

Dissertação apresentada à Pró-Reitoria de Pós-Graduação do Instituto Tecnológico de Aeronáutica, como parte dos requisitos para obtenção do título de Mestre em Ciências no Programa de Pós-Graduação em Engenharia Eletrônica e Computação, Área de Sistemas e Controle.

Luciana Araujo Lemos

**LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS
USANDO *SCANNER A LASER* E
CORRESPONDÊNCIA ENTRE AS CARACTERÍSTICAS
DO AMBIENTE**

Dissertação aprovada em sua versão final pelos abaixo assinados:

Prof. Dr. Cairo Lúcio Nascimento Júnior

Orientador

Profa. Dra. Emília Villani

Pró-Reitora de Pós-Graduação

Campo Montenegro
São José dos Campos, SP - Brasil
2022

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

Araujo Lemos, Luciana

Localização e mapeamento simultâneos usando *Scanner a Laser* e correspondência entre as características do ambiente / Luciana Araujo Lemos.

São José dos Campos, 2022.

130f.

Dissertação de Mestrado – Curso de Engenharia Eletrônica e Computação. Área de Sistemas e Controle – Instituto Tecnológico de Aeronáutica, 2022. Orientador: Prof. Dr. Cairo Lúcio Nascimento Júnior.

1. Dinâmica de robôs.
 2. Localização e mapeamento simultâneos.
 3. Navegação autônoma.
 4. Exploração.
 5. Algoritmos.
 6. Robótica.
 7. Controle.
- I. Instituto Tecnológico de Aeronáutica.
II. Título.

REFERÊNCIA BIBLIOGRÁFICA

ARAUJO LEMOS, Luciana. **Localização e mapeamento simultâneos usando Scanner a Laser e correspondência entre as características do ambiente**. 2022. 130f.
Dissertação de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

NOME DA AUTORA: Luciana Araujo Lemos

TÍTULO DO TRABALHO: Localização e mapeamento simultâneos usando *Scanner a Laser* e correspondência entre as características do ambiente.

TIPO DO TRABALHO/ANO: Dissertação / 2022

É concedida ao Instituto Tecnológico de Aeronáutica permissão para reproduzir cópias desta dissertação e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. A autora reserva outros direitos de publicação e nenhuma parte desta dissertação pode ser reproduzida sem a autorização da autora.

Luciana Araujo Lemos
Praça Marechal Eduardo Gomes, 50
12228-900 – São José dos Campos – SP

**LOCALIZAÇÃO E MAPEAMENTO SIMULTÂNEOS
USANDO *SCANNER A LASER* E
CORRESPONDÊNCIA ENTRE AS CARACTERÍSTICAS
DO AMBIENTE**

Luciana Araujo Lemos

Composição da Banca Examinadora:

Prof. Dr. Osamu Saotome	Presidente	-	ITA
Prof. Dr. Cairo Lúcio Nascimento Júnior	Orientador	-	ITA
Prof. Dr. Jacques Waldmann	Membro Interno	-	ITA
Prof. Dr. Benedito Isaías Lima Fuly	Membro Externo	-	UNIFEI

ITA

Aos meus pais, à minha irmã e a toda
minha família pelo apoio e motivação de
completar este trabalho.

Agradecimentos

Agradeço aos meus pais Maria Lucia e Elizeu Jr., pelo apoio e incentivo às minhas escolhas e pelos conselhos que me fizeram crescer e amadurecer tanto pessoalmente quanto profissionalmente. Obrigada por me ensinarem os valores e princípios desde cedo que vou levar para o resto da minha vida e que foram de extrema importância para me tornar a pessoa que sou. À minha irmã Ana Julia, pelos conselhos que foram essenciais para minhas tomadas de decisão. À minha família por me apoiar e ensinar que a base de tudo é o conhecimento, sendo de suma importância para a minha evolução pessoal e profissional e para a continuação na busca pelos meus sonhos e objetivos.

Ao meu orientador Prof. Dr. Cairo L. Nascimento Jr., pela dedicação, conselhos e ensinamentos.

Aos meus amigos e colegas do Laboratório de Máquinas Inteligentes, Juan Ramón e Luiz Eugênio Araújo pelas trocas de conhecimento constantes, pelos momentos de lazer, pelo esclarecimento das dúvidas e por me ajudarem a tornar este projeto realidade. Aos colegas Luiz Eugênio e Larissa pelo desenvolvimento do *hardware* do robô e *software* dos sensores embarcados no robô móvel que foram utilizados e serviram de base para o projeto.

Ao professor Dr. Luciano Buonocore pela construção das paredes de madeira utilizadas no experimento real.

Aos professores membros da banca, pelas sugestões de melhorias e pela relevante revisão do trabalho para a versão final desta dissertação.

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pela concessão de bolsa ao longo do mestrado.

*"If I have seen further than others,
it is because I stood on the shoulders of giants."*
— SIR ISAAC NEWTON

Resumo

Na robótica móvel, um dos problemas mais relevantes é o de SLAM (*Simultaneous Localization And Mapping*) cuja solução visa estimar a postura (posição e orientação) do robô e mapear o ambiente em que o robô se movimenta de forma simultânea. Este trabalho de dissertação propõe uma solução para o problema de localização e mapeamento simultâneos em duas dimensões usando a correspondência entre certas características do ambiente à medida que o robô navega por ele. Essa técnica é conhecida como *feature matching* e neste trabalho é usada a correspondência entre os segmentos de reta do ambiente.

No algoritmo de SLAM proposto, o robô navega pelo ambiente com guiamento manual e captura em pontos selecionados um *scan* local, ou seja, o conjunto de pontos gerados pela varredura de 360° do medidor de distância. O algoritmo RANSAC usa esse conjunto de pontos para extrair os segmentos de reta que representam os trechos das paredes do ambiente que são detectados pelo robô nessa parada. Para cada par de *scans* consecutivos é feita uma busca numérica, usando um algoritmo de minimização, pelo par {rotação, translação} que gera a melhor correspondência entre os segmentos de reta desses *scans*. Esse par {rotação, translação} é usado para o cálculo da postura do robô no Sistema de Coordenadas Global. É também proposto um algoritmo para junção dos segmentos inseridos no mapa global durante o procedimento de SLAM. Para fechamento de laço (*loop closure*), nos *scans* locais são detectados os cantos e as extremidades isoladas das novas paredes. As coordenadas (no Sistema de Coordenadas Global) desses novos marcos são armazenadas. Quando esses marcos são detectados novamente nos próximos *scans*, as medidas de distância e ângulo do robô para esses marcos são então usadas para recalcular a postura do robô. Para implementação e testes da solução proposta, foi usado um robô móvel equipado com um *scanner a laser* 2D e um computador embarcado Raspberry Pi 3B *plus*. São apresentados, primeiramente, experimentos simulados no *software* Gazebo do robô em um ambiente de 44 m² com 9 e 10 paredes. É também apresentado um experimento usando o robô real em um ambiente com 10 paredes de 41,44 m². Nesses experimentos são apresentados resultados sem e com fechamento de laço. Esses resultados demonstram o bom desempenho do algoritmo proposto nesses ambientes.

Abstract

In mobile robotics, one of the most relevant problems is SLAM (Simultaneous Localization And Mapping) whose solution aims to estimate simultaneously the pose (position and orientation) of the robot and map the environment in which the robot moves. This dissertation proposes a solution for the simultaneous localization and mapping problem in two dimensions using the correspondence between certain characteristics of the environment as the robot navigates through it. This technique is known as feature matching, and in this work the correspondence between line segments present in the environment is used.

In the proposed SLAM algorithm, the robot navigates in the environment with manual guidance and captures at selected points a local scan, i.e., the set of points generated by a 360° scan of the distance measurement device. The RANSAC algorithm uses this set of points to extract the line segments representing the sections of the environment walls that are detected by the robot at this stop. For each pair of consecutive scans a numerical search is performed, using a minimization algorithm, for the pair {rotation, translation} that generates the best match between the line segments of these scans. The pair {rotation, translation} is used to calculate the robot's pose in the Global Coordinate System. An algorithm to merge the line segments in the global map is also proposed. For loop closure, for each local scan the corners and isolated end points of the new walls are detected. The coordinates (in the Global Coordinate System) of these new landmarks are stored. When these landmarks are detected again in the next scans, the distance and angle measurements from the robot to these landmarks are then used to recalculate the robot's pose. For implementation and testing of the proposed solution, a mobile robot equipped with a 2D laser scanner and a Raspberry Pi 3B plus onboard computer was used. Firstly, the robot is simulated in a 44 m² environment with 9 and 10 walls using the software Gazebo. An experiment using the real robot in an environment with 10 walls of 41.44 m² is also presented. In these experiments results are presented without and with loop closure. These results demonstrate the good performance of the proposed algorithm in these environments.

Listas de Figuras

FIGURA 2.1 – Tarefas básicas na robótica móvel (STACHNISS, 2006).	26
FIGURA 2.2 – Grafo do problema da tarefa de localização na robótica móvel (THRUN <i>et al.</i> , 2005).	27
FIGURA 2.3 – Grafo do problema da tarefa de mapeamento na robótica móvel (THRUN <i>et al.</i> , 2005).	28
FIGURA 2.4 – Visualização das incertezas ao longo de mapeamento. (a) Exemplo de mapa formado com incertezas e (b) exemplo de mapa de grade de ocupação com menos incertezas (THRUN <i>et al.</i> , 2005).	29
FIGURA 2.5 – (a) Grafo referente ao movimento do robô por um ambiente desconhecido que gera incertezas em suas posturas (elipses cinzas) e no mapa (elipses vermelhas), acumulando erros. Em (b), o robô reconhece uma região que já havia visitado através de um ponto de referência com incerteza pequena e todas as posturas do robô e o mapa são atualizados, diminuindo suas incertezas (MONTEMERLO, 2003).	30
FIGURA 2.6 – Grafo representativo da técnica de <i>online</i> SLAM. Os nós brancos e sombreados representam o objetivo atual do processo, os nós apenas brancos são os objetivos já adquiridos em instantes anteriores e os nós cinzas são as variáveis auxiliares no alcance do objetivo atual da técnica (THRUN <i>et al.</i> , 2005).	30
FIGURA 2.7 – Grafo representativo da técnica de <i>full</i> SLAM. Os nós brancos e sombreados representam o objetivo do processo que precisam ser atualizados e os nós cinzas são as variáveis auxiliares no alcance do objetivo da técnica (THRUN <i>et al.</i> , 2005).	31
FIGURA 2.8 – Grafo formado por nós de posturas estimadas (MATHWORKS, 2022a).	32
FIGURA 2.9 – (a) Grafo de posturas antes da otimização e (b) Grafo de posturas após a otimização (MATHWORKS, 2022a).	33

FIGURA 2.10 –(a) Mapa construído por meio do algoritmo de SLAM antes do fechamento de laço, (b) Região no mapa onde há o ponto de fechamento do laço e (c) Mapa completo da região analisada após o fechamento de laço (NEWMAN; HO, 2005).	35
FIGURA 2.11 –Representação da distância entre um ponto e um segmento de reta no plano cartesiano.	36
FIGURA 2.12 –Visualização da reta ajustada pelo método de mínimos quadrados (PAETH, 1995).	39
FIGURA 2.13 –Extração de segmentos de reta delimitados por pontos terminais direcionais, em que as paredes do ambiente são visualizadas como linhas brancas no mapa, as linhas extraídas são representadas por linhas vermelhas e os pontos terminais por quadrados verdes (GAO <i>et al.</i> , 2018).	40
FIGURA 2.14 –Diferença entre as técnicas de correspondência de varredura (a) por meio da detecção de pontos característicos e de correspondência de características (b) a partir da detecção de linhas como característica (AGHAMOHAMMADI <i>et al.</i> , 2007).	41
FIGURA 2.15 –Duas posturas relacionadas ao <i>frame</i> global de referência (NIETO <i>et al.</i> , 2007).	43
FIGURA 2.16 –Exemplo de alinhamento entre <i>scans</i> , em que (a) mostra dois <i>scans</i> em diferentes posturas antes do alinhamento e (b) evidencia os <i>scans</i> alinhados (MATHWORKS, 2022a).	43
FIGURA 2.17 –Mapa de características, no qual a estrela vermelha representa a característica do <i>endpoint</i> direcional, a estrela azul denota o <i>endpoint</i> instável, a linha vermelha evidencia o segmento de reta extraído e a linha azul é a fronteira ou obstáculo no mapa e a postura do robô é indicada pelo círculo verde e o caminho do robô é representado pela linha preta (GAO <i>et al.</i> , 2018).	44
FIGURA 2.18 –Modelo de comunicação da rede ROS (YAMASHINA <i>et al.</i> , 2016).	47
FIGURA 3.1 – Leitura do sensor <i>scanner a laser</i> em uma postura do robô no Sistema de Coordenadas Global.	50
FIGURA 3.2 – Segmentos de reta estimados pelo algoritmo RANSAC a partir dos dados obtidos pelo <i>scanner a laser</i>	50
FIGURA 3.3 – Ilustração sobre o processo do algoritmo de detecção de linhas para corrigir as linhas ajustadas e obtidas pelo algoritmo RANSAC.	51

FIGURA 3.4 – Segmentos de reta adaptados do algoritmo de RANSAC com seus cantos destacados.	52
FIGURA 3.5 – Fluxograma do algoritmo RANSAC adaptado para detecção de segmentos de reta e cantos.	52
FIGURA 3.6 – Distâncias entre pontos terminais de dois segmentos para se obter a medida de correspondência dos segmentos de reta.	54
FIGURA 3.7 – Representação dos marcos com seus ângulos e distâncias em relação à uma postura real no plano cartesiano.	55
FIGURA 3.8 – Representação dos marcos pelos seus índices e suas posições no plano cartesiano.	57
FIGURA 3.9 – Representação da junção de segmentos com os pontos terminais identificados por seus índices no plano cartesiano.	59
FIGURA 3.10 – Robô Ísis desenvolvido.	61
FIGURA 3.11 – Sensor <i>Scanner a Laser</i> RPLiDAR modelo A2M8 presente no robô (SHANGHAI SLAMTEC CO., LTD, 2017).	62
FIGURA 3.12 – Sensor IMU modelo BWT901CL presente no robô (WITMOTION SHENZHEN CO., LTD, 2020a).	63
FIGURA 3.13 – Sensor <i>encoder</i> integrado ao motor CC localizado nas rodas do robô móvel (https://www.baudaelectronica.com.br/motor-dc-6v-com-encoder-100-rpm.html).	64
FIGURA 3.14 – Diagrama da comunicação entre <i>hardwares</i> que são os retângulos azuis e <i>softwares</i> que são os verdes (FILHO, 2021).	64
FIGURA 3.15 – Ilustração de um diagrama de nós cuja mensagem é enviada e recebida entre os nós através de um único tópico (MATHWORKS, 2022b). .	65
FIGURA 3.16 – Arquitetura de <i>hardware</i> dos dispositivos utilizados (PINTO, 2020). .	66
FIGURA 3.17 – Configuração da rede ROS desenvolvida.	67
FIGURA 3.18 – Diagrama original de nós e tópicos utilizados pelo robô móvel na rede ROS (PINTO, 2020).	68
FIGURA 3.19 – Modelo do robô desenvolvido no Gazebo.	68
FIGURA 3.20 – Modelo de um ambiente configurado no Gazebo, em que (a) possui nove paredes e (b) apresenta dez paredes.	69
FIGURA 3.21 – Ambiente construído no corredor do LMI.	70

FIGURA 4.1 – Sistema de coordenadas no <i>software</i> Gazebo com vista de cima para baixo do ambiente, em que (a) é o ambiente de nove paredes e (b) é o ambiente de dez paredes.	72
FIGURA 4.2 – Exemplo de segmentos adquiridos com o algoritmo RANSAC adaptado extraídos de um <i>scan</i> de pontos numa determinada postura.	74
FIGURA 4.3 – <i>Scans</i> de pontos gerados nas Posturas 1 a 6 do primeiro experimento.	75
FIGURA 4.4 – <i>Scans</i> de pontos gerados nas Posturas 7 a 12 do primeiro experimento.	76
FIGURA 4.5 – <i>Scans</i> de pontos gerados nas Posturas 13 a 18 do primeiro experimento.	77
FIGURA 4.6 – <i>Scans</i> de pontos gerados nas Posturas 19 a 24 do primeiro experimento.	78
FIGURA 4.7 – <i>Scans</i> de pontos gerados nas Posturas 25 a 30 do primeiro experimento.	79
FIGURA 4.8 – <i>Scans</i> de pontos gerados nas Posturas 31 a 36 do primeiro experimento.	80
FIGURA 4.9 – <i>Scans</i> de pontos gerados nas Posturas 37 a 42 do primeiro experimento.	81
FIGURA 4.10 – <i>Scans</i> de pontos gerados nas Posturas 43 a 45 do primeiro experimento.	82
FIGURA 4.11 – <i>Line matching</i> entre <i>scans</i> em posturas consecutivas de 1 a 7 do primeiro experimento.	85
FIGURA 4.12 – <i>Line matching</i> entre <i>scans</i> em posturas consecutivas de 7 a 13 do primeiro experimento.	86
FIGURA 4.13 – <i>Line matching</i> entre <i>scans</i> em posturas consecutivas de 13 a 19 do primeiro experimento.	87
FIGURA 4.14 – <i>Line matching</i> entre <i>scans</i> em posturas consecutivas de 19 a 25 do primeiro experimento.	88
FIGURA 4.15 – <i>Line matching</i> entre <i>scans</i> em posturas consecutivas de 25 a 31 do primeiro experimento.	89
FIGURA 4.16 – <i>Line matching</i> entre <i>scans</i> em posturas consecutivas de 31 a 37 do primeiro experimento.	90
FIGURA 4.17 – <i>Line matching</i> entre <i>scans</i> em posturas consecutivas de 37 a 43 do primeiro experimento.	91

FIGURA 4.18 – <i>Line matching</i> entre <i>scans</i> em posturas consecutivas de 43 a 45 do primeiro experimento.	92
FIGURA 4.19 –Comparação da posição X presente nas posturas reais e estimadas do primeiro experimento.	92
FIGURA 4.20 –Comparação da posição Y presente nas posturas reais e estimadas do primeiro experimento.	93
FIGURA 4.21 –Comparação da orientação presente nas posturas reais e estimadas do primeiro experimento.	93
FIGURA 4.22 –Comparação da orientação presente nas posturas reais e estimadas do primeiro experimento.	95
FIGURA 4.23 –Posturas estimadas do robô no mapa global do ambiente no sistema de coordenadas da postura inicial do primeiro experimento sem fechamento de laço e com a junção de segmentos.	95
FIGURA 4.24 –Comparação entre o mapa simulado do ambiente de nove paredes e o mapa estimado.	96
FIGURA 4.25 –Comparação da posição x em metros presente nas posturas reais e estimadas do segundo experimento sem fechamento de laço.	99
FIGURA 4.26 –Comparação da posição y em metros presente nas posturas reais e estimadas do segundo experimento sem fechamento de laço.	99
FIGURA 4.27 –Comparação da orientação em graus presente nas posturas reais e estimadas do segundo experimento sem fechamento de laço.	100
FIGURA 4.28 –Comparação do erro presente nas posturas reais e estimadas do segundo experimento sem fechamento de laço.	100
FIGURA 4.29 –Posturas estimadas do robô no mapa global do ambiente no sistema de coordenadas da postura inicial do segundo experimento sem fechamento de laço e com a junção de segmentos.	101
FIGURA 4.30 – <i>Line matchings</i> sem fechamento do laço entre <i>scans</i> em posturas consecutivas que foram inadequadas no segundo experimento.	102
FIGURA 4.31 –Pontos terminais associados aos marcos registrados no sistema de coordenadas local do <i>scan</i> nas posturas 21 e 27.	103
FIGURA 4.32 – <i>Scan</i> nos sistemas de coordenadas local e global da postura 21.	104
FIGURA 4.33 – <i>Scan</i> nos sistemas de coordenadas local e global da postura 23.	104
FIGURA 4.34 – <i>Scan</i> nos sistemas de coordenadas local e global da postura 28.	104

FIGURA 4.35 – <i>Scan</i> nos sistemas de coordenadas local e global da postura 29.	105
FIGURA 4.36 –Comparação da posição x em metros presente nas posturas reais e estimadas do segundo experimento com fechamento de laço.	105
FIGURA 4.37 –Comparação da posição y em metros presente nas posturas reais e estimadas do segundo experimento com fechamento de laço.	107
FIGURA 4.38 –Comparação da orientação em graus presente nas posturas reais e estimadas do segundo experimento com fechamento de laço.	107
FIGURA 4.39 –Comparação do erro presente nas posturas reais e estimadas do segundo experimento com fechamento de laço.	108
FIGURA 4.40 –Posturas estimadas do robô no mapa global do ambiente no sistema de coordenadas da postura inicial do segundo experimento com fechamento de laço e com a junção de segmentos.	108
FIGURA 4.41 –Comparação entre o mapa simulado do ambiente de dez paredes e o mapa estimado.	109
FIGURA 4.42 –Vista de satélite dos prédios e da biblioteca do ITA. O retângulo amarelo evidencia o corredor em frente ao LMI onde foi realizado o experimento com o robô real. Adaptado de Google (2022).	110
FIGURA 4.43 –Ambiente construído no corredor do LMI por paredes de madeira. .	110
FIGURA 4.44 –Gráfico da distância real em relação à distância medida pelo sensor em metros.	112
FIGURA 4.45 –Gráfico da distância medida pelo sensor em relação à distância corrigida em metros.	112
FIGURA 4.46 –Diagrama de blocos que exemplifica o método de correção de dados para calibração do sensor <i>scanner a laser</i>	113
FIGURA 4.47 –Curva calibrada do sensor de distância por meio da interpolação linear local.	114
FIGURA 4.48 –Amostras de <i>Line matchings</i> sem fechamento do laço entre <i>scans</i> em posturas consecutivas no experimento real.	118
FIGURA 4.49 –Trajetória das posturas estimadas do robô no mapa global do ambiente no sistema de coordenadas da postura inicial do experimento real sem fechamento de laço e com a junção de segmentos.	119

FIGURA 4.50 –Correspondência entre os marcos selecionados e os cantos identificados nas posturas 3, 16, 33, 48, 63 e 78 no sistema de coordenadas local de cada postura durante o fechamento de laço no experimento real.	120
FIGURA 4.51 –Quantidade de marcos detectados em cada <i>scan</i> das posturas para fechamento do laço com junção de segmentos.	121
FIGURA 4.52 –Comparação da orientação em graus presente nas posturas reais e estimadas do experimento real com fechamento de laço.	121
FIGURA 4.53 –Trajetória de posturas estimadas do robô no mapa global do ambiente no sistema de coordenadas da postura inicial do experimento real com fechamento de laço e com a junção de segmentos.	122
FIGURA 4.54 –Comparação entre o mapa do ambiente real e o mapa estimado do experimento real com fechamento do laço.	123

Lista de Tabelas

TABELA 3.1 – Parâmetros do Modelo Físico do Robô.	60
TABELA 3.2 – Especificações do sensor LiDAR modelo A2M8.	61
TABELA 3.3 – Especificações do sensor IMU modelo BWT901CL.	62
TABELA 3.4 – Especificações do <i>encoder</i> integrado ao motor CC.	63
TABELA 4.1 – Postura inicial do robô no sistema de coordenadas do Gazebo. . . .	71
TABELA 4.2 – Comprimento e coordenada do ponto médio das paredes do ambiente simulado.	72
TABELA 4.3 – Coordenadas (x,y) em metros dos pontos terminais dos segmentos da Figura 4.2.	74
TABELA 4.4 – Rotação em graus e translação em metros obtidas entre <i>scans</i> em posturas consecutivas do primeiro experimento.	83
TABELA 4.5 – Erro entre as posturas real e estimada no sistema de coordenadas da postura inicial do primeiro experimento.	94
TABELA 4.6 – Rotação em graus e translação em metros obtidas entre <i>scans</i> em posturas consecutivas do segundo experimento.	97
TABELA 4.7 – Erro entre as posturas real e estimada no sistema de coordenadas da postura inicial do segundo experimento sem fechamento do laço.	98
TABELA 4.8 – Erro entre as posturas real e estimada no sistema de coordenadas da postura inicial do segundo experimento com fechamento do laço.	106
TABELA 4.9 – Dados coletados da distância real obtida pela trena e da distância medida pelo sensor <i>scanner a laser</i>	111
TABELA 4.10 – Rotação (R) em graus e translação (T) em metros obtidas entre <i>scans</i> em posturas consecutivas do experimento real.	116

Lista de Abreviaturas e Siglas

CC	Corrente Contínua
EKF	<i>Extended Kalman Filter</i>
GPS	<i>Global Positioning System</i>
IDE	<i>Integrated Development Environment</i>
IMU	<i>Inertial Measurement Unit</i>
IP	<i>Internet Protocol</i>
LC	<i>Loop Closure</i>
LiDAR	<i>Light Detection and Ranging</i>
LMI	Laboratório de Máquinas Inteligentes
MC	Medida de Correspondência
RANSAC	<i>Random Sample And Consensus</i>
ROS	<i>Robot Operating System</i>
RPM	Rotações por Minuto
SC	Sistema de Coordenadas
SLAM	<i>Simultaneous Localization And Mapping</i>
TCP	<i>Transmission Control Protocol</i>
UART	<i>Universal Asynchronous Receiver/Transmitter</i>
Wi-Fi	<i>Wireless Fidelity</i>

Sumário

1	INTRODUÇÃO	21
1.1	Objetivos	21
1.2	Relevância e Motivação	22
1.3	Trabalhos Relacionados	22
1.4	Organização do trabalho	23
2	FUNDAMENTAÇÃO TEÓRICA	25
2.1	Robótica Móvel	25
2.2	Localização	26
2.3	Mapeamento	27
2.4	SLAM	28
2.4.1	<i>Online SLAM</i>	30
2.4.2	<i>Full SLAM</i>	31
2.4.3	<i>Pose Graph SLAM</i> e <i>Feature-based SLAM</i>	32
2.4.4	Fechamento de Laço	33
2.5	RANSAC	34
2.5.1	Distância entre ponto e linha	36
2.5.2	Ajuste de linha por mínimos quadrados	37
2.5.3	Extração de segmentos de reta	39
2.6	<i>Scan Matching</i> e <i>Feature Matching</i>	40
2.6.1	Transformação de Coordenadas para Alinhamento de Scans	41
2.6.2	<i>Feature Map</i>	43
2.6.3	Correspondência de Segmentos de Reta	44

2.7 Robot Operating System	45
3 PROPOSTA DE SOLUÇÃO	48
3.1 Estrutura do SLAM com correspondência de características	48
3.1.1 Objetivo da tarefa	48
3.1.2 Premissas do problema	49
3.1.3 Algoritmo de RANSAC modificado	49
3.1.4 Algoritmo de Correspondência de Características	53
3.1.5 Proposta para o Fechamento do Laço	55
3.1.6 Proposta para a Junção de Segmentos	58
3.2 Descrição do Robô Móvel	59
3.2.1 Sensores do Robô	60
3.3 Configuração Geral para Experimentos e Ambiente	63
3.3.1 Configuração da rede ROS	64
3.3.2 Arquitetura da rede de comunicação do robô	65
3.3.3 Simulação do robô móvel e do ambiente	68
3.3.4 Ambiente do Experimento Real	69
4 EXPERIMENTOS E RESULTADOS OBTIDOS	71
4.1 Estrutura da Simulação no Gazebo	71
4.2 Resultado dos Experimentos Simulados	73
4.2.1 Primeiro Experimento	73
4.2.2 Segundo Experimento	96
4.3 Resultado do Experimento Real	109
4.3.1 Caso 1: sem Fechamento de Laço	115
4.3.2 Caso 2: com Fechamento de Laço	117
5 CONCLUSÃO	124
REFERÊNCIAS	126
APÊNDICE A – PSEUDOCÓDIGO DO ALGORITMO DE SLAM	129

A.1 Passos principais do Algoritmo de SLAM 2D com <i>Line Matching</i> e <i>Loop Closure</i>	129
A.1.1 Algoritmo de SLAM 2D Geral	129
A.1.2 Rotina de <i>Loop Closure</i> (LC)	130
A.1.3 Rotina de Registro de Marcos Globais para LC	130

1 Introdução

Ao longo das últimas décadas, verificou-se a necessidade de mapear e explorar ambientes desconhecidos, sendo eles internos, subterrâneos, radioativos e hostis para o ser humano, além da falta de sinal de GPS em diversos lugares fechados dificultar a localização e mapeamento interno. Com isso, houve o surgimento de inúmeros projetos para solucionarem este problema. Os projetos mais frequentes tratados nessa linha foram os que envolveram a técnica de localização e mapeamento simultâneos (SLAM) que é muito empregada na área da robótica (LI *et al.*, 2016).

Existem várias formas de se resolver o problema com SLAM, como a utilização de métodos baseados em recursos visuais, na correspondência de pontos pela varredura a *laser* (*scan matching*) e na correspondência de características dos ambientes (*feature matching*), por exemplo. Com isso, o SLAM por combinação de *scans* é um método acurado que precisa de um elevado custo computacional a partir do armazenamento e uso de vários dados, porém o SLAM por combinação de características é uma técnica mais simples que apresenta um gasto computacional reduzido, já que utiliza menos dados como linhas e pontos extraídos do ambiente (RAYNER, 2022).

Diante disso, no processo de SLAM, o robô é capaz de obter, em cada postura, sua postura estimada ao mesmo tempo que constrói o mapa do ambiente a medida que o veículo trafega de forma autônoma ou por meio de um agente externo. A partir disso, o robô pode circular pelo ambiente de forma *online* ou *offline*, escaneando nuvens de pontos através do seu sensor para cada postura detectada ao longo da trajetória feita pelo robô, sendo considerado um tipo de *online* SLAM. Tais dados são extraídos e tratados para a estimativa das posturas e para a geração do mapa referente ao ambiente interno analisado.

1.1 Objetivos

O objetivo deste trabalho de dissertação é elaborar uma solução para o problema de SLAM 2D baseado em grafo de posturas por meio da técnica de correspondência de características em ambiente interno a partir de um robô móvel que apresenta *scanner a laser* (LiDAR), um *kinect*, *encoders* nos motores CC das rodas e uma IMU como sensores

e atuadores embarcados, porém apenas o sensor *scanner a laser* foi utilizado na solução do problema. Mais especificamente, o *feature matching* abordado apresenta o intuito de utilizar a correspondência de linhas e pontos terminais (*endpoints*) como características essenciais para a solução da técnica de SLAM.

1.2 Relevância e Motivação

Dante dos constantes avanços da robótica por distintos segmentos, verificou-se que os robôs móveis não estão mais presentes somente em pesquisas, indústrias e expedições espaciais e militares. A cada instante, há o surgimento de novas tecnologias e a inserção delas no cotidiano das pessoas está cada vez maior. Não somente os robôs móveis, mas também os sistemas que utilizam as técnicas de SLAM estão adentrando em tarefas do dia-a-dia que normalmente eram feitas apenas pelo ser humano.

Dessa forma, existem diversas áreas que aplicam robótica móvel com sistemas de SLAM. Em âmbitos mais recorrentes estão as áreas espacial e militar, em ambientes de chão de fábrica para controle de processos, no reconhecimento de ambientes impróprios para o ser humano, no âmbito hospitalar para auxiliar em processos que apresentam escassez de profissionais em regiões insalubres, por exemplo. Paralelo a isso, a robótica também está inserida no cotidiano das pessoas, como na limpeza de residências por meio de robôs do tipo aspirador de pó, além do transporte de pessoas, encomendas e refeições por veículos autônomos, entre outros.

1.3 Trabalhos Relacionados

O problema de localização e mapeamento simultâneos (SLAM), atualmente, é muito abordado na área da robótica móvel. Muitos dos desafios relacionados a essa técnica permitem a aplicabilidade em ambientes internos (*indoors*) ou externos (*outdoors*). Em Buonocore (2013), utilizou-se filtro de partículas do tipo FastSLAM para trafegar um robô móvel de baixo custo em um ambiente interno. Além disso, existem diversos outros tipos de soluções para o SLAM 2D, como em Nieto *et al.* (2007), que apresenta o *Scan SLAM* baseado no EKF-SLAM para realizar correspondência de varredura em ambientes externos, permitindo a estimativa de localização estocástica para pontos de referência como forma arbitrária dentro da estrutura do filtro de Kalman, o que torna a associação de dados mais confiável com o fornecimento de uma métrica para reduzir a ambiguidade entre pontos de referência.

Segundo Grisetti *et al.* (2010), uma maneira intuitiva para formular SLAM é com a utilização de um grafo de posturas (*Graph-Based SLAM*) para aplicações em ambientes

desconhecidos com a ausência de sistemas de referência externos (GPS), como um carro autônomo instrumentado e desenvolvido em Stanford, e também um robô guia em museu desenvolvido pela Toyota, dentre outras. Além disso, Gao *et al.* (2018) sugere utilizar *Graph-Based SLAM* por meio de recursos de pontos terminais direcionais para a construção de mapas compostos por polígonos e a exploração autônoma a fim de ser aplicado em um ambiente de escritório e corredor.

Para Aghamohammadi *et al.* (2007), é realizado um método de rastreamento de postura e de alta velocidade para robôs móveis baseado na correspondência de características (*Feature-Based SLAM*) extraídos de *scans* consecutivos. Já Li *et al.* (2016) propôs a utilização de *Feature-Based SLAM* 2D com *scan matching* por meio de um sensor *scanner a laser*, combinando recursos de pontos e linhas para mapeamento e modelagem de ambientes internos para fins de emergências, evacuação e localização.

Grande parte dos trabalhos relacionados visam aumentar a eficiência computacional a fim de realizar o mapeamento e a localização das posturas estimadas, reduzindo o erro existente, além da associação e aquisição de dados, não linearização de modelos, fechamento do laço e identificação de características como pontos de referência. Dessa forma, alguns trabalhos podem ser associados como Pinto (2020), Filho (2021), Newman e Ho (2005), Stachniss (2006), dentre outros.

1.4 Organização do trabalho

O Capítulo 1 apresenta a introdução desse trabalho de pesquisa com a demonstração dos problemas a serem solucionados para alcançar os objetivos evidenciados. A relevância e a motivação do projeto são mostradas e incentivam a execução deste trabalho.

O Capítulo 2 se refere à fundamentação teórica e apresenta definições de conceitos necessários para o desenvolvimento da pesquisa. Dessa forma, são destacados alguns conceitos de localização, mapeamento, SLAM, RANSAC, fechamento de laço, *scan matching*, *feature matching* e uma breve abordagem sobre o sistema ROS.

O Capítulo 3 aborda detalhes sobre a solução proposta, como a descrição do robô móvel utilizado, as características e restrições dos algoritmos, a configuração utilizada para os experimentos e para o ambiente, como também informações a respeito da simulação do robô e do ambiente real empregados.

O Capítulo 4 mostra os resultados de experimentos simulados e real e são feitos argumentos para considerar o que foi gerado pelos resultados.

O capítulo 5 apresenta as conclusões e algumas sugestões para trabalhos futuros que visam o aprimoramento e extensão deste projeto de pesquisa.

Todos os *softwares* desenvolvidos neste trabalho podem ser acessados em: https://github.com/Intelligent-Machines-Lab/Line_Matching_SLAM.

2 Fundamentação Teórica

Neste capítulo, as principais tarefas da robótica móvel são destacadas, além da descrição sobre SLAM, RANSAC e a correspondência de características (*feature matching*) a fim de, em seguida, realizar a proposta de solução para o problema evidenciado.

2.1 Robótica Móvel

A robótica é uma ciência ampla capaz de analisar e explorar o mundo físico a partir de dispositivos mecânicos que são controlados por computador. Tal ciência possui diversas áreas de pesquisa em específico na área da robótica móvel que apresenta inúmeras aplicações que contribuem diretamente com a sociedade, sendo possível a utilização de sistemas inteligentes para a realização de tarefas humanas, como, por exemplo, sistemas móveis desenvolvidos para exploração planetária, carros autônomos que trafegam em rodovias com maior segurança, sistemas robóticos empregados em linha de montagem de fábricas e robôs que auxiliam profissionais da saúde durante cirurgias (THRUN *et al.*, 2005).

Existem três tarefas básicas na robótica móvel, como o mapeamento, localização e o planejamento de trajetória, porém nenhuma delas pode ser resolvida de forma independente, já que para um robô saber como é o ambiente em que está presente, é necessário saber quais as suas posições no ambiente por meio de um conjunto de observações. Também, é complexo estimar a posição atual do sistema robótico sem saber como é o mapa e quais são os dados sobre as posturas do veículo a fim de planejar a trajetória do robô para alcançar um determinado objetivo (STACHNISS, 2006).

Com isso, a Figura 2.1 destaca um diagrama com as tarefas da robótica móvel que são o mapeamento, a localização e o planejamento de trajetória, além da fusão dessas tarefas. A combinação entre a localização e o mapeamento é conhecida como um problema de SLAM, a intersecção entre o mapeamento e o planejamento de caminho é um caso de exploração, já a junção da localização com o planejamento de trajetória demonstra um problema de localização ativa e, finalmente, a intersecção de todas as tarefas básicas é conhecida como um problema de abordagem integrada também chamado de SLAM ativo.

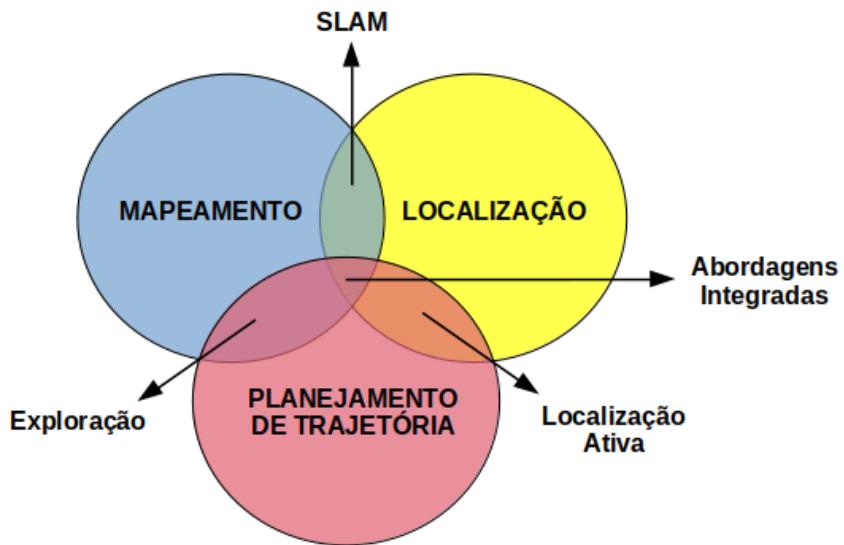


FIGURA 2.1 – Tarefas básicas na robótica móvel (STACHNISS, 2006).

2.2 Localização

A localização é uma tarefa da robótica móvel que consiste em estimar a postura (posição e orientação) de um robô a partir do mapa do ambiente, baseado nos movimentos do veículo. Esta tarefa é essencial para as demais, pois quase todas necessitam da informação da localização do veículo e de obstáculos utilizados em experimentos. Essa tarefa se difere entre o rastreamento da postura, em que a postura inicial do robô é conhecida, e a localização global, em que a postura inicial do robô não é conhecida *a priori* (STACHNISS, 2006).

A Figura 2.2 evidencia o grafo referente ao problema de localização presente na robótica móvel. Tal figura mostra as medidas do ambiente Z_t obtidas por sensores embarcados no robô, os controles u_t e o mapa m , sendo representados por nós cinzas já conhecidos. Com isso, a localização possui o intuito de estimar a postura do robô que é representada por X_t , evidenciada pelos nós brancos.

Segundo Thrun *et al.* (2005), essa tarefa transforma as coordenadas de uma postura que está em um determinado sistema de coordenadas para outro sistema relacionado a outra postura. Além disso, os mapas obtidos são definidos no sistema de coordenadas global independentemente da postura do robô a fim de realizar correspondência entre o sistema de coordenadas do mapa e o sistema de coordenadas local do robô. Com isso, o robô consegue localizar paredes e objetos a partir da transformação de coordenadas, estando no seu próprio sistema de coordenadas em relação ao mapa do ambiente.

No entanto, o maior problema da localização consiste na não obtenção da postura real do robô, já que grande parte dos robôs não possuem um sensor ideal que meça a postura

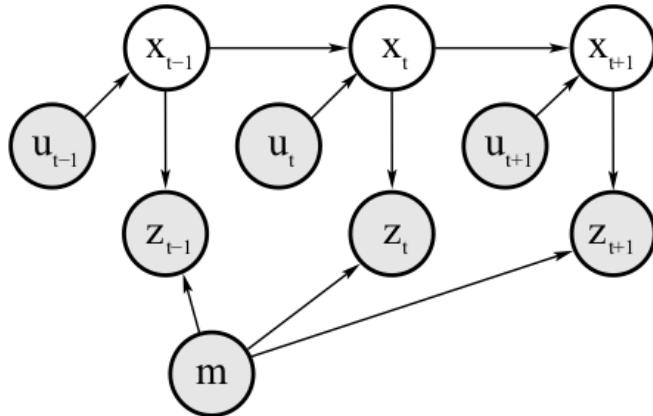


FIGURA 2.2 – Grafo do problema da tarefa de localização na robótica móvel (THRUN *et al.*, 2005).

sem que haja ruído, com isso, há incertezas na medição pelo uso de odometria. Assim, a postura deve ser estimada a partir das medidas dos sensores. Essas medidas são integradas ao longo do tempo para se calcular a postura estimada, pois existem regiões no ambiente muito parecidas que podem ser confundidas se fosse realizada apenas uma única medição da postura. Contudo, a estimativa da postura forma um erro que é acumulativo com a estimativa das posturas seguintes, pois há incertezas ao longo do caminho do veículo no ambiente, evidenciando mais um problema da tarefa de localização que difere a postura estimada da postura real do sistema robótico (THRUN *et al.*, 2005).

2.3 Mapeamento

A tarefa de mapeamento trata de desenvolver ou atualizar o mapa do ambiente baseada nas posturas adquiridas ao longo do processo de localização do robô dentro do local analisado. Além disso, os sensores presentes no robô escaneiam as paredes e objetos presentes no ambiente e geram dados no sistema de coordenadas da postura em que o robô móvel se encontra. Por isso, os dados do mapa são tratados com a tarefa de localização e geram uma representação aproximada do ambiente real em questão. Em suma, o mapeamento depende da representação do ambiente e da interpretação dos dados obtidos pelos sensores do veículo.

O grafo observado na Figura 2.3 representa a tarefa de mapeamento, em que as medidas do ambiente Z_t geradas por sensores embarcados no robô, os controles dos movimentos do robô u_t e as suas posturas estimadas X_t são demonstradas pelos nós escuros, já o nó claro mostra o mapa m do ambiente que é a variável que necessita ser determinada.

De acordo com Stachniss (2006), as abordagens para o desenvolvimento de mapas podem ser passivas ou ativas, em que as passivas recebem os dados sobre o ambiente para a construção do mapa e as ativas planejam o movimento do robô a fim de conduzi-lo pelo

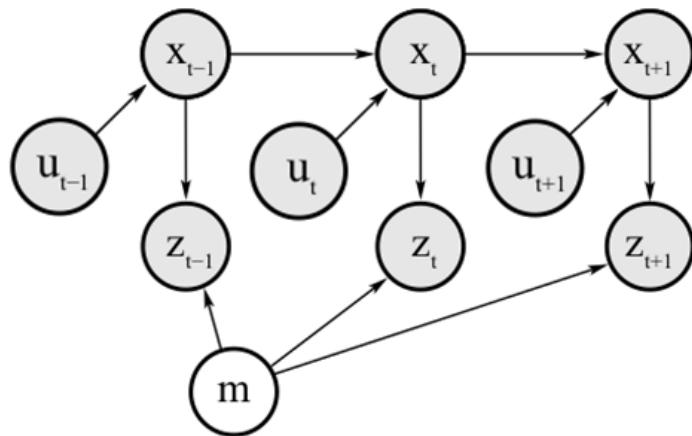


FIGURA 2.3 – Grafo do problema da tarefa de mapeamento na robótica móvel (THRUN *et al.*, 2005).

local analisado. Com isso, de acordo com a forma ativa, o robô pode ser considerado um veículo autônomo caso obtenha por conta própria um modelo de mapa acurado do ambiente.

2.4 SLAM

O problema de localização e mapeamento simultâneos (SLAM) é composto pela construção do mapa de um ambiente realizada por um robô e pelo cálculo da sua postura (posição e orientação) de forma paralela que representa a etapa de localização do robô no espaço analisado (BAILEY; DURRANT-WHYTE, 2006).

Segundo Li *et al.* (2016), essa técnica é muito empregada no mapeamento de ambientes internos, principalmente, com a falta de sinal de GPS (Sistema de Posicionamento Global) dentro de lugares fechados e subterrâneos que dificulta na modelagem e no mapeamento. Além disso, os sistemas com SLAM podem ser baseados em visão através de características visuais e em *laser*, em que o segundo é capaz de gerar mapas e modelos internos bem definidos, pois há a estimativa da posição relativa e da orientação entre dois *scans* de posturas medidos por um sensor *scanner a laser* através de uma transformação acurada que seja apta a alinhar o segundo *scan* ao primeiro, realizando, assim, a correspondência entre *scans* (*scan matching*).

No entanto, a estimativa das posturas é incerta e gera um erro que se torna acumulativo ao longo do movimento do robô pelo ambiente. A Figura 2.4 apresenta dois gráficos, nos quais o primeiro é um mapa repleto de incertezas (a), enquanto que o segundo (b) é um mapa de grade de ocupação com incerteza reduzida.

À medida que o robô se locomove pelo ambiente e coleta dados por meio de seus sensores integrados, ele salva tais informações que podem ser nuvens de pontos referentes

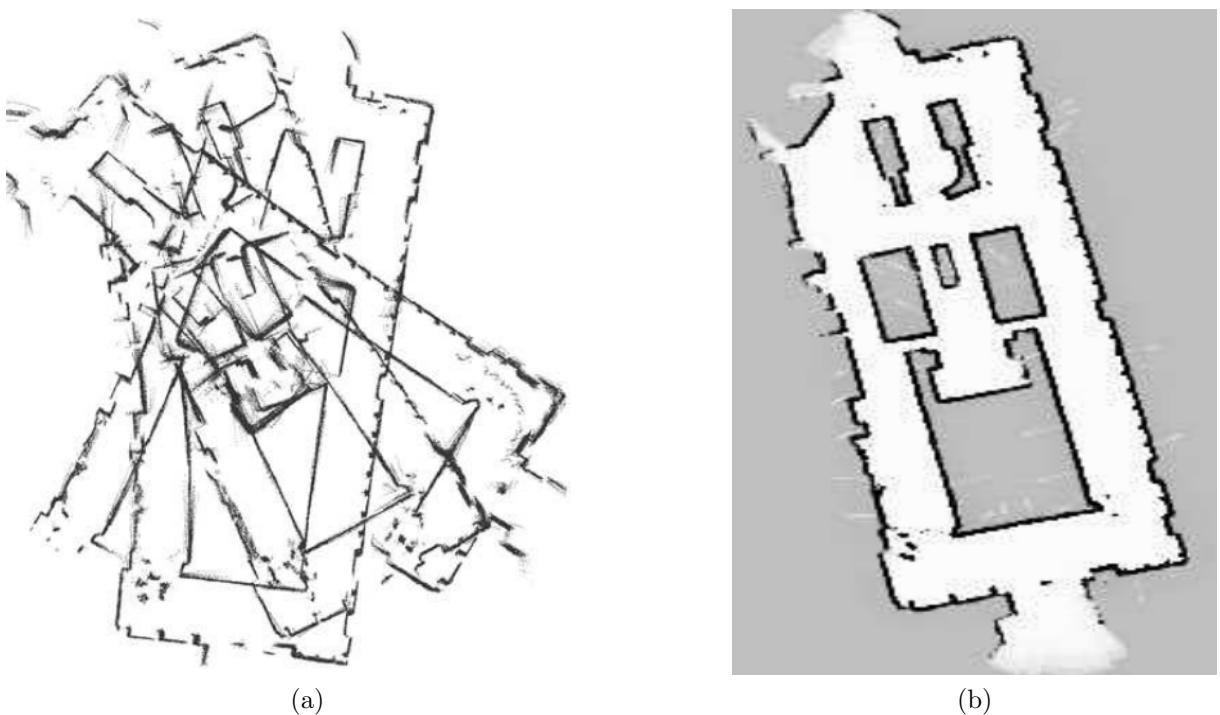


FIGURA 2.4 – Visualização das incertezas ao longo de mapeamento. (a) Exemplo de mapa formado com incertezas e (b) exemplo de mapa de grade de ocupação com menos incertezas (THRUN *et al.*, 2005).

às paredes e obstáculos presentes no ambiente, como também pontos de referência que podem ser essenciais quando o processo de *matching* desses *scans* for realizado. Essas informações mencionadas são exemplos de características do ambiente para o processo de SLAM.

A Figura 2.5 se refere a um exemplo de SLAM, no qual o robô se movimenta por um ambiente desconhecido e guarda informações a respeito do mesmo. Pode-se perceber que, ao longo da trajetória do robô em (a), as incertezas relacionadas à localização do veículo aumentam, assim como as incertezas na posição dos pontos de referência, pois ocorre o acúmulo de erros ao longo do tempo. No entanto, em (b), o robô se aproxima de uma região do ambiente que ele já havia circulado antes, verificando o ponto de referência registrado inicialmente que possuía uma incerteza pequena. Diante disso, o robô atualiza as demais estimativas das posturas, diminuindo as incertezas.

A técnica para se desenvolver SLAM não é única, existindo diversas formas de solucionar problemas com SLAM e inúmeros *hardwares* utilizados que podem simplificar o problema. Paralelo a isso, a partir de perspectivas probabilísticas, duas estruturas principais de solução para o problema de SLAM foram sugeridas por Thrun *et al.* (2005), que são: o *online* SLAM e o *full* SLAM.

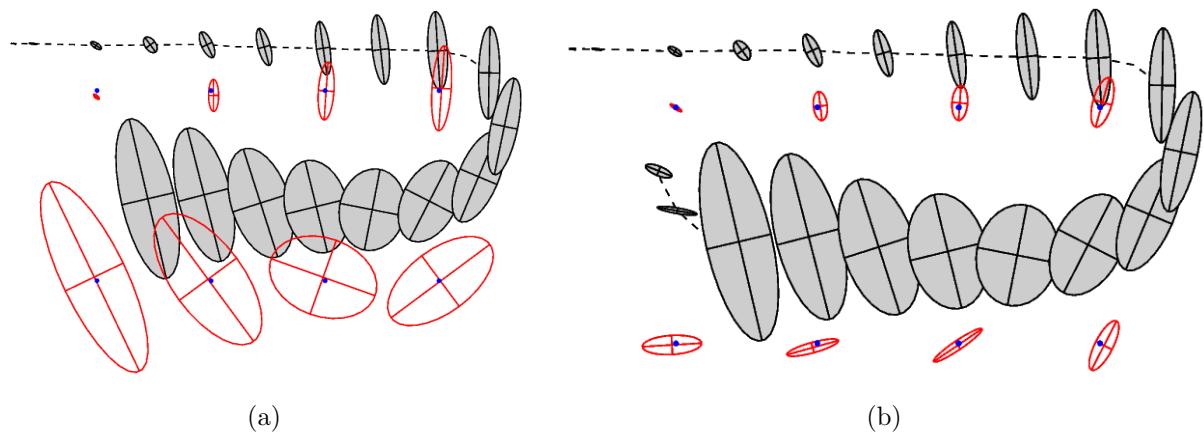


FIGURA 2.5 – (a) Grafo referente ao movimento do robô por um ambiente desconhecido que gera incertezas em suas posturas (elipses cinzas) e no mapa (elipses vermelhas), acumulando erros. Em (b), o robô reconhece uma região que já havia visitado através de um ponto de referência com incerteza pequena e todas as posturas do robô e o mapa são atualizados, diminuindo suas incertezas (MONTEMERLO, 2003).

2.4.1 *Online* SLAM

O *online* SLAM consiste em estimar a postura atual do robô e do mapa por meio de medições do sensor realizadas no instante atual em paralelo com o conhecimento do controle e é feito de forma recursiva, conforme pode ser verificado no grafo da Figura 2.6, em que a variável x_{t+1} representa a postura do robô atual que é atualizada juntamente com o mapa m , baseados na variável de controle (u_{t+1}) e na medição do sensor z_{t+1} no instante atual (THRUN *et al.*, 2005).

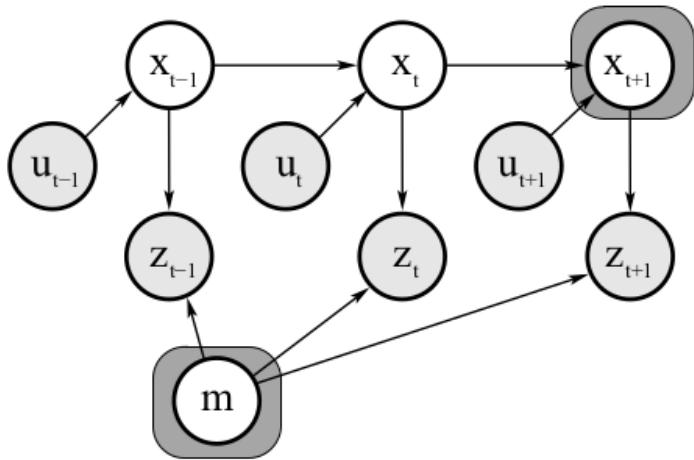


FIGURA 2.6 – Grafo representativo da técnica de *online* SLAM. Os nós brancos e sombreados representam o objetivo atual do processo, os nós apenas brancos são os objetivos já adquiridos em instantes anteriores e os nós cinzas são as variáveis auxiliares no alcance do objetivo atual da técnica (THRUN *et al.*, 2005).

Diversas soluções para o problema de *online* SLAM foram elaboradas, como o EKF-SLAM e o FastSLAM, por exemplo. O EKF-SLAM se baseia no algoritmo do Filtro de Kalman Estendido (FKE), o qual é limitado, pois precisa ser linearizado e apresenta

tempo de atualização quadrático. Tal algoritmo contém duas etapas principais que são a previsão e a atualização, e seu objetivo é estimar a postura posterior em relação à atual de forma conjunta com todos os pontos de referência que foram descobertos em posturas anteriores.

Paralelo a isso, o FastSLAM, diferentemente do EKF-SLAM, utiliza filtro de partículas para estimar a trajetória do robô. Dessa forma, para cada partícula existente, os erros individuais do mapa são condicionalmente independentes, dividindo o problema de mapeamento em problemas separados para cada característica do mapa. Em decorrência disso, o FastSLAM estima a localização das características do mapa por meio do filtro de Kalman estendido (EKF), porém com a aplicação de um algoritmo de EKF de baixa dimensão separado para cada característica individual do ambiente (THRUN *et al.*, 2005).

2.4.2 Full SLAM

O problema de *full SLAM* apresenta o propósito de calcular o conjunto completo de posturas em intervalos de tempo discretos juntamente com o mapa do ambiente, ou seja, é capaz de estimar a trajetória de posturas percorrida pelo robô e o mapa do ambiente à medida que as medidas dos sensores e os controles vão se atualizando para cada postura estimada (KUMAR, 2020).

Com isso, a depender do caminho feito pelo veículo, as posturas e o mapa são atualizados e suavizados com a redução do erro. A Figura 2.7 ilustra o *full SLAM*, em que todas as variáveis referentes às posturas ($x_{t-1:t+1}$) e ao mapa m necessitam ser atualizados, enquanto que as variáveis de controle (u) e as medidas dos sensores (z) auxiliam na atualização em cada instante das variáveis que devem ser adquiridas.

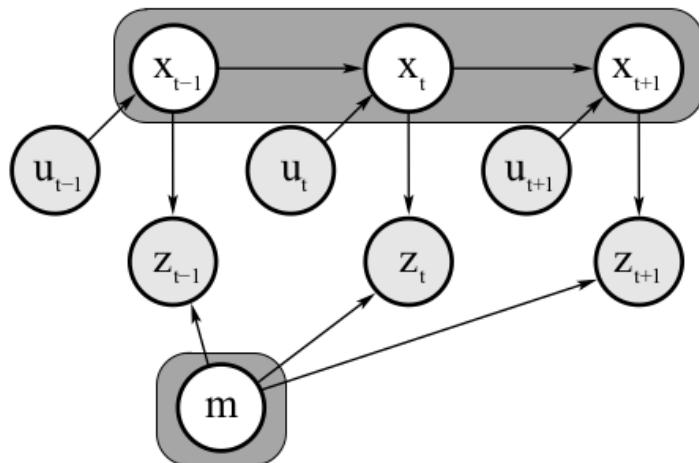


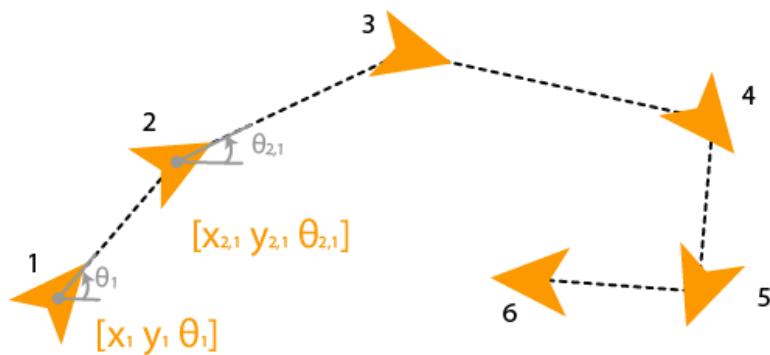
FIGURA 2.7 – Grafo representativo da técnica de *full SLAM*. Os nós brancos e sombreados representam o objetivo do processo que precisam ser atualizados e os nós cinzas são as variáveis auxiliares no alcance do objetivo da técnica (THRUN *et al.*, 2005).

Geralmente, o *full SLAM* minimiza o erro de mínimos quadrados em dados coletados pelas medições do sensor (GRISSETTI *et al.*, 2010). Um exemplo de problema com *full SLAM* é o *Graph SLAM* que é destaque em diversas aplicações com SLAM desenvolvidas nas últimas décadas.

2.4.3 Pose Graph SLAM e Feature-based SLAM

O problema de SLAM baseado em grafo de posturas foi proposto, inicialmente, por Lu e Milios (1997) e consiste em desenvolver um grafo cujos nós equivalem às posturas ou pontos de referência do robô móvel em diferentes regiões do ambiente no tempo e cujas arestas representam restrições entre as posturas. Tais restrições são adquiridas baseadas em medições de sensor e em observações do ambiente ou ações de mobilidade do robô durante sua circulação no ambiente.

Com isso, o mapa pode ser obtido com o desenvolvimento do grafo pela configuração espacial dos nós que seja mais adequada em relação às medidas das arestas (GRISSETTI *et al.*, 2010). A Figura 2.8 evidencia um grafo de posturas com a posição (x, y) e orientação (θ) definidas em algumas posturas, em que $[x_1 \ y_1 \ \theta_1]$ é a postura inicial com índice 1 e $[x_{2,1} \ y_{2,1} \ \theta_{2,1}]$ é a postura atual de índice 2 em relação à postura anterior de índice 1.



Estimativas de Nós das poses

FIGURA 2.8 – Grafo formado por nós de posturas estimadas (MATHWORKS, 2022a).

Além disso, as restrições presentes nas arestas que conectam as posturas podem ser contraditórias, pois as medidas dos sensores são afetadas por ruído. Assim que o grafo é desenvolvido, deve-se obter a configuração dos nós mais adequada, ou seja, mais consistente com as medidas, o que necessita minimizar o erro existente entre as posturas estimadas, corrigindo-as para que as posturas e o mapa sejam mais condizentes com o ambiente real analisado.

A Figura 2.9 representa um exemplo de SLAM por grafo de posturas a partir de dados coletados do *Intel Research Lab* e foram gerados por meio de informações obtidas da

odometria da roda e de um sensor de telêmetro a *laser* em um laboratório interno. A Figura 2.9 (a) mostra o conjunto de dados antes da otimização e a (b) evidencia os dados após a otimização por grafo de posturas. Os pontos azuis representam os nós e as linhas vermelhas são as arestas que conectam os nós (MATHWORKS, 2022a).

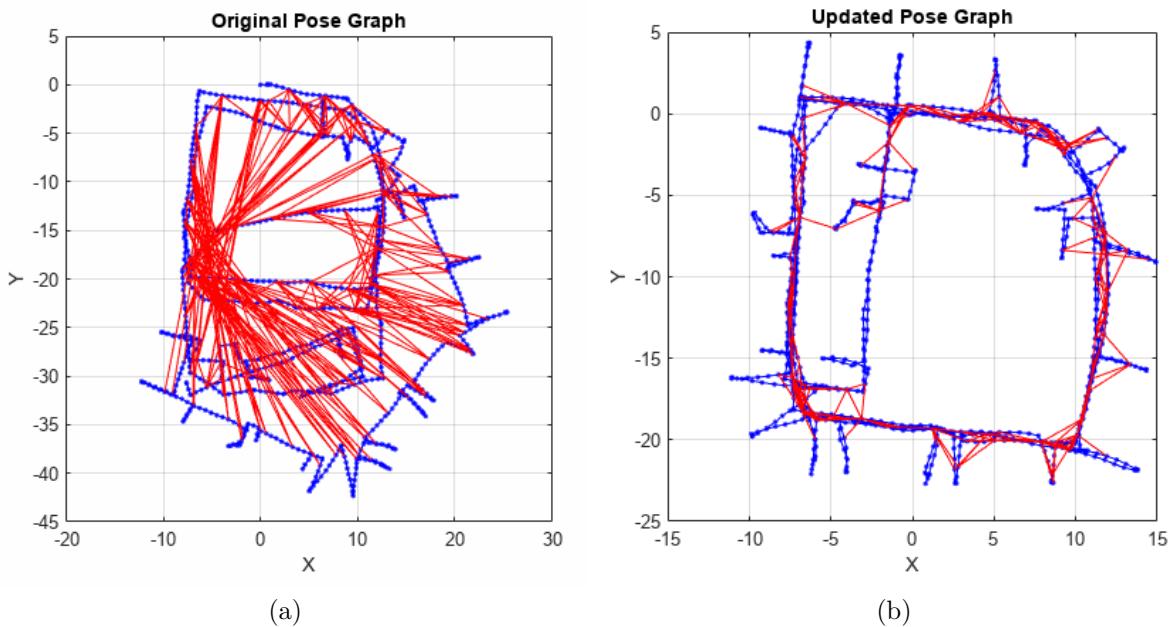


FIGURA 2.9 – (a) Grafo de posturas antes da otimização e (b) Grafo de posturas após a otimização (MATHWORKS, 2022a).

Enquanto isso, o SLAM baseado em características é muito popular para representar ambientes, pois as características extraídas por meio de sensores são utilizadas na estimação do mapa. A maioria das aplicações existentes desse tipo de método aborda características de baixo nível, como pontos, segmentos de linha, cantos e até círculos. No entanto, características de nível mais alto podem ser extraídas, como retângulos ou polígonos mais elaborados a fim de gerar um mapa mais estururado e detalhado do ambiente. Mediante isso, o *feature-based* SLAM pode apresentar a solução baseada no algoritmo de grafo de posturas, pois inclui as posturas do robô e as características observadas nos nós do grafo construído que retrata a trajetória do robô feita no ambiente (PUENTE; RODRIGUEZ-LOSADA, 2015). Mais detalhes a respeito desse método serão apresentados em seções posteriores.

2.4.4 Fechamento de Laço

O fechamento de laço (*loop closure*) é um recurso presente no problema de SLAM que determina se um robô móvel, após se locomover pelo ambiente, retorna ou não para uma área previamente já visitada. Além disso, a adição de uma aresta entre dois nós não sequenciais cria um fechamento de laço, pois são detectadas informações por onde o

robô já havia passado anteriormente, como características do ambiente por exemplo. No entanto, um fechamento de laço confiável é necessário, porém difícil de ocorrer, sendo um dos maiores problemas no SLAM mais robusto (NEWMAN; HO, 2005).

No exemplo da Figura 2.9, o grafo de posturas conteve 1228 nós, 1483 arestas representados por pontos azuis e linhas vermelhas respectivamente e 256 fechamentos de laços detectados após o robô retornar a regiões que já havia visitado anteriormente. Com isso, pode-se perceber que, além de obter a transformação entre *scans* de posturas consecutivas, também se deve observar o histórico de posturas anteriores registradas para garantir um resultado melhor ao longo do processo de SLAM. Dessa forma, assim que o fechamento do laço for detectado, a otimização do grafo de posturas é realizada, pois há maior eficiência e redução de custo computacional durante o processo do SLAM (GRISSETTI *et al.*, 2010).

A Figura 2.10 mostra em (a) um exemplo de mapa construído somente com o algoritmo de SLAM antes do processo de fechamento do laço, em (b) a detecção do fechamento de laço numa determinada região do mapa e em (c) o mapa completo da região do ambiente analisada após o processo de fechamento do laço.

2.5 RANSAC

O algoritmo de RANSAC (*RANdom Sample And Consensus*) foi proposto, inicialmente, por Bolles e Fischler (1981) com o objetivo de estimar os parâmetros de um modelo por meio de um conjunto de dados que possuíam um elevado número de *outliers*. Com isso, um *outlier* é um dado que não se adequa ao modelo real que foi instanciado por um conjunto de parâmetros verdadeiros que são localizados dentro de um limiar (*threshold*) de erro que define um desvio máximo antes de chegar ao ruído.

Diversas modificações no RANSAC podem ser realizadas, porém o algoritmo é formado por duas etapas que são repetidas iterativamente. A primeira é denominada de **Hipótese**, a qual seleciona os primeiros conjuntos de amostras mínimas de forma aleatória do conjunto de dados de entrada e os parâmetros do modelo são calculados com o uso apenas dos elementos do conjunto de amostras mínimas. Já a segunda é conhecida como **Tese**, em que nessa parte do algoritmo, o RANSAC verifica quais elementos do conjunto de dados são consistentes com o modelo instanciado com os parâmetros estimados na primeira etapa e são chamados de conjunto de consenso. Com isso, o RANSAC se encerra quando a probabilidade de se obter o conjunto de consenso melhor classificado for menor que um certo limite (ZULIANI, 2008).

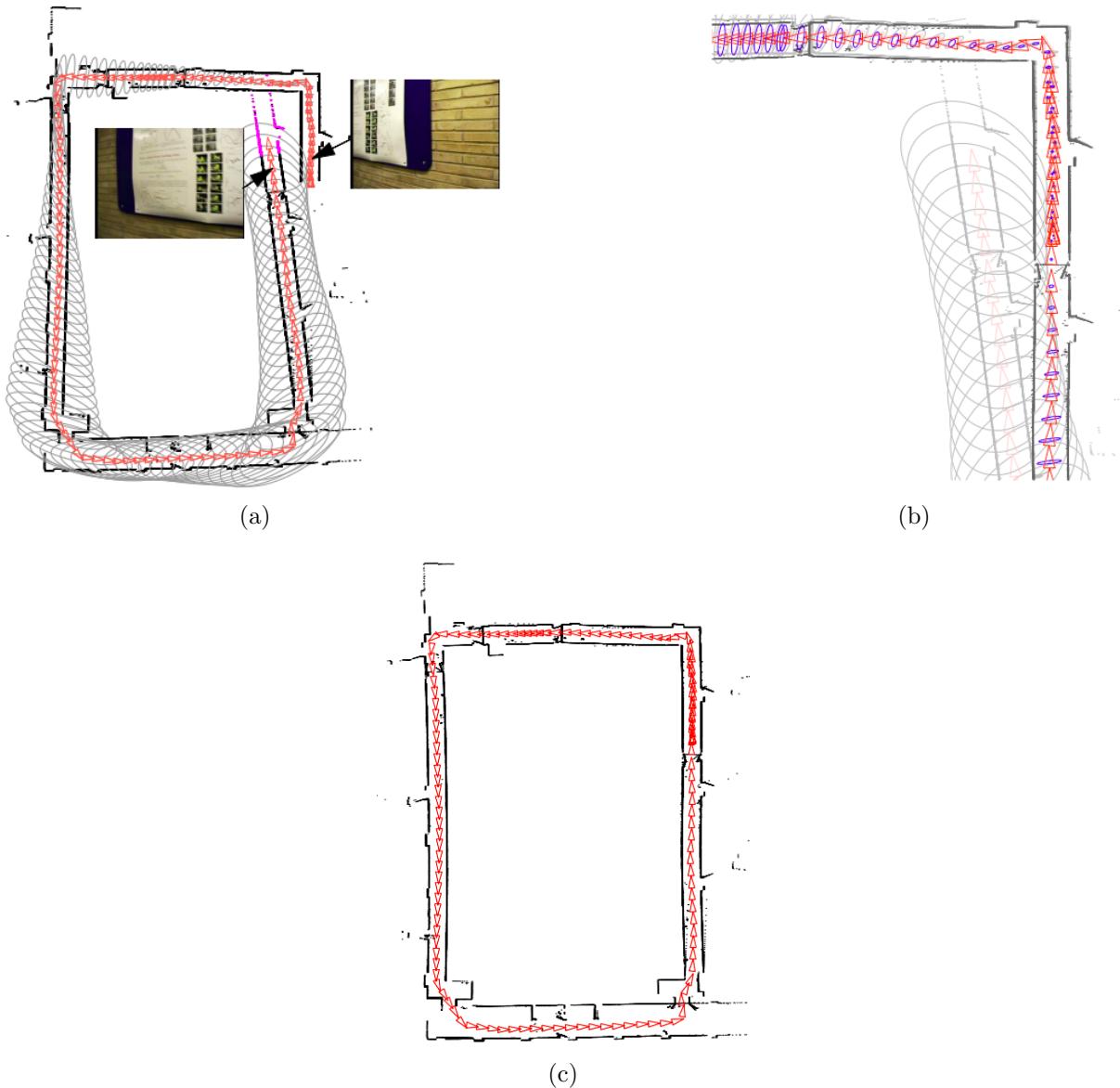


FIGURA 2.10 – (a) Mapa construído por meio do algoritmo de SLAM antes do fechamento de laço, (b) Região no mapa onde há o ponto de fechamento do laço e (c) Mapa completo da região analisada após o fechamento de laço (NEWMAN; HO, 2005).

2.5.1 Distância entre ponto e linha

Na geometria euclidiana, a distância entre um ponto e uma linha é a menor distância de um determinado ponto a qualquer ponto pertencente a uma linha reta. Como modelo do algoritmo de RANSAC foi utilizada a seguinte equação para calcular a distância:

$$distancia(P_1, P_2, (x_0, y_0)) = \frac{|(x_2 - x_1)(y_1 - y_0) - (x_1 - x_0)(y_2 - y_1)|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \quad (2.1)$$

onde:

- a linha é definida por um segmento de reta e destacada por dois pontos, o primeiro é $P_1 = (x_1, y_1)$ e o segundo é $P_2 = (x_2, y_2)$, sendo ambos formados por suas coordenadas x e y no plano cartesiano;
- o ponto analisado é formado pelas coordenadas do plano cartesiano (x_0, y_0) ;

A Figura 2.11 mostra o gráfico no plano cartesiano relacionado à equação da distância entre um ponto e um segmento de reta.

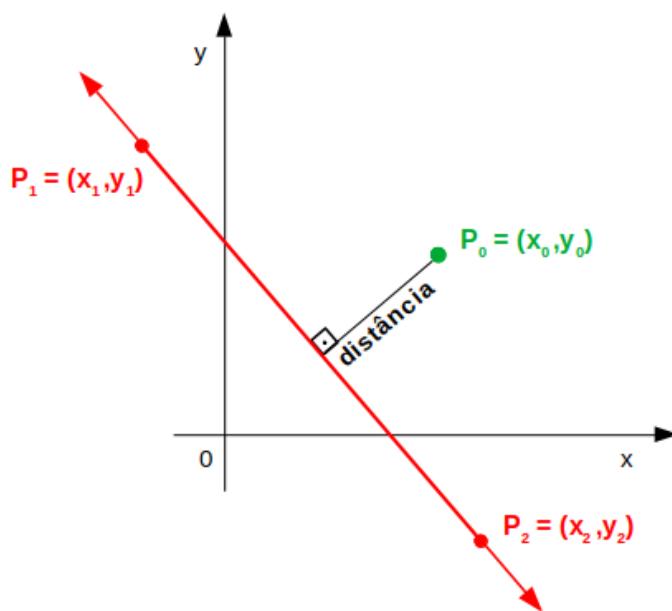


FIGURA 2.11 – Representação da distância entre um ponto e um segmento de reta no plano cartesiano.

Dante disso, esse modelo estabelece uma seleção dos pontos mais próximos de um segmento de reta por meio da delimitação de um *threshold*, assim tais pontos selecionados são atribuídos como verdadeiros ao consenso do RANSAC e considerados pontos *inliers*, enquanto que os pontos que estiverem fora do limite escolhido são considerados pontos *outliers*.

2.5.2 Ajuste de linha por mínimos quadrados

Os métodos tradicionais para realizar o ajuste de linhas por mínimos quadrados a um *cluster* de dados bidimensionais estão relacionados com a minimização da soma dos quadrados das distâncias verticais mínimas entre os pontos e a linha ajustada. Porém, isso exclui as observações independentes relacionadas ao eixo y. Com isso, pode ser modificado o processo para ajustar uma linha a um conjunto de pares ordenados com a minimização da distância por mínimos quadrados para cada ponto do conjunto de dados sem abordar a orientação (PAETH, 1995).

A equação da reta para o caso univariado é dada por:

$$y = m_y x + b_y \quad (2.2)$$

onde os coeficientes da reta m_y e b_y são calculados por:

$$m_y = \frac{N \sum x_i y_i - \sum x_i \sum y_i}{N \sum x_i^2 - (\sum x_i)^2}, \quad b_y = \frac{\sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i}{N \sum x_i^2 - (\sum x_i)^2} \quad (2.3)$$

x_i e y_i são as coordenadas de cada ponto do conjunto de dados e N é o número de pontos.

No entanto, segundo Paeth (1995), essa equação clássica não é aceita como um recurso geral para ajustar a reta, pois sua aplicação fornece resultados ruins quando as duas coordenadas são incertas ou quando a reta está próxima de ser vertical. O método é sensível à orientação do sistema de coordenadas. Dessa forma, para deixar o método insensível à orientação do sistema, minimiza-se a soma dos quadrados das distâncias perpendiculares entre os pontos dos dados e os mais próximos da reta analisada, pois as retas perpendiculares são características geométricas independentes do sistema de coordenadas.

Para isso, define-se uma linha aleatória com os parâmetros (θ, ρ) e faz-se o somatório dos quadrados das distâncias perpendiculares (r_i) entre os pontos (x_i, y_i) e os pontos mais próximos de tal reta, assim os valores de θ e de ρ são determinados por meio da minimização do somatório. A eq. (2.4) é minimizada, realizando a derivada parcial de Z com relação ao ângulo θ da reta em relação ao eixo da coordenada x do plano cartesiano e depois em relação à distância perpendicular à reta analisada e a origem que é ρ , e ambas as derivadas são iguais a zero, como mostrada na eq. (2.5).

$$Z = \sum r_i^2(\rho, \theta) \quad (2.4)$$

$$\frac{\partial Z}{\partial \theta} = 0 \text{ e } \frac{\partial Z}{\partial \rho} = 0 \quad (2.5)$$

Diante disso, a equação paramétrica da reta é dado pela eq. (2.6), definida pelo plano geométrico.

$$x \sin \theta + y \cos \theta + \rho = 0 \quad (2.6)$$

Além disso, a distância (r_i) perpendicular da linha com um determinado ponto que faz parte dos dados é definida pela eq. (2.7).

$$r_i = y_i \cos \theta - x_i \sin \theta - \rho \quad (2.7)$$

Após realizar as derivadas parciais da eq. (2.4) com a utilização da eq. (2.7), são obtidas as seguintes expressões que são as eqs. (2.8) e (2.9).

$$a \cos \theta \sin \theta + b((\sin \theta)^2 - (\cos \theta)^2) + c\rho \cos \theta + d\rho \sin \theta = 0 \quad (2.8)$$

$$d \cos \theta - c \sin \theta = N\rho \quad (2.9)$$

em que, as expressões da eq. (2.10) foram usadas como simplificação das equações mostradas anteriormente.

$$a = \sum_{i=1}^N x_i^2 - \sum_{i=1}^N y_i^2, \quad b = \sum_{i=1}^N x_i y_i, \quad c = \sum_{i=1}^N x_i, \quad d = \sum_{i=1}^N y_i \quad (2.10)$$

Por meio disso, a eq. (2.9) pode ser escrita na forma da eq. (2.11), pois \bar{x} e \bar{y} são as médias de x e y observadas, também chamadas de centroide do conjunto de dados (x_i, y_i) nas expressões da eq. (2.10).

$$\bar{x} \sin \theta - \bar{y} \cos \theta + \rho = 0 \quad (2.11)$$

Assim, a eq. (2.8) pode ser simplificada para que os dados originais sejam transladados e o centroide esteja na origem como mostra a eq. (2.12).

$$x'_i = x_i - \bar{x} \text{ e } y'_i = y_i - \bar{y} \quad (2.12)$$

Com isso, a translação resulta na expressão da eq. (2.13):

$$c' = d' = \rho' = 0 \quad (2.13)$$

Então, a eq. (2.9) é reduzida para:

$$a' \cos \theta \sin \theta + b'((\sin \theta)^2 - (\cos \theta)^2) = 0 \quad (2.14)$$

onde os parâmetros a' e b' são calculados por:

$$a' = \sum_{i=1}^N (x'_i)^2 - \sum_{i=1}^N (y'_i)^2, \quad b' = \sum_{i=1}^N x'_i y'_i \quad (2.15)$$

Por meio das equações anteriores, a equação da reta pode ser determinada por meio das expressões dos coeficientes A , B e C observadas na eq. (2.16).

$$A = 2b', \quad B = -(a' + \sqrt{(a')^2 + 4(b')^2}), \quad C = A\bar{x} + B\bar{y} \quad (2.16)$$

Em decorrência de todo o processo descrito anteriormente por Paeth (1995), pode-se visualizar na Figura 2.12 a representação da reta ajustada e definida por meio do processo de mínimos quadrados.

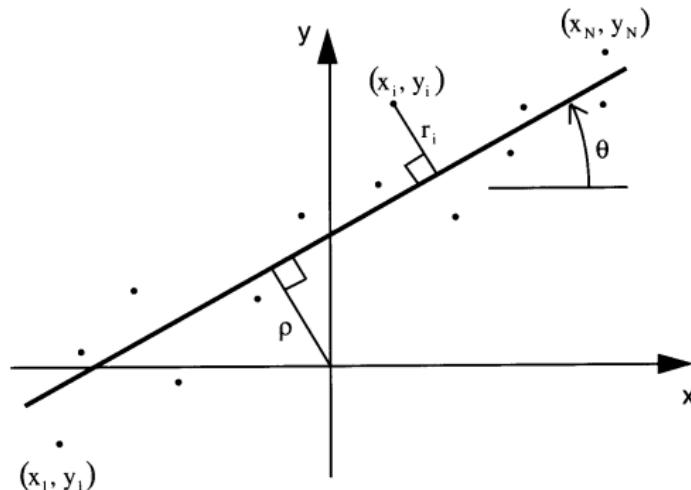


FIGURA 2.12 – Visualização da reta ajustada pelo método de mínimos quadrados (PAETH, 1995).

2.5.3 Extração de segmentos de reta

Diante das seções anteriores, o algoritmo de RANSAC foi utilizado para ajustar segmentos de reta, baseado nos dados coletados pelo *scanner a laser* do robô que representam as paredes e obstáculos do ambiente em forma de nuvem de pontos que contém ruído, já que o sensor não é ideal. Com isso, o conjunto de pontos extraído é convertido para segmentos de retas ajustados, em que cada segmento detectado possui dois pontos terminais (*endpoints*) para delimitá-lo.

No entanto, os segmentos de reta encontrados pelo RANSAC captam todos os pontos próximos a reta detectada que foram reconhecidos como pontos *inliers*. Isso nem sempre está correto, pois o segmento de reta pode detectar duas paredes alinhadas como se fosse apenas uma parede, assim é construído um algoritmo para melhorar esse resultado do RANSAC a fim de delimitar as linhas da forma mais adequada. A Figura 2.13 mostra um exemplo de mapa com a extração de segmentos de reta delimitados por pontos terminais. No próximo capítulo, será detalhado o algoritmo desenvolvido para essa etapa.

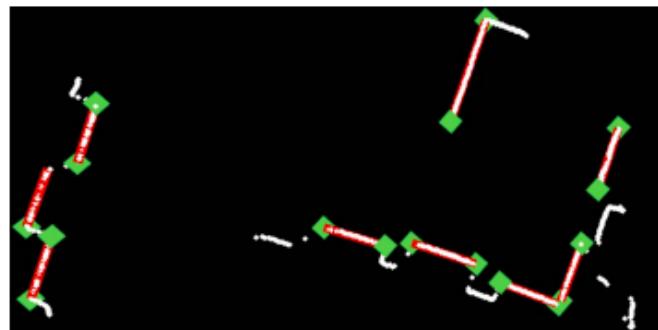


FIGURA 2.13 – Extração de segmentos de reta delimitados por pontos terminais direcionais, em que as paredes do ambiente são visualizadas como linhas brancas no mapa, as linhas extraídas são representadas por linhas vermelhas e os pontos terminais por quadrados verdes (GAO *et al.*, 2018).

2.6 *Scan Matching* e *Feature Matching*

A correspondência de varredura (*scan matching*) é de suma importância para o posicionamento e modelagem de um ambiente, sendo capaz de recuperar a posição e orientação entre dois *scans* de posturas diferentes com o uso de pontos correspondentes. O SLAM baseado no *scan matching* estima uma transformação rígida para projetar um *scan* que se alinhe com outro, formando o *matching* entre os conjuntos de pontos entre posturas consecutivas.

A correspondência de características (*feature matching*) é um método que compara os dados medidos pelos *scanners a laser* do robô com as características permanentes do mapa global de referência do ambiente que são paredes ou pilares, por exemplo. Uma vantagem dessa técnica é que, no lugar de pontos obtidos do *scanner*, as características conseguem se diferenciar entre si, devido ao tamanho e ângulo específicos de cada um, sendo mais simples de gerar a identificação e combinação entre eles, além de serem menos sensíveis a ruído.

Além disso, o mapa resultante feito com *scan matching* compara ponto a ponto a varredura de diferentes sistemas de coordenadas do robô, porém esta técnica apresenta um maior gasto computacional. Já, o mapa final feito com correspondência de características apresenta apenas as coordenadas das características estáticas e permanentes, sendo menos

dados analisados do que a correspondência de varredura, o que gera um menor gasto computacional.

Na Figura 2.14, as linhas pretas representam as paredes de um exemplo de ambiente, em (a) as linhas azuis e vermelhas pontilhadas evidenciam os *scans* adquiridos da postura **i** e **j** respectivamente, além das linhas tracejadas azuis e vermelhas que representam as linhas do *scanner* a *laser* do sensor nas diferentes posturas mostradas. Já em (b) as linhas azuis e vermelhas contínuas representam os *scans* nas posturas **i** e **j** respectivamente e demonstram linhas como características obtidas com a correspondência de características. Tais características em (b) variam em relação ao deslocamento do robô na parede superior, porém na parede inferior do ambiente não há variação, pois depende da posição da parede e da postura do veículo (AGHAMOHAMMADI *et al.*, 2007).

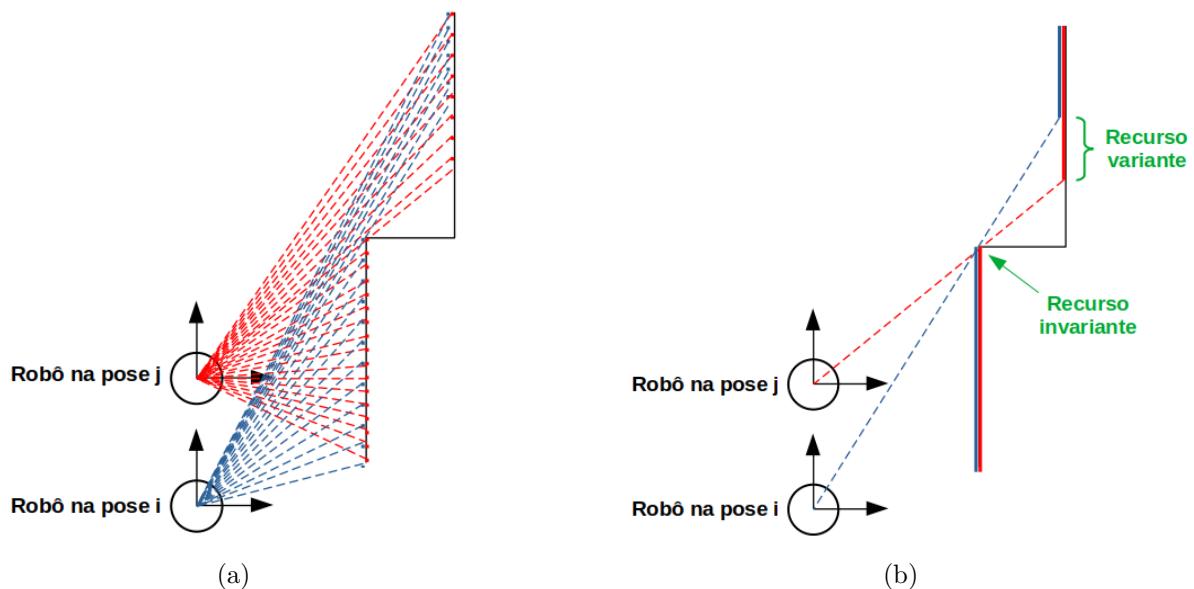


FIGURA 2.14 – Diferença entre as técnicas de correspondência de varredura (a) por meio da detecção de pontos característicos e de correspondência de características (b) a partir da detecção de linhas como característica (AGHAMOHAMMADI *et al.*, 2007).

2.6.1 Transformação de Coordenadas para Alinhamento de Scans

Para que seja feito o alinhamento de *scans*, é necessário realizar a transformação de dois *frames* de posturas consecutivas. Para coordenadas 2D, a transformação é feita por meio de uma matriz de rototranslação, conforme evidenciada na eq. (2.19). Tal equação pode ser obtida a partir da matriz de rotação e do vetor de translação observados pelas eqs. (2.17) e (2.18) (SOLÀ, 2017).

$$R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (2.17)$$

$$t = \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} d \cos \theta \\ d \sin \theta \end{bmatrix} \quad (2.18)$$

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & t_x \\ \sin \theta & \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.19)$$

A transformação é feita por três parâmetros que representam o ângulo de rotação (θ), a translação no eixo x (t_x) e a translação no eixo y (t_y) entre dois *frames* de posturas, d é a distância entre duas posturas consecutivas. Por meio da matriz T , pode-se obter informações do histórico de posturas e do mapa do robô. Os pontos 2D $[x, y]^T$ em coordenadas homogêneas podem ser transformados, multiplicando a matriz T como é evidenciado na eq. (2.20),

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (2.20)$$

onde x' e y' são as novas coordenadas do ponto 2D transformado.

De acordo com Biber e Straßer (2003), o objetivo do alinhamento de *scans* é recuperar os parâmetros anteriores, utilizando as varreduras a *laser* feitas em duas posições do veículo. Outra forma de visualizar as equações anteriores está na eq. (2.21).

$$T : \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad (2.21)$$

Além disso, o modelo de observação referente à localização de cada postura é obtida de acordo com o *frame* da coordenada global de referência. Com isso, a primeira postura (P_1) precisa ser representada no *frame* global de referência e ser considerada a postura global do veículo após realizar a transformação por meio da eq. (2.21), para estar no mesmo *frame* global da referência. Com isso, a segunda postura (P_2) utiliza a mesma equação da transformação para se adequar ao *frame* de coordenadas da primeira postura que, consequentemente, é o mesmo *frame* da referência. Dessa forma, a terceira postura também realiza o mesmo processo com a matriz de transformação e assim por diante com as demais posturas presentes no grafo (NIETO *et al.*, 2007). Diante disso, a Figura 2.15 ilustra duas posturas (P_1 e P_2) com relação a um *frame* global de referência, além do ângulo θ_G que é o ângulo de rotação da postura P_1 com relação ao *frame* global, do ângulo θ que é o ângulo de rotação da postura P_2 relativa à postura P_1 , e das translações t_x e t_y nos dois eixos que são baseadas na distância d entre as posturas.

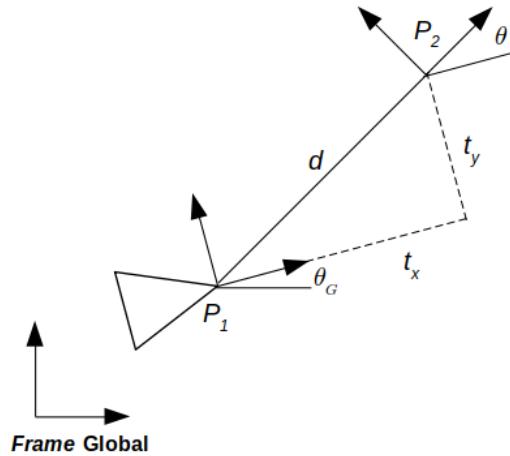


FIGURA 2.15 – Duas posturas relacionadas ao *frame* global de referência (NIETO *et al.*, 2007).

A Figura 2.16 ilustra um exemplo de alinhamento de *scans*, em que (a) pode-se observar o *scan* feito em duas posturas diferentes e, em (b), observa-se o alinhamento de *scans* após a transformação do *scan* vermelho para o *scan* preto com intuito de estarem no mesmo *frame* de coordenadas.

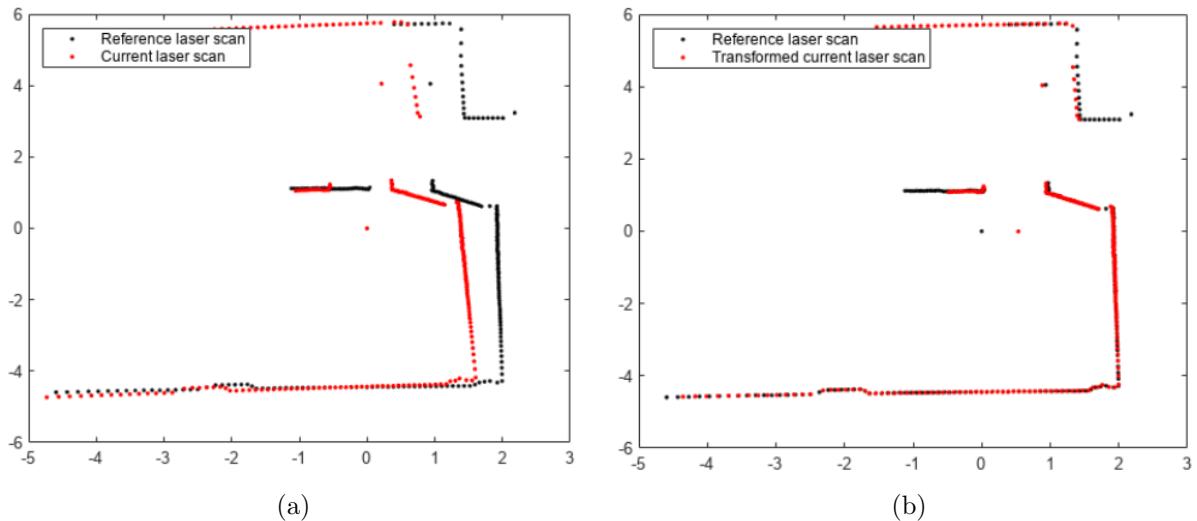


FIGURA 2.16 – Exemplo de alinhamento entre *scans*, em que (a) mostra dois *scans* em diferentes posturas antes do alinhamento e (b) evidencia os *scans* alinhados (MATHWORKS, 2022a).

2.6.2 Feature Map

O mapa de características (*feature map*) é um tipo de representação do estado do ambiente, o qual o robô utiliza para se localizar e se movimentar. Esse tipo de mapa armazena um conjunto de características distintas, como pontos ou linhas, por exemplo. Uma vantagem desse tipo de mapa é o gasto reduzido de recursos de memória, porém uma desvantagem é que o mapa apresenta um nível de detalhes prejudicado, pois a repre-

sentação do ambiente é simplificada (STACHNISS, 2006).

A Figura 2.17 representa um exemplo de mapa de características que utiliza segmentos de retas e pontos terminais como características extraídas para sua construção. A linha vermelha e a azul evidenciam o segmento de reta abstraído e a fronteira ou obstáculo presente no mapa, respectivamente, assim como a estrela vermelha e a azul que evidenciam um *endpoint* direcional e um *endpoint* instável respectivamente. E, por fim, as posturas do robô são denotadas pelos círculos verdes, enquanto que as linhas pretas mostram a trajetória do veículo (GAO *et al.*, 2018).

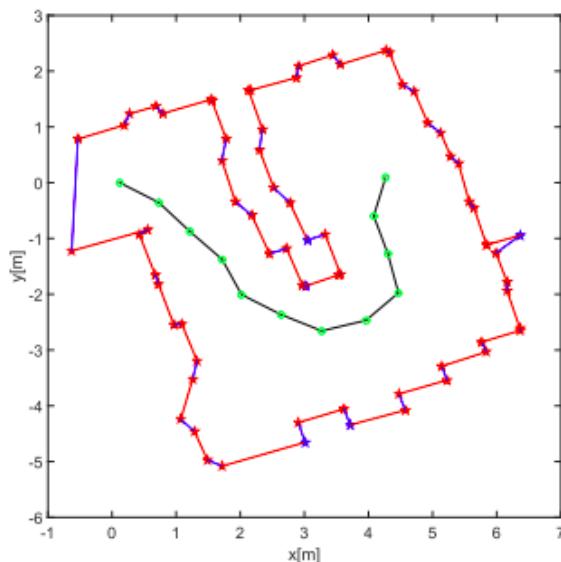


FIGURA 2.17 – Mapa de características, no qual a estrela vermelha representa a característica do *endpoint* direcional, a estrela azul denota o *endpoint* instável, a linha vermelha evidencia o segmento de reta extraído e a linha azul é a fronteira ou obstáculo no mapa e a postura do robô é indicada pelo círculo verde e o caminho do robô é representado pela linha preta (GAO *et al.*, 2018).

2.6.3 Correspondência de Segmentos de Reta

A correspondência por segmentos de reta é um tipo de *matching* de características que verifica se pode existir a combinação de mais de uma linha. Existem diversas formas de realizar essa correspondência, mas as linhas nem sempre apresentam o mesmo tamanho, pois depende da captação de regiões do ambiente pelo *scanner* a *laser* o que retrata que as linhas podem apresentar comprimentos aproximados ou uma linha ser sobreposta apenas em uma parte da outra. Além da linha propriamente dita, existe uma característica que é chamada de *endpoint* que representa a delimitação dos segmentos de reta e é essencial para saber onde é o início e o fim do segmento. Outra informação importante para realizar o *matching* entre as linhas é o ângulo que a linha apresenta com relação ao eixo das abscissas no plano cartesiano, se esse ângulo das linhas analisadas for próximo é possível associá-las.

Além disso, como o ambiente contém a maioria das paredes ortogonais entre si, também se pode detectar os cantos no mapa que é a intersecção entre as paredes. Tal característica é capaz de melhorar o processo de correspondência de características. Dessa forma, foi desenvolvido um código relacionado à detecção de cantos que é descrito com detalhes no próximo capítulo.

2.7 *Robot Operating System*

O *Robot Operating System* (ROS) é considerado um sistema meta-operacional de código aberto desenvolvido para a área da robótica. Isso implica em disponibilizar alguns serviços comuns de um sistema operacional, como a abstração de *hardware*, o controle de equipamentos de baixo nível, a implementação de aplicações usadas de forma comum, além da troca de mensagens entre sistemas e o gerenciamento de pacotes. Paralelo a isso, também são oferecidas ferramentas e bibliotecas capazes de escrever, obter, compilar e executar códigos por meio de vários computadores.

Dante da complexidade presente na elaboração de projetos tecnológicos, da existência de dispositivos programáveis com algoritmos específicos para determinado sistema e do tamanho de códigos desenvolvidos para programarem sensores e outros dispositivos que fazem parte da robótica, se viu a necessidade de se utilizar o ROS, pois possui o objetivo de oferecer suporte ao reaproveitamento de algoritmos em pesquisas relacionadas à robótica, sendo uma plataforma *open source* que é capaz de resolver problemas com uma certa economia de tempo (QUIGLEY *et al.*, 2009).

O ROS permite a troca de informações entre processos localizados em regiões diferentes de um sistema a fim de garantir que dados sejam enviados e recebidos entre elas. Geralmente, os arquivos desenvolvidos na rede ROS possuem suporte para as linguagens C++ e *Python*.

A rede ROS pode ser formada por diversos computadores que se comunicam entre si, porém uma dessas máquinas é o mestre que é responsável pela coordenação das regiões da rede, dos processos, além da inicialização de todos os serviços que garantam a comunicação adequada dos nós da rede. A máquina mestre é reconhecida por uma URI (*Uniform Resource Identifier*) mestre que possui o endereço IP (*Internet Protocol*) da máquina que está executando o mestre ou o nome do *host*.

Além disso, a arquitetura ROS contém três níveis de conceitos que são: o nível do sistema de arquivos, o nível do grafo computacional e o nível de comunidade. O primeiro nível possui a função de organizar a estrutura interna do ROS para ordenar as pastas e arquivos no disco rígido, é formado por pacotes, metapacotes, manifestos de pacotes, repositórios, tipos de mensagens e tipos de serviços. O segundo nível é formado pela rede

ponto a ponto dos processos ROS, é composto por nós que são os processos executáveis do sistema e pelo vértice que evidencia as comunicações estabelecidas entre os nós. E o terceiro nível engloba o compartilhamento de *software*, informações, dados e conhecimentos entre desenvolvedores, usuários e apoiadores da comunidade em *sites*, fóruns, distribuições e repositórios *online*, por ser uma plataforma de código aberto.

Dentre os níveis citados anteriormente, leva-se em consideração as principais definições do nível de grafo computacional mostradas abaixo, que são de suma importância para o entendimento da rede ROS.

- Nós (*nodes*) - são arquivos que executam tarefas ou processos do sistema;
- Mestre (*master*) - o mestre da rede ROS fornece o registro de nome para a pesquisa em todo o resto do grafo computacional, é necessário para interligar os processos e gerenciar as tarefas que são executadas na rede;
- Servidor de parâmetros (*parameter server*) - Faz parte do mestre e armazena os dados por chave em uma região central;
- Mensagens(*messages*) - são estruturas de dados que abrangem conjuntos de informações, apresenta diversos tipos como *int*, *float*, *bool*, etc.;
- Tópicos (*topics*) - são os meios de transporte das mensagens de mesmo tipo que permite a publicação/subscrição dos dados. Os nós podem publicar e/ou receber a mensagem em um mesmo tópico.
- Serviços (*services*) - são representados por um par de estrutura de mensagem, em que uma é a solicitação e a outra é a resposta;
- *Bags* - é um formato de arquivo que salva e reproduz dados das mensagens da rede ROS. É importante para armazenamento de informações, depuração de códigos, auxílio no isolamento e correção de falhas.

A Figura 2.18 representa a troca de dados na rede ROS a partir de tópicos ou serviços. A comunicação é feita pelo protocolo TCPROS (*TCP Protocol ROS*) que se fundamenta no protocolo TCP/IP (*Transmission Control Protocol/Internet Protocol*) para realizar o transporte de dados.

De acordo com a Figura 2.18, podem ocorrer três tipos de situações com os componentes da rede ROS, na primeira, um nó pode enviar (*publisher*) os dados por meio do tópico para outro nó, em que o segundo nó recebe a informação (*subscriber*). Já na segunda, o primeiro nó pode receber (*subscriber*) informações enviadas pelo segundo nó (*publisher*) por meio do tópico. Finalmente, na terceira situação, o primeiro nó pode enviar e receber

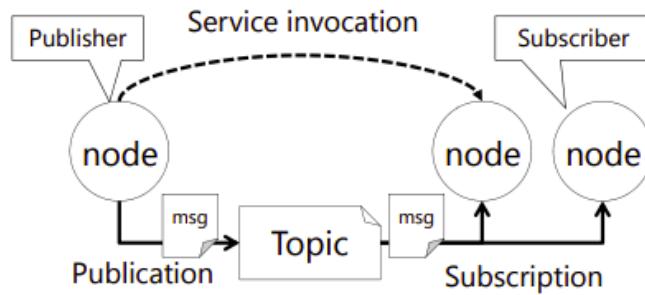


FIGURA 2.18 – Modelo de comunicação da rede ROS (YAMASHINA *et al.*, 2016).

dados vindos do segundo nó, transportados pelo mesmo tópico, nesse caso ambos os nós são nós *publisher* e *subscriber* ao mesmo tempo.

3 Proposta de Solução

Neste capítulo, é apresentada uma descrição sobre o robô utilizado no trabalho, além das definições e a implementação da proposta de solução para o SLAM com a técnica de correspondência de características para aplicação no robô móvel. O SLAM baseado em correspondência de características é responsável por estimar as posturas do robô e o mapa do ambiente mediante características processadas a partir dos dados obtidos pelos sensores embarcados no robô.

Nas próximas seções será realizado um breve detalhamento sobre o robô móvel e as etapas do projeto que se resumem à obtenção dos dados pelos sensores do robô, ao processamento desses dados pelo algoritmo de RANSAC, à adaptação desse algoritmo para detecção dos segmentos de reta que são as representações das paredes (características) do ambiente, além da detecção dos cantos das paredes que são essenciais para o processo do SLAM, a explicação sobre o algoritmo de SLAM com correspondência de características, o fechamento do laço e por fim à redução do ruído presente no mapa global gerado.

3.1 Estrutura do SLAM com correspondência de características

Este trabalho se baseia no caso de SLAM 2D que forma o grafo de posturas e o mapa global estimados em um plano bidimensional. Esta seção apresenta o propósito de discutir todas as condições e técnicas usadas para atingir o objetivo da tarefa.

3.1.1 Objetivo da tarefa

O objetivo é operar um robô móvel em um ambiente desconhecido para que realize o processo de SLAM a partir de um grafo de posturas a fim de conseguir se localizar e construir o mapa global referente ao ambiente por meio da técnica de correspondência de características. Os principais recursos abordados foram os segmentos de reta, como também pontos terminais que delimitam as linhas e cantos que interceptam as paredes,

ou seja, os pontos terminais podem ser considerados cantos.

3.1.2 Premissas do problema

Os requisitos ou aproximações estabelecidos para o desenvolvimento deste trabalho são os seguintes:

1. O ambiente possui terreno plano em todo o espaço disponível para realização da tarefa.
2. O ambiente possui paredes ortogonais ao plano.
3. O ambiente é estático.
4. O ambiente é globalmente fechado e aberto localmente, o que significa que o robô não pode escapar do espaço total disponível e não pode ficar preso em um local específico, como uma sala.
5. O robô contém comunicação sem fio com um computador central em todos os momentos para troca e extração de dados.
6. A posição inicial do robô é desconhecida.

Os experimentos foram realizados conforme as condições estabelecidas anteriormente a fim de testar a solução proposta e tornar a aplicação viável.

3.1.3 Algoritmo de RANSAC modificado

Para que o veículo obtenha dados do ambiente, utilizam-se sensores para medição do ambiente acoplados ao robô. O sensor empregado para esse propósito foi o *scanner a laser* (LiDAR). Tal sensor gera dados no formato de nuvem de pontos 2D que mostram como o robô enxerga o ambiente a partir do centro desse sensor, como pode ser evidenciado na Figura 3.1, em que os pontos 2D vermelhos representam a visão do ambiente pelo sensor do robô localizado em uma certa postura no ambiente analisado.

Diante disso, cada dado da nuvem retorna a distância e o ângulo em relação ao centro do sensor. Tais informações obtidas dos pontos estão em coordenadas polares e que são transformadas para coordenadas retangulares (x, y) e depois são processadas por meio do algoritmo de RANSAC (*Random sample Consensus*), que é um método iterativo de estimativa de parâmetros, para definir as coordenadas retangulares dos pontos terminais de cada linha detectada pelo algoritmo. A Figura 3.2 mostra os segmentos estimados pelo RANSAC, em que os círculos representam os pontos coletados do *scan* em uma certa

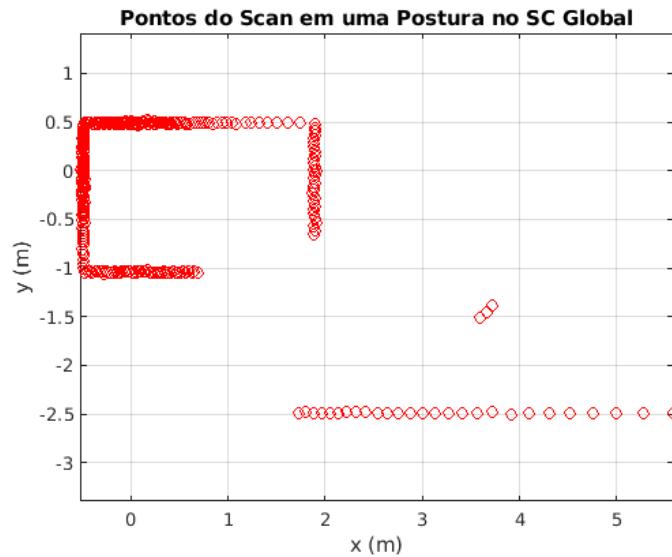


FIGURA 3.1 – Leitura do sensor *scanner a laser* em uma postura do robô no Sistema de Coordenadas Global.

postura do robô, enquanto que as linhas coloridas definem cada um dos segmentos de reta ajustados pelo algoritmo.

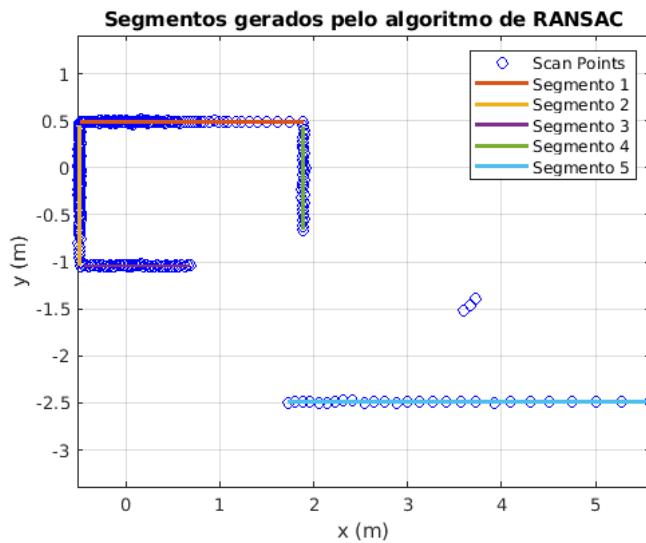


FIGURA 3.2 – Segmentos de reta estimados pelo algoritmo RANSAC a partir dos dados obtidos pelo *scanner a laser*.

Além disso, a Figura 3.2 foi construída após o tratamento das linhas estimadas pelo RANSAC, pois qualquer ponto localizado dentro do limite de distância do ajuste da linha era considerado um *inlier*. Dessa forma, duas paredes do ambiente alinhadas eram consideradas uma única linha estimada, por isso, os segmentos de reta foram melhorados em uma função desenvolvida para detecção das linhas.

A função desenvolvida para detecção das linhas ajusta o tamanho da linha até onde são detectados pontos *inliers*, assim a região em que não há pontos é uma área limite

do segmento, o que faz a linha ser dividida. Para isso, a linha analisada é discretizada e os pontos considerados *inliers* são sobrepostos à linha discretizada. Após a região que estiver sem pontos for detectada, os pontos que são encontrados depois dessa região vazia são considerados novos *outliers* e são incluídos na matriz de pontos *outliers* já existente para serem verificados novamente no algoritmo de RANSAC.

Esse processo explicado pode ser observado na Figura 3.3, em que os círculos alaranjados representam todos os pontos analisados antes da detecção, os círculos verdes evidenciam os pontos detectados como *inliers* e os círculos vermelhos são os pontos considerados *outliers*, como também a linha tracejada que é a linha ajustada pelo RANSAC que foi discretizada e a linha verde é a linha ajustada após o processo de detecção de linhas.

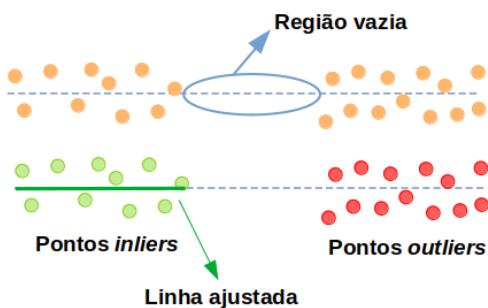


FIGURA 3.3 – Ilustração sobre o processo do algoritmo de detecção de linhas para corrigir as linhas ajustadas e obtidas pelo algoritmo RANSAC.

Com os segmentos de reta regulados depois de passarem pela função de detecção e ajuste dos segmentos, elaborou-se a função para adaptar os segmentos de reta ajustados anteriormente a fim de reconhecer os cantos entre as paredes próximas. Para isso, tal função se resumiu a reconhecer qual ponto terminal de um segmento do *scan* era mais próximo do ponto terminal de outro segmento. Essa etapa foi feita para todos os pontos terminais dos segmentos analisados.

Em seguida, aumentou-se o tamanho dos segmentos que foram associados pelos pontos terminais para verificar o ponto de intersecção entre as linhas próximas e, assim os pontos terminais das linhas foram novamente adequados para simular os cantos entre as paredes do ambiente, sem modificar os coeficientes da equação de cada linha. A Figura 3.4 ilustra os segmentos de reta com os cantos destacados pelos asteriscos vermelhos.

De acordo com as etapas descritas anteriormente, dentre elas, o algoritmo de RANSAC tradicional, a adaptação dos segmentos pela função de detecção de linhas e o ajuste dos segmentos com a função de detecção de cantos, foi gerado o fluxograma mostrado na Figura 3.5 e que resume as partes explicadas.

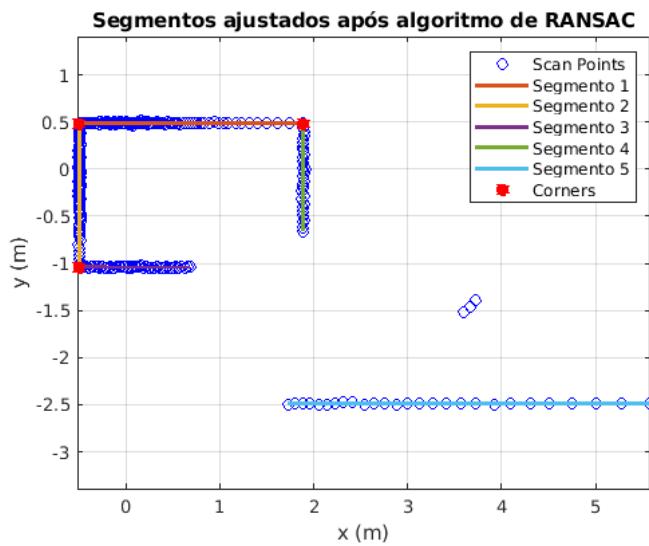


FIGURA 3.4 – Segmentos de reta adaptados do algoritmo de RANSAC com seus cantos destacados.

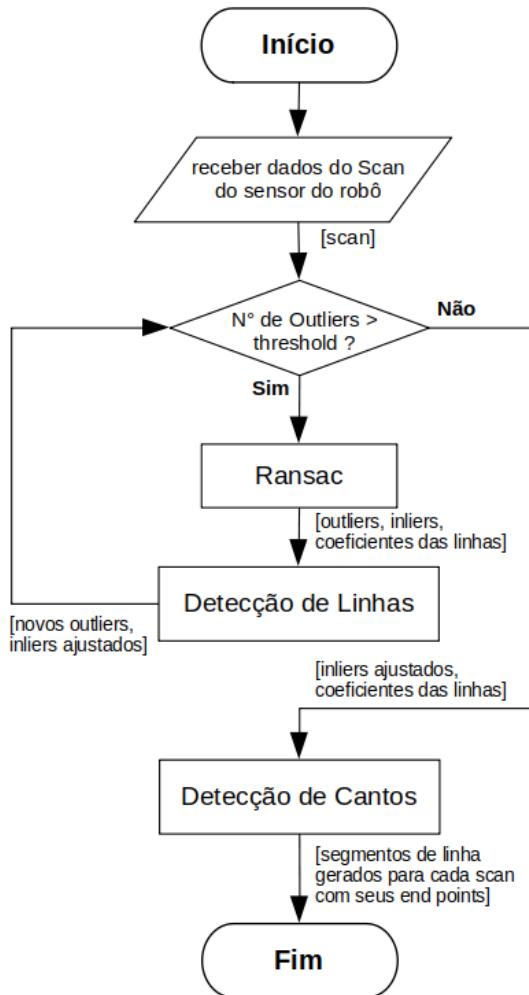


FIGURA 3.5 – Fluxograma do algoritmo RANSAC adaptado para detecção de segmentos de reta e cantos.

3.1.4 Algoritmo de Correspondência de Características

Neste algoritmo foi utilizado apenas o sensor *scanner a laser* (LiDAR) para utilização e processamento dos dados coletados. Após a extração do pontos obtidos pelo *scan* gerado pelo sensor LiDAR, fez-se a transformação dos pontos do *scan* para linhas, em seguida, fez-se a extração dos segmentos de reta e pontos terminais de cada *scan* referente a uma determinada postura do robô móvel por meio do algoritmo de RANSAC. Tais recursos gerados foram encaminhados para o algoritmo de execução do SLAM 2D com correspondência de características.

Esse algoritmo se inicia com o carregamento de todos os segmentos de reta e pontos terminais de cada *scan* nas posturas em uma estrutura referente ao ambiente. Após isso, é realizado o cálculo da rotação e translação entre *scans* de posturas consecutivas, verificando assim cada *endpoint* de cada segmento da postura_i com os da postura_{i+1}. Por exemplo, se a postura atual possui M segmentos, ela apresenta 2*M pontos terminais, enquanto que se a postura anterior à postura atual apresenta N segmentos, ela contém 2*N pontos terminais.

Dessa forma, em eq. (3.1) é feita uma otimização da função não linear da transformação de coordenadas a fim de encontrar o seu mínimo local e de realizar o *matching* entre os segmentos de reta, em que a solução é a minimização do ângulo de rotação (θ) e da translação (d), x' e y' são as coordenadas dos pontos terminais do segmento verificado do *scan* da postura anterior, como também x e y que são as coordenadas dos pontos terminais do segmento analisado do *scan* na postura atual. Com isso, cada solução encontrada e adequada para essa minimização da função foi armazenada em uma matriz.

$$f(\theta, d) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d \cos \theta \\ d \sin \theta \end{bmatrix} - \begin{bmatrix} x' \\ y' \end{bmatrix} \quad (3.1)$$

Para se obter a solução mais apropriada com melhor rotação e translação, calculou-se a medida de correspondência (MC) entre os segmentos de posturas consecutivas, utilizando as propostas de solução encontradas pela função de otimização. Diante disso, o valor de MC é obtido pela adição de uma função gaussiana da distância entre os pontos terminais de cada par de segmentos das posturas analisadas tanto na ordem direta quanto inversa da linha, conforme pode ser observado na Figura 3.6.

A eq. (3.2) se refere à distância (d_{ij}) entre um dos pontos terminais de um segmento com um dos pontos terminais de outro segmento analisados. Já a eq. (3.3) seleciona a distância mínima (d_{min}) entre as distâncias dos pontos terminais na ordem direta e inversa calculadas anteriormente, pois a menor distância entre os pontos terminais se refere à melhor comparação entre dois segmentos. A eq. (3.4) mostra que a distância mínima é

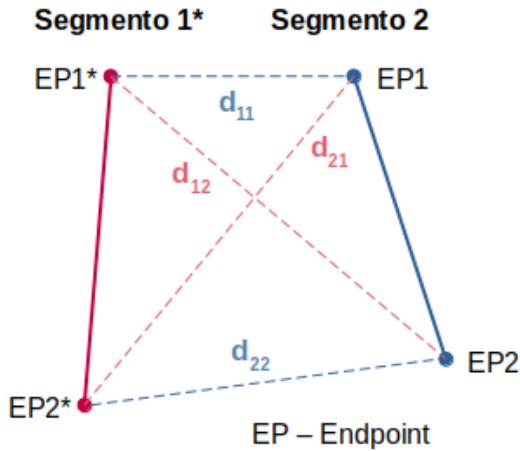


FIGURA 3.6 – Distâncias entre pontos terminais de dois segmentos para se obter a medida de correspondência dos segmentos de reta.

utilizada na função gaussiana para depois ser aplicada ao somatório das funções gaussianas das distâncias comparativas entre segmentos de dois *scans* correspondentes. Em suma, a medida de correspondência é feita para cada combinação de segmentos contidos em *scans* consecutivos que foi adquirida uma solução de rotação e translação. Finalmente, a MC é maximizada para gerar a solução mais adequada na comparação entre os *scans* de posturas consecutivas.

$$d_{ij} = \sqrt{(EP_{ix}^* - EP_{jx})^2 + (EP_{iy}^* - EP_{jy})^2} \quad (3.2)$$

$$d_{ij}^{min} = \min [(d_{11})^2 + (d_{22})^2, (d_{12})^2 + (d_{21})^2] \quad (3.3)$$

$$MC = \max(\sum_i \sum_j \exp [-(d_{ij}^{min})^2]) \quad (3.4)$$

Além disso, define-se o índice de cada segmento pertencente ao *scan* da postura atual que é associado ao índice referente ao segmento do *scan* da postura anterior e vice-versa, e também são determinados os segmentos do *scan* da postura atual que não possuem relação com nenhum segmento da postura anterior e que, por isso, são considerados novos segmentos.

Por meio disso, a rotação, a translação e os índices dos segmentos de reta da postura atual com os da anterior, assim como da postura anterior em relação à atual, são empregados para se estimar as posturas e definir o mapa global do ambiente para se completar o SLAM 2D sem fechamento de laço. Dessa forma, a postura inicial é considerada a postura com posição de 0 m tanto no eixo das abscissas quanto no eixo das ordenadas e orientação de 0°. Com isso, os segmentos de reta relacionados à postura inicial são os segmentos de

referência para a construção do mapa global sem fechamento do laço.

A partir disso, as posturas estimadas são calculadas por meio da rotação (ângulo) e da translação (distância) obtidas como solução mais apropriada para cada associação de *scans* em duas posturas consecutivas no processo de otimização evidenciado anteriormente, as posturas estimadas foram transformadas para o sistema de coordenadas da postura inicial, bem como os segmentos de reta dessas posturas. Por meio disso, o mapa global foi se desenvolvendo e se corrigindo ao longo da trajetória formada pelo robô.

Todavia, as posturas estimadas possuem erro que vai se acumulando ao longo da estimação, assim como os segmentos transformados, o que torna o mapa global ruidoso com erro acumulativo das posturas e dos segmentos. Esse problema é muito recorrente no processo de SLAM, sendo de suma importância o fechamento do laço ao longo da obtenção da trajetória do robô a fim de diminuir o erro existente. Além disso, a correção dos segmentos no mapa global ao longo da trajetória do robô é feita por meio da junção de segmentos proposta.

3.1.5 Proposta para o Fechamento do Laço

Para a etapa do fechamento do laço, se propôs determinar a postura do robô (x_R , y_R , θ_R) em um certo sistema de coordenadas (SC) global por meio da posição conhecida de marcos (naturais ou artificiais) presentes e selecionados no ambiente a partir do dados coletados pelo sensor do robô, conforme pode ser representado na Figura 3.7.

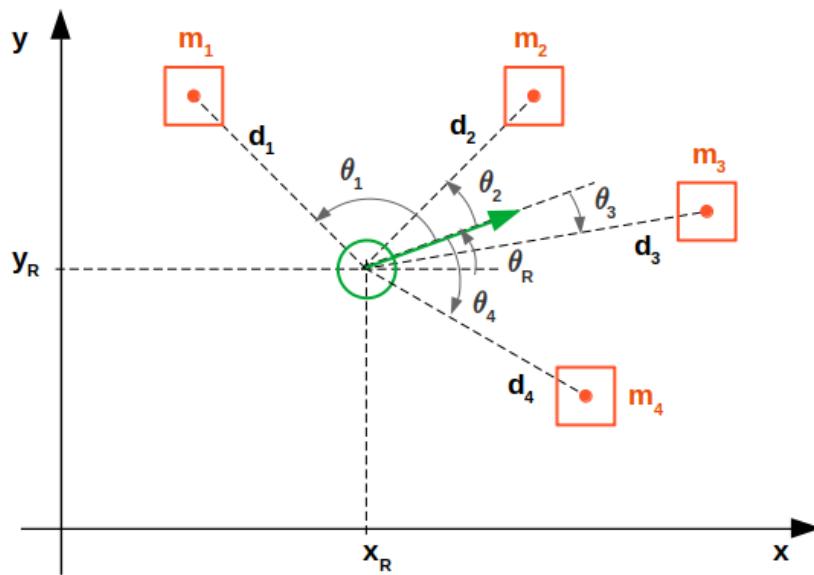


FIGURA 3.7 – Representação dos marcos com seus ângulos e distâncias em relação à uma postura real no plano cartesiano.

Dante disso, como dados do problema, define-se a posição (x , y) dos N marcos (m), a rotação medida no ambiente real para o robô apontar para cada marco através da postura

real do robô e a distância medida no ambiente real entre o veículo e cada marco diante da postura real do robô. Assim, os dados podem ser exemplificados nas expressões na eq. (3.5).

$$\begin{aligned} m : & (x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_N, y_N) \\ & \theta_1^r, \theta_2^r, \theta_3^r, \dots, \theta_N^r \\ & d_1^r, d_2^r, d_3^r, \dots, d_N^r \end{aligned} \quad (3.5)$$

Dessa forma, foram sugeridas as seguintes etapas para o fechamento do laço.

1. Para cada marco, usou-se (x_R, y_R, θ_R) e θ_i^r a fim de calcular a posição estimada do *endpoint* selecionado (x_i^c, y_i^c) a partir das expressões em 3.6, ou seja, para $i = 1$ a N , além da distância obtida através do cálculo do erro entre a posição real do marco e a estimada do *endpoint* selecionado na eq. (3.7).

$$\begin{aligned} x_i^c &= x_R + \cos(\theta_i^r + \theta_R) \\ y_i^c &= y_R + \sin(\theta_i^r + \theta_R) \end{aligned} \quad (3.6)$$

$$e_i = \sqrt{\left(x_i^m - x_i^c \right)^2 + \left(y_i^m - y_i^c \right)^2} \quad (3.7)$$

2. Definiu-se o escalar J por meio das expressões na eq. (3.8).

$$J = \sum_{i=1}^N e_i^2 = \sum_{i=1}^N (x_i^m - x_i^c)^2 + (y_i^m - y_i^c)^2 \quad (3.8)$$

3. Empregou-se um procedimento numérico para obter (x_R, y_R, θ_R) que minimize o escalar J por meio de alguma estimativa inicial $(x_R^0, y_R^0, \theta_R^0)$ realizada pela média entre as posições dos marcos, conforme pode ser observado nas expressões em 3.9.

$$\begin{aligned} x_R^0 &= \frac{1}{N} \sum_{i=1}^N x_i \\ y_R^0 &= \frac{1}{N} \sum_{i=1}^N y_i \\ \theta_R^0 &= 0 \end{aligned} \quad (3.9)$$

A Figura 3.8 ilustra a postura real representada pelo círculo azul, além dos marcos identificados pelos quadrados amarelos e por seus índices. Na imagem pode-se observar as posições de cada marco em relação à postura real atual.

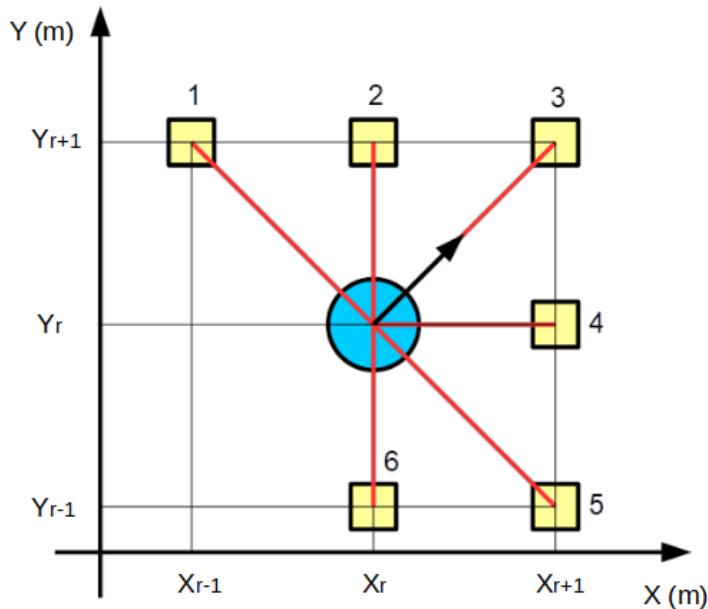


FIGURA 3.8 – Representação dos marcos pelos seus índices e suas posições no plano cartesiano.

Mediante as etapas citadas anteriormente, foi proposta uma solução de fechamento de laço para resolver o problema de erro acumulativo. Primeiramente, os pontos terminais isolados ou cantos do mapa global foram adquiridos ao longo do processo de SLAM 2D e reconhecidos como marco. Em seguida, alguns desses marcos foram selecionados para serem os marcos utilizados no processo de fechamento de laço. Os marcos selecionados no mapa global são reconhecidos e comuns em alguns *scans* de posturas. Por exemplo, se forem escolhidos quatro marcos localizados no centro do mapa global, os *scans* utilizados no fechamento de laço são os que reconhecem esses marcos em seus sistemas de coordenadas local.

Dessa maneira, quando o robô está circulando pelo ambiente, ele localiza esses marcos no mapa local, medindo o ângulo e distância entre o robô no mapa local e os marcos selecionados do mapa global transferidos para o sistema de coordenadas do mapa local.

Em suma, o processo anterior descrito pode ser enumerado da seguinte forma:

- Cada *endpoint* ou canto do *scan* local da postura é identificado por um índice.
- São selecionados os marcos no sistema de coordenadas do mapa global que são de suma importância para resolver o fechamento do laço.
- Os marcos escolhidos são transformados para o sistema de coordenadas do mapa local dos *scans* das posturas utilizados no processo.
- Verifica-se qual é o *endpoint* selecionado do *scan* que é mais próximo do marco local escolhido para o fechamento de laço.

- São calculados a distância e o ângulo de cada *endpoint* selecionado para o processo com relação à postura local onde o robô está presente.
- Os segmentos do *scan* e a postura estimada são ajustados, baseando nos marcos selecionados para o fechamento de laço.

3.1.6 Proposta para a Junção de Segmentos

A junção (*merge*) de segmentos se trata de corrigir e sobrepor segmentos de linha que possuem ângulos de inclinação próximos e que a distância entre cada ponto terminal (*endpoint*) de um segmento com o outro segmento seja menor do que um limite estabelecido. Esse processo é feito para segmentos horizontais, verticais e inclinados com ângulo entre 0° e 180° . Para os segmentos horizontais, levou-se em consideração apenas as coordenadas dos pontos terminais dos segmentos no eixo das abscissas (eixo x), para os segmentos verticais, foram utilizadas apenas as coordenadas dos pontos terminais do eixo das ordenadas (eixo y) e, para os segmentos inclinados, foram empregadas as coordenadas dos pontos terminais dos segmentos em ambos os eixos (x,y).

Além disso, foram elaborados três casos de verificação se há a sobreposição (*overlap*) entre dois segmentos e, assim, executar a junção dos segmentos que são: sobreposição total, sobreposição parcial e sem sobreposição. No primeiro caso, a sobreposição total ocorre quando o primeiro segmento possui comprimento menor ou igual que o segundo segmento e a posição dos pontos terminais do segundo segmento permite que haja a junção dos segmentos e o segundo segmento se torna o segmento resultante do processo, pois seus dois pontos terminais são os pontos mais externos. Assim como, se o segundo segmento fosse menor ou igual que o primeiro segmento, o segmento resultante seria o primeiro.

Para o segundo caso, a sobreposição parcial ocorre quando parte do primeiro segmento se sobrepõe à parte do segundo segmento ou ambos os segmentos são muito próximos sem se sobrepor entre si. Assim, o segmento resultante é obtido com os pontos terminais mais externos entre os dois segmentos. Para o terceiro caso, não há sobreposição quando o primeiro segmento possuir pontos terminais muito distantes dos pontos terminais do segundo segmento, não possibilitando a junção de segmentos.

A Figura 3.9 ilustra o processo da junção de segmentos descrito anteriormente. Para o caso de sobreposição total, o primeiro segmento é o de cor vermelha e possui os pontos terminais 1 e 2, já o segundo segmento é o de cor azul com pontos terminais 3 e 4, pode-se observar que os pontos terminais 1 e 2 são mais internos e os 3 e 4 são os mais externos, por isso o segmento resultante é o segundo segmento com pontos terminais 3 e 4.

Para o caso com sobreposição, o primeiro segmento é o de cor vermelha e existem duas possibilidades de posições para o segundo segmento que são os segmentos de cor verde.

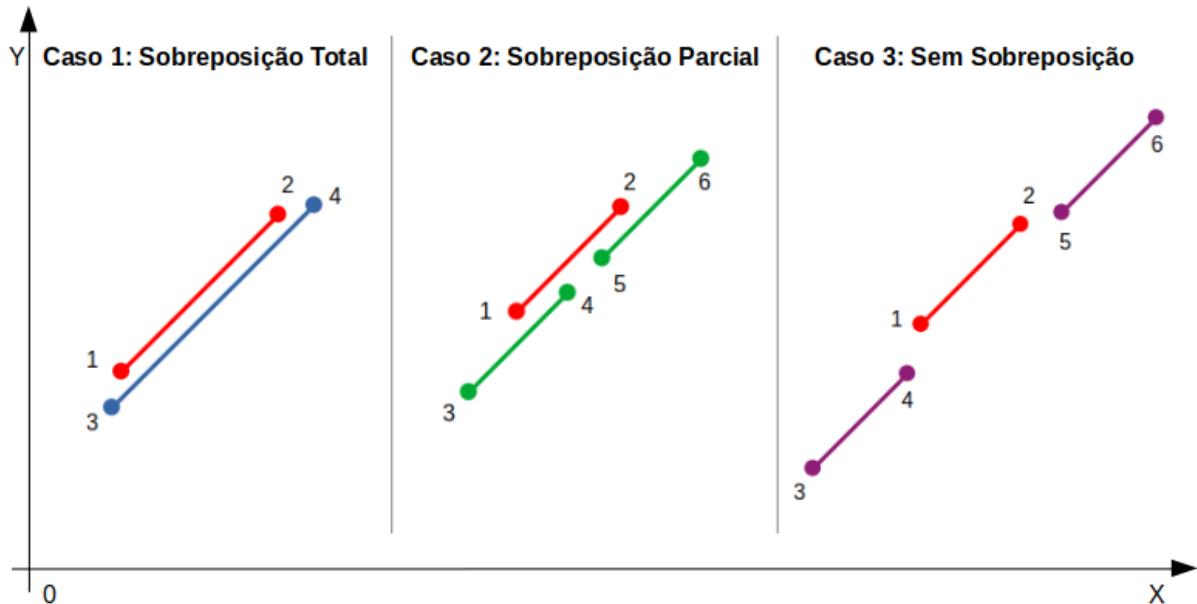


FIGURA 3.9 – Representação da junção de segmentos com os pontos terminais identificados por seus índices no plano cartesiano.

Dessa forma, os pontos terminais mais externos podem ser os de índices 2 e 3 ou 1 e 6, gerando o segmento resultante baseado na posição utilizada do segundo segmento.

Para o caso sem sobreposição, o primeiro segmento com índices 1 e 2 é distante das duas possibilidades do segundo segmento que são os segmentos de cor roxa com índices 3 e 4 ou 5 e 6. Com isso, não é possível realizar a junção dos segmentos evidenciados. Em todos os casos são verificados o ângulo de inclinação dos segmentos analisados e a distância entre cada ponto terminal de um segmento com o outro segmento.

Dessa forma, o algoritmo de SLAM com correspondência de características pode ser exemplificado com detalhes por meio do pseudocódigo desenvolvido no Apêndice A.

As seções seguintes descrevem o robô móvel utilizado e justificam as etapas de simulação e experimentos para o trabalho em questão.

3.2 Descrição do Robô Móvel

O robô móvel utilizado no trabalho foi o robô Ísis, que está presente no Laboratório de Máquinas Inteligentes (LMI) localizado no prédio do ITA. É um robô de base circular com duas rodas com motores diferenciais e duas rodas do tipo castor passivas, sua base é acoplada por uma torre de três andares, nos quais estão localizados os sensores, atuadores e demais componentes: o sensor kinect, a IMU, o *scanner a laser*, um *display LCD* e o microprocessador Raspberry Pi 3 B *plus*. Também contém um Arduino Mega em sua base, assim como os componentes de propulsão do robô (ponte H, placa de alimentação e

os motores CC) (PINTO, 2020).

O sistema móvel é capaz de se movimentar com bateria ou com alimentação externa, porém foi utilizada a alimentação externa nos experimentos realizados, em que o robô é alimentado por uma fonte de saída de 14,7 V. Além disso, o microcontrolador Arduino é responsável por enviar os comandos para acionamento dos motores e por ler os dados adquiridos pelos *encoders*. Já o microprocessador Raspberry Pi 3 B *plus* é responsável por receber os dados obtidos pela IMU e pelo *scanner a laser*, realizar a troca dessas informações com o Arduino e com o computador remoto e se comunicar via porta serial com o Arduino e via *Wi-Fi* com o computador remoto.

A Figura 3.10 mostra a estrutura real do robô Ísis e a Tabela 3.1 evidencia os parâmetros físicos do robô com suas medidas.

TABELA 3.1 – Parâmetros do Modelo Físico do Robô.

Parâmetro	Valor	Unidade
Massa do robô	3,24	kg
Largura do robô com as rodas	31,7	cm
Largura das rodas	4,2	cm
Diâmetro das rodas	10	cm
Diâmetro da base	23,3	cm
Altura entre o solo e a base do robô	6,47	cm
Altura total do robô	43	cm

3.2.1 Sensores do Robô

O robô utilizado no trabalho apresenta, como sensores, o *scanner a laser* (LiDAR), a bússola eletrônica presente na unidade de medida inercial (IMU), o sensor *kinect* e os *encoders* absolutos nas rodas. Desses sensores, apenas o *scanner a laser* e a IMU que foram usados no projeto.

O sensor LiDAR (*Light Detection And Ranging*) é um *scanner a laser* que varre 360° em torno de seu lugar, além do alcance com relação a um determinado obstáculo no ambiente. Esse tipo de sensor possui precisão, eficiência e menor gasto computacional com relação a outros existentes no mercado. No entanto, o *scanner* não funciona de forma adequada em alguns tipos de lugares. O *laser* atravessa, por exemplo, o vidro de forma ruidosa e o feixe do *laser* é interrompido e seu alcance diminui drasticamente em ambiente subaquático. Esse sensor é considerado de duas dimensões, pois ele consegue calcular a distância e o ângulo em relação a um obstáculo localizado no ambiente. O

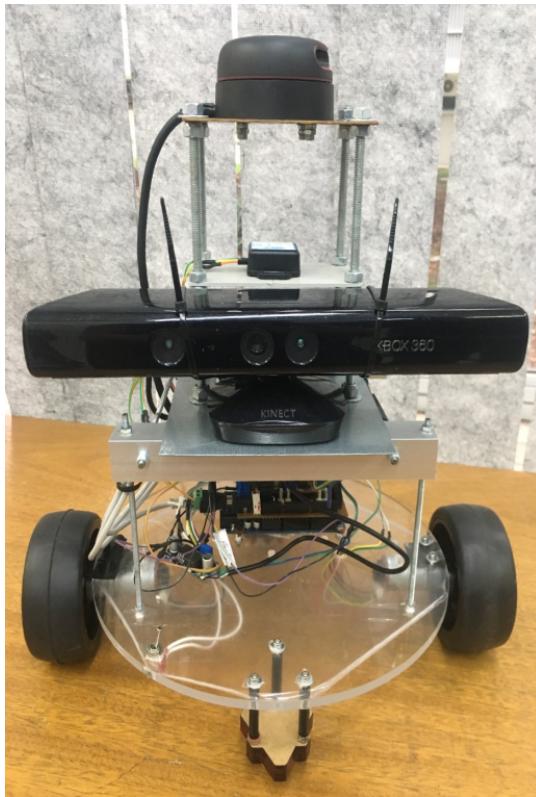


FIGURA 3.10 – Robô Ísis desenvolvido.

sensor presente no robô possui o modelo RPLiDAR A2M8 que pode ser observado na Figura 3.11 e suas especificações podem ser observadas na Tabela 3.2.

TABELA 3.2 – Especificações do sensor LiDAR modelo A2M8.

Especificação	Valor
Alcance	0,2 a 12 m
Frequência de amostragem	2 a 8 kHz
Resolução angular	0,9°
Frequência de varredura	5 a 15 rotações/s
Resolução da medida de distância	1 cm

A bússola eletrônica é um sensor componente de uma unidade de medida inercial (IMU) que possui a capacidade de medir o ângulo. O modelo da IMU utilizado foi o BWT901CL cujo fabricante é a empresa WiT Motion e está evidenciada na Figura 3.12. A IMU apresenta magnetômetro, girômetro e acelerômetro integrados e disponibiliza dois modos de operação, sendo que o primeiro modo é o de 9 eixos, pois integra os três sensores disponíveis e o segundo modo é o de 6 eixos, pois apresenta apenas o acelerômetro e girômetro.

Além disso, uma bússola eletrônica que utiliza magnetômetros pode apresentar me-



FIGURA 3.11 – Sensor *Scanner a Laser* RPLiDAR modelo A2M8 presente no robô (SHANGHAI SLAMTEC CO., LTD, 2017).

dições corrompidas por fontes de erro constantes e invariantes no tempo, pois o campo magnético sofre distorções com elementos ferromagnéticos localizados próximos do sensor. Diante disso, a calibração desse tipo de sensor é essencial para a determinação do ângulo com maior precisão (ALMEIDA, 2017).

Para esse trabalho foi empregado o modo de operação de 6 eixos, pois realiza a leitura do ângulo do eixo Z com base na integral da velocidade angular, porém contém erro calculado no ângulo do eixo Z. Já o modo de 9 eixos calcula o ângulo no eixo Z baseado no campo magnético e o ângulo possui poucos desvios, porém pode haver interferência do campo magnético em torno do ambiente instalado, sendo recomendável alternar para o modo do algoritmo de 6 eixos para a detecção do ângulo. Dessa forma, é necessário seguir os passos de configuração e calibração do sensor presentes no manual da IMU (WITMOTION SHENZHEN CO., LTD, 2020b). As especificações da IMU são mostradas na Tabela 3.3.

TABELA 3.3 – Especificações do sensor IMU modelo BWT901CL.

Especificação	Detalhe
Interface	Serial ou Bluetooth
Baudrate	115200
Tensão de alimentação	3,3 a 5 V
Corrente	< 40 mA
Frequência de saída	0,2 a 200 Hz

O *encoder* é um sensor capaz de converter um movimento mecânico angular ou linear em uma série de pulsos analógicos ou digitais elétricos. Esses pulsos são usados, por exemplo, para determinar velocidade, taxa de aceleração, distância, rotação, posição ou direção. No caso do robô móvel, os *encoders* estão localizados nos motores das rodas e



FIGURA 3.12 – Sensor IMU modelo BWT901CL presente no robô (WITMOTION SHENZHEN CO., LTD, 2020a).

são responsáveis por gerar dados de odometria a fim de verificar a posição aproximada do robô a partir das medidas de movimento das rodas.

Além disso, esse sensor é do tipo absoluto e apresenta um disco de marcações com código gravado, um componente emissor de luz e um foto receptor. Quando o disco gira, as marcações são contadas e o sinal enviado é proporcional ao número de pulsos contados por meio do movimento do disco, em que cada trilha do disco é nomeada de *bit*. A Figura 3.13 mostra o motor CC com o *encoder* acoplado a ele que foi implementado no robô e suas especificações podem ser observadas na Tabela 3.4.

TABELA 3.4 – Especificações do *encoder* integrado ao motor CC.

Especificação	Detalhe
Tipo	Absoluto
Velocidade	100 RPM
Tensão	6 V
Número de bits	11
Resolução	$11 \times \text{Redução } 74.83 = 823.1 \text{ PPR} \text{ (Pulsos Por Rotação)}$

3.3 Configuração Geral para Experimentos e Ambiente

O ambiente para simulação e experimentos é interno, com área conhecida inicialmente e formado por paredes finas e ortogonais ao chão de navegação. A proposta de solução foi realizada apenas para um robô. Paralelo a isso, o computador principal é responsável por executar os algoritmos relacionados ao SLAM e por realizar a comunicação sem fio



FIGURA 3.13 – Sensor *encoder* integrado ao motor CC localizado nas rodas do robô móvel (<https://www.baudaelectronica.com.br/motor-dc-6v-com-encoder-100-rpm.html>).

do robô móvel com o roteador Wi-Fi.

3.3.1 Configuração da rede ROS

Diante disso, a troca de dados entre computador e veículo é gerenciada pela rede ROS (ROS - Stanford Artificial Intelligence Laboratory et al., 2020) que pode ser ilustrada na Figura 3.14. Esse *firmware* é muito popular na área da robótica e pode ser utilizado para receber e enviar comandos tanto em robôs reais quanto em simulados, praticamente, com algoritmos semelhantes sem modificações complexas para obter êxito na execução. Nesse projeto, as versões ROS Kinetic e Noetic foram usadas. Tais versões são compatíveis entre si e dependem dos sistemas operacionais instalados no Raspberry Pi e no computador principal, nos quais elas atuam da forma mais apropriada (FILHO, 2021).

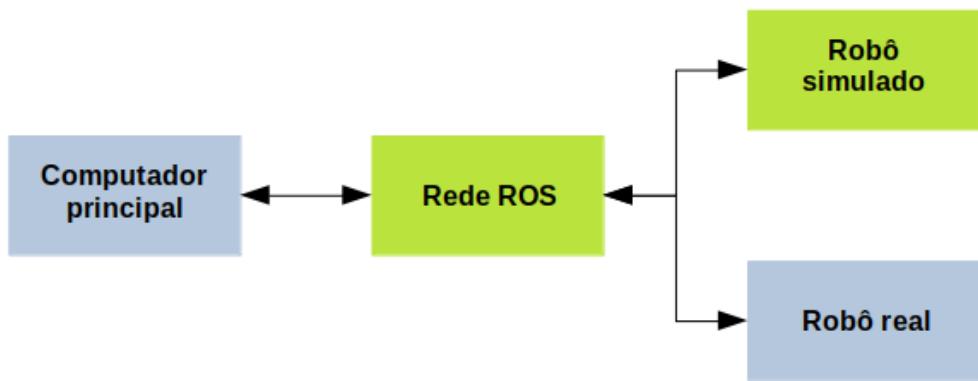


FIGURA 3.14 – Diagrama da comunicação entre *hardwares* que são os retângulos azuis e *softwares* que são os verdes (FILHO, 2021).

No computador principal, ocorre a conexão entre os códigos desenvolvidos no *software* MATLAB e a rede ROS, além da execução do algoritmo de SLAM. Esse *software* foi desenvolvido para oferecer ferramentas da área de engenharia a fim de construir um protótipo rapidamente, assim como desenvolver um produto completo (MATHWORKS, 2022a). Além

disso, a comunicação da rede ROS envia e recebe dados a partir de tópicos que são canais para transporte de informações entre nós. Tais dados podem ser comandos de velocidade, pontos do *scanner a laser*, ângulo da IMU, dentre outros, e essas informações são enviadas dentro de mensagens para que a rede ROS reconheça. As mensagens enviadas por um nó dentro de um tópico devem possuir o mesmo tipo das mensagens recebidas por outro nó do mesmo tópico para que a rede funcione de forma pertinente. A Figura 3.15 ilustra o processo da rede ROS descrito previamente.

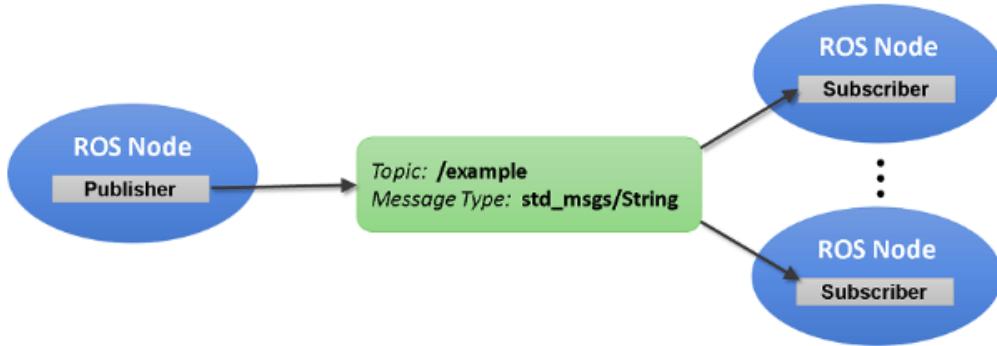


FIGURA 3.15 – Ilustração de um diagrama de nós cuja mensagem é enviada e recebida entre os nós através de um único tópico (MATHWORKS, 2022b).

A Figura 3.15 evidenciou o mecanismo de transporte de dados pela rede ROS, no qual existe um nó que envia (*publisher*) informações em forma de mensagem do tipo **String** por meio do tópico chamado de **/example**. Esse tópico transporta a mensagem para os nós *subscribers* que recebem a mensagem. Dessa forma, o sentido da seta presente na imagem auxilia no estabelecimento da direção do transporte dos dados.

3.3.2 Arquitetura da rede de comunicação do robô

Como solução proposta para o trabalho, todo o processamento de dados e realização das operações são feitos por meio do computador principal que é o mestre da rede ROS e é o responsável pela execução do *software* MATLAB. Paralelo a isso, o computador se comunica via Wi-Fi com o microprocessador Raspberry Pi 3 B *plus*, o qual se conecta via USB com o microcontrolador Arduino. A Figura 3.16 ilustra a arquitetura de *hardware* dos sensores, atuadores, placas embarcadas no robô e computador utilizados.

Por meio disso, o Raspberry Pi 3 B *plus* localizado no robô é a unidade principal do veículo que troca informações através da rede ROS com o computador mestre. Assim, as leituras da IMU e do *scanner a laser* (LiDAR) são realizadas pelo microprocessador em questão a partir dos pacotes *rplidar* e *isis*.

No entanto, o microcontrolador Arduino envia dados dos *encoders* das rodas e recebe comandos de velocidade para os motores por meio do pacote do ROS denominado *rosse-*

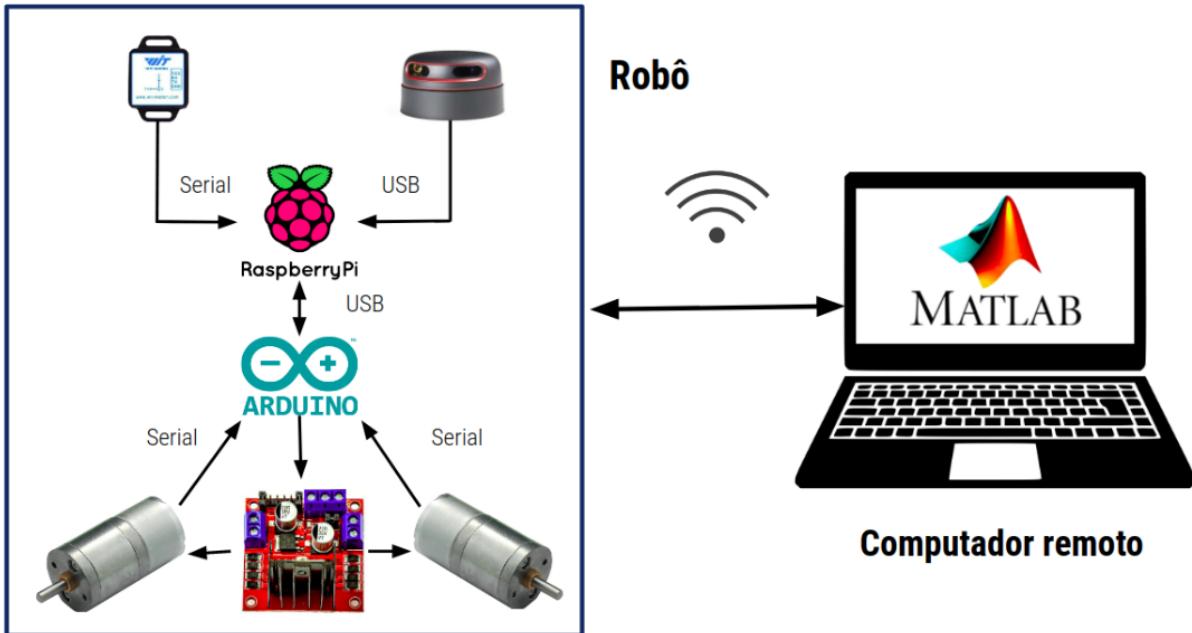


FIGURA 3.16 – Arquitetura de *hardware* dos dispositivos utilizados (PINTO, 2020).

rial_arduino, interligando o Arduino ao Raspberry Pi via USB. Já o nó que troca dados pela rede ROS no computador é o *matlab_global_node* criado pelo software MATLAB para se comunicar com o robô.

Em suma, os pacotes abordados no desenvolvimento da solução podem ser observados da seguinte forma.

- *isis/isis_gazebo* - O pacote *isis* apresenta um código desenvolvido na linguagem *python* que faz a leitura do sensor IMU e gera o ângulo no eixo z. Tal informação é enviada pelo nó *pub_yaw* através do tópico */yaw*. Já o pacote *isis_gazebo*, apresenta todos os arquivos e configurações do robô, dos sensores e do ambiente simulados no Gazebo.
- *rplidar* - Esse pacote permite a integração do *scanner a laser* com a rede ROS, sendo um pacote *open source* disponibilizado pela empresa SLAMTEC. Por meio disso, o pacote envia dados obtidos no *scanner* pelo tópico */scan* e o tipo de mensagem é *sensor_msgs*. Além disso, esse pacote apresenta alguns parâmetros que podem ser definidos para a operação do sensor e os serviços *start_motor* e *stop_motor* são usados para iniciar e parar, respectivamente, o funcionamento do motor de rotação do *scanner a laser*.
- *rosserial_arduino* - Tal pacote permite a implantação do sistema ROS em algoritmos desenvolvidos na IDE (*Integrated Development Environment*) do Arduino de forma direta. Fornece um protocolo de comunicação no UART (*Universal Asynchronous Receiver/Transmitter*) do Arduino. A partir disso, o Arduino se torna um nó ROS

completo que envia (*publisher*) e recebe (*subscriber*) mensagens ROS, que publica transformações (TF) e registra a hora do sistema ROS. Assim, o pacote, no caso do trabalho, constroi o nó *serial_node* que envia dados dos *encoders* pelo tópico */xy* e recebe comandos de velocidade pelo tópico */cmd_vel* que acionam os motores do robô.

Além disso, o MATLAB possui a ROS *Toolbox* que permite conectar tal *software* ao *firmware* ROS, criando ou se conectando à rede ROS. Dessa forma, o computador com o MATLAB instalado é capaz de receber informações dos tópicos */scan*, */xy* e */yaw* que são originárias do *scanner a laser*, dos *encoders* e da IMU, respectivamente. O nó *matlab_global_node* proveniente do MATLAB também pode enviar dados de velocidade para movimentar os motores do robô através do tópico */cmd_vel*. Todo o processo exemplificado anteriormente pode ser ilustrado na Figura 3.17.

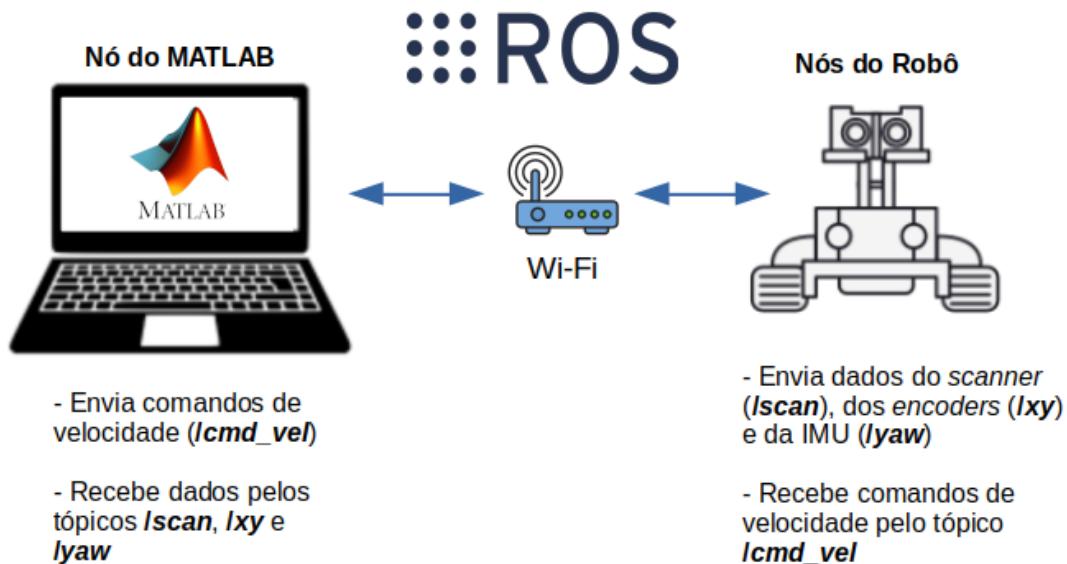


FIGURA 3.17 – Configuração da rede ROS desenvolvida.

A arquitetura de *software* original e criada por Pinto (2020) pode ser visualizada por meio do diagrama de nós e tópicos da Figura 3.18, em que os nós são representados por elipses e os tópicos por retângulos. As setas representam o sentido do transporte das informações ou mensagens entre os nós por meio do canal de comunicação ou tópico.

Todos os *softwares* desenvolvidos, assim como outros arquivos utilizados neste trabalho podem ser acessados em: https://github.com/Intelligent-Machines-Lab/Line-Matching_SLAM.

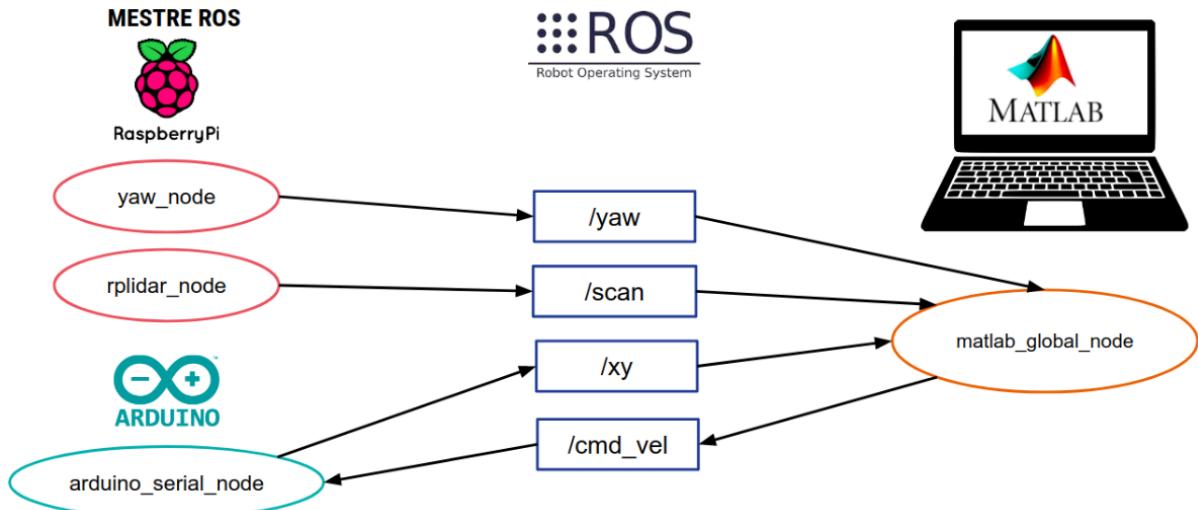


FIGURA 3.18 – Diagrama original de nós e tópicos utilizados pelo robô móvel na rede ROS (PINTO, 2020).

3.3.3 Simulação do robô móvel e do ambiente

Para se desenvolver as simulações do trabalho, utilizou-se o *software* Gazebo que é capaz de construir modelos de simulação em 3D com diversos robôs, sendo uma plataforma de código aberto e compatível com o ROS (GAZEBO, 2022). Essa plataforma pode simular modelos físicos com suas fórmulas, o que torna a simulação precisa e confiável com o modelo real. Dessa forma, sensores e atuadores para diversas finalidades podem ser simulados no Gazebo, como um *scanner a laser*, câmeras, dentre outros.

Um sistema robótico no Gazebo pode ser desenvolvido por meio da modelagem URDF (*Unified Robot Description Format*). Com isso, o modelo apresenta a cinemática e dinâmica detalhadas do robô, dos sensores e atuadores embarcados nele. Além disso, esse sistema também é capaz de obter uma apresentação visual 3D de um modelo de colisão do robô e do ambiente simulado. Assim, a Figura 3.19 ilustra um modelo robótico simulado construído no Gazebo que se refere ao robô Ísis.

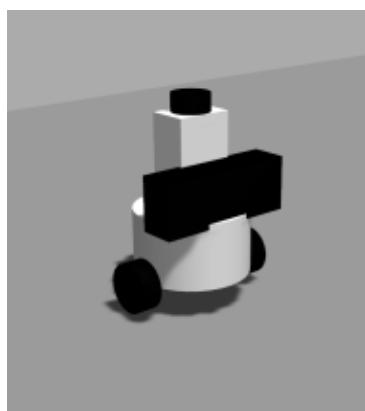


FIGURA 3.19 – Modelo do robô desenvolvido no Gazebo.

Para se construir um modelo de ambiente interno no *software Gazebo*, utiliza-se o editor que possibilita projetar paredes de forma simples com a visualização de perspectiva de cima para baixo, além da inclusão de objetos de formas comuns em 3D. Dessa forma, foi desenvolvido um ambiente interno de aproximadamente 44 metros quadrados (14,6 m x 3 m) apenas com paredes ortogonais ao chão, conforme pode ser observado na Figura 3.20, na qual (a) possui uma parede a menos que o ambiente em (b).

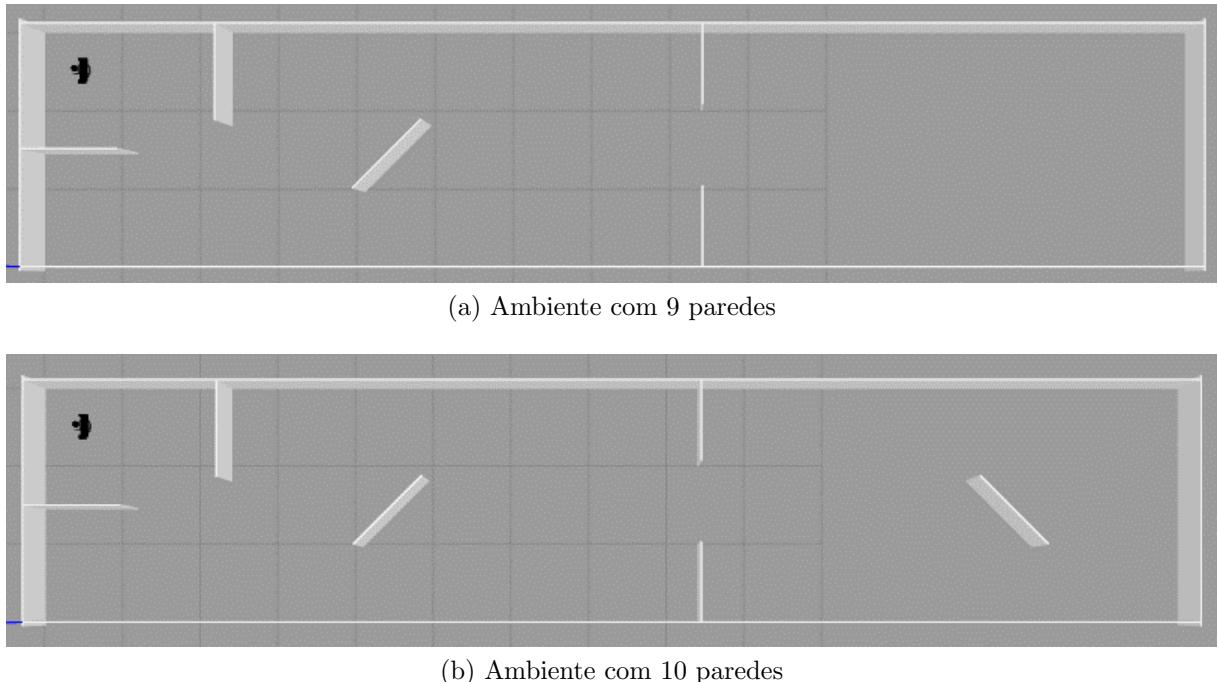


FIGURA 3.20 – Modelo de um ambiente configurado no Gazebo, em que (a) possui nove paredes e (b) apresenta dez paredes.

3.3.4 Ambiente do Experimento Real

Os experimentos foram realizados no corredor do LMI (Laboratório de Máquinas Inteligentes) localizado no prédio do ITA. O ambiente elaborado no corredor se baseou no simulado pelo Gazebo. Além disso, paredes de madeira foram incluídas para simular algumas paredes do ambiente e a altura delas influenciou na altura dos sensores projetados no robô móvel. Contudo, pequenas brechas entre algumas paredes e saliências ou desníveis no piso podem proporcionar erros de medição do *scanner a laser* e, consequentemente, dados errôneos são formados. Diante disso, a Figura 3.21 mostra uma foto do ambiente criado.



FIGURA 3.21 – Ambiente construído no corredor do LMI.

4 Experimentos e Resultados Obtidos

Este capítulo retrata os resultados dos experimentos simulados e real para avaliação da proposta de solução implementada no *software* MATLAB versão R2021b. Os experimentos foram considerados determinísticos e foram implementados em ambiente desenvolvido no *software* Gazebo e no ambiente real com o robô móvel.

Além disso, para a solução dos problemas de minimização da correspondência entre linhas de *scans* nas posturas consecutivas e do fechamento de laço, utilizou-se a função “fminsearch” do MATLAB que soluciona problemas de minimização com função objetivo não linear sem restrições (MATHWORKS, 2022c). A otimização presente na função do MATLAB é baseada no algoritmo de Nelder-Mead que é um método de busca direta para minimização irrestrita multidimensional (LAGARIAS *et al.*, 1998).

4.1 Estrutura da Simulação no Gazebo

A simulação do ambiente no *software* Gazebo mostrada na Figura 4.1 apresentou o sistema de coordenadas inicial e respectiva origem (0 m, 0 m) por meio das setas vermelhas, exemplificando os eixos das abscissas (X) e das ordenadas (Y) para o ambiente configurado de nove paredes em (a) e dez paredes em (b).

Com o sistema de coordenadas inicial definido pelas setas vermelhas na Figura 4.1, a postura inicial do robô móvel simulado e localizado no ambiente pode ser evidenciada na Tabela 4.1 e foi definida de forma arbitrária nos dois experimentos simulados no *software* Gazebo.

TABELA 4.1 – Postura inicial do robô no sistema de coordenadas do Gazebo.

Coordenada X (m)	Coordenada Y (m)	Orientação (graus)
0,5	2,5	0

Além disso, o comprimento e as coordenadas (x,y) do ponto médio das paredes do ambiente podem ser observados na Tabela 4.2.

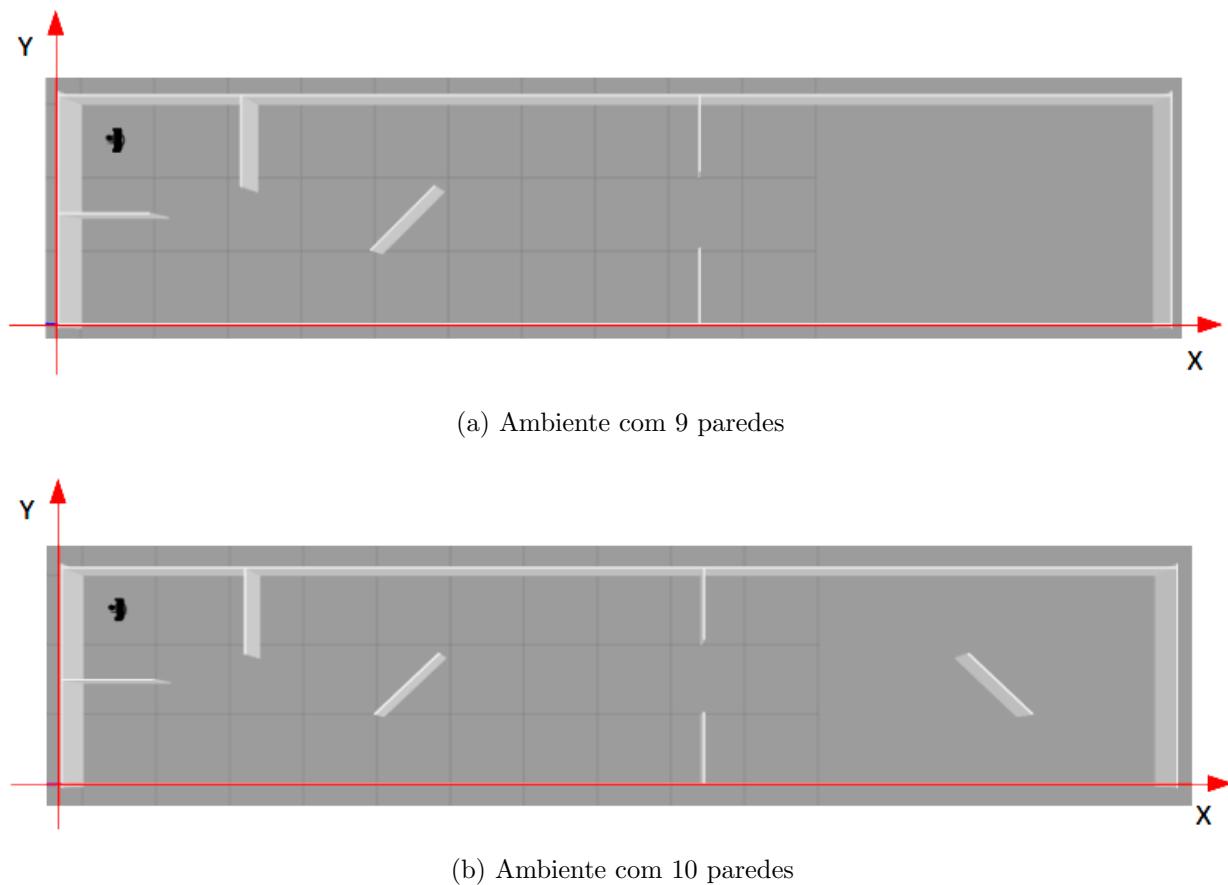


FIGURA 4.1 – Sistema de coordenadas no *software* Gazebo com vista de cima para baixo do ambiente, em que (a) é o ambiente de nove paredes e (b) é o ambiente de dez paredes.

TABELA 4.2 – Comprimento e coordenada do ponto médio das paredes do ambiente simulado.

Parede	Comprimento (m)	Ponto Médio (x,y) (m)
1	14,6	(7,3 0)
2	14,6	(7,3 3)
3	3,1	(0 1,5)
4	1	(8,41 2,5)
5	1,2	(0,6 1,45)
6	1,2	(2,4 2,4)
7	1,2	(4,52 1,39)
8	1	(8,41 0,5)
9	3,1	(14,6 1,5)
10	1,2	(12,285 1,39)

A partir do ambiente configurado no simulador, a configuração do sensor *scanner a laser* (LiDAR) simulado foi definida da seguinte forma.

- Alcance mínimo: 5 cm.
- Alcance máximo: 10 m.
- Ângulo mínimo: 0° .
- Ângulo máximo: 360° .
- Incremento do ângulo: $\approx 1,2^\circ$.
- Número de leituras do sensor: ≈ 299 pontos.

4.2 Resultado dos Experimentos Simulados

Foram simulados dois experimentos, sendo um no ambiente com nove paredes e o outro no ambiente com dez paredes. A parede a mais incluída no segundo experimento foi utilizada para evidenciar que existem regiões comuns essenciais que o robô não consegue captar com o sensor *scanner a laser* e que podem prejudicar o mapa global gerado sem o fechamento de laço. No primeiro experimento, foram obtidas 45 posturas e não foi usada a etapa de fechamento de laço. No segundo experimento, foram extraídas 38 posturas e foram mostrados os resultados referentes ao caso sem o fechamento de laço e ao caso com a técnica de fechamento de laço. Em ambos os experimentos foi feita a etapa de junção de segmentos proposta.

4.2.1 Primeiro Experimento

O primeiro experimento se iniciou com a exploração manual do robô no ambiente simulado com nove paredes no Gazebo, em que sua postura inicial apresentou as coordenadas x igual a 0,5 metros, y igual a 2,5 m e θ igual a 0 graus que foi definida de forma arbitrária. Dessa forma, foram coletadas 45 posturas ao longo do processo e o *scanner a laser* realizou a leitura da nuvem de pontos do ambiente para cada postura adquirida.

4.2.1.1 Sem Fechamento de Laço

Após a coleta desses dados simulados, cada *scan* de pontos foi utilizado no algoritmo de RANSAC para serem obtidos os segmentos de reta referentes às paredes do ambiente. Cada segmento de reta foi adicionado à uma matriz de segmentos para cada *scan*, em que

tais linhas eram delimitadas pelos pontos terminais (*endpoints*). Com isso, por exemplo, caso um *scan* de pontos tenha 5 segmentos de reta detectados, a matriz de segmentos apresentaria 2 linhas e 10 colunas, pois cada linha representa cada um dos elementos do par de coordenadas (x,y) dos pontos terminais e cada coluna evidencia um ponto terminal e, assim, cada segmento é composto por dois pontos terminais, ou seja, 10 colunas, conforme Tabela 4.3. A Figura 4.2 mostra um exemplo de *scan* de pontos em uma determinada postura após o algoritmo de RANSAC com seus segmentos extraídos e a Tabela 4.3 mostra as coordenadas de cada *endpoint* desses segmentos do exemplo citado.

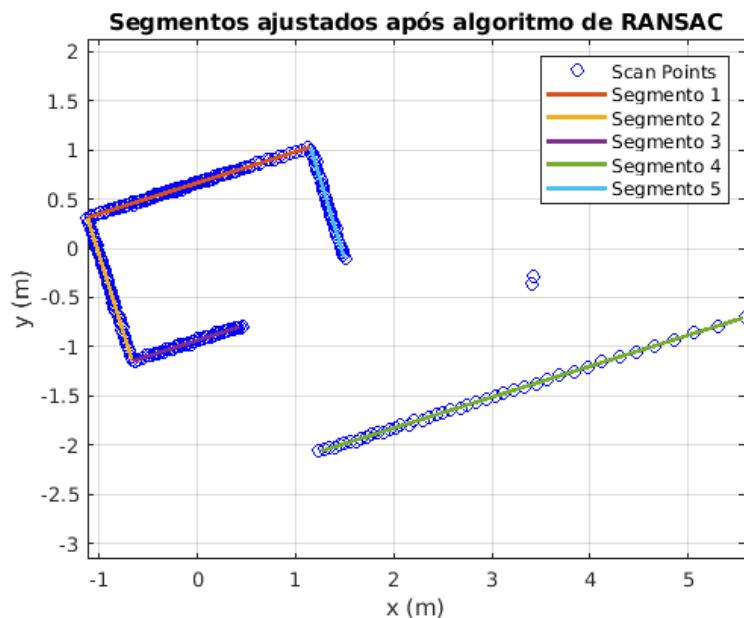


FIGURA 4.2 – Exemplo de segmentos adquiridos com o algoritmo RANSAC adaptado extraídos de um *scan* de pontos numa determinada postura.

TABELA 4.3 – Coordenadas (x,y) em metros dos pontos terminais dos segmentos da Figura 4.2.

	Segmento 1		Segmento 2		Segmento 3		Segmento 4		Segmento 5	
x	1,15	-1,12	-1,12	-0,66	0,43	-0,66	5,56	1,26	1,15	1,50
y	1,03	0,31	0,31	-1,15	-0,80	-1,15	-0,71	-2,06	1,03	-0,09

Ao final da execução do algoritmo de RANSAC adaptado que foi descrito no capítulo anterior, obteve-se todos os segmentos gerados a partir da leitura dos pontos do *scan* nas 45 posturas medidas, conforme pode ser visualizado nas Figuras 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9 e 4.10. Em tais imagens, pode-se notar que nem todos os pontos do *scan* foram associados aos segmentos gerados, pois foi definido um limite para a quantidade de pontos que seriam avaliados no algoritmo de RANSAC.

Além disso, pontos muito distantes de um conjunto acumulado de pontos pertencentes a mesma linha eram considerados *outliers*, devido à possibilidade de existirem duas ou

mais paredes alinhadas que poderiam ser consideradas apenas uma por serem do mesmo segmento de reta. Por isso, alguns pontos não foram utilizados na geração das linhas.

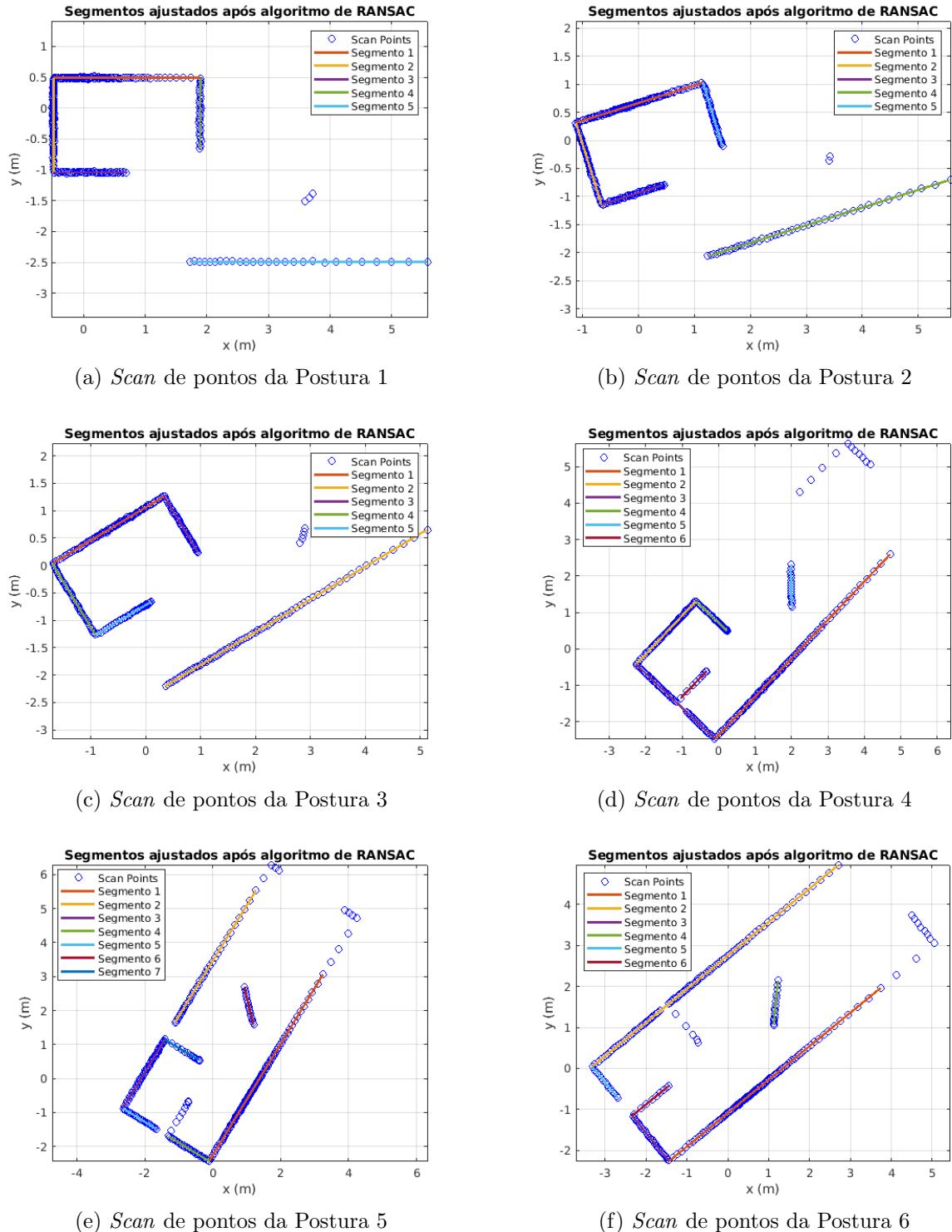


FIGURA 4.3 – Scans de pontos gerados nas Posturas 1 a 6 do primeiro experimento.

Com os segmentos de reta gerados para cada *scan* de pontos para as 45 posturas, executou-se o algoritmo de SLAM com correspondência de características, o qual associou os segmentos de posturas consecutivas para encontrar a melhor solução de rotação e

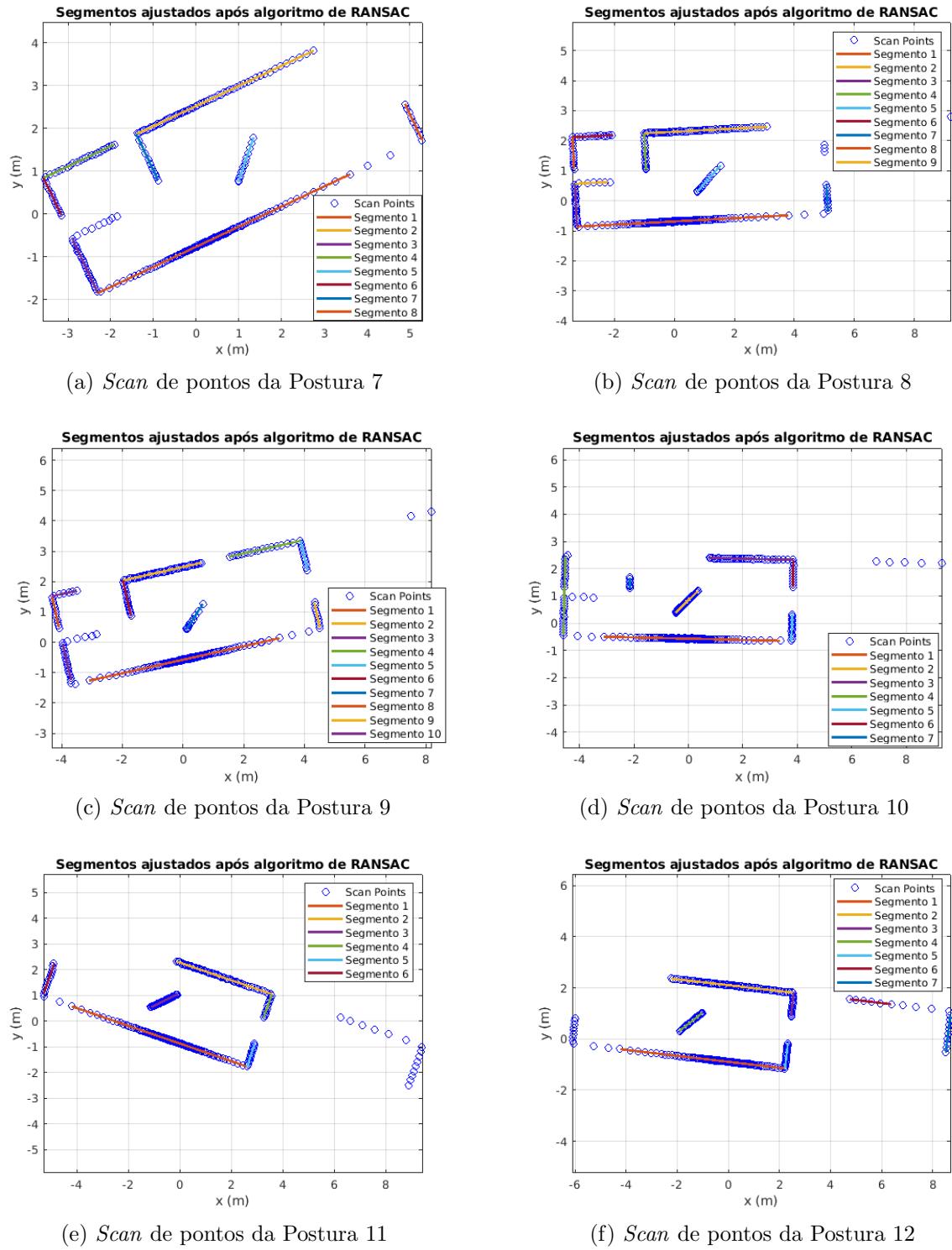


FIGURA 4.4 – *Scans* de pontos gerados nas Posturas 7 a 12 do primeiro experimento.

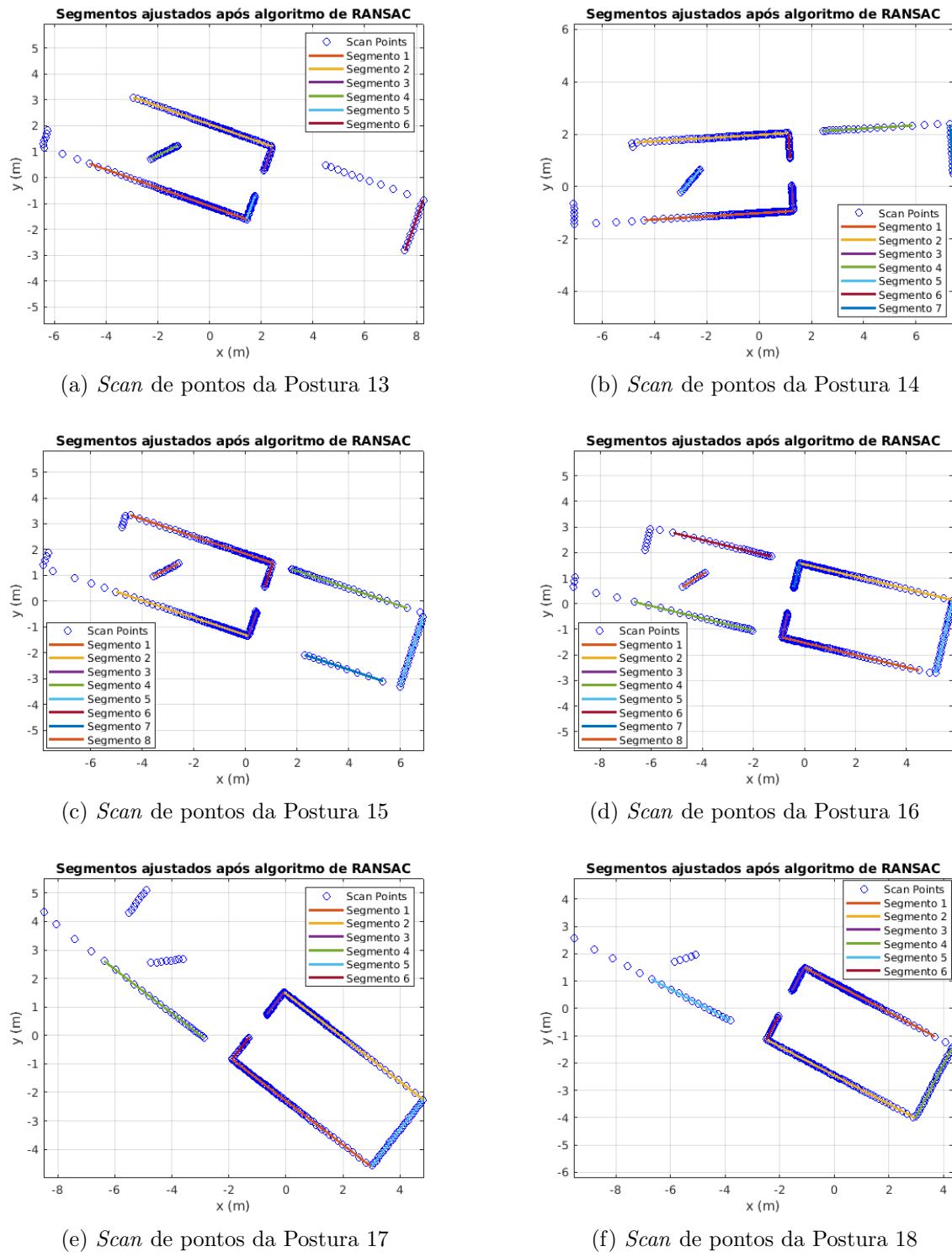


FIGURA 4.5 – *Scans* de pontos gerados nas Posturas 13 a 18 do primeiro experimento.

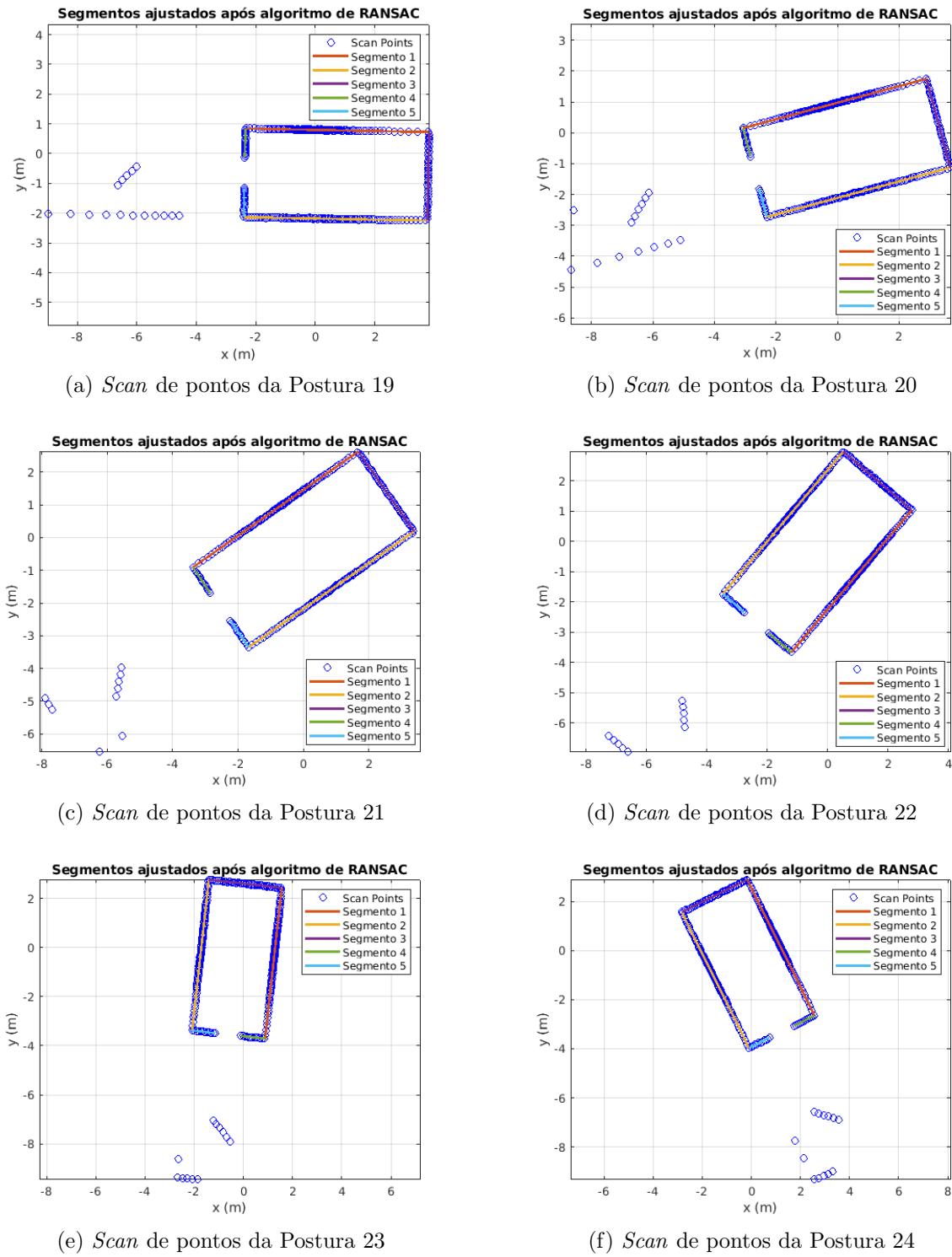


FIGURA 4.6 – Scans de pontos gerados nas Posturas 19 a 24 do primeiro experimento.

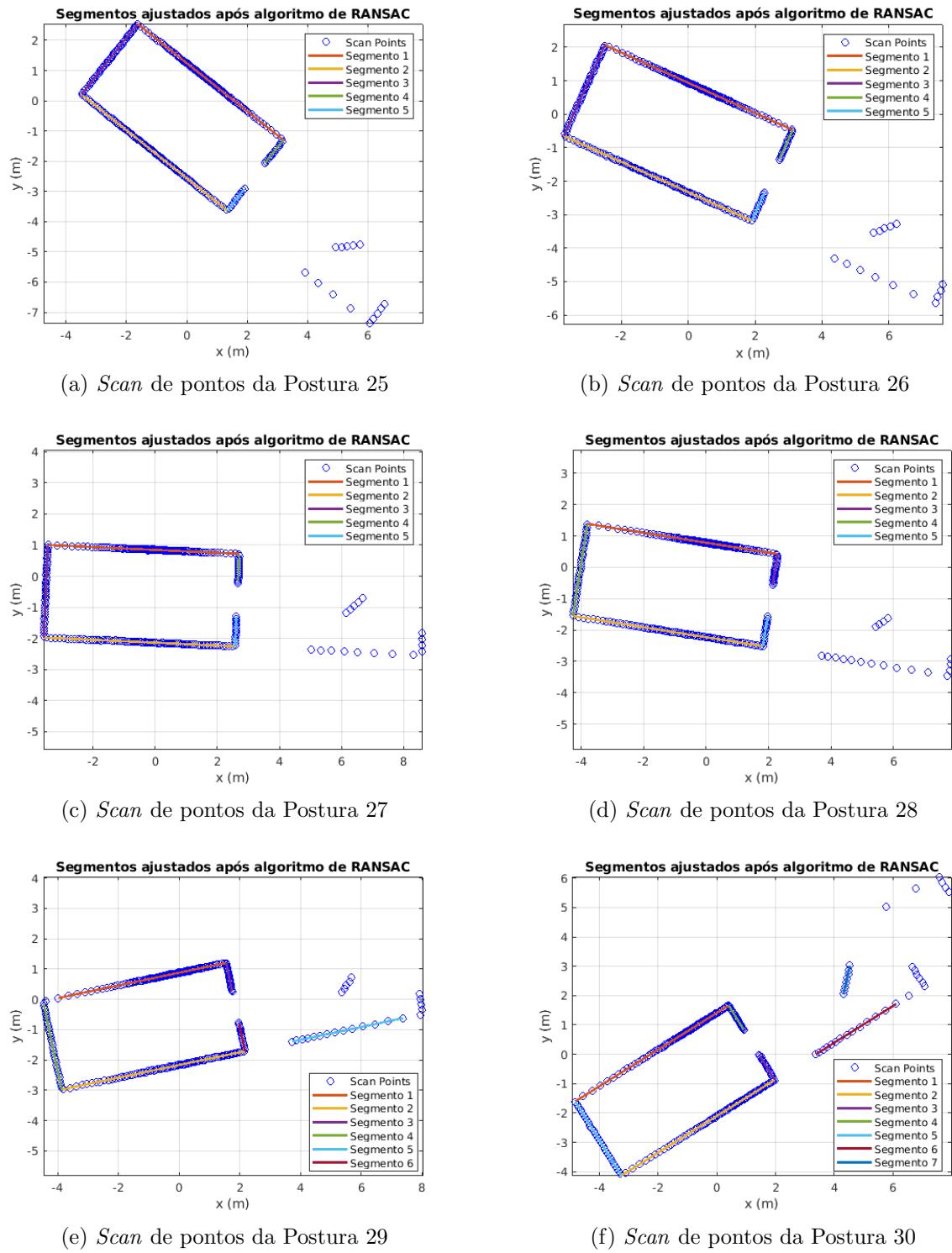


FIGURA 4.7 – *Scans* de pontos gerados nas Posturas 25 a 30 do primeiro experimento.

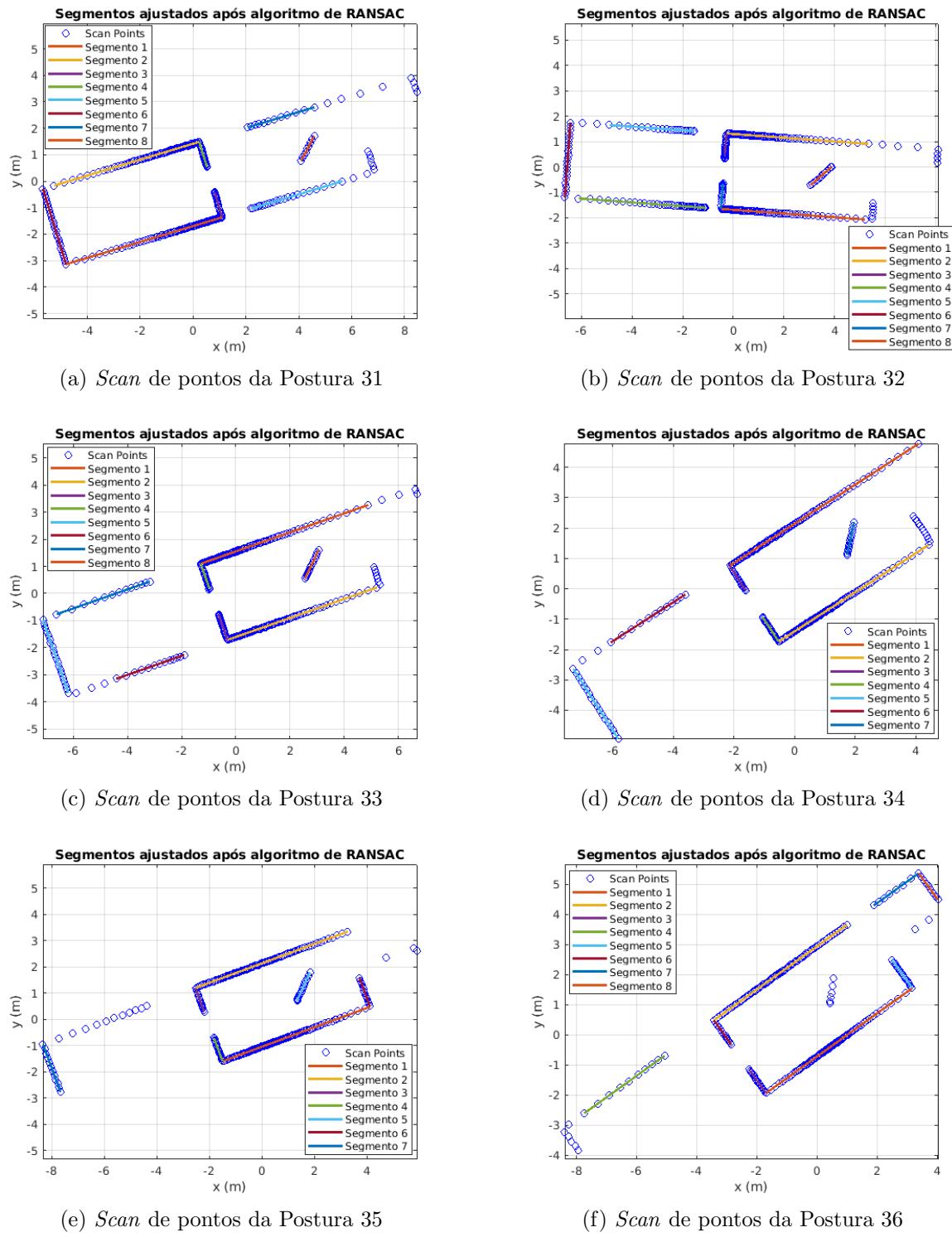
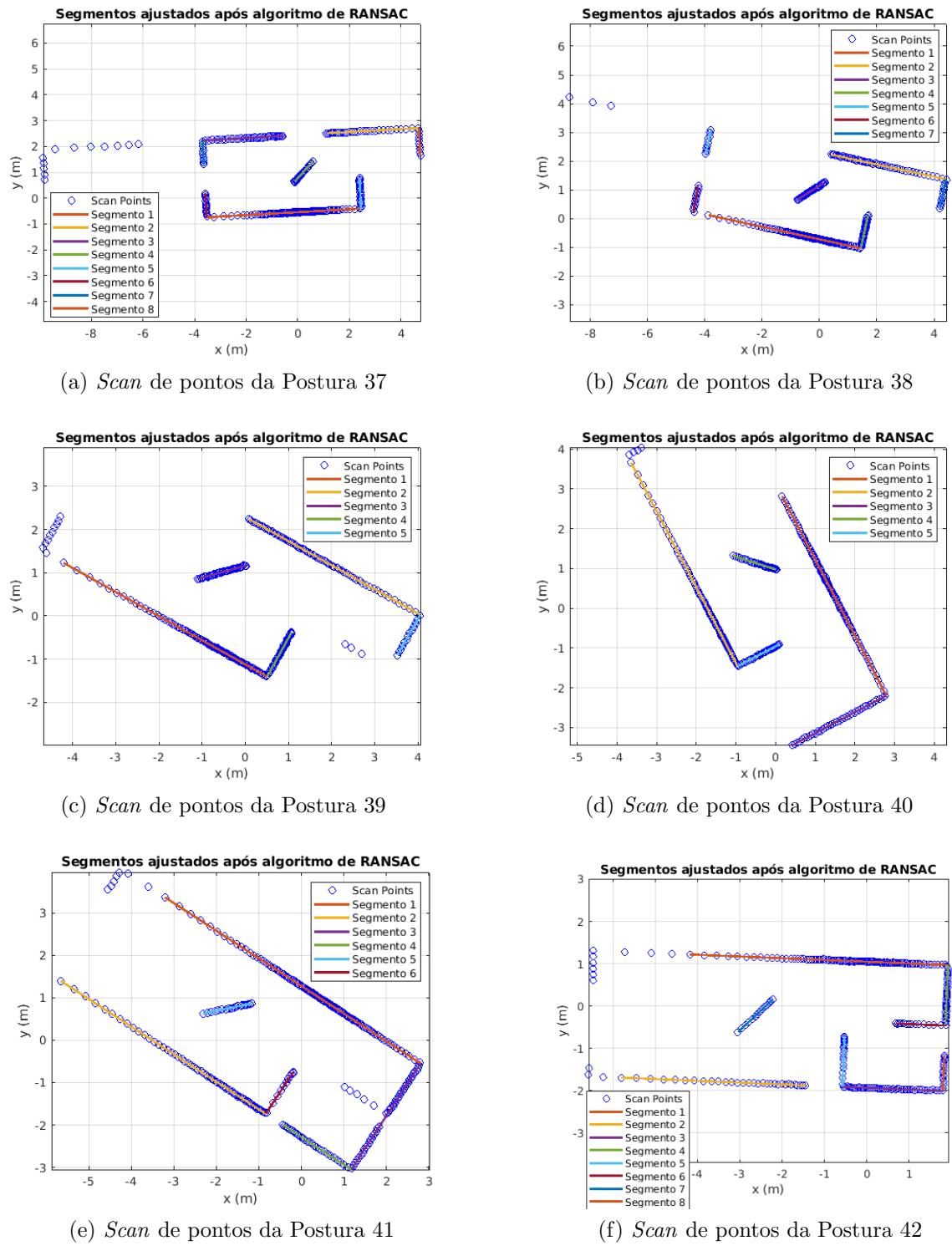


FIGURA 4.8 – *Scans* de pontos gerados nas Posturas 31 a 36 do primeiro experimento.

FIGURA 4.9 – *Scans* de pontos gerados nas Posturas 37 a 42 do primeiro experimento.

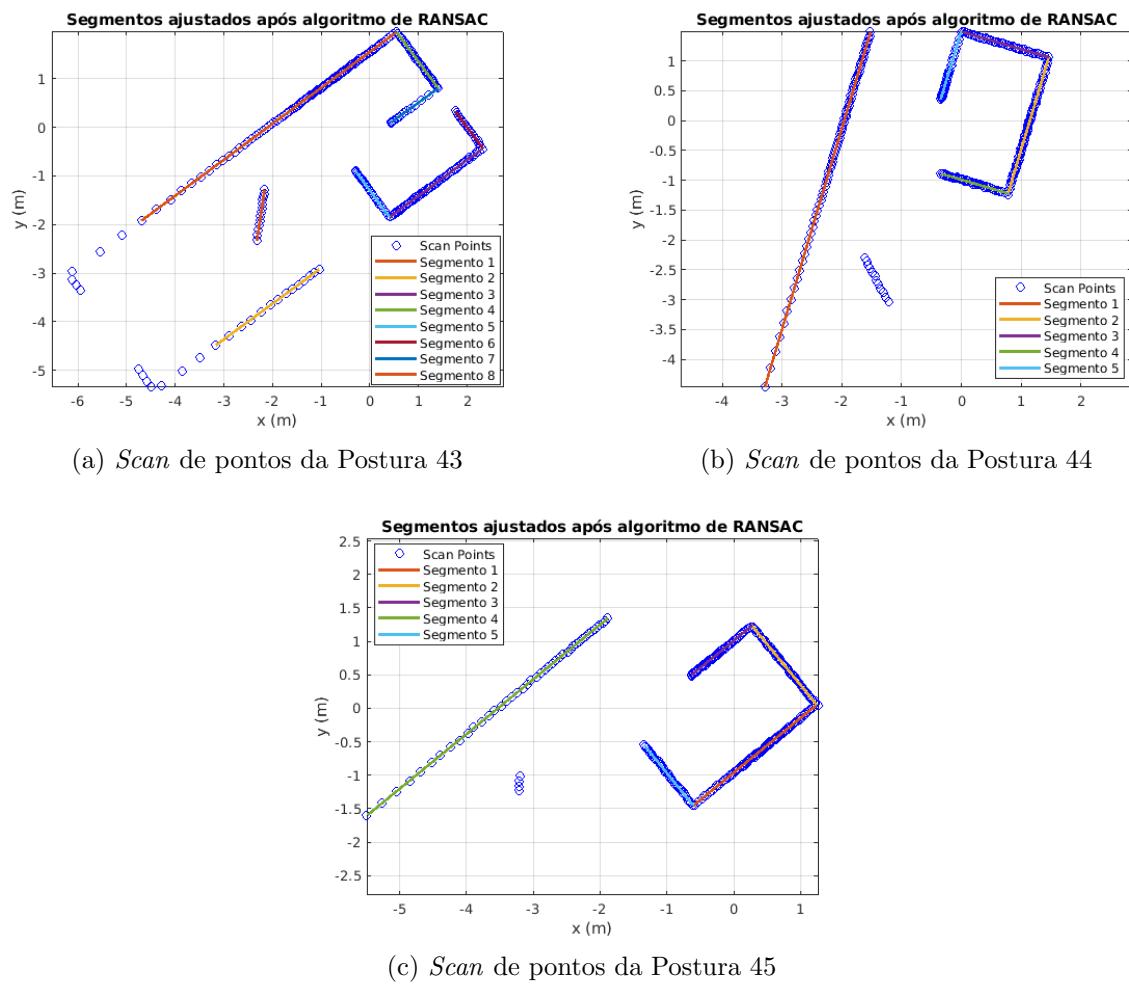


FIGURA 4.10 – Scans de pontos gerados nas Posturas 43 a 45 do primeiro experimento.

translação a fim de gerar a correspondência (*matching*) adequada entre os segmentos de cada postura.

Dessa maneira, a Tabela 4.4 mostra a rotação e a translação mais apropriadas que foram obtidas pelo algoritmo de SLAM para assegurar uma combinação razoável entre os segmentos de posturas consecutivas.

TABELA 4.4 – Rotação em graus e translação em metros obtidas entre *scans* em posturas consecutivas do primeiro experimento.

Scans	Rotação(graus)	Translação(m)	Scans	Rotação(graus)	Translação(m)
1/2	-17,358	0,506	23/24	-32,688	0,168
2/3	-13,591	0,543	24/25	-25,114	0,258
3/4	-15,595	0,611	25/26	-14,221	0,296
4/5	-12,072	0,525	26/27	-21,640	0,369
5/6	19,597	0,528	27/28	6,425	0,484
6/7	14,120	0,366	28/29	-20,855	0,428
7/8	21,530	0,403	29/30	-20,111	0,647
8/9	-9,608	0,609	30/31	15,650	0,610
9/10	13,755	0,702	31/32	20,569	1,025
10/11	17,440	0,789	32/33	-23,131	0,514
11/12	-12,839	0,720	33/34	-13,527	0,599
12/13	12,594	0,477	34/35	12,548	0,618
13/14	-22,952	0,647	35/36	-15,342	0,665
14/15	21,963	0,751	36/37	32,402	1,057
15/16	-5,772	1,125	37/38	15,328	0,812
16/17	23,961	0,543	38/39	16,817	0,553
17/18	-9,790	0,788	39/40	33,074	0,642
18/19	-26,470	0,682	40/41	-29,161	0,714
19/20	-16,110	0,587	41/42	-30,114	0,748
20/21	-19,895	0,445	42/43	-38,916	0,341
21/22	-14,707	0,432	43/44	-37,138	0,652
22/23	-33,974	0,173	44/45	34,231	0,566

Além disso, depois da geração da rotação e translação dos segmentos de cada postura consecutiva, verificou-se também os índices dos segmentos da postura atual que se combi-

navam com as linhas da postura seguinte, além dos índices dos segmentos que não combinavam entre si. Por meio disso, foram geradas as Figuras 4.11, 4.12, 4.13, 4.14, 4.15, 4.16, 4.17 e 4.18 que representam o *matching* entre linhas dos *scans* em posturas consecutivas.

Essas imagens destacaram os segmentos de linha da postura atual como linhas pretas e seus pontos terminais como asteriscos pretos, enquanto que os segmentos da postura consecutiva são representados por linhas tracejadas vermelhas e seus pontos terminais por losangos vermelhos, cada segmento da postura atual é representado por seu índice na cor preta, enquanto que os segmentos da postura consecutiva são evidenciados pelos índices da cor vermelha.

Diante do evidenciado anteriormente, pode-se verificar que existiram alguns segmentos que foram detectados na postura atual, mas não foram obtidos na postura consecutiva e vice-versa. Também, se observou que o comprimento entre alguns segmentos combinados foi diferente, devido ao comprimento original da parede do ambiente ser maior que o alcance máximo do *scanner a laser* que foi configurado para 10 metros.

Mediante isso, pode-se calcular as posturas estimadas por meio dos dados de cada rotação e translação gerados. Tais posturas foram representadas em relação ao sistema de coordenadas da postura inicial local, considerada (0 m, 0 m, 0°). Fez-se a comparação da posição X e Y em metros e da orientação θ em graus entre as posturas reais e estimadas nas Figuras 4.19, 4.20 e 4.21, respectivamente, nas quais a linha azul corresponde às posturas reais e a linha vermelha se refere às posturas estimadas.

Paralelo a isso, a Tabela 4.5 mostra o erro adquirido entre cada postura real do robô obtida na simulação e cada postura estimada no algoritmo de SLAM com correspondência de características, assim como a Figura 4.22 evidencia graficamente o erro entre as posturas, em que a linha azul contínua é correspondente ao erro da posição X em metros, a linha azul tracejada é referente ao erro da posição Y em metros e a linha vermelha contínua representa o erro do ângulo de rotação em graus.

Em decorrência do erro gerado entre as posturas reais e estimadas, notou-se através da Figura 4.22 que o maior erro existente para a posição X foi de -8 cm na postura 39, para a posição Y foi de 22,6 cm na postura 23 e 2,43° na postura 17 para a rotação θ .

Ao final do processo de SLAM por grafo de posturas com a utilização de *line matching* como correspondência de características descrito pela proposta de solução, obteve-se a Figura 4.23 que representa a trajetória do robô no mapa global gerado e baseado no ambiente simulado. As linhas pretas são as paredes do ambiente, os quadrados azuis são as posturas estimadas, as linhas finas pretas ilustram a trajetória feita pelo robô, o quadrado verde é a postura inicial, o quadrado azul ciano é a postura final e a linha vermelha em cada quadrado é a orientação da postura estimada.

Dessa forma, observou-se que o mapa global obtido é aproximado ao ambiente simulado

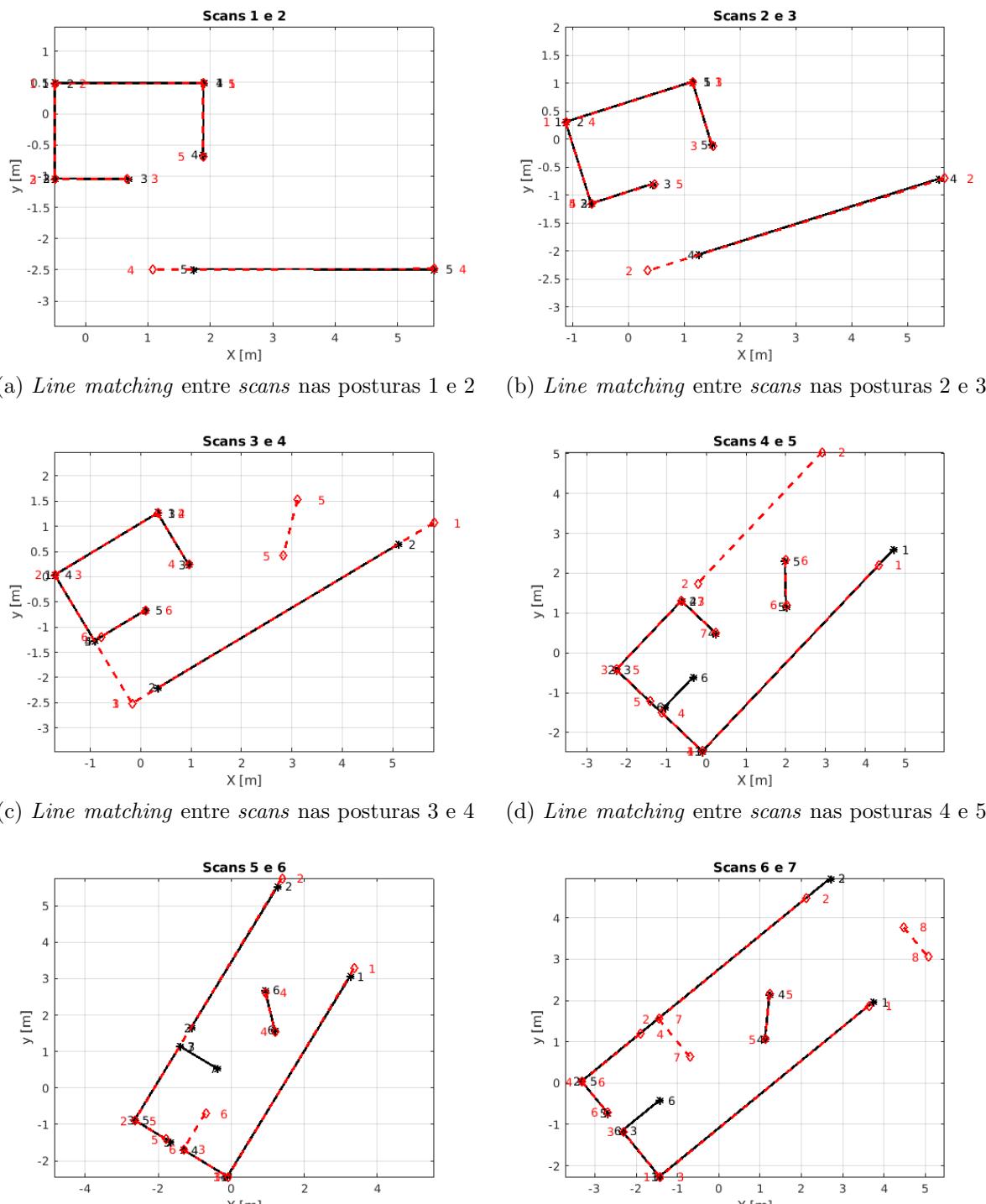
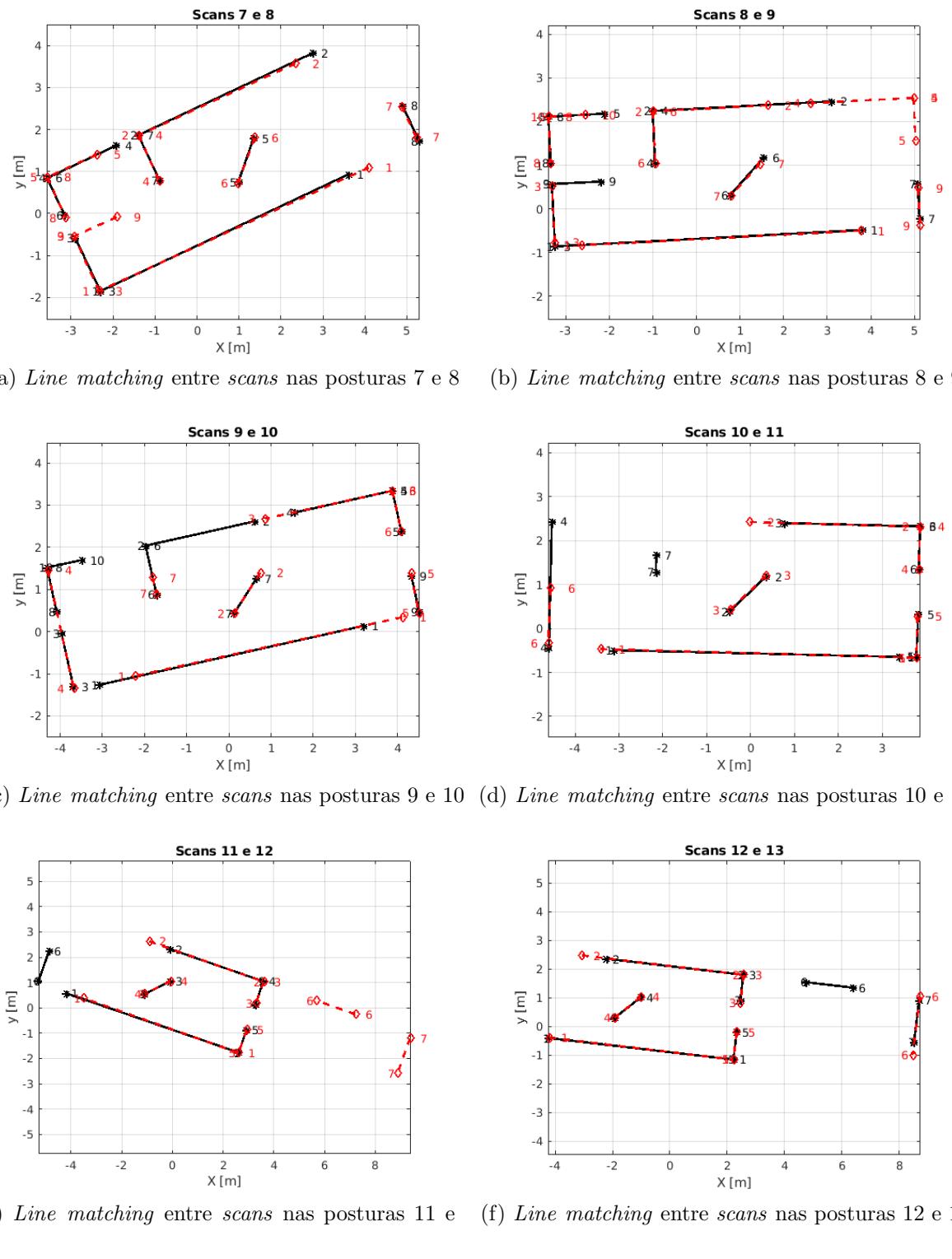


FIGURA 4.11 – Line matching entre scans em posturas consecutivas de 1 a 7 do primeiro experimento.



(e) *Line matching entre scans nas posturas 11 e 12* (f) *Line matching entre scans nas posturas 12 e 13*

FIGURA 4.12 – *Line matching entre scans* em posturas consecutivas de 7 a 13 do primeiro experimento.

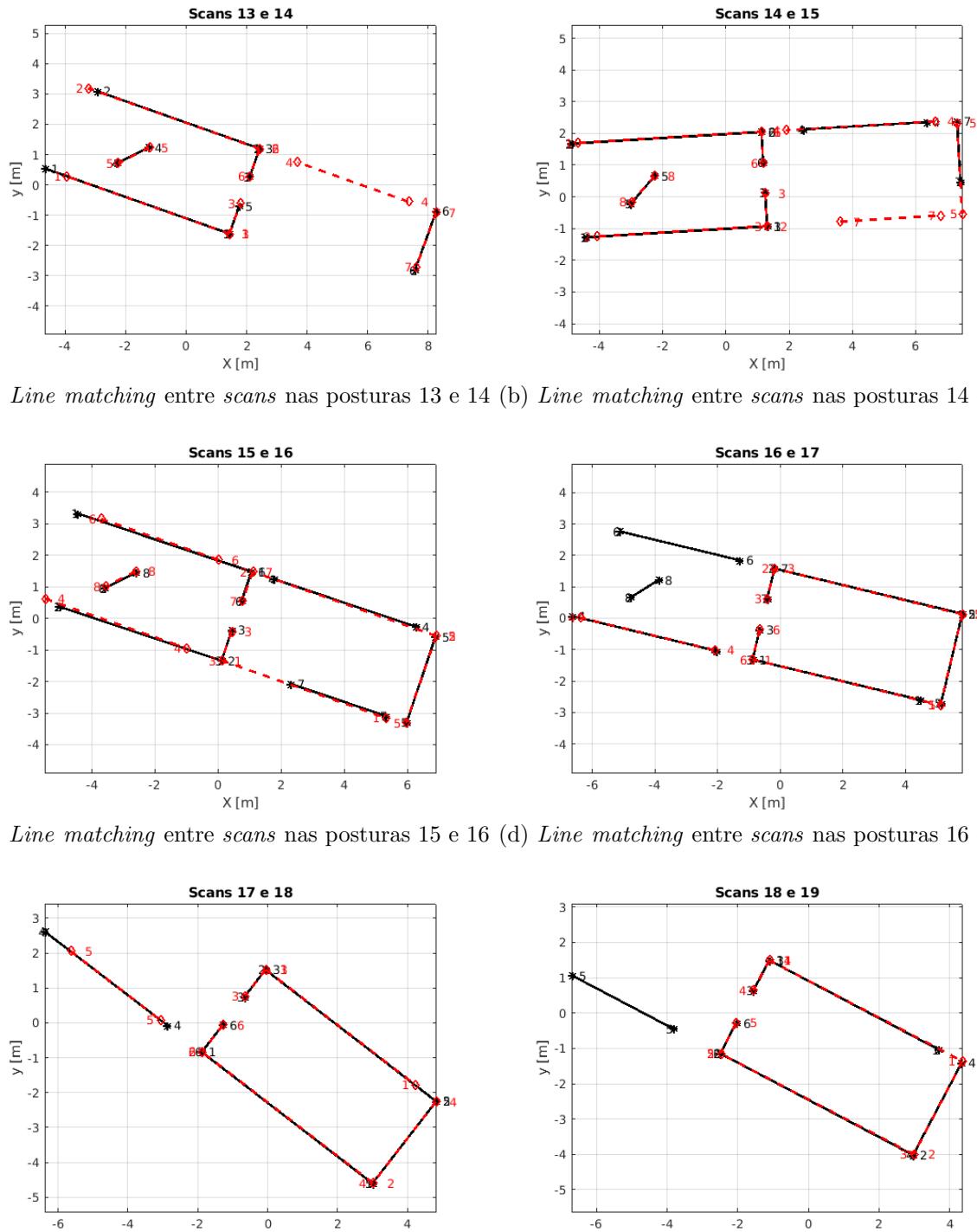
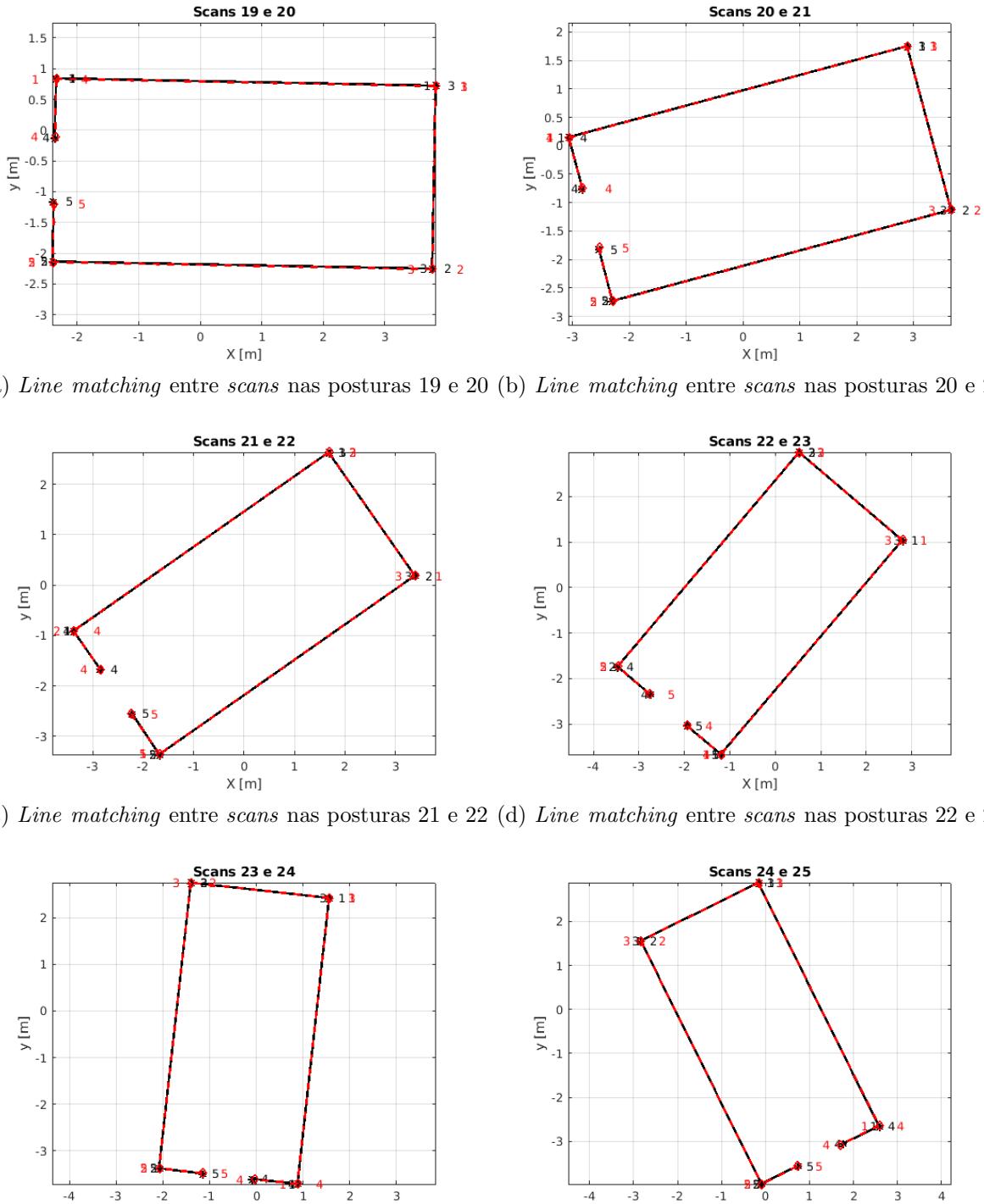
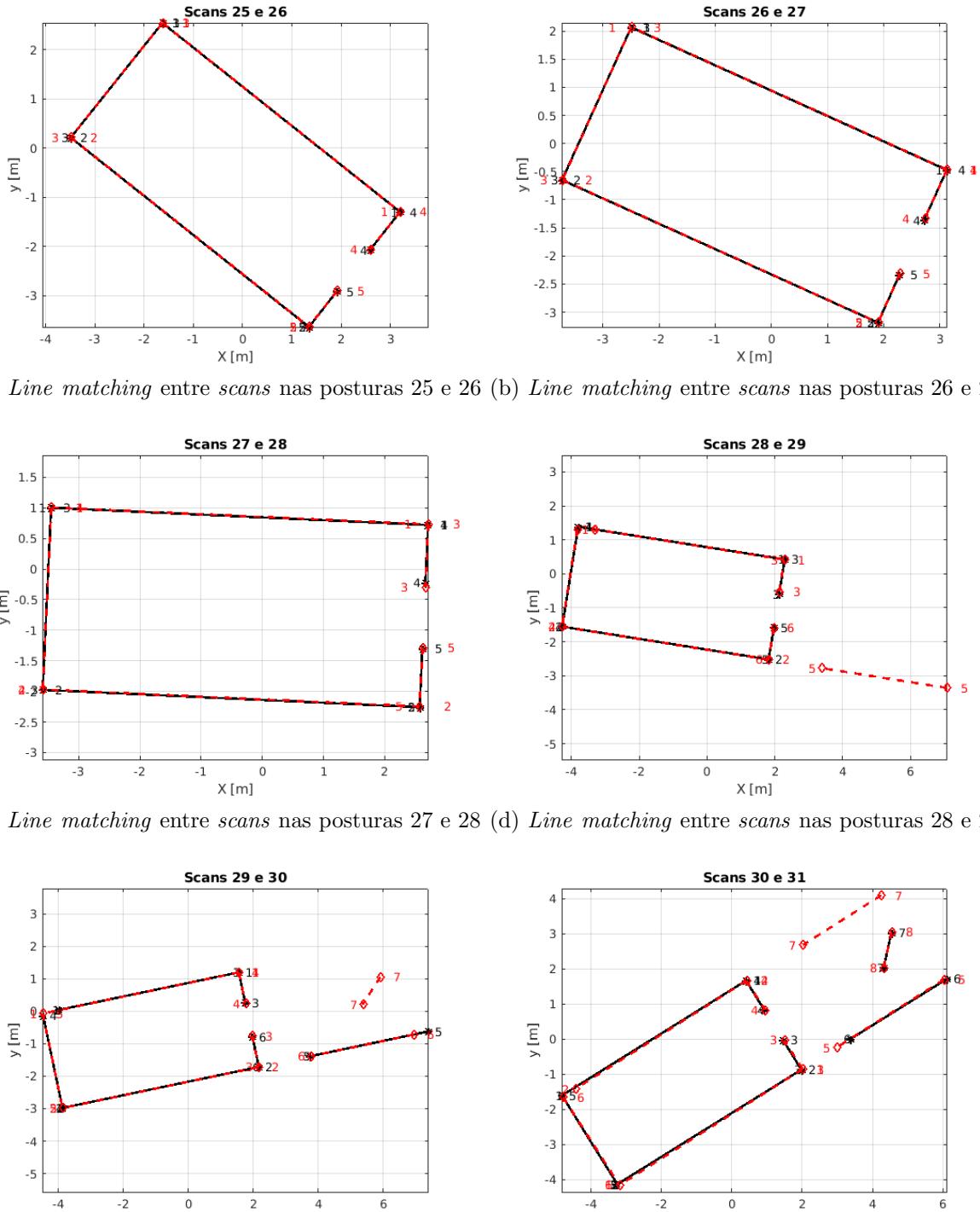


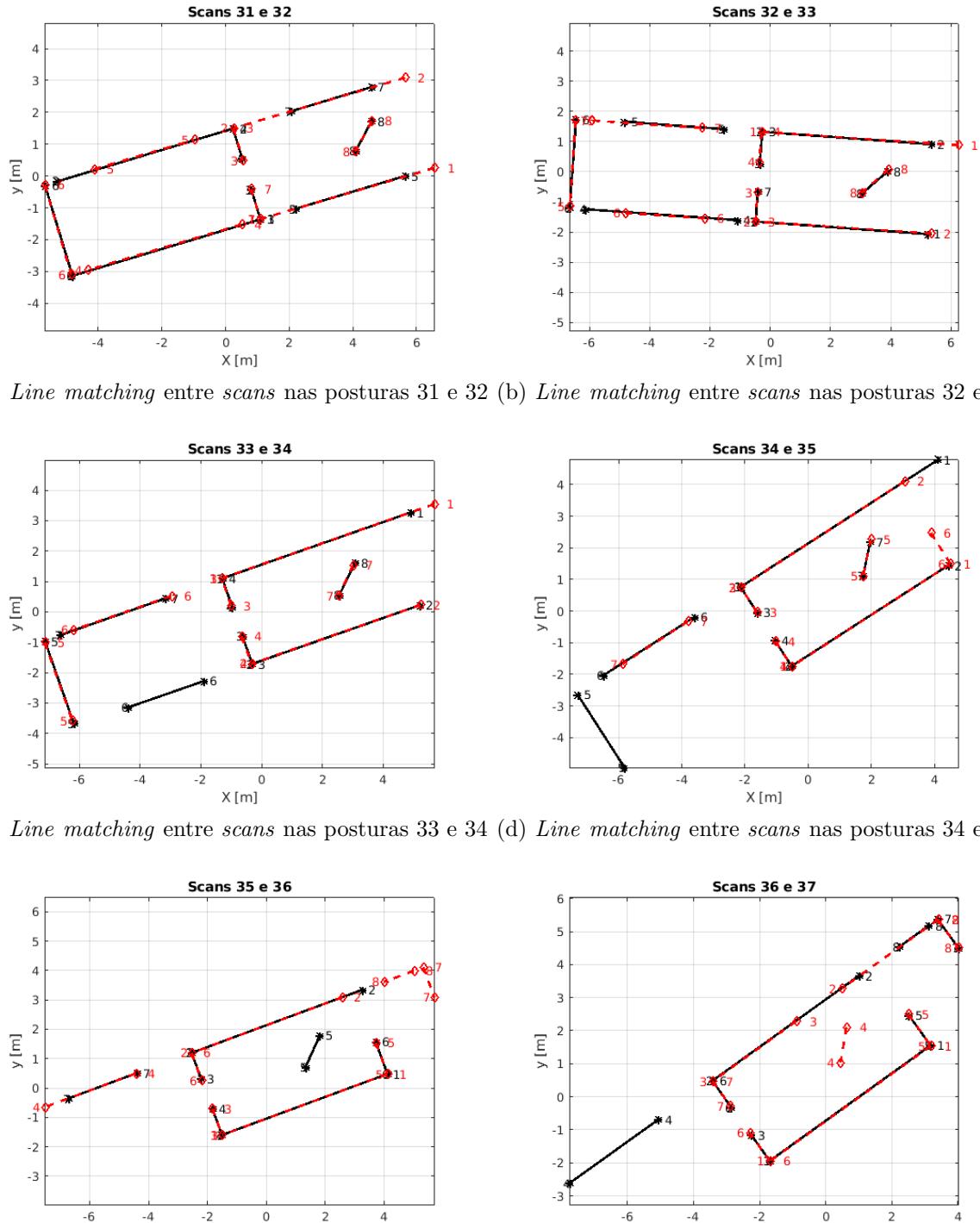
FIGURA 4.13 – *Line matching* entre *scans* em posturas consecutivas de 13 a 19 do primeiro experimento.



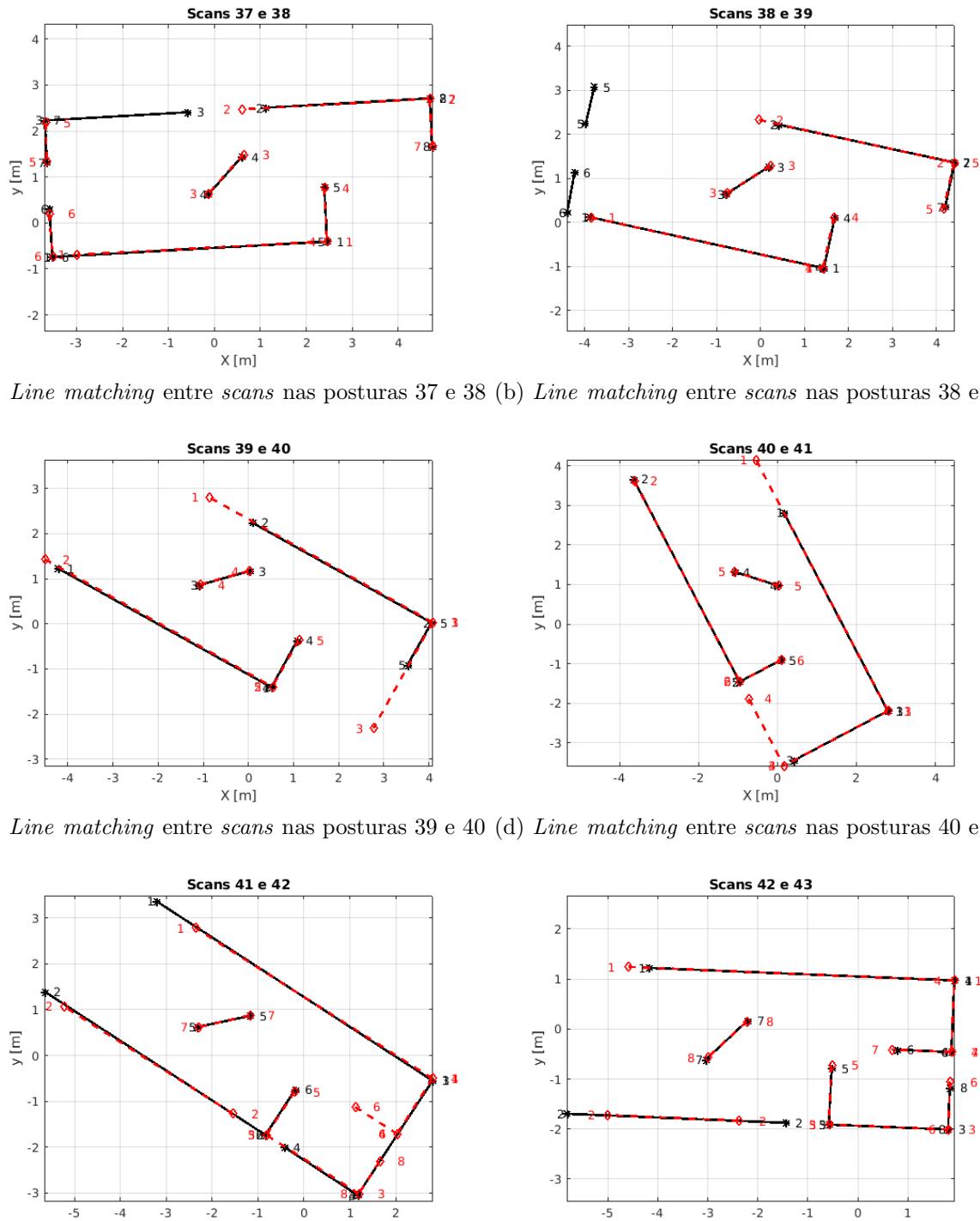
(e) Line matching entre scans nas posturas 23 e 24 (f) Line matching entre scans nas posturas 24 e 25
FIGURA 4.14 – Line matching entre scans em posturas consecutivas de 19 a 25 do primeiro experimento.



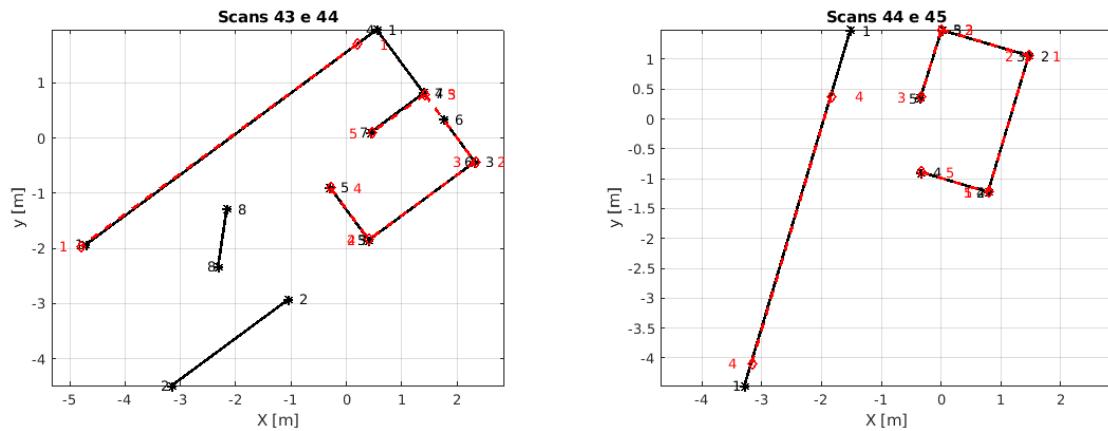
(e) *Line matching entre scans nas posturas 29 e 30* (f) *Line matching entre scans nas posturas 30 e 31*
FIGURA 4.15 – *Line matching entre scans em posturas consecutivas de 25 a 31 do primeiro experimento.*



(e) *Line matching entre scans nas posturas 35 e 36* (f) *Line matching entre scans nas posturas 36 e 37*
FIGURA 4.16 – *Line matching entre scans em posturas consecutivas de 31 a 37 do primeiro experimento.*



(e) *Line matching entre scans nas posturas 41 e 42* (f) *Line matching entre scans nas posturas 42 e 43*
FIGURA 4.17 – *Line matching entre scans em posturas consecutivas de 37 a 43 do primeiro experimento.*



(a) *Line matching* entre *scans* nas posturas 43 e 44 (b) *Line matching* entre *scans* nas posturas 44 e 45
 FIGURA 4.18 – *Line matching* entre *scans* em posturas consecutivas de 43 a 45 do primeiro experimento.

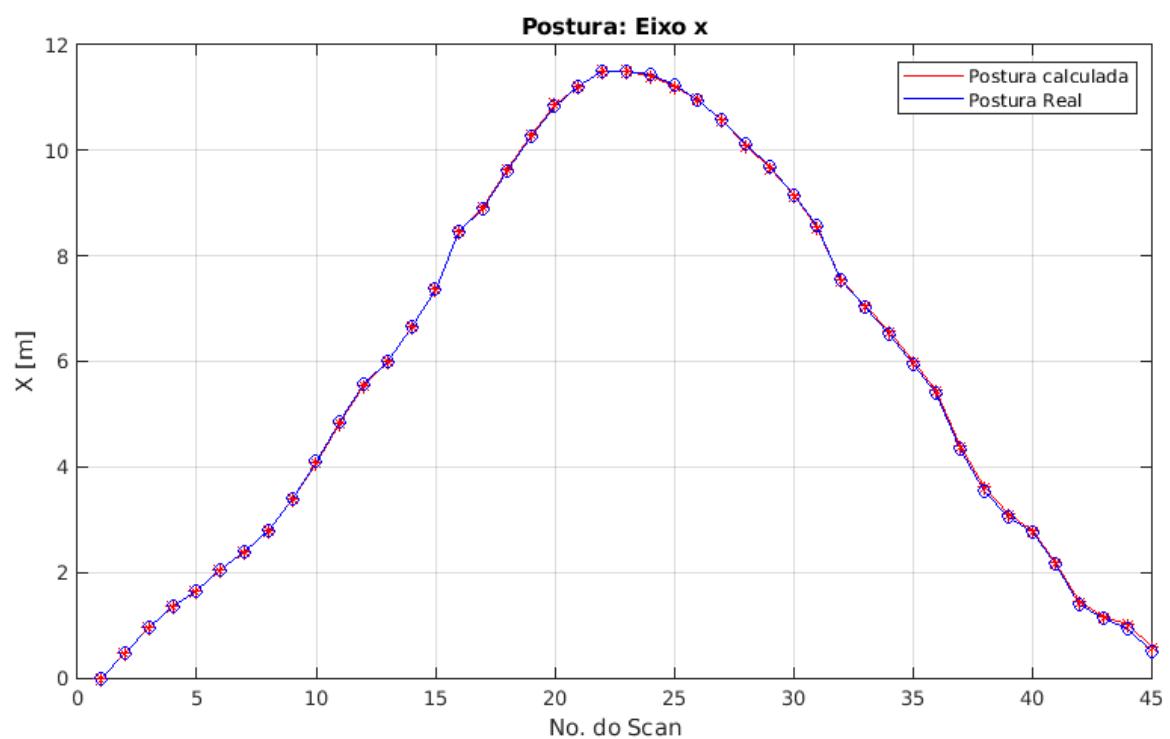


FIGURA 4.19 – Comparação da posição X presente nas posturas reais e estimadas do primeiro experimento.

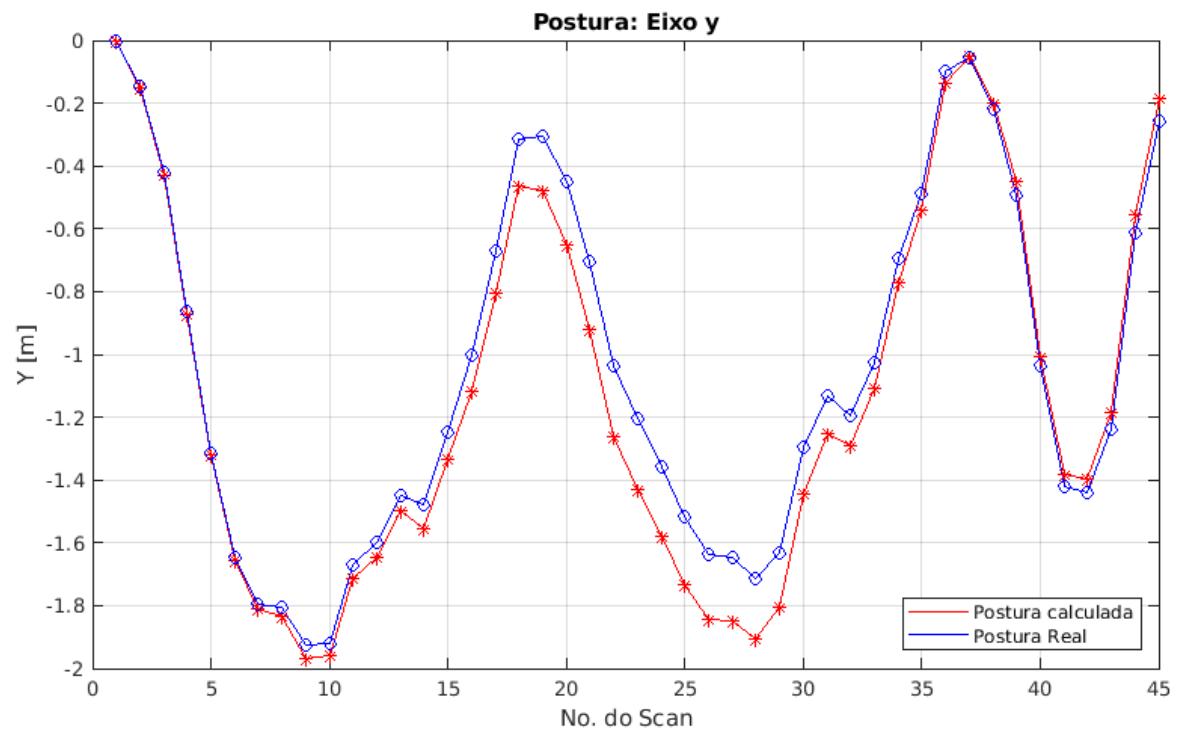


FIGURA 4.20 – Comparaçāo da posição Y presente nas posturas reais e estimadas do primeiro experimento.

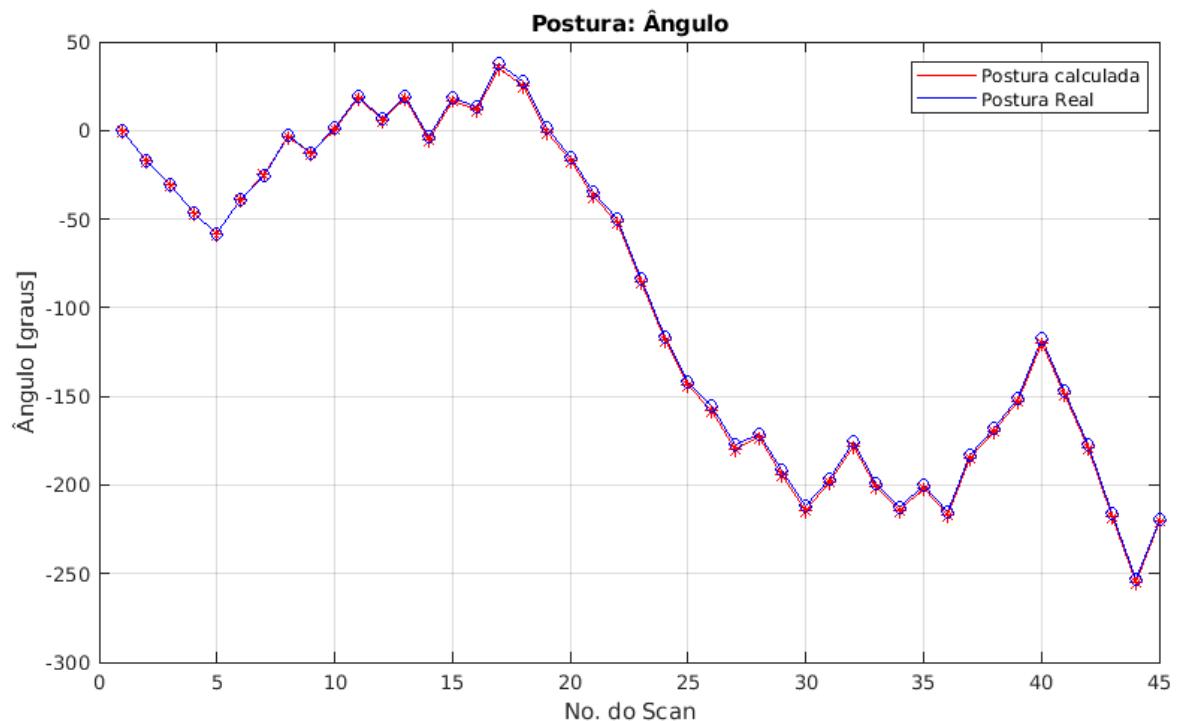


FIGURA 4.21 – Comparaçāo da orientação presente nas posturas reais e estimadas do primeiro experimento.

TABELA 4.5 – Erro entre as posturas real e estimada no sistema de coordenadas da postura inicial do primeiro experimento.

Erro entre Postura Real e Estimada							
Postura	x(m)	y(m)	$\theta(^{\circ})$	Postura	x(m)	y(m)	$\theta(^{\circ})$
1	0,000	0,000	0,000	24	0,011	0,223	2,085
2	0,003	0,005	-0,090	25	0,016	0,216	2,084
3	0,004	0,010	0,055	26	0,019	0,209	2,080
4	0,003	0,011	0,035	27	0,022	0,201	2,053
5	0,001	0,008	0,048	28	0,025	0,193	2,016
6	-0,000	0,009	-0,213	29	0,014	0,173	2,036
7	-0,006	0,013	-0,331	30	0,001	0,149	2,032
8	0,014	0,028	0,303	31	0,015	0,118	1,607
9	0,014	0,042	0,407	32	0,003	0,097	1,938
10	0,014	0,039	0,549	33	-0,016	0,082	1,567
11	0,021	0,042	0,780	34	-0,047	0,077	1,647
12	0,015	0,045	1,283	35	-0,051	0,053	1,545
13	0,008	0,049	1,469	36	-0,066	0,037	1,551
14	0,006	0,075	1,465	37	-0,068	-0,006	1,538
15	0,003	0,084	1,506	38	-0,069	-0,019	1,767
16	-0,002	0,112	2,273	39	-0,080	-0,040	1,843
17	-0,011	0,132	2,427	40	-0,043	-0,026	1,885
18	-0,030	0,152	2,380	41	-0,033	-0,042	1,819
19	-0,029	0,171	2,179	42	-0,049	-0,047	1,212
20	-0,020	0,200	2,175	43	-0,055	-0,054	1,286
21	-0,012	0,217	2,121	44	-0,063	-0,055	1,529
22	0,001	0,225	2,101	45	-0,072	-0,069	1,416
23	0,004	0,226	2,069				

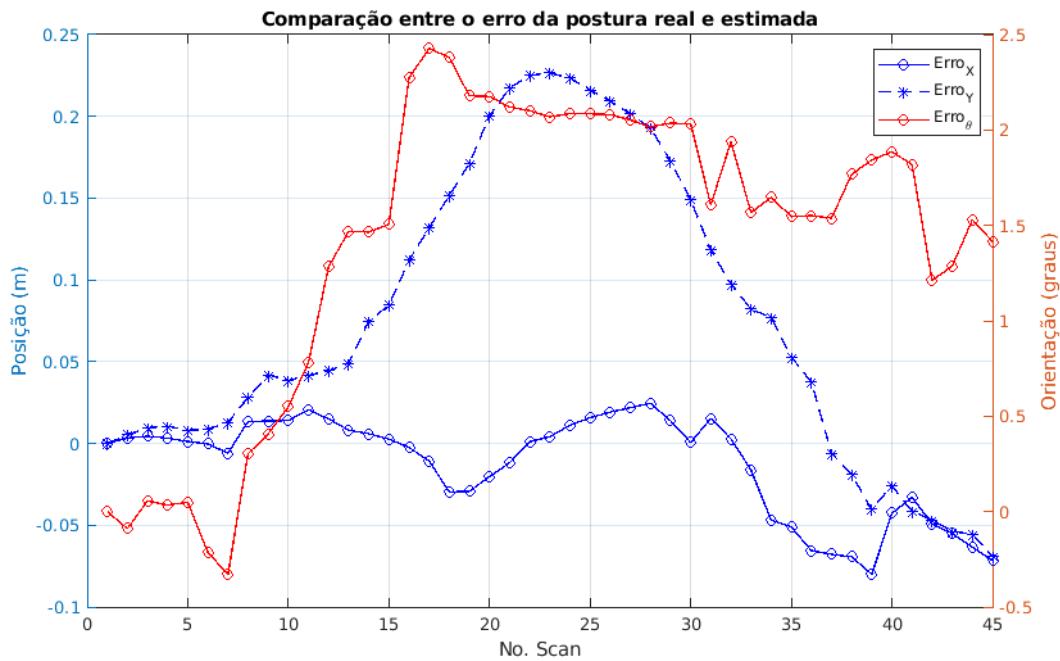


FIGURA 4.22 – Comparação da orientação presente nas posturas reais e estimadas do primeiro experimento.

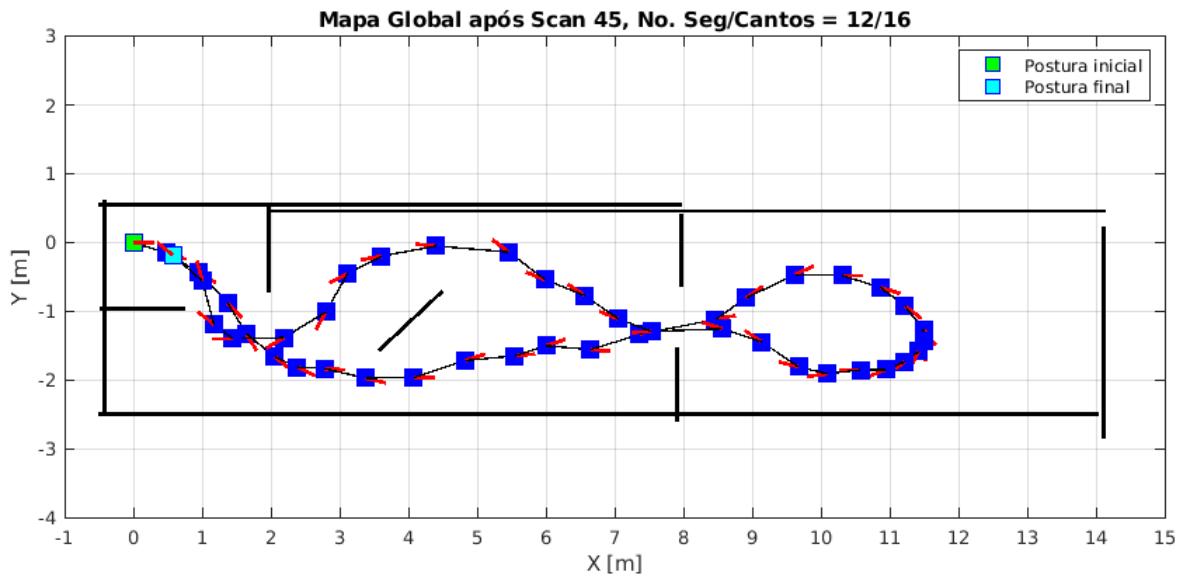


FIGURA 4.23 – Posturas estimadas do robô no mapa global do ambiente no sistema de coordenadas da postura inicial do primeiro experimento sem fechamento de laço e com a junção de segmentos.

e os segmentos foram ajustados por meio da junção de segmentos, apresenta 12 segmentos e 16 cantos e não possui muito erro, apesar de que em determinadas posturas estimadas foi detectado um erro acumulativo que pode ser corrigido por meio do fechamento do laço a fim de melhorar a estimativa das posturas e o mapa global. O mapa global em questão não precisaria ser corrigido na etapa de fechamento do laço, pois apresenta um erro acumulativo reduzido e, consequentemente, um erro não tão elevado, mas para se

obter um resultado ainda melhor a etapa de fechamento de laço pode ser utilizada.

Mediante tais resultados, pode-se comparar o mapa simulado do ambiente de nove paredes criado no *software Gazebo* com o mapa estimado gerado, sendo ambos em relação à postura inicial, conforme mostrado na Figura 4.24.

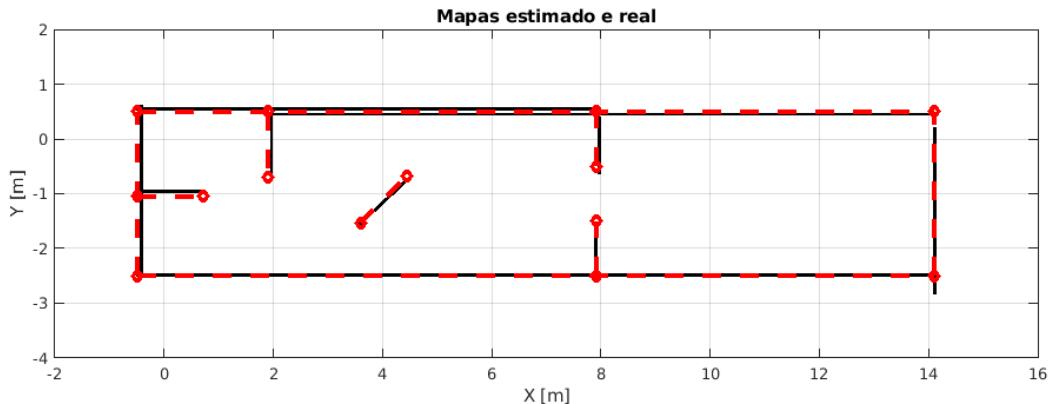


FIGURA 4.24 – Comparaçāo entre o mapa simulado do ambiente de nove paredes e o mapa estimado.

De acordo com a Figura 4.24, as linhas tracejadas e os círculos vermelhos representam os segmentos e pontos terminais do ambiente simulado de nove paredes, enquanto que as linhas contínuas pretas retratam os segmentos estimados do mapa global estimado sem fechamento de laço e com a junção de segmentos. Com isso, pode-se observar que, mesmo sem o processo de fechamento do laço, o mapa estimado do primeiro experimento foi próximo ao mapa simulado do ambiente desenvolvido de nove paredes no Gazebo.

4.2.2 Segundo Experimento

O segundo experimento se iniciou com a exploração manual do robô no ambiente simulado com dez paredes no Gazebo, em que sua postura inicial apresentou as coordenadas x igual a 0,5 metros, y igual a 2,5 m e θ igual a 0 graus e foi obtida de forma arbitrária. Com isso, foram extraídas 38 posturas ao longo do processo e o sensor *scanner a laser* realizou a leitura da nuvem de pontos do ambiente para cada postura obtida. A partir disso, fez-se testes sem e com fechamento de laço a fim de validar o uso da técnica para melhoria dos resultados e com a junção de segmentos. Assim, primeiramente, foram apresentados os resultados sem o fechamento de laço e depois com o fechamento de laço.

4.2.2.1 Caso 1: sem Fechamento de Laço

Dante disso, as mesmas etapas explicadas no primeiro experimento foram feitas no segundo experimento, as quais foram a extração dos dados e tratamento dos mesmos para

o ajuste dos segmentos obtidos no final do algoritmo de RANSAC para o *scan* verificado a cada postura.

Em seguida, foi feita a correspondência entre os segmentos de *scans* consecutivos em cada postura por meio do algoritmo de SLAM com correspondência de características. Com isso, foram gerados os dados de rotação e translação para realizar o *matching* entre os segmentos pertencentes a posturas consecutivas, conforme pode ser evidenciado na Tabela 4.6.

TABELA 4.6 – Rotação em graus e translação em metros obtidas entre *scans* em posturas consecutivas do segundo experimento.

Scans	Rotação(graus)	Translação(m)	Scans	Rotação(graus)	Translação(m)
1/2	-23,910	0,000	20/21	7,007	0,476
2/3	-0,017	0,838	21/22	-12,176	0,237
3/4	-17,993	0,740	22/23	-13,055	0,637
4/5	-8,300	0,647	23/24	-13,766	0,600
5/6	33,142	0,681	24/25	-24,277	0,632
6/7	-19,158	0,764	25/26	24,952	0,747
7/8	45,901	1,253	26/27	-9,581	0,400
8/9	-0,185	1,708	27/28	-1,544	1,382
9/10	8,340	1,238	28/29	17,390	1,042
10/11	-8,476	1,439	29/30	-47,718	0,610
11/12	22,888	1,515	30/31	8,518	0,885
12/13	-23,251	1,138	31/32	28,554	0,837
13/14	-18,172	1,609	32/33	55,521	0,787
14/15	-14,506	0,595	33/34	-13,968	0,698
15/16	-26,077	0,137	34/35	-18,458	0,971
16/17	-33,560	1,118	35/36	-19,444	1,015
17/18	-62,554	0,425	36/37	-24,520	0,725
18/19	-27,125	0,874	37/38	-10,672	0,523
19/20	6,942	0,300			

Mediante tais dados adquiridos de rotação em graus e translação em metros sem utilizar a técnica de fechamento de laço, foram calculadas as posturas estimadas com relação às posturas reais. Assim, a Tabela 4.7 apresentou os valores dos erros calculados entre as posturas reais e as posturas estimadas.

TABELA 4.7 – Erro entre as posturas real e estimada no sistema de coordenadas da postura inicial do segundo experimento sem fechamento do laço.

Erro entre Postura Real e Estimada							
Postura	x(m)	y(m)	$\theta(^{\circ})$	Postura	x(m)	y(m)	$\theta(^{\circ})$
1	0,000	0,000	0,000	20	0,010	-0,070	0,121
2	-0,000	0,000	0,033	21	0,018	0,066	1,897
3	0,002	0,004	0,050	22	0,013	0,068	2,077
4	0,002	0,002	-0,220	23	0,160	0,231	6,105
5	-0,007	0,004	-0,384	24	0,141	0,215	7,053
6	0,010	0,007	-0,459	25	0,091	0,152	6,992
7	0,011	0,002	-0,347	26	0,067	0,054	6,823
8	0,005	-0,020	-0,652	27	0,061	-0,076	7,412
9	0,017	-0,040	-0,467	28	0,019	-0,349	5,114
10	0,016	-0,053	-0,336	29	0,073	-0,892	9,110
11	0,039	-0,063	-0,296	30	-0,021	-0,949	9,155
12	0,056	-0,084	-0,735	31	-0,171	-1,092	8,525
13	0,054	-0,109	-0,874	32	-0,188	-1,321	7,698
14	0,052	-0,129	-0,966	33	-0,123	-1,409	8,022
15	0,037	-0,160	-0,298	34	-0,083	-1,489	7,877
16	0,033	-0,166	-0,147	35	-0,049	-1,603	7,887
17	0,015	-0,172	0,074	36	-0,087	-1,760	8,216
18	-0,002	-0,149	-0,398	37	-0,153	-1,842	8,012
19	0,003	-0,117	-0,011	38	-0,209	-1,888	8,015

A partir da Tabela 4.7, verificou-se que o maior erro obtido para as posições x e y foi de -20,9 centímetros e -1,888 metros na postura 38, enquanto que a orientação com maior erro foi de 9,155 graus na postura 30. Isso pode ser visualizado de forma gráfica nas Figuras 4.25, 4.26, 4.27, em que a linha azul representa a curva referente à trajetória das posturas reais e a linha vermelha evidencia a curva de posturas estimadas, enquanto que os círculos azuis são as posturas reais e os asteriscos vermelhos são as posturas estimadas. Já a Figura 4.28 mostra o erro obtido para cada postura, em que a curva azul contínua representa o erro na posição x em metros com cada postura retratada por um círculo azul, a curva azul tracejada evidencia o erro na posição y em metros com cada postura simbolizada por um asterisco azul e a curva vermelha contínua mostra o erro na orientação

em graus com cada postura representada por um círculo vermelho.

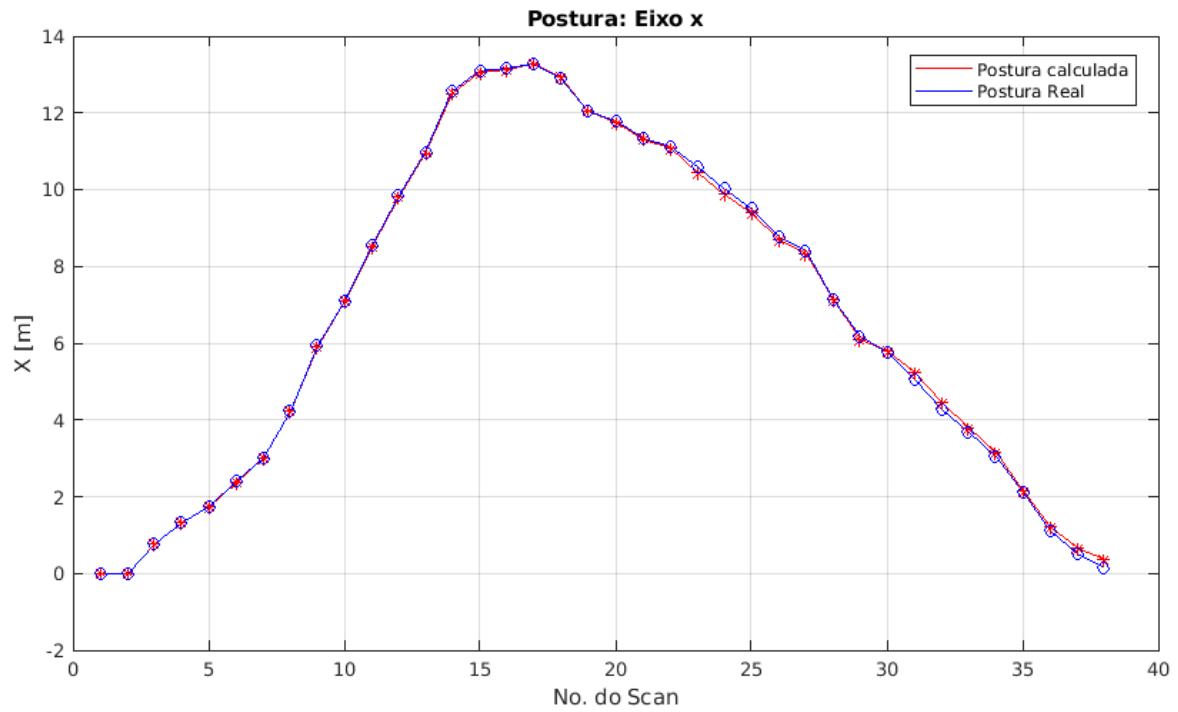


FIGURA 4.25 – Comparação da posição x em metros presente nas posturas reais e estimadas do segundo experimento sem fechamento de laço.

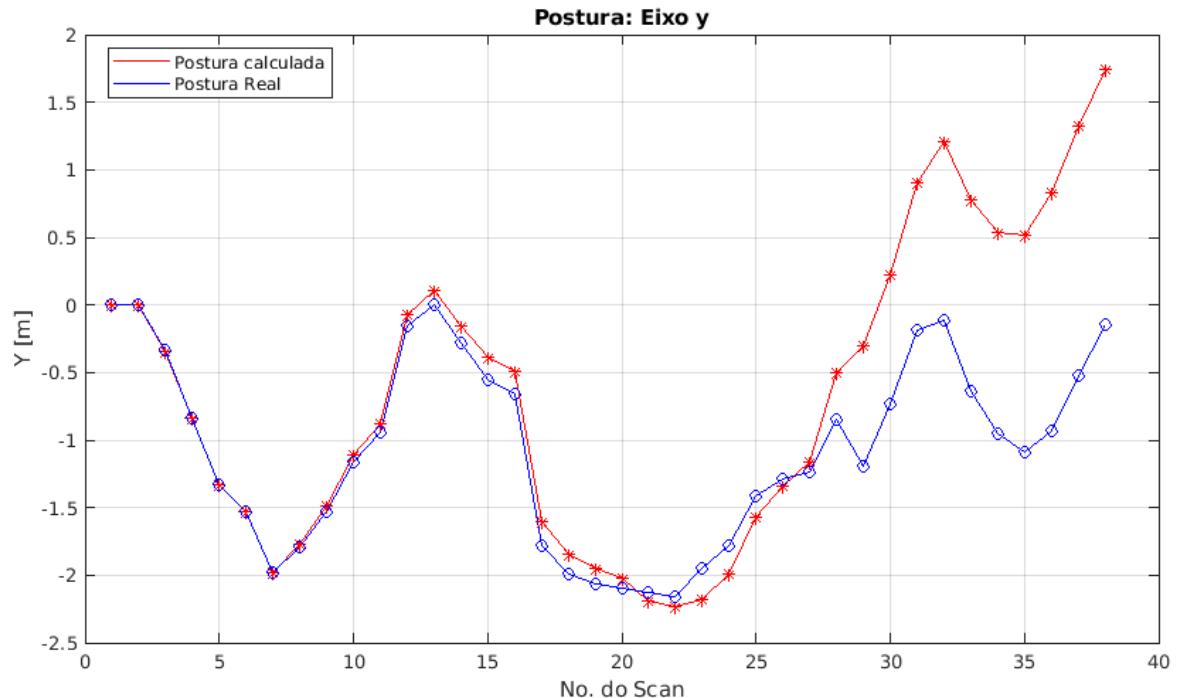


FIGURA 4.26 – Comparação da posição y em metros presente nas posturas reais e estimadas do segundo experimento sem fechamento de laço.

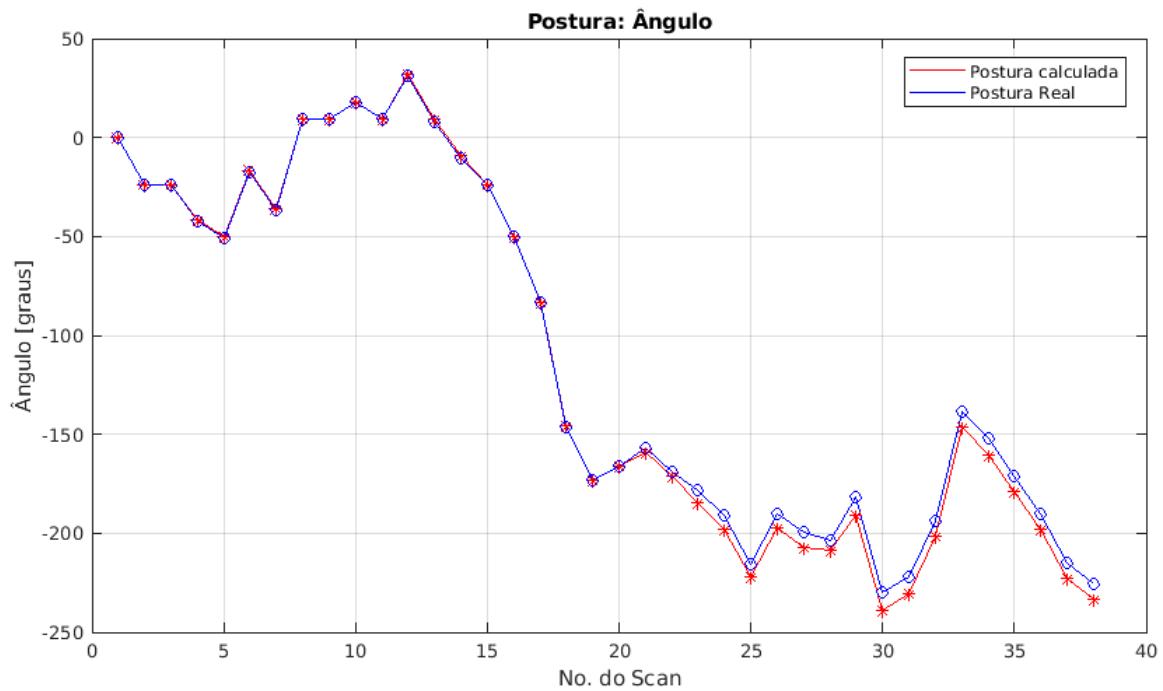


FIGURA 4.27 – Comparação da orientação em graus presente nas posturas reais e estimadas do segundo experimento sem fechamento de laço.

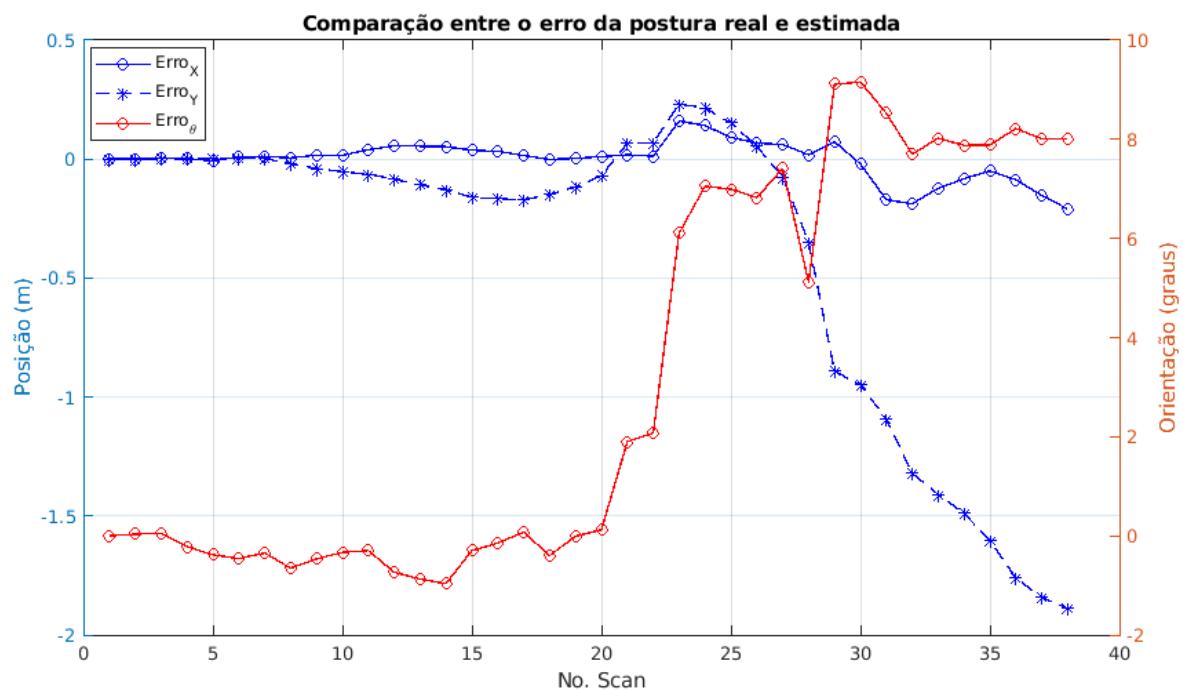


FIGURA 4.28 – Comparação do erro presente nas posturas reais e estimadas do segundo experimento sem fechamento de laço.

De acordo com as posturas estimadas obtidas e o erro calculado entre as posturas reais e estimadas, observou-se que, mesmo sem a etapa de fechamento do laço, os gráficos do erro na posição x e no ângulo θ convergiram em quase toda a trajetória do robô até o final das 38 posturas, enquanto que o gráfico do erro da posição y das posturas convergiu no início da trajetória do robô, porém no final divergiu. Diante disso, além da correspondência dos segmentos de reta feita entre os *scans* nas posturas consecutivas, foi realizada a estimativa do mapa global sem a utilização da técnica de fechamento de laço (*loop closure*) e com a junção de segmentos, conforme mostrado na Figura 4.29.

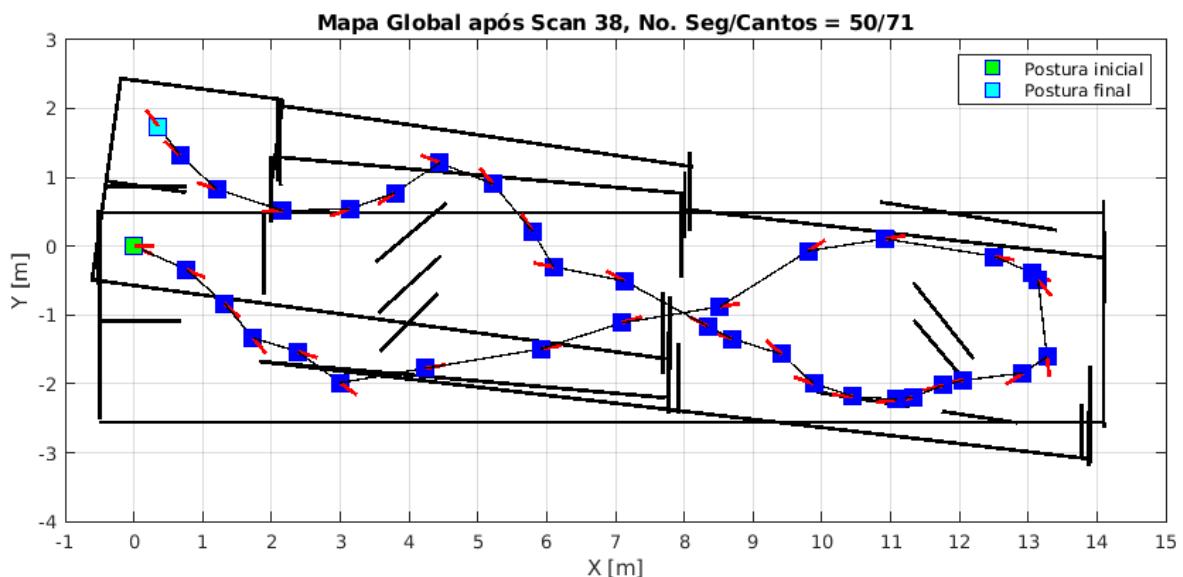


FIGURA 4.29 – Posturas estimadas do robô no mapa global do ambiente no sistema de coordenadas da postura inicial do segundo experimento sem fechamento de laço e com a junção de segmentos.

A Figura 4.29 evidenciou que foram detectados 50 segmentos e 71 cantos no ambiente simulado, o mapa foi construído com a junção de segmentos e apresentou erro em seu resultado, devido aos diversos segmentos e posturas estimados que geraram erro acumulativo ao longo do processo de SLAM 2D, que é um problema muito comum, e que não foi feito o fechamento de laço para esse caso.

Na imagem, as linhas pretas mais espessas representam as paredes do ambiente retratadas no mapa, os quadrados azuis são as posturas estimadas calculadas ao longo da trajetória do robô representada pelas linhas mais finas, a linha vermelha é a orientação de cada postura, o quadrado verde retrata a postura inicial do robô e o quadrado da cor ciano evidencia a postura final do robô na simulação.

Dessa forma, a partir dos resultados adquiridos no segundo experimento simulado, viu-se a necessidade de realizar o mesmo procedimento, mas com a etapa de fechamento de laço pertencente ao processo, como pode ser retratado na próxima seção.

4.2.2.2 Caso 2: com Fechamento de Laço

Para esse caso foram aplicadas as mesmas etapas explicadas na seção anterior, porém incluiu-se o processo de fechamento do laço. Por meio disso, foram avaliadas as correspondências das características entre os *scans*, primeiramente, sem o fechamento de laço e foram obtidas as rotações e translações entre dois *scans* consecutivos, conforme foi visto na Tabela 4.6 da seção anterior.

A partir disso, foram verificadas as correspondências entre *scans* que foram inadequadas em alguns *scans*, como mostra a Figura 4.30. As combinações inapropriadas observadas na figura, como exemplo, foram essenciais para se realizar o fechamento do laço a fim de corrigir as posturas estimadas referentes aos *scans* nas posturas 21, 23, 28 e 29 que foram as linhas vermelhas tracejadas, enquanto que os *scans* nas posturas anteriores foram as linhas pretas.

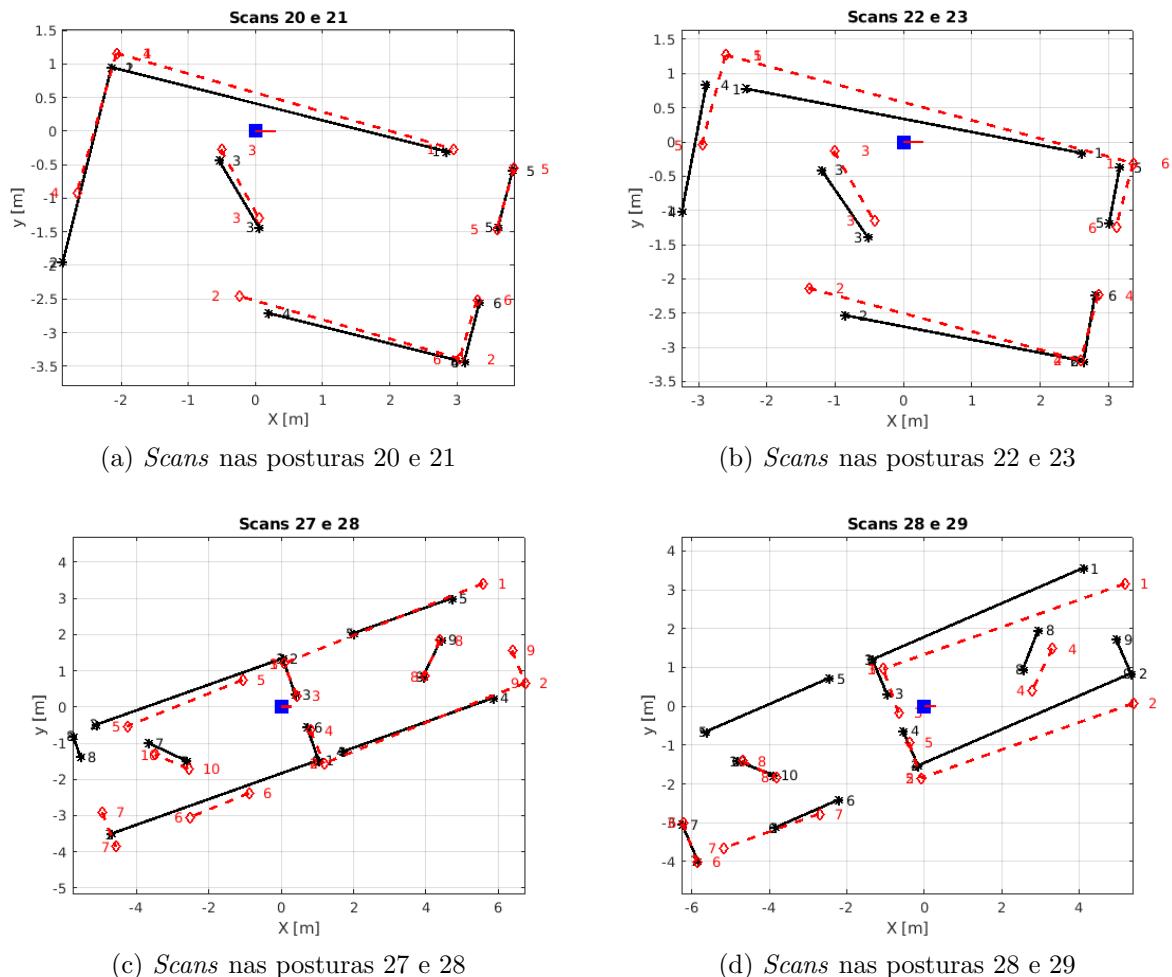


FIGURA 4.30 – *Line matchings* sem fechamento do laço entre *scans* em posturas consecutivas que foram inadequadas no segundo experimento.

Dante disso, o processo do fechamento de laço foi realizado entre as posturas 21 a 32, sendo visualizados os *scans* nas posturas 21, 23, 28 e 29 nos sistemas de coordenadas local

e global a fim de destacar os marcos que foram os pontos terminais ou cantos novos que foram obtidos localmente com os já identificados no sistema de coordenadas global.

Por meio disso, para as posturas analisadas, definiu-se os marcos usados para calcular novamente a rotação e translação e redefinir a postura estimada. Com isso, foi determinada a região no mapa global que os marcos utilizados no processo de fechamento de laço estariam presentes que foi a região que apresenta as duas paredes do meio do ambiente que separam as salas maiores, conforme a Figura 4.31 que mostra, como exemplo, os marcos registrados (círculos verdes) que são identificados e comparados com os pontos terminais locais (círculos vermelhos) dos *scans* locais nas posturas 21 e 27.

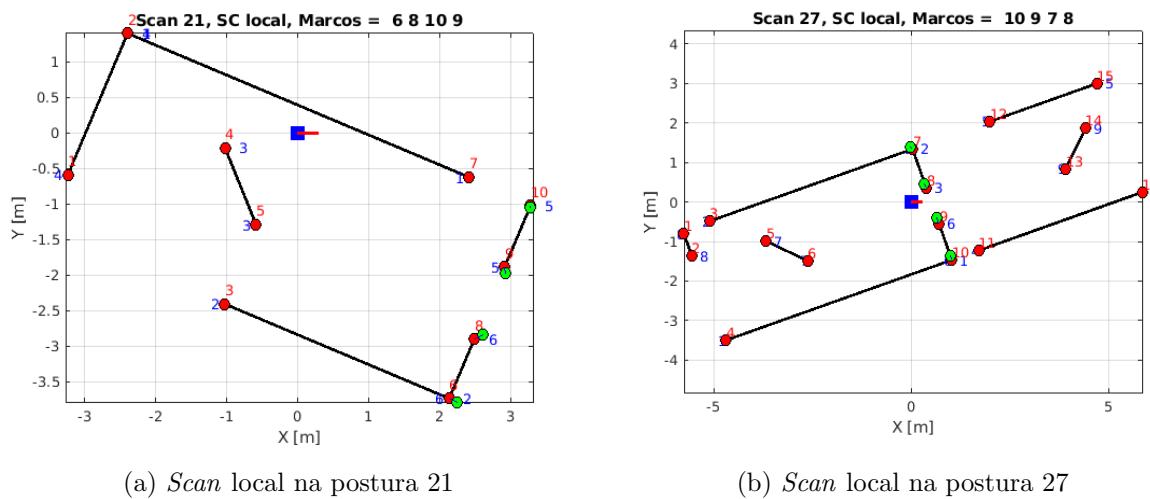
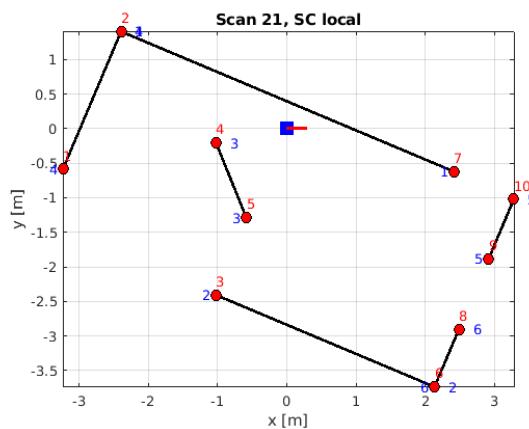


FIGURA 4.31 – Pontos terminais associados aos marcos registrados no sistema de coordenadas local do *scan* nas posturas 21 e 27.

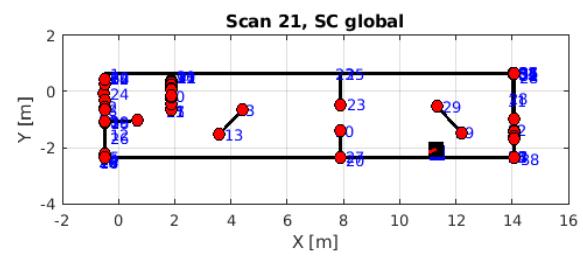
Dessa forma, calculou-se a distância e o ângulo dos pontos terminais selecionados com relação à postura local que são associados com os marcos registrados e os resultados foram atribuídos à postura estimada no sistema de coordenadas global e gerou-se a nova postura estimada para cada *scan* analisado, conforme pode ser visto nas Figuras 4.32, 4.33, 4.34 e 4.35 para os *scans* nas posturas 21, 23, 28 e 29, respectivamente.

Dessa forma, quatro posturas estimadas foram ajustadas, o quadrado azul do *scan* no sistema de coordenadas global representou a postura ajustada depois da detecção dos marcos e o quadrado preto representa a postura antes do ajuste. Com isso, os novos erros foram calculados entre todas as 38 posturas reais e estimadas após a etapa de fechamento do laço, como pode ser evidenciado na Tabela 4.8.

Com isso, observou-se que o maior erro na posição x foi de 5,6 centímetros na postura 12, na posição y foi de -17,2 centímetros na postura 17, o maior erro na orientação foi de -0,966 graus na postura 14. Além disso, tais resultados podem ser ilustrados de forma gráfica nas Figuras 4.36, 4.37 e 4.38, nas quais a linha azul representa a trajetória das posturas reais que são retratadas pelos círculos azuis, a linha vermelha evindencia a

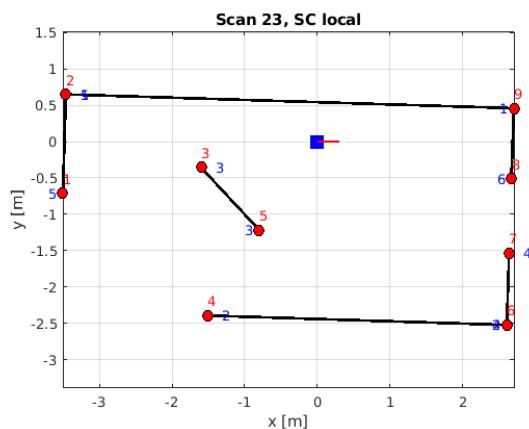


(a) Scan local da postura 21

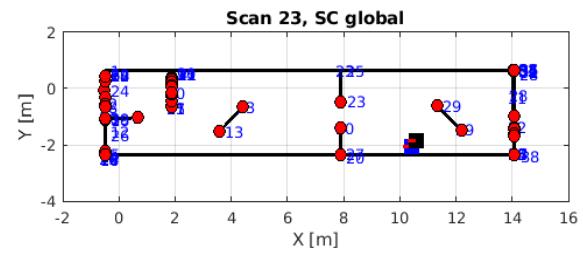


(b) Scan global da postura 21

FIGURA 4.32 – Scan nos sistemas de coordenadas local e global da postura 21.

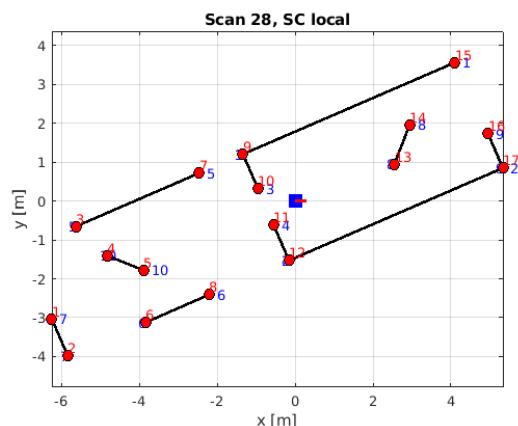


(a) Scan local da postura 23

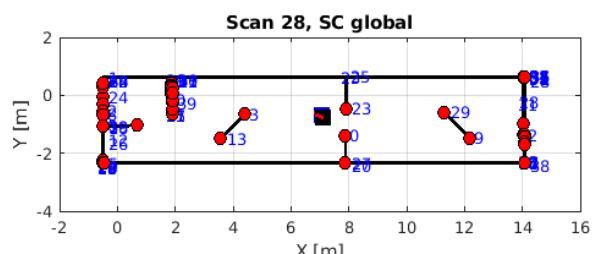


(b) Scan global da postura 23

FIGURA 4.33 – Scan nos sistemas de coordenadas local e global da postura 23.

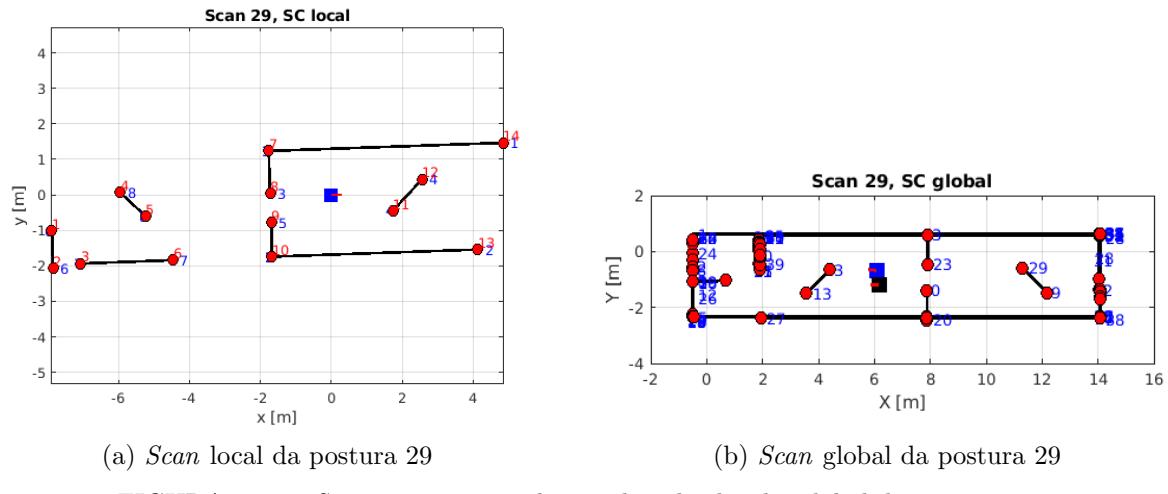


(a) Scan local da postura 28



(b) Scan global da postura 28

FIGURA 4.34 – Scan nos sistemas de coordenadas local e global da postura 28.



sequência de posturas estimadas e o asterisco vermelho representa cada uma das posturas estimadas que foram calculadas após o fechamento de laço. Na Figura 4.39, a linha contínua azul com os círculos azuis representa a trajetória do erro na posição x em metros entre as posturas reais e estimadas, a linha tracejada azul com asteriscos azuis retrata a trajetória do erro na posição y em metros entre as posturas, e a linha vermelha com círculos vermelhos é o erro da orientação em graus entre as posturas.

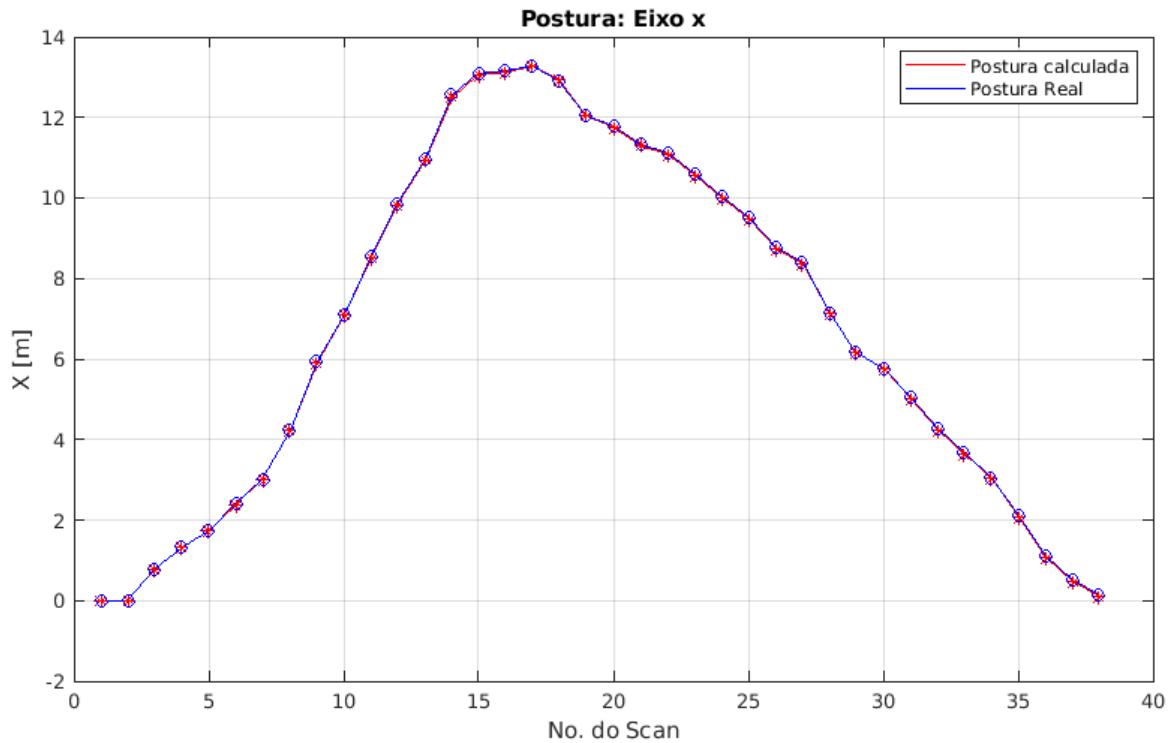


FIGURA 4.36 – Comparação da posição x em metros presente nas posturas reais e estimadas do segundo experimento com fechamento de laço.

Por meio das posturas estimadas ajustadas e do erro calculado entre as posturas reais

TABELA 4.8 – Erro entre as posturas real e estimada no sistema de coordenadas da postura inicial do segundo experimento com fechamento do laço.

Erro entre Postura Real e Estimada							
Postura	x(m)	y(m)	$\theta(^{\circ})$	Postura	x(m)	y(m)	$\theta(^{\circ})$
1	0,000	0,000	0,000	20	0,010	-0,070	0,121
2	-0,000	0,000	0,033	21	0,031	-0,077	-0,529
3	0,002	0,004	0,050	22	0,033	-0,077	-0,290
4	0,002	0,002	-0,220	23	0,034	-0,101	-0,514
5	-0,007	0,004	-0,384	24	0,037	-0,087	-0,447
6	0,010	0,007	-0,459	25	0,038	-0,080	-0,452
7	0,011	0,002	-0,347	26	0,040	-0,081	-0,485
8	0,005	-0,020	-0,652	27	0,039	-0,057	-0,520
9	0,017	-0,040	-0,467	28	0,023	-0,071	-0,478
10	0,016	-0,053	-0,336	29	0,021	0,006	-0,614
11	0,039	-0,063	-0,296	30	0,024	-0,055	-0,439
12	0,056	-0,084	-0,735	31	0,026	-0,052	-0,396
13	0,054	-0,109	-0,874	32	0,030	-0,028	-0,492
14	0,052	-0,129	-0,966	33	0,025	-0,027	-0,168
15	0,037	-0,160	-0,298	34	0,025	-0,016	-0,314
16	0,033	-0,166	-0,147	35	0,045	0,009	-0,303
17	0,015	-0,172	0,074	36	0,043	-0,008	0,026
18	-0,002	-0,149	-0,398	37	0,042	-0,009	-0,178
19	0,003	-0,117	-0,011	38	0,042	-0,006	-0,175

e as novas posturas estimadas, observou-se que, depois da etapa de fechamento do laço, os gráficos do erro nas posições x e y e na orientação θ convergiram ao final da trajetória do robô. Isso demonstra a eficiência do processo de fechamento do laço que reduz o erro acumulativo no sistema de SLAM. Com isso, além da correspondência dos segmentos de reta feita entre os *scans* nas posturas consecutivas, foi realizada a estimação do novo mapa global com a técnica de fechamento de laço e com a junção de segmentos, conforme mostrado na Figura 4.40.

A Figura 4.40 mostrou que foram reconhecidos 13 segmentos e 16 cantos no ambiente simulado com dez paredes. As paredes do ambiente são representadas pelas linhas mais espessas, enquanto que cada quadrado azul ilustra cada postura da trajetória do robô

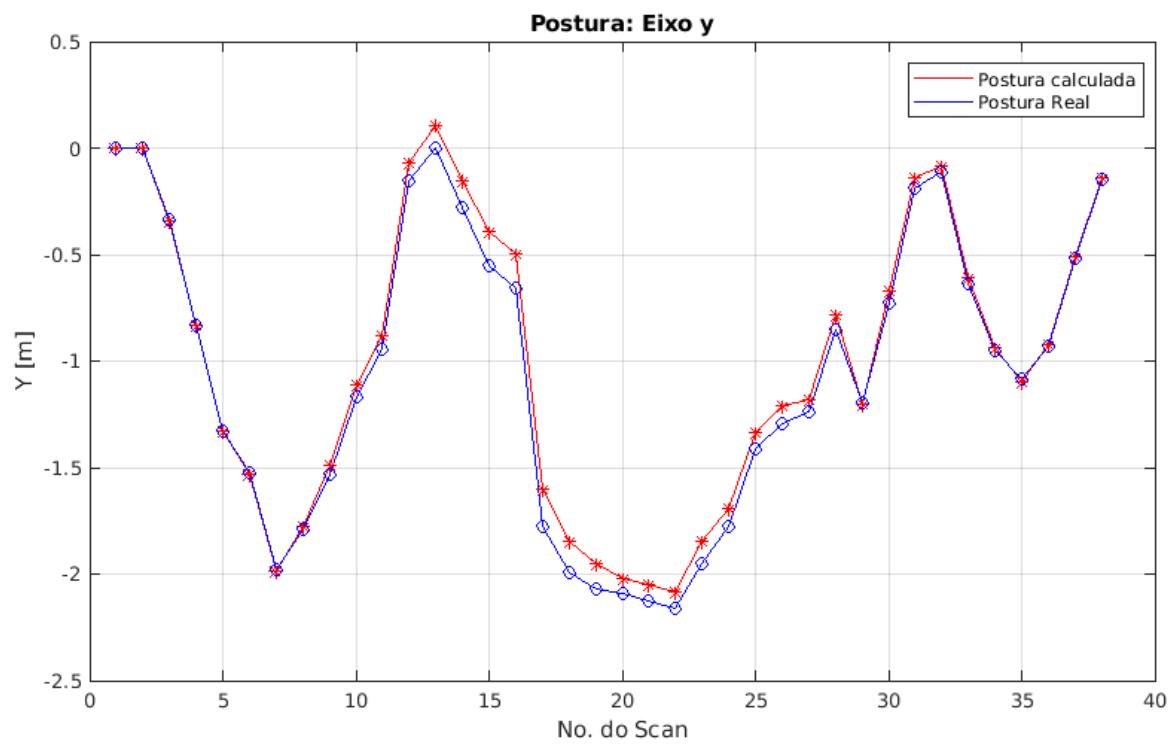


FIGURA 4.37 – Comparação da posição y em metros presente nas posturas reais e estimadas do segundo experimento com fechamento de laço.

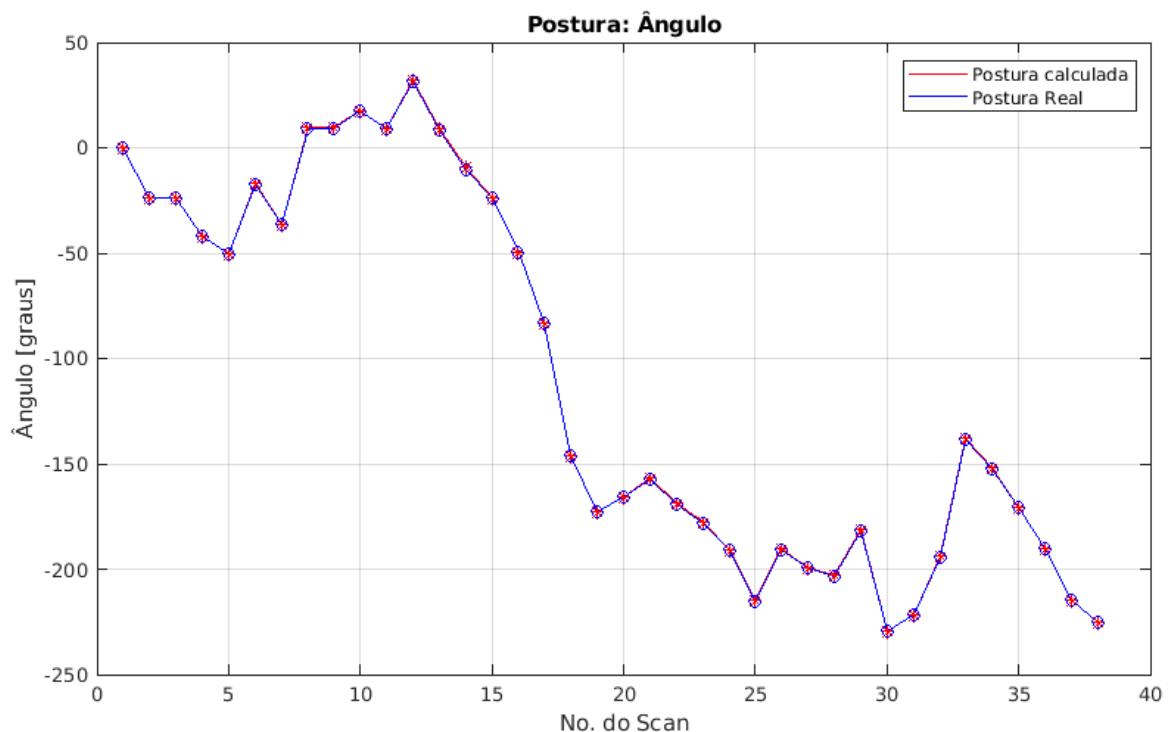


FIGURA 4.38 – Comparação da orientação em graus presente nas posturas reais e estimadas do segundo experimento com fechamento de laço.

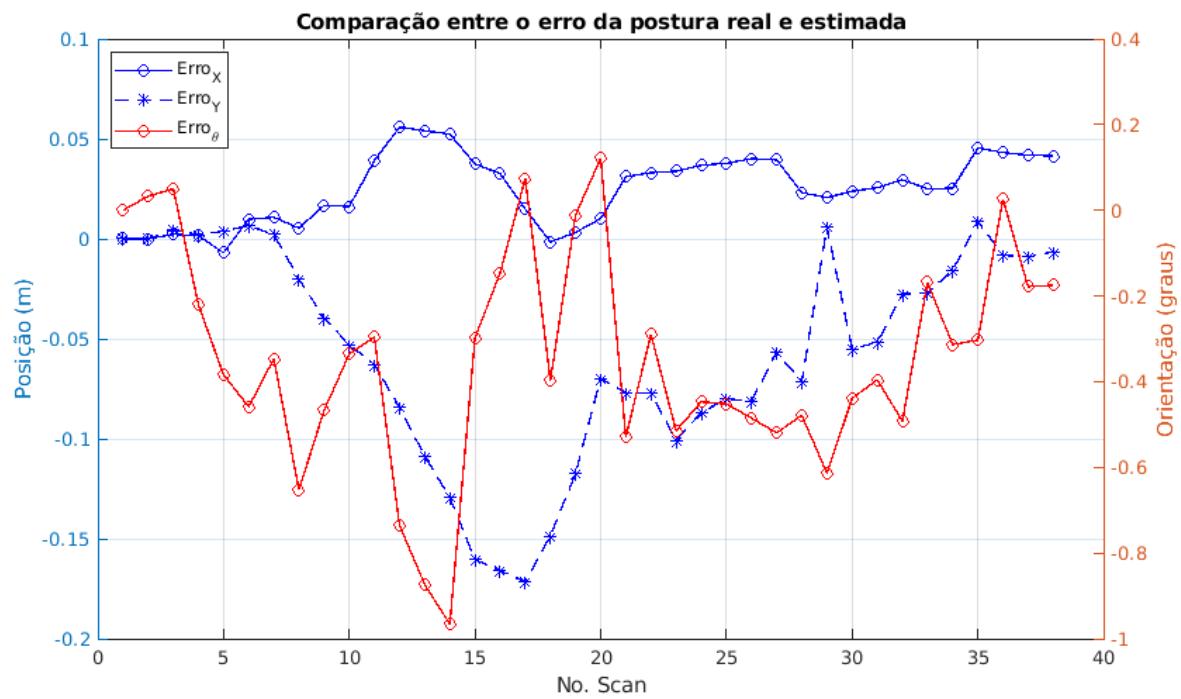


FIGURA 4.39 – Comparação do erro presente nas posturas reais e estimadas do segundo experimento com fechamento de laço.

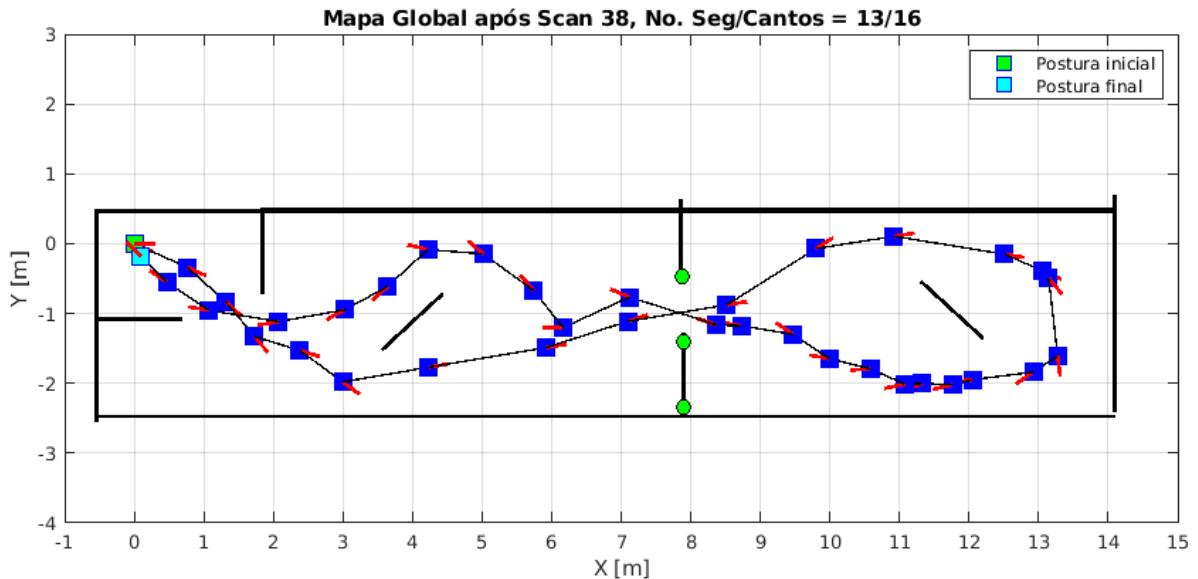


FIGURA 4.40 – Posturas estimadas do robô no mapa global do ambiente no sistema de coordenadas da postura inicial do segundo experimento com fechamento de laço e com a junção de segmentos.

que é visualizada pelas linhas pretas finas, a linha vermelha representa a orientação de cada postura do robô, o quadrado verde ilustra a postura inicial da trajetória do robô na simulação, o quadrado da cor ciano é a postura final estimada pela trajetória do robô na simulação, e os círculos verdes representam os marcos selecionados para o processo de fechamento de laço.

Com isso, o mapa gerado apresentou um erro reduzido no seu resultado em relação ao mapa do caso anterior sem fechamento do laço, pois o erro acumulativo entre as 38 posturas reais e estimadas diminuiu depois da técnica de fechamento de laço, melhorando assim o resultado do sistema de SLAM 2D com correspondência de características.

De acordo com os resultados do segundo experimento, comparou-se o mapa simulado do ambiente de dez paredes desenvolvido no *software* Gazebo com o mapa estimado gerado, sendo ambos em relação à postura inicial, como pode ser evidenciado na Figura 4.41.

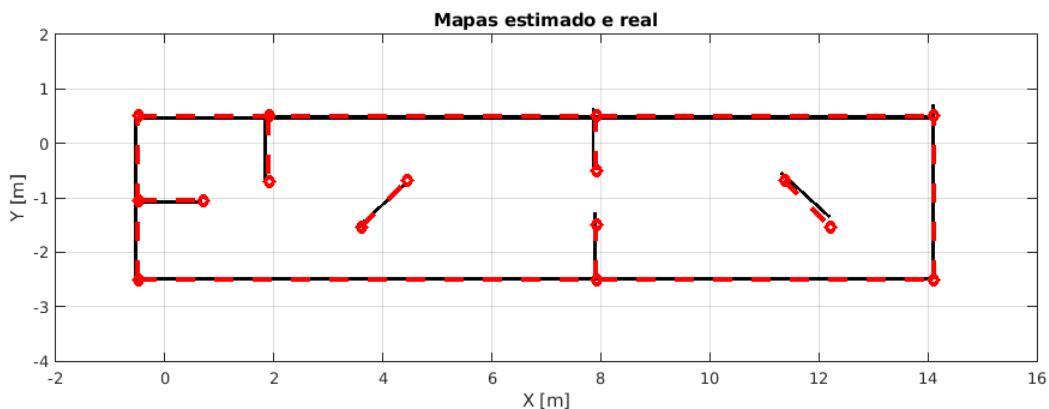


FIGURA 4.41 – Comparaçāo entre o mapa simulado do ambiente de dez paredes e o mapa estimado.

Na Figura 4.41, as linhas tracejadas e os círculos vermelhos ilustram as linhas e pontos terminais do ambiente simulado de dez paredes. As linhas contínuas pretas representam os segmentos de reta estimados do mapa global estimado com fechamento de laço. Com isso, observou-se que os mapas apresentaram poucas diferenças e com menor erro por causa da técnica de fechamento de laço, evidenciando que o mapa estimado do segundo experimento convergiu para o mapa simulado do ambiente desenvolvido de dez paredes no Gazebo. Dessa forma, o processo de SLAM com correspondência de características mostrou-se viável.

4.3 Resultado do Experimento Real

O experimento real foi realizado no corredor em frente ao Laboratório de Máquinas Inteligentes (LMI) localizado no ITA. A Figura 4.42 mostra a vista de satélite dos prédios e da biblioteca do ITA, bem como o prédio onde fica o LMI (destacado por um retângulo amarelo) em cujo corredor foi realizado o experimento com o robô real.

O ambiente real foi adaptado para que fosse semelhante ao ambiente simulado no Gazebo, porém apresentou área de 41,44 m², sendo 14,54 m de comprimento e 2,85 m de largura. Dessa forma, o ambiente que serviu de base para a construção do ambiente

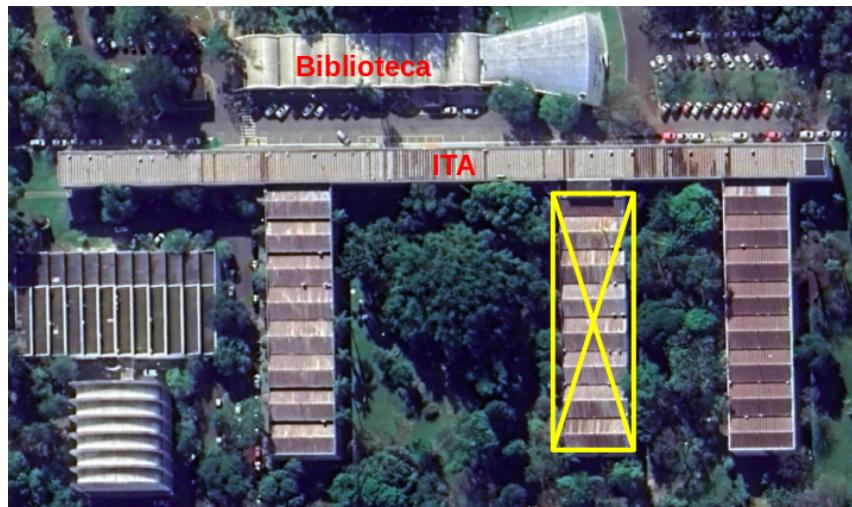


FIGURA 4.42 – Vista de satélite dos prédios e da biblioteca do ITA. O retângulo amarelo evidencia o corredor em frente ao LMI onde foi realizado o experimento com o robô real. Adaptado de Google (2022).

real foi o de dez paredes, e assim foram acrescentadas paredes de madeira ao corredor na altura do sensor LiDAR do robô móvel, conforme pode ser visto na Figura 4.43.



FIGURA 4.43 – Ambiente construído no corredor do LMI por paredes de madeira.

Além disso, observou-se que o sensor *scanner a laser* precisava ser calibrado, pois existia um erro de medida que variava a cada aumento do valor da distância entre o sensor e alguma parede do ambiente. Com isso, o processo de calibração do sensor foi realizado por meio dos seguintes passos.

Primeiramente, o alcance máximo do sensor real era de 12 metros, mas a calibração foi ajustada até 8,5 m. Assim, foram medidas, de forma manual com uma trena, as distâncias reais do sensor do robô em relação à parede de 0,5 m até 8,5 m com passo de 0,5 m, os

dados coletados de distância real e medida pelo sensor foram evidenciados na Tabela 4.9. Para isso, fixou-se o robô e variou-se a distância de uma parede localizada em frente ao sensor. Em seguida, foi registrada a distância medida pelo sensor para cada distância real estabelecida. Diante disso, elaborou-se um gráfico da distância real pela distância medida pelo sensor, conforme mostrado na Figura 4.44 e baseado nos dados da Tabela 4.9.

TABELA 4.9 – Dados coletados da distância real obtida pela trena e da distância medida pelo sensor *scanner a laser*.

Dados para Calibração do Sensor de Distância	
Distância real (m)	Distância medida (m)
0	0
0,5	0,517
1	1,031
1,5	1,555
2	2,072
2,5	2,602
3	3,121
3,5	3,661
4	4,258
4,5	4,77
5	5,294
5,5	5,841
6	6,46
6,5	7,085
7	7,682
7,5	8,334
8	8,93
8,5	9,522

Na Figura 4.44, a linha e os asteriscos vermelhos representam os pontos referentes à distância real e à distância medida pelo sensor, já a linha azul evidencia as distâncias do sensor ideal e a linha preta com asteriscos ilustra o erro de medida do sensor real em relação ao ideal.

Por meio disso, para corrigir as medidas obtidas pelo sensor, fez-se a interpolação linear local entre os pontos das distâncias medidas pelo sensor a fim de se adquirir a distância real entre o sensor e a parede do ambiente. Dessa maneira, desenvolveu-se o gráfico da Figura 4.45 que mostra as distâncias medidas pelo sensor em relação à distância corrigida em metros, na qual a linha vermelha com asteriscos representa a curva da distância medida pelo sensor em relação à distância corrigida pela interpolação e a linha azul representa a curva do sensor ideal.

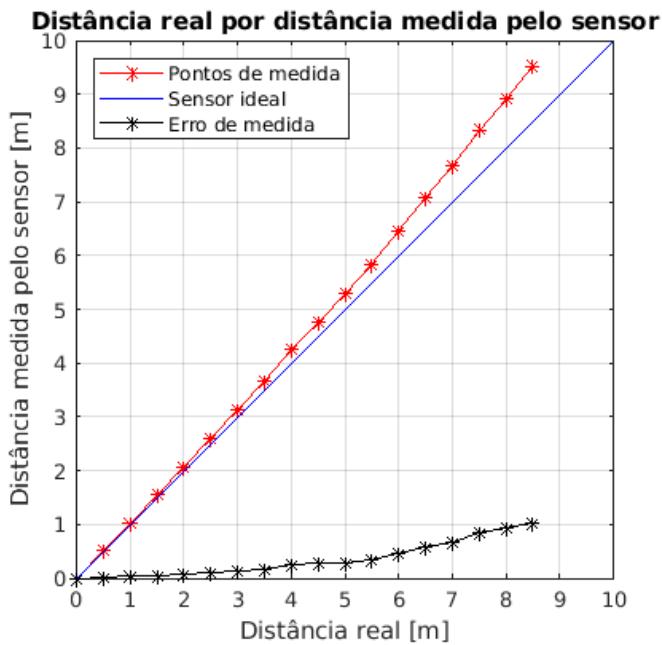


FIGURA 4.44 – Gráfico da distância real em relação à distância medida pelo sensor em metros.

Na Figura 4.45 as distâncias medidas pelo sensor passam a pertencer ao eixo das abscissas e, com esses dados, as distâncias medidas pelo sensor são corrigidas por meio da interpolação linear local a fim de que essas distâncias corrigidas sejam proporcionais às distâncias reais medidas pela trena.

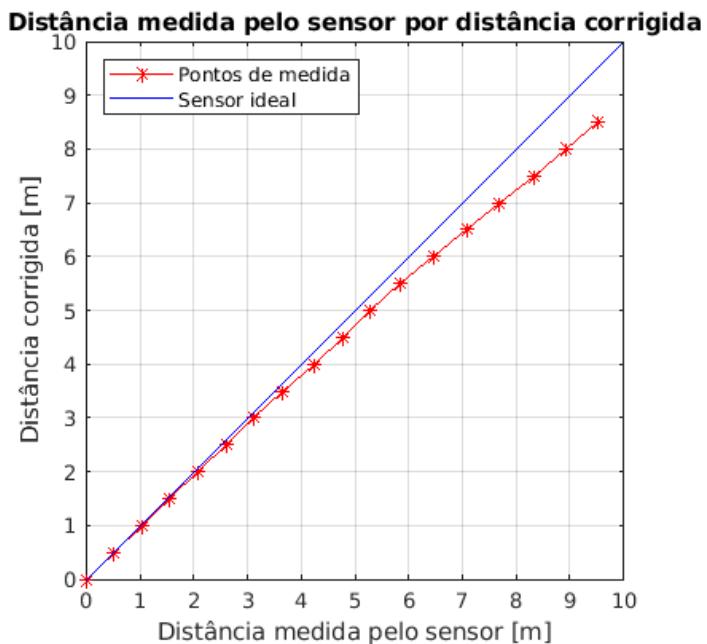


FIGURA 4.45 – Gráfico da distância medida pelo sensor em relação à distância corrigida em metros.

Para se entender de forma mais adequada a etapa de correção dos dados para calibração do sensor *scanner* a *laser*, pode-se observar o diagrama de blocos na Figura 4.46 que foi criado para tornar claro o método proposto de correção das distâncias medidas pelo

sensor para que seja calibrado, assim como o algoritmo evidenciado a seguir. Os dados da Tabela 4.9 são localizados no arquivo `distances_calibration.mat` e foram carregados no algoritmo de correção para calibração do sensor construído no MATLAB.

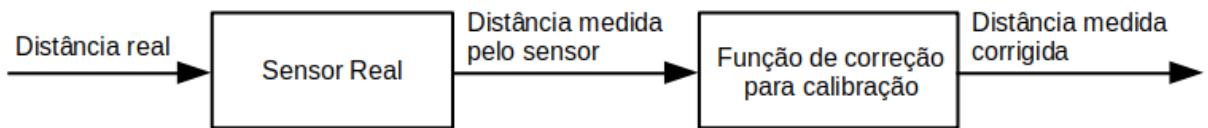


FIGURA 4.46 – Diagrama de blocos que exemplifica o método de correção de dados para calibração do sensor *scanner a laser*.

```

1 % Algoritmo de Correção para Calibração do Sensor
2 % por Interpolação Linear Local
3 clear all; close all;
4 load distances_calibration.mat;
5 % Geração do 1o plot
6 distances=[[0;0] distances];
7 erro=distances(2,:)-distances(1,:);
8 figure(1);
9 plot(distances(1,:),distances(2,:),'*r'); hold on;
10 plot([0 10],[0 10],'b');
11 plot(distances(1,:),erro,'*-k');
12 xlabel('Distância real [m]');
13 ylabel('Distância medida pelo sensor [m]');
14 title('Distância real por distância medida pelo sensor');
15 legend('Pontos de medida','Sensor ideal','Erro de ... medida','Location','northwest');
16 grid on; axis equal; axis([0 10 0 10]);
17 set(gca,'Xtick',[0:10]);
18 % Geração do 2o plot
19 % vq = interp1(x,v,xq)
20 dist=distances(2,:);
21 dist_corrigido=interp1(distances(2,:),distances(1,:),dist);
22 save distances distances
23 figure(2);
24 plot(dist,dist_corrigido,'*r'); hold on;
25 plot([0 10],[0 10],'b');
26 xlabel('Distância medida pelo sensor [m]');
27 ylabel('Distância corrigida [m]');
28 title('Distância medida pelo sensor por distância corrigida');
  
```

```

30 legend( 'Pontos de medida' , 'Sensor ideal' , 'Location' , ...
31 'northwest' );
32 grid on; axis equal; axis([0 10 0 10]);
33 set(gca , 'Xtick' ,[0:10]);
34 figure(3);
35 plot(distances(1,:) , dist_corrigido , '*-b');
36 xlabel('Distância real [m]');
37 ylabel('Distância medida corrigida [m]');
38 title('Calibração do sensor de distância');
39 legend( 'Pontos de medida' , 'Location' , 'northwest' );
40 grid on; axis equal; axis([0 10 0 10]);
41 set(gca , 'Xtick' ,[0:10]);

```

A função principal do algoritmo de correção de dados é a “interp1” que retorna valores interpolados de uma função 1D em pontos especificados usando interpolação linear. O primeiro argumento da função contém os pontos da amostra e o segundo argumento contém os valores correspondentes. E o terceiro argumento contém as coordenadas dos pontos de consulta (MATHWORKS, 2022d).

Dante disso, pode-se construir o gráfico referente à curva calibrada do sensor *scanner a laser*, na qual o eixo das abscissas é representado pelos valores das distâncias reais medidas com a trena e o eixo das ordenadas se refere às distâncias medidas pelo sensor que foram corrigidas, conforme se observa na Figura 4.47. A curva azul obtida foi comparada com a curva do sensor ideal das Figuras 4.44 e 4.45, evidenciando sua adequação com tal curva.

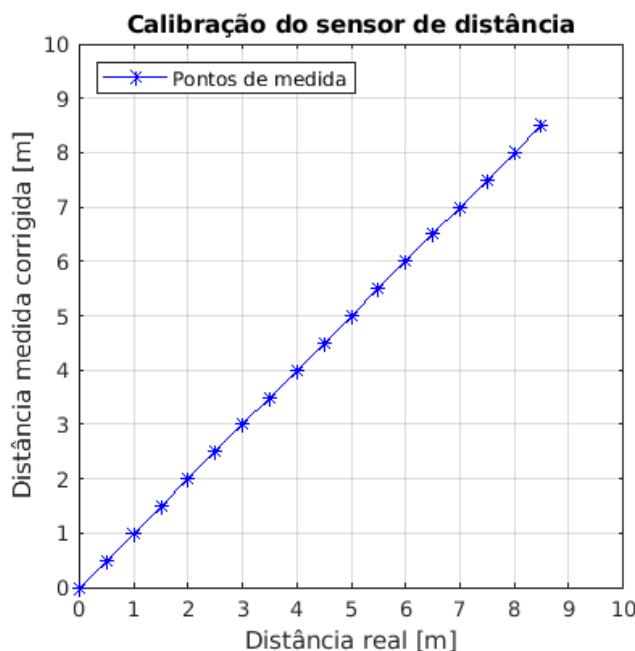


FIGURA 4.47 – Curva calibrada do sensor de distância por meio da interpolação linear local.

Dessa forma, foi feito um experimento no ambiente real, separando-o em dois casos para verificação dos resultados sem e com a técnica de fechamento de laço, sendo ambos com a junção de segmentos.

4.3.1 Caso 1: sem Fechamento de Laço

Inicialmente, realizou-se um primeiro teste no ambiente real, coletando uma quantidade de posturas próxima do número de posturas coletadas nos experimentos simulados. Dessa forma, foram coletadas 42 posturas com o guiamento do robô móvel de forma manual e arbitrária.

No entanto, os segmentos gerados em cada postura com o algoritmo de RANSAC modificado a partir dos dados coletados não foram adequados para obter o melhor par de rotação e translação que realizasse a correspondência de segmentos no algoritmo SLAM. Isso ocorreu por causa do guiamento manual do robô móvel que foi feito de forma arbitrária, o que gerou posturas com uma distância considerável entre elas em certas regiões da trajetória do robô que não permitiram a execução adequada do experimento.

No segundo teste feito, aumentou-se o número de posturas coletadas para que o experimento fosse feito de forma adequada. Com isso, para o experimento ocorrido no ambiente real, foram coletados os dados do *scanner a laser* em 81 posturas do robô de forma arbitrária e para que as posturas fossem mais próximas entre si. Tais informações estiveram no formato de nuvens de pontos e foram tratadas no algoritmo de RANSAC, obtendo-se assim, um conjunto de segmentos de reta ajustados para o *scan* a cada postura. Com isso, foram gerados os segmentos de reta e seus pontos terminais para cada *scan* que são usados no algoritmo de SLAM 2D.

Primeiramente, no algoritmo de SLAM, calculou-se o ângulo de uma parede por meio dos pontos terminais do segmento referente e existente na leitura dos segmentos do primeiro *scan* para definir a postura inicial do robô, em que os valores da posição foram definidos como zero e apenas se fez o cálculo da orientação.

Em seguida, foram calculados os dados de rotação e translação entre os *scans* em posturas consecutivas por meio da otimização da função não linear da transformação de coordenadas. Dessa forma, a melhor solução para cada *scan* em posturas consecutivas foi adquirida a partir da medida de correspondência. A Tabela 4.10 evidencia as rotações e translações calculadas entre os *scans* em posturas consecutivas.

Além disso, os *scans* em posturas consecutivas foram rotacionados e transladados de acordo com as rotações e translações obtidas na Tabela 4.10, representando assim, os segmentos de linha, os pontos terminais e os cantos nos gráficos de posturas consecutivas. Dessa maneira, foram evidenciadas algumas amostras de *scans* em posturas consecutivas

TABELA 4.10 – Rotação (R) em graus e translação (T) em metros obtidas entre *scans* em posturas consecutivas do experimento real.

Scans	R(graus)	T(m)	Scans	R(graus)	T(m)	Scans	R(graus)	T(m)
1/2	-52,721	0,425	28/29	21,136	0,585	55/56	7,987	0,410
2/3	37,588	0,187	29/30	18,555	0,302	56/57	-11,942	0,364
3/4	-79,025	0,471	30/31	6,029	0,323	57/58	-5,480	0,322
4/5	47,335	0,799	31/32	-7,620	0,443	58/59	-0,295	0,383
5/6	-34,323	0,250	32/33	-19,819	0,213	59/60	-0,024	0,337
6/7	19,731	0,227	33/34	28,508	0,439	60/61	-8,058	0,341
7/8	20,907	0,211	34/35	-36,763	0,478	61/62	6,843	0,290
8/9	8,497	0,311	35/36	6,075	0,462	62/63	-5,489	0,289
9/10	-1,554	0,239	36/37	-16,619	0,402	63/64	14,293	0,511
10/11	-27,654	0,257	37/38	4,789	0,324	64/65	-1,277	0,414
11/12	13,854	0,287	38/39	-35,693	0,329	65/66	18,268	0,340
12/13	15,281	0,482	39/40	-5,254	0,463	66/67	-3,100	0,232
13/14	6,120	0,563	40/41	5,125	0,247	67/68	17,316	0,274
14/15	21,922	0,355	41/42	-25,382	0,348	68/69	16,117	0,289
15/16	7,332	0,238	42/43	-26,963	0,272	69/70	25,716	0,383
16/17	-29,953	0,261	43/44	0,535	0,141	70/71	-9,183	0,357
17/18	31,448	0,361	44/45	-37,207	0,475	71/72	-15,087	0,314
18/19	9,798	0,229	45/46	-44,600	0,427	72/73	-27,590	0,416
19/20	1,463	0,205	46/47	-19,188	0,500	73/74	-2,053	0,303
20/21	-47,573	0,218	47/48	-20,803	0,553	74/75	1,806	0,351
21/22	12,042	0,251	48/49	5,729	0,395	75/76	-16,604	0,393
22/23	24,702	0,268	49/50	-26,845	0,185	76/77	-27,602	0,319
23/24	7,188	0,230	50/51	23,382	0,231	77/78	-18,860	0,330
24/25	-40,421	0,292	51/52	1,082	0,473	78/79	-31,703	0,408
25/26	20,419	0,307	52/53	78,105	0,438	79/80	25,277	0,180
26/27	-37,231	0,424	53/54	-22,353	0,514	80/81	27,096	0,329
27/28	4,896	0,506	54/55	-29,435	0,474			

que realizaram o *matching* de linhas.

Os *scans* nas posturas mostrados na Figura 4.48 foram os de números 2, 3, 15, 16, 32, 33, 47, 48, 62, 63, 77 e 78, que estiveram no sistema de coordenadas local das posturas 2, 15, 32, 47, 62 e 77. Na Figura 4.48, as linhas pretas contínuas e os asteriscos pretos representam os segmentos de reta e pontos terminais dos *scans* nas posturas atuais, enquanto que as linhas tracejadas e os círculos vermelhos representam os segmentos e pontos terminais do *scan* na postura posterior que foi rotacionada e transladada para se realizar o *line matching*.

Dante do que foi observado na Figura 4.48, notou-se que o *matching* das linhas ocorreu de forma adequada em alguns casos. No entanto, os casos que não foram adequados geraram um erro que se tornou acumulativo ao longo da trajetória do robô. Com isso, a Figura 4.49 ilustra o mapa global estimado do experimento real que foi construído sem o fechamento do laço, o que gera um erro evidente que necessita ser corrigido.

O mapa global da Figura 4.49 possui as linhas pretas que representam os segmentos de reta do *scan*, a linha vermelha ilustra a trajetória feita pelo robô, em que cada postura é evidenciada por um asterisco vermelho, já o quadrado azul ciano representa a postura inicial e o quadrado azul escuro retrata a postura final do experimento. Além disso, a linha vermelha associada a cada quadrado é a orientação da postura e os círculos verdes são os cantos ou pontos terminais isolados detectados.

Paralelo a isso, os cantos ou pontos terminais isolados foram selecionados de acordo com cada *scan* local existente. O *scan* analisado já salva os cantos localmente. Caso fossem reconhecidos mais de três cantos muito próximos aos já salvos, os cantos salvos eram definidos como marcos para serem usados no processo do fechamento de laço. Com isso, os marcos são detectados no sistema de coordenadas local e depois são transformados para o sistema de coordenadas global juntamente com a postura estimada. Para que isso funcione, o erro no *matching* entre os *scans* deve ser pequeno, pois erros muito grandes invalida o *matching* entre *scans*.

Além disso, observou-se que o mapa global gerado apresenta um erro acumulativo elevado, isso ocorre porque o erro existente no sensor *scanner a laser* real acoplado no robô móvel possui maiores incertezas do que o sensor simulado no Gazebo. Esse erro pode fazer com que os segmentos e pontos terminais gerados já possuam erros com relação às paredes e cantos reais do ambiente.

4.3.2 Caso 2: com Fechamento de Laço

O processo de fechamento de laço no experimento real é essencial para reduzir o erro acumulativo e, por consequência, o erro existente no mapa global. Entretanto, como os

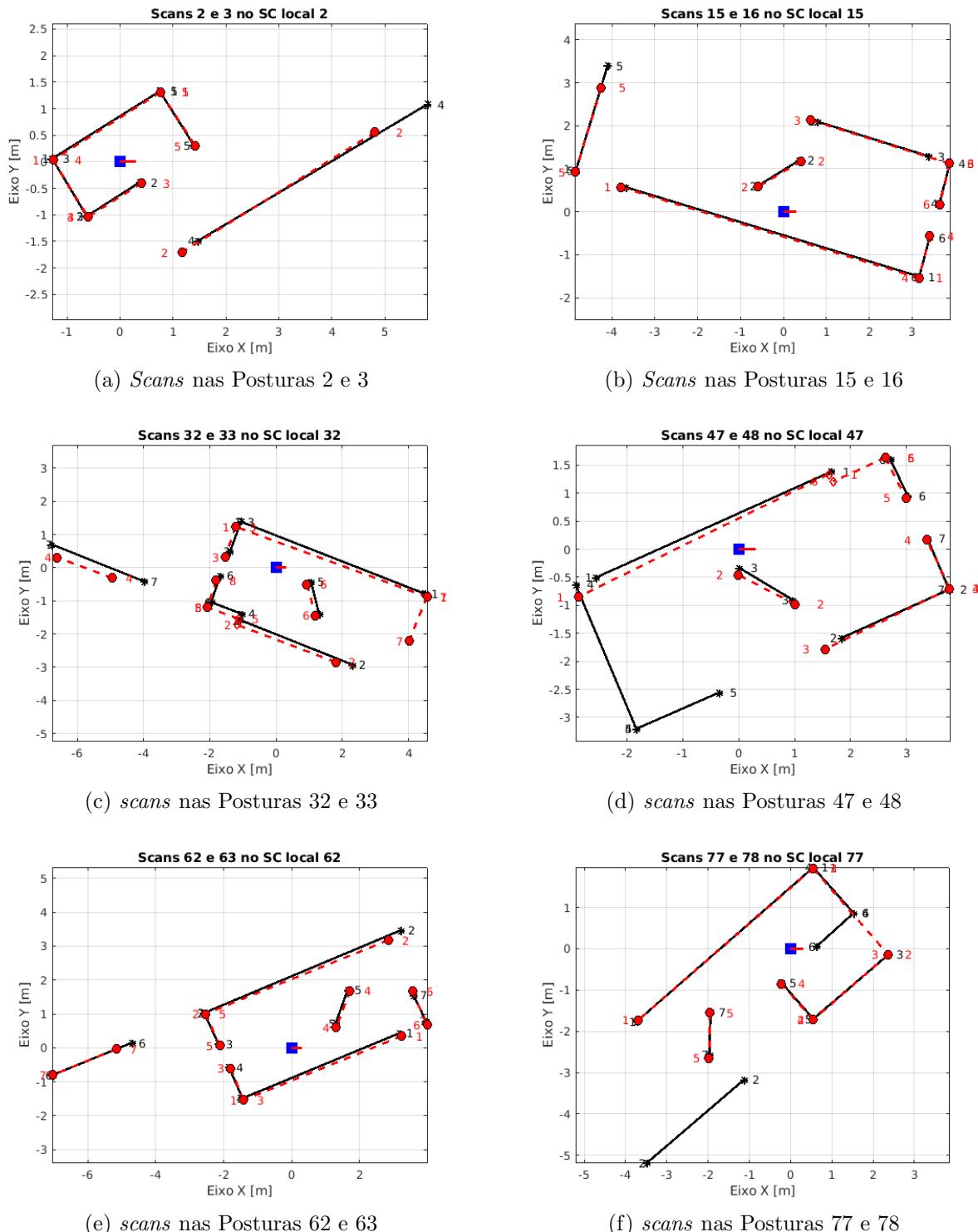


FIGURA 4.48 – Amostras de *Line matchings* sem fechamento do laço entre *scans* em posturas consecutivas no experimento real.

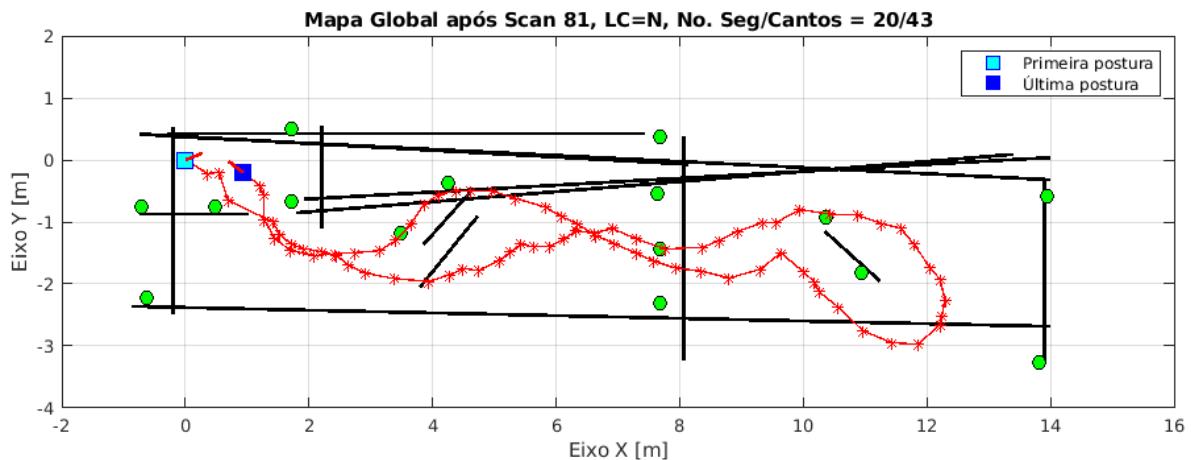


FIGURA 4.49 – Trajetória das posturas estimadas do robô no mapa global do ambiente no sistema de coordenadas da postura inicial do experimento real sem fechamento de laço e com a junção de segmentos.

scans nas posturas coletadas no ambiente real apresentam um erro considerável, a etapa de fechamento de laço já ocorre nas primeiras posturas coletadas.

Com isso, a partir do segundo *scan* se iniciou o fechamento do laço com a detecção dos cantos ou pontos terminais isolados em cada *scan* local nas posturas coletadas que depois são transformados para o sistema de coordenadas global. Dessa forma, os marcos são selecionados ao longo da trajetória do robô. Assim, para existir o fechamento de laço, é preciso que sejam detectados no *scan* local na postura no mínimo três pontos terminais ou cantos.

Em suma, o conjunto de cantos selecionados são armazenados no mapa global e são definidos como marcos. Quando o laço for fechado, esses marcos são transformados para o mapa local do *scan* na postura analisada e são verificados quais cantos ou pontos terminais isolados do *scan* local são mais próximos do conjunto de marcos. Por exemplo, se existissem dez marcos selecionados, mas apenas seis cantos do *scan* local são mais próximos desses marcos, é feita a correspondência dos marcos com os cantos isolados, e assim o fechamento do laço é realizado e a postura estimada é corrigida.

Dante disso, a Figura 4.50 ilustra alguns exemplos da correspondência entre os marcos selecionados ao longo do caminho desenvolvido pelo robô e os cantos ou pontos terminais isolados do *scan* local das posturas 3, 16, 33, 48, 63 e 78. Nas imagens, as linhas pretas são os segmentos de reta, os círculos vermelhos são os cantos ou pontos terminais dos *scans* locais das posturas, os círculos verdes são os marcos detectados no mapa global que são transformados para o sistema de coordenadas local, os quadrados azuis representam a postura local, a linha vermelha é a orientação da postura e os índices dos cantos do *scan* local são numericamente identificados.

Além disso, o número de marcos detectados em cada *scan* de posturas foram registrados no gráfico da Figura 4.51. Com isso, observou-se que os marcos somente eram

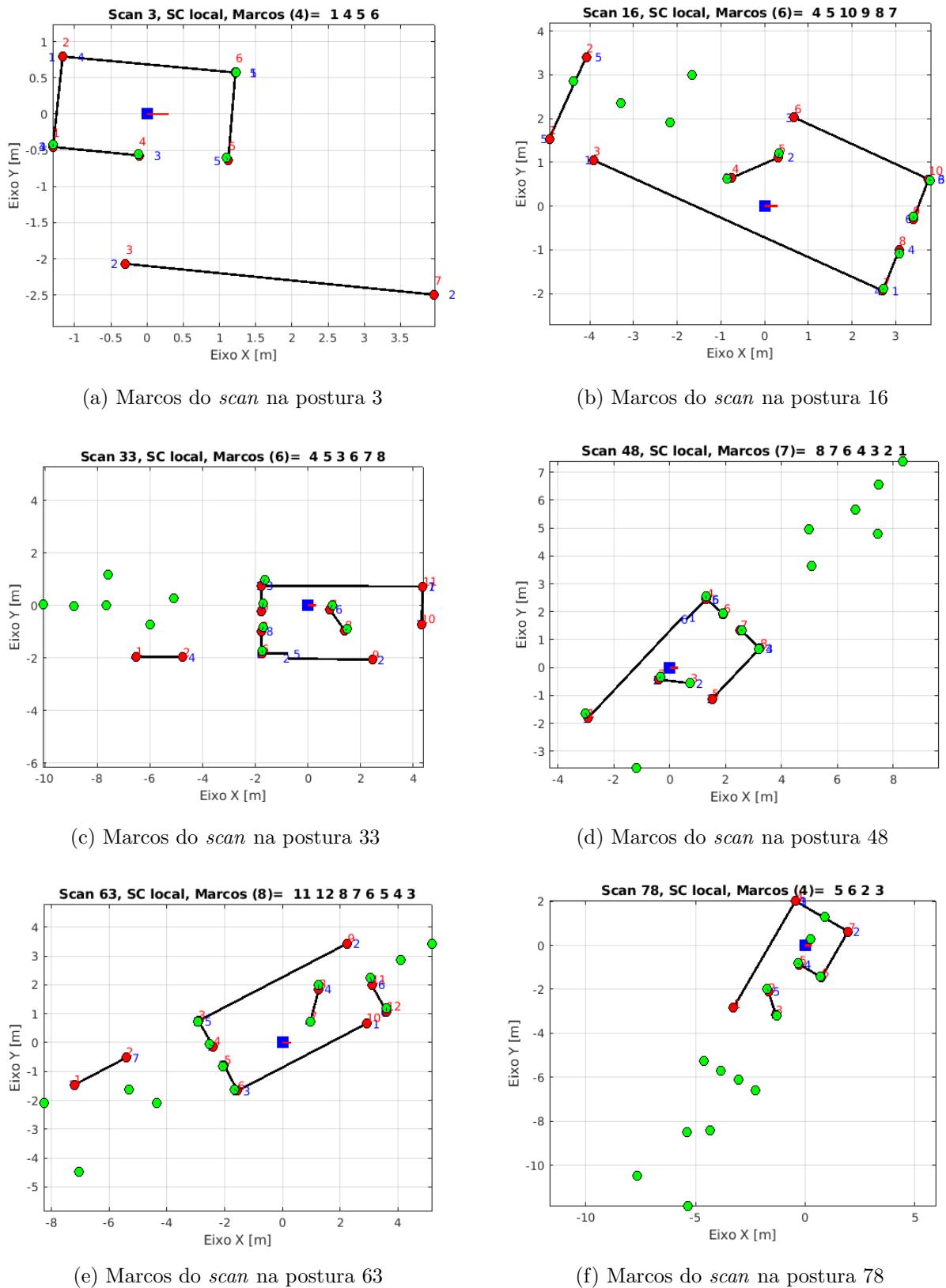


FIGURA 4.50 – Correspondência entre os marcos selecionados e os cantos identificados nas posturas 3, 16, 33, 48, 63 e 78 no sistema de coordenadas local de cada postura durante o fechamento de laço no experimento real.

registrados caso fossem reconhecidos a partir de três cantos no sistema de coordenadas local próximos aos marcos selecionados no sistema global que foram transformados para o sistema de coordenadas local das posturas.



FIGURA 4.51 – Quantidade de marcos detectados em cada *scan* das posturas para fechamento do laço com junção de segmentos.

Paralelo a isso, a IMU embarcada no robô móvel mediu o ângulo em relação ao eixo z em cada postura do robô ao longo do experimento real. Dessa maneira, esses dados foram salvos e comparados com os dados do ângulo de cada postura estimada no algoritmo de SLAM com correspondência de características, conforme observado na Figura 4.52.

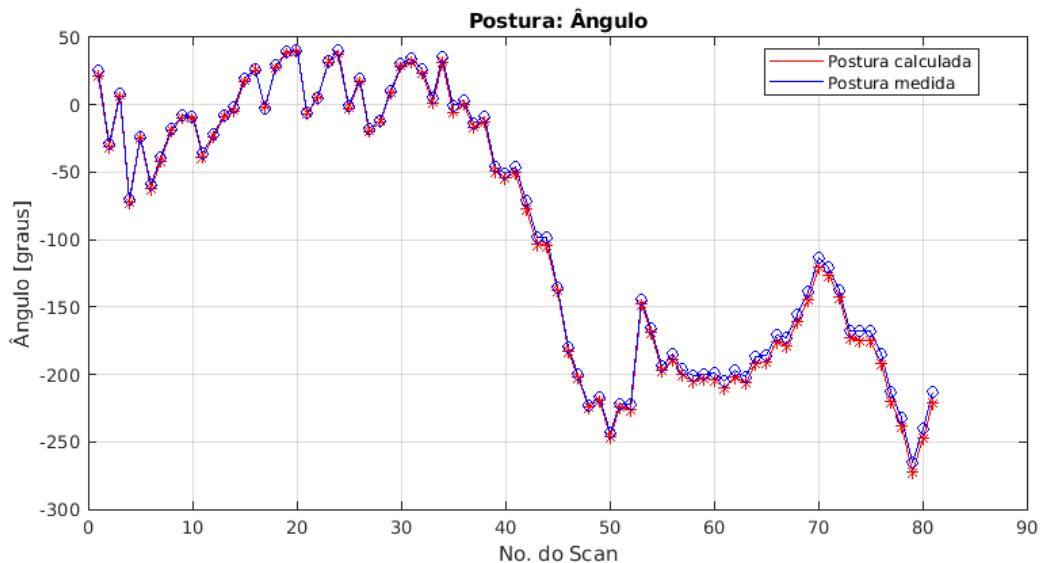


FIGURA 4.52 – Comparação da orientação em graus presente nas posturas reais e estimadas do experimento real com fechamento de laço.

A partir dos segmentos ajustados, das posturas estimadas corrigidas e dos marcos selecionados, foi feito o processo de fechamento de laço e foi construído o mapa global estimado do experimento real, conforme representado na Figura 4.53.

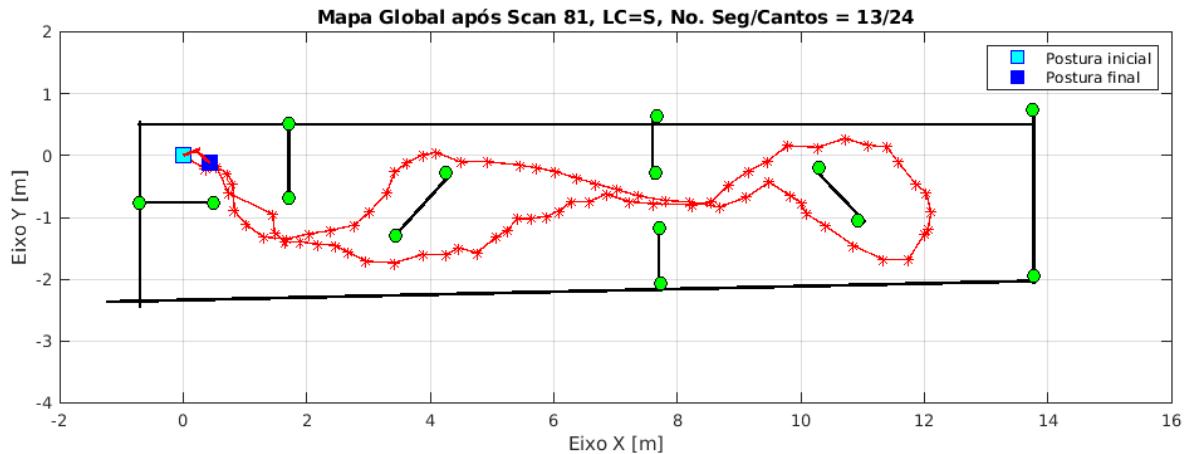


FIGURA 4.53 – Trajetória de posturas estimadas do robô no mapa global do ambiente no sistema de coordenadas da postura inicial do experimento real com fechamento de laço e com a junção de segmentos.

Na Figura 4.53, foram detectados 13 segmentos e 24 cantos, as linhas pretas são os segmentos de reta referentes às paredes do ambiente, a linha e o asterisco vermelhos representam a trajetória das posturas feita pelo robô real, os círculos verdes são os marcos selecionados, o quadrado azul cianon é a postura inicial e o quadrado azul escuro é a postura final do experimento real.

Pode-se perceber no mapa global obtido que o erro existente é muito menor para o caso com o fechamento do laço, pois o erro acumulativo é reduzido, corrigindo as posturas estimadas e os segmentos de reta. Em experimentos reais os sensores embarcados no robô apresentam erro na medição, o que esclarece que o erro ainda evidente no mapa global com fechamento de laço não invalida o resultado adquirido no experimento real.

Dante dos resultados expostos, fica clara a importância do processo de fechamento do laço em problemas que envolvem SLAM 2D, as diferenças nos resultados com e sem a técnica de fechamento de laço são verificadas. Podem existir experimentos que não precisam realizar tal técnica para comprovar o sistema do SLAM, porém os experimentos que a possuem apresentam menores erros, permitindo a convergência dos resultados de forma mais adequada.

Dessa forma, a Figura 4.54 mostra a comparação realizada entre o mapa real e o mapa estimado pelo algoritmo com fechamento de laço. As linhas tracejadas e círculos vermelhos ilustram os segmentos de reta e pontos terminais do mapa do ambiente real e as linhas contínuas pretas representam os segmentos gerados do mapa estimado.

Por meio da Figura 4.54, observou-se que as linhas do mapa estimado foram próximas

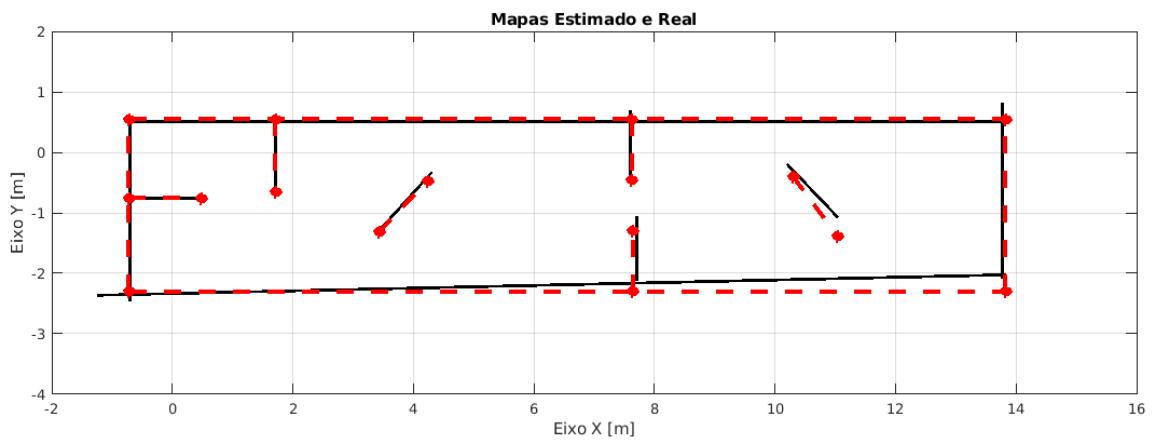


FIGURA 4.54 – Comparaçāo entre o mapa do ambiente real e o mapa estimado do experimento real com fechamento do laço.

das linhas do mapa real no experimento feito com o robô móvel no ambiente real, apresentando um erro reduzido e a convergência existente entre o mapa real e o mapa estimado, o que evidencia a eficiência do processo desenvolvido para SLAM com correspondência de características utilizando a técnica de fechamento de laço.

5 Conclusão

Esse trabalho de dissertação propôs solução para o problema de SLAM 2D com grafo de posturas por meio da correspondência de características para que um robô móvel explore de forma manual um ambiente desconhecido com o uso do sensor embarcado LiDAR necessário para realizar as tarefas de localização e mapeamento com a estimação das posturas do veículo e do mapa do ambiente interno.

Com isso, a solução proposta utilizou segmentos de reta como um tipo de características do ambiente para resolver o problema de localização e mapeamento simultâneos com correspondência de características. Por isso, a simulação é de suma importância para a realização de testes e da validação das hipóteses deliberadas a partir do que foi proposto.

Dante disso, foram produzidos experimentos do sistema de SLAM com o robô simulado e real em ambientes internos estruturados por paredes finas de aproximadamente 44 m² e 41,44 m², respectivamente. Observou-se que nem sempre o fechamento de laço é necessário, porém é preciso realizar o fechamento de laço na maioria dos ambientes, pois é reduzido o erro entre as posturas estimadas do robô.

Além disso, a inclusão de uma parede extra no ambiente simulado foi essencial para mostrar que, em posições aproximadas do robô no ambiente, existem regiões que não são reconhecidas pelo sensor *scanner a laser* no ambiente com a parede extra, assim como posturas consecutivas do robô muito distantes entre si quando coletadas, que podem desestruturar o mapa do ambiente estimado, sendo de suma importância o fechamento do laço.

A técnica de junção de segmentos proposta foi essencial para diminuir a quantidade de segmentos e de pontos terminais do mapa global estimado, assim como para corrigilos, tornando o mapa estimado mais próximo do mapa original referente ao ambiente analisado.

Por meio disso, os resultados obtidos pelo sistema desenvolvido de SLAM foram de acordo com o previsto, em que as rotações e translações adquiridas para associar as características dos *scans* em cada postura foram adequadas para a realização da correspondência entre os segmentos de reta e a construção do mapa global estimado. Apesar da existência de erros entre algumas posturas, o objetivo deste trabalho foi cumprido de

forma apropriada.

Como trabalhos futuros, algumas alternativas são sugeridas:

- Resolver o problema de exploração autônoma utilizando SLAM com correspondências de características;
- Estudar e implementar soluções para que o robô extraia outros objetos presentes no ambiente, além de paredes, caixas, linhas e *endpoints*, como cilindros com eixo principal na vertical e objetos retangulares que possuam cantos arredondados, por exemplo, a fim de tornarem o ambiente mais detalhado e dinâmico;
- Realizar a etapa de mapeamento do ambiente por meio de sensores que obtenham medidas em 3D, como câmeras de profundidade, para obter uma representação 3D do mapa e aplicar SLAM 3D com correspondência de características;
- Fazer um estudo mais aprofundado da comparação entre os diferentes tipos de SLAM feitos com as técnicas de *scan matching* e *feature matching*;
- Realizar testes em ambientes mais complexos, tanto interno quanto externo;
- Investigar como melhorar o guiamento do robô para permitir solução do problema SLAM (distância/rotação mínima entre as paradas);
- Melhorar critérios para escolher os *scans* onde deve ser feito o fechamento de laço e a seleção dos marcos para o fechamento de laço.

Referências

- AGHAMOHAMMADI, A. A.; TAGHIRAD, H. D.; TAMJIDI, A. H.; MIHANKHAH, E. Feature-based laser scan matching for accurate and high speed mobile robot localization. *In: EMCR. Proceedings [...]. [S.l.: s.n.], 2007.*
- ALMEIDA, H. P. **Navegação Autônoma de um Veículo Terrestre em Escala Reduzida Usando GPS/INS de Baixo Custo.** 2017. 111 f. Dissertation (Mestrado) — Instituto Tecnológico de Aeronáutica, São José dos Campos, 2017.
- BAILEY, T.; DURRANT-WHYTE, H. Simultaneous localization and mapping (slam): Part ii. **IEEE robotics & automation magazine**, IEEE, v. 13, n. 3, p. 108–117, 2006.
- BIBER, P.; STRASSER, W. The normal distributions transform: A new approach to laser scan matching. *In: IEEE. Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453).* **Proceedings [...]. [S.l.: s.n.], 2003.** v. 3, p. 2743–2748.
- BOLLES, R. C.; FISCHLER, M. A. A ransac-based approach to model fitting and its application to finding cylinders in range data. *In: IJCAI. Proceedings [...]. [S.l.: s.n.], 1981.* v. 1981, p. 637–643.
- BUONOCORE, L. **SLAM EM AMBIENTES INTERNOS UTILIZANDO ROBÔ DE BAIXO CUSTO.** 2013. 275 f. Thesis (Doutorado) — Instituto Tecnológico de Aeronáutica, São José dos Campos, 2013.
- FILHO, L. E. S. A. **Multi-robot autonomous exploration and map merging in unknown environments.** 2021. 98 f. Dissertation (Mestrado) — Instituto Tecnológico de Aeronáutica, São José dos Campos, 2021.
- GAO, H.; ZHANG, X.; WEN, J.; YUAN, J.; FANG, Y. Autonomous indoor exploration via polygon map construction and graph-based slam using directional endpoint features. **IEEE Transactions on Automation Science and Engineering**, IEEE, v. 16, n. 4, p. 1531–1542, 2018.
- GAZEBO. **Gazebo - Robot simulation made easy.** 2022. Disponível em: <https://gazebosim.org/>, Acesso em: 01/11/2022.
- GOOGLE. **Google earth.** 2022. Disponível em: <https://www.google.com.br/intl/pt-BR/earth/>, Acesso em: 14/11/2022.

- GRISSETTI, G.; KÜMMERLE, R.; STACHNISS, C.; BURGARD, W. A tutorial on graph-based slam. **IEEE Intelligent Transportation Systems Magazine**, IEEE, v. 2, n. 4, p. 31–43, 2010.
- KUMAR, A. **An Introduction to Simultaneous Localization and Mapping (SLAM) for Robots**. 2020. Disponível em: <https://control.com/technical-articles/an-introduction-to-simultaneous-localization-and-mapping-slam-for-robots/>. Acesso em : 18/01/2020.
- LAGARIAS, J. C.; REEDS, J. A.; WRIGHT, M. H.; WRIGHT, P. E. Convergence properties of the nelder–mead simplex method in low dimensions. **SIAM Journal on optimization**, SIAM, v. 9, n. 1, p. 112–147, 1998.
- LI, J.; ZHONG, R.; HU, Q.; AI, M. Feature-based laser scan matching and its application for indoor mapping. **Sensors**, MDPI, v. 16, n. 8, p. 1265, 2016.
- LU, F.; MILIOS, E. Robot pose estimation in unknown environments by matching 2d range scans. **Journal of Intelligent and Robotic systems**, Springer, v. 18, n. 3, p. 249–275, 1997.
- MATHWORKS. **Create a 2-D pose graph - MATLAB**. 2022. Disponível em: <https://www.mathworks.com/help/nav/ref/posegraph.html>. Acesso em: 21/10/2022.
- MATHWORKS. **Exchange Data with ROS Publishers and Subscribers - MATLAB**. 2022. Disponível em: <https://la.mathworks.com/help//ros/ug/exchange-data-with-ros-publishers-and-subscribers.html>. Acesso em: 31/10/2022.
- MATHWORKS. **MATLAB - fminsearch**. 2022. Disponível em: <https://www.mathworks.com/help/matlab/ref/fminsearch.html>. Acesso em: 31/10/2022.
- MATHWORKS. **MATLAB - interp1**. 2022. Disponível em: <https://www.mathworks.com/help/matlab/ref/interp1.html>. Acesso em: 19/01/2023.
- MONTEMERLO, M. **FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association**. Thesis (Doutorado) — Carnegie Mellon University, Pittsburgh, PA, July 2003.
- NEWMAN, P.; HO, K. Slam-loop closing with visually salient features. In: **IEEE. proceedings of the 2005 IEEE International Conference on Robotics and Automation. Proceedings [...]**. [S.l.: s.n.], 2005. p. 635–642.
- NIETO, J.; BAILEY, T.; NEBOT, E. Recursive scan-matching slam. **Robotics and Autonomous systems**, Elsevier, v. 55, n. 1, p. 39–49, 2007.
- PAETH, A. **Graphics Gems V**. AP Professional, 1995. (Graphics gems series : a collection of practical techniques for the computer graphics programmer). ISBN 9780125434553. Available at: <https://books.google.com.br/books?id=JhNEAQAAIAAJ>.

- PINTO, L. de S. **Exploração Autônoma de Ambientes Internos Utilizando EKF-SLAM**. 2020. 101 f. Dissertation (Mestrado) — Instituto Tecnológico de Aeronáutica, São José dos Campos, 2020.
- PUENTE, P. de la; RODRIGUEZ-LOSADA, D. Feature based graph slam with high level representation using rectangles. **Robotics and Autonomous Systems**, Elsevier, v. 63, p. 80–88, 2015.
- QUIGLEY, M.; CONLEY, K.; GERKEY, B.; FAUST, J.; FOOTE, T.; LEIBS, J.; WHEELER, R.; NG, A. Y. *et al.* Ros: an open-source robot operating system. In: KOBE, JAPAN. **ICRA workshop on open source software. Proceedings** [...]. [S.l.: s.n.], 2009. v. 3, n. 3.2, p. 5.
- RAYNER, R. **Natural Navigation Explained: Scan Matching vs. Feature Matching**. 2022. Disponível em: https://www.robotics247.com/article/natural_navigation_explained_scan_matching_vs_feature_matching. Online. Acesso em: 05/11/2022.
- ROS - Stanford Artificial Intelligence Laboratory et al. **Robotic Operating System**. 2020. Disponível em: <https://www.ros.org>. Acesso em: 24/10/2022.
- SHANGHAI SLAMTEC CO., LTD. **RPLiDAR A2M8 - Datasheet**. [S.l.], 2017. Available at:
https://bucket-download.slamtec.com/20b2e974dd7c381e46c78db374772e31ea74370d-LD208_SLAMTEC_rplidar_datasheet_A2M8_v2.6_en.pdf.
- SOLÀ, J. Course on slam. In: . **Proceedings** [...]. [S.l.: s.n.], 2017.
- STACHNISS, C. **Exploration And Mapping With Mobile Robots**. Thesis (Doutorado) — University of Freiburg, Department of Computer Science, 2006.
- THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic robotics (Intelligent Robotics and Autonomous Agents)**. Cambridge, Mass.: MIT Press, 2005. ISBN 0262201623.
- WITMOTION SHENZHEN CO., LTD. **IMU BWT901CL - Datasheet**. [S.l.], 2020. Available at: <https://www.wit-motion.com/9-axis/witmotion-bluetooth-2-0-mult.html>.
- WITMOTION SHENZHEN CO., LTD. **IMU BWT901CL - Manual**. [S.l.], 2020. Available at:
<https://drive.google.com/file/d/18bScCGO5vVZYcEeNKjXNtjnT8OVlrHGI/view>.
- YAMASHINA, K.; KIMURA, H.; OHKAWA, T.; OOTSU, K.; YOKOTA, T. crecomp: Automated design tool for ros-compliant fpga component. In: IEEE. **2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC). Proceedings** [...]. [S.l.: s.n.], 2016. p. 138–145.
- ZULIANI, M. **RANSAC For Dummies With Examples**. 2008. Disponível em: https://www.researchgate.net/publication/277285801_RANSAC_for_Dummies_-_With_examples/citation/download. Acesso em: 21/10/2022.

Apêndice A - Pseudocódigo do Algoritmo de SLAM

A.1 Passos principais do Algoritmo de SLAM 2D com *Line Matching* e *Loop Closure*

O pseudocódigo desenvolvido foi dividido em partes e descrito em passos para facilitar o entendimento do algoritmo. Os comentários são representados por %. Sem os passos 13 e 16, o pseudocódigo se refere ao algoritmo sem o fechamento do laço e, com todos os passos, utiliza o fechamento de laço.

A.1.1 Algoritmo de SLAM 2D Geral

```
% Inicialização e 1a leitura do Scanner
1. Colocar robô na Pose_Real(1)
2. Ler Scanner a Laser: Scan(1)
3. Extrair as linhas de Scan(1) no Mapa_Local(1): Linhas_Scan(1)
% Mapa_Ambiente, Pose_Real, Pose_Estimada são definidos no SC Global
4. Mapa_Ambiente = Linhas_Scan(1)
5. Pose_Estimada(1) = Pose_Real(1)
% Movimentação do Robô e leitura do Scanner
6. Loop k=2:NScan
7. Movimentar robô para Pose_Real(k)
8. Ler Scanner a Laser: Scan(k)
% Mapa_Local(k) é definido no SC Local
9. Extrair as linhas de Scan(k) no Mapa_Local(k): Linhas_Scan(k)
10. Calcular “Translação + Rotação” entre as Poses do robô usando Line Matching
```

- entre Linhas_Scan(k-1) e Linhas_Scan(k)
11. Calcular Pose_Estimada(k) usando “Translação + Rotação”
 12. Recalcular posição de Linhas_Scan(k) no SC Global: Linhas_Scan_SCG(k)
 13. Caso Scan(k) tenha sido selecionado para *Loop Closure*: executar rotina de *Loop Closure*
 14. Adicionar as linhas em Linhas_Scan_SCG(k) à variável Mapa_Ambiente
 15. Executar em Mapa_Ambiente a rotina de junção de segmentos
 16. Caso Scan(k) tenha sido selecionado para “Registro de Marcos”: executar rotina de “Registro de Marcos Globais para LC”
 17. End Loop k

A.1.2 Rotina de *Loop Closure* (LC)

% Marcos = cantos detectados quando segmentos de linha possuem pontos terminais suficientemente próximos e pontos terminais de segmentos de linha que estão isolados

1. Determinar a posição dos marcos no SC Local a serem usados para LC usando Linhas_Scan(k): Marcos_Locais(k)
2. Recalcular a posição dos marcos globais (Marcos_Globais) para o SC Local: Marcos_Globais_SC_Local(k)
3. Para cada marco em Marcos_Locais(k) detectar o marco em Marcos_Globais_SC_Local mais próximo: Marcos_LC(k)
4. Calcular “Translação_LC + Rotação_LC” que minimiza a soma das distâncias entre Marcos_Locais(k) e Marcos_LC(k)
5. Recalcular Pose_Estimada(k) usando “Translação_LC + Rotação_LC”
6. Recalcular Linhas_Scan_SCG(k) usando “Translação_LC + Rotação_LC”

A.1.3 Rotina de Registro de Marcos Globais para LC

1. Determinar a posição dos marcos no SC Local a serem usados para LC usando Linhas_Scan(k): Marcos_Locais(k)
2. Recalcular a posição dos marcos locais (Marcos_Globais) para o SC Global: Marcos_Locais_SC_Global(k)
3. Adicionar os marcos Marcos_Locais_SC_Global(k) à variável Marcos_Globais

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO DM	2. DATA 25 de janeiro de 2023	3. DOCUMENTO Nº DCTA/ITA/DM-161/2022	4. Nº DE PÁGINAS 130
5. TÍTULO E SUBTÍTULO: Localização e mapeamento simultâneos usando <i>Scanner a Laser</i> e correspondência entre as características do ambiente			
6. AUTORA(ES): Luciana Araujo Lemos			
7. INSTITUIÇÃO(ÓES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÓES): Instituto Tecnológico de Aeronáutica – ITA			
8. PALAVRAS-CHAVE SUGERIDAS PELA AUTORA: Dinâmica de robôs; Robótica; Controle			
9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO: Dinâmica de robôs; Localização e mapeamento simultâneos; Navegação autônoma; Exploração; Algoritmos; Robótica; Controle			
10. APRESENTAÇÃO: <input checked="" type="checkbox"/> Nacional <input type="checkbox"/> Internacional ITA, São José dos Campos. Curso de Mestrado. Programa de Pós-Graduação em Engenharia Eletrônica e Computação. Área de Sistemas e Controle. Orientador: Prof. Dr. Cairo Lúcio Nascimento Júnior. Defesa em 14/12/2022. Publicada em 2022.			
11. RESUMO: <p>Na robótica móvel, um dos problemas mais relevantes é o de SLAM (<i>Simultaneous Localization And Mapping</i>) cuja solução visa estimar a postura (posição e orientação) do robô e mapear o ambiente em que o robô se movimenta de forma simultânea. Este trabalho de dissertação propõe uma solução para o problema de localização e mapeamento simultâneos em duas dimensões usando a correspondência entre certas características do ambiente à medida que o robô navega por ele. Essa técnica é conhecida como <i>feature matching</i> e neste trabalho é usada a correspondência entre os segmentos de reta do ambiente.</p> <p>No algoritmo de SLAM proposto, o robô navega pelo ambiente com guiamento manual e captura em pontos selecionados um <i>scan</i> local, ou seja, o conjunto de pontos gerados pela varredura de 360° do medidor de distância. O algoritmo RANSAC usa esse conjunto de pontos para extrair os segmentos de reta que representam os trechos das paredes do ambiente que são detectados pelo robô nessa parada. Para cada par de <i>scans</i> consecutivos é feita uma busca numérica, usando um algoritmo de minimização, pelo par {rotação, translação} que gera a melhor correspondência entre os segmentos de reta desses <i>scans</i>. Esse par {rotação, translação} é usado para o cálculo da postura do robô no Sistema de Coordenadas Global. É também proposto um algoritmo para junção dos segmentos inseridos no mapa global durante o procedimento de SLAM. Para fechamento de laço (<i>loop closure</i>), nos <i>scans</i> locais são detectados os cantos e as extremidades isoladas das novas paredes. As coordenadas (no Sistema de Coordenadas Global) desses novos marcos são armazenadas. Quando esses marcos são detectados novamente nos próximos <i>scans</i>, as medidas de distância e ângulo do robô para esses marcos são então usadas para recalcular a postura do robô. Para implementação e testes da solução proposta, foi usado um robô móvel equipado com um <i>scanner a laser</i> 2D e um computador embarcado Raspberry Pi 3B plus. São apresentados, primeiramente, experimentos simulados no <i>software Gazebo</i> do robô em um ambiente de 44 m² com 9 e 10 paredes. É também apresentado um experimento usando o robô real em um ambiente com 10 paredes de 41,44 m². Nesses experimentos são apresentados resultados sem e com fechamento de laço. Esses resultados demonstram o bom desempenho do algoritmo proposto nesses ambientes.</p>			
12. GRAU DE SIGILO: <input checked="" type="checkbox"/> OSTENSIVO <input type="checkbox"/> RESERVADO <input type="checkbox"/> SECRETO			