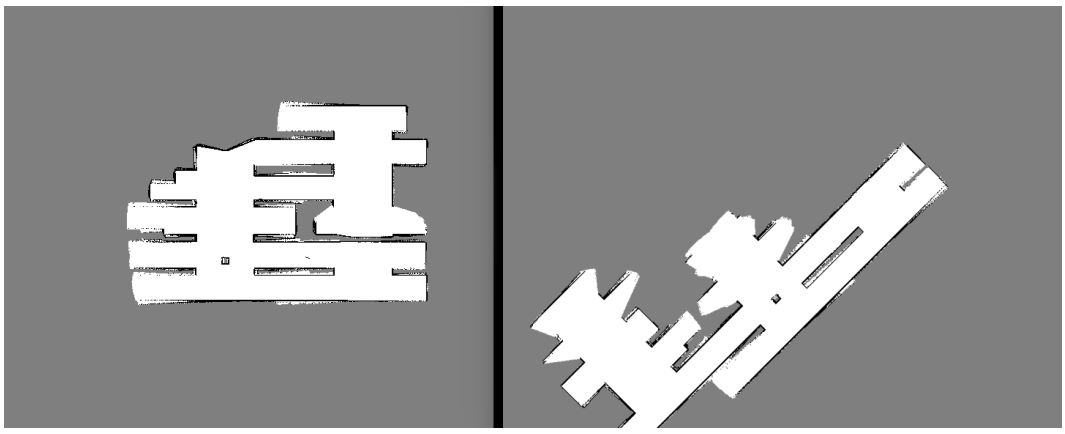

Figura 33 – Mapas de ambos os robôs após terem percorridos todos os *waypoints* utilizando o DNIFM no terceiro teste.

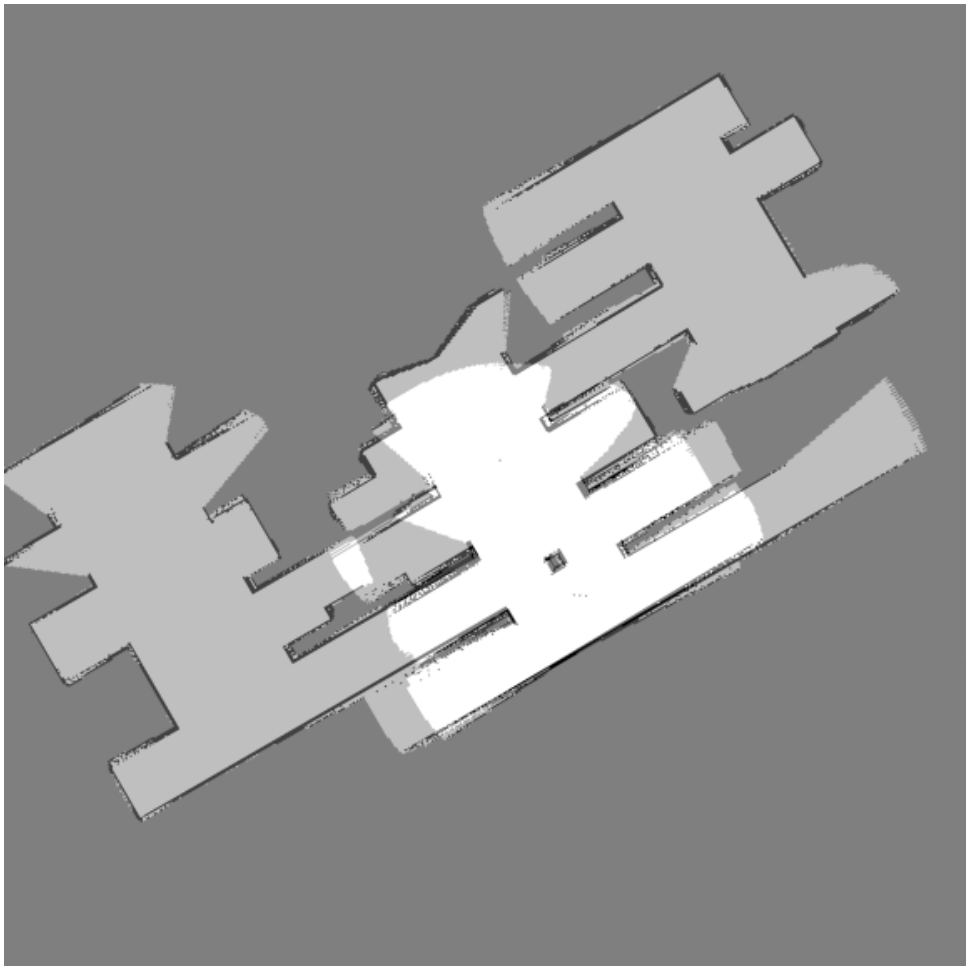

Figura 34 – Fusão de mapas gerada pelo SM no terceiro teste.

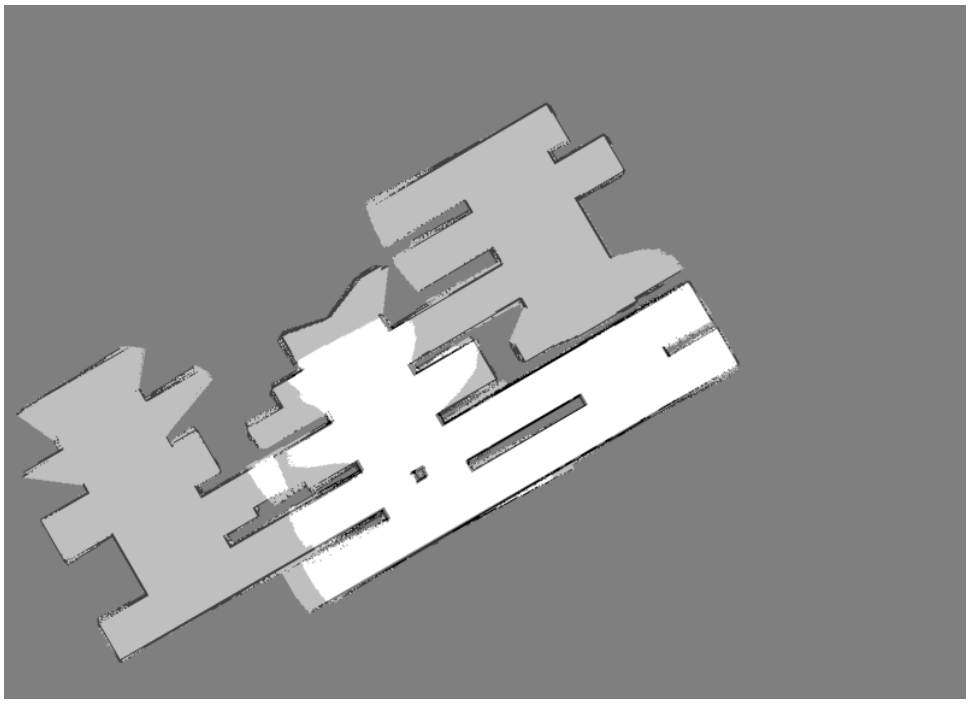

Figura 35 – Fusão de mapas gerada pelo $SM^2$ no terceiro teste.

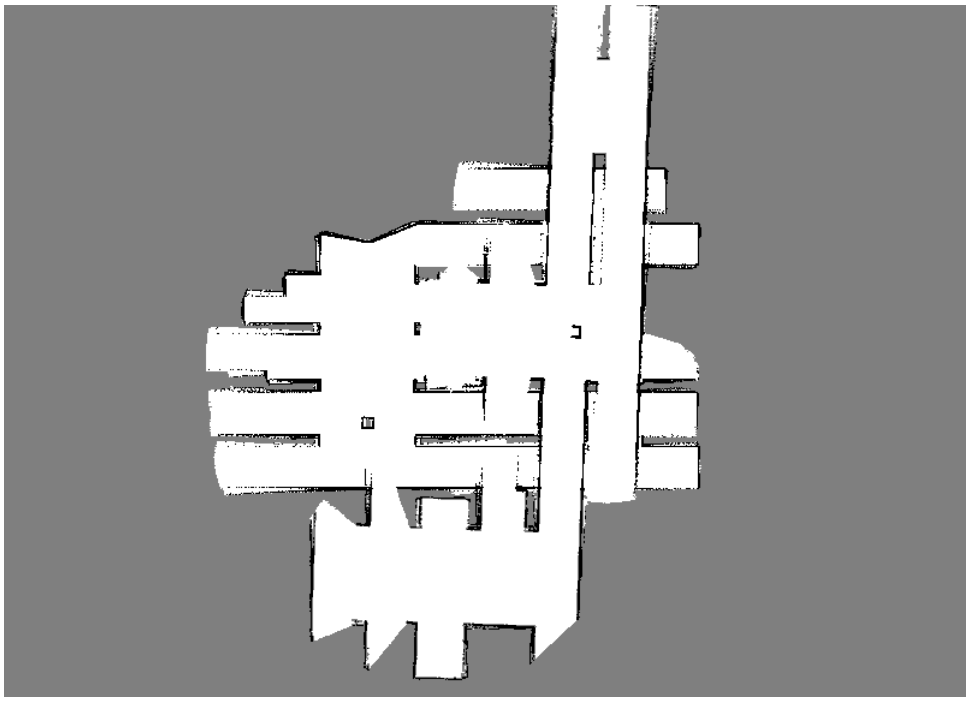

Figura 36 – Fusão de mapas incorreta gerada pelo DNIFM após os robôs terem percorridos todos os *waypoints* no terceiro teste.

# 6 SM e SM$^2$: mudança na função de custo e resultados

Durante a execução dos testes foi observado que o ponto pivô foi selecionado corretamente em ambos SM e SM$^2$, mas a transformação que possuía a menor mediana não era a correta, ou seja, a utilização da mediana como função custo para encontrar a melhor transformação nem sempre estava dando resultados satisfatórios.

Nos mapas, geralmente, há muito mais células com áreas livres do que ocupadas, já que as áreas ocupadas são apenas paredes ou objetos fixos no ambiente, enquanto as áreas livres é todo o ambiente em que o robô pode percorrer. Sendo assim, se utilizar o índice de aceitação como função custo, um mapa completamente vazio combinado com outro mapa qualquer pode haver um alto índice de aceitação. Para evitar que isso ocorram, uma nova função que da maior importância às células ocupadas foi elaborada. Essa função foi definida como:

$$C_o = O(M_1, M_2) - dis(M_1, M_2) \tag{6.1}$$

Onde $O(M_1, M_2)$ é a quantidade de células $M_1$ e $M_2$ que estão ambas ocupadas, e $dis(M_1, M_2)$ é a mesma função explicitada na equação 3.1.

O objetivo ao usar $C_o$ é maximizar a quantidade de obstáculos que coincidem (primeiro termo na equação 6.1 ) e minimizar as células que não coincidem (segundo termo). O algoritmo 3 a seguir detalha método $SM_2$ com a nova função de custo.

Os testes da seção 5 foram feitos novamente para os algoritmos $SM$ e $SM^2$ utilizando a nova função custo.

As Figuras (37 e 39), (41 e 43) e (45 e 47) ilustram a área necessária a ser explorada para que aconteça uma fusão de mapas bem sucedida pelo SM e SM$^2$ em cada um dos testes.

E as Figuras (38 e 40), (42 e 44) e (46 e 48) mostram os mapas gerados após a fusão dos respectivos mapas.

Utilizando essa nova função de custo, além da área de exploração necessária serem menores, em ambos SM e SM$^2$, essas também foram menores do que a área necessária na exploração feita pelo DNIFM no primeiro teste(Figura 26).

**Algoritmo 3** $SM^2$ com nova função custo.

**Input:** $\text{Mapa}_1$, $\text{Mapa}_2$
**Output:** Matriz de transformação $R_m$ de $\text{Mapa}_1$ para $\text{Mapa}_2$

$R_m \leftarrow \text{Identidade}_{3\times3}$
**for** $i \in \{1, 2\}$ **do**
    BLUR($\text{Map}_1$)
    BLUR($\text{Map}_2$)
    $\text{keypoints}_1 \leftarrow \text{SIFT}(\text{Map}_1)$
    $\text{keypoints}_2 \leftarrow \text{SIFT}(\text{Map}_2)$
    matches $\leftarrow$ FLANN($\text{keypoints}_1$, $\text{keypoints}_2$)
    best_match $\leftarrow$ SELECT_BEST_MATCH(matches)
    $\text{pivot}_1$, $\text{pivot}_2 \leftarrow$ best_match.pair
    $t_{12} \leftarrow \text{pivot}_2 - \text{pivot}_1$
    best_score $\leftarrow 0$
    **for** match in matches **do**
        **if** (match $\neq$ best_match) **then**
            $p_1$, $p_2 \leftarrow$ match.pair
            $v_1 \leftarrow p_1 - \text{pivot}_1$
            $v_2 \leftarrow p_2 - \text{pivot}_2$
            $\theta \leftarrow$ CALCULATE_ANGLE($v_1$, $v_2$)
            ratio $\leftarrow v_1$.length/$v_2$.length
            score $\leftarrow$ CALCULATE_$C_0$($M_1$, $M_2$, $pivot_1$, $\theta$, $t_{12}$)
            **if** (score > best_score) **then**
                best_score $\leftarrow$ score
                selected_angle $\leftarrow \theta$
                selected_ratio $\leftarrow$ ratio
            **end if**
        **end if**
    **end for**
    $T \leftarrow$ TRANSLATION_MATRIX($\text{pivot}_1$, $\text{pivot}_2$)
    $A \leftarrow$ AFFINE_MATRIX($\text{pivot}_2$, selected_ratio, selected_angle)
    $R_m \leftarrow R_m \times A \times T$
**end for**
**return** $R_m$

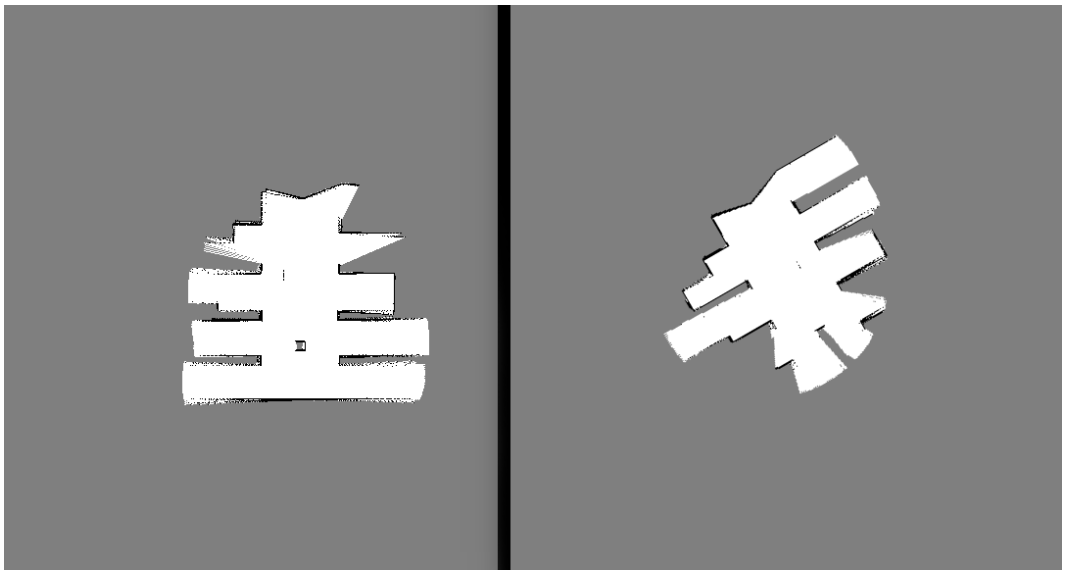

Figura 37 – Área mínima necessária a ser explorada pelos robôs azul e vermelho respectivamente no primeiro teste utilizando o SM com nova função custo.

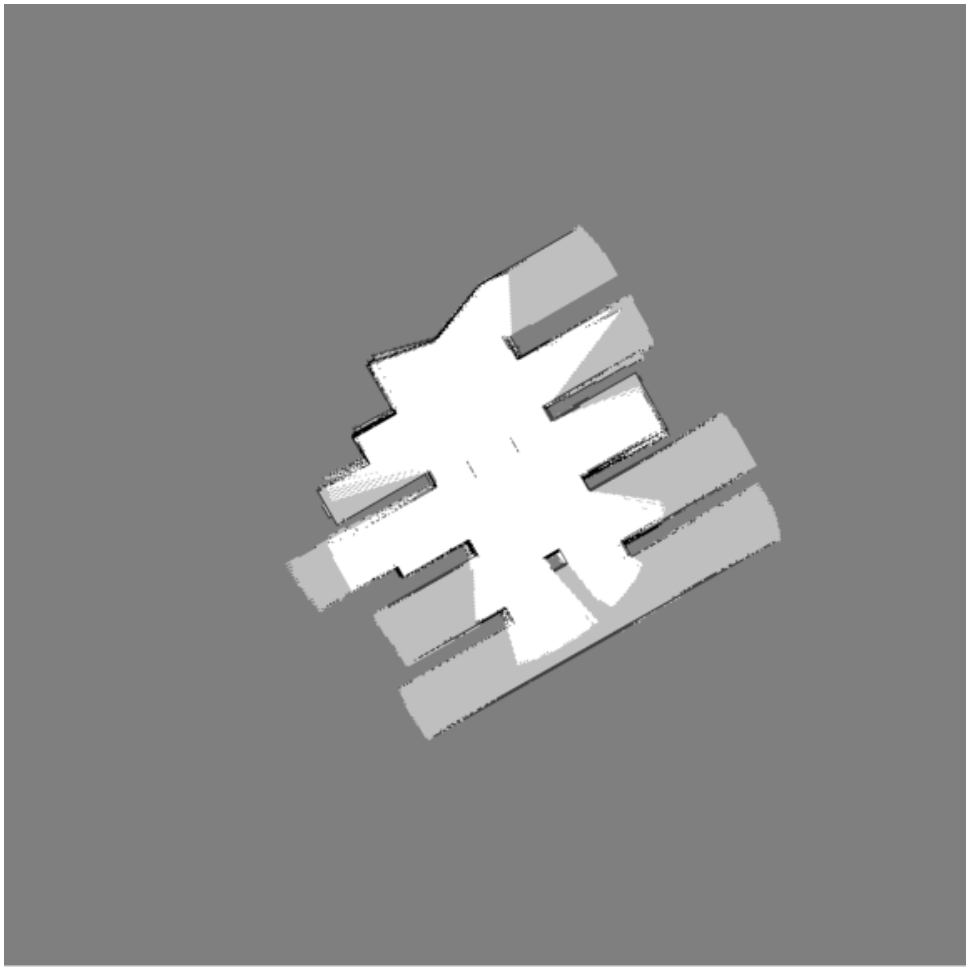

Figura 38 – Fusão de mapas gerado pelo SM no primeiro teste utilizando a nova função de custo.

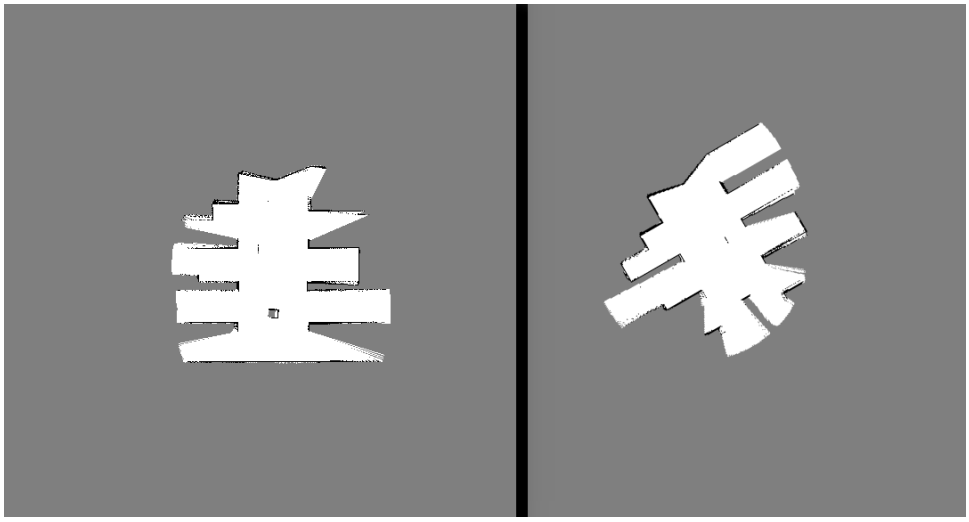

Figura 39 – Área mínima necessária a ser explorada pelos robôs azul e vermelho respectivamente utilizando o SM² com nova função custo.

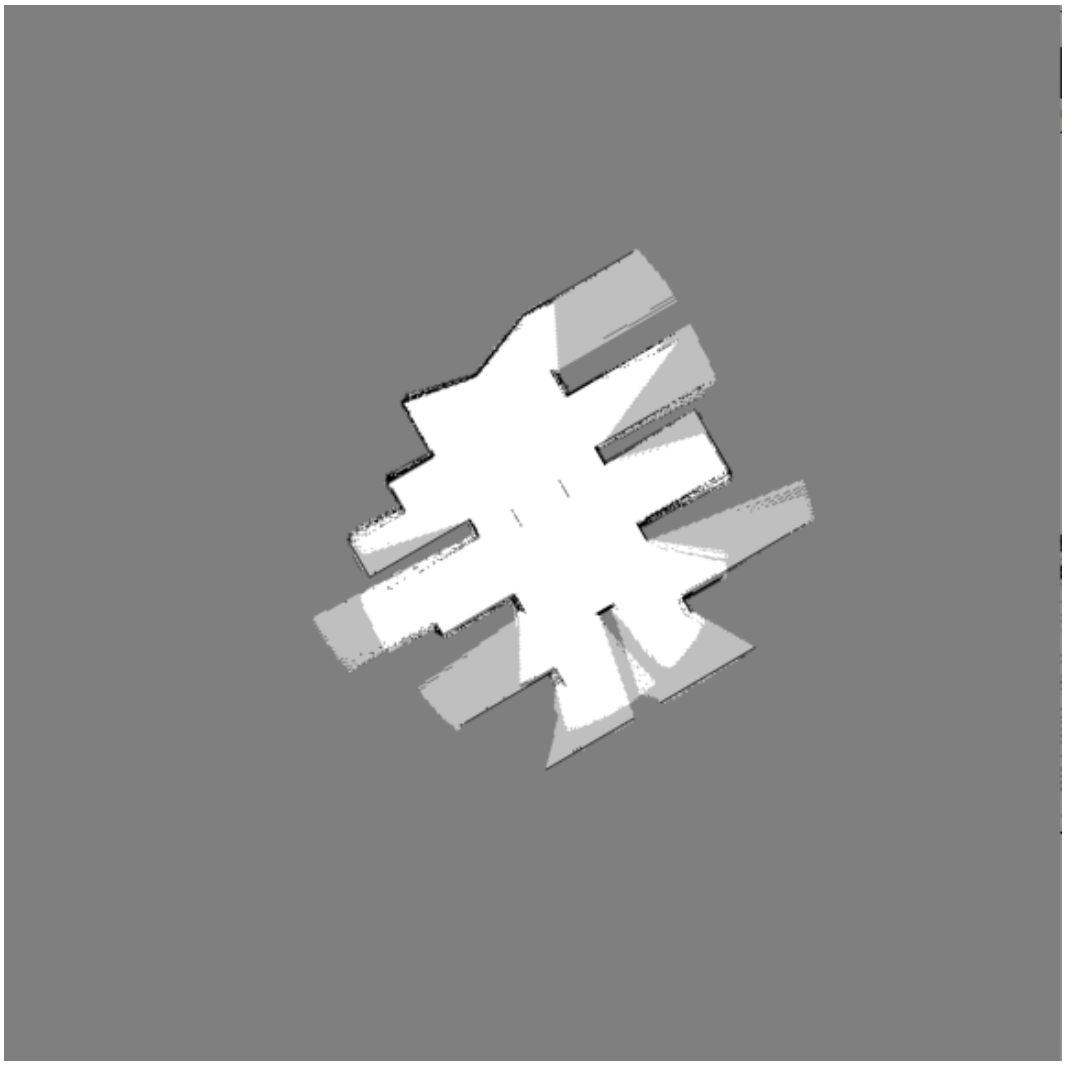

Figura 40 – Fusão de mapas gerado pelo SM² no primeiro teste utilizando a nova função de custo.

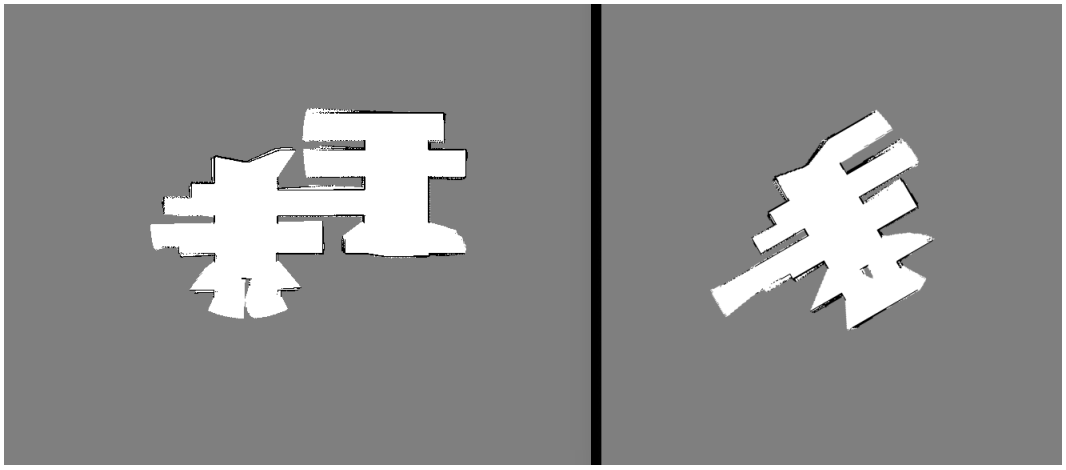

Figura 41 – Área mínima necessária a ser explorada pelos robôs azul e vermelho respectivamente no segundo teste utilizando o SM com nova função custo.

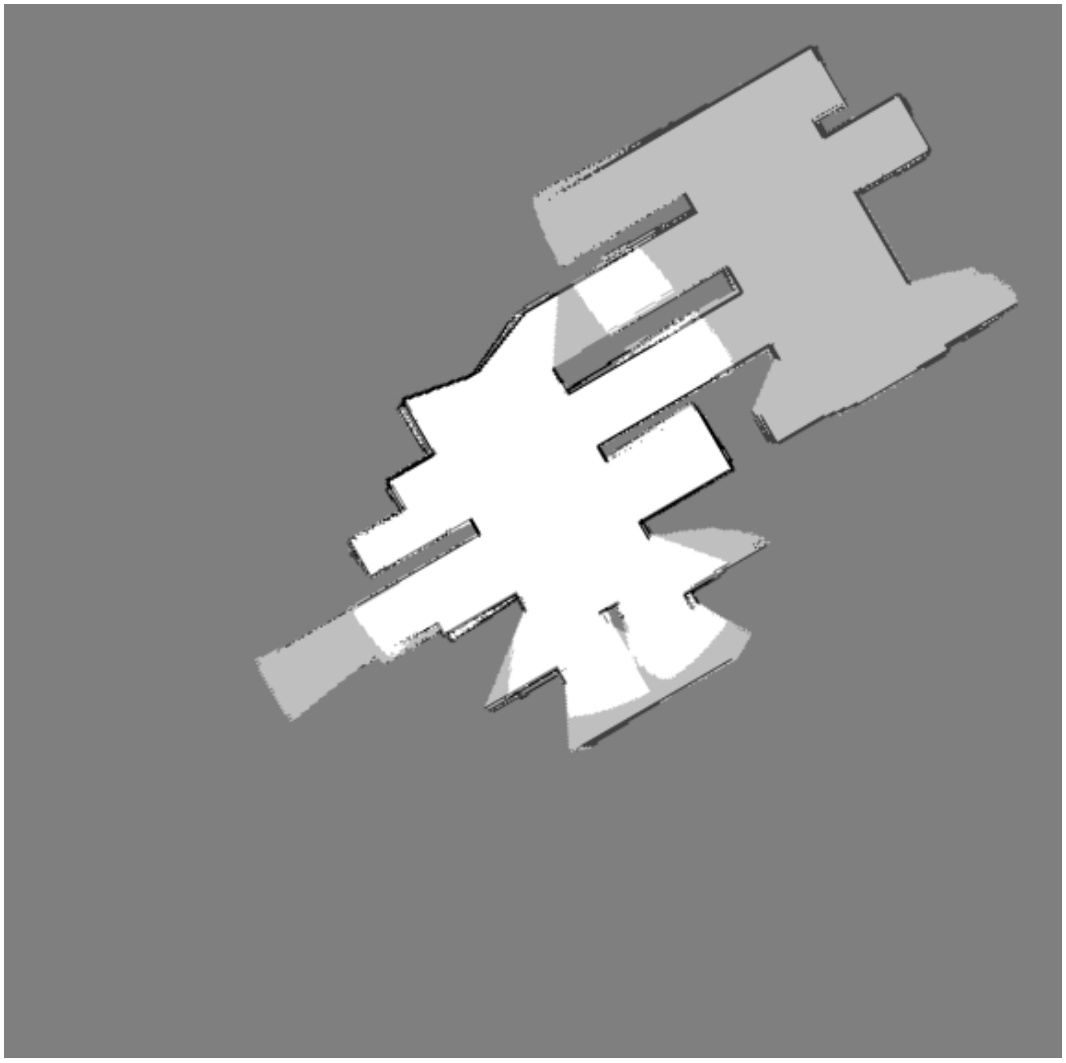

Figura 42 – Fusão de mapas gerado pelo SM no segundo teste utilizando a nova função de custo.

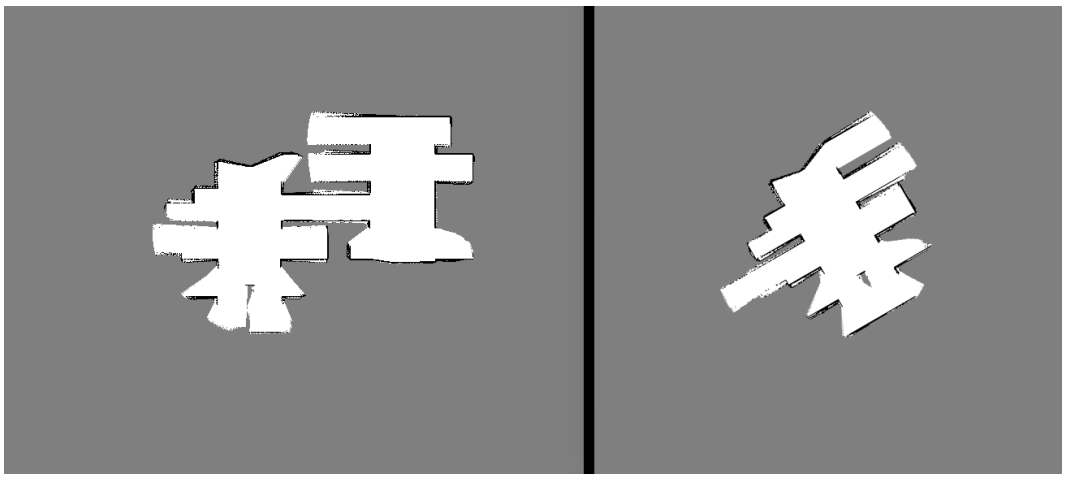

Figura 43 – Área mínima necessária a ser explorada pelos robôs azul e vermelho respectivamente no primeiro teste utilizando o SM² com nova função custo.

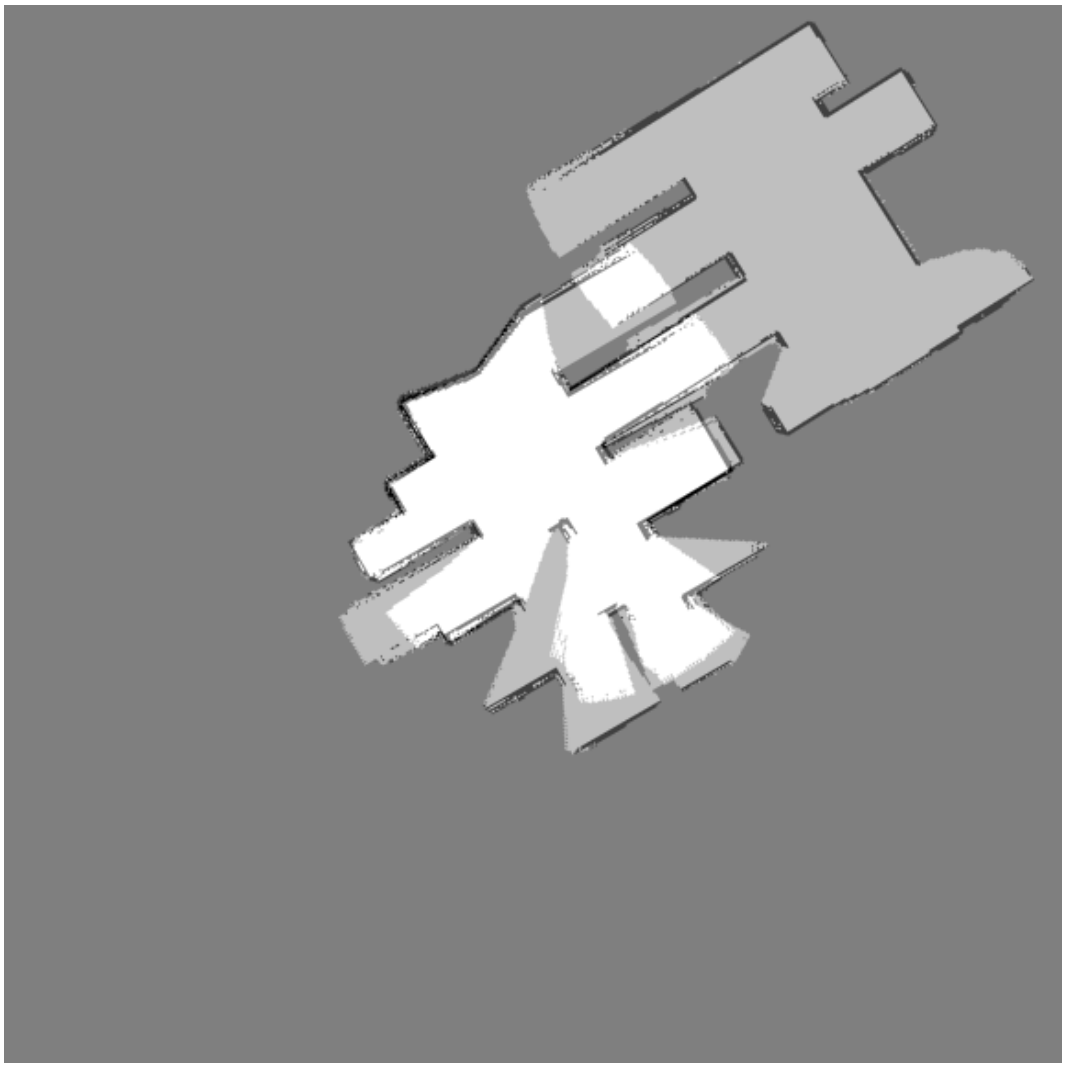

Figura 44 – Fusão de mapas gerado pelo SM² no segundo teste utilizando a nova função de custo.

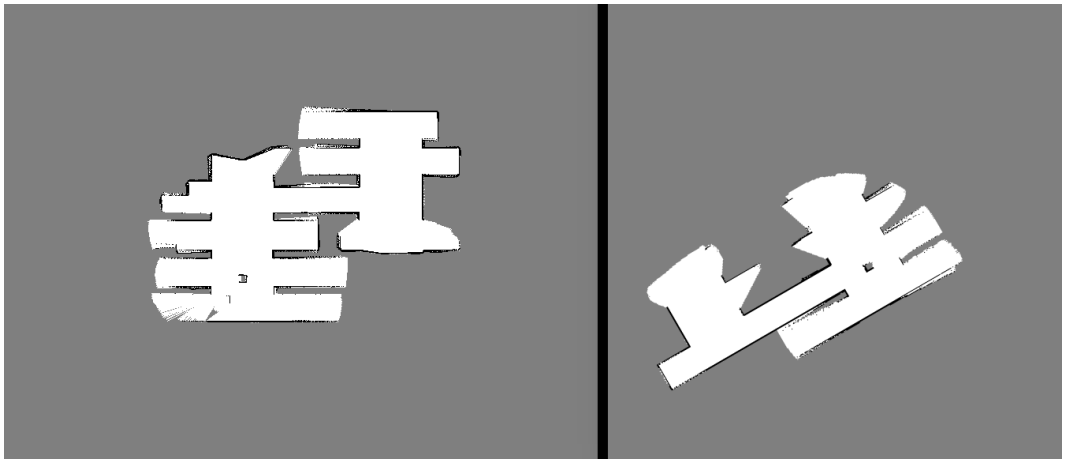

Figura 45 – Área mínima necessária a ser explorada pelos robôs azul e vermelho respectivamente no terceiro teste utilizando o SM com nova função custo.

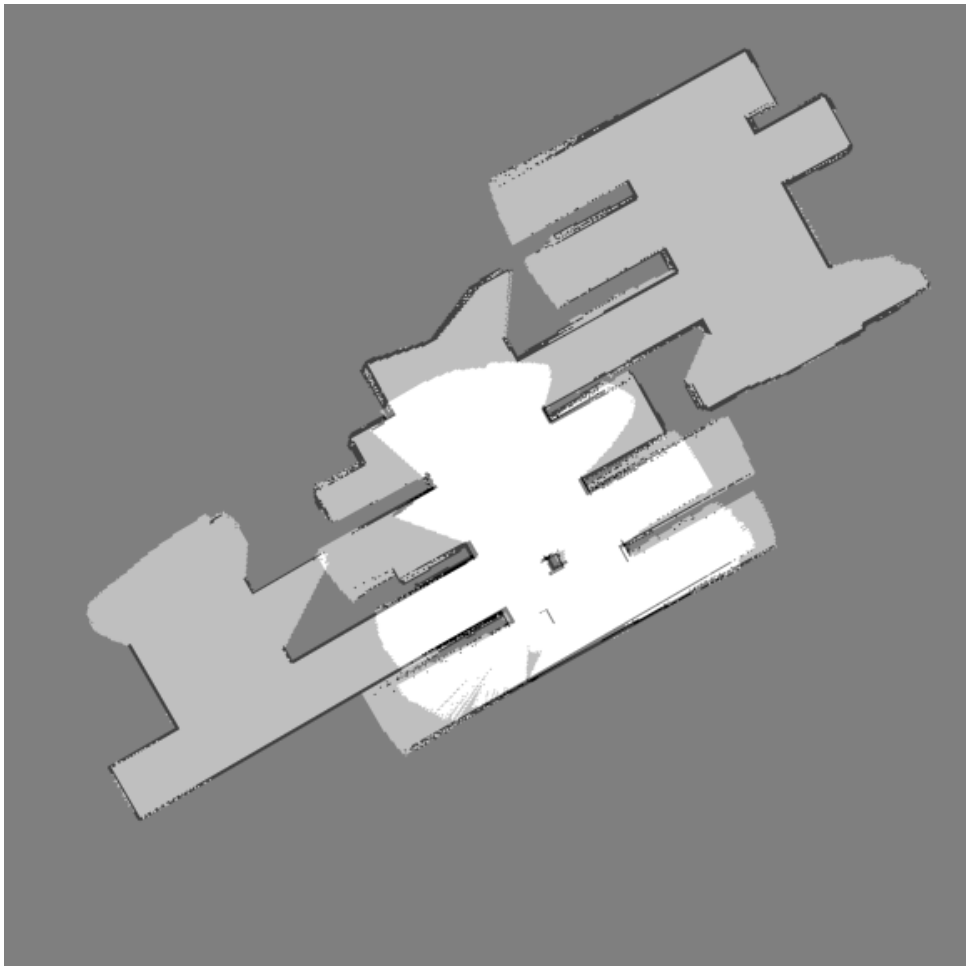

Figura 46 – Fusão de mapas gerado pelo SM no terceiro teste utilizando a nova função de custo.

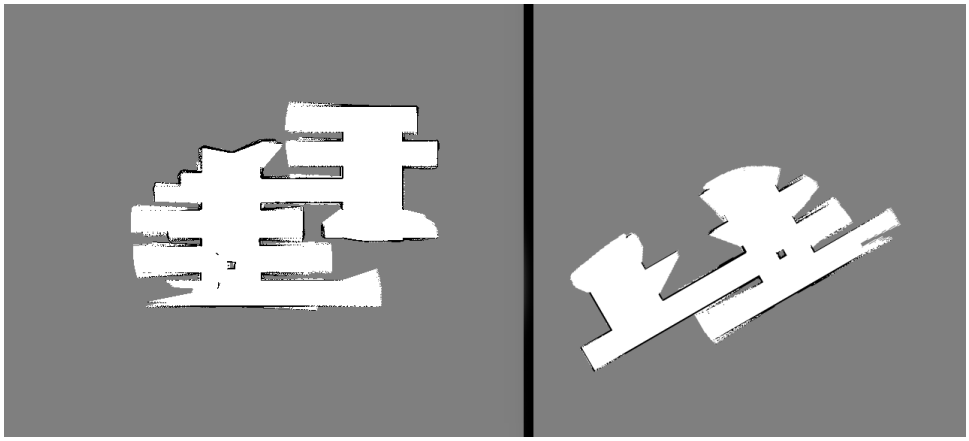

Figura 47 – Área mínima necessária a ser explorada pelos robôs azul e vermelho respectivamente no terceiro teste utilizando o SM² com nova função custo.

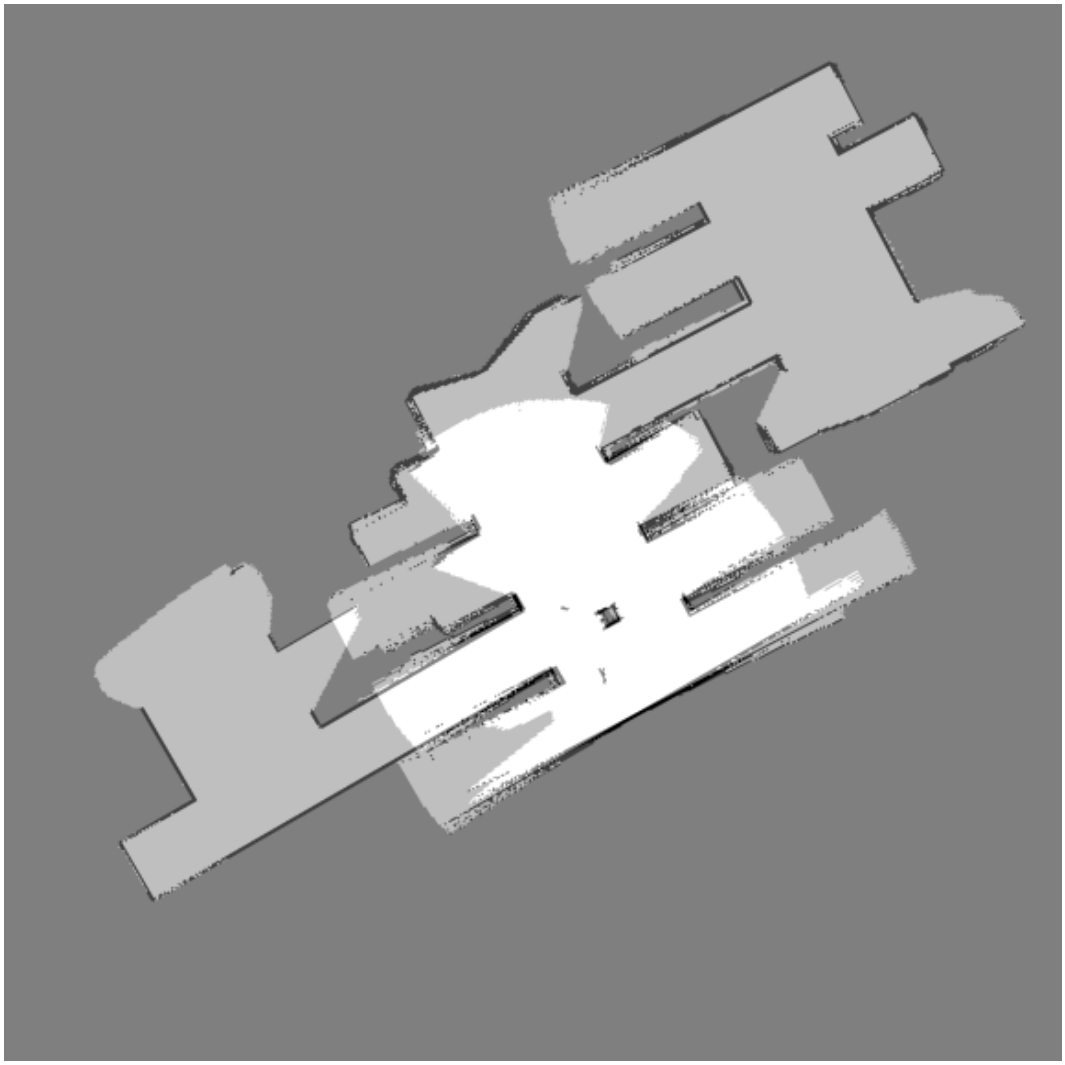

Figura 48 – Fusão de mapas gerado pelo SM² no terceiro teste utilizando a nova função de custo.

# Conclusão

Foi apresentado o progresso de desenvolvimento de um algoritmo eficiente para a unir dois mapas de ocupação, que podem ter escalas diferentes. Em cada iteração nova do desenvolvimento, algumas melhoria foi feita, seja para diminuir o tempo de processamento ou melhorar a precisão na tarefa.

Mesmo com as iterações para melhorar o algoritmo e deixá-lo mais robusto, ele ainda possui a limitação de que o pivô deve representar o mesmo lugar físico nos dois mapas durante cada iteração. Se essa limitação for superada, pode ser necessário um nível menor de sobreposição entre os mapas para poder mesclá-los.

Ainda com o $SM^2$ tenha se mostrado mais lento do que o SM, por uma questão de como foi projetado, este se mantém mais rápido do que o DNIFM. Para casos onde a reposta em tempo real é mais importante, o SM é mais indicado para ser utilizado. Mas se a precisão entre a união dos mapas for prioridade, o $SM^2$ chega mais próximo de ter uma união perfeita. Durante os testes em ambiente simulado, utilizando a função custo da mediana, em apenas um dos testes, o DNIFM necessitou de menos informação para fazer uma fusão bem sucedida do que o SM e o $SM^2$. No entanto ao aprimorar a funçao de custo do SM e do $SM^2$, ambos necessitaram menos informação sobre os mapas do que o DNIFM em todos os testes.

# Referências

ARULAMPALAM, M. S. et al. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, Ieee, v. 50, n. 2, p. 174–188, 2002. Citado na página 21.

BATALIN, M. A.; SUKHATME, G. S.; HATTIG, M. Mobile robot navigation using a sensor network. In: IEEE. *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. [S.l.], 2004. v. 1, p. 636–641. Citado 2 vezes nas páginas 17 e 19.

BIRK, A.; CARPIN, S. Merging occupancy grid maps from multiple robots. *Proceedings of the IEEE*, IEEE, v. 94, n. 7, p. 1384–1397, 2006. Citado na página 30.

BROWN, R. G.; HWANG, P. Y. et al. *Introduction to random signals and applied Kalman filtering*. [S.l.]: Wiley New York, 1992. v. 3. Citado na página 21.

CADENA, C. et al. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, IEEE, v. 32, n. 6, p. 1309–1332, 2016. Citado na página 21.

CARPIN, S. Fast and accurate map merging for multi-robot systems. *Autonomous Robots*, Springer, v. 25, n. 3, p. 305–316, 2008. Citado 4 vezes nas páginas 17, 23, 24 e 32.

CARPIN, S.; BIRK, A.; JUCIKAS, V. On map merging. *Robotics and autonomous systems*, Elsevier, v. 53, n. 1, p. 1–14, 2005. Citado na página 23.

COOPER, S.; DURRANT-WHYTE, H. A kalman filter model for gps navigation of land vehicles. In: IEEE. *Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on*. [S.l.], 1994. v. 1, p. 157–163. Citado na página 23.

ELFES, A. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation*, IEEE, v. 3, n. 3, p. 249–265, 1987. Citado 2 vezes nas páginas 17 e 19.

ERINC, G.; CARPIN, S. Anytime merging of appearance-based maps. *Autonomous Robots*, Springer, v. 36, n. 3, p. 241–256, 2014. Citado na página 17.

FERRAO, V.; VINHAL, C.; JR, G. da C. An occupancy grid map merging algorithm invariant to scale, rotation and translation. In: *Brazilian Conference on Intelligent Systems*. [S.l.: s.n.], 2017. DOI:10.1109/BRACIS.2017.69. Citado na página 17.

FOX, D. et al. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, IEEE, v. 94, n. 7, p. 1325–1339, 2006. Citado 2 vezes nas páginas 17 e 23.

GERKEY, B.; VAUGHAN, R. T.; HOWARD, A. The player/stage project: Tools for multi-robot and distributed sensor systems. In: *Proceedings of the 11th international conference on advanced robotics*. [S.l.: s.n.], 2003. v. 1, p. 317–323. Citado 2 vezes nas páginas 21 e 42.

HOWARD, A.; PARKER, L. E.; SUKHATME, G. S. Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. *The International Journal of Robotics Research*, SAGE Publications, v. 25, n. 5-6, p. 431–447, 2006. Citado na página 23.

KIM, S.-H. et al. Outdoor navigation of a mobile robot using differential gps and curb detection. In: IEEE. *Robotics and Automation, 2007 IEEE International Conference on*. [S.l.], 2007. p. 3414–3419. Citado na página 23.

KOENIG, N. P.; HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: CITESEER. *IROS*. [S.l.], 2004. v. 4, p. 2149–2154. Citado na página 21.

LEHEY, G. *The complete FreeBSD: documentation from the source*. [S.l.]: "O'Reilly Media, Inc.", 2003. Citado na página 23.

LINDEBERG, T. Scale invariant feature transform. *Scholarpedia*, v. 7, n. 5, p. 10491, 2012. Citado na página 17.

MA, X. et al. Adaptive genetic algorithm for occupancy grid maps merging. In: IEEE. *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*. [S.l.], 2008. p. 5716–5720. Citado na página 23.

MARJOVI, A.; MARQUES, L. Multi-robot topological exploration using olfactory cues. In: *Distributed Autonomous Robotic Systems*. [S.l.]: Springer, 2013. p. 47–60. Citado na página 17.

MATÉS, J. M. et al. Roles of dioxins and heavy metals in cancer and neurological diseases using ros-mediated mechanisms. *Free Radical Biology and Medicine*, Elsevier, v. 49, n. 9, p. 1328–1341, 2010. Citado na página 21.

MITCHELL, W. C.; STANIFORTH, A.; SCOTT, I. *Analysis of ackermann steering geometry*. [S.l.], 2006. Citado na página 22.

MURRAY, D.; LITTLE, J. J. Using real-time stereo vision for mobile robot navigation. *Autonomous Robots*, Springer, v. 8, n. 2, p. 161–171, 2000. Citado 2 vezes nas páginas 17 e 19.

ÖZKUCUR, N. E.; AKIN, H. L. Cooperative multi-robot map merging using fast-slam. In: SPRINGER. *Robot Soccer World Cup*. [S.l.], 2009. p. 449–460. Citado na página 23.

PARKER, L. E. Current state of the art in distributed autonomous mobile robotics. In: *Distributed Autonomous Robotic Systems 4*. [S.l.]: Springer, 2000. p. 3–12. Citado na página 17.

QUIGLEY, M. et al. Ros: an open-source robot operating system. In: KOBE, JAPAN. *ICRA workshop on open source software*. [S.l.], 2009. v. 3, n. 3.2, p. 5. Citado na página 42.

QUIGLEY, M.; GERKEY, B.; SMART, W. D. *Programming Robots with ROS: a practical introduction to the Robot Operating System*. [S.l.]: "O'Reilly Media, Inc.", 2015. Citado na página 21.

RUIZ, E. et al. Development of a control platform for the mobile robot roomba using ros and a kinect sensor. In: IEEE. *Robotics Symposium and Competition (LARS/LARC), 2013 Latin American*. [S.l.], 2013. p. 55–60. Citado na página 21.

SICILIANO, B.; KHATIB, O. *Springer handbook of robotics*. [S.l.]: Springer, 2016. Citado na página 17.

SUKKARIEH, S.; NEBOT, E. M.; DURRANT-WHYTE, H. F. A high integrity imu/gps navigation loop for autonomous land vehicle applications. *IEEE Transactions on Robotics and Automation*, IEEE, v. 15, n. 3, p. 572–578, 1999. Citado na página 23.

THRUN, S. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, Elsevier, v. 99, n. 1, p. 21–71, 1998. Citado 2 vezes nas páginas 17 e 19.

THRUN, S.; MONTEMERLO, M. The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, SAGE Publications, v. 25, n. 5-6, p. 403–429, 2006. Citado na página 21.

VAUGHAN, R. Massively multi-robot simulation in stage. *Swarm intelligence*, Springer, v. 2, n. 2-4, p. 189–208, 2008. Citado na página 22.