

UNIVERSIDADE FEDERAL DO MARANHÃO  
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIAS  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA  
GRADUAÇÃO EM ENGENHARIA ELÉTRICA

LUIZ EUGÊNIO SANTOS ARAÚJO FILHO

**LOCALIZAÇÃO EM AMBIENTES INTERNOS E ESTÁTICOS UTILIZANDO  
FILTRO MONTE CARLO**

SÃO LUÍS

2018

LUIZ EUGÊNIO SANTOS ARAÚJO FILHO

**LOCALIZAÇÃO EM AMBIENTES INTERNOS E ESTÁTICOS UTILIZANDO  
FILTRO MONTE CARLO**

Monografia apresentada ao Curso de Engenharia Elétrica da Universidade Federal do Maranhão como requisito para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Luciano Buonocore.

São Luís

2018

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

Araújo Filho, Luiz Eugênio Santos.

Localização em ambientes internos e estáticos  
utilizando filtro Monte Carlo / Luiz Eugênio Santos Araújo  
Filho. - 2018.

133 f.

Orientador(a): Luciano Buonocore.

Monografia (Graduação) - Curso de Engenharia Elétrica,  
Universidade Federal do Maranhão, Centro de Ciências  
Exatas e Tecnologia, 2018.

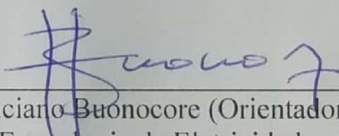
1. Filtro de partículas. 2. Localização global. 3.  
Robótica móvel. I. Buonocore, Luciano. II. Título.

LUIZ EUGÊNIO SANTOS ARAÚJO FILHO

**LOCALIZAÇÃO EM AMBIENTES INTERNOS E ESTÁTICOS UTILIZANDO  
FILTRO MONTE CARLO**

Aprovada em 13/04/2018.

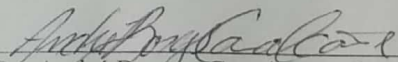
BANCA EXAMINADORA



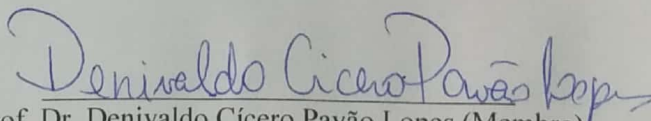
Prof. Dr. Luciano Buonocore (Orientador)  
Departamento de Engenharia de Eletricidade - UFMA



Prof. Dr. Areolino de Almeida Neto (Membro)  
Departamento de Informática - UFMA



Prof. Dr. André Borges Cavalcante (Membro)  
Departamento de Engenharia de Eletricidade - UFMA



Prof. Dr. Denivaldo Cícero Pavão Lopes (Membro)  
Departamento de Engenharia de Eletricidade - UFMA

## **Agradecimentos**

Agradeço aos meus familiares, pelo apoio e amparo nos momentos mais difíceis, além da compreensão e amor incondicional.

Ao professor Luciano Buonocore, pela orientação, força de vontade e disposição, atitudes que foram indispensáveis para conclusão da minha formação profissional concluída com este trabalho. Agradeço imensamente pelo excelente profissional e pessoa que tive a honra de trabalhar desde início, muito obrigado.

A todos os amigos do LRC, pela grande ajuda e conselhos. Agradeço a todos pelo tempo dedicado às minhas dúvidas, em especial aos colegas de curso Ricardo Vinícios, Daniel, João e Walber.

Aos colegas e professores do curso de Engenharia Elétrica, por participarem dessa jornada árdua e longa, porém, recompensadora. Agradeço a todos pelos ensinamentos e lições.

## RESUMO

A robótica é uma área de pesquisa dividida em plataformas com bases fixas (manipuladores), móveis ou híbridas (móvel com capacidade de manipulação de objetos) e tem ocupado cada vez mais espaços nos diversos setores, seja em indústrias, no comércio e em serviços. Este trabalho abordou o problema da localização global de um robô de baixo custo em um ambiente interno e estático de pequenas dimensões, sendo ambos (robô e ambiente) projetados para esta pesquisa. A solução utilizou a abordagem probabilística para estimar a pose global do robô no ambiente, onde o mapa é formado por cilindros de mesmo tamanho e diâmetro. Na implementação da solução foi utilizado o filtro de partículas Monte Carlo (MCL) que fez uso da associação de dados desconhecida na fase de incorporação das medidas efetuadas no ambiente. Foi empregada uma técnica de triangulação de objetos antes de iniciar a ação do MCL, de modo a restringir o espaço de amostragem das partículas ou hipóteses de poses no início, com o objetivo de obter uma convergência do filtro com menos poses de navegação do robô. Um sistema de câmera externa ao sistema robótico, posicionado acima do ambiente (tablado de madeira de 278 x 143 cm<sup>2</sup>), foi implementado com finalidade de facilitar a avaliação dos erros entre as poses estimadas e reais. Além disso, esse sistema óptico foi utilizado para a geração automática do mapa para fornecer com entrada para o filtro. Foram feitos testes do filtro em três experimentos, de maneira a avaliar a sua convergência com relação ao grau de simetrias dos mapas que definiram os ambientes. No primeiro teste o ambiente apresentou baixo grau de simetria e teve o objetivo de observar a quantidade de poses em que o filtro conseguiu fazer a convergência. Nos dois outros experimentos, feitos em ambientes com alto grau de simetria, o uso da técnica de triangulação mostrou-se bastante eficaz, uma vez que manteve o mesmo resultado que o anterior (converge em poucas poses iniciais), mesmo tendo mais áreas onde foram amostradas as partículas. Os erros nas coordenadas x e y das poses estimadas em relação as reais foram inferiores a 1,75 cm e 0,48 cm, respectivamente, e em orientação não ultrapassando 7,2°. Esses resultados são considerados satisfatórios sob dois importantes aspectos: o filtro converge em poucas poses iniciais e os erros são considerados satisfatórios para a finalidade de navegação do robô nas poses sequenciais. Eles mostram que o robô pode ser usado para realizar trabalhos nos ambientes, mesmo com alto grau de simetria.

**Palavras-chave:** Robótica Móvel; Localização global; Filtro de partículas; Associação de dados desconhecida; Técnica de triangulação de objetos.

## Abstract

Robotics in general, divided into platforms with fixed bases (manipulators), mobile or hybrid (mobile with ability to manipulate objects) has increasingly occupied spaces in various sectors, be it in industries, commerce and services. This work addresses the problem of the global location of a low cost robot in a small internal environment, both of which (robot and environment) are designed for this research. The solution used the probabilistic approach to estimate the overall robot pose in the environment, where the map, formed by cylinders of the same size and diameter, is known a priori. In the implementation of the solution, a particle filter was used that made use of the data association (correspondences) unknown in the incorporation phase of the measurements made in the environment. A technique of triangulation of objects was used before starting the action of the filter, in order to restrict the space of sampling of the particles or hypothesis of poses in the beginning, with the objective of obtaining a convergence of the filter with less poses of navigation of the robot. A camera system external to the robotic system, positioned above the environment (wooden board of 278 x 143 cm<sup>2</sup>), was implemented in order to facilitate the evaluation of the errors between the estimated and actual poses. In addition, this optical system was used for the automatic generation of the map to provide with input to the filter. The filter was tested in three experiments, in order to evaluate its convergence with the degree of symmetry of the maps that defined the environment. In the first test the environment presented low degree of symmetry and had the objective of observing the number of poses in which the filter was able to make the convergence. In the other two experiments, in environments with a high degree of symmetry, the use of the triangulation technique proved to be quite effective, since it maintained the same result as the previous one (converges in few initial poses), even though there were more areas where they were particles are sampled. The percentage errors of the poses in their coordinate components lie at the most at 0.43%, 0.21% and 3.4% in the components of the poses, in this case the two coordinates and orientation, respectively. These results are considered satisfactory fewer than two important aspects: the filter converges in few initial poses and the errors are considered satisfactory for the purpose of navigation of the robot in the sequential poses. These results show that the robot can be used to perform work in environments, even with a high degree of symmetry.

**KEYWORDS:** Mobile Robotics; Global localization; Particles filter; Unknown data association; Triangulation technique.

## Lista de figuras

Figura 1	– Tarefas comuns em robótica móvel.	19
Figura 2	– Os quatro tipos básicos de rodas.	24
Figura 3	– Exemplo de mapa contínuo ou de recursos. (a) Mapa real. (b) Representação como um conjunto de linhas infinitas.	27
Figura 4	– Representação de um mapa por entidades geométricas.	28
Figura 5	– Representação de um ambiente em grade de ocupação.	28
Figura 6	– Pose do robô em um sistema de coordenadas global.	30
Figura 7	– Robô móvel em tarefa de localização global.	33
Figura 8	– Robô móvel em tarefa de localização global através do filtro de partículas (Monte Carlo).	37
Figura 9	– Rede Dinâmica de Bayes.	38
Figura 10	– Decomposição do modelo odométrico em três movimentos.	40
Figura 11	– Aproximação por amostras da pose de um robô sem incorporação de medidas.	41
Figura 12	– Câmeras utilizadas em robótica móvel.	46
Figura 13	– Sensores infravermelhos de distância.	46
Figura 14	– Sonar comumente utilizado em robótica móvel.	47
Figura 15	– Sonar mostrando o cone acústico na medição da distância.	48
Figura 16	– Arquitetura de fusão sensorial em três níveis.	50
Figura 17	– Robôs autônomos fazendo organização de mercadorias no armazém da empresa Amazon.	52
Figura 18	– Robô Curiosity enviado pela NASA para exploração em Marte.	52
Figura 19	– Plataforma robótica WOR-R1 junto com o kit RD02.	54
Figura 20	– Dimensões do chassi.	55
Figura 21	– Torre de sensores: (a) após a construção (b) projeto contendo as dimensões. Dimensões fora de escala.	56
Figura 22	– Imagem contida na <i>tag</i> para identificação da pose do robô.	57
Figura 23	– Raspberry Pi 3 modelo B e seus periféricos.	59
Figura 24	– Placa de controle MD25 destacando barramentos de controle e conectores para os motores EMG30.	59
Figura 25	– <i>Layout</i> desenvolvido no Eagle CAD da Placa de Interface.	60
Figura 26	– Esquemático do circuito presente na Placa de Interface.	61
Figura 27	– Diagrama em blocos da Placa de Interface.	63
Figura 28	– Sonar SRF08 destacando seu circuito.	63
Figura 29	– Diagrama de irradiação para o SRF08.	64



Figura 30	– Sensor infravermelho da Sharp.	65
Figura 31	– Gráfico da distância ao objeto pela tensão na saída do sensor infravermelho.	65
Figura 32	– <i>Encoder</i> de efeito <i>Hall</i> .	66
Figura 33	– <i>Bumper switch</i> .	67
Figura 34	– Motores EMG30.	67
Figura 35	– Servo motor S3003.	68
Figura 36	– Baterias de LiPo utilizadas nos experimentos com o robô.	69
Figura 37	– Diagrama em blocos simplificado de intercomunicação dos elementos do sistema robótico.	70
Figura 38	– Disposição dos pinos e suas funções no RPi3.	71
Figura 39	– Tablado montado para realização dos experimentos com o robô.	73
Figura 40	– Cilindros usados como marcos artificiais.	74
Figura 41	– Estrutura de sustentação da câmera.	76
Figura 42	– Câmera utilizada para confirmar resultados do algoritmo e rápida realização de testes.	76
Figura 43	– Tipos de distorções por tipo de lente.	77
Figura 44	– Padrão verificador para calibração.	77
Figura 45	– Sistemas RGB e CMYK de cores.	79
Figura 46	– Três tipos de representação de imagens, (a) RGB, (b) escala de cinza e (c) binária.	80
Figura 47	– Comparação da imagem distorcida com a corrigida.	81
Figura 48	– Imagem em RGB destacando obstáculos.	82
Figura 49	– Imagem binária aplicada à cor verde.	82
Figura 50	– Diferenças entre distâncias base-base e <i>tag-tag</i> .	84
Figura 51	– Detalhes da projeção de objetos na imagem.	85
Figura 52	– Arquitetura dos módulos de <i>software</i>	87
Figura 53	– Fluxograma de execução do Módulo Principal.	89
Figura 54	– Fluxograma de execução do Módulo TCP no servidor.	91
Figura 55	– Árvore de dependências do Módulo MATLAB.	92
Figura 56	– Fluxograma do algoritmo de localização.	93
Figura 57	– Espalhamento das partículas em todo o espaço amostral.	94
Figura 58	– Espalhamento de partículas em <i>clusters</i> .	95
Figura 59	– Exemplo de triangulação em três obstáculos.	95
Figura 60	– Exemplo de fusão sensorial.	98
Figura 61	– Resultado dos testes de translação em 30 cm.	103

Figura 62	– Resultado dos testes de translação em 50 cm.	103
Figura 63	– Resultado dos testes de translação em 100 cm.	104
Figura 64	– Resultado dos testes de medição em 30 cm.	105
Figura 65	– Resultado dos testes de medição em 50 cm.	105
Figura 66	– Resultado dos testes de medição em 100 cm.	106
Figura 67	– Resultado dos testes de medição em 120 cm.	106
Figura 68	– Simbologia dos elementos presentes nas figuras dos resultados dos experimentos.	108
Figura 69	– Foto ilustrando a primeira pose e o mapa (primeiro experimento).	108
Figura 70	– Experimento 1 - Resultado do algoritmo após processamento da pose 1.	110
Figura 71	– Experimento 1 - Resultado do algoritmo após processamento da pose 3.	111
Figura 72	– Experimento 1 - Resultado do algoritmo após processamento da pose 8.	112
Figura 73	– Experimento 1 - Resultado do algoritmo após processamento da pose 20.	113
Figura 74	– Foto ilustrando a primeira pose e o mapa (segundo experimento).	114
Figura 75	– Ilustração da trajetória feita pelo robô (segundo experimento).	114
Figura 76	– Experimento 2 - Resultado do algoritmo após processamento da pose 1.	115
Figura 77	– Experimento 2 - Resultado do algoritmo após processamento da pose 3.	116
Figura 78	– Experimento 2 - Resultado do algoritmo após processamento da pose 9.	117
Figura 79	– Experimento 2 - Resultado do algoritmo após processamento da pose 15.	118
Figura 80	– Foto ilustrando a primeira pose e o mapa (terceiro experimento).	120
Figura 81	– Ilustração da trajetória feita pelo robô (terceiro experimento).	120
Figura 82	– Experimento 3 - Resultado do algoritmo após processamento da pose 1.	121
Figura 83	– Experimento 3 - Resultado do algoritmo após processamento da pose 3.	122
Figura 84	– Experimento 3 - Resultado do algoritmo após processamento da pose 10.	123
Figura 85	– Experimento 3 - Resultado do algoritmo após processamento da pose 15.	124
Figura 86	– Erro entre a pose real e a pose estimada no primeiro experimento.	125
Figura 87	– Erro entre a pose real e a pose estimada no segundo experimento.	125
Figura 88	– Erro entre a pose real e a pose estimada no terceiro experimento.	126

## Lista de Quadros

Quadro 1	– Configuração de rodas para veículos móveis.	22
Quadro 2	– Desvios-padrões e erros normalizados de movimentos.	104
Quadro 3	– Desvios-padrões para testes repetitivos de medições.	107
Quadro 4	– Valores numéricos de erros absolutos entre a pose real e estimada no primeiro experimento.	126
Quadro 5	– Valores numéricos de erros absolutos entre a pose real e estimada no segundo experimento.	127
Quadro 6	– Valores numéricos de erros absolutos entre a pose real e estimada no terceiro experimento.	127
Quadro 7	– Valores numéricos de erros relativos entre a pose real e estimada no primeiro experimento.	128
Quadro 8	– Valores numéricos de erros relativos entre a pose real e estimada no segundo experimento.	128
Quadro 9	– Valores numéricos de erros relativos entre a pose real e estimada no terceiro experimento.	129

## Lista de Abreviaturas e Siglas

MCL	<i>Monte Carlo Localization.</i>
PVC	<i>Polyvinyl chloride.</i>
MILO	<i>Monte carlo Indoor LOcalization.</i>
MATLAB	<i>Matrix Laboratory.</i>
TEEE	Tópicos Especiais em Engenharia Elétrica.
SLAM	<i>Simultaneous Localization And Mapping.</i>
PDF	<i>Probability Density Function.</i>
KF	<i>Kalman Filter.</i>
EKF	<i>Extended Kalman Filter.</i>
HMM	<i>Hidden Markov Model.</i>
DBN	<i>Dynamic Bayes network.</i>
CCD	<i>Charged-Coupled Device.</i>
CMOS	<i>Complementary Metal-Oxide-Semiconductor.</i>
NASA	<i>National Aeronautics and Space Administration.</i>
LRC	Laboratório de Robótica Móvel e Comunicação Sem Fio.
CC	Corrente contínua.
SBC	<i>Single Board Computer.</i>
SO	Sistema Operacional.
BLE	<i>Bluetooth Low Energy.</i>
GPIO	<i>General Purpose Input Output.</i>
CSI	<i>Camera Serial Interface.</i>
DSI	<i>Display Serial Interface.</i>
SoC	<i>System on a Chip.</i>
UART	<i>Universal Asynchronous Receiver-Transmitter.</i>
I2C	<i>Inter-Integrated Circuit.</i>
SPI	<i>Serial Peripheral Interface.</i>
CI	Circuito Integrado.
LED	<i>Light-Emitting Diode.</i>
LDR	<i>Light Dependent Resistor.</i>

PSD	<i>Position Sensitive Detector.</i>
IREM	<i>Infrared Emitting Diode.</i>
PWM	<i>Pulse Width Modulation.</i>
LiPo	<i>Lítio Polímero.</i>
SSH	<i>Secure Shell.</i>
IP	<i>Internet Protocol.</i>
PC	<i>Computador Pessoal.</i>
TCP	<i>Transmission Control Protocol.</i>
API	<i>Application Programming Interface.</i>
RGB	<i>Red, Green, Blue.</i>
CMYK	<i>Cyan, Magenta, Yellow, black.</i>
HSV	<i>Hue, Saturation, Value.</i>
ROI	<i>Region of Interest.</i>
ROS	<i>Robot Operating System.</i>

## SUMÁRIO

Lista de Figuras		
Lista de Quadros		
Lista de Abreviaturas e Siglas		
1	<b>Introdução</b>	15
1.1	Objetivos	16
1.2	Motivação	17
1.3	Organização do trabalho	17
2	<b>Fundamentação teórica</b>	19
2.1	Problemas em robótica móvel	19
2.2	Tipos de plataformas móveis	20
2.3	Revisão de probabilidade	23
2.4	Tipos de mapas na representação do ambiente	26
2.5	Localização de robôs móveis	29
2.5.1	Tipos de solução para o problema de localização	34
2.5.1.1	Filtro de Kalman	34
2.5.1.2	Localização de Markov topológica e baseada em grade	34
2.5.1.3	Métodos baseados em amostras	35
2.5.2	Modelo probabilístico de movimento	38
2.5.3	Modelo probabilístico de medição	41
2.5.4	Associação de dados	42
2.6	Sensores	43
2.7	Fusão sensorial	47
2.8	Estruturação do ambiente interno	50
2.8.1	Estruturação de ambientes	51
2.8.2	Complexidade de ambientes internos	52
3	<b>Projeto do robô MILO</b>	54
3.1	Parte mecânica	54
3.2	Parte eletrônica	57
3.2.1	Placas eletrônicas	57
3.2.1.1	Placa Raspberry Pi 3	57
3.2.1.2	Placa MD25	58
3.2.1.3	Placa de interface	60
3.2.2	Sensores	62

3.2.2.1	Sonar SRF08	62
3.2.2.2	Sensor infravermelho GP2Y0A02YK0F	64
3.2.2.3	<i>Encoders</i> dos motores CC	65
3.2.2.4	<i>Bumper switch</i>	66
3.2.3	Motores	66
3.2.3.1	Motor CC EMG30	66
3.2.3.2	Servo motor	68
3.2.4	Outros detalhes da eletrônica embarcada	68
3.3	Parte computacional	70
3.4	Proposta de ambiente interno e obstáculos que formam o mapa	72
4	<b>Sistema de captura dos elementos reais presentes no ambiente</b>	75
4.1	Estrutura de captura do mapa e pose do robô por câmera externa	75
4.2	Calibração da câmera	76
4.3	Aquisição de dados na imagem	78
4.3.1	Imagens no MATLAB	78
4.3.2	Processamento de imagens no MATLAB	80
4.3.3	Correção dos erros de projeção na identificação das <i>tags</i>	83
5	<b>Módulos de <i>software</i> do robô MILO</b>	86
5.1	Arquitetura dos módulos de <i>software</i>	86
5.2	Módulo Principal	87
5.3	Módulo PWM	90
5.4	Módulo TCP	90
5.5	Módulos MATLAB e Câmera	92
6	<b>Experimentos e Resultados</b>	102
6.1	Testes de repetição para levantamento de ruído	102
6.1.1	Testes de movimento	102
6.1.2	Testes de medição	104
6.2	Experimentos de localização	107
6.3	Resultados dos experimentos	119
7	<b>Conclusões e trabalhos futuros</b>	130
	<b>Referências Bibliográficas</b>	

## 1 Introdução

A robótica é uma área de pesquisa que pode ser compreendida em três grandes linhas: a de base fixa (manipuladores), a móvel e a híbrida, esta última agregando funcionalidades das outras duas, no caso mobilidade e manipulação de objetos. Na linha da robótica móvel se projetam máquinas capazes de locomoção e percepção para a realização dos mais diversos tipos de tarefas. Os robôs móveis podem ter graus de autonomia, que vão desde os totalmente autônomos, que não requerem intervenção humana, até os que requisitam níveis variáveis de ação externa para realizarem as tarefas a eles atribuídas.

A robótica móvel está cada vez mais presente em ambientes comerciais, industriais e domésticos. Por exemplo, restaurantes, museus, armazéns, hospitais, entre outros, são locais que já possuem robôs na operacionalização mais eficiente de seu funcionamento. Robôs já estão sendo empregados para realizarem tarefas domésticas, como aspiração de pó. As produções industriais em seus diversos segmentos fazem uso intenso de robôs, nas três linhas da robótica. Além disso, áreas militares estão requisitando patrulhas robóticas para procedimentos de resgates, vigilância, entre outras tarefas de forma otimizadas. Com a grande demanda mundial por robôs móveis, as pesquisas envolvendo o aperfeiçoamento e o descobrimento das técnicas para realização das tarefas em robótica móvel está crescendo dentro das pesquisas conduzidas nas universidades e empresas.

A proposta que foi desenvolvida neste trabalho de monografia abordou o problema da localização global em ambientes internos e estáticos (não existem movimentos de outros elementos que não seja o do robô). Foi utilizada uma plataforma robótica de baixo custo que fez uso da abordagem probabilística para apresentar uma solução ao problema, o qual consiste na localização do robô em poses referenciadas a um sistema de coordenadas global. As estimações dessas poses do robô no ambiente são feitas por meio de um filtro de partículas clássico, denominado de MCL (*Monte Carlo Localization*), escolhido em função de ser multimodal, ou seja, manipula várias hipóteses das possíveis poses do robô que são mantidas nas partículas. Essa característica do filtro permite que o mesmo se recupere de associação de dados equivocadas, quando do processo de incorporação das medidas na pesagem das partículas. Fez-se a opção por utilizar a associação de dados desconhecida, onde os objetos que compõem o mapa não são identificados, por tratar-se de um problema mais complexo e de caráter mais realista. Entretanto, para facilitar a convergência das partículas ou hipótese das poses próximo da que se encontra o robô (erro minimizado e aceitável), foi empregada uma técnica disponível na literatura, porém não utilizada para a finalidade empregada neste



trabalho. Essa técnica envolve a triangulação de objetos medidos pelo robô antes de iniciar a aplicação do filtro (pose inicial e desconhecida) com o objetivo de reduzir o espaço de amostragem das partículas no início, ao invés de considerar toda a área do ambiente. Essa decisão teve como objetivo acelerar a convergência do filtro em poucas iterações iniciais, com os movimentos e medições sucessivos do robô, na medida em que concentra mais partículas em uma menor região do ambiente próxima a pose mais provável do robô obtida pela técnica de triangulação.

A implementação do projeto robótico foi feita exclusivamente para esta pesquisa. O robô foi projetado e equipado com um subsistema de medição, montado em uma torre e alinhado com o centro de giro do robô, onde foram posicionados dois pares de sensores sonar e infravermelho, defasados de 180°. Nessa configuração, um giro completo da torre em 180° permite que todo o entorno do robô no ambiente possa ser medido. O sistema embarcado utiliza a placa Raspberry Pi 3 que possui recurso *Wi-Fi* para comunicação do código do robô com o algoritmo de localização executando em um computador remoto. Todo o aparato mecânico e eletrônico utilizado justifica definir o robô como sendo de baixo custo.

Da mesma forma que o robô, o ambiente foi projetado e construído em um tablado de madeira de 278 x 143 cm<sup>2</sup>, suspenso do solo. Os obstáculos que compõem o mapa são sempre cilindros de PVC (*Polyvinyl chloride*) de altura 35 cm e raio de 5 cm. Tanto o robô quanto os obstáculos têm alturas aproximadamente iguais pelo motivo de ser facilitada a projeção de seus centros (obstáculos e robô) na base do tablado, a partir de imagens capturadas por um sistema de câmeras. As partes superiores dos obstáculos e do robô possuem *tags* com coloração distintas e adequadas ao tratamento da imagem pelo sistema de câmera. Esses dados não têm nenhuma influência no filtro MCL, servindo apenas para levantamento automatizado dos ruídos de movimento do robô e de avaliação dos erros entre as poses estimadas e reais capturadas pelo sistema óptico durante os três experimentos realizados.

## 1.1 Objetivos

O objetivo geral deste trabalho de monografia é apresentar uma solução ao problema de localização em ambientes internos e estáticos, utilizando um robô considerado de baixo custo por utilizar apenas sensores sonar e infravermelho, fazendo uso da abordagem probabilística.

Os objetivos específicos foram elaborados de acordo com as etapas para uma solução robusta e rápida ao problema proposto:

- a) Obter uma solução ao problema de localização em ambientes internos e estáticos, usando o filtro Monte Carlo em associação com a técnica de triangulação de objetos para viabilizar a convergência do filtro em poucas iterações para várias condições de ambientes;
- b) Propor e implementar um sistema robótico que realize a tarefa de localização especificada e que inclua todo *hardware* e *software* necessários, além do ambiente de navegação; e
- c) Após convergência do filtro, mantê-la em níveis baixos e considerados aceitáveis nas demais iterações que venha a ser realizados no algoritmo.

## 1.2 Motivação

Muitos trabalhos envolvendo localização de robôs móveis se utilizam de sensores de grande precisão, como os *lasers scanners*, possibilitando que o nível de incerteza na tarefa realiza sejam minimizados de maneira a simplificar a ação pelo algoritmo que executa a tarefa. Abordar esse problema da perspectiva de uso de sensores bem mais baratos e, portanto, ruidosos nas medidas que adquirem, são de mais difícil solução, sendo um dos motivos principais para a escolha feita nesta proposta de trabalho. Neste sentido, a proposição desse tema de monografia tem o intuito de se construir uma base sólida para futuras implementações e melhoramentos em cima das soluções para os problemas em robótica móvel, como planejamento de trajetórias, além de se aproveitar o robô construído para ensino na disciplina TEEE (Tópicos Especiais em Engenharia Elétrica) - Robótica Móvel Probabilística oferecida ao curso de Engenharia Elétrica da Universidade Federal do Maranhão.

Além disso, o aluno possui interesse em seguir carreira de pesquisador na área de robótica móvel, o que fez desse trabalho um importante aprendizado em sua formação acadêmica com o projeto e implementação do sistema robótico aplicado à localização global, servindo como base para futuros trabalhos ao nível de pós-graduação.

## 1.3 Organização do trabalho

Este trabalho de monografia está dividido em sete capítulos, incluindo este primeiro que introduz o assunto. No Capítulo 2 são feitas as revisões bibliográficas dos temas relacionados ao problema a ser resolvido. No Capítulo 3, descrevem-se o projeto e as etapas de construção do robô, da sua estrutura mecânica aos componentes eletrônicos. O Capítulo 4 trata de uma

descrição a respeito do uso de um sistema de câmera para capturar de forma mais rápida os dados do ambiente e utilizá-los de forma eficiente ao longo do desenvolvimento do trabalho, no levantamento de ruídos do modelo de medição e nos experimentos de localização realizados. No Capítulo 5 são discutidas as formas que os módulos de *software* foram implementados, sejam eles na forma de pseudocódigo, ou fluxograma de execução. No Capítulo 6 são mostrados os experimentos realizados para comprovação da solução e são feitas discussões sobre os resultados obtidos. Finalmente, o Capítulo 7 faz-se a conclusão, citando possíveis trabalhos futuros.

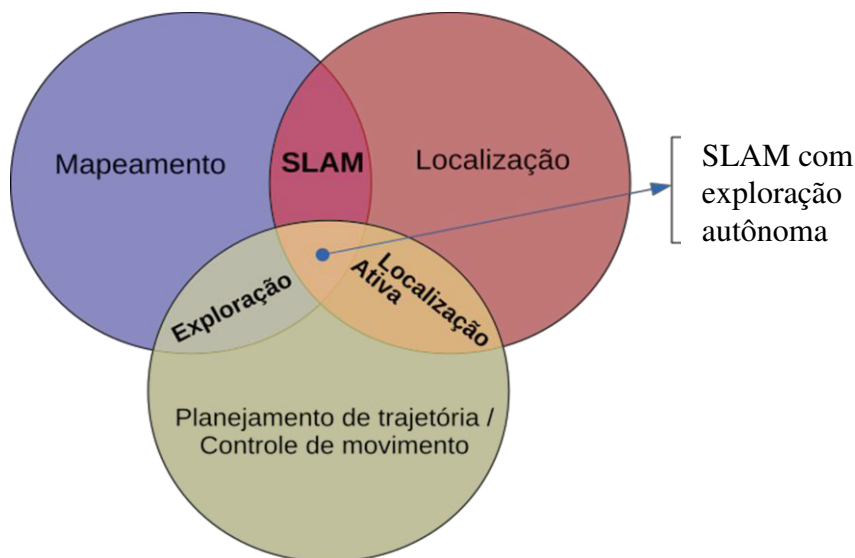
## 2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão tratados temas relacionados de forma mais direta com o desenvolvimento deste trabalho de monografia, envolvendo o assunto central que é uma proposta de solução ao problema de localização global em ambientes internos 2D (duas dimensões) usando um robô com sensores de baixo custo.

### 2.1 Problemas em Robótica Móvel

Os problemas relacionados à robótica móvel podem ser resolvidos de forma probabilística, onde os algoritmos fazem as manipulações das incertezas que um robô tem de sua localização e/ou dos obstáculos presentes no ambiente. As tarefas mais comuns a serem realizadas por um robô móvel são: mapeamento, localização e planejamento de trajetória. A Figura 1 mostra essas tarefas e as que são derivadas, usando duas ou todas de forma simultânea.

Figura 1 – Tarefas comuns em robótica móvel.



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

Mapeamento é o problema de integrar as informações adquiridas pelos sensores do robô em uma dada representação, respondendo à pergunta “Como é o ambiente ao meu redor?”. Em contraste a isso, a localização lida com a necessidade de estimar a pose do robô relativa a um mapa conhecido, devendo responder à pergunta “Onde estou?”. Finalmente, o planejamento de trajetória ou controle de movimento envolve a questão de como guiar o robô

a um local desejado ou ao longo de uma trajetória de forma eficiente, podendo ser expresso como uma pergunta simples: “Como chego a um local especificado?” (STACHNISS, 2006).

As tarefas que empregam a realização de duas ou três desses problemas básicos em robótica móvel, são mais amplamente pesquisadas. Como exemplo pode ser citada a localização ativa, onde o robô ao deslocar-se entre duas poses do ambiente (em 2D essas variáveis são compostas por três informações: duas coordenadas e uma orientação) decide o caminho mais adequado e possível de navegar. Nesse problema, está sendo abordada tanto a localização quanto o planejamento de trajetória. O problema SLAM (*Simultaneous Localization And Mapping*) autônomo considera todas as três tarefas básicas, sendo que o robô deve estimar sua localização e adquirir o mapa do ambiente de forma simultânea, decidindo os caminhos para isso (BUONOCORE, 2016).

Ao realizar essas tarefas o robô deve ser capaz de acomodar a grande quantidade de incertezas que existem no mundo físico, existindo vários fatores que contribuem para suas existências, entre eles (THRUN *et al.*, 2006):

- a) O ambiente do robô;
- b) Ruídos dos sensores;
- c) Ruídos dos atuadores usados na propulsão do robô;
- d) Utilização de modelos de movimento e medição empregados nos algoritmos que realizam a tarefa;
- e) Aproximações feitas pelo algoritmo como forma de minimizar a carga computacional;

O nível de incerteza depende do domínio de aplicação. Em alguns empregos de robôs, como linhas de montagem, podem-se desenvolver sistemas onde as incertezas se tornam somente fatores marginais ao problema. Em contraste, robôs operando em ambientes residenciais ou outros planetas terão que lidar com incertezas substanciais. Esses robôs são forçados a agir mesmo que seus sensores e modelos internos não possam prover informação suficiente para tomar a decisão certa com máxima certeza (THRUN *et al.*, 2006).

## **2.2 Tipos de plataformas móveis**

Um robô móvel precisa de mecanismos de locomoção que possibilitam ele realizar os seus movimentos sem dificuldades dentro do ambiente. Existe uma vasta variedade de modos possíveis para alcançar o movimento desejado. Logo, a escolha de um tipo de mecanismo é um importante passo no projeto do robô. Dentre as mais utilizadas em pesquisas de

desenvolvimento, se encaixam duas grandes subáreas: robôs com pernas e robôs com rodas (SIEGWART *et al.*, 2004).

A roda tem sido o método de deslocamento mais popular em robótica móvel, constituindo veículos feitos pelo homem de uma forma geral, podendo alcançar excelente eficiência com implementação mecânica relativamente simples. Dos problemas encontrados em robôs com rodas, equilíbrio não é uma adversidade geralmente associada a eles, pois são quase sempre projetados para que todas as rodas fiquem em contato com a superfície a todo o momento.

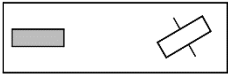
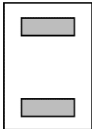
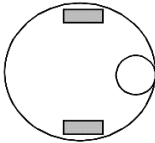
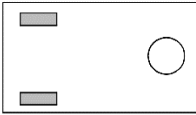
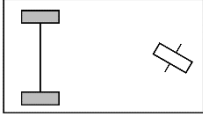
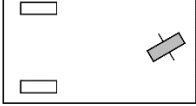
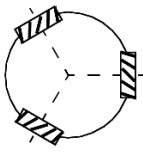
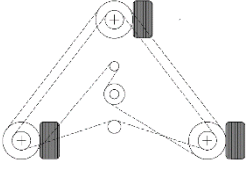
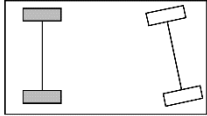
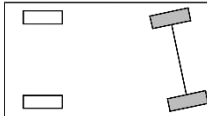
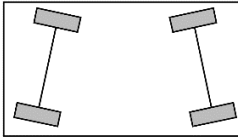
Desse modo, robôs com três rodas, ou triciclos, são usualmente o suficiente para garantir equilíbrio. Dessa forma, a maior parte dos problemas relacionados aos robôs com rodas está associada a problemas de tração e estabilidade, capacidade de manobra e controle. Alguns exemplos de combinações envolvendo o número de rodas, o arranjo geométrico e uma descrição, se encontram no Quadro 1 (SIEGWART *et al.*, 2004).

A Figura 2 a seguir mostra os diferentes tipos de rodas: a) padrão, b) de rodízio, c) suecas e d) esféricas (SIEGWART *et al.*, 2004). As rodas padrão e de rodízio têm um eixo primário de rotação, sendo altamente direcionais. A principal diferença é que a roda padrão produz esse movimento primário sem muitos efeitos colaterais, devido a seu centro de rotação coincidir com o ponto de contato da superfície, enquanto que a roda de rodízio funciona com um eixo vertical deslocado, causando a transmissão de uma força ao chassi durante a direção.

Rodas suecas e esféricas são projetadas para serem menos contidas pela direcionalidade do que as rodas padrão. A roda sueca funciona como uma roda padrão, porém, possui baixa resistência ao movimento em outros eixos que não o eixo de rotação. Como exemplo, a sueca 90° possui rolamentos ao longo da circunferência da roda de forma a permitir movimentos perpendiculares ao sentido convencional. Esses rolamentos funcionam como rodas passivas possibilitando menos fricção também em sentidos de movimento que não sejam para frente e para trás.

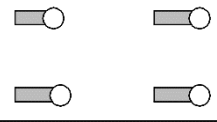
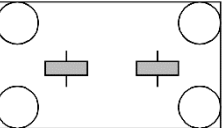



A roda esférica é também chamada de omnidirecional verdadeira, pois são projetadas de forma a permitir o movimento em qualquer direção, podendo ser alimentada ou não por um motor. Ela possui um mecanismo de difícil implementação, na forma motora ou de tração, sendo mais utilizada como roda passiva no movimento.

Quadro 1 – Configurações de rodas para veículos móveis.

Número de rodas	Arranjo	Descrição
2		Uma roda direcional na frente, 1 roda de tração atrás.
		Diferencial centralizado de duas rodas com centro de massa abaixo do eixo.
3		Diferencial centralizado de duas rodas com terceiro ponto de contato.
		Dois rodas motorizadas independentes atrás/frente, 1 roda omnidirecional não motorizada na frente/atrás.
		Dois rodas de tração conectadas atrás, 1 roda direcionada livre na frente.
		Dois rodas livre atrás, 1 roda direcionada de tração na frente.
		Três rodas suetas motorizadas ou esféricas arranjadas em triângulo; movimento omnidirecional é possível.
		Três rodas direcionadas e motorizadas de forma síncrona; orientação não é controlável.
4		Dois rodas motorizadas atrás, 2 rodas livres na frente; direção deve ser diferente para as duas rodas para evitar deslizamento.
		Dois rodas motorizadas na frente, 2 rodas livres atrás; direção deve ser diferente para as duas rodas para evitar deslizamento.
		Quatro rodas direcionadas e motorizadas conectadas em pares.

Fonte: (SIEGWART *et al.*, 2004)

Quadro 1 – Configurações de rodas para veículos móveis (continuação).

Número de rodas	Arranjo	Descrição
4		Duas rodas motorizadas diferenciais com 2 pontos de apoio adicionais.
		Quatro rodas de rodízio motorizadas e direcionadas.
6		Duas rodas motorizadas e direcionadas alinhadas no centro, 1 roda omnidirecional em cada canto
		Duas rodas tracionadas (diferenciais) no centro, 1 roda omnidirecional em cada canto.
Ícones para cada tipo de roda:		
	Roda omnidirecional não motorizada;	
	Roda Sueca motorizada;	
	Roda padrão não motorizada;	
	Roda padrão motorizada;	
	Roda de rodízio direcionada e motorizada;	
	Roda padrão direcionada;	
	Rodas conectadas;	

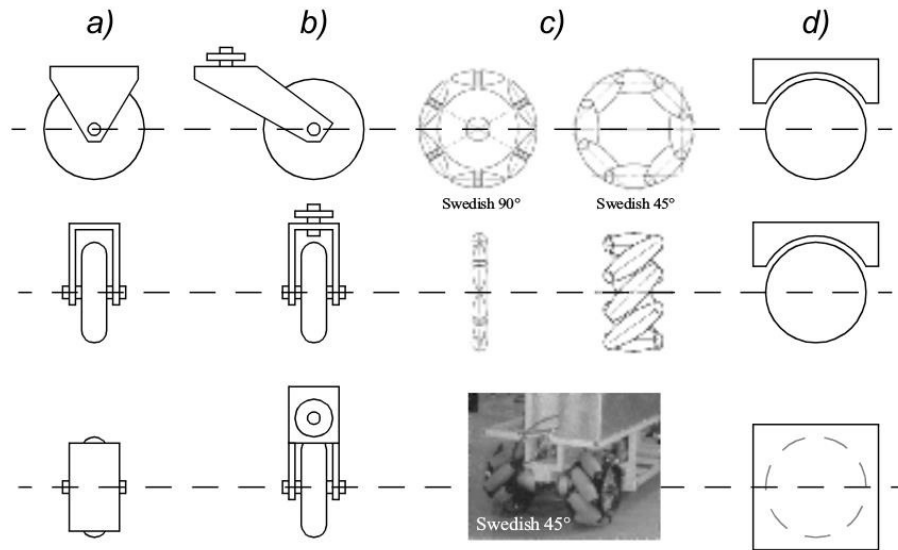
Fonte: (SIEGWART *et al.*, 2004)

### 2.3 Revisão de Probabilidade

A fim de familiarizar o leitor com os alguns conceitos básicos e termos que serão usados durante a escrita deste trabalho, é necessário fazer uma revisão sobre probabilidade e suas propriedades, usados nos algoritmos de solução de tarefas em robótica. Nessa abordagem as quantidades são modeladas como variáveis aleatórias, sendo esses valores os dados de sensores, do controle aplicado ao robô, do estado do robô, dentre outros. Essa seção será baseada na referência (THRUN *et al.*, 2006) e, por questões de simplificação, serão omitidas quando dos conceitos formulados matematicamente.



Figura 2 – Os quatro tipos básicos de rodas.



Fonte: (SIEGWART *et al.*, 2004)

Variáveis aleatórias podem assumir múltiplos valores possíveis de acordo com o resultado de um fenômeno aleatório. Considerando  $X$  uma variável aleatória, o possível valor que ela pode assumir é designado por  $x$ :

$$p(X = x) \quad (1)$$

No caso de variáveis aleatórias discretas, os valores que elas podem assumir são finitos, sendo que a soma de todos os seus possíveis resultados sempre é igual à unidade:

$$\sum_x p(X = x) = 1 \quad (2)$$

Valores de probabilidades atribuídos às variáveis são sempre positivas. Para simplificar a notação, usa-se  $p(x)$  ao invés de  $p(X = x)$ .

Uma função densidade de probabilidade (*Probability Density Function* - PDF) é aquela aplicada a uma variável aleatória contínua cujo valor em certa amostra ou ponto no espaço amostral pode ser interpretado como a probabilidade relativa de que o valor da variável aleatória seja igual a essa amostra. Uma função comumente usada é a distribuição normal unidimensional ou gaussiana, com média  $\mu$  e variância  $\sigma^2$ , dada por:

$$p(x) = (2\pi\sigma^2)^{-\frac{1}{2}} e^{\left(-\frac{1(x-\mu)^2}{2\sigma^2}\right)} \quad (3)$$

A Eq. 3 pode ser abreviada como  $N(x; \mu, \sigma^2)$ , que especifica a variável aleatória, sua média e sua variância.

A probabilidade conjunta de duas variáveis aleatórias  $X$  e  $Y$  é dada por:

$$p(x, y) = p(X = x \text{ e } Y = y) \quad (4)$$

Essa expressão descreve a probabilidade do evento que a variável  $X$  assume o valor  $x$  e  $Y$  assume o valor  $y$ . Se  $X$  e  $Y$  são eventos independentes, tem-se que:

$$p(x, y) = p(x) \cdot p(y) \quad (5)$$

Pode-se ainda, definir a probabilidade de  $X$  assumir o valor  $x$ , condicionando o fato de  $Y$  já ser conhecido ( $y$ ), sendo denotada por:

$$p(x|y) = p(X = x | Y = y) \quad (6)$$

Sendo  $p(y) > 0$ , então a Eq. 6 pode ser definida por:

$$p(x|y) = \frac{p(x, y)}{p(y)} \quad (7)$$

Considerando na Eq. 7 que os eventos  $X$  e  $Y$  são independentes, tem-se:

$$p(x|y) = p(x)p(y)/p(y) = p(x) \quad (8)$$

Em outras palavras, se  $X$  e  $Y$  forem eventos independentes, o valor de  $Y$  não interfere nos valores da variável  $X$ . Logo, é irrelevante saber o valor de  $Y$ .

Existem dois teoremas de extrema importância para a robótica móvel probabilística: a) o Teorema da Probabilidade Total; e b) a Regra de Bayes. Ambos os teoremas usam o conceito de probabilidade condicional. O primeiro, na forma discreta (Eq. 9) e contínua (Eq. 10), respectivamente:

$$p(x) = \sum_y p(x|y)p(y) \quad (9)$$

$$p(x) = \int p(x|y)p(y)dy \quad (10)$$

A regra de Bayes, considerando  $p(y) > 0$ , relaciona um condicional  $p(x|y)$  ao seu “inverso”  $p(y|x)$ :

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \quad (11)$$

A formulação da regra de Bayes é de fundamental importância na implementação dos algoritmos que realizam as tarefas em robótica móvel. A explicação da regra de Bayes pode ser mais facilmente entendida através de um exemplo prático. Assumindo que a variável aleatória  $X$  se refere à pose do robô (par ordenado com orientação em um sistema cartesiano), e outra  $Y$  informando à medição de sensores do robô, emprega-se a regra de Bayes para inferir a probabilidade  $p(x|y)$  que significa a probabilidade do robô estar na pose  $x$  dado que a medida  $y$  foi observada. Logo, de acordo com a Eq. 11, a probabilidade  $p(y|x)$  se torna necessária para esse cálculo. Essa probabilidade inversa define a probabilidade da medição  $y$ ,

assumindo que  $x$  é a pose do robô naquele instante. Em robótica, a probabilidade de  $p(y|x)$  é um modelo generativo, uma vez que descreve, em certo nível de abstração, como as variáveis de estado  $X$  causam medições de sensor  $Y$ .

Pode-se também condicionar a regra de Bayes sobre múltiplas variáveis aleatórias, como  $Y$  e  $Z$ . Por exemplo, condicionando a Regra de Bayes no valor conhecido da variável  $Z = z$ , tem-se:

$$p(x|y, z) = \frac{p(y|x, z)p(x|z)}{p(y|z)} \quad (12)$$

A validade da Eq. 12 só se verifica na consideração que  $p(y|z) > 0$ . A aplicação dessa ampliação do número de variáveis condicionais também tem aplicação em robótica móvel, por exemplo, em associação de dados conhecidos. Na Seção 2.5.4 esse tema será mais detalhado.

De forma semelhante, pode-se condicionar a regra para combinar probabilidades de variáveis aleatórias independentes, dada a observação de um evento  $Z = z$ , chamada de independência condicional:

$$p(x, y|z) = p(x|z)p(y|z) \quad (13)$$

Independência condicional tem um papel importante em robótica probabilística, pois se aplica sempre que uma variável  $y$  não carrega informação a respeito de outra variável  $x$  se outra variável  $z$  possui valor conhecido. Os algoritmos da solução do problema SLAM, onde a pose e o mapa devem ser estimados ( $x$  e  $y$  desconhecidos), fazem uso desse teorema probabilístico.

## 2.4 Tipos de mapas na representação do ambiente

Existem diferentes tipos de modelos para representar o ambiente utilizado em robótica móvel. Os mais comuns são os mapas métricos e os topológicos. O primeiro detalha as dimensões com certa precisão dos contornos do ambiente, enquanto o segundo conecta os diversos tipos de mapas métricos e é utilizado na navegação de áreas contíguas do ambiente. A solução de uma tarefa em robótica móvel pode envolver apenas mapas métricos ou ambos, este último sendo referenciado como mapeamento híbrido (THRUN *et al.*, 2006).

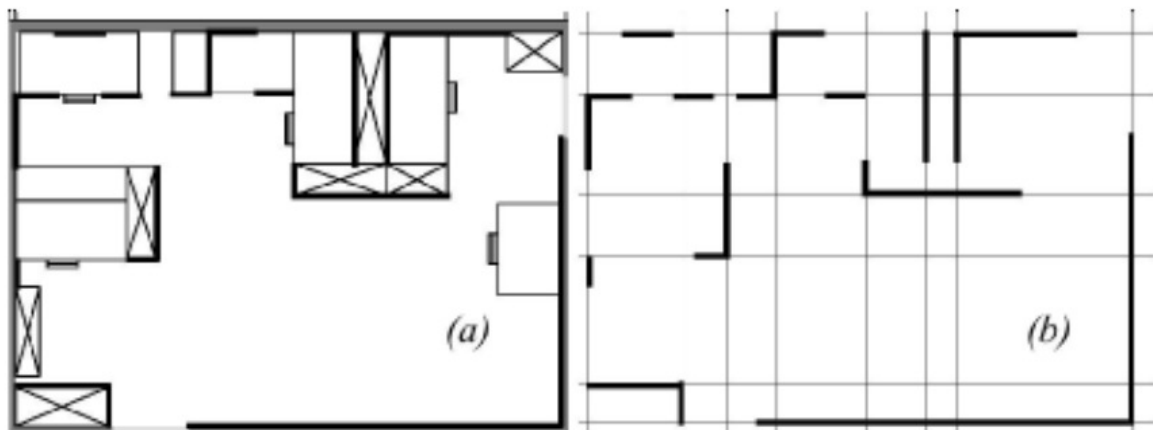
Com relação aos mapas métricos, os mais comuns são os de características, os geométricos e os de grade. Um mapa de características (*feature*) armazena um conjunto de informações detectado no ambiente, tipicamente representando linhas ou cantos, quando sensores de distância são utilizados.

Mapas geométricos representam obstáculos detectados pelo robô como figuras geométricas (por exemplo, círculos e polígonos). Esse tipo de representação é mais compacto, se comparada a outros métodos, pois se utiliza de menos recursos de memória.

Mapas de grade discretiza o ambiente em células, sendo denominados de grade de ocupação (*occupancy grid*), onde em cada célula é armazenada a informação sobre a área que ocupa em termos de probabilidade. Mais comumente usados são os mapas de ocupação de grade que armazenam um valor por célula, relacionando a probabilidade de aquela célula estar ocupada por um obstáculo, estar livre (sem ocupação) ou ser desconhecida (ainda não investigada). A vantagem de se usar mapas de grade é que não se depende de recursos pré-definidos e que precisam ser extraídos de dados do sensor, no entanto, possuem a desvantagem de erros de discretização e o alto uso de memória (STACHNISS, 2006).

As figuras a seguir mostram as representações citadas. No caso da Figura 3(a) mostra a planta real do local, enquanto na Figura 3(b) a representação em forma de linhas infinitas. Já na Figura 4 são apresentados os obstáculos de um ambiente como polígonos e na Figura 5 um ambiente é representado na forma de grade de ocupação.

Figura 3 – Exemplo de mapa contínuo ou de recursos. (a) Mapa real. (b) Representação como um conjunto de linhas infinitas.

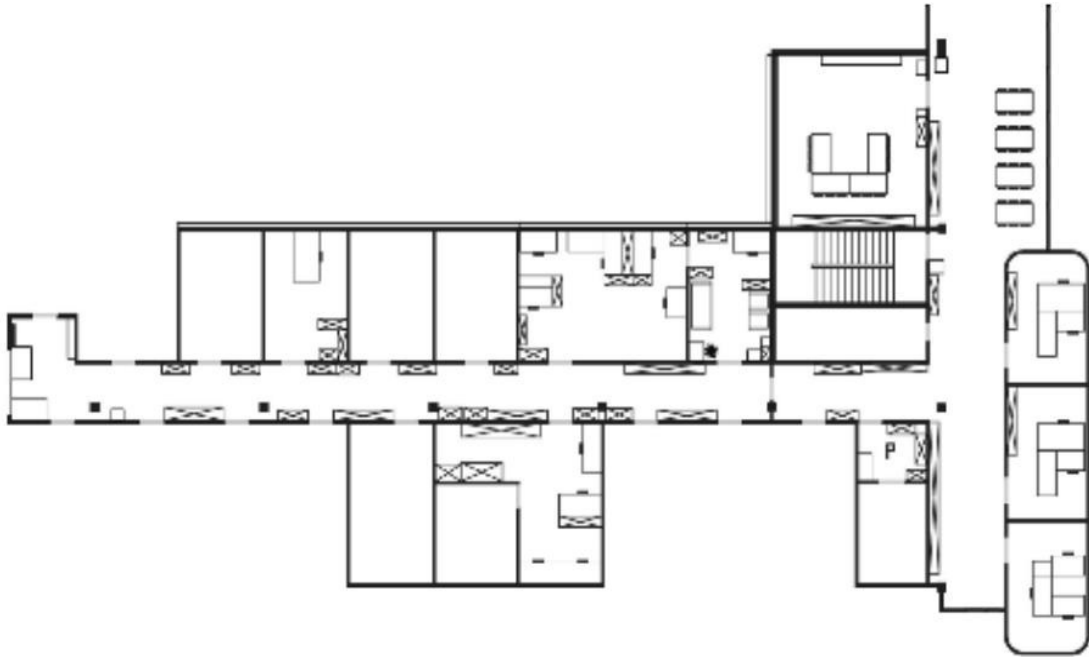


Fonte: (SIEGWART *et al.*, 2004)

Basicamente, três relações fundamentais precisam ser entendidas quando se escolhe uma representação para o mapa do ambiente (SIEGWART *et al.*, 2004):

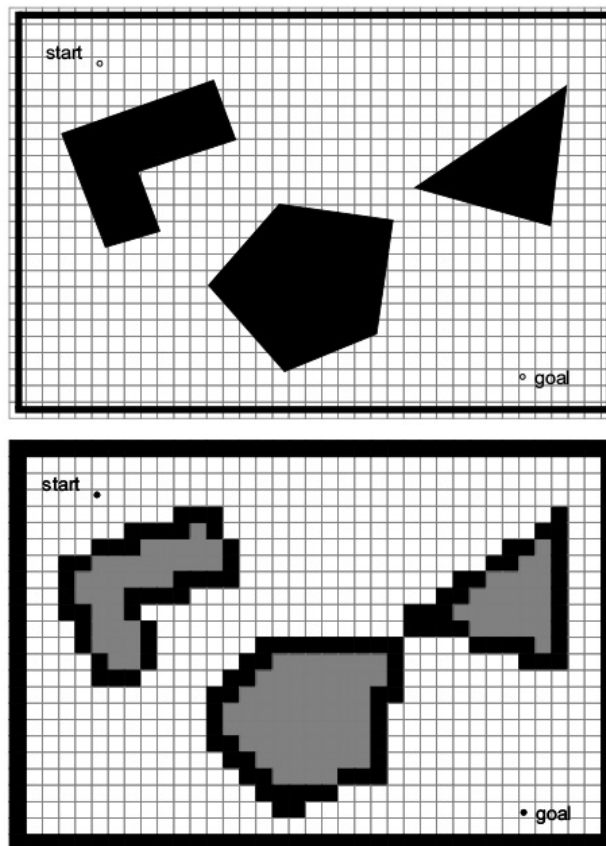
1. A precisão do mapa deve ser tal que permite representar o ambiente com precisão suficiente para o robô alcançar seus objetivos na realização da tarefa;

Figura 4 – Representação de um mapa por entidades geométricas.



Fonte: (SIEGWART *et al.*, 2004)

Figura 5 – Representação de um ambiente em grade de ocupação.



Fonte: (SIEGWART *et al.*, 2004)

2. A precisão do mapa e o tipo de característica representada devem se relacionar com a precisão e os tipos de dados retornados pelos sensores presentes no robô;
3. A complexidade da representação do mapa é diretamente proporcional à complexidade computacional envolvida na solução do problema, seja ele mapeamento, localização, navegação, etc.

## 2.5 Localização de robôs móveis

Para navegar em um ambiente com segurança, um robô precisa saber onde se encontra, sendo a estimação da pose do robô o problema a ser resolvido. Duas importantes questões relacionadas à localização na robótica móvel são a estimativa da pose global e o rastreamento da pose local que, de forma simples, podem ser chamados de localização global e rastreamento local, respectivamente.

A primeira se refere à habilidade de determinar a pose do robô em um mapa conhecido *a priori*, dado que o robô está em algum lugar inicialmente desconhecido. Caso não haja nenhum mapa conhecido pelo robô, existem aplicações onde o robô pode construir o mapa ao mesmo tempo em que explora o ambiente, conhecidas por SLAM, porém estão além do escopo deste trabalho.

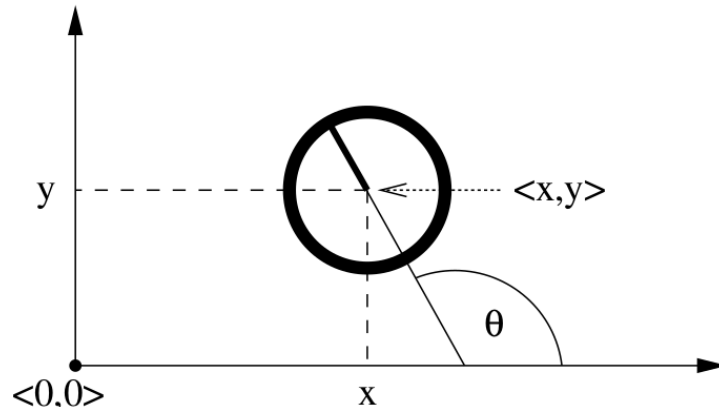
O segundo problema é aplicado quando o robô sabe sua pose global no mapa no instante atual, porém precisa rastrear essa pose de forma local ao longo do tempo, enquanto navega. Ambas as aplicações são necessárias para possibilitar o robô executar tarefas úteis, como entregas de mercadorias em escritórios ou fornecer *tours* para visitantes em museus (DELLAERT *et al.*, 1999).

Nas soluções para esses problemas, é importante ressaltar que todas as abordagens estão sendo citadas para resolver situações onde o mapa é bidimensional, ou seja, há apenas duas coordenadas para determinar o local onde o robô se encontra (posição). Porém, a informação das coordenadas não é suficiente, sendo necessária a orientação do robô. Tipicamente denota-se pose o vetor coluna (Eq. 14) que contém a posição e a orientação do robô em relação a um sistema de coordenadas global (GF ou *global frame*), conforme ilustra a Figura 6:

$$pose = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \quad (14)$$

Este vetor representa o estado do robô em determinado instante discreto, representado por  $x_t$ , onde  $t$  é o estado atual (0, 1, 2, etc.) estimado pelo algoritmo de localização. Para realizar o deslocamento do robô no ambiente e realizar a estimação da nova pose, é necessário

Figura 6 – Pose do robô em um sistema de coordenadas global.



Fonte: (THRUN *et al.*, 2006).

enviar um comando com os movimentos e adquirir as medições feitas pelos sensores do robô, as quais informam as distâncias e o ângulo que os objetos do ambiente são percebidos pelo sistema de medição do robô. As representações temporais do controle aplicado e das medições dos sensores são  $u_t$  e  $z_t$ , respectivamente, comentadas com maiores detalhes mais adiante.

Na abordagem Bayesiana é de interesse a construção da densidade posterior  $p(x_t | z_t)$ , no caso, o estado em processamento na estimação das variáveis de interesse. No problema de localização, essas variáveis compõem a informação da pose do robô, que no plano são representadas pelas coordenadas ( $x$  e  $y$ ) e pela orientação ( $\theta$ ). A solução probabilística a esse problema é o objetivo desse trabalho.

A PDF é utilizada para representar todo o conhecimento que se tem a respeito do estado  $x_{t-1}$  (estado *a priori*) para, a partir daí, estimar a pose posterior  $x_t$ . Dessa forma, sucessivos estados alcançados pelo robô são estimados pelo algoritmo denominado filtro probabilístico, conforme será exemplificado nos experimentos realizados e apresentados no Capítulo 6.

No problema do rastreamento da pose local (localização relativa) é possível tanto a utilização de filtros de estimação que possuem apenas uma hipótese relativa ao estado atual, denominados de monomodal, quanto algoritmos que operam com múltiplas hipóteses desse estado, referenciados como filtros multimodais (DELLAERT *et al.*, 1999).

Entretanto, na localização global só é possível o uso de filtros multimodais, na medida em que não se conhece *a priori* o estado inicial do robô, cabendo ao filtro a tarefa de conseguir a convergência da estimação da pose após alguns estados iniciais de estimação ser realizados. Um filtro irá convergir em sua estimação quando os erros entre as poses estimada

e real do robô em estados temporais sucessivos no ambiente se mantiver dentro de níveis aceitáveis. Caso contrário, diz-se que o filtro diverge (DELLAERT *et al.*, 1999).

Dessa forma, os filtros probabilísticos ao estimar a localização do robô em um ambiente, devem calcular de forma recursiva a probabilidade do próximo estado, sempre considerando duas fases sucessivas (DELLAERT *et al.*, 1999):

- a) Fase de predição: usa um modelo de movimento do robô (representado pela probabilidade condicional  $p(x_t|x_{t-1}, u_t)$ ) para estimar a probabilidade da pose posterior  $p(x_t)$ , assumindo que ela só depende da pose *a priori*  $x_{t-1}$  e do controle aplicado  $u_t$ . A probabilidade da posterior, calculada nesta fase, é então obtida por:

$$p(x_t|z_{t-1}) = \int p(x_t|x_{t-1}, u_t)p(x_{t-1}|z_{t-1})d(x_{t-1}) \quad (15)$$

- b) Fase de atualização: emprega um modelo de medição, cuja probabilidade é definida por  $p(z_t|x_t)$ , para estimar a posterior  $p(x_t|z_t)$ , assumindo que o vetor de medidas atual ( $z_t$ ) é condicionalmente independente de medidas anteriores ( $z_{t-1}$ ). Nessa etapa, busca-se incorporar as medidas efetuadas no estado com a movimentação do robô, de maneira a minimizar as incertezas, sendo representado pela regra de Bayes, dada por:

$$p(x_t|z_t) = \frac{p(z_t|x_t)p(x_t|z_{t-1})}{p(z_t|z_{t-1})} \quad (16)$$

Na fase de predição, quando ocorre o movimento do robô, necessariamente as incertezas são ampliadas, enquanto na fase de atualização, com a aquisição e incorporação das medidas na estimação, elas são minimizadas. Esse efeito “ampliar-reduzir” incertezas é próprio dos filtros probabilísticos que, se ocorrem mantendo a sua convergência, indicam que o filtro foi apropriadamente implementado, principalmente no que diz respeito aos modelos de movimento e medição empregados (DELLAERT *et al.*, 1999). O detalhamento da implementação desses modelos matemáticos para serem computados pelo filtro incluem ruídos que serão detalhados nas Seções 2.5.2 e 2.5.3.

Após a fase de atualização, o processo é repetido de forma recursiva. O tempo de estimação de cada estado pelo filtro é considerado discreto, sendo que em  $t = 0$  (primeiro estado) o que se sabe é dado pela probabilidade  $p(x_0)$ , a qual tem representações diferentes para o tipo do problema da localização a ser abordado. No caso da localização relativa, essa probabilidade geralmente é definida como a média e a covariância de uma função gaussiana centrada ao redor de  $x_0$ , uma vez que se conhece a pose do robô relativa ao GF. No caso da localização global essa probabilidade pode ser uma densidade uniforme em cima de todas as



poses possíveis no ambiente (DELLAERT *et al.*, 1999). Neste trabalho será implementado e testado um filtro que possibilita obter uma solução ao problema de localização global.

Apenas como exemplificação ao problema de localização, a Figura 7 ilustra uma visualização da localização Markoviana de robôs móveis de forma unidimensional (simplificada), onde um robô tenta se localizar em um corredor composto de paredes e portas. O sensor utilizado no robô detecta portas (THRUN *et al.*, 2006).

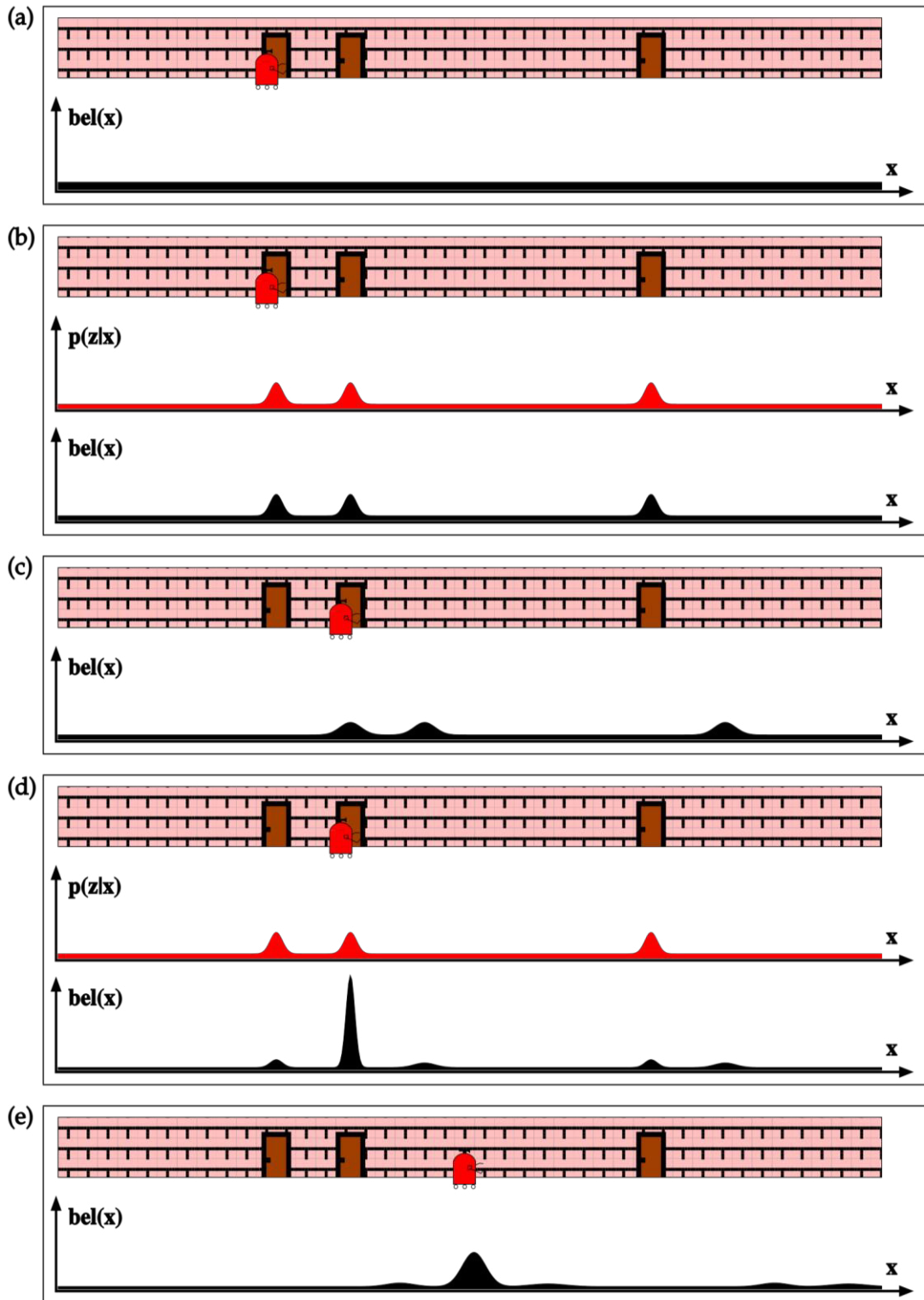
Na Figura 7(a) o robô é mostrado no estado inicial ( $t = 0$ ), onde o gráfico mostra uma distribuição gaussiana uniforme da confiança da pose do robô (*Belief*, definida por  $bel(x)$ ) em todos os possíveis locais do mapa (localização global). No próximo instante discreto mostrado na Figura 7(b), o robô faz a primeira medição, definida matematicamente por  $p(z_t|x_t)$ . Nessa medição ele verifica que está próximo a uma porta, incorporando a medida e atualizando a confiança de sua pose, onde é calculada uma probabilidade maior nos locais onde se encontram as portas e menor nas seções das paredes. A integral da probabilidade em todo o espaço possível de se localizar o robô deve ser sempre a unidade. Ainda que o estado atualizado pela medição permitiu aumentar a confiança nos pontos onde existem portas, o robô ainda não tem condições de estimar com confiança satisfatória em qual porta do corredor se encontra. É importante observar um fenômeno presente em filtros probabilísticos que, mesmo calculando com maior probabilidade os locais onde existem portas no mapa, no restante do ambiente a probabilidade nunca é nula. Nesses filtros a certeza absoluta ou uma grande incerteza a respeito das variáveis estimadas (poses, no caso da localização) nunca é desejável, pois pode induzir o filtro à divergência (THRUN *et al.*, 2006).

Em um próximo instante (Figura 7(c)) o robô faz um deslocamento no corredor, sendo o efeito desse movimento observado na diminuição da confiança que amplia a incerteza de sua pose. Esse fenômeno ocorre na medida em que seu modelo de movimento sempre insere ruídos. Na Figura 7(d), após o deslocamento, mais uma medição é realizada próxima a porta do meio e, mais uma vez, a confiança é atualizada, ampliando as probabilidades próximas das portas em relação às seções no corredor. Entretanto, percebe-se que ele tem maior confiança de ter detectado a porta do meio em função do acentuado aumento da probabilidade em suas proximidades em relação às outras duas portas e paredes.

Na Figura 7(e) o robô novamente faz um movimento e o mesmo efeito ocorre na distribuição da confiança em todo o espaço do corredor, ou seja, a maior confiança em relação a sua localização é agora maior, porém diminuída em relação ao movimento em curso. Nesse exemplo foi demonstrado o que é conhecido como Filtro de Bayes para a estimação da

posterior, onde são usadas as fases de predição e atualização de forma simples com modelos gaussianos.

Figura 7 – Robô móvel em tarefa de localização global.



### 2.5.1 Tipos de soluções para o problema de localização

A solução para o problema de localização pode ser obtida executando de forma recursiva e sequencial as Eq. 15 (modelo de movimento) e Eq. 16 (modelo de medição) das fases de predição e atualização, respectivamente. No entanto, dependendo de como se representa a posterior  $p(x_t | z_t)$ , pode-se obter vários algoritmos ou filtros com diferentes propriedades, alguns deles comentados a seguir.

#### 2.5.1.1 Filtro de Kalman (KF) e Kalman Estendido (EKF)

O filtro de Kalman (KF – *Kalman Filter*) é uma solução já consagrada para rastreamento local, onde a representação da PDF emprega a média e a matriz de covariância. Esse tipo de algoritmo de estimação já é suficiente por ser satisfatoriamente aplicado a sistemas lineares, tendo em vista que tanto os modelos de movimento quanto de medição empregados utilizam densidades gaussianas. No entanto, normalmente em robótica móvel os modelos são por natureza não linear, dificultando o emprego desse filtro.

O Filtro de Kalman Estendido (EKF – *Extended Kalman Filter*) foi uma alternativa ao KF para solucionar aplicações onde ocorre a presença de modelos não lineares. Para isso, basicamente, ele lineariza os modelos no ponto de interesse, através do emprego da série de Taylor (MEDEIROS, 2011; DELLAERT *et al.*, 1999).

Tanto o KF quanto o EKF são considerados filtros monomodais de estimação na medida em que propaga a probabilidade da posterior (no caso do problema da localização representa a pose do robô) centrada em apenas uma hipótese. Essa condição impossibilita sua aplicação para algumas situações como: a) estimação de estado global (ex: localização, SLAM); e b) recuperação de associação de dados indevidas, quando da incorporação das medidas no estágio de atualização. Existem versões EKF para tratar com mais de uma hipótese de estado do filtro, não comentadas neste trabalho (THRUN *et al.*, 2006).

#### 2.5.1.2 Localização de Markov topológica e baseada em grade

Diferente do KF e do EKF, a localização de Markov é uma derivação direta do filtro de Bayes, o filtro mais básico para a inferência da posterior utilizando densidades gaussianas. Sua forma topológica é baseada em marcos no ambiente utilizado para navegação e os estados são organizados de acordo com a estrutura topológica do ambiente (DELLAERT *et al.*, 1999).

Uma abordagem mais fina, em se tratando de resolução, pode ser encontrada utilizando a versão da localização de Markov baseada em grade, que é calculada com base na integral de

uma grade de pontos, podendo lidar assim, com densidades multimodais e PDF não gaussianas. Esse tipo de resolução sofre em relação às demais versões de filtro, devido ao alto custo computacional e o comprometimento prematuro do tamanho do espaço de estado, tendo em vista que o método discretiza somente a parte de interesse no espaço de estados (DELLAERT *et al.*, 1999).

### 2.5.1.3 Métodos baseados em amostras

Em métodos baseados em amostras, podem-se representar densidades de probabilidade através de um número definido de hipóteses aleatórias, comumente referidas como filtros não paramétricos, pois não são fixados necessariamente em forma de funções gaussianas para representar o estado posterior do filtro. Ao contrário, esses filtros são aplicáveis a situações onde os filtros KF e o EFK possuem restrições, incorporando modelos não lineares e sendo considerados filtros multimodais, já que cada amostra carrega uma estimativa do estado (DELLAERT *et al.*, 1999). Exemplos implementáveis desse tipo de filtro são o histograma e o de partículas, sendo nesta seção abordado apenas o segundo pelo seu massivo uso nas pesquisas.

No filtro de partículas o objetivo é calcular recursivamente, em cada instante de tempo, o conjunto de amostras  $S_t$  que são gerados a partir da densidade  $p(x_t | z_t)$ . Dessa forma, o valor contínuo da posterior é aproximado por um número finito de valores, cada um correspondendo a uma hipótese possível no espaço de estados.

Neste trabalho de monografia, a solução para o problema de localização foi implementada, baseando-se em partículas. A esse tipo de localização dá-se o nome Localização Monte Carlo.

O algoritmo para localização usando esse método segue de forma recursiva assim como descrito em 2.5(a) e 2.5(b), porém com a adição do conjunto de  $N$  partículas que representam a probabilidade da pose do robô no mapa.

Na fase de predição, aplica-se o controle dado a cada partícula  $s_{t-1}^i$  amostrando a partir de  $p(x_t | s_{t-1}^i, u_{t-1})$ . Logo, assume-se que já se tem o conjunto de partículas  $S_{t-1}$ , mesmo na primeira interação, quando se define que o robô pode estar em qualquer lugar do mapa. Em resumo, nas duas fases recursivas do filtro (predição e atualização) tem-se (THRUN *et al.*, 2006):

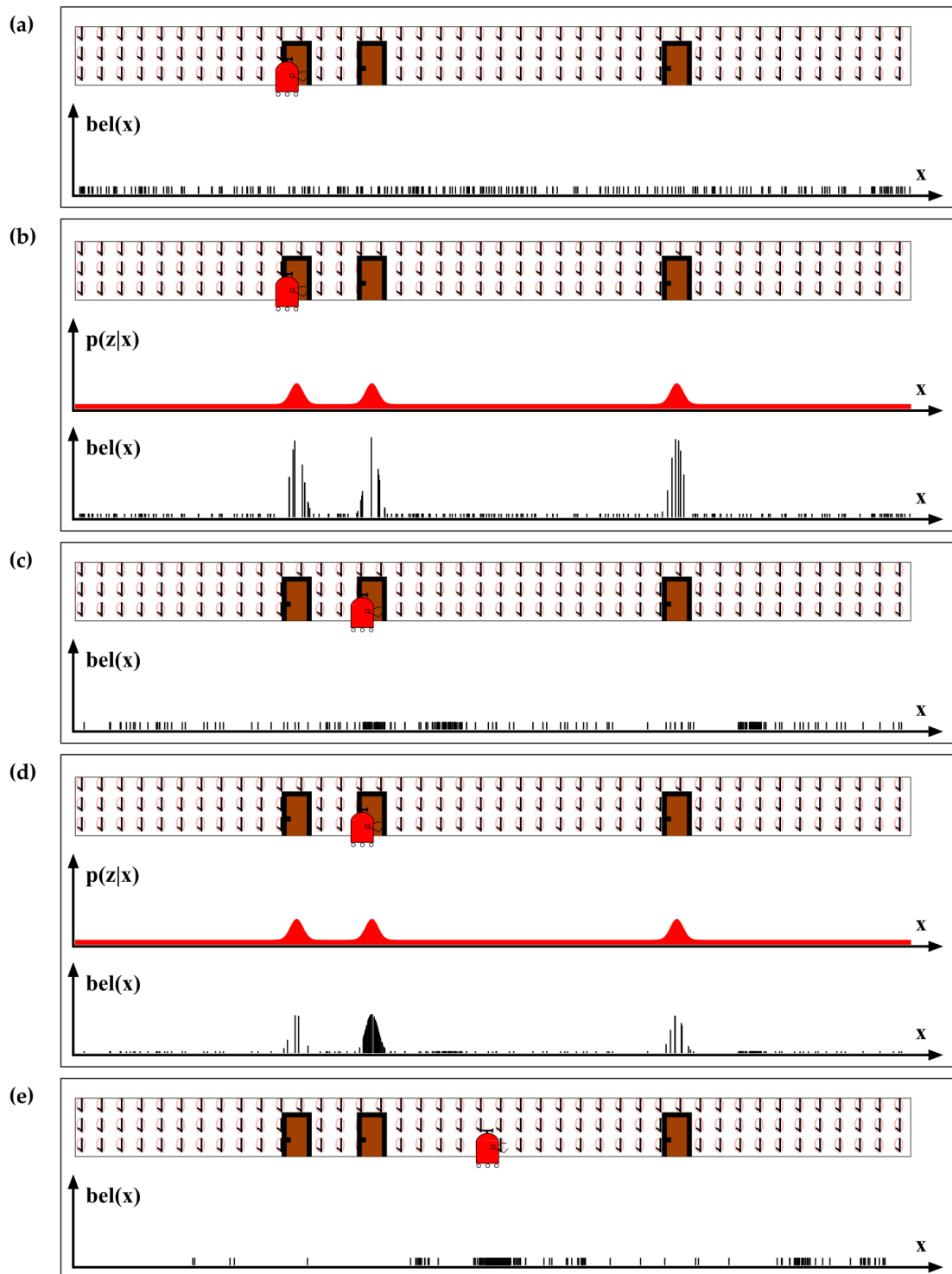
- a) Para cada partícula  $s_{t-1}^i$  amostra-se uma partícula  $\hat{s}_t^i$ , baseado no modelo de movimento, cuja probabilidade é dada por  $p(x_t | s_{t-1}^i, u_{t-1})$ .

- b) Calcula-se o peso de cada partícula amostrada, incorporando as medidas obtidas, através da probabilidade  $m_t^i = p(z_t | \hat{s}_t^i)$ ; e
- c) Para cada partícula amostrada e pesada, faz-se o processo de reamostragem da partícula, possibilitando que sobrevivam no conjunto as que reflitam (através de seu peso) a mais provável de ser o estado posterior.

Após as amostragens das  $N$  partículas, o resultado é o conjunto  $\hat{S}_t$  que ainda não incorporou as medidas dos sensores, a qual é feita só na fase de atualização do filtro. A incorporação da medida  $z_t$  às partículas permite que se defina a importância da partícula amostrada, sendo quantificada através do seu peso, calculada pela probabilidade  $m_t^i = p(z_t | \hat{s}_t^i)$ . A probabilidade obtida na pesagem de cada partícula ( $\hat{s}_t^i$ ) do conjunto  $\hat{S}_t$  proporcionará a sobrevivência das partículas de maior importância no conjunto, depois de aplicado o processo de reamostragem. Esse procedimento é feito inserindo uma partícula  $s_t^j$  a partir de  $\{\hat{s}_t^i, m_t^i\}$  (amostrada e pesada). É importante considerar que nesse processo a quantidade de partículas permanece a mesma, ainda que o número de hipóteses distintas normalmente diminua em função da sobrevivência das que tem maior peso. A compatibilidade entre esses valores dá-se em função de replicações de partículas no conjunto reamostrado. O processo de amostragem, pesagem e reamostragem ocorrem recursivamente e, normalmente na convergência do filtro, as incertezas da pose do robô aumentam na amostragem e diminuem na reamostragem. O filtro se inicializa com o conjunto  $S_0$  apenas de  $p(x_0)$  (DELLAERT *et al.*, 1999).

A Figura 8 utiliza o mesmo exemplo da Figura 7, no entanto, com a explicação do algoritmo MCL. Inicialmente a incerteza da pose global é definida através de um conjunto de partículas colocadas aleatoriamente e uniformemente em todo o espaço amostral, simbolizadas por traços (hipóteses) como ilustrado na Figura 8(a). Com a detecção de uma porta, o MCL calcula um fator de importância (peso) para cada partícula, sendo esse aglomerado de hipóteses resultante mostrado na Figura 8(b), com as alturas de cada uma correspondendo a sua importância. Vale ressaltar que o conjunto ainda é idêntico ao do quadro anterior em (a), tendo como diferença o fato dos fatores de importância, ou peso da partícula, serem diferentes (THRUN *et al.*, 2006). A Figura 8(c) mostra o conjunto após reamostragem e após a incorporação dos ruídos no movimento, levando a um novo conjunto com fatores de importância normalizados, porém, com mais partículas acumuladas nos locais próximos às portas. Uma nova medição é realizada atribuindo pesos não uniformes ao conjunto de partículas, como ilustrado na Figura 8(d).

Figura 8 – Robô móvel em tarefa de localização global através do filtro de partículas (Monte Carlo).



Fonte: (THRUN *et al.*, 2006)

A maior parte das partículas se concentra na região ao redor da segunda porta, cuja localização real é a mais provável. A Figura 8(e) mostra o conjunto de partículas após mais uma reamostragem e um novo movimento realizado pelo robô, sendo normalizados pela quantidade de partículas. Esse exemplo descrito mostra que o conjunto de partículas aproxima o estado posterior como derivado através de um filtro de Bayes (THRUN *et al.*, 2006).

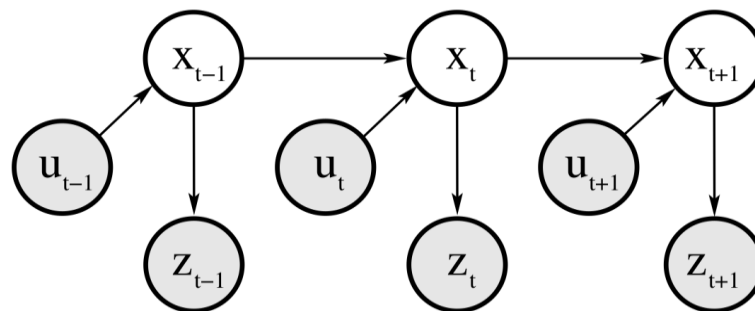
### 2.5.2 Modelo probabilístico de movimento

Cinemática robótica tem sido estudada ao longo das últimas décadas, no entanto, esse estudo tem sido quase que exclusivamente endereçado de forma determinística principalmente no início das pesquisas dos desenvolvimentos dos algoritmos. Robótica probabilística generaliza as equações cinemáticas devido ao fato de que o efeito de um controle (movimento) é incerto, sendo difícil modelar de forma precisa todos os parâmetros que ocasionam erros, incluindo a presença de ruídos externos impossíveis de serem incorporados ao modelo (THRUN *et al.*, 2006).

O emprego do modelo de movimento probabilístico é facilitado em sua construção computacional para o tratamento das incertezas presentes e previstas a partir dos erros obtidos em ensaios empíricos. De forma análoga se implementa o modelo de medição, sendo que a abordagem probabilística permite a criação desses dois modelos relativamente simplificados (THRUN *et al.*, 2006).

O modelo de movimento desempenha a função de promover a transição de estado do filtro, definido pela densidade  $p(x_t|x_{t-1}, u_t)$ , onde  $x_t$  e  $x_{t-1}$  são as poses nos instantes discretos do filtro  $t$  (posterior) e  $t - 1$  (priori) e  $u_t$  o comando de movimento. A Figura 9 demonstra a evolução dos estados a partir dos controles enviados e medidas  $z_t$  efetuadas e incorporadas. Assim, o estado no instante  $t$  é estocasticamente dependente do estado no

Figura 9 – Rede Dinâmica de Bayes.



instante  $t - 1$  e do controle  $u_t$ , enquanto a medida  $z_t$  depende estocasticamente do estado no instante  $t$ . Essa sucessão de estimação representa um modelo generativo temporal, conhecido como modelo de Markov oculto (HMM – *Hidden Markov Model*) ou rede dinâmica de Bayes (DBN – *Dynamic Bayes network*) (THRUN *et al.*, 2006).

Os dois modelos probabilísticos normalmente empregados na literatura são o de velocidade e o odométrico, considerados complementares quanto ao tipo de informação de movimento que está sendo processada. O primeiro assume que o controle  $u_t$  especifica as velocidades linear e angular que devem ser atuadas no movimento do robô, em que é considerada a cinemática do modelo, onde essas velocidades atuam em tempos determinados. O segundo assume que se tem informação de odometria, utilizada como controle (THRUN *et al.*, 2006).

Odômetros são sensores que medem a revolução da roda de um robô. Apesar de serem sensores, considera-se a odometria como dados de controle, já que sua informação diz respeito à mudança de pose do robô. Na prática, modelos de odometria tendem a ser mais precisos do que modelos de velocidade, pelo motivo que a precisão alcançada com robôs que executam comandos de velocidade não ser do mesmo nível da dos odômetros. Usa-se o modelo de velocidade em planejamento de trajetórias, enquanto o modelo odométrico é mais facilmente implementado e gerenciado pelo filtro, somente que a informação do movimento pelos odômetros só estão disponíveis após o movimento (THRUN *et al.*, 2006).

Neste trabalho se adota o modelo odométrico para computar o movimento do robô, sendo o vetor de controle  $u_t$  definido como:

$$u_t = \begin{pmatrix} \delta_{rot1} \\ \delta_{trans} \\ \delta_{rot2} \end{pmatrix} \quad (17)$$

Considerando que o robô se desloca da pose  $x_{t-1} = (x \ y \ \theta)^T$  para a pose  $x_t = (x' \ y' \ \theta')^T$ , o controle entre duas poses sempre será decomposto em três movimentos, como mostra a Figura 10. Uma rotação  $\delta_{rot1}$  inicial é necessária para definir a direção que o robô fará a translação  $\delta_{trans}$ , alcançando a posição desejada (coordenadas apenas), finalizando com a última rotação  $\delta_{rot2}$  para que se ajuste para a orientação desejada (MEDEIROS, 2011).



A computação de cada componente do vetor de controle  $u_t$  é obtida por:

$$\delta_{rot1} = \tan^{-1} \left( \frac{y' - y}{x' - x} \right) - \theta \quad (18)$$

$$\delta_{trans} = \sqrt{(x - x')^2 + (y - y')^2} \quad (19)$$

$$\delta_{rot2} = \theta' - \theta - \delta_{rot1} \quad (20)$$

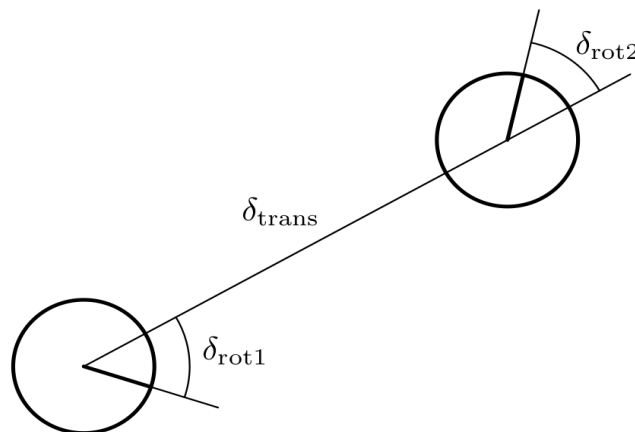
Após cada um dos três movimentos previstos no modelo, os odômetros fornecem informações que necessariamente serão diferentes dos valores enviados pelo controle (Eq. 17), as quais são consideradas ideais. Esse fato se dá devido a ruídos sistemáticos (por exemplo, diferença no diâmetro das rodas) e não sistemáticos (derrapagem) de odometria, cujo modelamento é importante para a computação no filtro. Os erros não sistemáticos são difíceis de serem modelados, dada a sua natureza aleatória, ficando a cargo do filtro corrigir essas situações para manter sua convergência. Os valores de controle processados (variáveis com chapéus nas Eq. 21 a Eq. 23) para o vetor de controle são adquiridos subtraindo os erros ( $\epsilon$ ) de cada movimento (THRUN *et al.*, 2006):

$$\hat{\delta}_{rot1} = \delta_{rot1} - \epsilon_{rot1} \quad (21)$$

$$\hat{\delta}_{trans} = \delta_{trans} - \epsilon_{trans} \quad (22)$$

$$\hat{\delta}_{rot2} = \delta_{rot2} - \epsilon_{rot2} \quad (23)$$

Figura 10 – Decomposição do modelo odométrico em três movimentos.



Fonte: (THRUN *et al.*, 2006)

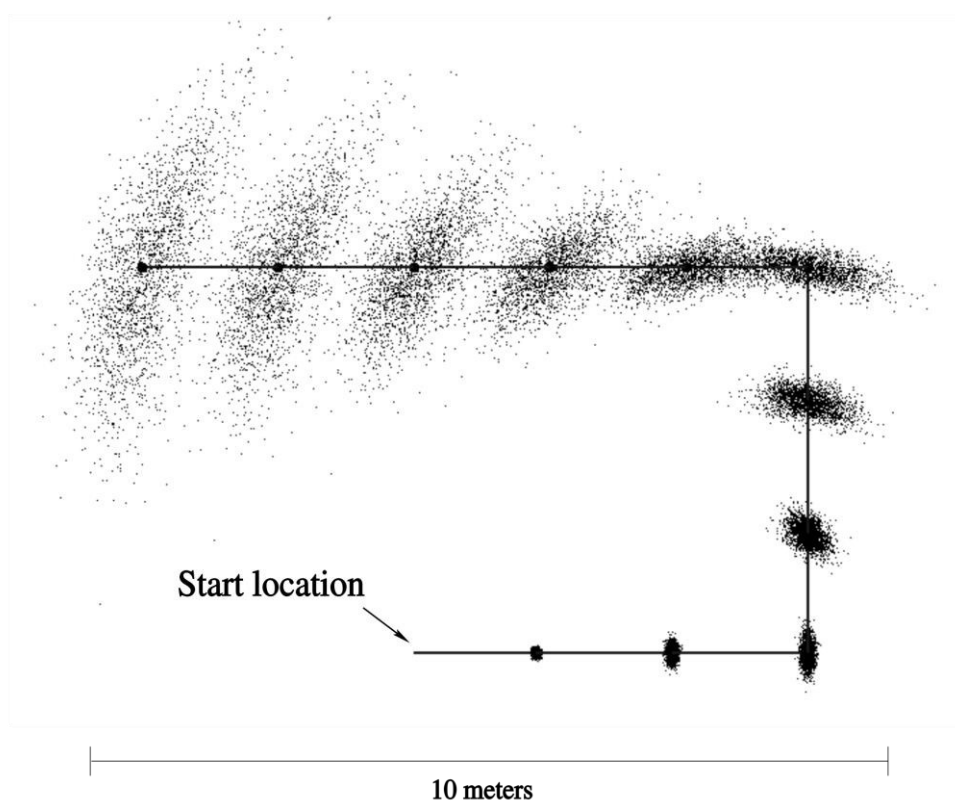
A Figura 11 demonstra o espalhamento das partículas levando em consideração somente o modelo de movimento (sem a incorporação das medidas efetuadas). A linha sólida representa a trajetória comandada do robô. Além disso, nota-se que o erro odométrico se propaga de acordo com a distância percorrida, aumentando em cada estado estimado no sentido longitudinal (deslocamentos lineares) e angular (giros).

### 2.5.3 Modelo probabilístico de medição

Modelos de medição descrevem o processo de formação pelo qual medidas de sensores são geradas no mundo físico. Entretanto, existem diferentes modalidades de sensores, como os que medem distâncias, que respondem à estímulos táteis ou aquisição de imagens de câmeras. Os detalhes do modelo dependem de qual sensor está se tratando, sensores de imagem são melhores modelados através de geometria projetiva, enquanto que sonares devem ser modelados descrevendo a onda de som e seu reflexo em superfícies no ambiente (THRUN *et al.*, 2006).

O modelo de medição usado neste trabalho se baseia na extração de obstáculos a partir

Figura 11 – Aproximação por amostras da pose de um robô sem incorporação de medidas.



Fonte: (THRUN *et al.*, 2006)

de dados brutos dos sensores, no caso, sonar e infravermelho. As medidas em cada pose do ambiente serão agrupadas para tentar caracterizar os obstáculos que compõem o mapa. Por sua vez, o mapa é um vetor de obstáculos  $M = \{m_1, m_2, m_3 \dots m_{No}\}$ , em que cada elemento possui uma localização global  $m_{i,x}$  e  $m_{i,y}$ , onde  $1 \leq i \leq No$  ( $No$  é o número do obstáculo) e  $x$  e  $y$  são as coordenadas do centróide do cilindro em relação ao FG. Para compor o vetor da estimativa de medição  $\hat{z}_t = (d \ \phi)^T$  é necessário a distância  $d$  e a orientação  $\phi$  do obstáculo (parâmetros dos objetos), ambos em relação ao frame do sistema de medição do robô. Assim, dada a pose do robô  $x_t = (x \ y \ \theta)^T$ , tem-se (MEDEIROS, 2011):

$$d_t^j = \sqrt{(m_{i,x} - x)^2 + (m_{i,y} - y)^2} + \epsilon_d \quad (24)$$

$$\phi_t^j = \tan^{-1} \left( \frac{m_{i,y} - y}{m_{i,x} - x} \right) - \theta + \epsilon_\phi \quad (25)$$

As equações acima definem a  $j$ -ésima medida associada ao  $i$ -ésimo obstáculo, no tempo  $t$ . A variável  $r$  é a distância do centróide à borda do obstáculo e  $\epsilon_d$  e  $\epsilon_\phi$  os ruídos associados às medidas dos sensores sonar e infravermelho, respectivamente.

#### 2.5.4 Associação de dados

Associação de dados é o procedimento para buscar a correspondência de cada medida  $z_t^j$  obtida e os obstáculos existentes no mapa  $M$ , que é fornecido *a priori* no problema da localização probabilística.

Existem dois tipos de associações de dados empregados pelos filtros em robótica probabilística (THRUN *et al.*, 2006): conhecida e desconhecida. A diferença entre ambas é que na associação conhecida cada característica no ambiente possui uma assinatura, ou seja, algo (cor, forma, etc.) que ao ser detectado pelo sistema de medição é tratado de forma inequívoca no mapa para a incorporação da medida no filtro usado na solução da tarefa.

Neste trabalho fez-se uso da correspondência de dados desconhecida, onde não é possível determinar com absoluta certeza a correspondência entre as leituras dos sensores e os obstáculos (cilindros). Estima-se a correspondência através de máxima verossimilhança, processo que permite associar cada medida de distância e ângulo do robô ao mais provável obstáculo do mapa. Esse procedimento é feito a partir do resultado do modelo de medição, dos dados fornecidos pelos sensores, a pose atual do robô e o que se sabe do mapa até aquele instante para definir qual o obstáculo mais provável de ter sido observado (MEDEIROS, 2011).

A incerteza  $S_t^i$  relativa à medida estimada  $\hat{z}_t^i$  é calculada por:

$$S_t^i = H_t^i \bar{\Sigma}_t [H_t^i]^T + Q_t \quad (26)$$

Na Eq. 26  $H_t^i$  é a matriz jacobiana que lineariza o modelo de medição em relação a  $x_t$ . O termo  $\bar{\Sigma}_t$  é a matriz de covariância da estimativa de localização do robô e  $Q_t$  é a matriz de ruído de medição, esta última dada por:

$$Q_t = \begin{pmatrix} \alpha_d d & 0 \\ 0 & \alpha_\phi \phi \end{pmatrix} \quad (27)$$

Os elementos da matriz  $Q_t$  são obtidos empiricamente, da mesma forma que os ruídos do modelo de movimento. Os parâmetros  $\alpha_d$  e  $\alpha_\phi$  representam as incertezas relativas à distância e à orientação, respectivamente, usadas na estimação pelo modelo de medição. Pode-se substituir o produto  $\alpha_\phi \phi$  por uma constante que significa quantos graus de erro existe na estimativa de orientação. A equação a seguir define a correspondência através do método de máxima verossimilhança, fazendo com que, das medidas tomadas pelos sensores, vale aquela que mais se aproxima da medida estimada (MEDEIROS, 2011):

$$c(j) = \underset{i}{\operatorname{argmax}} \det(2\pi S_t^i)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (z_t^j - \hat{z}_t^i)^T (S_t^i)^{-1} (z_t^j - \hat{z}_t^i) \right\} \quad (28)$$

A Eq. 28 utiliza a distribuição de probabilidade normal, onde o vetor das médias são as medidas obtidas e a variância é a matriz de incerteza de cada medida  $i$  mantida no vetor  $(S_t^i)$ .

## 2.6 Sensores

Sensores são dispositivos que podem detectar mudanças ou fenômenos no ambiente e podem transmitir uma resposta correspondente, geralmente, na forma de impulso elétrico.

Atualmente, a robótica móvel dispõe de uma variedade de sensores para percepção do ambiente. A escolha do sensor a ser empregado em um projeto deve considerar os seguintes aspectos (SCARAMUZZA, 2015): princípio físico, precisão, custo computacional, custo energético e preço.

Com o passar dos anos, os sensores foram se tornando cada vez mais refinado em resolução e, conseqüentemente, em sua precisão, além de serem vendidos a preços mais baratos. Três importantes características, dentre outras, estão associadas aos sensores, as quais implicam em sua escolha para uma determinada aplicação<sup>1</sup>:

<sup>1</sup> Site: [http://www.cpdee.ufmg.br/~palhares/Instrumentacao\\_NotasAula.pdf](http://www.cpdee.ufmg.br/~palhares/Instrumentacao_NotasAula.pdf)

- a) Exatidão – informa o quanto o valor adquirido pelo sensor se aproxima do real, não sendo possível tratá-la de forma isolada. Deve-se levar em conta o meio onde estão sendo feitas, bem como as perturbações nas capturas das medidas;
- b) Precisão – informa a consistência do sensor em relação à repetibilidade das medidas efetuadas; e
- c) Resolução – define a menor variação percebida pelo sensor que irá provocar uma alteração em sua saída.

Devido aos sensores de baixo custo sonar e o infravermelho utilizados neste trabalho apresentarem maior ruído nas medições que efetuam, comparados aos de maiores custos como o *scanner* a laser, o tratamento das medidas requer maior complexidade e esforço computacional. Ainda que as medidas adquiridas pelo sonar e infravermelho apresentam maiores erros, os mesmos devem ser resolvidos pelo modelo de medição empregado no filtro. O poder computacional disponibilizado nos lançamentos de novas placas eletrônicas embarcadas de baixo custo possibilita minimizar o tempo de processamento requerido pelo filtro (SCARAMUZZA, 2015).

A classificação dos sensores pode ser organizada em quatro grandes grupos:

- a) Sensores proprioceptivos – são aqueles que medem valores internamente ao sistema (robô), e que podem ou não ser embarcados no robô. Como exemplos tem-se *encoders* óticos (odômetros), medidores de bateria etc.
- b) Sensores exteroceptivos – é todo sensor que pode coletar informação do ambiente, podendo estar ou não embarcados ao robô. Alguns exemplos são os sensores que medem distância de objetos, intensidade da luz ambiente, adquirem imagens etc.
- c) Sensores passivos – são aqueles que medem a reação natural do ambiente, ou seja, não é necessária uma interferência, que não a do próprio fenômeno. Como exemplos, têm-se os detectores de colisão (*bumpers*) e o sistema estereoscópico de visão com o uso de duas câmeras em analogia a visão humana.
- d) Sensores ativos – emitem certa energia a fim de medir a reação do ambiente. Por mais que o sensoriamento ativo seja independente de estímulo natural, ele ainda é fortemente influenciado pelo ambiente, porém, na maioria dos casos possui melhor desempenho comparado aos passivos. Alguns exemplos são os sensores como o que emprega uma ponteira laser e uma câmera na definição da distância e ângulo do ponto do objeto iluminado (BUONOCORE, 2013).

As duas primeiras classificações podem ser entendidas como sendo informações adquiridas pelo o que sensor está medindo, seja o estado do robô ou os objetos que compõem

o ambiente. Os dois últimos casos são os que determinam como os sensores fazem suas medidas. É importante ressaltar que um sensor pode ser ativo e passivo em tempos distintos de funcionamentos, como é o caso de câmeras com visão noturna. Nesse sistema, a presença da luz torna-os passivos e, no escuro, se utilizam da luz infravermelha emitida, tornando-se ativos (SCARAMUZZA, 2015).

Neste trabalho, apenas três sensores serão abordados, devido à grande quantidade desses dispositivos que poderiam estender o trabalho sem acrescentar melhorias de forma significativa na solução do problema abordado. São eles o sonar, o infravermelho e a câmera.

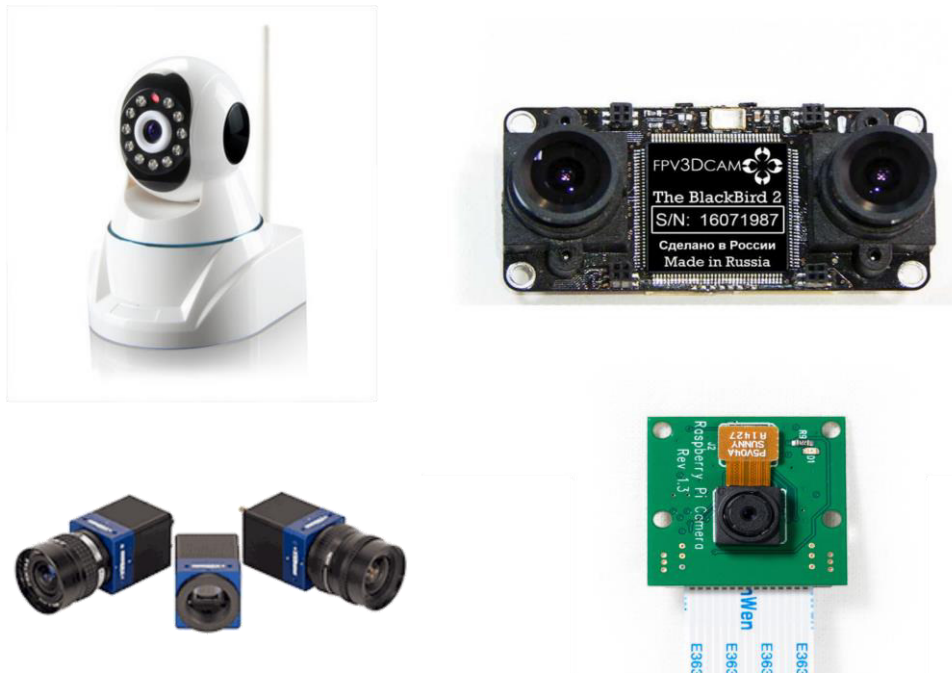
A câmera digital é um sensor de imagem que recebe a luz refletida por um objeto e a converte em sinal elétrico contendo a informação da cena visualizada. Os tipos mais comuns são do tipo CCD (*Charged-Coupled Device*) ou CMOS (*Complementary Metal-Oxide-Semiconductor*). O resultado da aquisição dos dados por câmeras é muito rico em conteúdo e seu uso como sensor para detecção de obstáculo em robótica tem sido explorado de forma intensa na última década (DISCANT, 2007). A Figura 12 mostra câmeras utilizadas em robótica móvel.

Devido a seu uso como sensor passivo, utilizando o espectro de luz visível, possuem limitações em condições de baixa iluminação, o que deu início a vastas pesquisas para desenvolvimento de filtros que aproveitem ao máximo as informações contidas em imagens (DISCANT, 2007). Mais detalhes de processamento de imagens e captura de características, serão exemplificados no Capítulo 4.

A radiação infravermelha representa uma faixa de frequências no espectro eletromagnético, que compreende os comprimentos de onda de 700 nm a 1 m. Sensores que se utilizam desse tipo de onda eletromagnética se beneficiam do fato de que todo corpo quente emite radiação infravermelha. Logo, podem ser capturados em ambiente sem nenhuma luz através, por exemplo, de uma câmera infravermelha (DISCANT, 2007).

Sensores infravermelhos de distância devem emitir o feixe de radiação e receber o sinal refletido no mesmo dispositivo, envolvendo técnicas de triangulação para a definição da distância medida. Devido à natureza da onda infravermelha, o feixe disparado pelo sensor possui uma abertura estreita e, em muitos casos, um sistema ótico com lentes é usado com o fim de tornar o local de incidência o mais pontual possível. Devido a esse aparato de correção do feixe, os dispositivos infravermelhos são considerados sensores direcionais (DISCANT, 2007). A Figura 13 mostra os sensores infravermelhos utilizados em robótica móvel.

Figura 12 – Câmeras utilizadas em robótica móvel.



Fonte: <http://www.cmucam.org/>

Atualmente, em plataformas móveis, a popularidade dos sensores que medem a distância tem crescido, e dentre os mais populares se destaca o sonar, pelo baixo custo, excelente desempenho e precisão. Sonar é acrônimo para *Sound Navigation And Ranging*, o que indica se tratar de um sensor baseado em som em uma frequência muito alta (ultrassônica). Entretanto, existem sonares com frequências muito pequenas (infrassônicas), mais usado em sistemas submersos (DISCANT, 2007).

Figura 13 – Sensores infravermelho de distância.

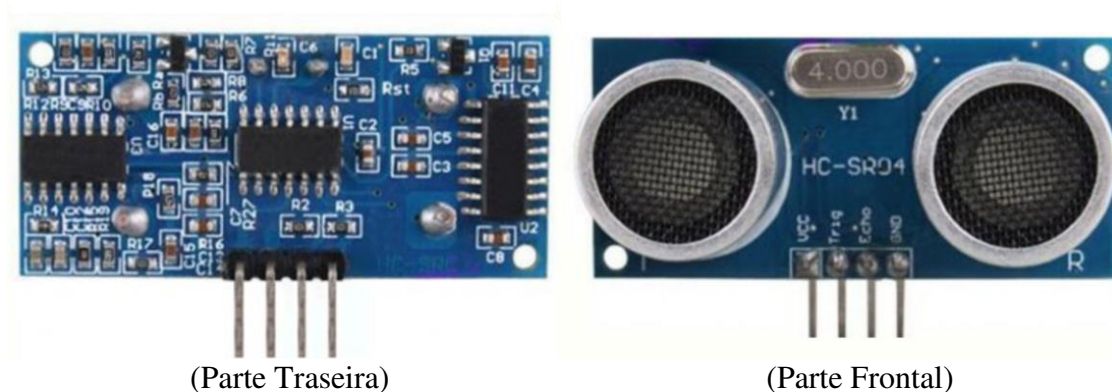


Fonte: <https://a.pololu-files.com/picture/0J6051.1200.jpg?4567baeed058eddf63955f8fdf44bde>

O sonar possui o mesmo princípio de funcionamento para medição de distância que o infravermelho. Um elemento emissor dispara um trem de pulsos de ondas ultrassônicas para ser parcialmente ou totalmente refletido em objetos e detectado por um receptor no mesmo dispositivo. O tempo de voo, considerando a velocidade do som, pode ser então usado no cálculo da distância ao alvo. No entanto, o som se propaga em um feixe amplo em largura (cone acústico), o que dificulta a definição da localidade do obstáculo no campo de propagação do som (DISCANT, 2007). Na Figura 14 está ilustrado um sonar popularmente utilizado em robótica, destacando seu circuito para detecção da onda de som.

Neste trabalho foram usados dois sensores infravermelhos e dois sonares. Foi utilizada uma técnica de fusão dos dados de ambos os sensores com o objetivo de aproveitar as vantagens de cada dispositivo, sendo mais bem exemplificada no Capítulo 5.

Figura 14 – Sonar comumente utilizado em robótica móvel.



Fonte:

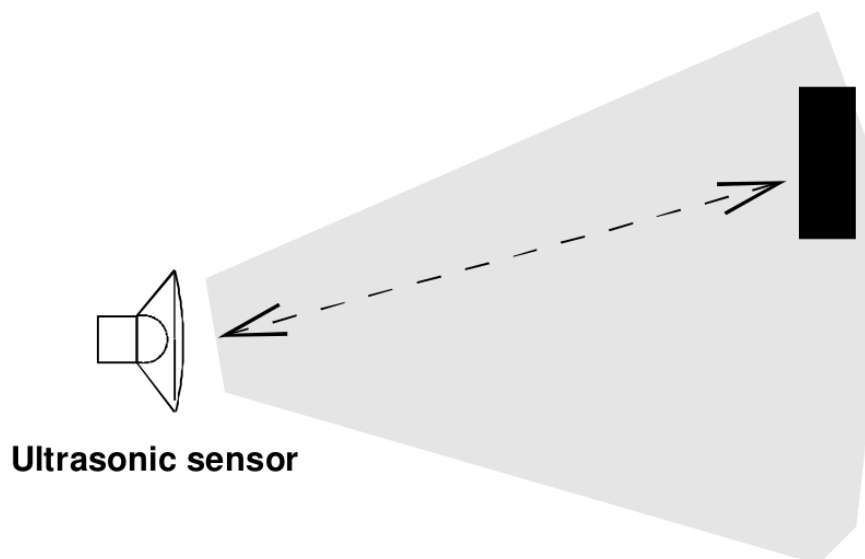
<https://nebula.wsimg.com/obj/QUYwNEVCRjVBNkY0NjJDOTY4QzA6MmYyYjUzNzUxYmFkMzljMTcxNzY4NWM3MDBiYWZhZTI6Ojo6OjA>

## 2.7 Fusão Sensorial

O uso de um único sensor na aquisição da transdução de um fenômeno, geralmente, não é o suficiente para obter de forma consistente o valor da medida desejada em uma determinada aplicação. Além disso, como visto na seção anterior, as medidas adquiridas contêm ruídos que podem tornar a informação errônea ou mesmo incompleta. Essa incerteza quanto ao dado proveniente da medida, deve ser modelada e incorporada no algoritmo (VISSER, 1999).



Figura 15 – Sonar mostrando o cone acústico na medição da distância.



Fonte: (VISSER, 1999)

Um exemplo do que pode ser considerada uma medida incompleta, seria o sensor ultrassônico de distância (sonar) utilizado largamente em robótica móvel, como ilustrado na Figura 15. Sua medida incorpora a incerteza subordinada à natureza do transdutor que, nesse caso, é um canhão emissor de ondas de som em frequência ultrassônica. A emissão do sinal ultrassônico cria o que se chama de cone acústico na região onde é enviado, o que dificulta a definição da orientação da medida, por ser indeterminada a posição do objeto dentro do cone (VISSER, 1999).

Devido aos ruídos de medição, é essencial que o sistema possa fazer a fusão de dados redundantes provenientes de múltiplos sensores. O termo redundante foi empregado com o sentido de que os múltiplos sensores, nesse caso, podem medir a mesma informação, a fim de reduzir o seu erro (VISSER, 1999).

Dessa forma, na maioria dos casos se faz necessário o uso de múltiplos sensores para a realização da fusão sensorial. A aplicação de fusão sensorial em um único sensor é possível quando múltiplos dados de um único sensor estão disponíveis. Como exemplo, têm-se as medidas obtidas uniformemente em intervalos de tempo sucessivos e, aplicando-se um filtro de Kalman, é atualizada a cada instante a estimativa da informação (linhas, imagem, polígonos etc.) (VISSER, 1999). Porém, o enfoque deste trabalho é em fusão sensorial de múltiplos sensores.

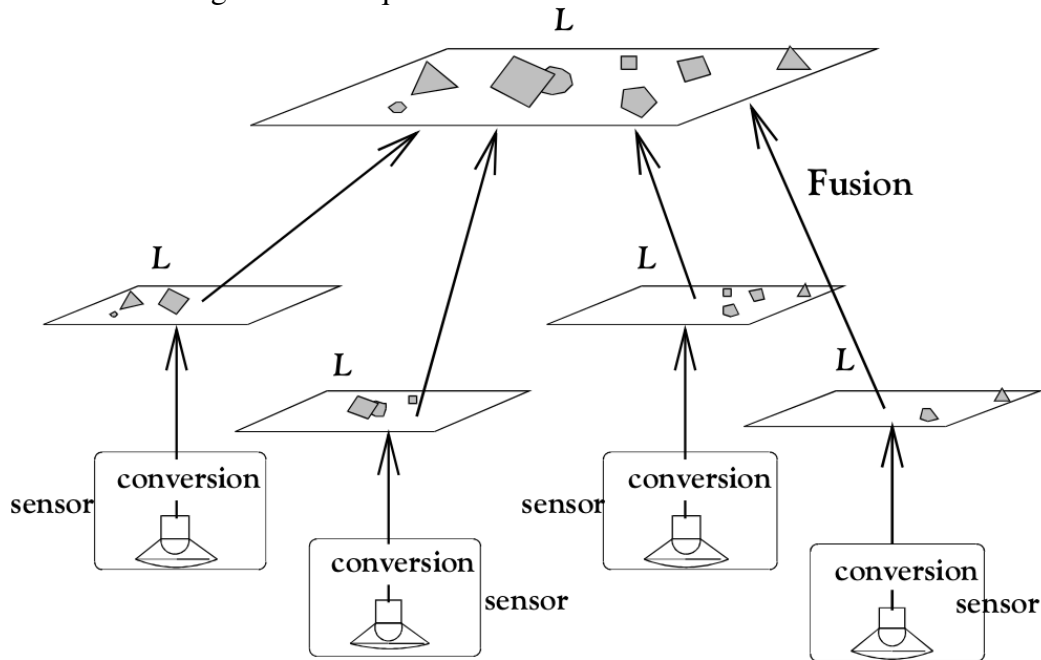
A fusão sensorial em múltiplos sensores se divide em categorias que descrevem a natureza da solução procurada (VISSER, 1999):

- a) Fusão complementar – é empregada quando se tem diferentes sensores que fornecem a informação de forma parcial, o que ajuda a resolver um dado incompleto. Um exemplo bastante comum é o sistema de visão estereoscópico (visão binocular humana), onde são necessárias as imagens obtidas pelas duas câmeras para a formação da informação medida;
- b) Fusão competitiva – dois ou mais sensores podem medir o mesmo obstáculo ou característica, a fim de reduzir a incerteza a respeito daquela medida, ou mesmo sua natureza errônea. Para a redução da incerteza, uma mesma característica pode ser medida utilizando sensores com capacidades diferentes onde, as estimações dos parâmetros avaliados por esses dispositivos tende a reduzir o erro; e
- c) Fusão cooperativa – é definida quando um sensor depende da informação adquirida por um ou mais sensores para, enfim, estimar a característica desejada. Dois ou mais sensores com parâmetros de retorno diferentes podem cooperar para completar a informação, como por exemplo, um sensor de toque e um sonar utilizando fusão sensorial para determinar a curvatura de um objeto, onde o sensor de toque pode refinar a estimativa da curvatura do objeto percebida pelo sonar.

Muitos dos algoritmos de fusão encontrados na literatura são construídos considerando as aplicações específicas. Logo, para cada sistema robótico, se usa um método de fusão sensorial para resolver as incertezas das medidas dos sensores que compõem o robô. Neste trabalho, será apresentado o método específico utilizado para um par de sensores infravermelho e um par de sonares. Cada par de sensores diferentes são dispostos com defasagem de  $180^\circ$  (um par de costas para o outro) a fim de cobrir toda a área ao redor do robô ( $360^\circ$ ), comandando uma varredura em todo o entorno do robô para aquisição de medidas com uma rotação do sistema de medição de apenas  $180^\circ$ .

Em VISSER (1999) é apresentada uma arquitetura genérica para fusão de dados sensorial em três níveis, conforme ilustrado na Figura 16. No primeiro nível se encontram os sensores implementados fisicamente e que retornam valores brutos dos parâmetros dos objetos adquiridos que, nesta pesquisa, constituem a distância e o ângulo em relação ao sistema de coordenadas do robô. No segundo nível, todos os parâmetros medidos são convertidos a uma representação interna comum (abstrata e para ser manipulada computacionalmente), sendo fundamental para a fase seguinte que constitui o terceiro nível, onde as informações separadas são fundidas em um único mapa.

Figura 16 – Arquitetura de fusão sensorial em três níveis.



Fonte: (VISSER, 1999)

Através dessa arquitetura é possível a utilização dos três tipos de categorias de fusão mencionadas. A fusão cooperativa pode ser aplicada antes mesmo dos dados serem convertidos à representação interna comum como, por exemplo, a fusão das medidas de um sistema de câmeras estereoscópica, onde os parâmetros de ambas as câmeras devem formar a informação para o nível superior (VISSER, 1999). Já a fusão competitiva e também a complementar são os casos onde sensores sonar e infravermelho são usados para definir o parâmetro final ou a informação parcial, respectivamente, ambos ocorrendo do segundo para o terceiro nível da arquitetura (MEDEIROS, 2011).

## 2.8 Estruturação do ambiente interno

Este tópico trata dos níveis de estruturação em que o ambiente pode se encontrar em relação às características utilizadas pelo robô, as quais caracterizam o nível da dificuldade com que o robô irá realizar a tarefa. Tratando-se de ambientes internos, ocasionalmente, a estruturação se torna mais viável devido à pouca complexidade necessária tanto de *hardware* quanto de *software* para realização da tarefa pelo robô. Quanto mais estruturado for o ambiente é mais fácil dele executar sua função no ambiente.

### 2.8.1 Estruturação de ambientes

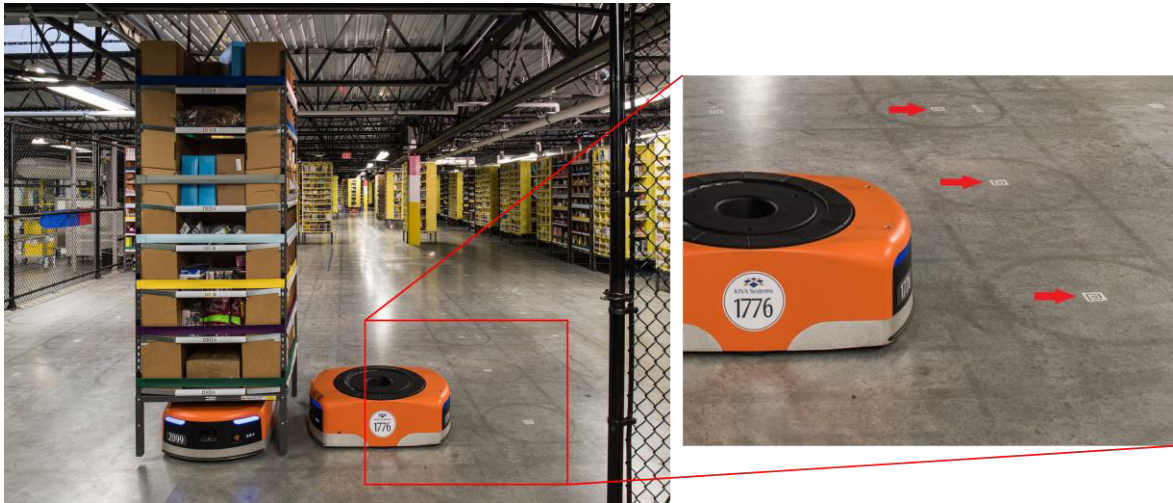
As estimações das características do ambiente pode ser um processo que demanda tempo e custo computacional. Para esses casos, certo nível de estruturação do ambiente é necessário, visando facilitar a implementação da tarefa e a execução do algoritmo. Em um ambiente denominado estruturado, existem certas suposições impostas fisicamente a fim de garantir que os movimentos e as medidas de sensores ocorram de acordo com os valores esperados.

Aplicações nesses tipos de ambiente geralmente não possuem um grande fluxo de pessoas presentes, sendo normalmente encontrados em locais como laboratórios, armazéns, indústrias etc. O uso de *tags*, linhas para percurso, marcos e cabos-guia são exemplos de estruturas utilizadas. Tais recursos servem para limitar as possibilidades de movimentos e medições se comparado a um ambiente não estruturado. Nos ambientes não estruturados o robô tem maior liberdade para realizar movimentos, porém a um custo computacional maior e de menor possibilidade de convergência do filtro, ou seja, fazer as estimações dos estados de forma a permanecer próxima do valor real. No caso da localização, o estado estimado é a pose do robô.

Ambientes não estruturados, atualmente, representam a maior parte das pesquisas em robótica móvel, devido ao fato de não necessitarem de modificações no ambiente para realização das tarefas do robô. Além disso, nem sempre é possível fazer a estruturação do ambiente e, em alguns casos, isso é impossível, com robôs de exploração em ambientes fora do planeta Terra. Nestas condições, é requerido que o robô esteja sempre pronto para situações adversas e que são difíceis de prever. A complexidade da tarefa pode ser ainda um problema a mais como são os casos de filtros que devem proporcionar uma solução para o problema SLAM onde, nem o mapa do local e nem a pose do robô são conhecidos, devendo ambos serem estimados.

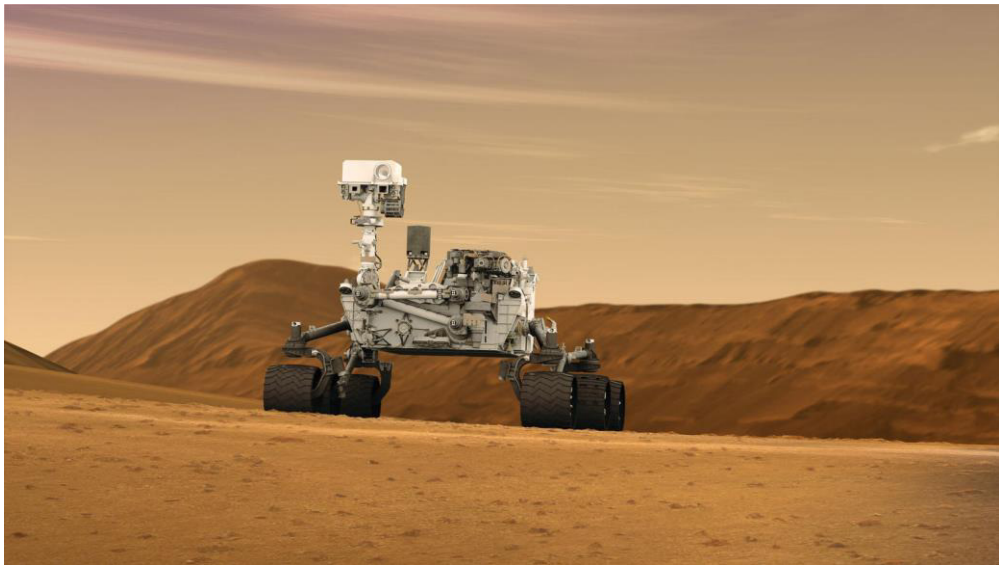
A Figura 17 mostra um exemplo de robô em um ambiente estruturado, enquanto a Figura 18 apresenta o robô em um ambiente totalmente não estruturado. Na primeira está uma foto do robô utilizado pela empresa Amazon para navegação e organização de mercadorias nas prateleiras de seus armazéns, sendo que *tags* são coladas no chão do ambiente para identificar aquele local no mapa (destacado pelo *zoom* feito na figura original). A Figura 18 apresenta o robô Curiosity enviado pela NASA (*National Aeronautics and Space Administration*) para coleta de dados na superfície do planeta Marte, cujas informações sobre o mapa e localização de objetos são desconhecidas.

Figura 17 – Robôs autônomos fazendo organização de mercadorias no armazém da empresa Amazon.



Fonte: <https://www.competitivefutures.com/wp-content/uploads/2015/06/amazon-warehouse.jpg>

Figura 18 – Robô Curiosity enviado pela NASA para exploração em Marte.



Fonte: [https://mars.nasa.gov/system/resources/detail\\_files/3512\\_msl20110526\\_MSL\\_Artist\\_Concept\\_PIA14165-full2.jpg](https://mars.nasa.gov/system/resources/detail_files/3512_msl20110526_MSL_Artist_Concept_PIA14165-full2.jpg)

### 2.8.2 Complexidade de ambientes internos

Ambientes internos são locais geralmente fechados, caracterizados por conter paredes, portas, objetos e poucas pessoas se movimentando. A presença ou não de pessoas no ambiente pode tornar o mapa dinâmico ou estático, respectivamente. Atualmente, muitos robôs que

realizam tarefas em ambientes internos simplificam o mapa de forma bidimensional, por conter pouca diferença da abordagem tridimensional e por limitação dos sensores utilizados.

Neste trabalho foi feita a simplificação do ambiente de forma estática, ou seja, todos os objetos não se movem, e não existem pessoas passando no local de movimento do robô. O caráter baixo custo do projeto foi o principal motivo dessa restrição.

### 3 Projeto do robô MILO

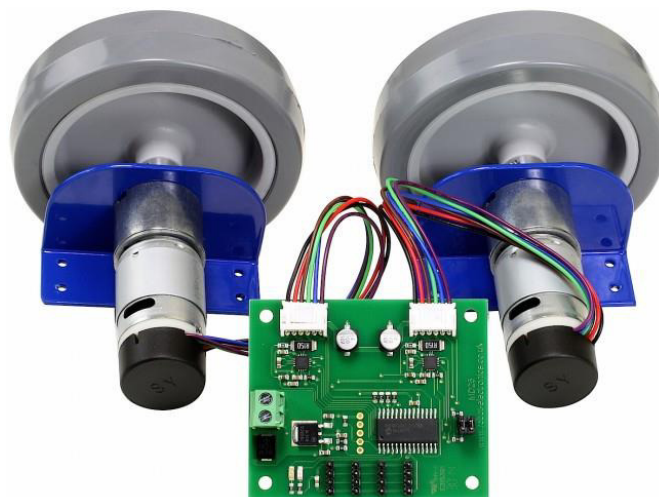
Neste capítulo será feita a apresentação do desenvolvimento do robô MILO (*Monte carlo Indoor LOCALization*) com o propósito de realizar as tarefas contidas nesse trabalho, descrevendo seus componentes. Toda a parte eletrônica e uma parte da estrutura mecânica do robô foi construído no LRC (Laboratório de Robótica Móvel e Comunicação Sem Fio).

Primeiro é descrita a parte mecânica, composta pela base e pela torre de sensores, feita em alumínio. Posteriormente é detalhada a parte eletrônica, no caso o *hardware* do sistema de controle, sensores e motores. Por último é feita a descrição em alto nível da parte computacional, mencionando os recursos e protocolos utilizados para funcionamento do robô e finalizando com a proposta de ambiente e obstáculos utilizados no trabalho.

#### 3.1 Parte mecânica

A base do sistema é sustentada por um chassi robótico feito em alumínio de modelo WOR-R1 comercializado pela loja online europeia Robot Electronics<sup>2</sup>. O modelo foi feito para ser usado com o *kit* de propulsão RD02, que já contém os motores de corrente contínua (CC) com rodas de tração que devem ser acopladas ao chassi e comandada pela placa de controle de velocidade. A Figura 19 ilustra o chassi montado com os motores e rodas do *kit* RD02.

Figura 19 – Plataforma robótica WOR-R1 junto com o kit RD02.



Fonte: <http://www.robot-electronics.co.uk/products/robot-kits-and-bases/wor-r1-research-platform-for-rd02.html>

---

<sup>2</sup> Site: <http://www.robot-electronics.co.uk/products/robot-kits-and-bases/wor-r1-research-platform-for-rd02.html>

O sistema funciona na configuração de acionamento diferencial centralizado de duas rodas com terceiro ponto de apoio livre. Esse tipo de configuração de base de apoio do robô em 3 pontos permite movimentos apenas de translação para frente ou para trás, apenas giros horários e anti-horários e a combinação de translação e giro em todas as direções. O chassi é vendido com uma roda livre de rodízio. No entanto, foi feita a troca para uma roda livre esférica, devido à baixa influência no movimento se comparada à anterior, uma vez que minimiza o atrito com o solo. As dimensões do chassi em milímetros são mostradas na Figura 20.

Uma modificação foi feita no chassi a fim de abrigar os sensores e o servo motor para rotação dos sensores. Denominou-se esse subsistema de torre de medição, mostrada na Figura 21(a). Foi desenvolvido o projeto contendo as dimensões da torre, sendo confeccionado em alumínio por uma empresa local para ser montada na parte superior do chassi, como mostrado na Figura 21(b). O objetivo dessa modificação foi tornar o eixo de rotação da torre coincidente com o centro de rotação da plataforma, além de elevar os sensores para evitar que

Figura 20 – Dimensões do chassi.

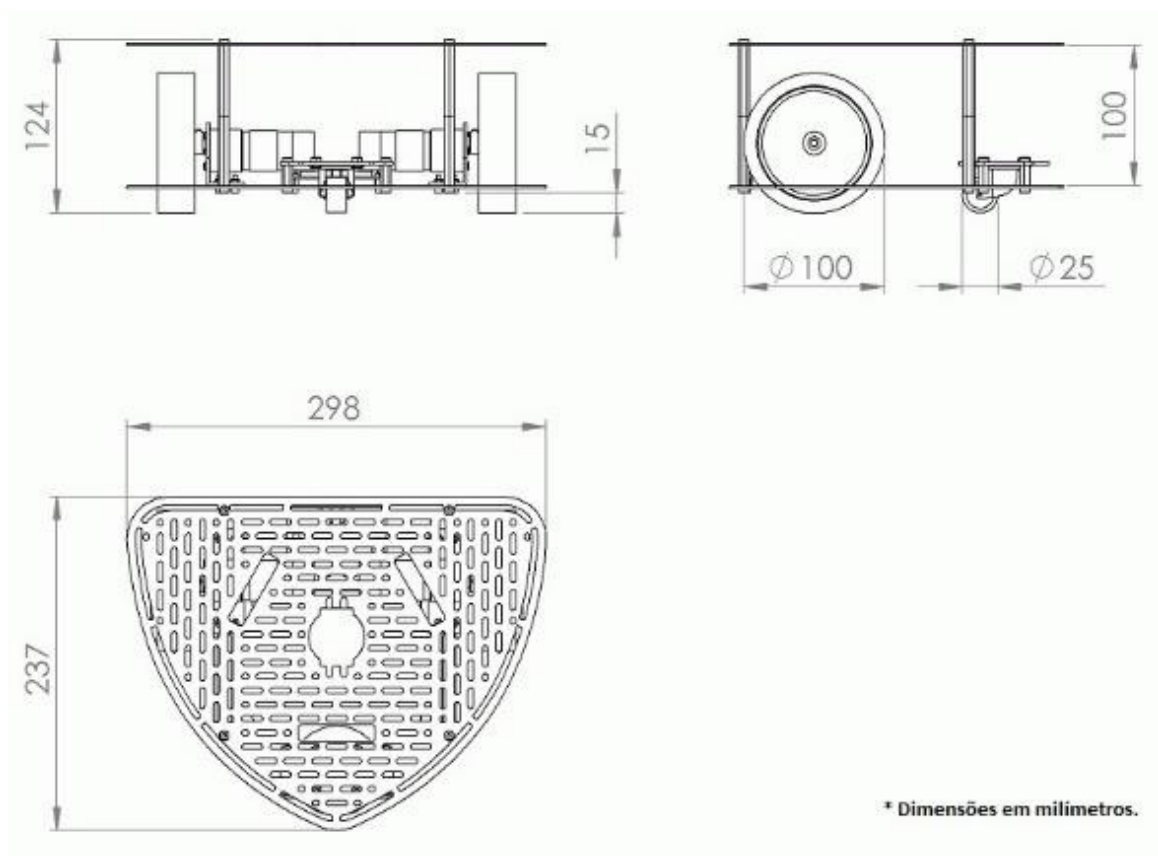
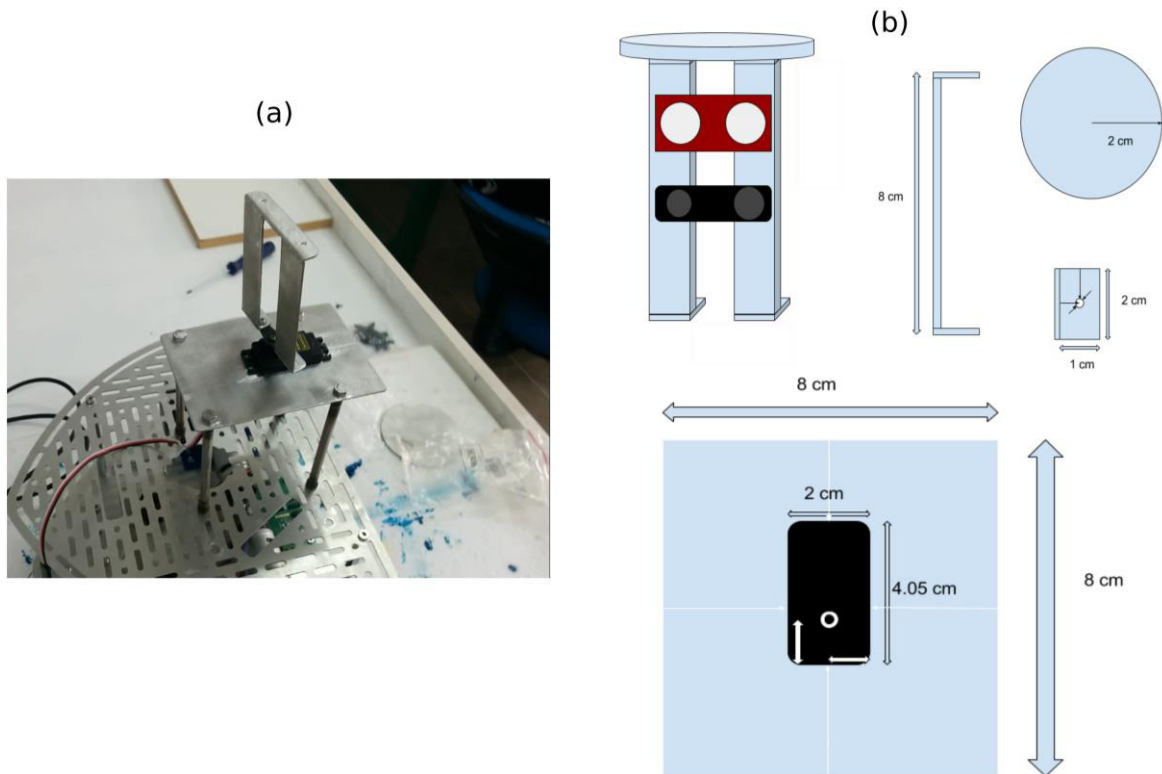




Figura 21 – Torre de sensores: (a) após a construção (b) projeto contendo as dimensões. Dimensões fora de escala.



Fonte: (AUTOR)

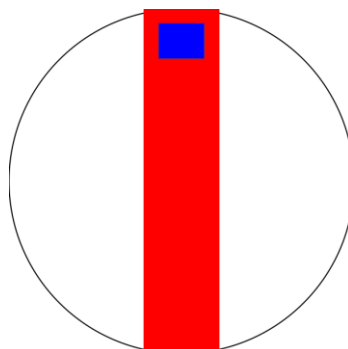
o feixe do sonar não toque na parte superior do chassi no momento das medições com o giro da torre.

A torre de medição é composta de duas peças mecânicas em alumínio: a base para o servo e as hastes que sustentam os sensores. A base para o servo é uma chapa de alumínio com um furo para encaixar o servo motor Futaba S3003, considerando o local de seu eixo de rotação para coincidir com o eixo de rotação do chassi. Todo esse sistema é elevado por 4 colunas (espaçadores) a uma altura de aproximadamente 10 cm, sendo cada coluna feita de um perfil cilíndrico de alumínio parafusado nas duas extremidades.

As hastes que sustentam os sensores são duas chapas de alumínio paralelas com altura o suficiente para abrigar os dois sensores como mostrado anteriormente na Figura 21(b). São fixadas com parafusos no eixo do servo motor para se moverem de acordo com a rotação do servo. Além disso, possui parafusada acima das hastes, uma chapa de alumínio em forma circular que auxilia mecanicamente a estabilidade das hastes quando estiverem em movimento e tem o propósito de abrigar uma *tag* identificadora.

Essa *tag* tem o formato como ilustrado na Figura 22. O propósito da imagem na *tag* é poder informar o centro de rotação e a orientação do robô como visto pela câmera acima do

Figura 22 – Imagem contida na *tag* para identificação da pose do robô.



Fonte: (AUTOR)

tablado. Como já foi dito anteriormente, o uso da *tag* só é necessário a fim de confirmar os resultados obtidos pelo algoritmo e facilitar os testes feitos para levantamento de ruídos no movimento e nas medições realizados pelo robô, evitando um trabalho extensivo de verificação métrica devido à quantidade de repetições. Essas incertezas devem ser incorporadas aos respectivos modelos do filtro MCL. Com isso, as estruturas de sustentação descritas acima formam a parte mecânica que é responsável por acomodar toda a eletrônica que controlará o sistema, sendo os componentes presentes descritos a seguir.

## 3.2 Parte eletrônica

Esse tópico tem o objetivo de apresentar os detalhes a respeito das placas eletrônicas que o robô MILO possui, os sensores, os motores e outros componentes relevantes ao escopo elétrico-eletrônico.

### 3.2.1. Placas eletrônicas

São três placas no total: o *Single Board Computer* (SBC) Raspberry Pi 3, a placa MD25 contida no kit RD02 e uma placa construída pelo autor.

#### 3.2.1.1 Placa Raspberry Pi 3

O elemento eletrônico principal constitui a placa Raspberry Pi 3 (RPi3) modelo B, a versão mais nova da família de SBCs da empresa Raspberry Pi Foundation até a data de realização do trabalho. É um computador contido em uma única placa do tamanho de um cartão de crédito, possuindo sistema operacional (SO) Raspbian, baseado no Linux<sup>3</sup>. As principais especificações estão descritas a seguir:

---

<sup>3</sup> Site: <https://www.raspberrypi.org/downloads/raspbian/>

- a) Processador ARMv8 Cortex-A53 *quad core* 1.2 GHz Broadcom BCM2837 64bits;
- b) Memória RAM 1 GB;
- c) Interface *Wireless LAN e Bluetooth Low Energy 4.1 (BLE)* BCM43438;
- d) 40 pinos GPIO (*General Purpose Input Output*) usados para propósitos geral e específicos;
- e) Quatro portas USB 2.0;
- f) Um Conector HDMI;
- g) Um Conector 8P8C (RJ45) *Ethernet*;
- h) Uma interface para câmera (CSI – *Camera Serial Interface*);
- i) Uma interface para display (DSI – *Display Serial Interface*); e
- j) Um slot para cartão micro SD;

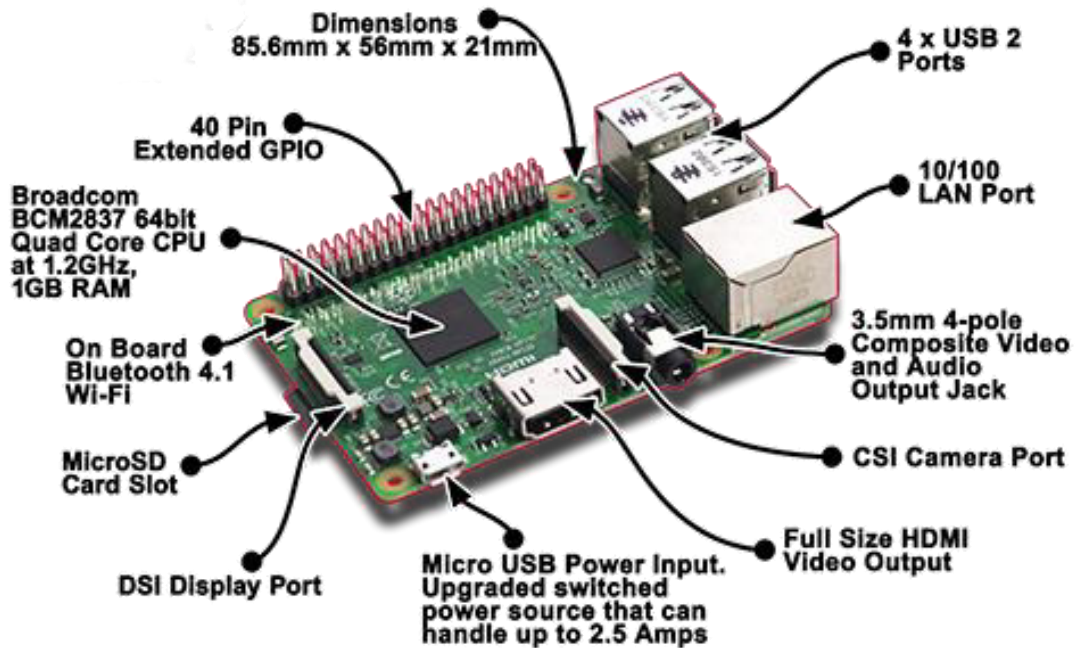
A Figura 23 mostra a localização dos componentes listados acima na placa. Sua central de processamento integra o que se chama de *System on a Chip* (SoC), uma nova tendência em placas microprocessadas. Nesse SoC se encontram os quatro núcleos ARMv8 Cortex-A53 com 32 kB e 512 kB de memória *cache* Nível 1 e Nível 2 respectivamente, além de um processador gráfico VideoCore IV, tudo encapsulado em um único *chip*.

O SO instalado no RPi3 é o Raspbian uma distribuição Linux baseada no Debian que foi otimizada para funcionar com a placa. Todo o código do robô foi desenvolvido em linguagem C. Nativamente, já está instalado o compilador `<gcc>` capaz de gerar executável que realizam as tarefas como movimentar o robô, leitura dos sensores e comunicação com o computador. Um ambiente *multithreading* foi desenvolvido para lidar com tarefas simultâneas, facilitado pelo alto poder de processamento da placa e quantidade de núcleos.

### 3.2.1.2 Placa MD25

A MD25 é uma placa de controle de motores CC baseada no microcontrolador PIC 16F873 e projetada para ser utilizada com os motores EMG30. No entanto, podem controlar outros motores CC, desde que obedecendo as conexões corretamente. Pode ser controlada via protocolos UART (*Universal Asynchronous Receiver-Transmitter*) ou I2C (*Inter-Integrated Circuit*), sendo essa escolha definida através de *jumpers* inseridos manualmente na placa. A Figura 24 a seguir ilustra a placa e seus recursos *onboard*.

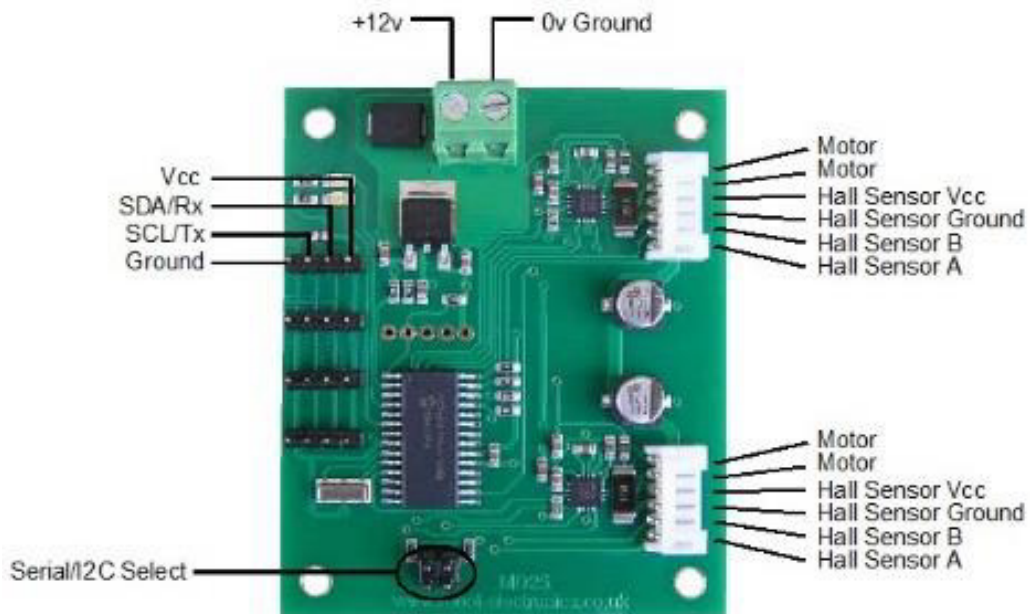
Figura 23 – Raspberry Pi 3 modelo B e seus periféricos.



Fonte: <http://d-intelligence.com/wp-content/uploads/2017/01/RPi3-B-intro.png>

Dentre os recursos presentes na placa, destacam-se a inversão do sentido dos motores, a possibilidade de leitura da corrente nos motores, o acionamento dos motores ao mesmo tempo ou de forma independente (diferencial) e, caso o motor em questão possua *encoders* baseados

Figura 24 – Placa de controle MD25 destacando barramentos de controle e conectores para os motores EMG30.



Fonte: <http://www.robot-electronics.co.uk/htm/md25tech.htm>

em efeito *hall*, a leitura do contador de rotação do motor. Neste trabalho a placa opera em modo I2C, se utilizando da alta taxa de transferência proporcionada pelo protocolo.

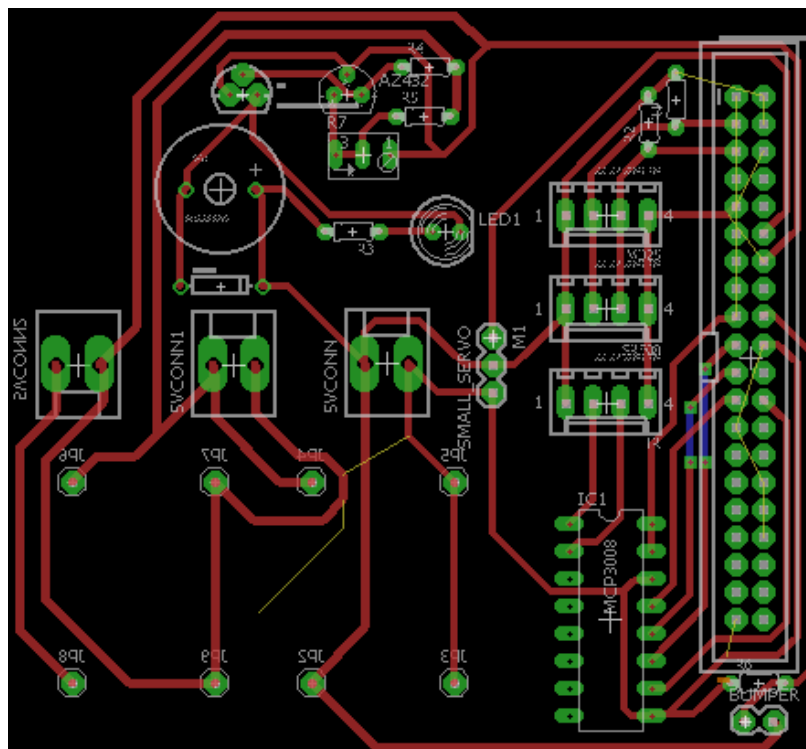
A robustez da placa acompanhada de bons motores, refletem em boa odometria e movimentação com razoável precisão do robô, atributos que facilitam o processo de localização no filtro MCL.

### 3.2.1.3 Placa de interface

A fim de proporcionar tensões reguladas, interconexões para diferentes sinais e acomodar dois circuitos importantes, foi projetada uma placa de circuito impresso denominada placa de interface. Essa placa foi construída a partir de uma placa de fenolite virgem. Seu circuito foi transposto à placa através do processo fotossensível, aperfeiçoado no LRC para prototipagem de projetos.

A Figura 25 mostra o *layout* da placa roteada no *software* Eagle CAD<sup>4</sup> e a Figura 26 mostra o esquemático de circuito correspondente.

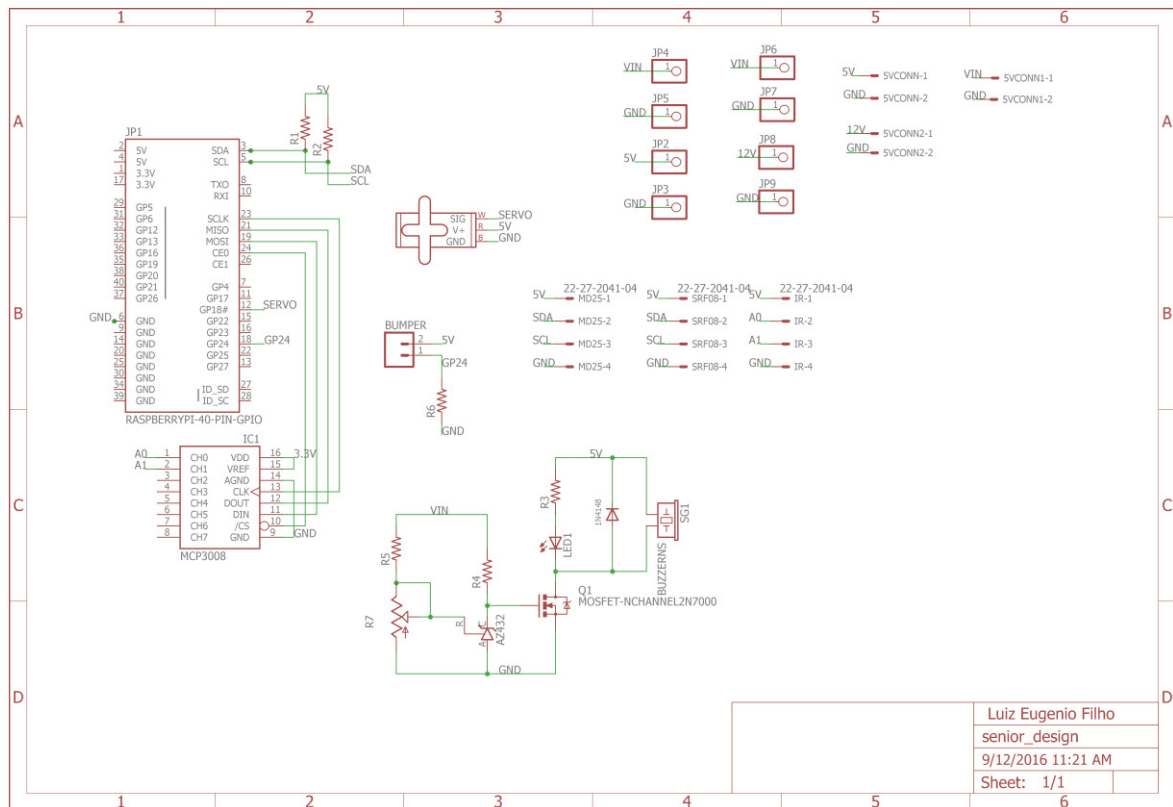
Figura 25 – *Layout* desenvolvido no Eagle CAD da Placa de Interface.



Fonte: (AUTOR)

<sup>4</sup> Site: <https://www.autodesk.com/products/eagle/overview>

Figura 26 – Esquemático do circuito presente na Placa de Interface.



Fonte: (AUTOR)

Um dos circuitos contidos na placa de interface é o MCP3008, um conversor analógico digital (A/D) com oito canais de 10 bits de resolução que se comunica através do protocolo SPI (*Serial Peripheral Interface*). Seu propósito é fazer a leitura das medições do sensor infravermelho (Seção 3.2.2), cuja saída é uma tensão analógica, relacionada à medida de distância, que deve ser discretizada e transmitida para RPi3.

O circuito com componentes discretos localizado na parte inferior da Figura 26 é um detector de bateria com nível baixo de tensão, que emite alerta sonoro e visual quando a tensão de alimentação do robô atinge o valor de aproximadamente 14.5 V. Com funcionamento em torno do CI (Circuito Integrado) AZ432, esse circuito consome baixíssima potência enquanto monitora a tensão, sendo ideal para essa aplicação. Dessa forma, a finalidade desse circuito é informar que a bateria (Seção 3.2.4) necessita ser carregada por atingir um nível considerado no projeto como crítico.

Através da modificação de um potenciômetro, o regulador de tensão AZ432<sup>5</sup> de baixo consumo cria uma tensão de referência diferente na saída, configurando a tensão para o limite do chaveamento do transistor quando a tensão de alimentação for aproximadamente 14.5 V. Nessa situação um LED (*Light-Emitting Diode*) e um *buzzer* presentes nesse circuito são acionados, avisando a necessidade de carga da bateria na forma visual e sonora.

Os últimos componentes são dois módulos conversores CC-CC do tipo *buck*, responsáveis por garantir as tensões de 5 V e 12 V a partir da tensão de alimentação<sup>6</sup>. A tensão de 5 V fornece energia para todos os componentes no robô exceto a placa MD25, cuja energia provém da tensão regulada de 12 V. Além disso, a placa ainda dispõe dos dois resistores de *pull up*, necessários para a comunicação I2C, um resistor de *pull down* para leitura de um *bumper switch* conectado a um GPIO do RPi3 e todos os conectores para comunicação e alimentação dos sensores, servo motor, MCP3008 e o RPi3.

A Figura 27 a seguir ilustra os componentes interconectados de forma simplificada, substanciando o que foi dito até agora a respeito da Placa de Interface.

### 3.2.2 Sensores

Nesta seção serão apresentadas as características específicas dos sensores utilizados no robô MILO, que são o sonar, o infravermelho, os *encoders* acoplados aos motores CC e o *bumper switch*.

#### 3.2.2.1 Sonar SRF08

O sonar SRF08 fabricado pela Devantech<sup>7</sup> pode medir distâncias de 3 cm a 6 m, sendo o resultado da medida adquirida através da medida do tempo de voo da onda ultrassônica. Essa medida pode ser retornada na forma de valor da distância em centímetros, polegadas ou na forma original (tempo de voo em  $\mu$ s). Sua alimentação é de 5 V típico, consumindo 15 mA, e se comunica se dá por meio do protocolo I2C, gerenciado por um microcontrolador PIC 16F872 interno que permite o uso de 16 SRF08 no mesmo barramento I2C. Para este trabalho, apenas dois são necessários, um de costas para o outro (180° defasados). O circuito do sensor, dispõe também de um LDR (*Light Dependent Resistor*), porém seu uso não foi necessário. A Figura 28 a seguir mostra a parte da frente (elementos de envio e recepção do trem de pulsos da onda) e de trás (circuito do sonar) do sensor.

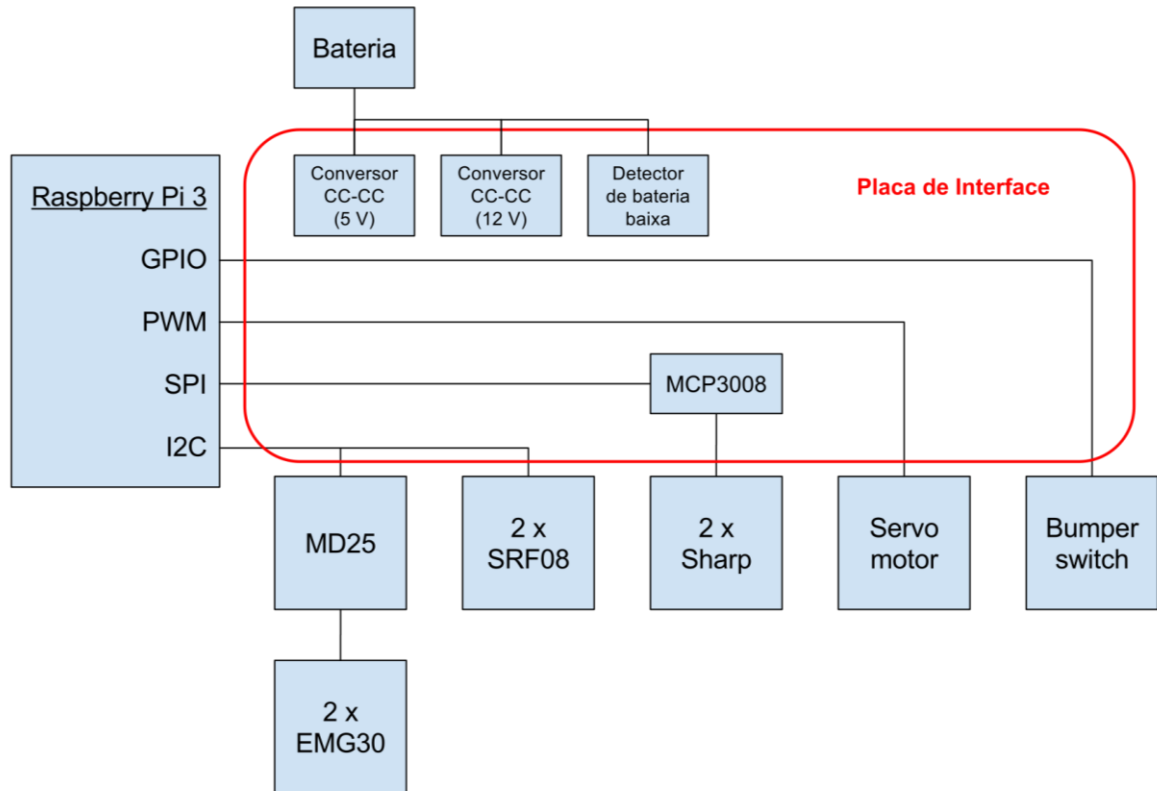
---

<sup>5</sup> Site: <https://www.diodes.com/assets/Datasheets/AZ432.pdf>

<sup>6</sup> Site: <http://www.eletródex.com.br/modulo-ultra-reduzido-step-down-3a.html>

<sup>7</sup> Site: <http://www.robot-electronics.co.uk/products/sensors/ultrasonics/srf08-i2c-range-finder.html>

Figura 27 – Diagrama em blocos da Placa de Interface e suas conexões.



Fonte: (AUTOR)

Neste trabalho foi usado o retorno da medida como tempo de voo por conferir maior precisão na medida (distância com casas decimais). Esse valor, que representa o tempo de ida e volta da onda de som até o obstáculo, juntamente com a informação da velocidade do som

Figura 28 – Sonar SRF08 destacando seu circuito.



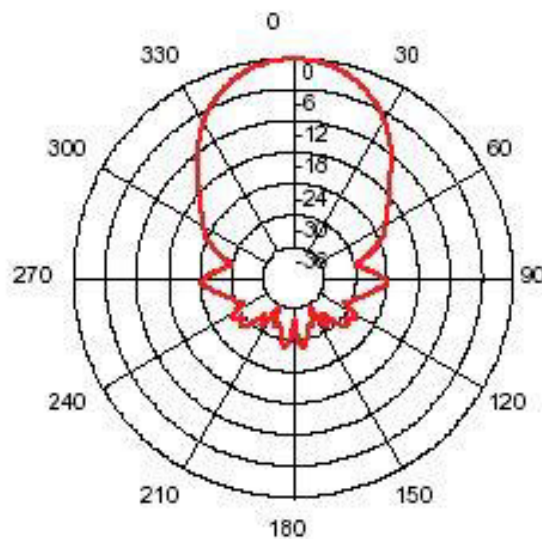
Fonte: <http://www.robot-electronics.co.uk/srf08-i2c-range-finder.html>



ao ar livre, é possível encontrar a distância ao obstáculo.

A Figura 29 mostra o diagrama de irradiação da onda de som, fornecido pelo fabricante<sup>8</sup>. Nessa figura nota-se o ganho do sinal ficando mais fraco em direções laterais ao centro do cone emitido pelo sensor. O ângulo de abertura da onda de som para esse sensor é de aproximadamente 55°.

Figura 29 – Diagrama de irradiação para o SRF08.



Fonte: <http://www.robot-electronics.co.uk/files/t400s16.pdf>

### 3.2.2.2 Sensor infravermelho GP2Y0A02YK0F

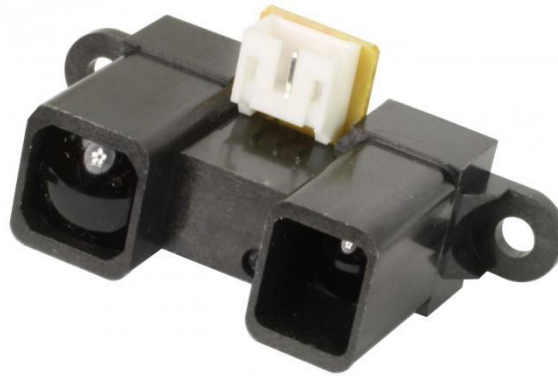
O sensor infravermelho GP2Y0A02YK0F fabricado pela SHARP (doravante denominado apenas Sharp) pode medir distâncias entre 20 cm e 150 cm. Sua alimentação é de 5 V típico, consumindo 33 mA, e sua comunicação se dá de forma analógica, motivo pelo qual se usou um conversor A/D para interpretar sua leitura, necessária em virtude do RPi3 não possuir um módulo *onboard* para realizar essa tarefa. A Figura 30 ilustra o sensor.

O sensor se utiliza de um PSD (*Position Sensitive Detector*) em combinação com um IRED (*Infrared Emitting Diode*), além de um circuito para processar o sinal<sup>9</sup>. A detecção do objeto é feita por uma triangulação do sinal, visto anteriormente no Capítulo 2. A Figura 31 mostra a relação de distância detectada e tensão na saída do sensor.

<sup>8</sup> Site: <http://www.robot-electronics.co.uk/htm/srf08tech.html>

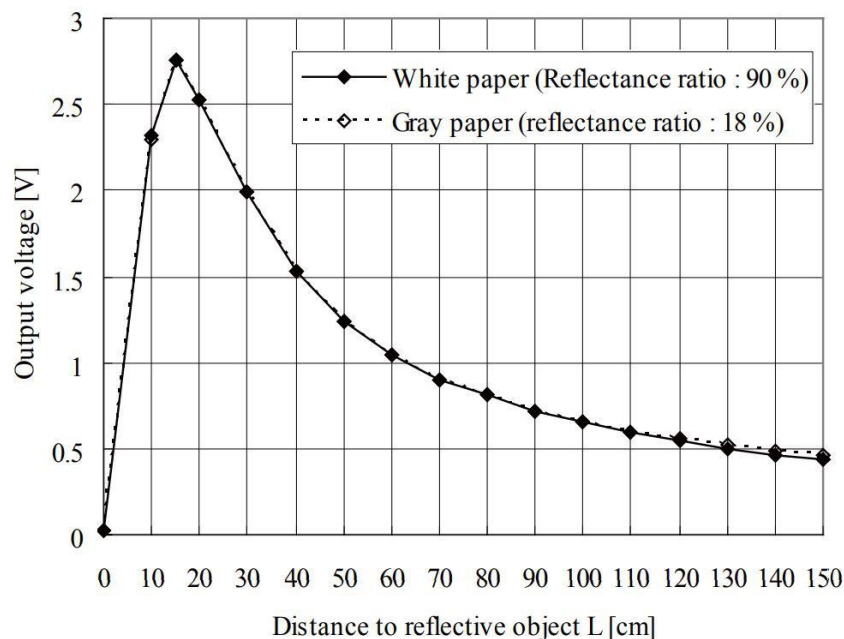
<sup>9</sup> Site: [http://www.robot-electronics.co.uk/files/gp2y0a02\\_e.pdf](http://www.robot-electronics.co.uk/files/gp2y0a02_e.pdf)

Figura 30 – Sensor infravermelho da Sharp.



Fonte: <http://www.robot-electronics.co.uk/products/sensors/infrared-range/gp2y0a02yk.html>

Figura 31 – Gráfico da distância ao objeto pela tensão na saída do sensor infravermelho.



Fonte: [http://www.sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a02yk\\_e.pdf](http://www.sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a02yk_e.pdf)

### 3.2.2.3 Encoders dos motores CC

Acoplados aos eixos de cada um dos dois motores CC estão os *encoders* que medem a rotação, através de sensores de efeito *Hall*. Os sensores utilizados nesse trabalho acompanham os motores EMG30, comercializados pela Robot Electronics, são alimentados pela MD25 com uma tensão regulada de 5 V. Contabilizam a medição através de um contador de 12 passos para atingir uma revolução completa do eixo do motor. Devido a um redutor que liga o eixo à roda ser de 30:1, um giro do eixo proporciona 360 contagens em uma volta da roda. O

reductor tem a finalidade de aumentar o torque do motor, diminuindo a velocidade máxima de acionamento. Com o raio da roda definido é possível calcular a distância percorrida pela leitura do *encoder*, considerando que o seu perímetro corresponde a 360 contagens. A Figura 32 ilustra um *encoder* se utilizando de um sensor de efeito *Hall*.

Figura 32 – *Encoder* de efeito *Hall*.



Fonte: <https://www.epictinker.com/v/vspfiles/photos/FIT0185-4.jpg>

#### 3.2.2.4 *Bumper switch*

O último sensor tem o objetivo de assegurar o robô de choques frontais, quando em deslocamento. Em se tratando de um *bumper switch* que fica à frente do robô, quando acionado, para toda a atividade que o robô esteja fazendo finalizando o movimento e o programa em execução. Um *thread* foi implementado para verificar o estado do GPIO ligado ao *bumper*. A Figura 33 a seguir mostra a chave de contato.

### 3.2.3 Motores

Existem operando no robô dois motores CC e um servo motor. Esse tópico traz os detalhes dos dois tipos de motores.

#### 3.2.3.1 Motor CC EMG30

Comercializado pela Robot Electronics, o EMG30 é um motor CC equipado com *encoder* e caixa de redução de 30:1 ideal para aplicações em robôs de pequeno e médio porte. Dois motores são utilizados em conjunto com a MD25. As especificações são listadas a seguir e logo após a Figura 34 mostra os dois motores utilizados:

- a) Tensão de operação: 12 V
- b) Torque nominal: 1.5 kg/cm
- c) Velocidade nominal: 170 rpm
- d) Corrente nominal: 530 mA
- e) Velocidade mínima (a vazio): 1.5 RPM
- f) Velocidade máxima (a vazio): 200 RPM
- g) Corrente a vazio: 150 mA
- h) Corrente de rotor bloqueado: 2.5 A
- i) Potência de saída: 4.22 W

Figura 33– *Bumper switch*.



Fonte: [https://http2.mlstatic.com/D\\_Q\\_NP\\_879265-MLA25558370268\\_052017-Q.jpg](https://http2.mlstatic.com/D_Q_NP_879265-MLA25558370268_052017-Q.jpg)

Figura 34 – Motores EMG30.



Fonte: <http://www.robot-electronics.co.uk/rd02-12v-robot-drive.html>

### 3.2.3.2 Servo motor S3003

O servo motor utilizado é um Futaba S3003 com abertura de aproximadamente 180°, com engrenagens de plástico com torque variável entre 3.2 kg/cm e 4 kg/cm<sup>10</sup>. Para seu funcionamento é necessário receber um sinal modulado em largura de pulso (PWM – *Pulse Width Modulation*).

Esse sinal, segundo o manual do fabricante, deve possuir largura de pulso entre 1520 µs e 1900 µs, porém, dados experimentais (valores empíricos) mostram que os valores necessários para realizar um giro de 180° são de 500 µs a 2020 µs. Esse tipo de observação é comum em servo motores que possuem engrenagens de plástico, uma vez que podem deslizar e desajustar o servo mecanismo. A Figura 35 mostra o servo motor utilizado.

Figura 35 – Servo motor S3003.



Fonte: <http://www.gpdealera.com/cgi-bin/wgainf100p.pgm?I=FUTM0031>

O RPi3 não possui *hardware* específico que possa gerar um sinal modulado em PWM. Portanto, foi desenvolvido um método para utilizar um GPIO para gerar o sinal via *software*, detalhes desse desenvolvimento no Capítulo 5.

### 3.2.4 Outros detalhes da eletrônica embarcada

Para fornecer energia ao robô é feito o uso de baterias, que são do tipo Lítio Polímero (LiPo) de 4 células recarregáveis, totalizando 14.8 V nominal em seus terminais (16,8 V quando totalmente carregadas). Essas baterias possuem uma boa relação entre a carga que armazenam por peso e são utilizadas em aeromodelos<sup>11</sup>. O LRC dispõe de duas baterias LiPo

<sup>10</sup> Site: <https://servodatabase.com/servo/futaba/s3003>

<sup>11</sup> Site: <https://colecões.mercadolivre.com.br/modelismo-profissional/aeromodelismo/bateria-lipo-thunder-power-4s>

da marca Thunder Power, uma de 2600 mAh e a outra de 4400 mAh, ambas servindo aos trabalhos envolvendo robótica móvel. Para a carga dessas baterias é utilizado um carregador inteligente da marca Bantam, que faz esse processo de forma balanceada<sup>12</sup>. Quando totalmente carregadas, a tensão nos terminais das baterias passa dos 16 V, com taxa de descarga mais lenta até a tensão nominal e quando passam de 14.5 V com o uso, tendem a descarregar mais rápido.

Dessa forma, foi necessário projetar um circuito na placa de interface para alertar quando a tensão de entrada chegar a 14.5 V, de modo a evitar que a bateria passe da tensão de ponto crítico, situado abaixo dos 14 V, onde a bateria entra em uma carga acelerada e, abaixo de 12 V, o fabricante não garante a possibilidade de carga. Além disso, adverte que caso alcance uma tensão em seus terminais em torno de 7 V, existe o perigo de ela explodir. A Figura 36 abaixo mostra as duas baterias utilizadas no LRC.

Figura 36 – Baterias de LiPo utilizadas nos experimentos com o robô.

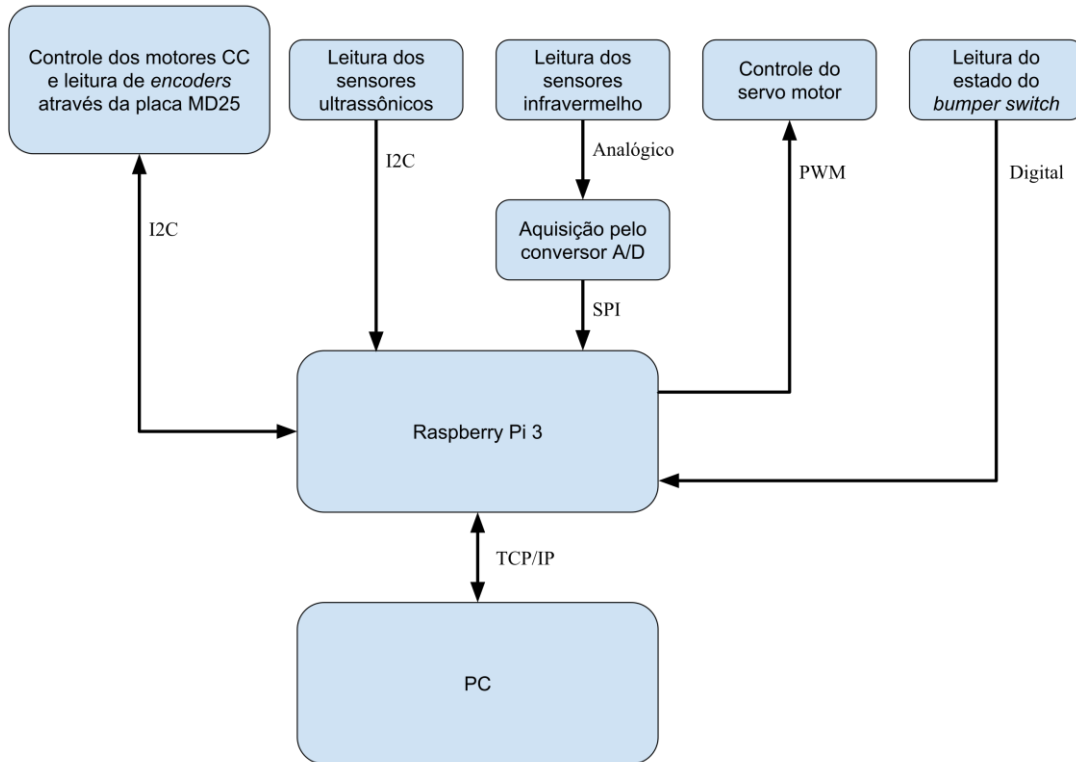


Fonte: (AUTOR)

<sup>12</sup> Site: <https://produto.mercadolivre.com.br/MLB-986371106-carregador-digital-chargerdischarger-e-station-bantam-902- JM>

Sumarizando os componentes físicos presentes no robô, a Figura 37 a seguir mostra um diagrama de forma simples como ocorre a intercomunicação desses componentes.

Figura 37 – Diagrama em blocos simplificado de intercomunicação dos elementos do sistema robótico.



Fonte: (AUTOR)

### 3.3 Parte computacional

Esta seção é dedicada a descrever o funcionamento dos recursos computacionais utilizados nesse trabalho. A maioria dos recursos está disponível em bibliotecas nativas do SO e, o que não havia disponível, foi desenvolvido também em forma de biblioteca para facilitar entendimento do código que se tornou extenso ao longo do desenvolvimento do trabalho.

O uso do RPi3 no robô é feito de forma *headless*, ou seja, não existe monitor, teclado ou mouse plugados nos conectores HDMI e USB, respectivamente. Logo, é necessária que a interação com ele aconteça de uma forma diferente da convencional, ou seja, se utilizando da interface gráfica. A alternativa utilizada foi o SSH (*Secure Shell*), um protocolo criptografado que utiliza a rede local para *login* remoto confiável e seguro através da linha de comando (*shell*) sabendo-se somente o *Internet Protocol* (IP) e a senha do usuário que se deseja conectar no RPi3. Mesmo não se utilizando da interface gráfica para a utilização do RPi3, a desativação desse recurso não foi feita, a fim de poder utiliza-la após o trabalho ser concluído.

O protocolo é chamado por um cliente SSH instalado na máquina *host*, no caso um computador pessoal (PC – *Personal computer*). A máquina que se deseja alcançar, no caso o RPi3, também deve possuir um cliente SSH instalado, no entanto, ambos os computadores utilizados são baseados em Linux e já possuem os *softwares* instalados nativamente. Com isso, através do *shell* no PC, um único comando é capaz de estabelecer a conexão, seguida da fase de autenticação e confirmação da chave de encriptação.

Uma vez conectado, o PC pode navegar normalmente no RPi3, e através do *shell*, foram criados programas para executar as tarefas físicas do robô.

Em se tratando de linguagem C, recursos de *hardware* são acessados através de manipulação de arquivos, como escrita e leitura. As bibliotecas nativas, cujos protótipos são localizados em `<sys/ioctl.h>`, `<linux/i2c-dev.h>`, `<linux/spi/spidev.h>` e `<fcntl.h>` contém as funções necessárias para habilitar as comunicações seriais como I2C e SPI acessadas nos

Figura 38 – Disposição dos pinos e suas funções no RPi3.

Raspberry Pi 3 GPIO Header				
Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I <sup>2</sup> C)		DC Power 5v	04
05	GPIO03 (SCL1 , I <sup>2</sup> C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I <sup>2</sup> C ID EEPROM)		(I <sup>2</sup> C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Rev. 2  
29/02/2016

www.element14.com/RaspberryPi

Fonte: [https://www.element14.com/community/servlet/JiveServlet/previewBody/73950-102-11-339300/pi3\\_gpio.png](https://www.element14.com/community/servlet/JiveServlet/previewBody/73950-102-11-339300/pi3_gpio.png)



GPIOs indicados conforme a Figura 38, que mostra a disposição dos pinos e suas funções no RPi3.

As funções que não existem em bibliotecas nativas foram escritas no código fonte `<lowlevelfunctions.c>` e seus protótipos em `<lowLevel.h>`. Utilizando esses arquivos na compilação, é possível escrever um terceiro código fonte contendo a ação a ser executada, incluindo no cabeçalho `<lowLevel.h>`. Devido à extensa quantidade de instruções de compilação, criou-se um arquivo `'makefile'` contendo um *script* de compilação que pode ser invocado utilizando o comando `'make'`.

Incluindo-se as bibliotecas, cujos cabeçalhos são `<sys/socket.h>` e `<netinet/in.h>`, ambas nativas, foi desenvolvida programação em *socket* utilizando o *Transmission Control Protocol* (TCP/IP), criando um processo servidor no RPi3 e um processo cliente no PC executado no ambiente MATLAB. O filtro probabilístico está implementado no MATLAB, logo, dados de sensores e controle de movimento são passados através desse *link* TCP/IP.

A câmera utilizada para confirmação dos dados foi controlada através de *Wi-Fi* por uma *Application Programming Interface* (API) não oficial desenvolvida por terceiros. A API está escrita em linguagem Python, de fácil implementação e contendo funcionalidades suficientes para a aplicação.

De modo geral, foi descrito os principais aspectos da implementação lógica do funcionamento do robô. Detalhes em pseudocódigo e fluxogramas de execução são exemplificados com mais detalhes no Capítulo 5.

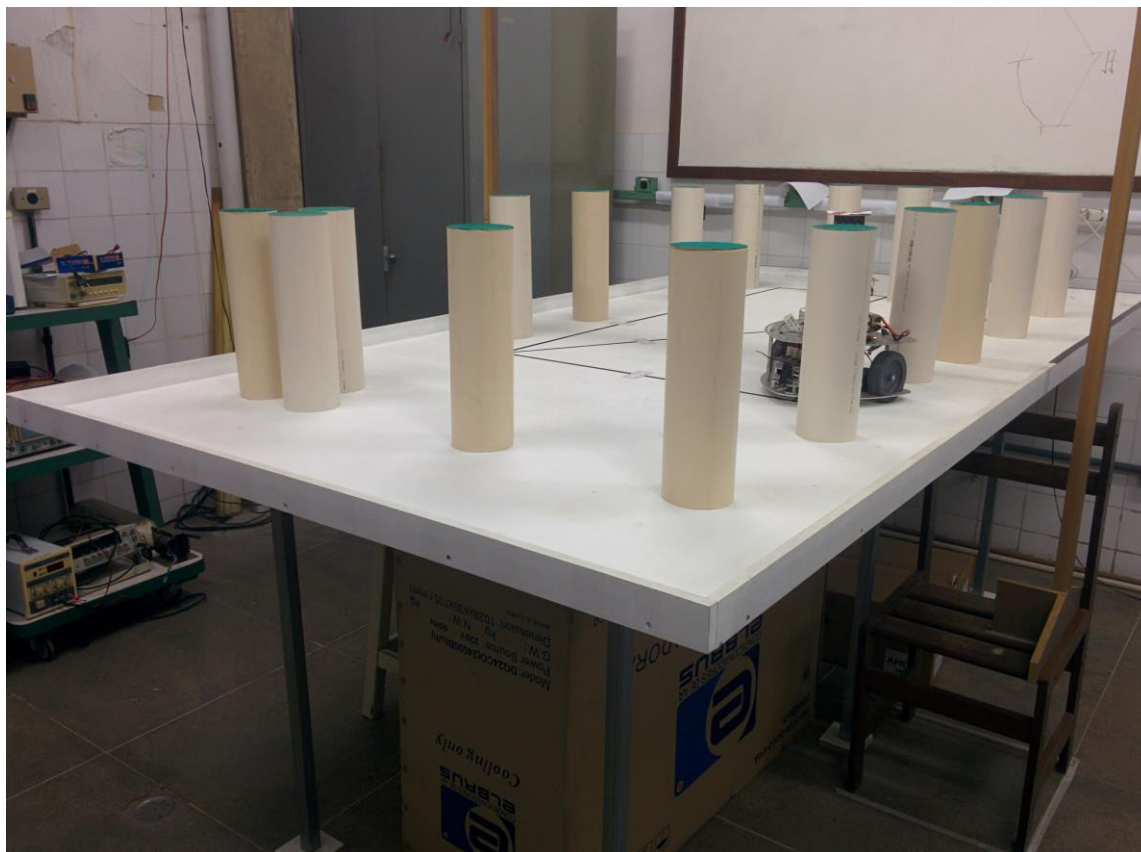
### **3.4 Proposta de ambiente interno e obstáculos que formam o mapa**

O LRC, local de desenvolvimento da pesquisa deste trabalho de monografia, dispõe de 3 ambientes com 4 paredes e uma porta cada um. No entanto, devido ao uso desses ambientes frequentemente por alunos em atividades de pesquisa e de ensino, e a existência de muitas mesas e cadeiras, o espaço livre disponível para locomoção do robô no chão do laboratório é limitado.

Para prover o LRC com o ambiente necessário a esta e outras duas pesquisas em robótica móvel já finalizadas (RIBEIRO, 2017 e PINTO Jr, 2017), projetou-se uma mesa plana para acomodação do robô e dos obstáculos, com propósito de ajudar nos testes e implementação do algoritmo de forma controlável. Esse tablado foi construído em madeira MDF com seis pernas em perfil de alumínio para evitar que o plano envergue. A Figura 39 mostra a mesa já montada no LRC.

A mesa, com dimensões de 278 cm por 143 cm, além de servir para o desenvolvimento destes três primeiros trabalhos citados, será de uso para pesquisa futuras de outros robôs que

Figura 39 – Tablado montado para realização dos experimentos com o robô.



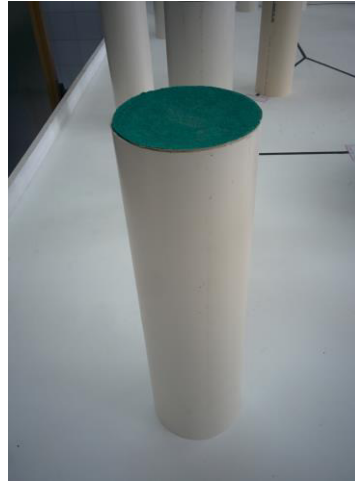
Fonte: (AUTOR)

farão acréscimos a essas pesquisas iniciais. O tablado possui, ainda, uma parede de 3 cm levantada nas bordas a fim de impedir quedas acidentais de robôs em fase de testes e de operação após a finalização do trabalho.

Os obstáculos são cilindros de 35 cm de altura e 10 cm de diâmetros (Figura 40), utilizados como marcos artificiais (*landmarks*) e feitos de corte em canos PVC. No topo de cada cilindro foi colocado papéis duro e pintado de verde para identificação de seu centroide por meio de uma câmera instalada acima do tablado, a qual foi empregada na aquisição real dos obstáculos e robô para confirmação da eficiência do filtro em relação aos dados estimados.

No próximo capítulo, serão apresentados detalhes sobre a estrutura construída para aquisição de imagens do tablado e o papel da câmera aliado com o *software* para detectar os obstáculos e o robô.

Figura 40 – Cilindros usados como marcos artificiais.



Fonte: (AUTOR)

## 4 Sistema de captura dos elementos reais presentes no ambiente

Neste capítulo, descrevem-se os detalhes a respeito do processamento de imagem feito para capturar a pose do robô MILO e as localizações dos obstáculos no tablado.

É importante destacar que o objetivo do trabalho não envolve o aperfeiçoamento e nem o desenvolvimento de métodos para processamento de imagem, contudo, a implementação mostrada ao longo deste capítulo foi usada como ferramenta para acelerar alguns processos, que, do contrário seriam feitos manualmente. Além disso, tem uma importante função automática de servir para as capturas de informações do ambiente (poses do robô e coordenadas dos obstáculos) a serem usadas na comparação com os resultados obtidos pelo filtro MCL.

### 4.1 Estrutura de captura do mapa e pose do robô por câmera externa

A câmera não faz parte do filtro na forma de sensor, servindo apenas como facilitador na aquisição de parâmetros que tornariam algumas tarefas exaustivas caso feitas manualmente. Foi usada como artifício de confirmação dos resultados nos experimentos, além de facilitar nos testes repetitivos realizados *a priori* no levantamento de ruídos de movimentos (linear e angular).

A Figura 41 ilustra o sistema de sustentação da câmera localizado acima do tablado. Trata-se de um arco de madeira projetado para ter altura variável e possibilidade de movimentação ao longo da largura do tablado. Com isso, pode-se centralizar a câmera no centro do tablado e conseguir que o campo de visão capture praticamente todo o ambiente. Evidentemente, devido a questões projetivas presentes na captura de imagens, foram feitos os cálculos geométricos para a definição dos centroides dos obstáculos e das coordenadas do robô (aproximadamente de 35 cm de altura) no plano do tablado (referencial básico ou altura 0), conforme será detalhado ao longo deste capítulo.

A câmera utilizada foi uma GoPro Hero 4 Silver, cuja foto é mostrada na Figura 42, a qual possui uma grande abertura para captura de imagens, possibilitando amplo campo de visão para adquirir o cenário de todo o tablado. Essa câmera possui comunicação *Wi-Fi* e grande resolução para adquirir a imagem com maior detalhamento, o que significa maior precisão e confiabilidade nos dados a serem definidos geometricamente.

O uso da câmera como meio de identificar *tags*, definidas por círculos coloridos (obstáculos) e retângulos (robô), permitiu calcular distâncias em coordenadas reais, tendo sido

Figura 41 – Estrutura de sustentação da câmera.



Fonte: (AUTOR)

de extrema importância para o desenvolvimento rápido do projeto. A resolução das fotos tiradas pela câmera foi configurada para 1280x720 pixels.

## 4.2 Calibração da câmera

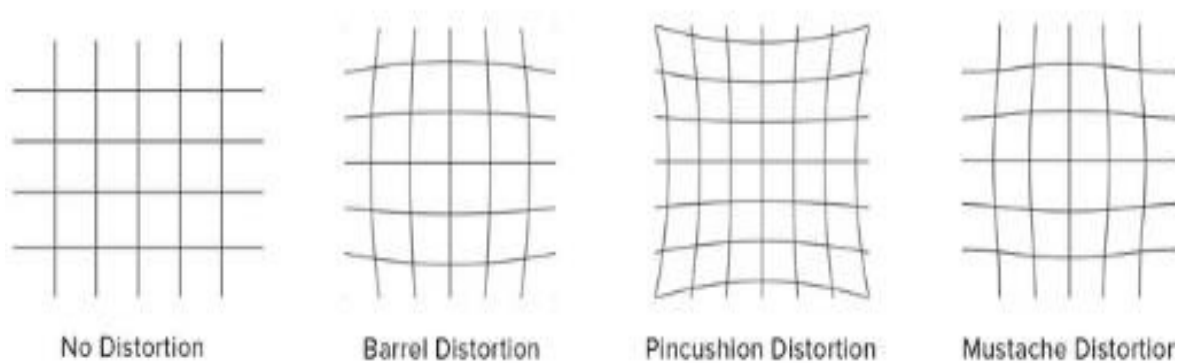
A calibração da câmera é o processo ao qual um *software* é usado para determinar os parâmetros ópticos específicos àquele tipo de câmera a fim de prover informações para a remoção de possíveis distorções presentes nas fotos e que tornam sua imagem diferente da realidade. Tais distorções na imagem são causadas pelo posicionamento da lente em relação ao sensor óptico e pelo tipo da mesma. A Figura 43 mostra os tipos mais comuns de

Figura 42 – Câmera utilizada para confirmar resultados do algoritmo e rápida realização de testes.



Fonte: <https://www.vahire.com/wp-content/uploads/GoPro-Hero-4-Silver-1100x1099.jpg>

Figura 43 – Tipos de distorções por tipo de lente.

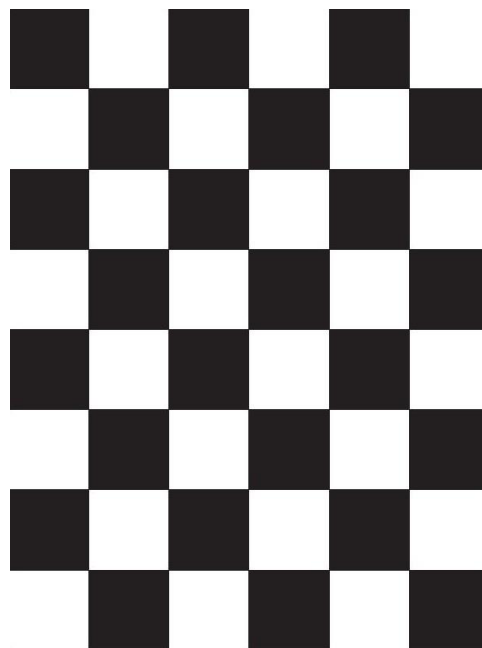


Fonte: <http://www.drewgrayphoto.com/learn/distortion101>

distorções pelo tipo e disposição da lente em relação ao anteparo para entrada da luz.

Com o intuito de deixar os dados de imagem da câmera o mais confiável possível, foi feita a calibração da mesma através do MATLAB, que captura um conjunto de imagens contendo um padrão verificador a fim de facilitar a identificação de seus parâmetros ópticos. O padrão verificador é um xadrez com quadrados pretos e brancos de 25 mm de lado, como ilustrado na Figura 44.

Figura 44 – Padrão verificador para calibração.



Fonte: (AUTOR)

Esse padrão já é uma forma comumente utilizada para calibração de câmeras, logo, existe uma forma de calibragem automática feita através de um aplicativo no MATLAB chamado <**Camera Calibrator**> que gera uma rotina a ser executada e configurada de acordo com as condições existentes para a câmera, como o tamanho do xadrez e a quantidade de fotos utilizadas.

A calibração é feita através da análise de um conjunto de imagens tiradas com a câmera parada e com o xadrez posicionado em locais diferentes em cada imagem. É recomendado colocar o xadrez, em muitas imagens, a uma distância que será a região de interesse no processamento da imagem, no caso, a superfície do tablado.

Após a coleta de imagens, executa-se a rotina de calibração, o resultado é mostrado graficamente com o erro de reprojeção na remoção da distorção por imagem e o erro médio geral. A avaliação da calibração é feita analisando o gráfico dos erros, portanto, diante de um erro médio geral muito alto para a aplicação, devem-se adicionar mais imagens à calibração ou a remoção de imagens com o erro individual muito alto.

### **4.3 Aquisição de dados na imagem**

Uma vez que a câmera foi calibrada, é necessário processar as imagens a fim de encontrar as *tags* identificadoras do robô e dos obstáculos. No MATLAB, esse processamento foi facilmente obtido através das *toolboxes* <**Image Processing**> e <**Computer Vision**>. Dessa forma, uma série de procedimentos é feita para se ter os dados necessários, sendo eles:

- a) Captura de uma foto a cada vez que uma pose nova é feita;
- b) *Download* da imagem presente no cartão de memória da câmera para o PC;
- c) Tratamento da distorção presente na imagem tornando-a mais próxima do real;
- d) Processamento da imagem, identificando e armazenando a pose do robô e os obstáculos que compõem o mapa;

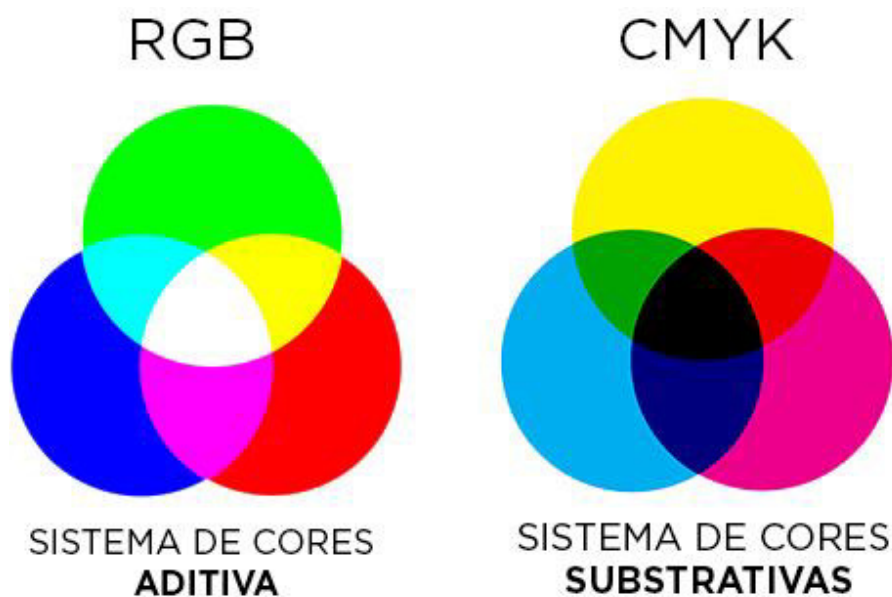
Os passos ‘a’ e ‘b’ são realizados através de um programa escrito em linguagem Python (porém, invocados pelo MATLAB), enquanto que os passos ‘c’ e ‘d’ são executados no MATLAB. Ao final desses procedimentos, têm-se as informações necessárias para fins de comparação e visualização com o resultado do filtro.

#### **4.3.1 Imagens no MATLAB**

Imagens são matrizes numéricas onde cada pixel é representado por um número, o qual está relacionado com a cor daquele pixel. Foram desenvolvidos sistemas de cores para

padronizar os valores que os elementos de uma matriz devem ter para que certa cor seja reproduzida, os mais populares são os sistemas RGB (*Red, Green, Blue*), CMYK (*Cyan, Magenta, Yellow, black*) e HSV (*Hue, Saturation, Value*)<sup>13</sup>. A Figura 45 ilustra como os sistemas RGB e CMYK, também conhecidos como aditivo e subtrativo respectivamente, compõem as cores.

Figura 45 – Sistemas RGB e CMYK de cores.



Fonte: <http://sala7design.com.br/wp-content/uploads/2016/06/o-que-e-rgb-cmyk-pantone-sala7design-cor.jpg>

O MATLAB possui recursos para representar a imagem adquirida e enviada pela câmera em vários sistemas de cores, sendo o RGB o utilizado neste trabalho. Nesse sistema uma imagem é constituída de três planos diferentes, um para cada cor base (vermelho, verde e azul). Cada plano é representado por uma matriz de intensidade de cor, ou seja, ela é uma contribuição na composição da cor de cada pixel formada pela contribuição das cores básicas.

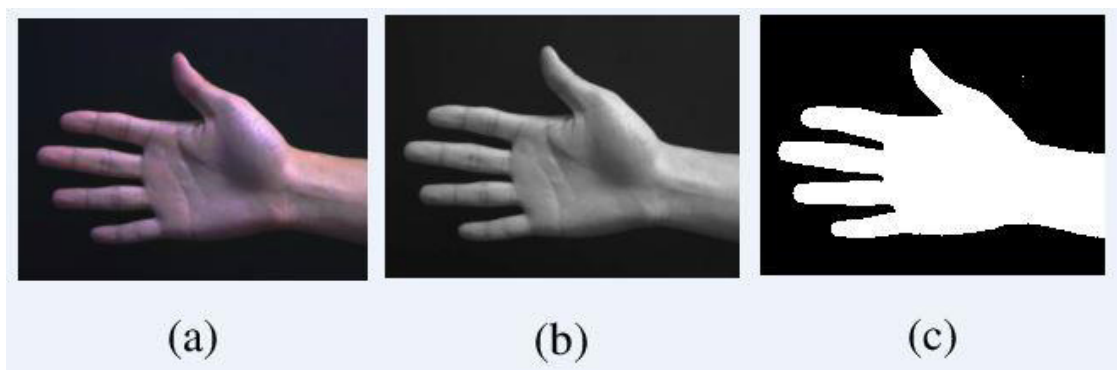
Uma matriz de intensidade é baseada na escala de cinza, onde se tem uma imagem com preto (menor valor da escala), tons de cinza (valores intermediários) e branco (maior valor da escala). A escala é determinada pelo tipo ou classe da matriz, sendo que alguns tipos usuais no MATLAB são o *uint8*, o *uint16* e o *double* com escalas [0, 255], [0, 65535] e [0, 1] respectivamente. O tipo da matriz está relacionado com a quantidade e variedade de cores disponíveis e, logo, uma imagem do tipo *uint8* possui menos opções de cores do que o tipo *uint16*, ao custo de espaço requerido para armazenamento na memória.

<sup>13</sup> Site: <http://sala7design.com.br/wp-content/uploads/2016/06/o-que-e-rgb-cmyk-pantone-sala7design-cor.jpg>



A fim de realizar operações em determinada área da imagem ou filtrar determinadas características, existe um tipo de imagem que corresponde à uma máscara binária que, no caso, constitui a imagem binária. Esse tipo de imagem é composto de apenas as cores preta e branca, com sua matriz representada pelo par binário 0 e 1. A Figura 46 mostra as três representações apresentadas até agora com um exemplo da foto de uma mão em cores reais (RGB), escala de cinza (*grayscale*) e imagem binária. Imagens binárias são usadas neste

Figura 46 – Três tipos de representação de imagens, (a) RGB, (b) escala de cinza e (c) binária.



Fonte:

[https://www.researchgate.net/publication/304190732\\_Human\\_authentication\\_with\\_finger\\_textures\\_based\\_on\\_image\\_feature\\_enhancement](https://www.researchgate.net/publication/304190732_Human_authentication_with_finger_textures_based_on_image_feature_enhancement)

trabalho para isolar determinada região na imagem e realizar o processamento apenas na região de interesse (ROI – *Region of Interest*).

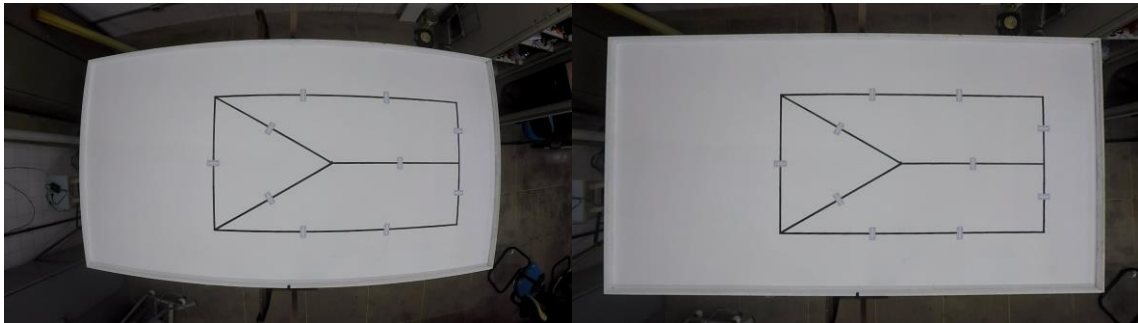
#### 4.3.2 Processamento de imagens no MATLAB

Neste tópico, serão vistas as principais funções das *toolboxes* <Image Processing> e <Computer Vision> do MATLAB, detalhando e seu uso para aquisição de informações dos obstáculos e do robô através do processamento de imagens.

Como mencionado anteriormente, a câmera utilizada no trabalho produz distorção na imagem tirada devido ao tipo de lente, essa distorção foi removida via *software* através da função <undistortImage()> da *toolbox* <Computer Vision>. A função possui como argumento de entrada a imagem a ser corrigida e os dados resultantes da calibração da câmera, retornando a imagem desejada sem a distorção. A Figura 47 mostra uma foto tirada do tablado com distorção (esquerda) e após a aplicação da função (direita). Esse tipo de deformação conhecida como *barrel distortion* (BUONOCORE, 2010) é proveniente de câmeras usadas em vigilância, onde a área a ser capturada deve ser a maior possível. Percebe-

se que as deformações são maiores quanto mais distantes os pontos estiverem do centro da imagem.

Figura 47– Comparação da imagem distorcida com a corrigida.



Fonte: (AUTOR)

Uma pequena área em volta da imagem original é perdida para que a remoção da distorção produza uma imagem com as mesmas dimensões. Em virtude desse fato, posicionou-se a câmera um pouco mais alta para compensar essa perda no campo de visão. Todas as funções mencionadas daqui em diante que recebem imagens em seus argumentos, receberão imagens após a remoção da distorção.

A próxima função transforma uma imagem RGB em *grayscale*, denominada `<rgb2gray()>`. Seu único argumento é uma imagem em RGB e seu parâmetro de retorno é a imagem transformada em *grayscale*. A utilidade dessa função neste trabalho está no fato de que é possível fazer uma subtração de algum dos três componentes RGB de uma imagem por sua versão em *grayscale*, resultando em uma imagem também em *grayscale*. Essa operação faz sentido quando se transforma o resultado em uma imagem binária, onde se vê uma imagem com plano de fundo preto com os locais da cor escolhida (na imagem RGB) em branco.

A Figura 48 mostra a imagem RGB dos obstáculos e a Figura 49 a transformação para imagem binária quando a cor verde foi a escolhida. A função `<imbinarize()>` transforma uma imagem em *grayscale* para uma imagem binária. O argumento dessa função é a imagem em *grayscale* e seu parâmetro de retorno é a imagem binária. Geralmente essa função é acompanhada do uso de uma função chamada `<imfill()>`, que tem o propósito de preencher quaisquer regiões pretas localizadas dentro de regiões em branco.

As funções `<imbinarize()>` e `'imfill()'` pertencem à *toolbox* `<Image Processing>` do MATLAB e, incluindo a próxima função, formam as principais ferramentas utilizadas no processamento de imagens realizado neste trabalho.

Figura 48 – Imagem em RGB destacando obstáculos.

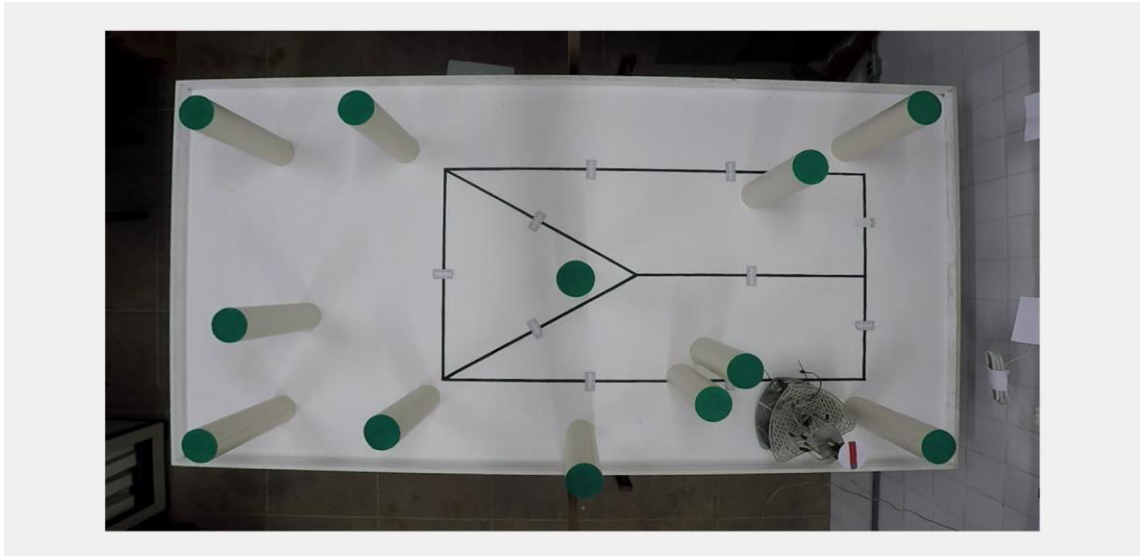
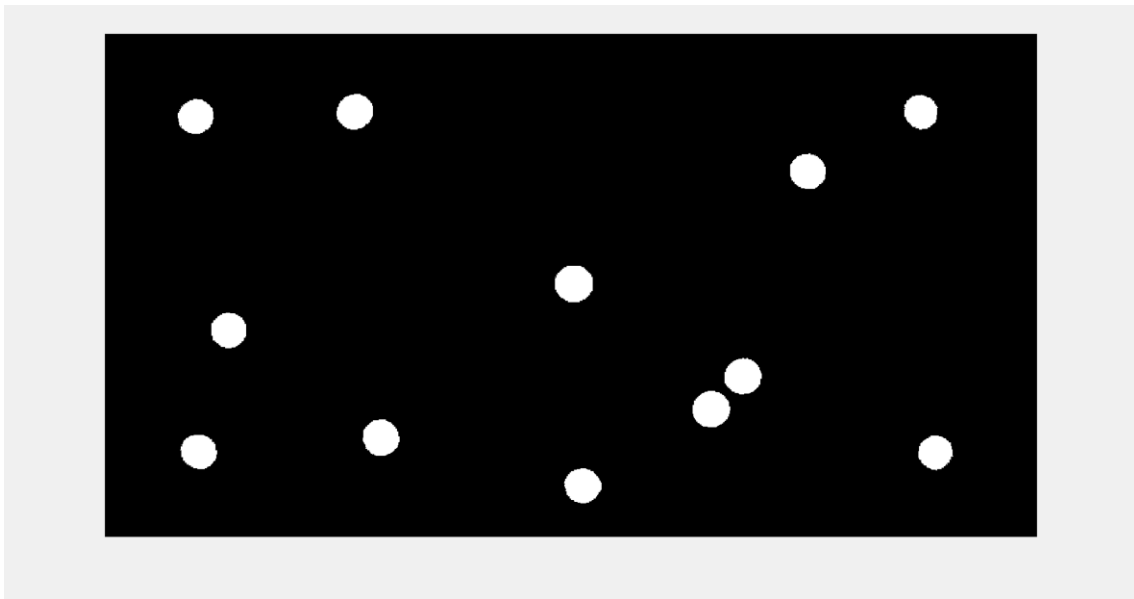


Figura 49 – Imagem binária aplicada à cor verde.



Fonte: (AUTOR)

A última função é a `<regionprops()>`, sendo considerada a mais importante para a realização das tarefas com a câmera, devido a sua capacidade de fazer medições de propriedades em regiões na imagem. Seu argumento é uma imagem binária e a lista de propriedades a serem medidas e seu parâmetro de retorno é uma *struct* contendo as

informações requisitadas. A lista completa de propriedades se encontra na documentação<sup>14</sup> da função, no entanto, as propriedades necessárias para as tarefas deste trabalho são apenas as seguintes:

- a) Área: retorna o número de pixels presente na região;
- b) Centroide: retorna as coordenadas do centro de massa da região;
- c) Pontos extremos: retorna oito pontos de cantos da região;
- d) Tamanho do eixo maior da elipse formada pela região; e
- e) Tamanho do eixo menor da elipse formada pela região.

O seu funcionamento é baseado em regiões presentes na imagem binária passada como argumento. Uma região é qualquer conjunto, isolado ou não, de valores não nulos na matriz que compõe a imagem, ou seja, regiões brancas.

As propriedades que retornam pontos, como centroide e pontos extremos, usam o índice do elemento da matriz para designar um ponto no espaço. As demais propriedades retornam escalares normalmente, porém, com a presença de múltiplas regiões na imagem, a medição dessas propriedades são vetores onde cada índice representa uma região.

Através do uso dessas propriedades, as seguintes informações que foram utilizadas neste trabalho puderam ser extraídas:

- a) Centroide das *tags* dos obstáculos;
- b) Centroide da *tag* do robô;
- c) Orientação da *tag* do robô; e
- d) Quatro pontos que compõem os cantos do tablado;

Enquanto que os pontos do tablado estão na mesma altura que o mesmo (nível de base da imagem), as outras informações estão elevadas a 35 cm de altura do tablado. Portanto, são medidas distorcidas pela projeção dessa altura na imagem. A Figura 50 mostra em azul, que a distância entre os centros das *tags* é diferente da distância entre suas bases, no caso, 80,58 e 70,93, respectivamente.

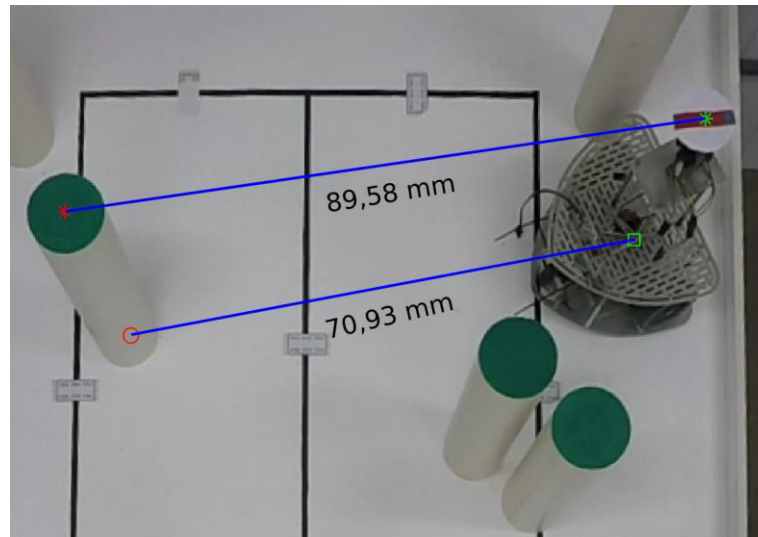
### 4.3.3 Correção de erros de projeção na identificação das *tags*

Câmeras geram imagens com perspectiva, ou seja, todos os raios de luz responsáveis pelos pontos da imagem passam pelo mesmo ponto, chamado centro de perspectiva, onde se encontra o centro óptico da câmera. Esse fenômeno em associação com a localização da lente

---

<sup>14</sup> Site: <https://www.mathworks.com/help/images/ref/regionprops.html>

Figura 50 – Diferenças entre distâncias base-base e *tag-tag*.



Fonte: (AUTOR)

é o responsável pelo efeito visto na seção anterior e deve ser corrigido para que se possam computar os valores na sua forma mais próxima do real.

Foi usada uma solução que se aproveita da geometria envolvida no problema para determinar a correção da projeção em pontos na imagem. É necessário conhecer alguns valores antes de iniciar a solução:

- O ponto principal da imagem<sup>15</sup>;
- A largura e o comprimento do tablado em centímetros;
- A altura do objeto em centímetros;
- A altura da câmera relativa à superfície do tablado em centímetros; e
- As coordenadas do centroide da *tag*;

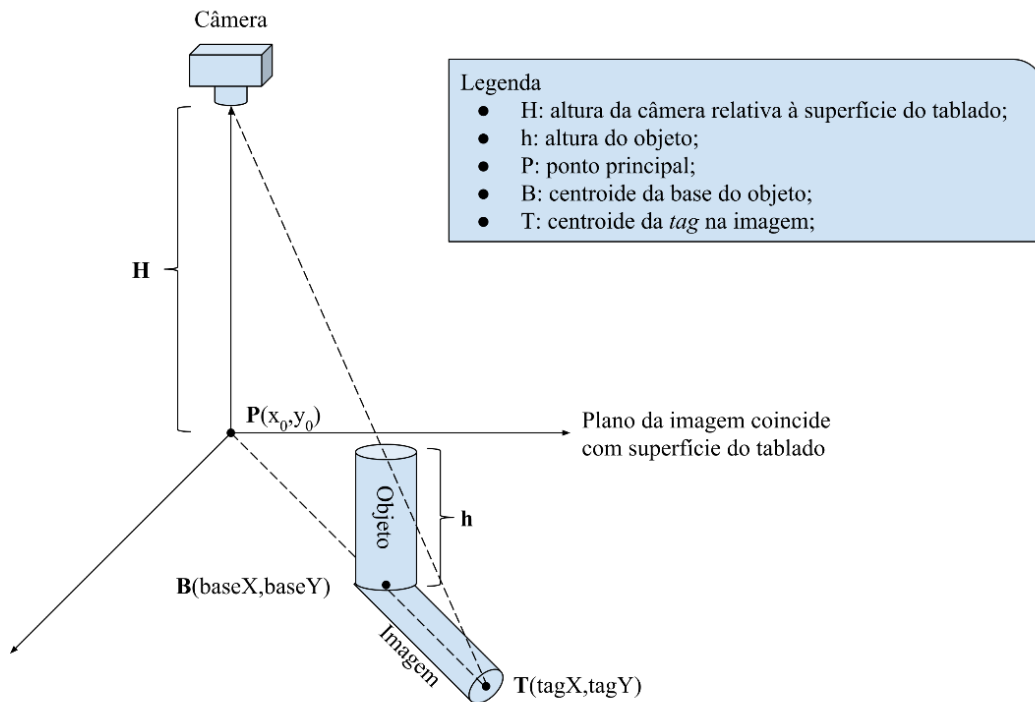
O ponto principal é encontrado como informação da câmera oriunda do algoritmo de calibração. Com exceção das coordenadas do centroide da *tag*, todos os outros são constantes e podem ser medidos manualmente com uma trena.

A Figura 51 a seguir ilustra graficamente o problema. A solução se trata de uma simples semelhança de triângulos com as Equações 26 e 27 a seguir realizando uma transformação do ponto em cada coordenada. Esse processo é conhecido como ortorretificação<sup>16</sup> de imagens, sendo utilizado quando a aplicação necessita de imagens com perspectiva ortogonais, ou seja, os raios de luz refletidos são todos paralelos ao eixo ótico. Com essa solução é possível transformar o ponto para a superfície do tablado.

<sup>15</sup> Ponto central da imagem obtido pela interseção do eixo ótico com o plano da imagem.

<sup>16</sup> Site: [http://www.sigmat.com.br/index.php?option=com\\_content&view=article&id=158&Itemid=273](http://www.sigmat.com.br/index.php?option=com_content&view=article&id=158&Itemid=273)

Figura 51 – Detalhes da projeção de objetos na imagem.



Fonte: (AUTOR)

$$baseX = \frac{(tagX - x_0) * (H - h)}{H} + x_0 \quad (26)$$

$$baseY = \frac{(tagY - y_0) * (H - h)}{H} + y_0 \quad (27)$$

Neste capítulo foi apresentado o sistema de captura de imagem digital para auxílio do desenvolvimento do trabalho, trazendo maior nível de automação em ensaios do robô e, principalmente, nos experimentos realizados e que serão apresentados no Capítulo 6. No próximo capítulo serão descritos os módulos de *software* operantes no robô e no PC.

## 5 Módulos de *software* do robô MILO

Neste capítulo, a parte computacional é detalhada na forma de módulos, os quais são separados de acordo com características compartilhadas e o sistema onde são executados, no caso o RPi3 embarcado no robô e o computador remoto.

### 5.1 Arquitetura dos módulos de *software*

A organização dos módulos se dá de forma a simplificar o entendimento do escopo global da estrutura de execução dos programas em funcionamento no sistema robótico desenvolvido neste trabalho de monografia. Os módulos, seu sistema de chamada as funções e uma descrição de seus funcionamentos estão listados a seguir:

- a) Módulo Principal (executado no RPi3): Contém a função *main* (código principal), sendo a responsável por fazer a criação dos outros módulos, tendo sua execução contínua durante todo o processo de localização realizado pelo robô;
- b) Módulo PWM (executado no RPi3): Desenvolvido para suprir a deficiência de um *hardware* específico para geração do sinal PWM, responsável pelo movimento do servo motor, usado no giro da torre de medição;
- c) Módulo TCP (executado pelo RPi3 e PC): Estabelece comunicação TCP/IP através de *sockets*<sup>17</sup> para transferência de dados entre o robô e o PC;
- d) Módulo MATLAB (executado pelo PC): Ambiente de execução do algoritmo Localização Monte Carlo ou MCL; e
- e) Módulo Câmera (executado pelo PC): Realiza as aquisições de imagens para comparação com o resultado do algoritmo MCL e definição dos ruídos do movimento de movimento em sentido longitudinal e rotacional.

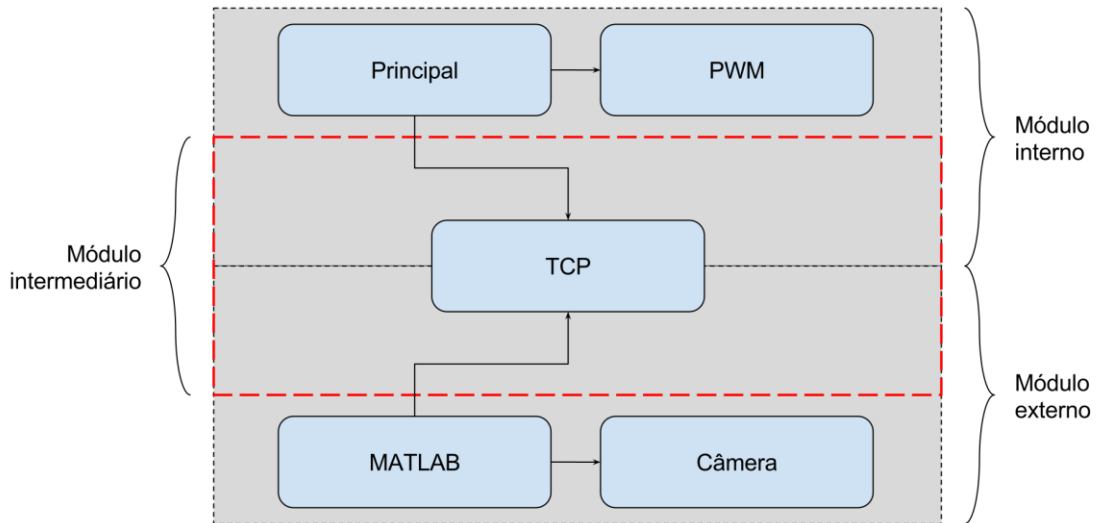
Os módulos podem ser compostos por um código fonte apenas (itens ‘a’ e ‘b’), dois programas executando em sistemas diferentes (‘c’) ou mais de um programa executados no mesmo sistema em momentos diferentes (‘d’ e ‘e’). Os dois primeiros são denominados módulos internos, o terceiro módulo intermediário e os dois últimos de módulos externos, sendo essa definição relacionada ao robô. A Figura 52 ilustra na forma de blocos a arquitetura dos módulos e como se interagem.

A fim de tornar eficiente o uso dos núcleos no SoC do RPi3, alguns módulos internos foram divididos em *threads*. Um processo pode se dividir em partes a fim de executar

---

<sup>17</sup> Sites: <http://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html> e <http://www.csd.uoc.gr/~hy556/material/tutorials/cs556-3rd-tutorial.pdf>

Figura 52 – Arquitetura dos módulos de *software*.



Fonte: (AUTOR)

tarefas de forma concorrentes ou simultâneas, dando-se o nome de *thread* as tarefas de um único processo (programa executável). Um *thread* é organizado pelo SO e podem ser facilmente criados e destruídos de forma a garantir baixa carga de processamento para o sistema em que está sendo executado.

Em sistemas baseados em UNIX, como é o caso do Raspbian<sup>18</sup> presente no robô, *threads* podem ser facilmente implementados através da biblioteca nativa `<pthread.h>` (acrônimo para POSIX<sup>19</sup> *Thread*). O Módulo Principal é o responsável pela criação e destruição de *threads* conectados a outros módulos, se utilizando das funções contidas na referida biblioteca, organizando o fluxo de informações pelo gerente do sistema *multithreading*.

A seguir é feita uma breve descrição do funcionamento dos módulos e suas implementações em pseudocódigos.

## 5.2 Módulo Principal

O RPi3 contém os módulos internos divididos nos seguintes códigos fontes (destacados em negrito, como `<nome_do_módulo.c>`), cabeçalhos (`<nome_do_módulo.h>`) e arquivo de lote. A linguagem usada na implementação foi a C:

- `<lowlevelfunctions.c>`: constitui as funções básicas para acionamento dos motores, leitura de sensores e geração do sinal modulado em PWM;
- `<lowLevel.h>`: cabeçalho contendo os protótipos das funções básicas;

<sup>18</sup> Site: <http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2016/10/veja-como-instalar-o-raspbian-no-raspberry-pi.html>

<sup>19</sup> Site: <https://computing.llnl.gov/tutorials/pthreads/>



- c) **<main.c>**: código contendo a função `main`, sendo o responsável por chamar as funções base e realizar a comunicação TCP/IP como servidor; e
- d) **Makefile**: não constitui um código a ser transformado em executável. São instruções para compilação e ligação dos módulos executáveis acima, através de comandos do compilador `'gcc'` do SO Linux, chamadas pelo comando `'make'`.

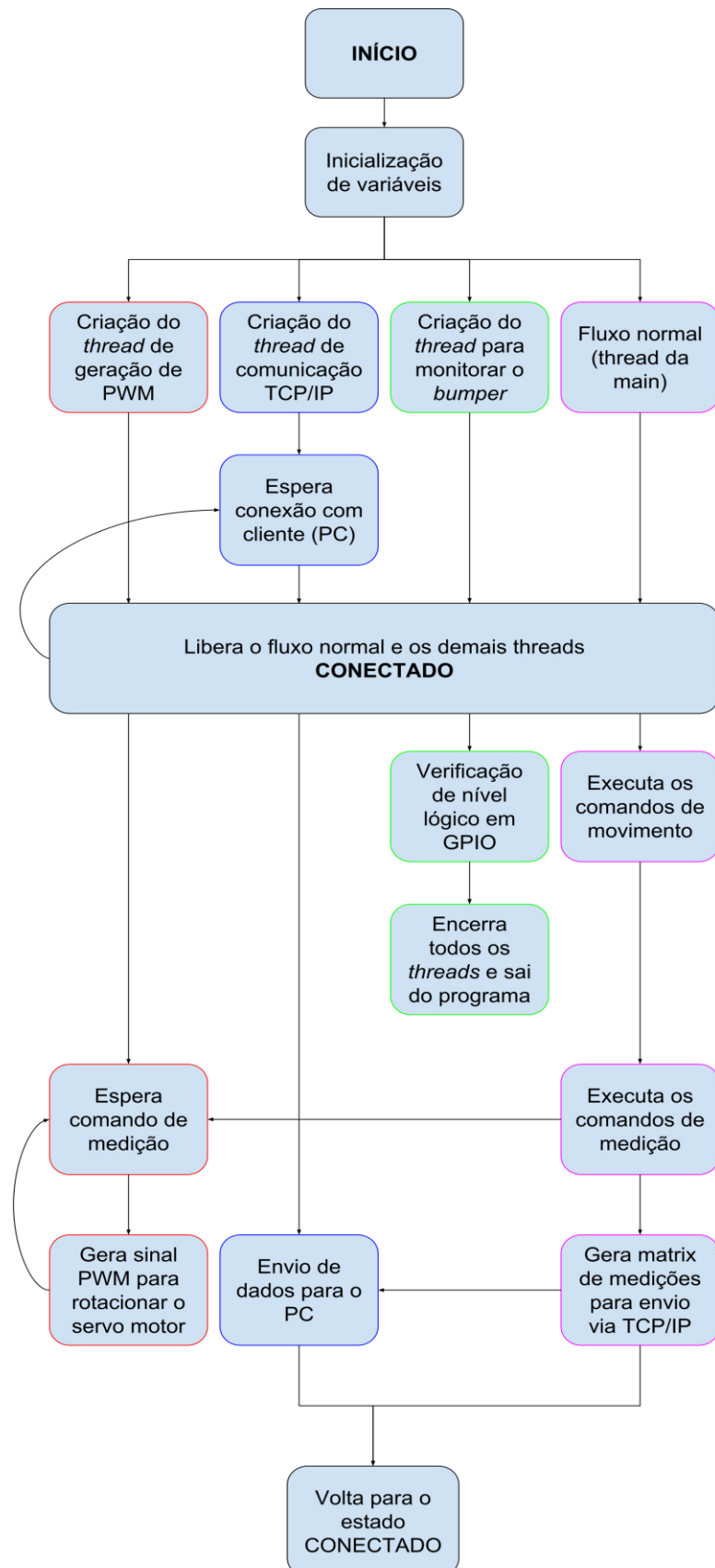
Apenas o arquivo **<main.c>** está representando o Módulo Principal, o qual é responsável por chamar os módulos PWM e TCP/IP, descritos mais adiante. O fluxograma de execução do Módulo Principal está ilustrado na Figura 53.

Em resumo, o fluxo de execução desse módulo no RPi3 ocorre da seguinte maneira: após a criação dos *threads*, espera-se até que a comunicação com o PC seja estabelecida para que o programa possa fazer os movimentos e medições necessárias e comandadas pelo algoritmo MCL. As ações são dadas da seguinte forma:

1. 1ª ação – movimento de rotação e medição da odometria feita pelos *encoders*;
2. 2ª ação – movimento de translação e medição da odometria feita pelos *encoders*;
3. 3ª ação – movimento de rotação e medição da odometria feita pelos *encoders*;
4. 4ª ação – giro da torre de medição no intervalo 0° a 180° (total de 360°), adquirindo dados dos pares de sensores frontais e traseiros; e
5. 5ª ação – envio dos dados de medição acumulados em vetores ao PC usando o protocolo TCP/IP.

Caso ocorra o chaveamento no *bumper switch* (chave tátil), o sistema encerra o que está fazendo e sai do programa. O *thread* de geração do sinal PWM só executa sua função quando se entra na rotina de medição com os sensores, após o robô ser comandado para deslocar entre poses sucessivas no ambiente. Nesse *thread* o ângulo de medição da torre, que é constante e definido em aproximadamente 4°, é alcançado pelo giro do eixo do servo-motor, conforme mencionado no capítulo 3. Após a aquisição de dados, é composta a matriz dos parâmetros dos objetos medidos no ambiente pelo robô, no caso a distância e o ângulo com que o sistema mede cada obstáculo, e que são relativos às coordenadas do sistema de medição. Ao final, a torre é retornada ao seu ponto de referência (0°) no sistema de coordenadas do robô e a matriz dos parâmetros dos objetos medidos é enviada como pacote para o PC, através da comunicação TCP/IP.

Figura 53 – Fluxograma de execução do Módulo Principal.



Fonte: (AUTOR)

### 5.3 Módulo PWM

O Módulo PWM está situado em um *thread* criado pelo Módulo Principal. O sinal de PWM é gerado através de uma rotina implementada para produzir um pulso dentro do período de 20 ms de acordo com a posição desejada no servo motor. O *thread* recebe o valor de *duty cycle* (ciclo de trabalho) e gera pulsos no pino GPIO dentro de um laço infinito de acordo com o *duty cycle* recebido. Este *thread* é terminado no Módulo Principal, que determina o instante certo para iniciá-lo novamente com outro valor de *duty cycle* e, assim, sucessivamente.

O módulo não se utiliza de funções temporizadas bloqueantes, como **sleep()** ou **usleep()**, ao invés, uma função nativa<sup>20</sup> que retorna o tempo corrido em microssegundos desde a inicialização do RPi3 foi utilizada para realizar a temporização dos intervalos onde o estado do GPIO é alto ou baixo.

### 5.4 Módulo TCP

Esse código executa o processo da troca de dados do RPi3 com o PC usando o protocolo TCP/IP. Neste trabalho, esse processo foi facilitado na programação da placa embarcada devido à existência de bibliotecas nativas que configuram e executam esse protocolo, sendo elas a `<sys/socket.h>` e a `<netinet/in.h>`, e no MATLAB através do *Toolbox* **<Instrument Control>**.

Nesse módulo, dois sistemas estão conectados através de programas executando em cada um deles o mesmo protocolo de comunicação. O PC fará o papel de cliente e o RPi3 será o servidor. Através das funções citadas abaixo o MATLAB pode fazer a solicitação de conexão e o envio ou recebimento de dados:

- a) `obj = tcpip (RemoteHost, RemotePort, RemotePort, Value);`
- b) `fopen (obj)` ou `fclose (obj);`
- c) `fwrite (obj, data);` e
- d) `data = fread (obj).`

O comando em ‘a’ cria um objeto TCPIP ‘**obj**’ para uso com as demais funções e especifica o IP da máquina destino da comunicação (‘**RemoteHost**’), a porta de comunicação (‘**RemotePort**’) e o papel que o MATLAB fará (‘**RemotePort**’) na comunicação (servidor ou cliente).

O comando em ‘b’ abre (**<fopen>**) ou fecha (**<fclose>**) a conexão com o objeto TCPIP previamente configurado em seu argumento.

---

<sup>20</sup> Site: [https://linux.die.net/man/3/clock\\_gettime](https://linux.die.net/man/3/clock_gettime)

O comando em ‘c’ escreve dados no objeto TCP/IP passado como argumento, enviando ‘data’ do MATLAB para o destino.

O comando em ‘d’ lê os dados no objeto TCPIP passado como argumento, retornando ‘data’ para o MATLAB oriundo da máquina destino.

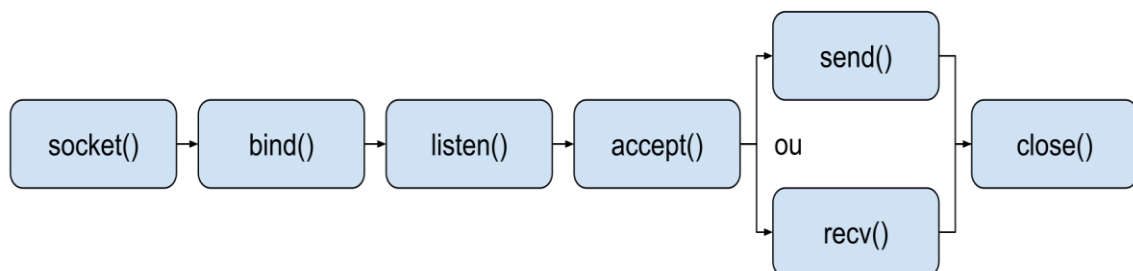
No RPi3, as funções para estabelecer a conexão são mais detalhadas e serão resumidas em sua apresentação. Como previamente mencionado, as bibliotecas <sys/socket.h> e <netinet/in.h> são incluídas com o objetivo de se poderem chamar as funções que realizam o protocolo em C, sendo seus nomes seguidos de seus propósitos:

- socket(...): cria novo nó de comunicação;
- bind(...): atribui um endereço local a um *socket*;
- listen(...): entra em estado disponível para conexões;
- accept(...): bloqueia o código até a conexão do cliente ser pedida;
- connect(...): tentativa de estabelecer a conexão com o servidor;
- send(...): envio de dados;
- recv(...): recebimento de dados; e
- close(...): libera o nó da conexão.

As funções têm certa dependência e usabilidade a depender do papel da máquina, ou seja, servidores usam determinadas funções que o cliente não utiliza, como é o caso de ‘bind()’ e ‘listen()’. A Figura 54 a seguir ilustra o fluxograma de chamada das funções no Módulo TCP no servidor (RPi3).

O Módulo TCP funciona se utilizando da sincronia criada pela conexão e a transmissão de informações para, principalmente, compartilhar dados do RPi3 com o MATLAB. Os dados estão na forma de vetores os quais são passados como entrada para o algoritmo MCL, sendo armazenados no *workspace* do MATLAB.

Figura 54 – Fluxograma de execução do Módulo TCP no servidor.



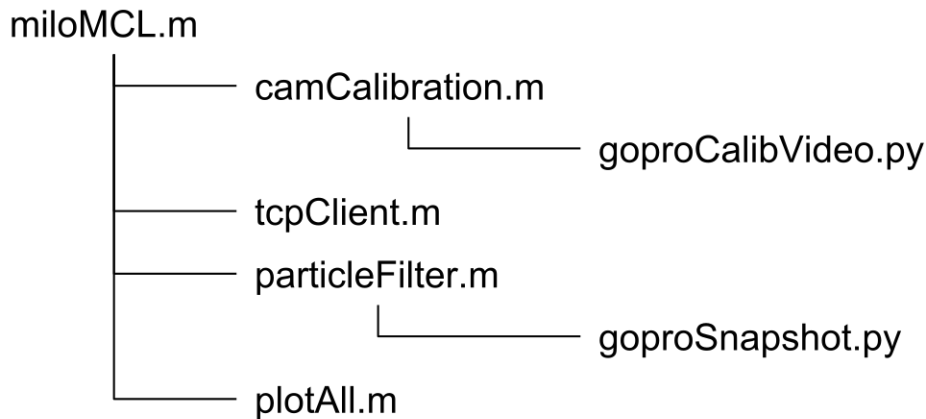
## 5.5 Módulos MATLAB e Câmera

O Módulo MATLAB é responsável pela execução de *scripts* no ambiente MATLAB e está diretamente relacionado com o Módulo Câmera e o Módulo TCP, pois ambos são chamados pelo Módulo MATLAB. Enquanto o Módulo Câmera é formado pelos códigos fonte escritos na linguagem Python, os outros dois são *scripts* para o ambiente MATLAB. As funções usadas em Python foram importadas de uma biblioteca de terceiros<sup>21</sup> projetada para lidar com as necessidades deste trabalho.

Existe um *script* raiz por onde outros códigos são chamados de forma aninhada, compondo o Módulo MATLAB. A Figura 55 mostra de forma simplificada as dependências do *script* raiz, denominado ‘**milomCL**’.

A sequência de execução está mostrada também na Figura 55. Logo, o primeiro *script* a

Figura 55 – Árvore de dependências do Módulo MATLAB.



Fonte: (AUTOR)

ser executado é o de calibração da câmera, <**camCalibration**>, o qual verifica algum arquivo previamente salvo da última calibração válida que, se existente, será carregado no *workspace* para uso. Caso não encontre, é chamado um código fonte escrito em Python para gravar um vídeo de 1 minuto, chamado <**goproCalibVideo**>.

Um padrão de xadrez impresso é utilizado na calibração da câmera, procedimento detalhado na Seção 4.2, devendo ser movido ao longo do tablado para registro do grau de deformação no vídeo. Após esse passo, a execução volta ao *script* <**camCalibration**>, onde é feita a separação em *frames* e executada uma rotina pré-programada<sup>22</sup> pelo MATLAB para detectar, selecionar e usar o padrão de xadrez nos *frames* para remover a distorção presente

<sup>21</sup> Site: <https://github.com/KonradIT/gopro-py-api>

<sup>22</sup> Site: <https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html>

nas imagens. Os dados dessa operação são salvos em disco para serem usados futuramente como arquivo de calibração.

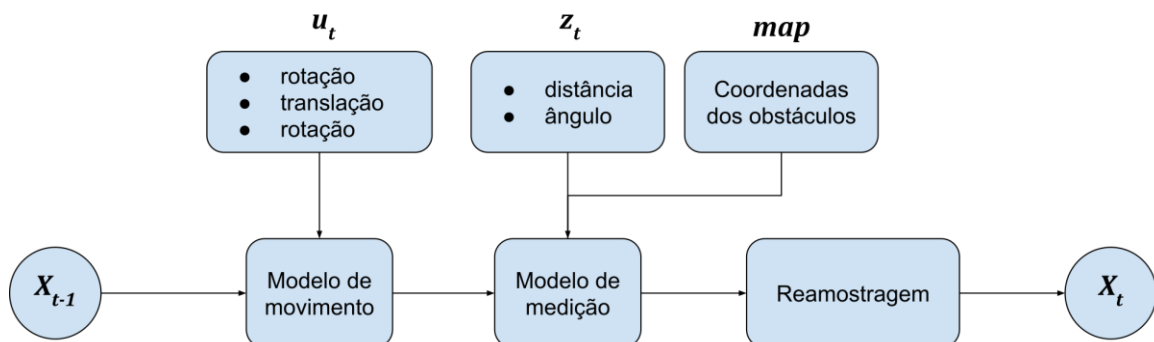
Quando a execução se encontra no *script* <tcpClient>, o código se torna bloqueante até o ponto onde é feita a tentativa de conexão com o robô (servidor), ou seja, caso o código no robô não esteja esperando conexões ou esteja desligado, a execução no MATLAB é suspensa até essa condição ser atendida ou até ocorrer o *timeout* da conexão (por padrão esse valor é de 10 segundos). Caso as condições estejam permitindo a comunicação, o *script* continua o fluxo de execução e espera os dados de medição do robô para entrar no próximo *script*.

Na sequência, o *script* <particleFilter> é iniciado, onde se tem os primeiros dados de entrada: o controle  $u_t$  e o vetor de medições  $z_t$ , obtidos após as cinco ações mencionadas no Módulo Principal. A primeira ação tomada ao receber esses dados e armazená-los em variáveis no *workspace* do MATLAB, é a chamada ao código em Python <goproSnapshot> para armazenar a pose real do robô naquele instante para uso comparativo ao final da rotina, além das coordenadas dos obstáculos que formam o mapa. Em seguida, o algoritmo de localização é então executado, como ilustrado no fluxograma da Figura 56.

O fluxo de execução do algoritmo segue do bloco  $X_{t-1}$  para o  $X_t$ , representando o conjunto de hipóteses no instante anterior e atual, respectivamente. As hipóteses são as possíveis poses que o robô pode assumir no espaço delimitado pelo tablado, no formato  $[x, y, \theta]^T$ , e que neste trabalho totalizam 1000 partículas.

O início do algoritmo é caracterizado pela amostragem (espalhamento probabilístico) das partículas de forma a se obter um estado  $X_{t-1}$  *a priori*, que permitira iniciar o filtro. Esse espalhamento, em se tratando de localização global (problema abordado neste trabalho), geralmente é feito de forma aleatória em todo o espaço do ambiente e amostrado através de

Figura 56 – Fluxograma do algoritmo de localização.

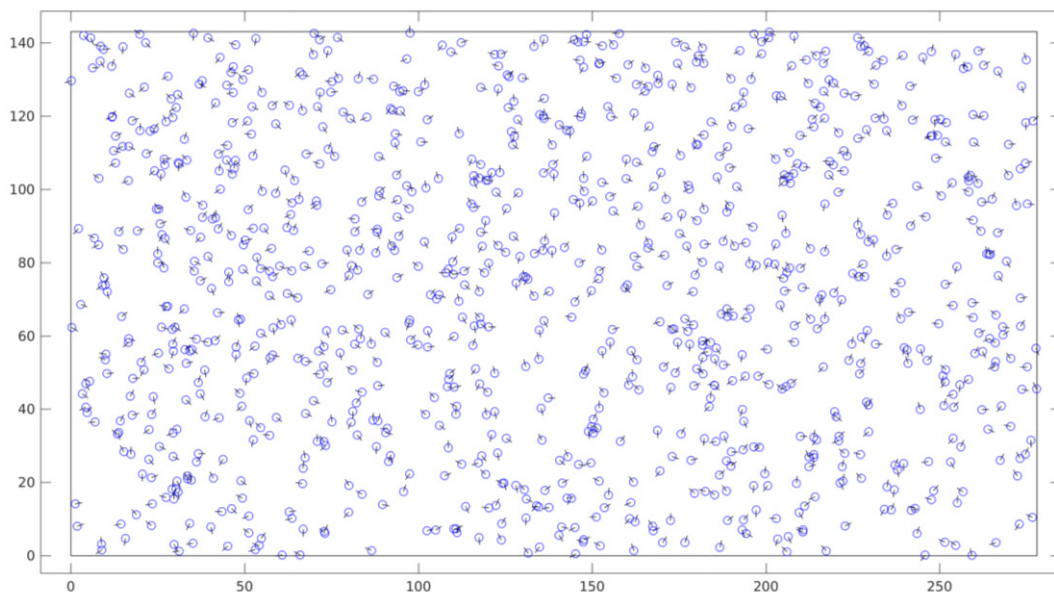


uma distribuição *gaussiana*. No entanto, para este trabalho, foi feita uma abordagem diferente e que se mostrou mais eficiente na convergência do filtro de partículas em uma quantidade menor de poses. O recurso utilizado é a redução do espaço amostral a uma elipse centrada num ponto calculado através do processo de triangulação de obstáculos que correspondem às medidas adquiridas. As Figuras 57 e 58 mostram o espalhamento em todo o espaço amostral (tablado) e em forma de elipses menores, denominadas de *clusters* de partículas.

Os *clusters* de partículas são formados de acordo com a correspondência encontrada, onde se tomam as medidas da primeira iteração do MCL, ou seja, o processo de criação desses *clusters* é executado somente uma vez ao início do filtro. Uma vez que se obtém o primeiro conjunto de medidas, faz-se uma matriz de distâncias entre cada medição e entre cada obstáculo no mapa, para em seguida realizar comparações da matriz de distâncias das medições com a matriz de distâncias dos obstáculos, envolvendo todas elas uma a uma.

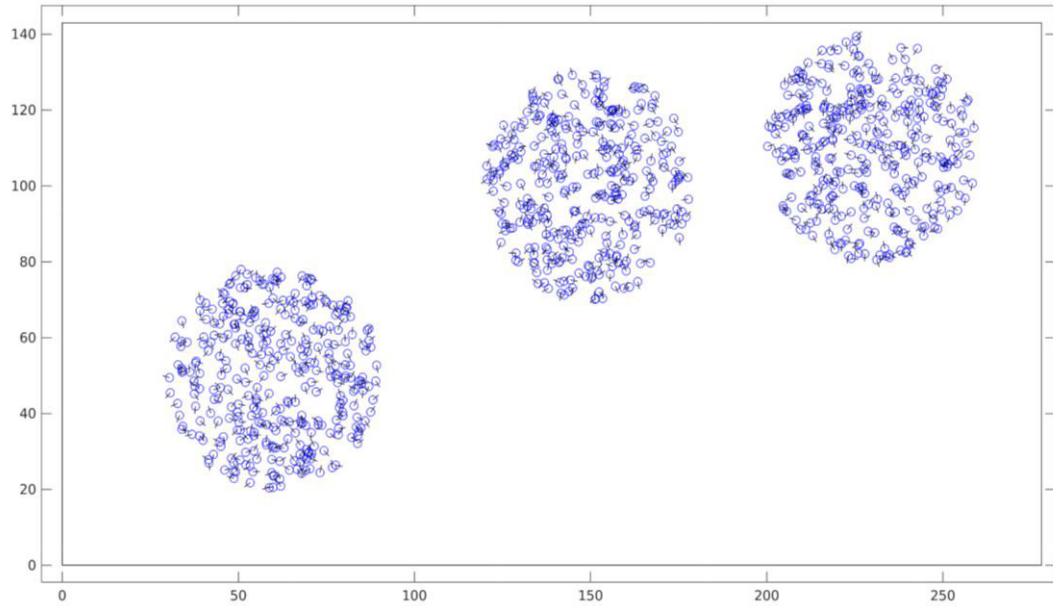
Quando uma correspondência for encontrada no mapa, considerando uma tolerância nas distâncias medidas, são feitas circunferências centrada nos obstáculos e com raio igual às distâncias nas medições para se encontrar os pontos de interseção dessas circunferências e, por fim, o baricentro desses pontos. A Figura 59 abaixo ilustra a situação onde três medidas foram tomadas e a correspondência em três obstáculos determinando uma pose aproximada de acordo com a triangulação estabelecida.

Figura 57 – Espalhamento das partículas em todo o espaço amostral.



Fonte: (AUTOR)

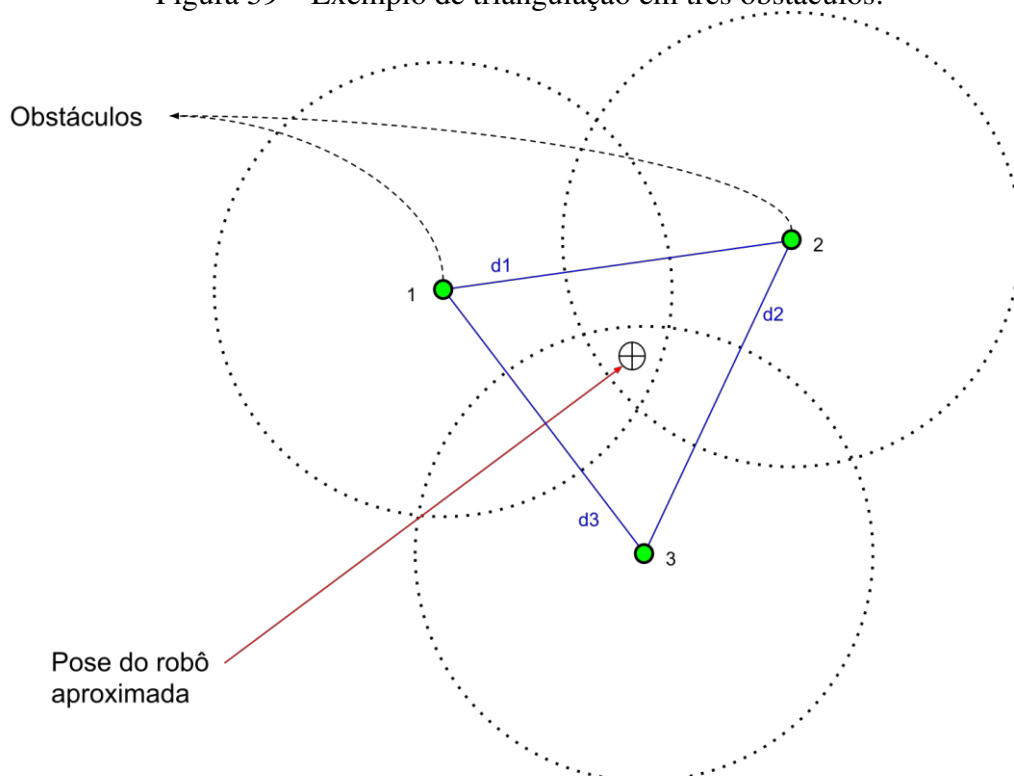
Figura 58 – Espalhamento de partículas em *clusters*.



Fonte: (AUTOR)

A depender da disposição dos obstáculos, pode-se encontrar mais de uma correspondência, ou seja, em mais de uma vez as distâncias entre as medições possuem correspondências em distâncias entre os obstáculos. Quando essa situação acontece, é

Figura 59 – Exemplo de triangulação em três obstáculos.



Fonte: (AUTOR)



atribuída uma probabilidade à ocorrência daquela correspondência, de acordo com os ângulos nas medições, a qual determina a quantidade de partículas que serão espalhadas naquela elipse. A Figura 58 mostrou esse fenômeno quando foram encontradas três correspondências, requerendo a formação de três *clusters*.

Essa simetria que os obstáculos do mapa podem fornecer, foi testada e será discutida no Capítulo 6, mostrando um experimento realizado onde aparecem mais de uma correspondência. Da mesma forma, foi feito um experimento onde o ambiente é assimétrico no sentido de retornar apenas uma correspondência.

Após o uso da triangulação para reduzir o espalhamento das partículas, o algoritmo segue a execução para a amostragem através do modelo de movimento. A amostragem é o processo ao qual cada partícula no conjunto  $X_{t-1}$  executa o controle  $u_t$ . Logo, todas as partículas se movimentam no espaço amostral de acordo com as medições odométricas para os três movimentos executados no robô. O pseudocódigo de amostragem é apresentado no Algoritmo 1, onde  $x_{t-1} = (x, y, \theta)^T$  e  $u_t = (\delta_{rot1}, \delta_{trans}, \delta_{rot2})^T$ .

Algoritmo 1:  $x'_t = \text{amostragemModeloOdometrico}(u_t, x_{t-1})$

1.  $\delta_{rot1} = u_t(1)$
2.  $\delta_{trans} = u_t(2)$
3.  $\delta_{rot2} = u_t(3)$
4.  $\hat{\delta}_{rot1} = \delta_{rot1} - \text{amostrar}(\alpha_{rot1}^2 + \alpha_{trans}^2)$
5.  $\hat{\delta}_{trans} = \delta_{trans} - \text{amostrar}(\alpha_{trans}^2 + \alpha_{rot1}^2 + \alpha_{rot2}^2)$
6.  $\hat{\delta}_{rot2} = \delta_{rot2} - \text{amostrar}(\alpha_{rot2}^2 + \alpha_{trans}^2)$
7.  $x' = x_{t-1}(1) + \hat{\delta}_{trans} * \cos(\theta + \hat{\delta}_{rot1})$
8.  $y' = x_{t-1}(2) + \hat{\delta}_{trans} * \sin(\theta + \hat{\delta}_{rot1})$
9.  $\theta' = x_{t-1}(3) + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$
10. **retorne**  $x'_t = (x', y', \theta')^T$

A função <amostrar> é uma PDF com média zero e variância determinada pelo seu argumento. Os parâmetros  $\alpha_b^2$  são os ruídos de movimento previamente determinados através de ensaios, onde  $b$  está associado a cada um dos três movimentos realizados pelo robô.

Após a execução desse algoritmo, um novo conjunto de partículas  $X'_t$  é formado e servirá de parâmetro de entrada para o modelo de medição juntamente com os dados de medição. No

entanto, a fusão dos dados sensoriais deve ser aplicada ao vetor de medições, antes do mesmo fazer parte do modelo de medição.

A fusão sensorial é feita primariamente para agrupar as medidas dos quatro sensores e compor apenas valores que representariam uma medida pontual de um obstáculo. A Figura 60 mostra o resultado de uma fusão onde os círculos vermelhos são medições brutas antes da fusão e quadrados azuis são dados fundidos, oriundos dos sensores sonar e infravermelho.

As medidas passam primeiro pelo agrupamento em tipo, ou seja, os valores medidos pelos dois sonares antes separados, agora são agrupados, o mesmo acontecendo com os dos infravermelhos. Em seguida é feita uma redução em valores acima de 150 centímetros para o fundo de escala (1,5 metros), tanto no infravermelho quanto no sonar. Todos os valores abaixo de 20 centímetros para o infravermelho são ignorados, assim como todas as leituras com valor zero (possíveis erros). Esse tratamento inicial é conhecido como remoção de *outliers*, onde o foco é eliminar medidas equivocadas oriundas dos sensores.

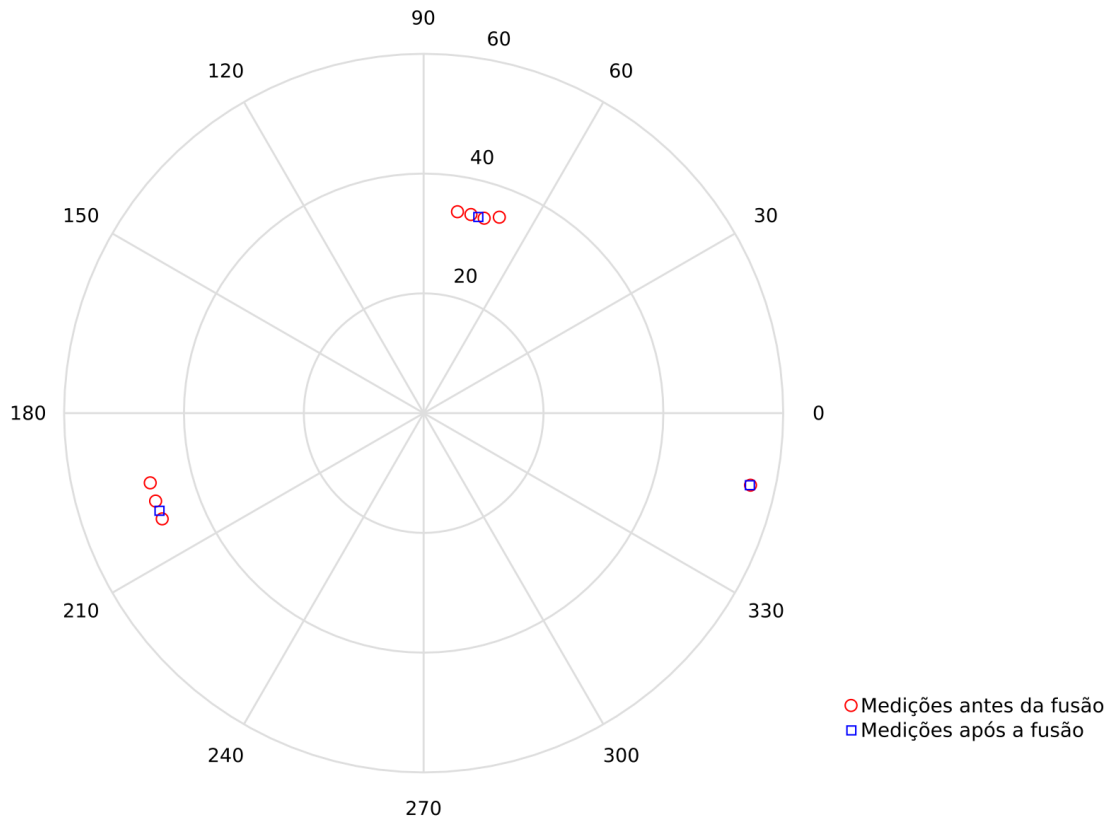
Até esse ponto têm-se dois vetores, um com os dois sonares e outro com os dois infravermelhos. O próximo passo é fazer a fusão dos dois tipos de sensores. Compara-se a medida do sonar com a do infravermelho (pares das partes frontal e traseira) e, se a diferença entre elas estiver dentro de uma tolerância, usa-se a distância do sonar e o ângulo do infravermelho para a composição da medida fundida e armazenada em um novo vetor.

Outra redução é feita, nesse caso, para mais de uma medição consecutiva com distâncias muito próximas (medidas da parede cilíndrica de um mesmo obstáculo), como é o caso ilustrado na Figura 60. Para fundir esses valores, aplica-se a mediana desse subconjunto para se obter o valor central, que representará as coordenadas do obstáculo relativo àquele conjunto medido. Ao fim, os dados são os resultados da fusão sensorial, que simbolizam apenas obstáculos detectados na varredura (total de 360°) de medição por ambos os pares de sensores.

O modelo de medição é então aplicado em cada partícula a fim de se determinar a importância da mesma, ou seja, o peso que dada partícula terá no processo de reamostragem na sequência de execução. O funcionamento do modelo de medição é mostrado no Algoritmo 2, onde  $z_t = (d, \phi)^T$  é composto de um vetor de medições ao longo de 360° (ao redor do robô),  $x_t = (x, y, \theta)^T$  é a partícula ao qual o modelo será aplicado, e  $map = (x, y)^T$  é um vetor de coordenadas dos centros dos obstáculos.

No algoritmo 2,  $N$  é o número de medições e  $M$  é o número de obstáculos. Esse processo é realizado para cada partícula do conjunto  $X'_t$ . A atribuição da variável  $w_t$  é feita baseado na probabilidade da distância e do ângulo medidos em relação à distância e ângulo daquela

Figura 60 – Exemplo de fusão sensorial.



Fonte: (AUTOR)

partícula para o obstáculo ( $\hat{d}$  e  $\hat{\phi}$  são medições ideais). Logo, uma probabilidade mais alta cairá sobre a partícula com maior semelhança em medições ideais em relação ao robô real.

Algoritmo 2:  $w_t = \text{modeloMedicao}(z_t, x_t, \text{map})$

1. *for*  $i = 1$  até  $N$
2.     *for*  $j = 1$  até  $M$
3.          $\hat{d} = \sqrt{(\text{map}^j(1) - x_t(1))^2 + (\text{map}^j(2) - x_t(2))^2} + r$
4.          $\hat{\phi} = \text{atan2}(\text{map}^j(2) - x_t(2), \text{map}^j(1) - x_t(1)) - x_t(3)$
5.          $w_t = \text{prob}(z_t^i(1) - \hat{d}, \sigma_d) * \text{prob}(z_t^i(2) - \hat{\phi}, \sigma_\phi)$
6.     *end for*
7. *end for*
8. **retorne**  $w_t$

Ao final desse algoritmo é composto um vetor de pesos correspondente a cada partícula, esse vetor será a entrada do próximo algoritmo, a reamostragem.

O Algoritmo 3 mostra o processo de reamostragem das partículas, o qual se utiliza dos pesos (importância) das partículas para selecionar qual delas sobreviverá e fará parte da próxima iteração, compondo assim, o último bloco da iteração: o vetor de poses posterior  $X_t$ . O argumento do algoritmo é o conjunto resultado do modelo de movimento  $X'_t$  e o vetor de pesos  $w_t$ .

Algoritmo 3:  $X_t = \text{reamostragem}(X'_t, W_t)$

1.  $r = \text{rand}\left(0; \frac{1}{P}\right)$
2.  $c = w_t(1)$
3.  $i = 1$
4. *for*  $p = 1$  até  $P$
5.      $U = r + (p - 1) * \frac{1}{P}$
6.     *while*  $U > c$
7.          $i = i + 1$
8.          $c = c + w_t(i)$
9.     *end while*
10.     *acrescentar*  $X'_t(t)$  a  $X_t$
11. *end for*
12. **retorne**  $X_t$

A reamostragem realizada neste algoritmo é conhecida como sendo de baixa variância e é feita de acordo com a geração de um único número aleatório  $r$  entre 0 e  $P^{-1}$ , com  $P$  sendo o número total de partículas. Esse número é então comparado com o vetor de pesos, que permite selecionar as partículas proporcionalmente ao valor de sua importância. A complexidade desta técnica cresce linearmente com o número de partículas, enquanto que para outras técnicas como as que fazem buscas aleatórias, crescem em escala logarítmica com o número de partículas (THRUN *et al.* 2006). Considerando que o custo computacional reduzido é crucial para o desempenho do filtro, menos complexidade resulta em menos custo e mais velocidade, o que se considera uma escolha acertada, concordando com o caráter baixo custo do trabalho.

A recursividade do filtro acontece nesse ponto, pois o resultado  $X_t$  do Algoritmo 3 servirá de entrada para o Algoritmo 1, substituindo  $X_{t-1}$  em uma nova iteração, juntamente com um novo controle  $u_t$  proveniente do movimento seguinte realizado pelo robô.

O último código a ser executado na árvore de dependências, o **<plotAll>**, se resume a uma sequência de funções de plotagem nativas do MATLAB para visualização dos elementos de forma gráfica. Esse procedimento permitiu avaliar graficamente a evolução dinâmica do filtro. O código mostra os principais elementos na execução de cada iteração do código **<particleFilter>**:

- a) Os limites do tablado;
- b) Os obstáculos;
- c) O conjunto  $X'_t$  (após aplicação do modelo de movimento);
- d) O conjunto  $X_t$  (após reamostragem);
- e) Uma pose representativa do conjunto  $X_t$  (média das partículas); e
- f) A pose real obtida através da câmera;

Os dois últimos ('e' e 'f') são parâmetros comparativos, utilizados na análise de convergência. Mais informações poderiam ser adicionadas, porém, devido à grande quantidade de partículas sendo mostradas em cada iteração, apenas as principais informações de análise foram escolhidas.

Pode-se resumir o algoritmo de localização utilizando filtro de partículas como mostra o Algoritmo 4, os argumentos são o conjunto de partículas inicial  $X_{t-1}$ , o controle  $u_t$ , as medições  $z_t$  e as coordenadas dos obstáculos  $map$

Algoritmo 4:  $X_t = MCL(X_{t-1}, u_t, z_t, map)$

1.  $X_t = W_t = \emptyset$
2. *for*  $p = 1$  até  $P$
3.      $x'_t(p) = amostragemModeloOdometrico(u_t, x_{t-1}(p))$
4.      $w_t(p) = modeloMedicao(z_t, x'_t(p), map)$
5.     *acrescentar*  $x'_t(p)$  a  $X'_t$
6.     *acrescentar*  $w_t(p)$  a  $W_t$
7. *end for*
8.  $X_t = reamostragem(X'_t, W_t)$
9. **retorne**  $X_t$

É importante ressaltar que o uso do *software* MATLAB no desenvolvimento do módulo externo foi dado pela facilidade e agilidade no desenvolvimento do filtro e de outros módulos chamados por ele. Porém, é possível o desenvolvimento do módulo externo completamente em linguagens de programação como o C ANSI ou Python, podendo haver, inclusive, a união de mais de uma linguagem configuradas para executarem em conjunto. Uma alternativa para tal implementação seria a utilização da ferramenta MATLAB Coder<sup>23</sup>, que exporta a aplicação desenvolvida no ambiente MATLAB para outras linguagens como C e C++, podendo ser usada de forma embarcada pelo robô.

Assim, finaliza-se o detalhamento dos módulos de *software* presentes no trabalho. O próximo capítulo apresenta os experimentos realizados e os resultados obtidos em três tipos de experimentos, dois deles usando mapas simétricos e um com bastante assimetria, objetivando tanto validar a eficácia da técnica de triangulação para reduzir a área de amostragem inicial do ambiente, quanto para verificar a capacidade de convergência do filtro e em quantas poses isso já ocorre em ambas as situações.

---

<sup>23</sup> Site: <https://www.mathworks.com/products/matlab-coder.html>

## **6 Experimentos e resultados**

Este capítulo apresenta a realização dos experimentos e os resultados obtidos com o robô a fim de validar a solução do problema de localização global, usando a abordagem probabilística. Primeiro serão mostrados os testes feitos para levantamento dos dados de ruído de movimento e de medição, necessários para uso pelo filtro MCL. Posteriormente serão apresentados os três experimentos envolvendo a tarefa de localização global pelo filtro MCL, em ambientes com grau de simetria diversos para a validação da técnica de triangulação utilizada na amostragem inicial das partículas. Todos os experimentos foram realizados no LRC e executados no tablado (ambientes de testes), mencionado na Seção 3.4.

### **6.1 Testes de repetição para levantamento de ruído**

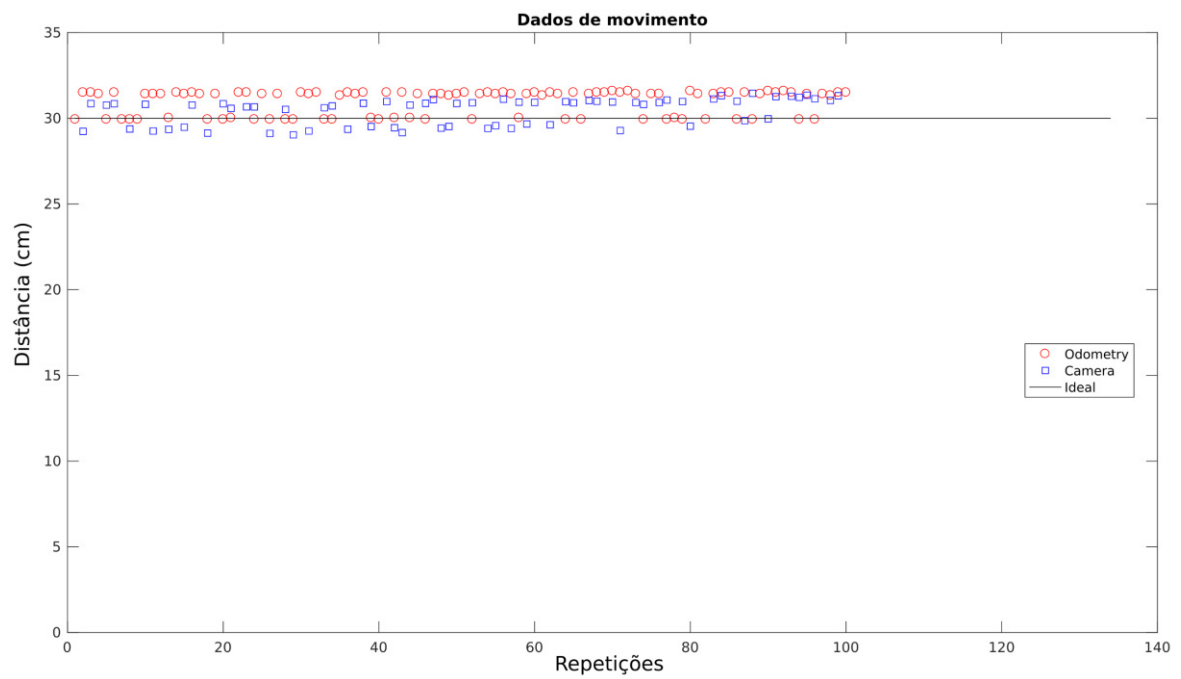
Os modelos de movimento e medição probabilísticos utilizam a PDF, com muita frequência a gaussiana ou normal, a qual necessita de uma média e um desvio-padrão para que o MCL possa computar as probabilidades envolvidas e citadas no capítulo anterior. Para incluir o desvio-padrão nessas densidades, é necessária a realização de testes repetitivos que possam conter uma quantidade de amostras que permitam definir numericamente os ruídos a serem modelados. Os testes de movimento são os de translação e de rotação em distâncias e ângulos fixos, respectivamente, enquanto que os de medição são feitos apenas com distâncias com o robô e os obstáculos parados, porém em distanciados de valores distintos.

A câmera foi utilizada para a rápida aquisição das medidas de distância nos testes de movimentos, através do cálculo de distância entre dois pontos em duas imagens sobrepostas, contendo as informações do centroide da base do robô no instante de partida e no final do movimento. Sendo assim, realizou-se todo o processo dos testes de forma automática, tendo sido realizadas 100 repetições em cada teste que considerou uma determinada distância.

#### **6.1.1 Testes de movimentos**

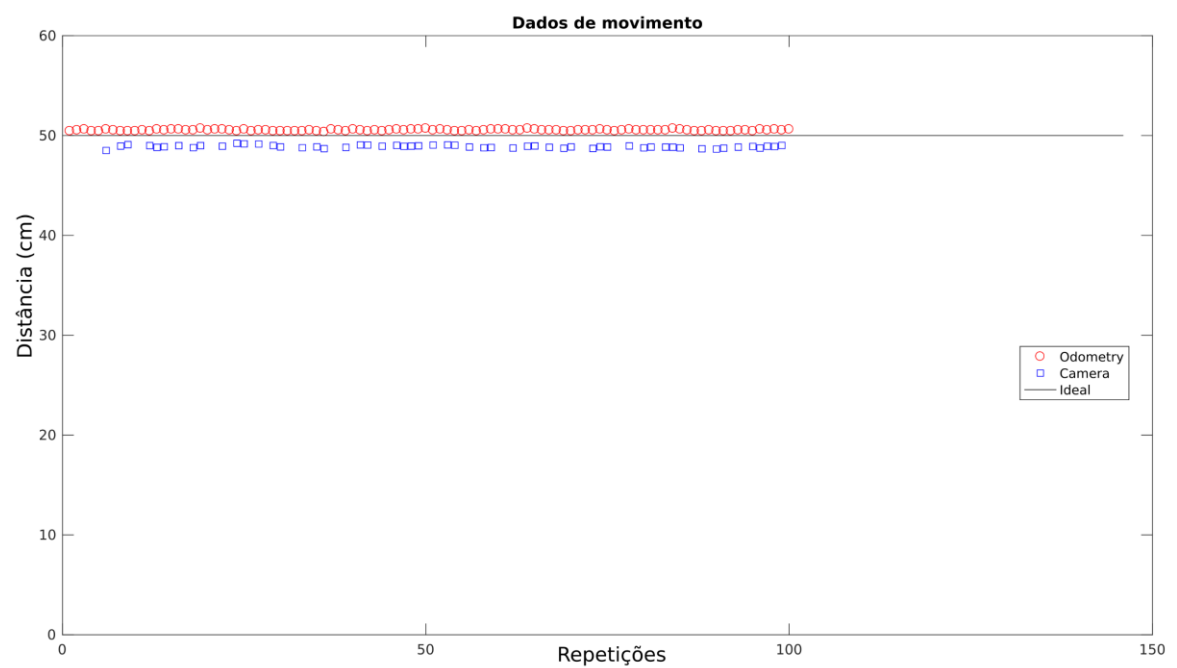
Os testes feitos foram nas distâncias de 30, 50 e 100 centímetros para a translação e 20, 45 e 60 graus para rotação. Os resultados mostrados nas Figuras 61 a 63 ilustram graficamente os dados adquiridos da odometria e da câmera. Os valores dos erros foram calculados subtraindo-se os valores do odômetro pelos obtidos pela câmera, normalizado pelos da câmera.

Figura 61 – Resultado dos testes de translação em 30 cm.



Fonte: (AUTOR)

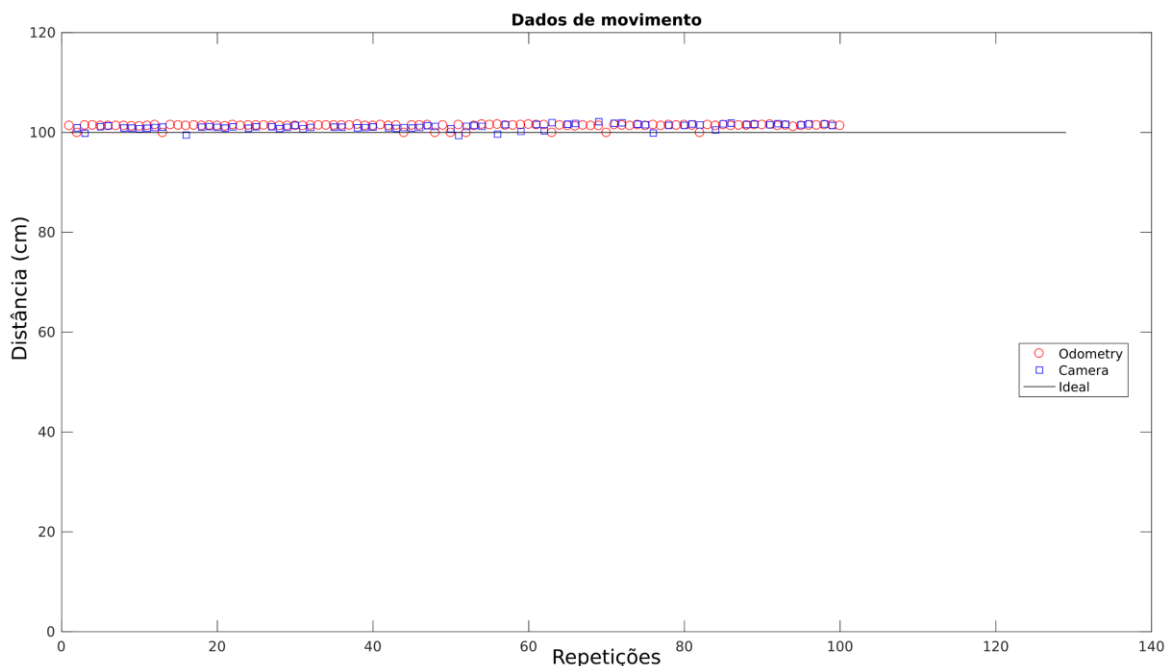
Figura 62 – Resultado dos testes de translação em 50 cm.



Fonte: (AUTOR)



Figura 63 – Resultado dos testes de translação em 100 cm.



Fonte: (AUTOR)

Os desvios-padrões e erros médios produzidos nos testes estão inseridos na Quadro 2. A diferença dos erros de translação não ultrapassa 3 centímetros, os quais foram aumentando com a distância. O mesmo fenômeno ocorreu com os ângulos, onde o erro não foi maior do que 4 graus. Os desvios-padrões foram calculados com as amostras coletadas se utilizando da função `<normfit()>` da *toolbox* `<Statistics and Machine Learning>` do MATLAB.

Quadro 2 – Desvios-padrões e erros normalizados de movimentos.

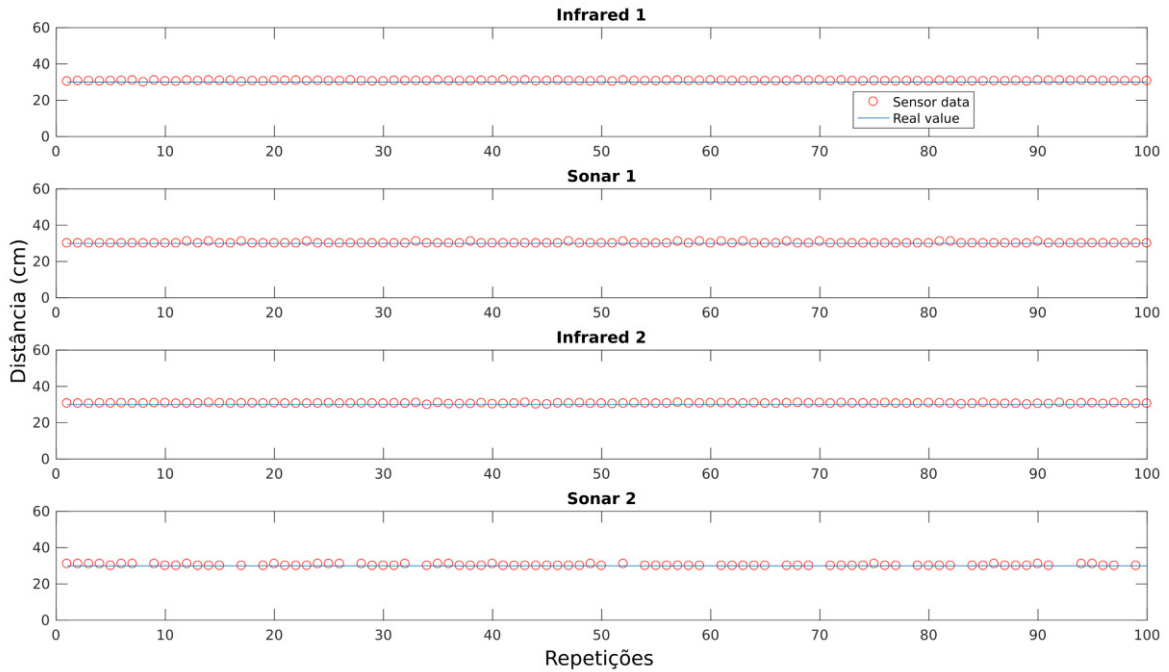
<b>Distâncias testadas</b>	<b>Desvio padrão</b>	<b>Erros normalizados</b>
Translação 30 cm	0,7160	0,0269
Translação 50 cm	0,0754	0,05
Translação 100 cm	0,4325	0,0171
Rotação 20°	0,051	0,017
Rotação 45°	0,059	0,0176
Rotação 60°	0,081	0,065

### 6.1.2 Testes de medições

Os testes de medições com os sensores de distância foram feitos com o robô e a torre de sensores parados e dois obstáculos (um a frente e outro atrás) distanciados igualmente do robô. As distâncias testadas foram 30, 50, 100 e 120 cm. Os limites de 30 (mínima) e 120 (máxima) foram definidos em virtude do alcance do sensor infravermelho e também da

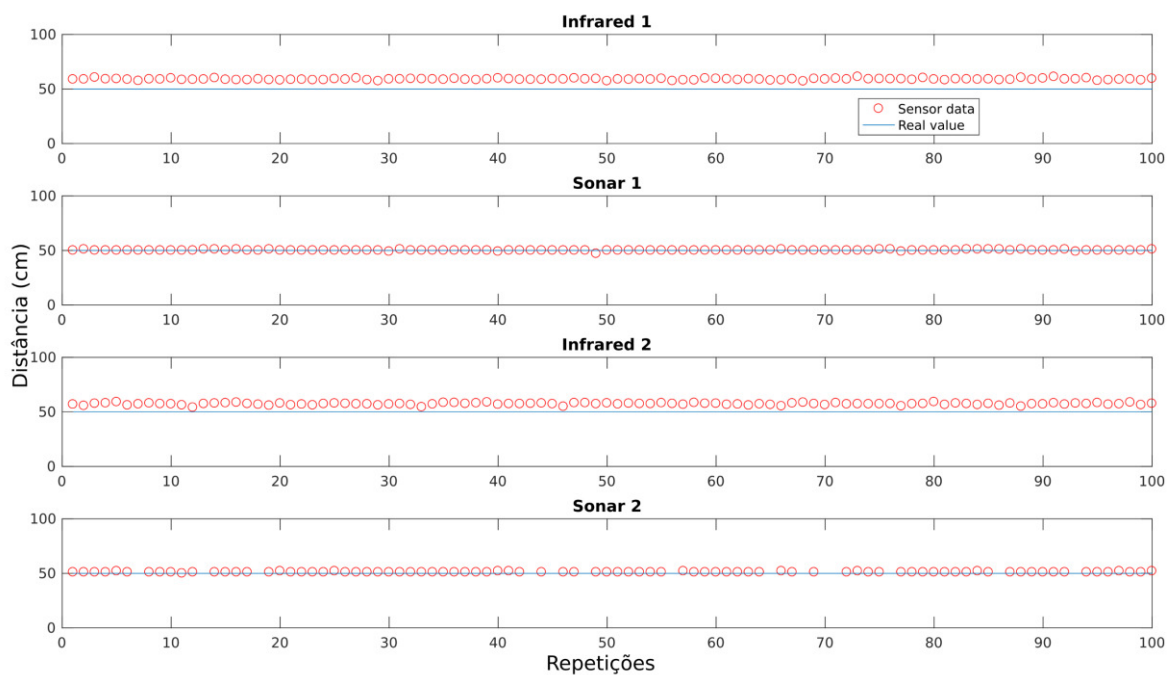
relativa pequena área dos testes, restrições das dimensões do tablado. As Figuras 64 a 67 ilustram os dados colhidos dos quatro sensores e uma linha constante na distância testada.

Figura 64 – Resultado dos testes de medição em 30 cm.



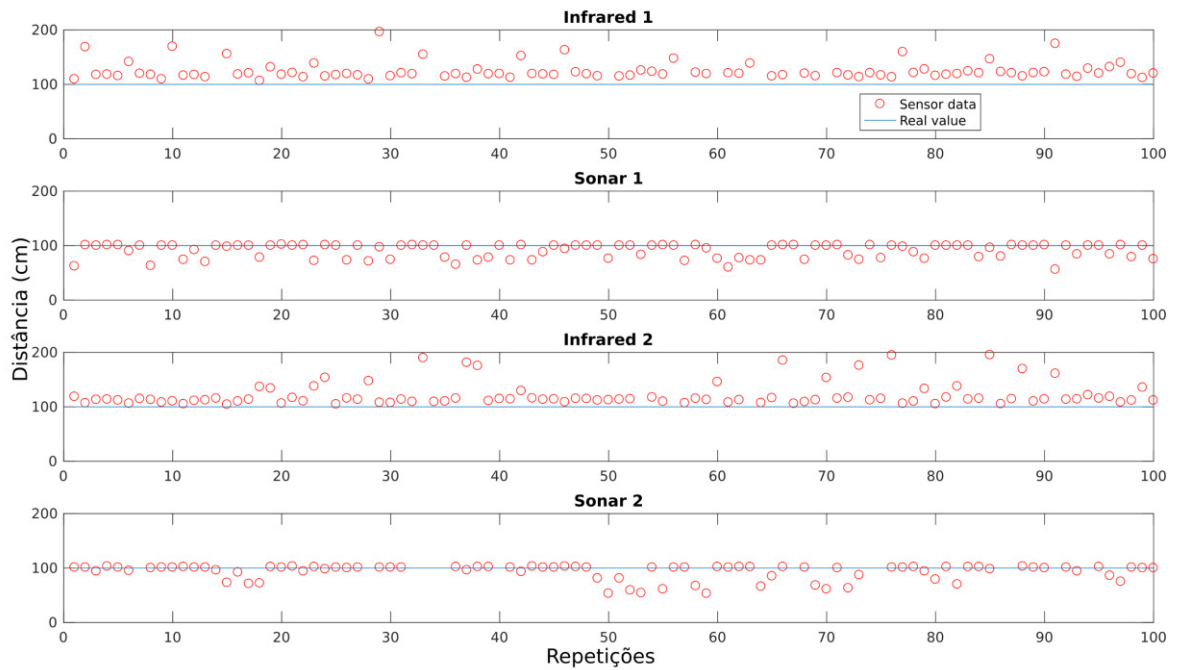
Fonte: (AUTOR)

Figura 65 – Resultado dos testes de medição em 50 cm.



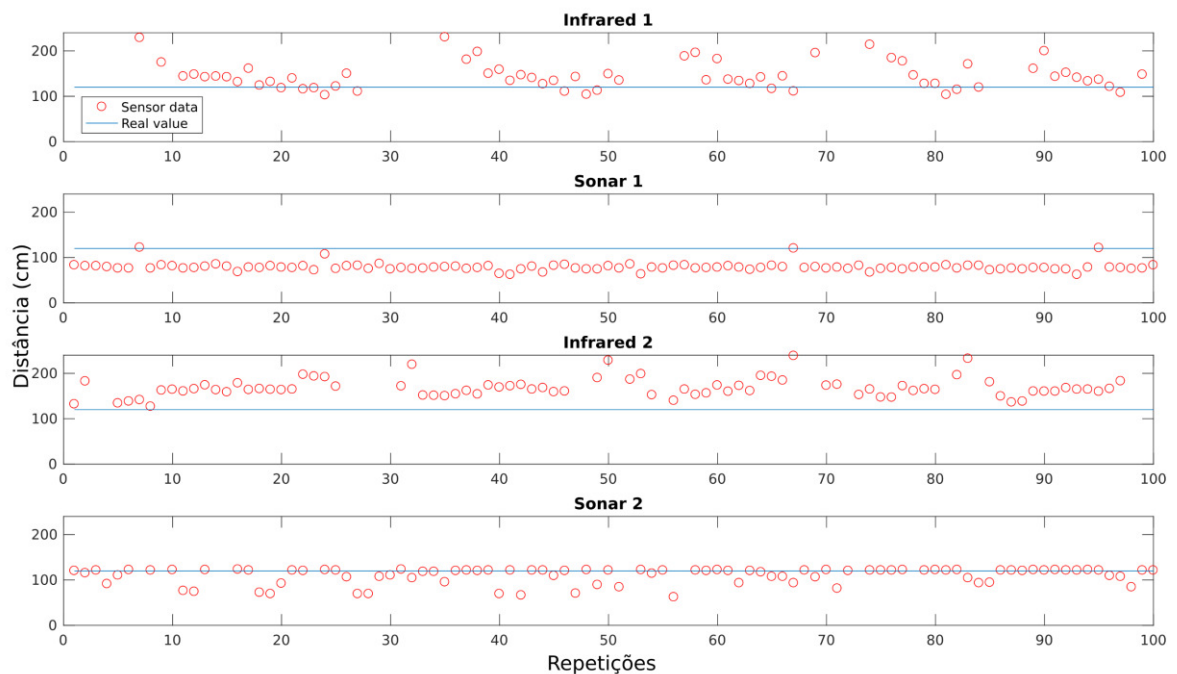
Fonte: (AUTOR)

Figura 66 – Resultado dos testes de medição em 100 cm.



Fonte: (AUTOR)

Figura 67 – Resultado dos testes de medição em 120 cm.



Fonte: (AUTOR)

Os sensores Sonar 1 e IR 1 são os posicionados a frente do robô, enquanto que Sonar 2 e IR 2 ficam virados para trás (pares defasados de 180°). Alguns valores não aparecem em

algumas sequências de repetições (principalmente nos infravermelhos), tendo sido removidos por caracterizarem *outliers*.

Quadro 3 – Desvios-padrões para testes repetitivos de medições.

	<b>30 cm</b>	<b>50 cm</b>	<b>100 cm</b>	<b>120 cm</b>
<b>Infravermelho 1</b>	0,1907	0,7348	16,0403	25,0802
<b>Infravermelho 2</b>	0,2174	0,9444	21,7626	15,8674
<b>Sonar 1</b>	0,3775	0,5336	12,9491	9,2745
<b>Sonar 2</b>	0,4545	0,3571	14,3725	17,6678

Os erros médios associados a cada sensor oriundos dos testes estão contidos na Quadro 3. Pode-se concluir que os sensores sonares são mais precisos que os infravermelhos, e com o aumento da distância, notou-se neste último um desvio-padrão muito elevado, o que torna o seu uso praticamente inviabilizado para distâncias acima de 100 cm, motivo pelo qual se optou por usar as distâncias fornecidas pelo sonar e não do infravermelho no processo de fusão. Já para pequenas distâncias, ambos os sensores se mostraram confiáveis.

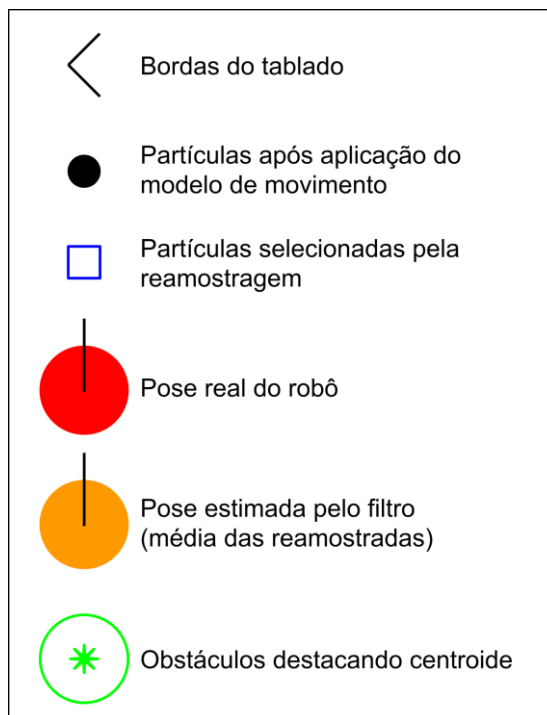
## 6.2 Experimentos de localização

Na realização dos experimentos de localização, o algoritmo MCL foi testado em três condições do ambiente com o propósito de analisar sua robustez na convergência das poses estimadas. As imagens mostradas nesta seção são oriundas do código me MATLAB `<plotAll.m>`, que demonstram os elementos do algoritmo com visão idêntica à perspectiva da câmera, conforme comentado na Seção 5.5. Os elementos presentes nas figuras que seguem, mais adiante utilizam a simbologia apresentada na Figura 68.

O primeiro experimento é simples em termos de configuração do mapa e movimentos do robô. O objetivo desse experimento é analisar o número de iterações necessárias para a convergência do filtro, ou seja, quantas poses são necessárias para as partículas se agruparem bem próximas ao robô de forma a minimizar o erro em relação à condição inicial.

Na Figura 69 é mostrada a foto do ambiente com a configuração dos obstáculos e a primeira pose do robô. Observa-se que o mapa possui simetria (verticalmente), entretanto, o trajeto navegado pelo robô acontece na linha comum entre as duas regiões simétricas, a fim de isolar o objetivo do experimento com a exclusão do problema de simetria. O controle aplicado ao robô foi de vinte poses, compostas apenas de translações de cinco centímetros (sem rotações), com exceção das poses 10 e 11 que possuem no controle informações de ambas as rotações de 90 graus (sem translação) em seus controles. Em resumo, sua trajetória foi uma

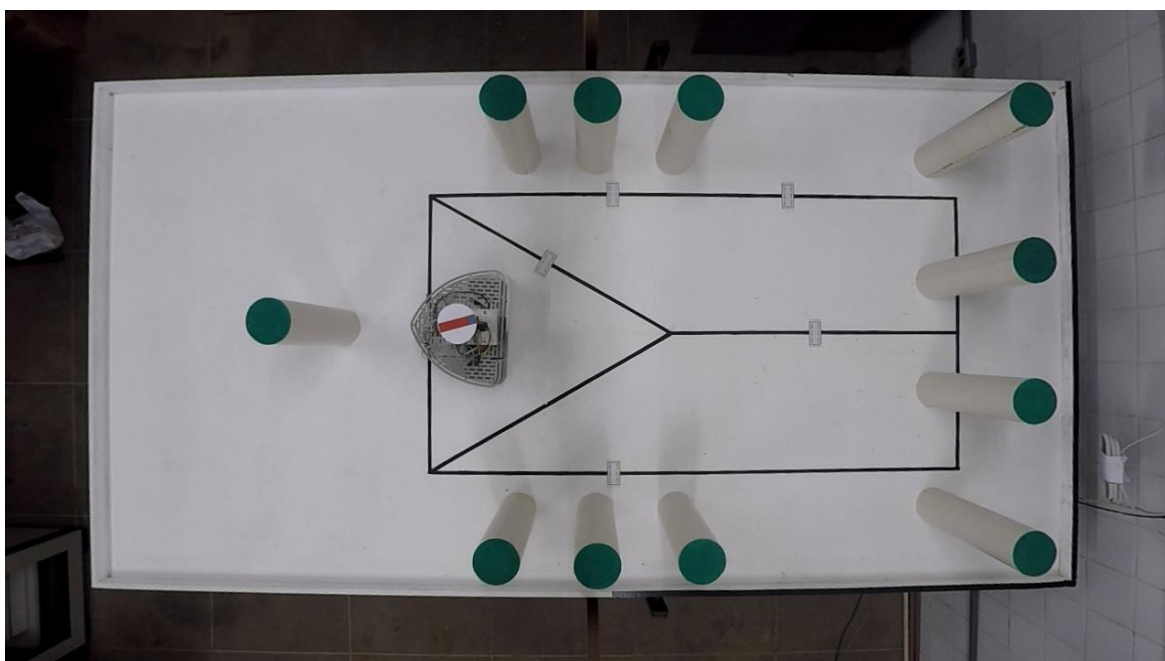
Figura 68 – Simbologia dos elementos presentes nas figuras dos resultados dos experimentos.



Fonte: (AUTOR)

linha reta por 50 centímetros, um giro de 180 graus, seguido de uma linha reta por mais 50 centímetros.

Figura 69 – Foto ilustrando a primeira pose e o mapa (primeiro experimento).



Fonte: (AUTOR)

A Figura 70 apresenta o resultado da primeira iteração do algoritmo, mostrando as partículas antes e depois da reamostragem, além das poses real e estimada do robô, esta última formada pela média das partículas reamostradas. Nas Figuras 71 a 73 são mostrados os resultados para as poses 3, 8 e 20 respectivamente, usando a convenção da Figura 68.

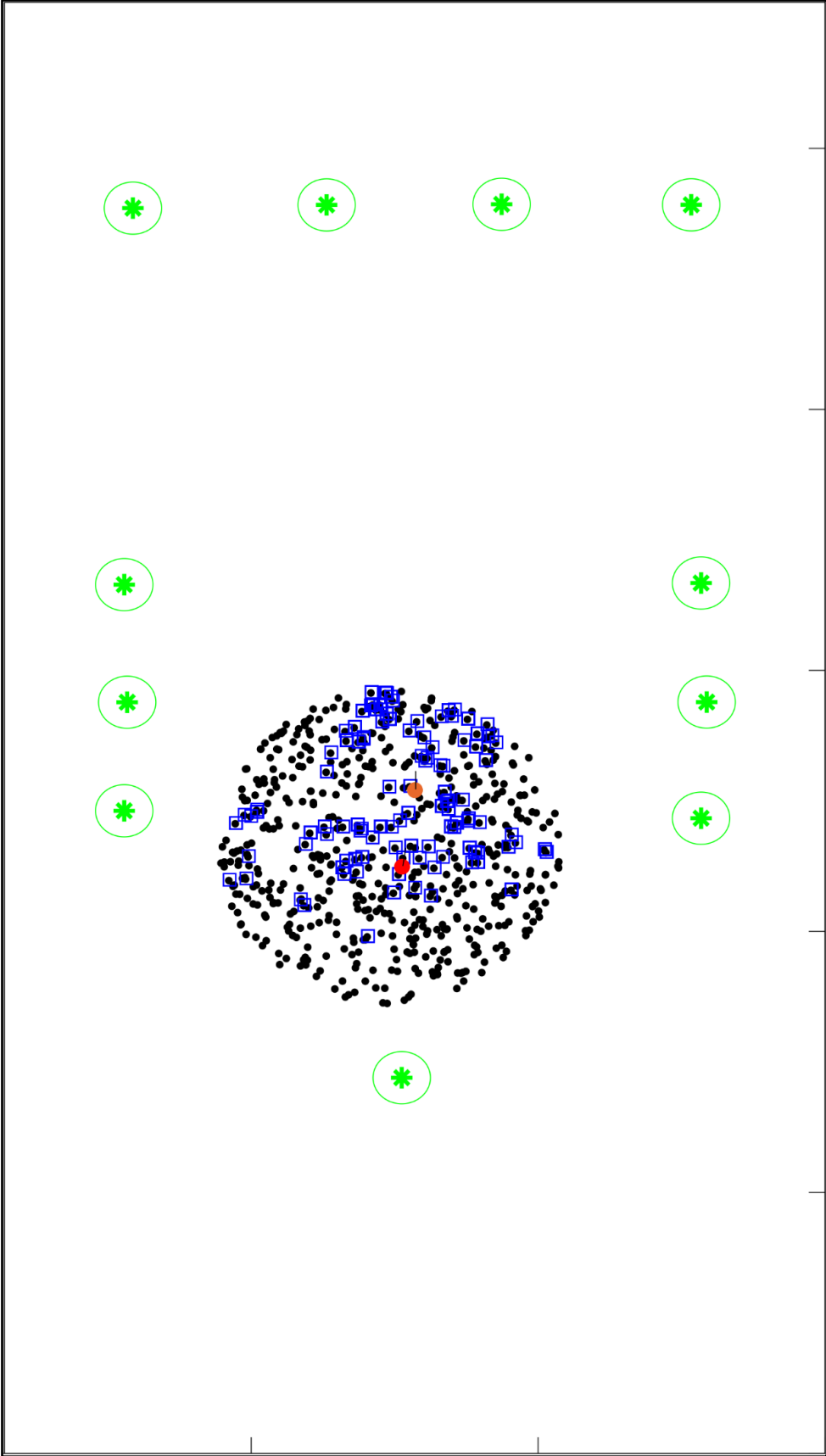
As imagens mostram a convergência do filtro a começar da pose 8, tendo sido observados nos demais testes ainda a serem apresentados. Além disso, observou-se que a convergência não é perdida nas poses subsequentes, caracterizando o rastreamento local descrito no capítulo 2. Em outras palavras, o filtro mantém as convergências das poses, requisito fundamental para seu funcionamento na solução da tarefa, no caso a localização global.

Um segundo experimento foi feito, porém dessa vez com a finalidade de analisar a robustez do filtro quanto à simetria do mapa, considerando a técnica de triangulação para gerar mais de um espaço de amostragem (*clusters*) de partículas. Nesse experimento o objetivo foi comprovar que, considerando mapas simétricos que produz mais de um espaço de amostragem, o filtro é capaz de manter a mesma eficiência que em mapas assimétricos. A verificação da associação de dados por máxima verossimilhança aplicado aos *clusters* permite que apenas o conjunto de partículas onde realmente ele se encontra sobreviva no processo de reamostragem e consiga a convergência.

A Figura 74 mostra a foto do mapa aplicado nesse segundo experimento e a primeira pose do robô, a Figura 75 a trajetória feita pelo robô de forma simplificada e nas Figuras 76 a 79 são apresentados os resultados visuais do algoritmo detalhando as partículas e o mapa.

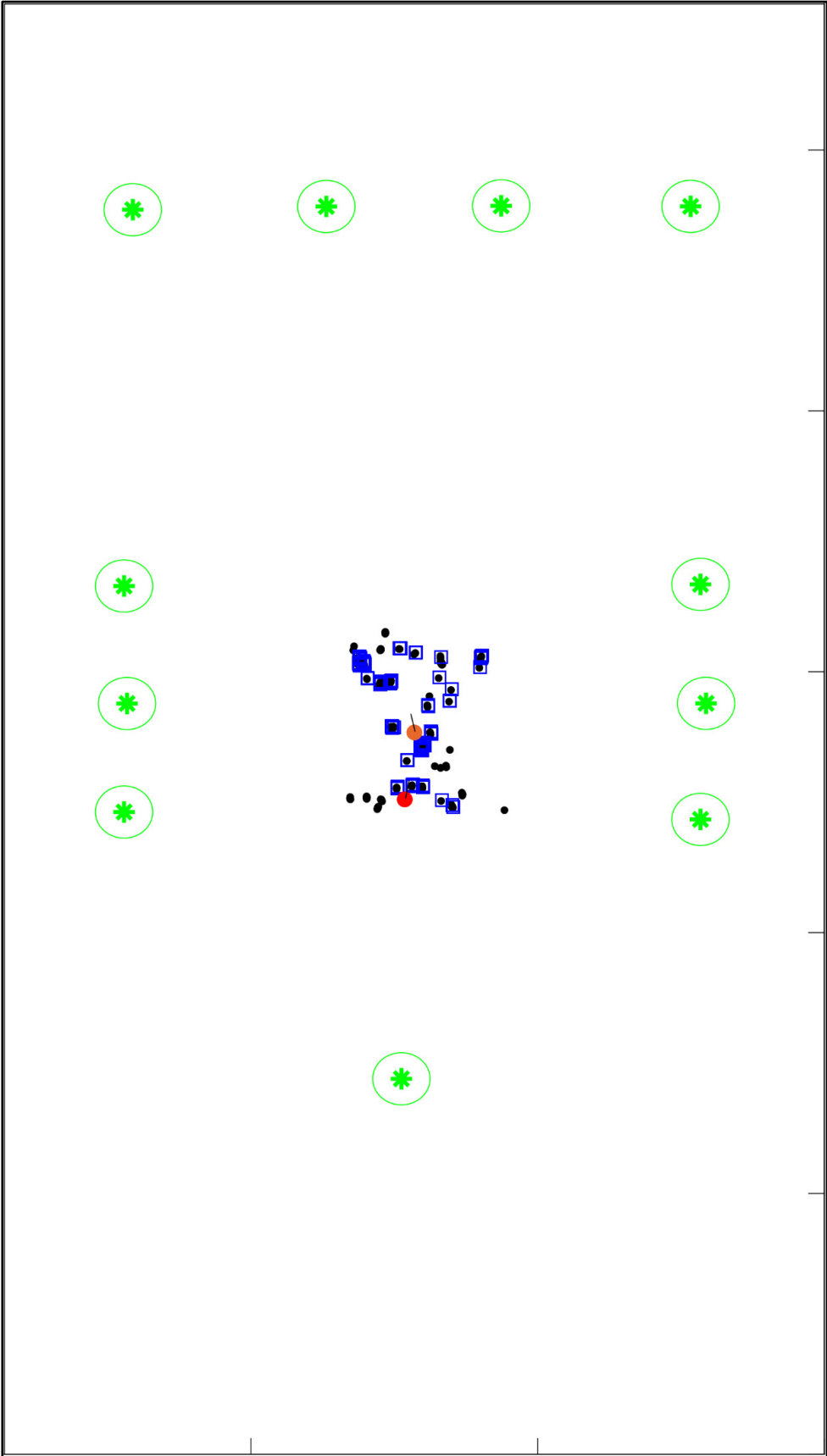
As imagens mostram dois *clusters* formados no mapa, produzidos pela técnica de triangulação de objetos. Esse número não unitário revela duas regiões com os marcos posicionados a distâncias aproximadas, de tal maneira que no processo de triangulação apareçam essas duas possíveis poses iniciais do robô, em torno das quais as partículas são amostradas. Após o espalhamento das partículas nesses dois *clusters*, o algoritmo MCL é iniciado e o robô é comandado para as poses seguintes, sendo que em cada uma delas as medidas são adquiridas, fundidas e incorporadas. Nesse processo recursivo, as partículas são pesadas e passam pelo processo de reamostragem, fazendo com que as hipóteses de maior peso sobrevivam no próximo conjunto que é usado para movimentar o robô para a próxima pose. A partir da pose 9, o filtro converge e segue até a última pose (15), rastreando o robô com erros entre as poses estimadas e reais em níveis aceitáveis.

Figura 70 – Experimento 1 - Resultado do algoritmo após processamento da pose 1.



Fonte: (AUTOR)

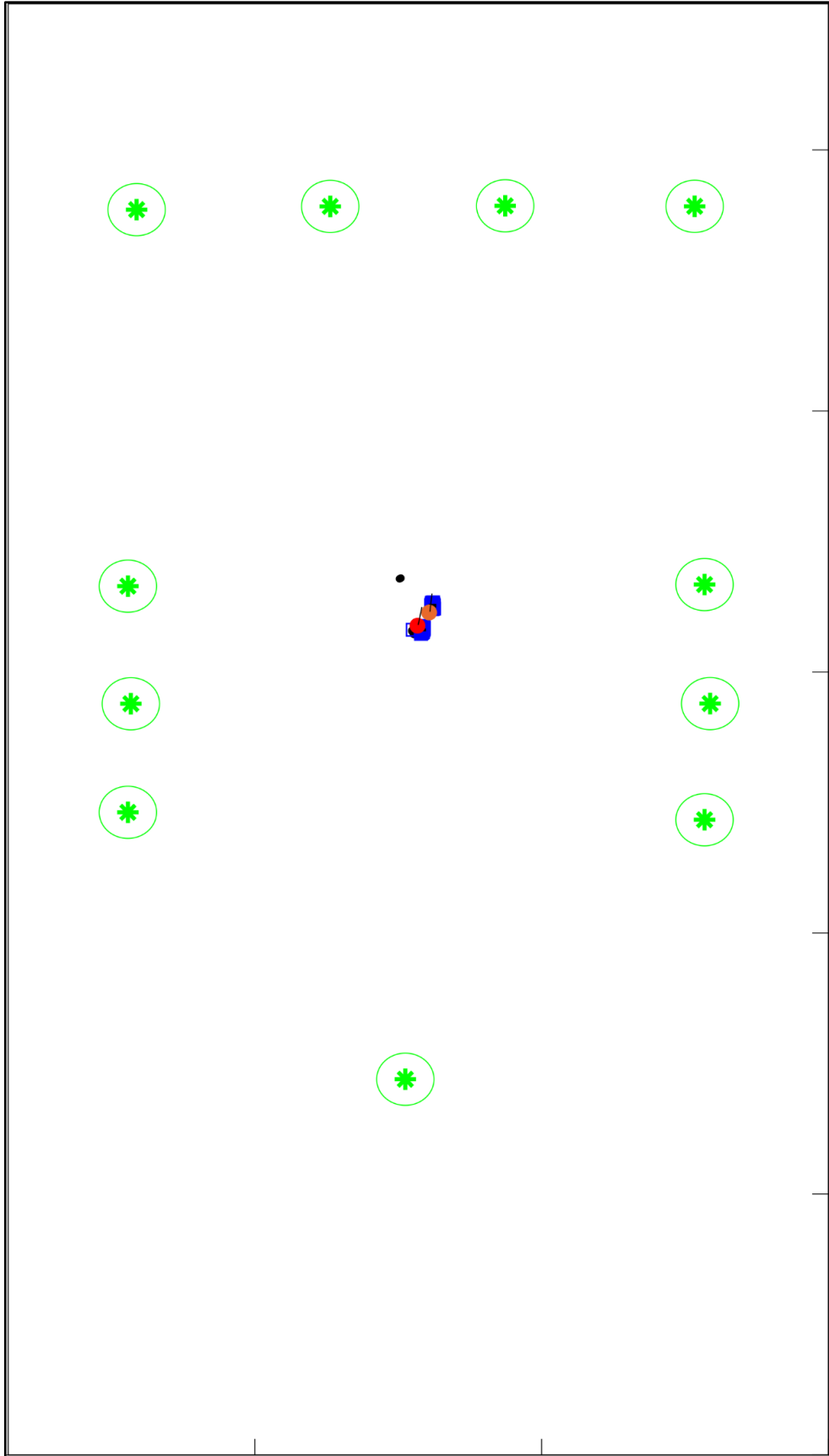
Figura 71 – Experimento 1 - Resultado do algoritmo após processamento da pose 3.



Fonte: (AUTOR)

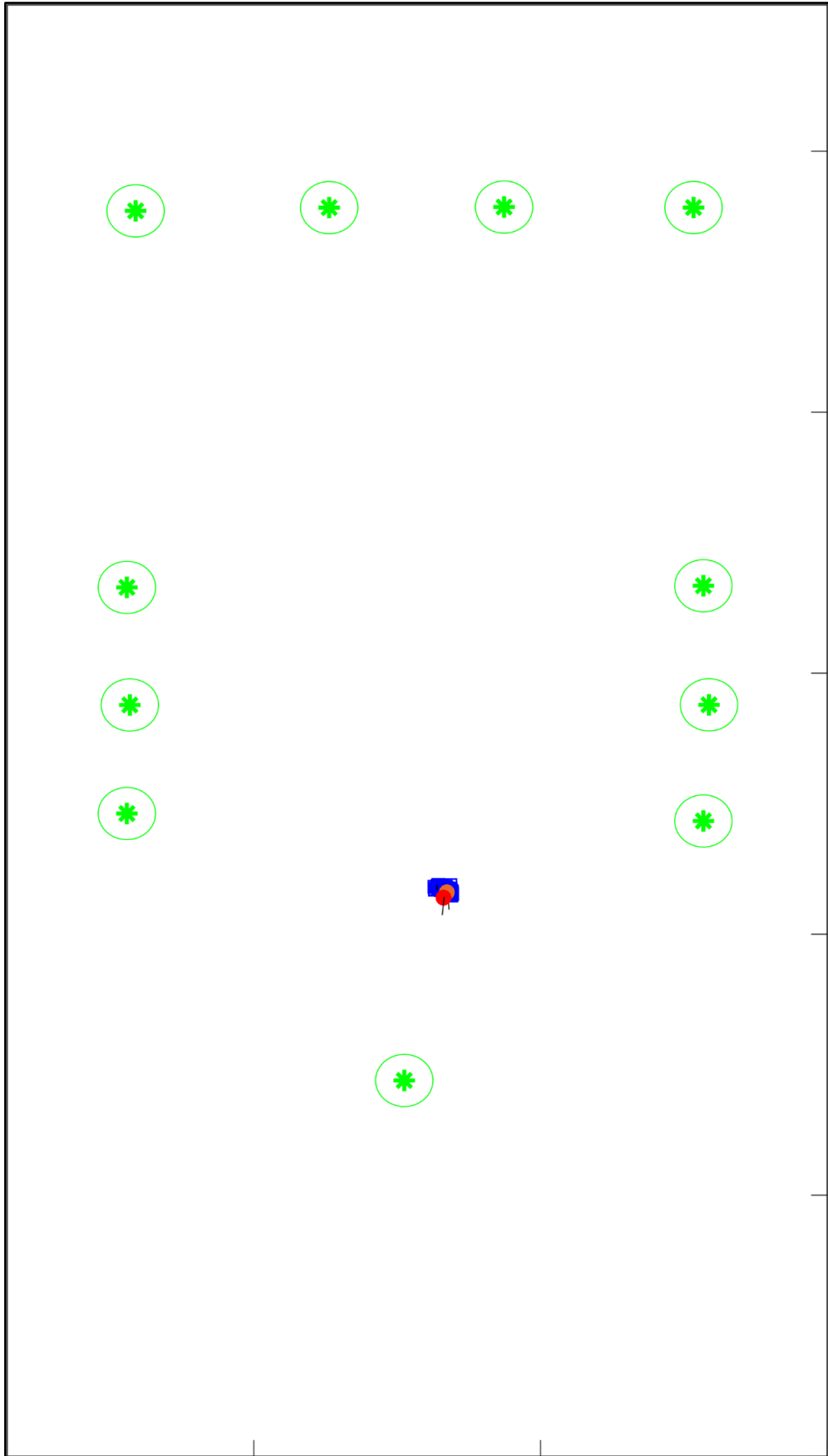


Figura 72 – Experimento 1- Resultado do algoritmo após processamento da pose 8.



Fonte: (AUTOR)

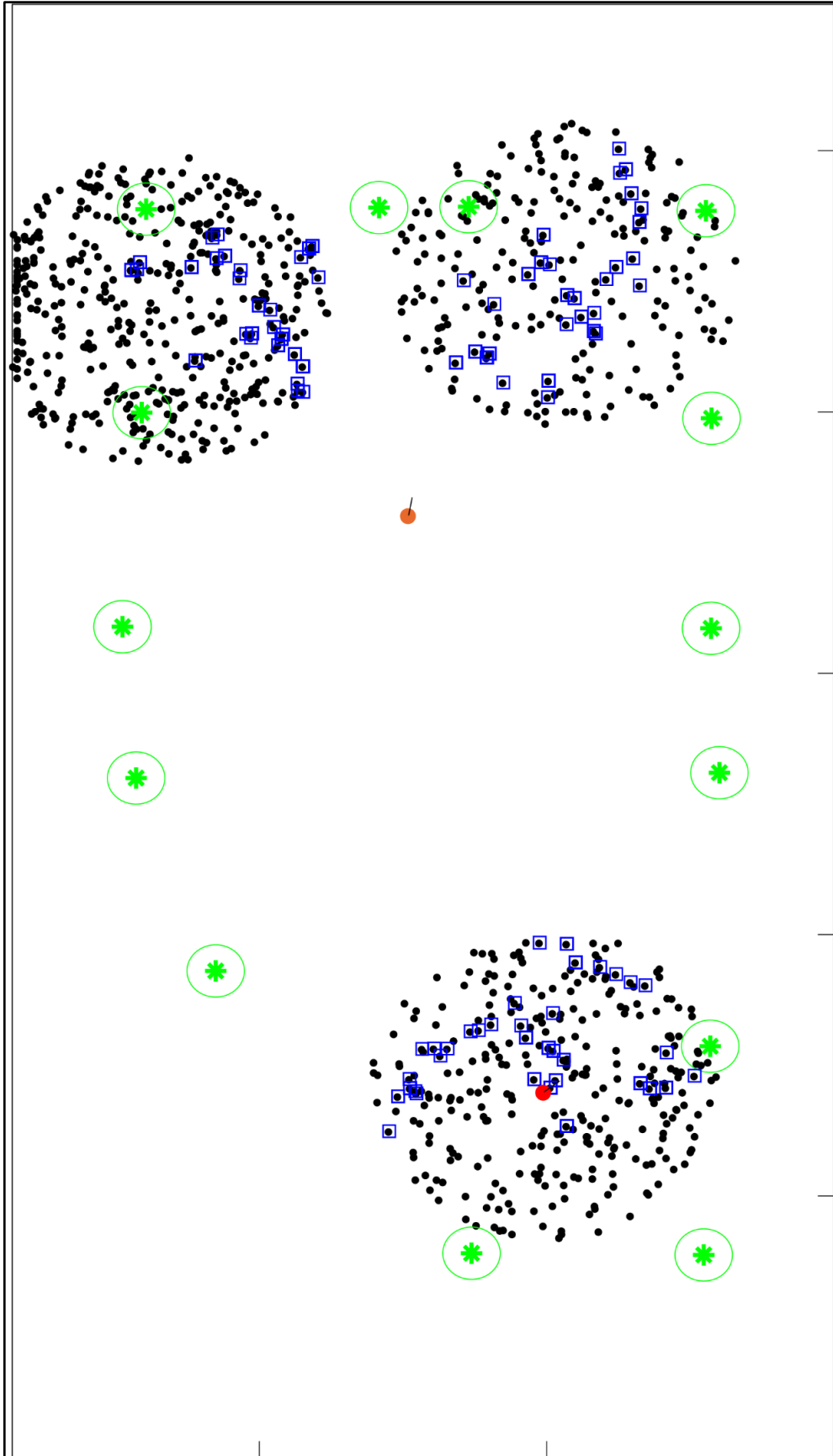
Figura 73 – Experimento 1- Resultado do algoritmo após processamento da pose 20.



Fonte: (AUTOR)

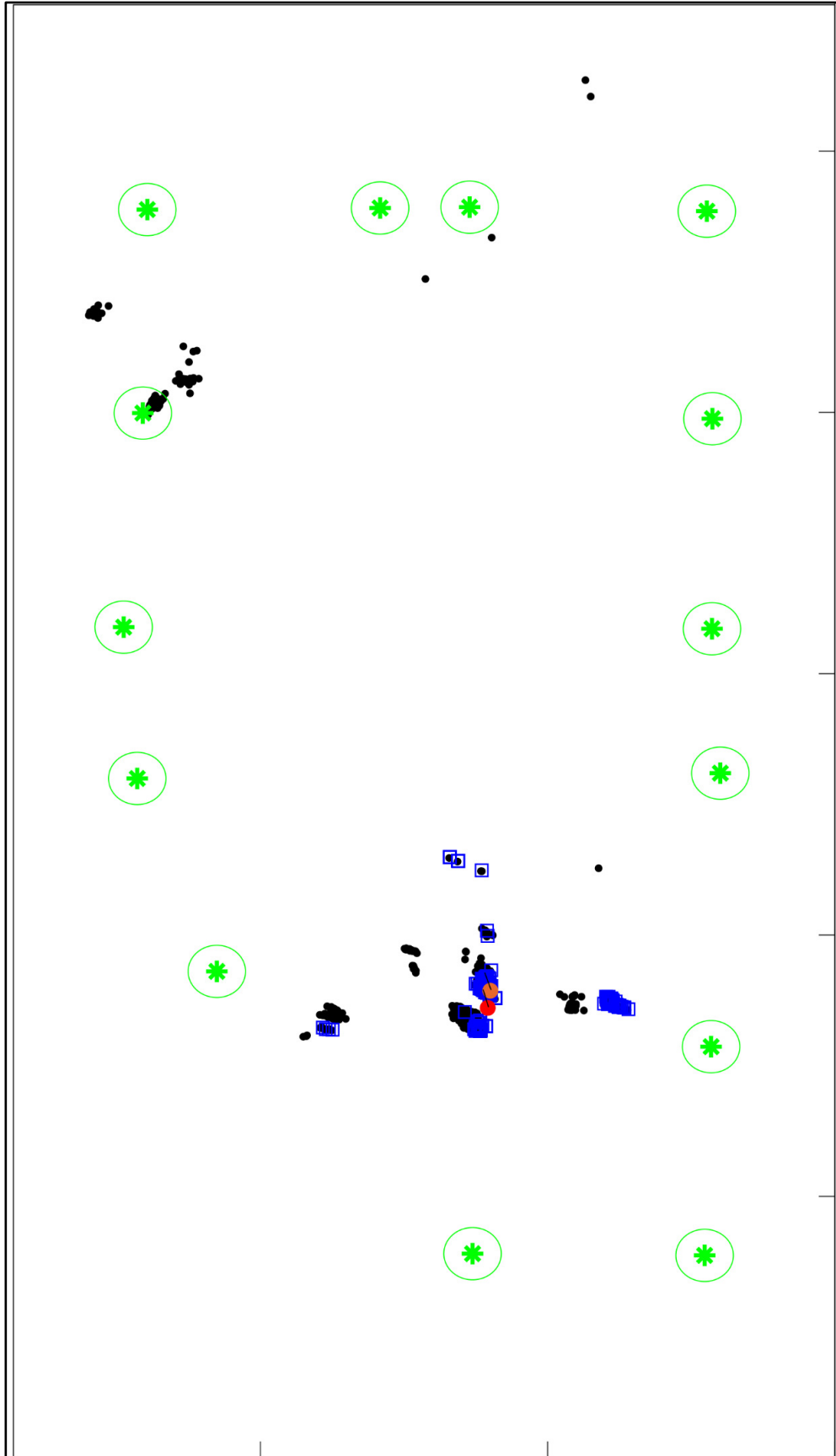


Figura 76 – Experimento 2 - Resultado do algoritmo após processamento da pose 1.



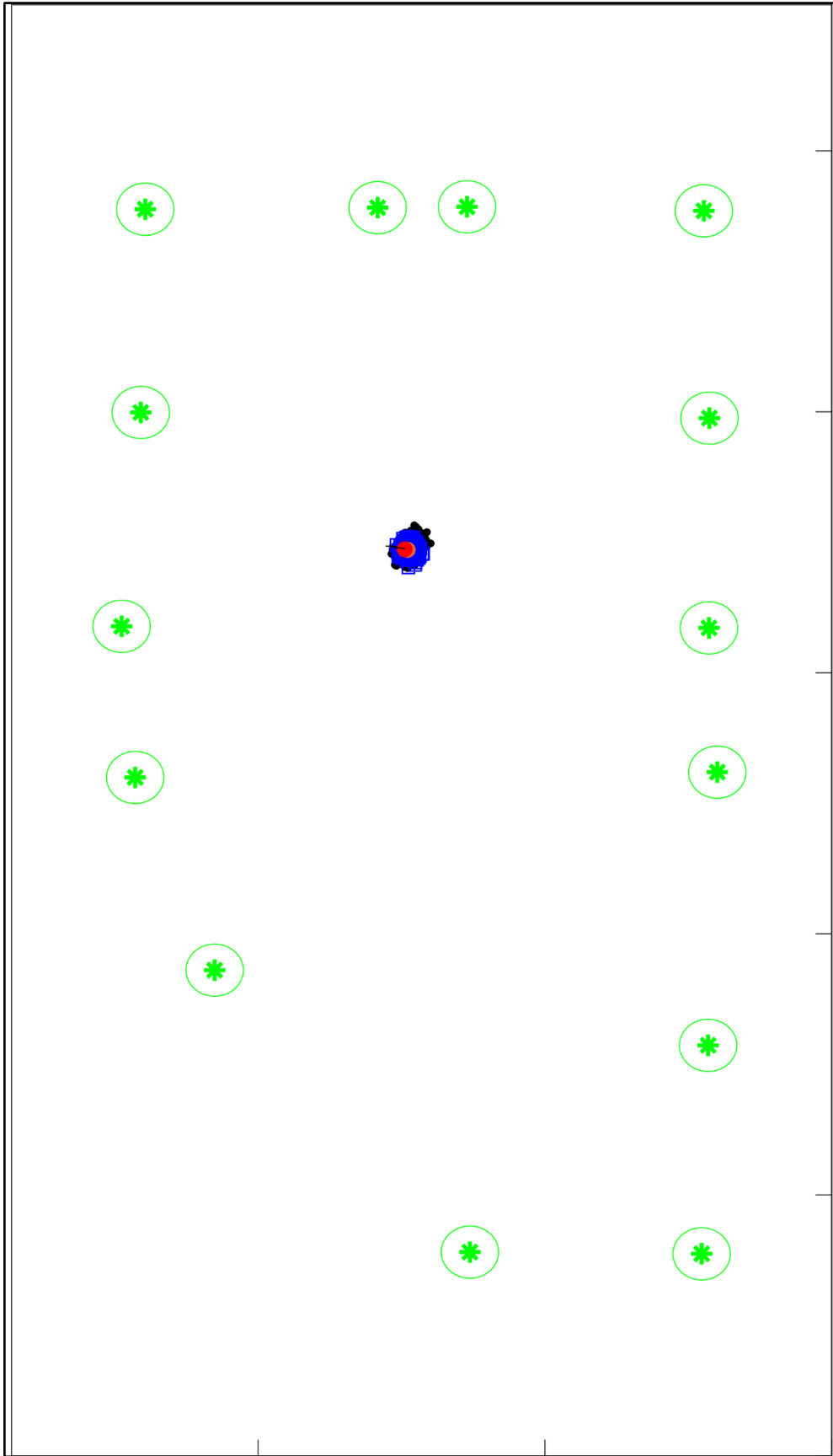
Fonte: (AUTOR)

Figura 77 – Experimento 2 - Resultado do algoritmo após processamento da pose 3.



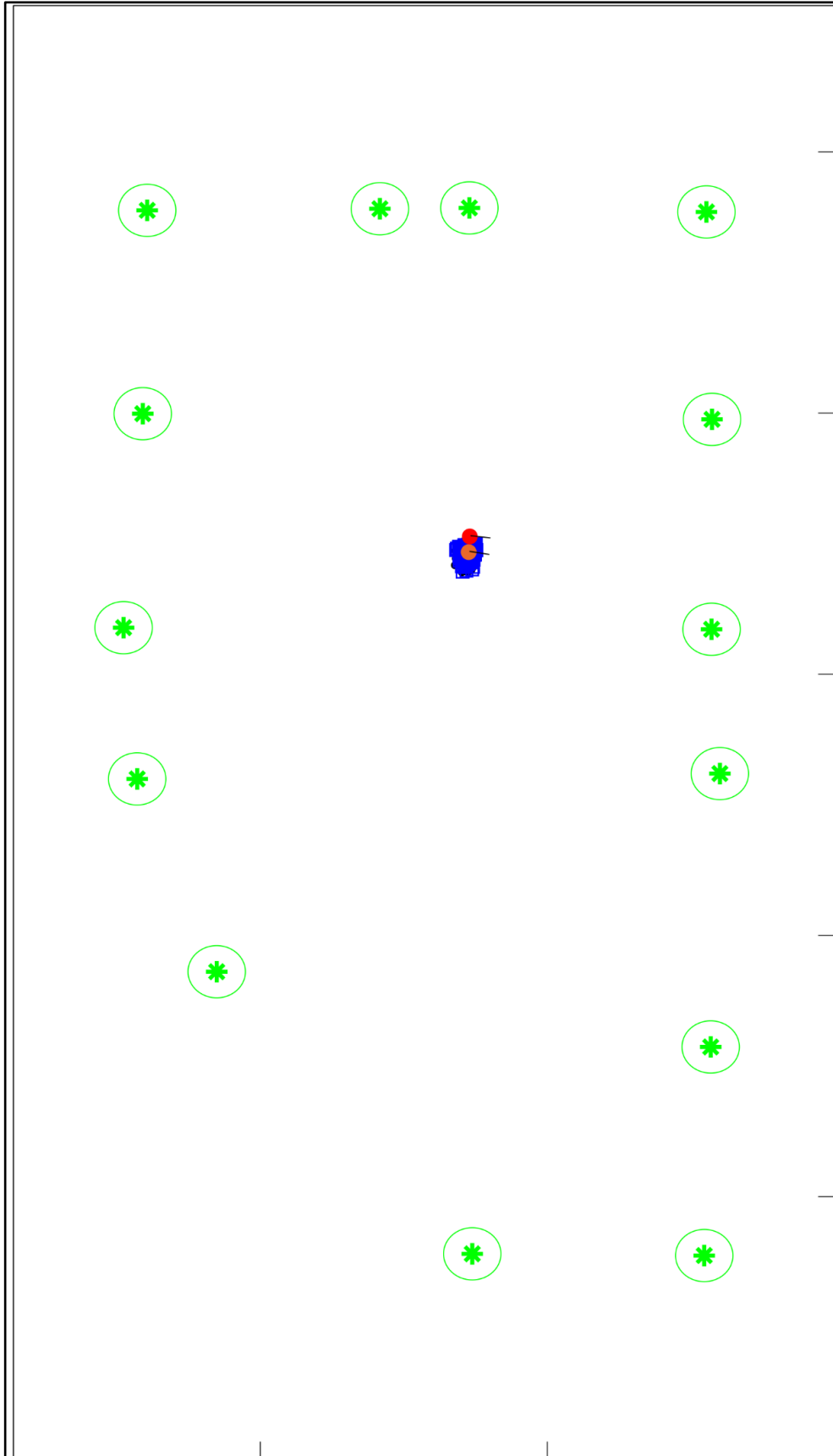
Fonte: (AUTOR)

Figura 78 – Experimento 2 - Resultado do algoritmo após processamento da pose 9.



Fonte: (AUTOR)

Figura 79 – Experimento 2 - Resultado do algoritmo após processamento da pose 15.



Fonte: (AUTOR)

O terceiro experimento foi realizado no mesmo mapa do experimento anterior, porém iniciando a trajetória em outro ponto de simetria, que não o do experimento 2. Da mesma forma que o teste anterior, a Figura 80 ilustra uma foto do mapa utilizado pelo robô e sua primeira pose, a Figura 81 mostra a trajetória feita pelo robô neste experimento e as Figuras 82 a 85 os resultados do algoritmo para as poses 1, 3, 10 e 15 respectivamente.

Os resultados deste experimento confirmam a capacidade do filtro MCL de convergir em simetria do ambiente com mais de uma condição inicial, da mesma maneira que o teste anterior. Observando-se os mesmos *clusters* formados no segundo experimento, o algoritmo se mostrou eficaz em se localizar mesmo em poses iniciais distintas.

### 6.3 Resultados dos experimentos

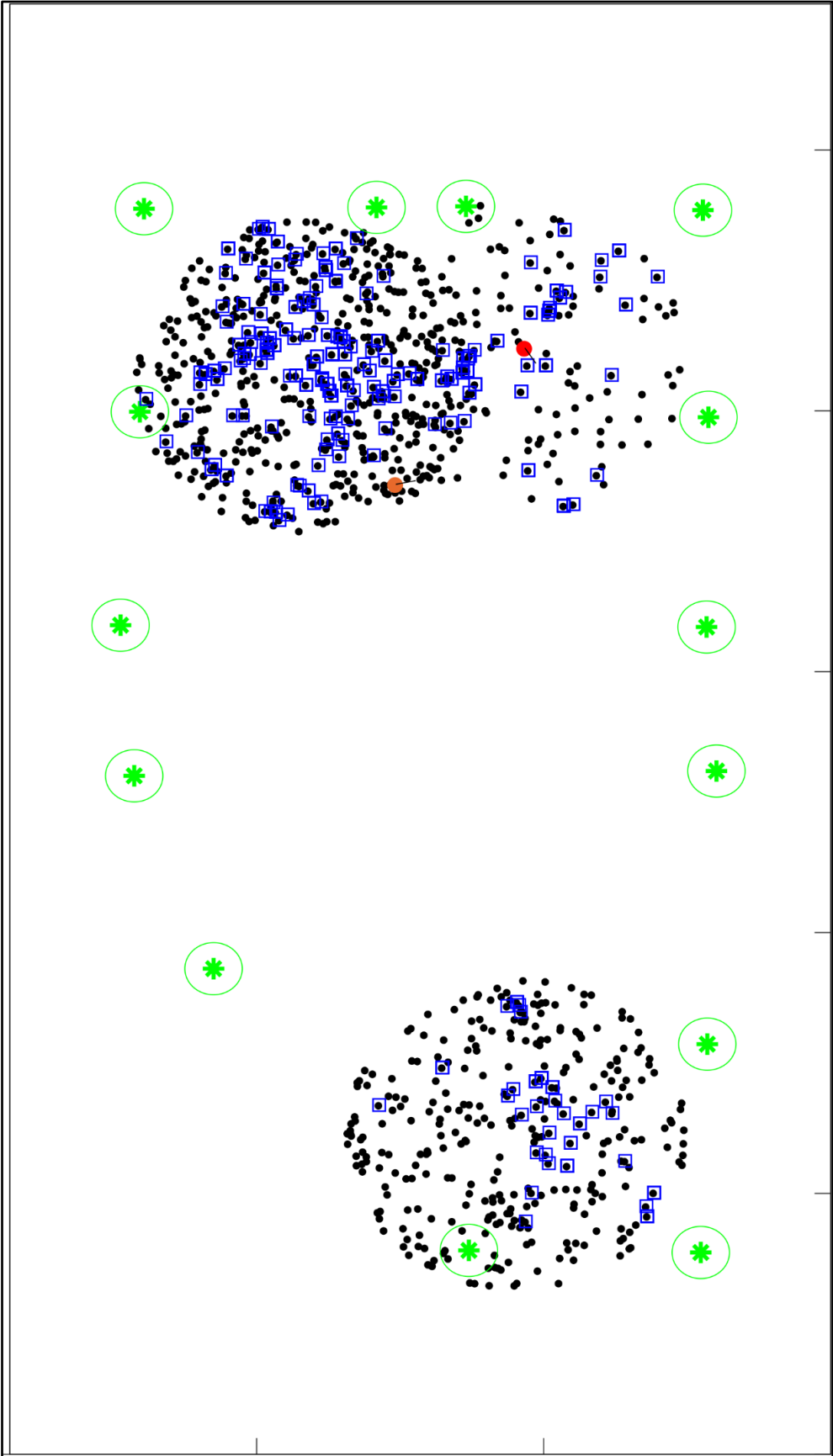
Os resultados dos experimentos foram feitos comparando-se a pose estimada com a real do robô a cada iteração. Os gráficos nas Figuras 86 a 88 mostram essas diferenças em relação tanto às coordenadas 'x' e 'y' quanto as orientações dos três experimentos, os quais são os componentes das poses do robô. Os valores numéricos foram colocados nos Quadros 4 a 9 na forma do erro absoluto e erro relativo.

O filtro convergiu de forma rápida em todos os experimentos, denotando um excelente custo benefício, no que se refere ao número de partículas e iteração onde ocorre a localização global (1000 partículas convergindo por volta de 9 iterações). Os resultados se mostraram satisfatórios, mesmo considerando o caráter baixo custo do projeto. Por se tratar da solução de localização global do robô, esses resultados devem ser avaliados após a situação que indica a convergência do filtro MCL. Nas coordenadas 'x' o erro máximo nos três experimentos foi menor que 1,75 cm, na 'y' foi no máximo de 0,48 cm e em orientação de 7,2°. A proposta de tornar a correspondência conhecida na primeira iteração através da técnica de triangulação contribuiu significativamente para os resultados, onde, caso contrário, haveria a necessidade do aumento de partículas (maior custo computacional e tempo de execução) ou aumento de iterações (maior tempo de execução). Em comparação com trabalhos semelhantes e recentes, como em (MEDEIROS, 2011), cuja utilização do sistema de medição (mesmo tipo de sensores) envolveu apenas uma varredura total de 180° ao redor do robô, a convergência ocorreu em mais de 10 iterações.



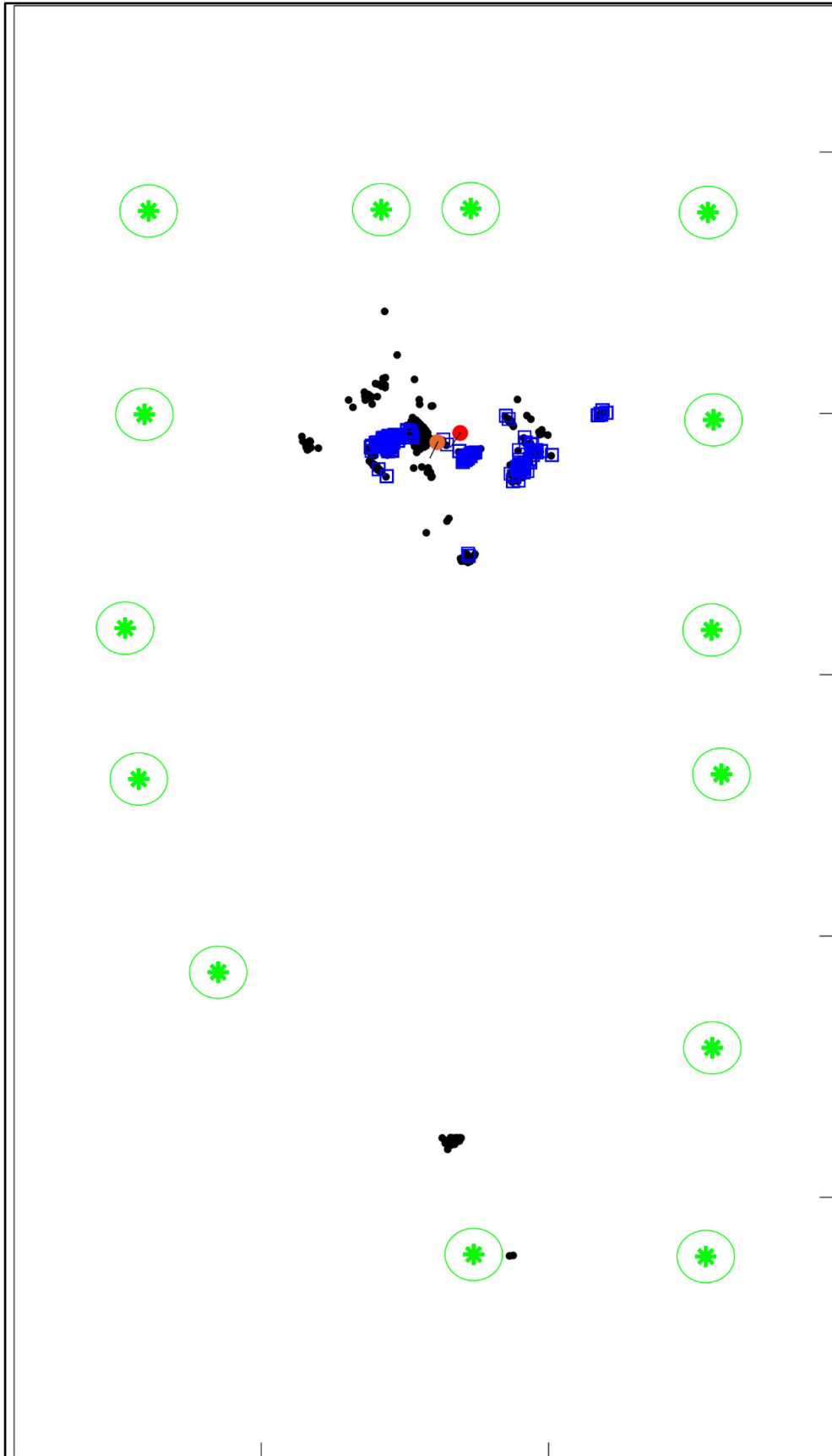


Figura 82 – Experimento 3 - Resultado do algoritmo após processamento da pose 1.



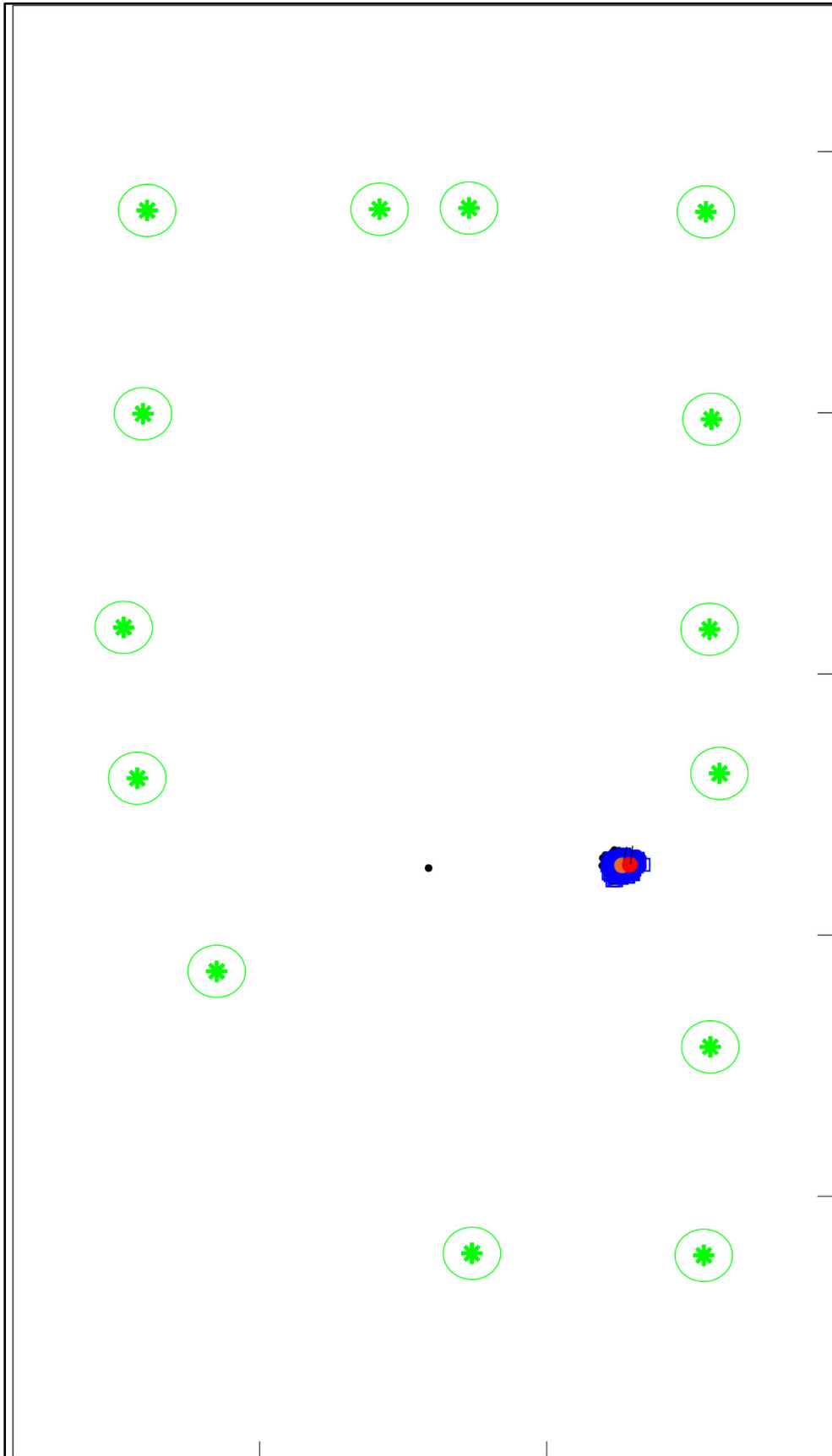
Fonte: (AUTOR)

Figura 83 – Experimento 3 - Resultado do algoritmo após processamento da pose 3.



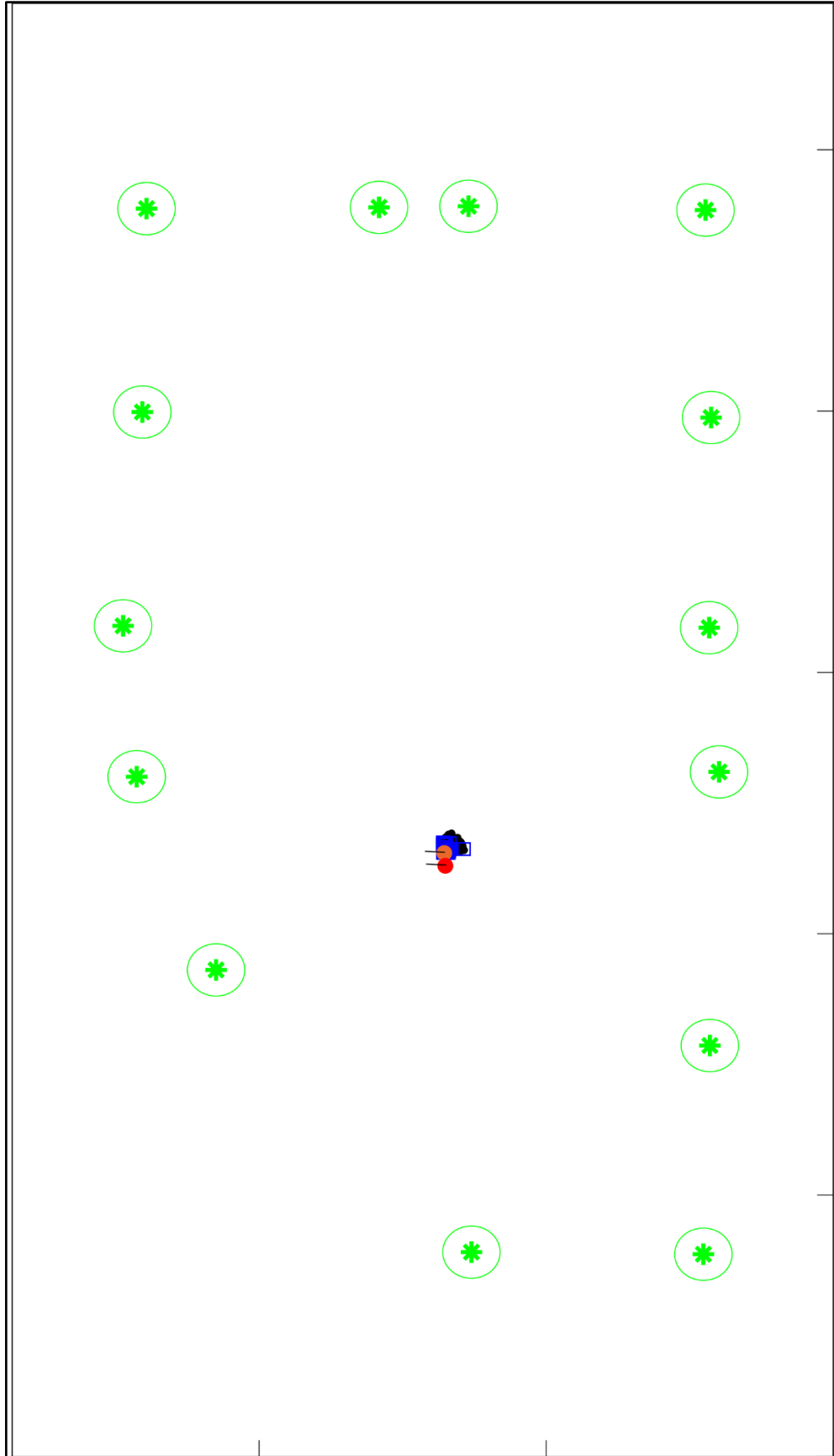
Fonte: (AUTOR)

Figura 84 – Experimento 3 - Resultado do algoritmo após processamento da pose 10.



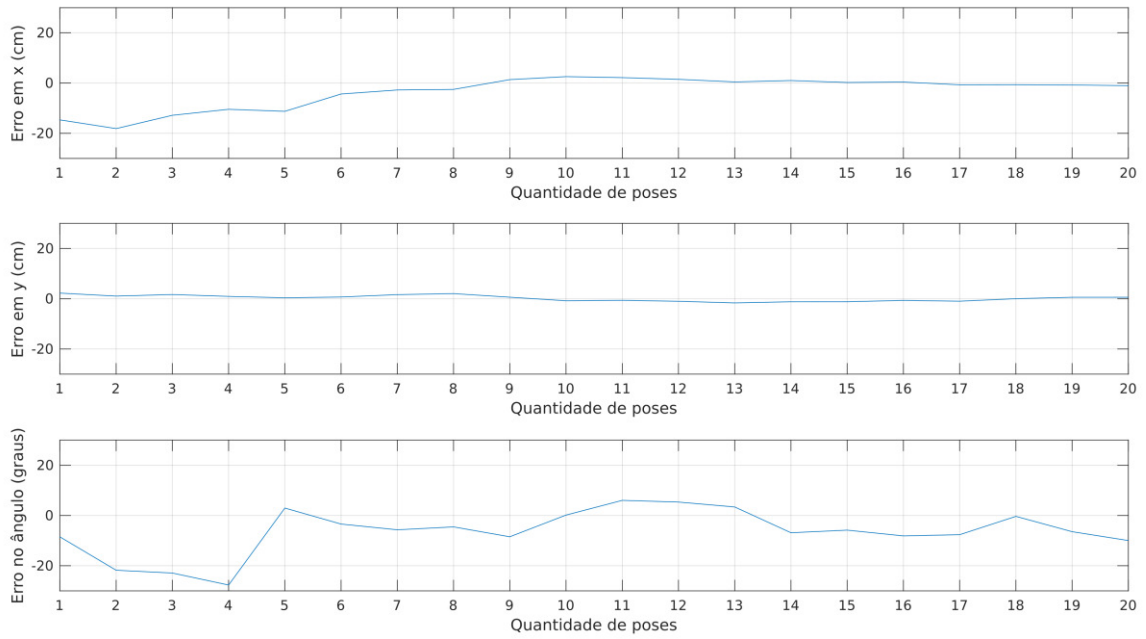
Fonte: (AUTOR)

Figura 85 – Experimento 3 - Resultado do algoritmo após processamento da pose 15.



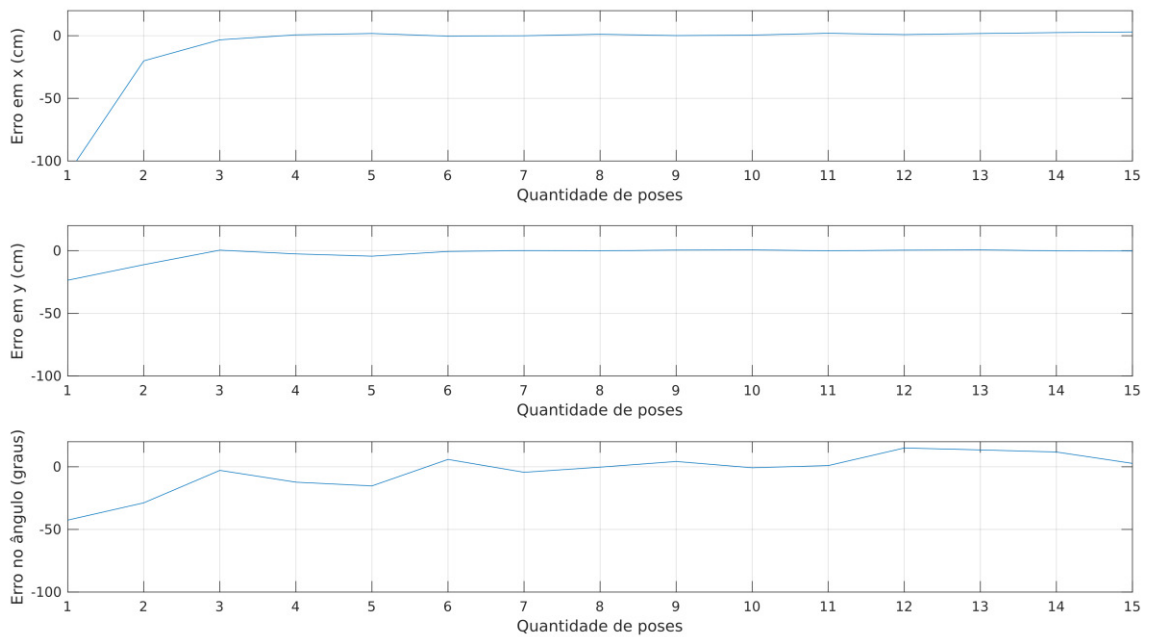
Fonte: (AUTOR)

Figura 86 – Erro entre a pose real e a pose estimada no primeiro experimento.



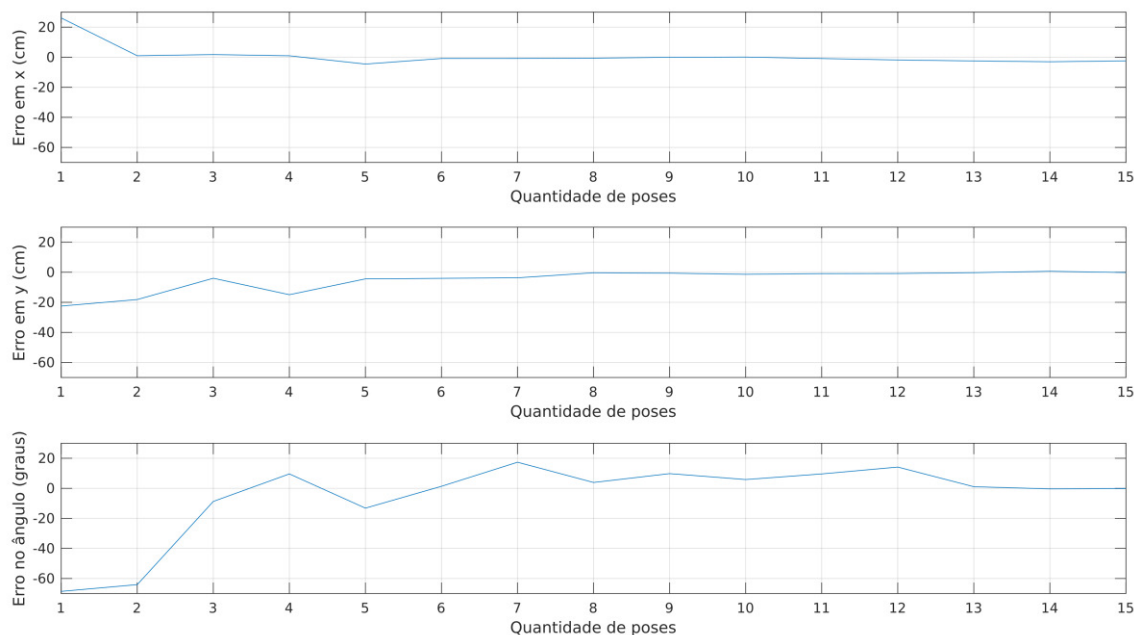
Fonte: (AUTOR)

Figura 87 – Erro entre a pose real e a pose estimada no segundo experimento.



Fonte: (AUTOR)

Figura 88 – Erro entre a pose real e a pose estimada no terceiro experimento.



Fonte: (AUTOR)

Quadro 4 – Valores numéricos de erros absolutos entre a pose real e estimada no primeiro experimento.

	Erro na coordenada x (cm)	Erro na coordenada y (cm)	Erro na orientação (graus)
Pose 1	-7,16965712	3,92296111	-7,18502578
Pose 2	-12,3836293	0,69226477	-8,37731641
Pose 3	-6,1129314	0,99988883	-11,4022172
Pose 4	-4,40339264	0,78570135	-15,1958051
Pose 5	-5,77352949	2,75811026	-1,38067693
Pose 6	-7,2392428	3,44200241	-10,7944064
Pose 7	-5,89711746	2,49637786	-13,9296709
Pose 8	0,47452424	1,51094121	-12,5257232
Pose 9	2,18586053	1,12325089	-8,81684744
Pose 10	3,31425358	-0,35315111	0,50908225
Pose 11	3,31162899	-0,37074977	5,30261933
Pose 12	2,91312527	-0,57974111	4,24076741
Pose 13	1,19759317	-1,11746286	0,58859109
Pose 14	0,80386291	-0,7929281	-7,41684529
Pose 15	-0,12845206	-0,5919347	-6,70482442
Pose 16	-0,47819495	-0,59118641	-8,62748396
Pose 17	-0,81640604	-0,81423957	-7,9774484
Pose 18	-0,59742455	-0,28357053	-0,02038073
Pose 19	-0,25780743	0,10799087	-5,67646015
Pose 20	-0,30046282	0,37820679	-9,07907768

Quadro 5 – Valores numéricos de erros absolutos entre a pose real e estimada no segundo experimento.

	Erro na coordenada x (cm)	Erro na coordenada y (cm)	Erro na orientação (graus)
Pose 1	-110,317495	-23,5292652	-42,5558793
Pose 2	-20,0686048	-11,1371079	-28,7599509
Pose 3	-3,28329324	0,44673187	-2,85685849
Pose 4	0,67217943	-2,53743999	-12,2826522
Pose 5	1,76348428	-4,34784263	-15,2599157
Pose 6	-0,35157232	-0,58244878	5,89056238
Pose 7	-0,02482471	0,11649669	-4,3846975
Pose 8	1,12347914	-0,10103566	-0,28390823
Pose 9	0,099105	0,53250665	4,24780981
Pose 10	0,48157602	0,69931853	-0,76230511
Pose 11	1,93920468	-0,08114408	0,96825834
Pose 12	0,87752494	0,44910959	15,0134521
Pose 13	1,73074008	0,71302631	13,4856791
Pose 14	2,48780925	-0,1196277	11,7939571
Pose 15	2,99512939	-0,18853919	2,72423542

Quadro 6 – Valores numéricos de erros absolutos entre a pose real e estimada no terceiro experimento.

	Erro na coordenada x (cm)	Erro na coordenada y (cm)	Erro na orientação (graus)
Pose 1	26,1325416	-22,4849889	-68,5605155
Pose 2	0,99308389	-18,1460472	-64,0580806
Pose 3	1,76778049	-3,98782114	-8,72191356
Pose 4	0,96707556	-14,9314936	9,59110819
Pose 5	-4,54190553	-4,40405414	-13,179137
Pose 6	-0,8457549	-4,04253738	1,34721933
Pose 7	-0,82844715	-3,6901715	17,3687703
Pose 8	-0,61000694	-0,33756096	3,89148914
Pose 9	-0,04098384	-0,61518714	9,78287892
Pose 10	0,12945659	-1,36881979	5,83959192
Pose 11	-0,87322547	-0,90928414	9,58527823
Pose 12	-1,83365916	-0,83478524	14,1371233
Pose 13	-2,47187091	-0,27100099	1,06949009
Pose 14	-3,00811252	0,64361	-0,36737364
Pose 15	-2,43847903	-0,19106409	-0,13652999



Quadro 7 – Valores numéricos de erros relativos entre a pose real e estimada no primeiro experimento.

	Erro na coordenada x	Erro na coordenada y	Erro na orientação
Pose 1	-0,06367032	0,05334525	0,88747433
Pose 2	-0,10389695	0,00944735	0,90859492
Pose 3	-0,04864652	0,01368756	1,07122885
Pose 4	-0,03329017	0,01079509	0,87217144
Pose 5	-0,04158241	0,03815483	1,92175252
Pose 6	-0,04971846	0,04781755	1,91744321
Pose 7	-0,03868451	0,03475824	1,3323658
Pose 8	0,00298477	0,02112725	1,1927307
Pose 9	0,01320421	0,01578305	0,85978754
Pose 10	0,0192376	-0,00507003	0,00576623
Pose 11	0,01990419	-0,00532815	-0,03020259
Pose 12	0,01820705	-0,00837408	-0,02410211
Pose 13	0,0078181	-0,01625125	-0,00327911
Pose 14	0,00548267	-0,01162208	-0,04231144
Pose 15	-0,00091729	-0,0087082	-0,03800058
Pose 16	-0,00358412	-0,00872466	-0,04941652
Pose 17	-0,00644554	-0,01206617	-0,04553249
Pose 18	-0,0049747	-0,00420174	0,00011548
Pose 19	-0,00227268	0,00160803	-0,03181253
Pose 20	-0,00280604	0,0056597	-0,05199126

Quadro 8 – Valores numéricos de erros relativos entre a pose real e estimada no segundo experimento.

	Erro na coordenada x	Erro na coordenada y	Erro na orientação
Pose 1	-1,57843931	-0,46616553	0,81224423
Pose 2	-0,24994939	-0,22052511	-1,32076818
Pose 3	-0,03808362	0,00740985	-0,18252462
Pose 4	0,00660422	-0,0461135	0,2182618
Pose 5	0,01502024	-0,07712764	0,62616615
Pose 6	-0,00264166	-0,0114948	-0,50194223
Pose 7	-0,000166	0,00223104	10,5548043
Pose 8	0,00657552	-0,00190101	-0,00513785
Pose 9	0,00057022	0,00716619	0,05172363
Pose 10	0,00272091	0,00734128	-0,0044852
Pose 11	0,01094539	-0,00090507	-0,0100418
Pose 12	0,00496381	0,00541235	-0,17709864
Pose 13	0,00979722	0,00933539	-0,15501553
Pose 14	0,01410386	-0,00171281	-0,13311225
Pose 15	0,01696775	-0,00297516	-0,02809237

Quadro 9 – Valores numéricos de erros relativos entre a pose real e estimada no terceiro experimento.

	Erro na coordenada x	Erro na coordenada y	Erro na orientação
Pose 1	0,12323827	-0,42208396	0,46622684
Pose 2	0,00494064	-0,32962059	-0,4237853
Pose 3	0,00900128	-0,06112082	-0,05917194
Pose 4	0,00534036	-0,2335474	-0,06797601
Pose 5	-0,02744001	-0,0643218	0,07348441
Pose 6	-0,00566906	-0,06045207	-0,00763371
Pose 7	-0,00616416	-0,06250674	-0,12574357
Pose 8	-0,0052758	-0,00693623	-0,0397107
Pose 9	-0,00035641	-0,01461859	-0,10607589
Pose 10	0,00113928	-0,03868558	-1,0026195
Pose 11	-0,00768604	-0,02212867	0,10701836
Pose 12	-0,01615624	-0,01760306	0,14906432
Pose 13	-0,02186286	-0,00502402	0,01288279
Pose 14	-0,02662226	0,01060154	-0,00440959
Pose 15	-0,02155152	-0,00283325	-0,0015712

## 7. CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho de monografia abordou uma solução probabilística para o problema de localização em ambiente (tablado de 278 x 143 cm<sup>2</sup>) interno e estático, cujo mapa foi constituído por vários cilindros de raio 5 cm, fazendo-se uso do robô de baixo custo, denominado MILO. A interação do robô com o ambiente foi feita através de um subsistema de medição formado por dois pares de sensores (infravermelho e sonar) defasados de 180°. Cada processo de aquisição de medidas (giro da torre em um total de 180°) fez as leituras das medidas de distâncias pelos quatro sensores a cada 4°, as quais ao final foram fundidas aos pares (frente e costa) para gerar um único vetor de medidas de varredura em 360°.

Na solução proposta foi desenvolvido e testado o filtro probabilístico MCL em três experimentos de localização no ambiente. Houve o acréscimo da técnica de triangulação de objetos medidos antes de iniciar o filtro, de maneira a diminuir a área de amostragem das partículas, que ficou restrita a um ou mais clusters no tablado. Os modelos de movimento e medição foram implementados empiricamente, com o levantamento de suas incertezas de translação e rotação (movimento) e de aquisição de medidas (medição). Este último modelo considerou o processo de fusão sensorial, o qual forneceu os possíveis obstáculos detectados em cada varredura pelo subsistema de medição a partir das medidas brutas obtidas por cada sensor dos dois pares.

Um sistema de câmera foi montado acima do tablado para aquisição de imagens do ambiente, de forma a poder obter automaticamente a posição dos obstáculos e as poses do robô no ambiente, todas essas informações com projeção ao solo e relativas a um sistema de coordenadas global. Esses dados gerados a partir das imagens permitiram um desenvolvimento mais rápido da pesquisa, evitando ter que aferir as coordenadas dos obstáculos que compõem o mapa e as poses percorridas pelo robô MILO durante os experimentos de localização. Também teve papel importante na definição empírica dos modelos probabilísticos de movimento e medição utilizados pelo filtro MCL.

Foram conduzidos três experimentos de localização, um para avaliar a rapidez de convergência do filtro MCL em iterações e outros dois para validar a convergência em situações de simetria do mapa na definição do ambiente. Os experimentos resultaram na convergência desejada com erros absolutos não superiores a 0,43%, 0,21% e 3,4% para as coordenadas 'x' e 'y' e orientação que formam as poses estimadas do robô, respectivamente, em relação às dimensões do tablado e ao semicírculo (180°).

O fato de se utilizar a câmera para capturar a pose real, inseriu ruído nesses resultados, no máximo de 1 cm para as duas dimensões do tablado, fato verificado empiricamente com o uso de uma trena. No entanto ele não foi considerado nos resultados por ser constante. Esses valores são considerados aceitáveis, levando-se em conta as dimensões do robô e a precisão dos sensores utilizados. Dessa forma, o objetivo do trabalho implementado foi alcançado e os resultados dos experimentos se mostraram satisfatórios, considerando que o robô se mostra eficiente em se localizar mesmo em ambientes onde existam simetrias.

Como trabalhos futuros a serem desenvolvidos a partir do atual estágio alcançado neste trabalho de monografia podem-se citar:

- a) Projeto, implementação e teste de um filtro para endereçar o problema SLAM no mesmo ambiente, usando os mesmos sensores ou um sensor externo ao robô MILO, como a própria câmera utilizada neste trabalho;
- b) A implantação do ROS (*Robot Operating System*) no ambiente operacional embarcado no robô, de maneira a abstrair no *software* o acesso aos recursos de *hardware* do robô;
- c) O desenvolvimento do mesmo código, incluindo o filtro, de forma embarcada no RPi3, onde a execução não dependeria da conexão com o PC;
- d) Projeto de adaptação do algoritmo e sensores do robô para operar com obstáculos dinâmicos;
- e) A implantação de um sistema de planejamento de trajetória, permitindo que o robô navegue no ambiente entre duas poses, partindo da que se encontra até outra fornecida pelo operador; e
- f) A implantação de um sistema de manipulação de objetos (peso mínimo) que permita ao robô navegar pelo ambiente, com o objetivo de realizar o transporte de carga.

## REFERÊNCIAS

- BUONOCORE, L. NASCIMENTO JR., C. L., NETO, A. A. **Sistema de baixo custo para medição de distância em duas dimensões usando câmera IP sem fio**. XVIII Congresso Brasileiro de Automática, Bonito, 2010, pp. 2239 – 2246.
- BUONOCORE, L. **SLAM em Ambientes Internos Usando Robô de Baixo Custo**. Tese de Doutorado. Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos, Brasil, 2013, 275 p.
- BUONOCORE, L.; DOS SANTOS, S. R. B.; NASCIMENTO JR. C. L.; ALMEIDA NETO, A. **FastSLAM filter implementation for indoor autonomous robot**. IEEE *Intelligent Vehicles Symposium (IV)*, Gothenburg, Sweden, 2016, pp. 484 – 489.
- DELLAERT, F. et al. **Monte carlo localization for mobile robots**. Proceedings of 1999 IEEE International Conference on Robotics and Automation, 1999, pp. 1322–1328.
- DISCANT, A.; ROGOZAN, A.; RUSU, C.; BENSRAHAI, A. **Sensor for Obstacle Detection – A Survey**. 30th International Spring Seminar on Electronics, 2007.
- MAKARENKO, A. A. et al. **An experiment in integrated exploration**. Switzerland: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2002.
- MEDEIROS, I. P. **Navegação do robô Trekker usando marcos artificiais e sensores de baixo custo**. Dissertação de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos, 2011. 96 p.
- PINTO JR., W. L. **Robô controlado por gestos com *feedback* tátil**. Monografia do Curso de Graduação em Engenharia Elétrica, Universidade Federal do Maranhão, 2017, 66 p.
- RIBEIRO, D. V. **ROSELI: Robô Seguidor de Linha para Mapeamento de Ambientes Internos**. Monografia do Curso de Graduação em Engenharia Elétrica, Universidade Federal do Maranhão, 2017, 94 p.
- SCARAMUZZA, D. et al. **Perception: Sensors**. Autonomous Mobile Robots Class: University of Zurich, 2015.
- SIEGWART, R.; NOURBAKHSI, I. R. **Introduction to autonomous mobile robots**. Cambridge: MIT Press, 2004, 321 p.
- STACHNISS, C. **Exploration and mapping with mobile robots**. Freiburg: Tese PhD - University of Freiburg, 2006, 244 p.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic robotics**. Cambridge: MIT Press, 2006, 647 p.

VISSER, A. **Sensor data fusion**. ORGANIZATION AND DESIGN OF AUTONOMOUS SYSTEMS. University of Amsterdam, 1999.