

Universidade Federal de Juiz de Fora
Faculdade de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Wolmar Araujo Neto

**Localização de Robôs Móveis Utilizando o Novo Algoritmo de Morcegos
Baseado em Líderes**

Juiz de Fora

2019

Wolmar Araujo Neto

**Localização de Robôs Móveis Utilizando o Novo Algoritmo de Morcegos
Baseado em Líderes**

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Juiz de Fora, na área de concentração em Sistemas de Energia, como requisito para obtenção do título de Doutor em Engenharia Elétrica.

Orientador: André Luís Marques Marcato

Coorientador: Ivo Chaves da Silva Junior

Juiz de Fora

2019

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Araujo Neto, Wolmar.

Localização de Robôs Móveis Utilizando o Novo Algoritmo de Morcegos
Baseado em Líderes / Wolmar Araujo Neto. – 2019.

77 f. : il.

Orientador: André Luís Marques Marcato

Coorientador: Ivo Chaves da Silva Junior

Tese (Doutorado) – Universidade Federal de Juiz de Fora, Faculdade de
Engenharia. Programa de Pós-Graduação em Engenharia Elétrica, 2019.

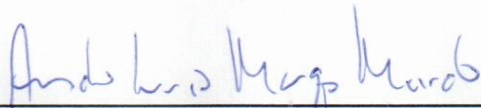
1. Robôs Móveis. 2. Algoritmo de Morcegos. 3. Localização. I. Luís
Marsques Marcato, André, orient. II. Chaves da Silva Junior, Ivo, coorient.
III. Título.

Wolmar Araujo Neto

Localização de Robôs Móveis Utilizando o Novo Algoritmo de Morcegos Baseado em Líderes

Tese apresentada ao Programa de Pós-graduação em Engenharia Elétrica, área de concentração: Sistemas de Energia, da Universidade Federal de Juiz de Fora, como requisito final para obtenção do grau de Doutor em Engenharia Elétrica.

Tese aprovada. Juiz de Fora, 8 de Abril de 2019:



André Luís Marques Marcato, D.Sc.
Orientador - UFJF



Ivo Chaves da Silva Junior, D.Sc.
Co-orientador - UFJF



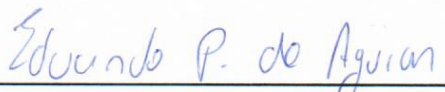
Edmarcio Antonio Belati, D.Sc.
UFABC



Fernando Luiz Cyrino Oliveira, D.Sc.
PUC - Rio



Leonardo Willer de Oliveira, D.Sc.
UFJF



Eduardo Pestana de Aguiar, D.Sc.
UFJF

AGRADECIMENTOS

Agradeço, primeiramente a Deus. Aos meus pais pelo amor incondicional, aos meus amigos por sempre acreditaram em mim e nunca me deixarem desistir dos meus sonhos, e a CAPES pelo auxílio na pesquisa durante o tempo de doutorado. Por fim, meu muito obrigado, aos amigos e amigas do GRIn, que foram minha família durante esses anos em Juiz de Fora.

“O homem científico não almeja resultados imediatos. Ele não espera que suas ideias mais avançadas sejam rapidamente retomadas. Seu trabalho é como o de um agricultor para o futuro. Seu dever é estabelecer bases para aqueles que estão por vir e apontar o caminho a ser seguido.”

Nikola Tesla (1856–1943)

RESUMO

Um problema estocástico complexo de importância essencial em aplicações de robôs móveis é a localização do robô. Esta pode ser tratada como um problema de otimização, devido à sua característica de apresentar diferentes possíveis soluções (mínimos locais). O método proposto nesse trabalho é baseado no Algoritmo de Morcegos, o qual apresentou melhores resultados para o problema de localização global e sequestro de robô quando comparado com outros métodos da literatura. Apresentou-se resultados favoráveis tanto para localização global, com uma celeridade tão hábil quanto os algoritmos comparados, como resultados significativos em uma situação de sequestro de robô. A nome proposto para a modificação do algoritmo clássico de morcegos foi Algoritmo de Morcegos Baseado em Líderes (*Leader-Based Bat Algorithm - LBBA*), em virtude de ter a característica de simular ações de líderes de grupo, os quais influenciam outros menos qualificados. A consequência da utilização de mais de um líder implica em uma busca, da melhor posição, mais diversificada no mapa, lidando melhor com cenários que apresentem muitas ambiguidades no decorrer da localização. Foram realizados testes em diferentes cenários (além do teste em um mapa real). O algoritmo bioinspirado de morcegos na sua forma clássica (BA) foi comparado com outros algoritmos (otimização de enxame de partículas - PSO e Filtro de Partículas - FP) e obteve bons resultados. Estes inspiraram a proposta de uma variante no algoritmo de morcegos capaz de lidar com situações mais complexas de localização onde o BA apresentava dificuldades, o LBBA obteve resultados notáveis quando comparado com Localização de Monte Carlo Adaptativa (AMCL).

Palavras-chave: Robôs Móveis. Algoritmo de Morcegos. Localização. Localização de Monte Carlo Adaptativa.

ABSTRACT

A complex stochastic problem of essential importance in mobile robot applications is the location of the robot. This can be treated as an optimization problem, due to its characteristic of presenting different possible solutions (local minimums). The method proposed in this work is based on the Bats Algorithm, which presented better results for the problem of global localization and robot sequestration when compared to other methods in the literature. We presented favorable results for both global localization, with a speed as skillful as the compared algorithms, as significant results in a robot sequestration situation. The proposed name for the modification of the classic bats algorithm was Leader-Based Bat Algorithm (LBBA), because it has the characteristic of simulating actions of group leaders, which influence others less skilled. The consequence of using more than one leader implies a search, the best position, the most diversified on the map, dealing better with scenarios that present many ambiguities during the localization. Tests were performed in different scenarios (besides the test on a real map). The bio - inspired algorithm of bats in its classical form (BA) was compared with other algorithms (particle swarm optimization - PSO and Particle Filter - FP) and obtained good results. These inspired the proposal of a variant in the bats algorithm capable of dealing with more complex situations of location where the BA presented difficulties, the LBBA obtained remarkable results when compared to Monte Carlo Adaptive Location (AMCL).

Key-words: AMCL. Bat Algorithm. Localization. Mobile Robots.

LISTA DE ILUSTRAÇÕES

Figura 1 – Exemplos de aplicações de robôs (a) Braço mecânico industrial. (b) Robô cirurgião. (c) humanóide NAO. (d) <i>Rover Cuerriosity</i> . (e) Pepper.	14
Figura 2 – Algoritmos Bioinspirados.	16
Figura 3 – Acúmulo de erro de odometria durante a realização de uma missão.	21
Figura 4 – SICK LMS-200.	27
Figura 5 – Modelo esquemático das medidas de distância.	28
Figura 6 – Mapa Occupancy Grid PPEE-UFJF.	28
Figura 7 – Mapa de Retas.	29
Figura 8 – Mapa de nuvem de pontos	29
Figura 9 – Modelo incremental da movimentação do robô.	30
Figura 10 – Eco-localização de morcegos.	35
Figura 11 – Robot Operating System (ROS).	37
Figura 12 – Demonstrativo de localização no mapa Occupancy Grid.	38
Figura 13 – Líderes influenciando na busca do ótimo global	39
Figura 14 – Representação Aproximada da Distribuição Normal	40
Figura 15 – Determinando os melhores morcegos da colônia - Líderes	44
Figura 16 – Cálculo da distância de cada morcego em relação aos Líderes	44
Figura 17 – Região de Influência dos Líderes	45
Figura 18 – Líderes influenciando na busca da otimização	45
Figura 19 – Ótimo Global é encontrado pelo melhor morcego da colônia	45
Figura 20 – Função Rastrigin.	48
Figura 21 – Função Eggcrate.	48
Figura 22 – Função Ackley.	49
Figura 23 – Função Rastrigin: Número de iterações \times Número de partículas	50
Figura 24 – Resultado da Função Rastrigin para o intervalo de 0 a 1000 partículas.	50
Figura 25 – Função Eggcrate: : Número de iterações \times Número de partículas	51
Figura 26 – Resultado da Função Eggcrate para o intervalo de 0 a 1000 partículas.	51
Figura 27 – Função Ackley: : Número de iterações \times Número de partículas	52
Figura 28 – Resultado da Função Ackley para o intervalo de 0 a 1000 partículas.	52
Figura 29 – Mapa Circular.	53
Figura 30 – Partículas espalhadas uniformemente pelo mapa.	54
Figura 31 – Localização P3DX realizada pelos diferentes algoritmos de otimização e testada no primeiro ambiente.	55
Figura 32 – Experimento no segundo ambiente usando <i>Leader-based Bats Algorithm</i> (LBBA) no mapa da grade. Os círculos vermelhos representam as medidas do sensor laser e a linha azul são as medições simuladas. Os pontos azuis representam os morcegos.	56
Figura 33 – Primeiro ambiente.	57

Figura 34 – Segundo ambiente.	57
Figura 35 – Usuário informa ao programa quando os <i>waypoints</i> devem ser visitados pelo robô durante a realização da missão.	58
Figura 36 – Simulação do mapa simples com os melhores morcegos representados por pontos azuis e leituras de laser representadas por círculos vermelhos. A localização foi realizada em um mapa disponível no MATLAB®.	59
Figura 37 – Simulação do mapa complexo com os melhores morcegos e as melhores leituras a laser. Localização realizada em um mapa disponível no MATLAB®.	59
Figura 38 – Comportamento do algoritmo LBBA no problema de sequestro do robô.	62
Figura 39 – Evolução da posição estimada para diferentes números de partículas na simulação de sequestro do robô.	63
Figura 40 – Evolução do erro para diferentes números de partículas na simulação de sequestro do robô.	63
Figura 41 – Comportamento do robô no experimento de sequestro.	64
Figura 42 – Evolução do erro de acordo com o passar do tempo.	64
Figura 43 – Comportamento do P3DX no cenário de sequestro.	65
Figura 44 – Evolução do erro do P3DX com o passar das iterações no cenário de sequestro.	65
Figura 45 – Marcadores - <i>Landmarks</i>	67
Figura 46 – <i>Landmarks</i> ao longo do mapa para serem utilizados na detecção.	69
Figura 47 – Protótipo de um sensor Bússola utilizado em testes.	69
Figura 48 – Localização Global utilizando o sensor bussola proposto durante a simulação.	70
Figura 49 – P3DX utilizado para testes, contendo o protótipo do sensor bussola.	70

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
UFJF	Universidade Federal de Juiz de Fora
IBGE	Instituto Brasileiro de Geografia e Estatística
DARPA	<i>Defense Advanced Research Projects Agency</i>
PDF	Função Densidade de Probabilidade
ICP	<i>Interactive Closest Point</i>
FP	Filtro de Partículas
MCL	<i>Monte Carlo Localization</i>
AMCL	<i>Adaptive Monte Carlo Localization</i>
SLAM	<i>Simultaneous Localization and Mapping</i>
vSLAM	Mapeamento Simultâneo Visual
FastSLAM	<i>Factored Solution to the Simultaneous Localization and Mapping</i>
EKF	Filtro de Kalman Estendido
KLD	Distância de <i>Kullback-Leiber</i>
MARS	<i>Mobile Autonomous Robot Software</i>
MVO	<i>Monocular Visual Odometry</i>
GOHB	Grupo de Otimização Heurística e Bioinspirada
GRIn	Grupo de Robótica Inteligente
PSO	<i>Particle Swarm Optimization</i>
BA	<i>Bat Algorithm</i>
ROS	<i>Robot Operating System</i>
LBBA	<i>Leader-based Bats Algorithm</i>
BatSLAM	<i>Bat-Simultaneous Localization And Mapping</i>

LISTA DE SÍMBOLOS

$\sigma_x, \sigma_y, \sigma_\theta$	Desvios padrões de x, y e θ
N	Número de partículas
ΔS	Deslocamento espacial das partículas
$\Delta\theta$	Deslocamento angular das partículas
x_i	Posição da partícula $i = 1, 2, \dots, N$
ΔX	Deslocamento espacial das partículas na direção x
ΔY	Deslocamento espacial das partículas na direção y
w_i	Peso da partícula $i = 1, 2, \dots, N$
v_r	Velocidade da roda direita
v_l	Velocidade da roda esquerda
Δt	Período de amostragem
L_p^i	Leitura do <i>laserscan</i> da partícula i
L_r	Leitura do <i>laserscan</i> do robô
L_i	Verossimilhança da partícula i
σ	Matriz de covariância do sensor
w_{slow}	Média de longo prazo
w_{fast}	Média de curto prazo
N_{kld}	Número máximo de partículas
x_t^i	Posição da partícula i no instante t
w_t^i	Peso da partícula i no instante t
X_t	Conjunto de partículas no instante t
N_t	Número de partículas no instante t
b	<i>Bins</i> de um histograma
G	Histograma
N_b	Número de <i>bins</i>

N_r	Número de partículas aleatoriamente adicionadas
\bar{X}_t	Conjunto de partículas amostradas de X_t
u_t	Sinal de entrada do robô no instante t
F	Medidas do sensor real
MAP	Mapa do ambiente
v_i	Velocidade do morcego i
f_i	Frequência de pulso i
f_{min}	Frequência de pulso mínima
f_{max}	Frequência de pulso máxima
ξ_1, ξ_2, ξ_3	Números randômicos uniformes normalizados
ϵ	Número randômico normal
$\alpha, \beta, \gamma, r_0$	Fatores de escala
A	Intensidade dos pulsos
$f(x_i)$	Peso da partícula i
\hat{x}_i	Posição estimada da partícula i
x^*	Posição da melhor partícula atual
L	Número de líderes
x_L^*	Posição dos líderes
M	Conjunto de indivíduos
r_i	Frequência do pulso emitido pela partícula i
$IC_{99\%}$	Intervalo de confiança de 99%
\bar{E}	Média dos erros de posição

SUMÁRIO

1	CONTEXTUALIZAÇÃO	13
1.1	JUSTIFICATIVA	15
1.2	OBJETIVO	18
1.3	ORGANIZAÇÃO DO TRABALHO	18
1.4	CONTRIBUIÇÃO DESTE TRABALHO	18
1.5	PUBLICAÇÕES DECORRENTES DESTE TRABALHO	19
2	REVISÃO BIBLIOGRÁFICA	20
2.1	CONCLUSÃO	26
3	FUNDAMENTOS TEÓRICOS	27
3.1	INTRODUÇÃO	27
3.2	LOCALIZAÇÃO DE MONTE CARLO ADAPTATIVA - AMCL	29
3.3	ALGORITMO DE MORCEGOS - <i>Bat Algorithm</i>	34
3.4	ROS MATLAB®	36
4	ALGORITMO PROPOSTO - LBBA	39
4.1	ALGORITMO DE MORCEGO BASEADO EM LÍDERES	39
4.2	CONCLUSÃO	46
5	RESULTADOS	47
5.1	Funções de Teste Multimodais e Análise Estatística	47
5.2	Localização de Robôs Móveis utilizando o LBBA	53
5.2.1	ALGORITMOS SEM VARIAÇÃO DE QUANTIDADE DE PARTÍCULAS	54
5.2.2	ALGORITMOS COM VARIAÇÃO DE QUANTIDADE DE PARTÍCULAS	58
5.2.3	SEQUESTRO DO ROBÔ	60
5.3	Conclusão	65
6	TRABALHOS FUTUROS	67
7	CONCLUSÃO	71
	REFERÊNCIAS	73

1 CONTEXTUALIZAÇÃO

A tecnologia é o alicerce para as mudanças do mundo desde a primeira revolução industrial, com máquinas a vapor e o tear, passando pela segunda com a energia elétrica e a produção seriada, até terceira revolução, com a implementação do controlador lógico programável (CLP) na indústria [1]. A robótica participa cada vez mais do cotidiano das pessoas, exercendo em algumas situações um papel fundamental para a realização de tarefas que antes colocariam a integridade do ser humano em risco nas indústrias. Em demais situações, a proposta é melhorar a qualidade de vida dos humanos por meio de robôs domésticos, além de já estar presente na maioria das linhas de produção das indústrias. A evolução de *hardware* e *software* possibilitou redução de custos e o aumento da capacidade de processamento, contribuindo consideravelmente no avanço da robótica e possibilitando o desenvolvimento de equipamentos mais robustos e precisos.

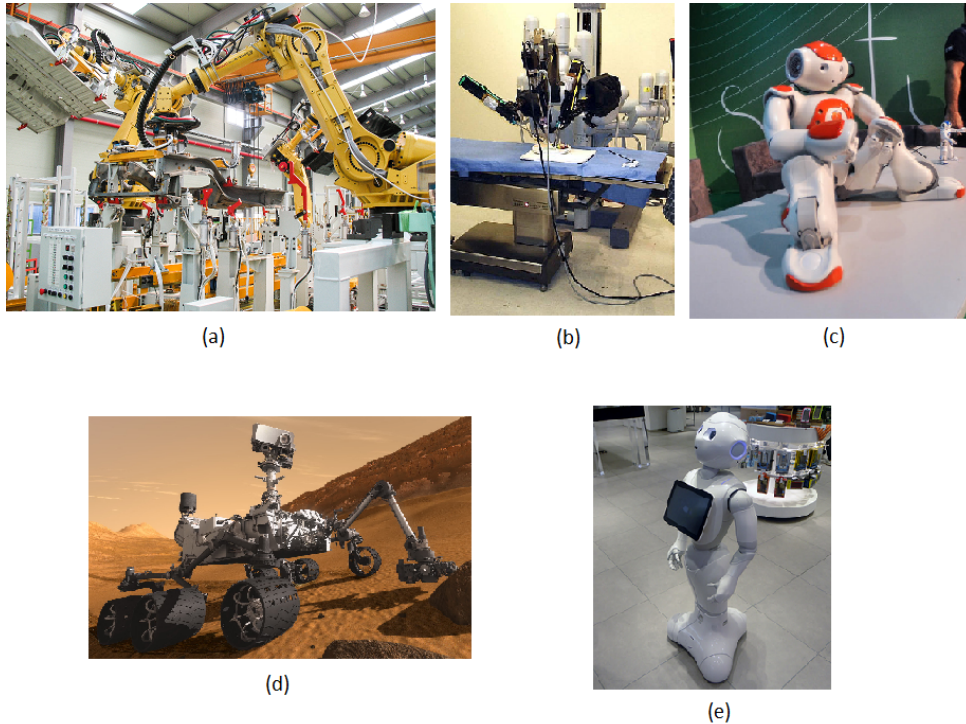
Os robôs podem ser utilizados em diversos tipos de ambientes, como por exemplo, nos espaços domésticos aspirando o pó [2], limpando piscinas e até mesmo como acompanhantes de idosos [3]; escolares através dos kits Lego e NAOs [4]; mesas de cirurgia; chão de fábrica da indústria (Figura 1), com destaque para o setor automobilístico e farmacêutico [5]. O aumento da utilização de robôs na sociedade levou alguns países a investirem de forma mais incisiva em pesquisa e desenvolvimento na área da robótica. É possível perceber a importância da robótica no mundo quando analisa-se os investimentos na área, no ano de 2009 a União Europeia planejava investir 536 milhões de euros e a Coreia do Sul se programou para aplicar aproximadamente 1 bilhão de dólares em investimento de tecnologias de robótica na educação [6], esse valores se elevam de forma considerável no ano de 2019, onde a robótica é a indústria que mais cresce no mundo, apresentando uma taxa de crescimento anual de 17%, os investimentos mundiais passaram de US \$71 bilhões em 2015 para US \$ 135,4 bilhões em 2019, segundo a *International Data Corporation* [7].

Guia Semestral Global de Investimento em Robótica (IDC) divulgou no ano de 2018 os resultados do estudo que mede a implementação de robôs em diferentes tipos de indústrias e modos de uso [8], o relatório aponta um investimento de US\$ 1,040 bilhão nesse segmento robótica na América Latina em 2018. A previsão é ter um crescimento de até 21% e investido mais US\$ 1,266 bilhão até o final do ano. Sendo a projeção desses é de 73% de autômatos industriais. [9]

O desenvolvimento de sensores mais precisos, como por exemplo, *laser scanners* e câmeras de vídeos, com preços inferiores viabilizam a possibilidade de produzir robôs mais sofisticados, aptos a lidar com ambientes mais complexos e desestruturados. Os Projetos de código aberto como *Robot Operating System* (ROS) [10] fornecem bibliotecas e ferramentas para ajudar desenvolvedores na integração de dispositivos e comunicação de diversos sensores e atuadores, facilitando o desenvolvimento de novas pesquisas e algoritmos

responsáveis pelo controle, processamento de imagens, reconhecimento de voz, e etc. Essa plataforma permite o compartilhamento de *software*, *drivers* e algoritmos de uma grande e ativa comunidade acadêmica, impulsionando grupos de pesquisa ao redor do mundo.

Figura 1 – Exemplos de aplicações de robôs (a) Braço mecânico industrial. (b) Robô cirurgião. (c) humanóide NAO. (d) *Rover Curiosity*. (e) Pepper.



Fonte: Google Imagens - Reutilização não comercial com modificação

Dentro das pesquisas em robótica móvel, a localização é um processo indispensável para a realização de missões de robôs móveis autônomos [11]. Existem diversos tipos de erros oriundos de sensores que são integrados ao longo do tempo que podem causar degeneração excessiva da estimativa pose do robô (localização exata do robô no mapa) durante a navegação e, portanto, o robô não será capaz de completar sua tarefa. Para superar esta dificuldade, vários métodos estocásticos foram desenvolvidos para a mitigação desses erros, como por exemplo os métodos estocásticos unimodais, como as variantes não-lineares do Filtro de Bayes [12] e do Filtro de Kalman [13]. Estas são técnicas amplamente utilizadas em todas as áreas do conhecimento porque são estatisticamente otimizadas, no sentido do erro quadrático, e são computacionalmente rápidas. No entanto, a robótica móvel impõe dificuldades a esses métodos porque as técnicas unimodais exigem um satisfatório estado inicial, que nem sempre está disponível [14].

Em contrapartida, as técnicas multimodais podem usar várias hipóteses para a posição do robô ao mesmo tempo. Estas técnicas multimodais têm um alto custo computacional quando comparadas com os métodos unimodais, o que pode causar dificuldades na implementação de algoritmos *on-line*. Não obstante, uma técnica multimodal é capaz de

se recuperar de singularidades e outros problemas de divergência para resolver problemas de localização global, onde métodos unimodais falham [15]. Os Filtros de Partículas são processos contínuos e rápidos, entretanto, suas limitações ocorrem quando o sistema não pode adquirir variabilidade dos *landmarks* (pontos de referência) no ambiente, durante a simulação dos estados do sistema pelas partículas [14].

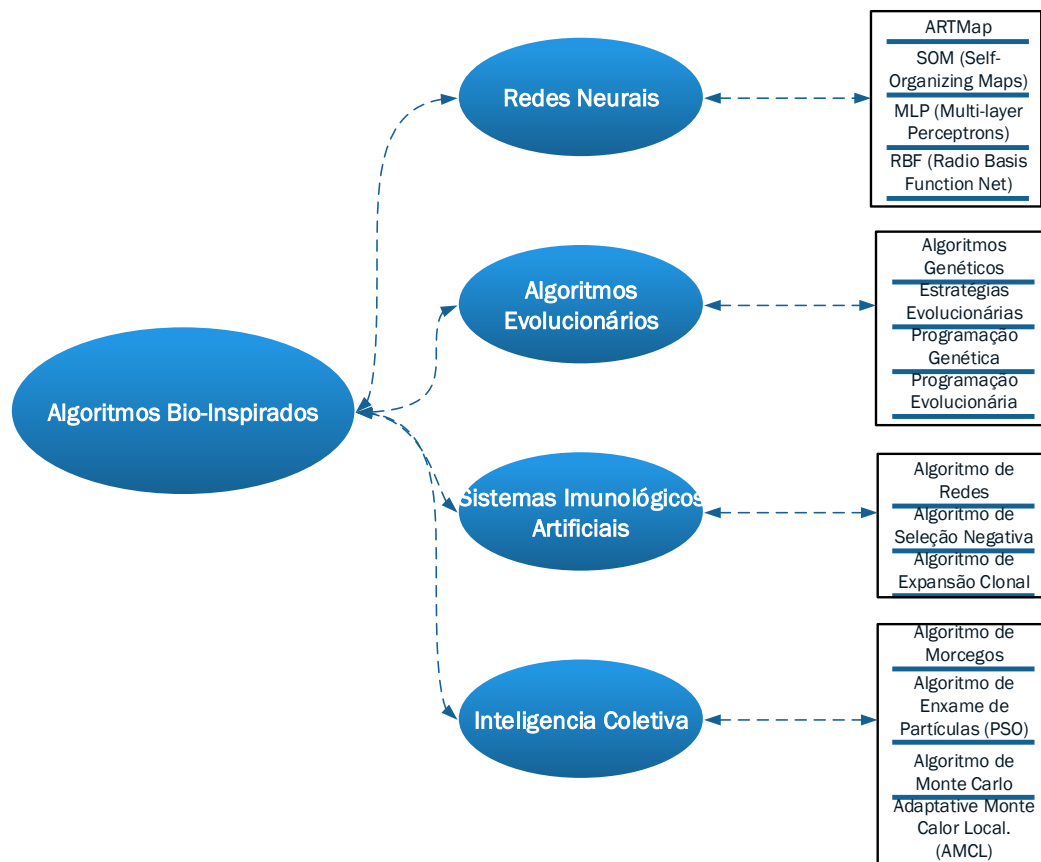
Usando variações de técnicas clássicas, existem vários trabalhos relacionados na literatura: um filtro de Kalman combinado com inferência Fuzzy para reduzir a ambiguidade na localização [16]; o método Monte Carlo estendido para a localização de robôs integrando câmeras para reduzir a incerteza sobre os *landmarks* [17]; o Filtro de Partículas Acelerado, baseado no cálculo das *landmarks* para reduzir o esforço computacional; o processo cooperativo usando um Filtro de Kalman estendido parcialmente descentralizado com base no abandono de mensagens [18]; e o método para localizar robôs móveis chamado *Interactive Closest Point* (ICP). O algoritmo calcula o movimento relativo do robô entre duas configurações consecutivas do ambiente, objetivando o alinhamento dessas medidas e a estimativa da posição do robô [19].

Uma técnica que ganhou atenção crescente dos pesquisadores é a localização utilizando o Mapeamento Simultâneo Visual (vSLAM). O algoritmo vSLAM executa reconhecimento de objetos para construir um mapa através de marcos visuais, permitindo a navegação robô móvel com boa precisão em vários ambientes do mundo real. É uma técnica capaz de processar imagens em 3D, detectar características de nível baixo (canto / lados) e alto (estruturas complexas). Um exemplo de aplicação vSLAM foi desenvolvido por Nguyen et al. (2014), onde um robô móvel fornece serviços de localização para pessoas com deficiência visual.

1.1 JUSTIFICATIVA

O problema de localização exige um algoritmo estocástico (heurístico ou meta-heurístico), capaz de entregar boas soluções sem a necessidade de que seja a melhor solução na maior parte das situações adversas. Em sua maioria, os algoritmos meta-heurísticos usam estratégias de pesquisa global e local. A randomização propõem uma estratégia de fugir da busca local para a busca na escala global. Logo, os algoritmos meta-heurísticos são adequados para otimização global [20]. A Figura 2 apresenta de modo esquemático alguns dos Algoritmos Bioinspirados apresentados na literatura, a começar por Redes Neurais [21, 22, 23, 24], passando por Algoritmos Evolucionários [25, 26, 27], algoritmos inspirados em Sistemas Imunológicos [28, 29], e findando com algoritmos que utilizam a proposta de Inteligencia Coletiva [30, 31, 32, 33].

Figura 2 – Algoritmos Bioinspirados.



Fonte: Elaborada pelo autor

Algoritmos Bioinspirados fazem parte da proposta de computação natural, que são técnicas computacionais que envolvem a natureza na forma de inspiração ou metáfora. A metáfora é utilizada em um alto nível de abstração, não contendo todos os elementos evolutivos dentro do método proposto, sabido que simplificações são necessárias quando baseia-se em elementos da natureza, devido a sua complexidade estrutural e dinâmica. Pode-se dividir a computação natural em três grandes grupos [34]:

- Computação inspirada
 - Computação Evolucionária
 - Inteligência Coletiva
 - Redes Neurais Artificiais
 - Sistemas Imunológicos Artificiais
 - Sistemas Endócrinos Artificiais
- Simulação e emulação de fenômenos

Geometrias Fractais

Vida Artificial

- Computação utilizando meios naturais

Computação baseada em DNA

Computação Quântica

Localizar um robô móvel em um ambiente, onde sua posição inicial e orientação não está disponível, requer a utilização de um algoritmo com propriedades globais na busca da solução, capaz de oferecer uma elucidação robusta e adaptativa. Para tanto, este trabalho irá apresentar os resultados da localização baseada no algoritmo bioinspirado de morcego, inspirado na capacidade biológica do processo de ecolocalização [30]. A característica dos morcegos serem capazes de identificar distância, tamanho, posição, e velocidade de deslocamento de suas presas utilizando um bio-sonar (ecolocalização) atrelado ao voo são responsáveis pelo sucesso e evolução da espécie [35], tornando o algoritmo de morcegos promissor.

No intuito de melhorar o desempenho do algoritmo de morcegos diante de cenários que exigem quantidade de cálculos significativa, para que se possa localizar o robô, é proposta uma modificação no algoritmo clássico proposto por [36]. O Algoritmo de morcegos é versátil e capaz de ser aplicado em diferentes problemas, como por exemplo, para resolver o problema do cacheiro viajante de forma discreta [37], o problema de despacho térmico, onde o algoritmo proposto tem como objetivo minimizar o custo do combustível e a emissão de poluentes simultaneamente [38].

Buscando aumentar a capacidade de exploração e melhorar a busca local surgem variantes do algoritmo de morcego (BA), o algoritmo de morcego modificado (MBA) por exemplo, apresenta resultados experimentais onde o MBA é comparativamente melhor do que o BA [39]. Pesquisas bem recentes apresentam o algoritmo de morcegos atrelado ao elemento da teoria do efeito Doppler, de modo a contribuir na busca da melhor solução, mostrando que essa modificação supera o algoritmo original e, portanto, pode ser um método alternativo eficiente na solução de problemas de otimização [40].

Há também na literatura, estratégias que buscam mesclar o filtro de partículas (FP) com o algoritmo de Morcegos, de modo a otimizar o algoritmo [41] e tentar se beneficiar das características de ambos os algoritmos, ao muito próximo a mescla do algoritmo de morcegos com o algoritmo de vagalumes em [42], um algoritmo híbrido inspirado na natureza.

1.2 OBJETIVO

O objetivo deste trabalho é propor um algoritmo de otimização eficiente, que seja eficaz na localização de robôs móveis em diferentes cenários (sensores, mapas, aplicações). É importante que o algoritmo garanta uma confiabilidade na localização apresentando erros menores ou iguais aos algoritmos comparados nos testes práticos e simulações.

Para tanto, este trabalho propõem uma extensão do algoritmo de morcego padrão, utilizando o comportamento padrão de colônias, capaz de lidar com situações laboriosas de localização utilizando até mesmo sensores de menor custo, além do sensor laser já utilizado, aumentando o horizonte de aplicabilidade do Algoritmo de Morcego Baseado em Líderes (LBBA) e mostrando a robustez do método.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em sete capítulos, sendo o primeiro (Cap. 1) destinado a apresentar a contextualização do tema pesquisado e justificativas pertinentes para a pesquisa do mesmo, além das publicações decorrentes do trabalho. O segundo capítulo (Cap. 2) disserta a respeito dos trabalhos relacionados ao tema que inspiraram e serviram de guia para aprimorar os conhecimentos relacionados a localização otimizada de robôs móveis. O capítulo três (Cap. 3) aborda os fundamentos teóricos necessários para a compreensão da pesquisa, reforçando os conceitos do algoritmo clássico de morcegos e a localização adaptativa de Monte Carlo. Em seguida tem-se o capítulo quatro (Cap. 4) onde se esclarece o algoritmo baseado em líderes, o qual gerou a publicação no *Journal of Control, Automation and Electrical Systems*.

Os resultados são apresentados no capítulo cinco (Cap. 5), onde se comparou o algoritmo proposto com outros algoritmos de otimização em diferentes mapas, foi feita uma análise comparativa também com algoritmo que varia o número de partículas, e um cenário onde o robô era sequestrado. O capítulo seis (Cap. 6) aponta os próximos passos a que talvez possam ser realizados em trabalhos futuros. Encerrando, tem-se no capítulo seis (Cap. 7) a conclusão deste trabalho.

1.4 CONTRIBUIÇÃO DESTE TRABALHO

A contribuição dessa pesquisa fez-se na utilização de líderes, característica que tornou o algoritmo de morcegos capaz de superar situações com respostas ambíguas (mínimos locais), intensificando a busca da localização real do robô e reduzindo a incidência de falsos positivos.

1.5 PUBLICAÇÕES DECORRENTES DESTE TRABALHO

Ao longo da pesquisa o trabalho gerou duas publicações: XII Simpósio Brasileiro de Automação Inteligente (XII SBAI), ‘Bat Search Algorithm Aplicado na Localização de Robôs Móveis’, e no *Journal of Control, Automation and Electrical Systems*, ‘*Mobile Robot Localization Based on the Novel Leader-Based Bat Algorithm*’ [43]. O vídeo deste trabalho pode ser visto no link <http://www.youtube.com/watch?v=dV8vDXjAmjM> - *Evolutionary Artificial Intelligence applied to Mobile Robotics Localization*.

2 REVISÃO BIBLIOGRÁFICA

A busca por métodos de localização é um tema recorrente e importante no âmbito da robótica móvel. Amplamente utilizada no transporte industrial e logística a utilização de robôs móveis necessita de precisão, estabilidade e capacidade de localizar de forma eficiente. Em algumas situações bases móveis são utilizadas com a estratégia de rastreamento de linha, uma maneira fácil e barata, para se lidar com o movimento autônomo de robôs, entretanto, é uma ferramenta que apresenta dificuldades em ser implantada nos mais variados ambientes [44].

Os métodos de localização vem evoluindo ao longo dos anos. Servindo-se dos conhecimentos obtidos em pesquisas centradas no reconhecimento, identificação e localização. Alguns métodos foram ganhando embasamentos teóricos que auxiliaram na navegação e localização de robôs móveis. O *Simultaneous Localization and Mapping* (SLAM), por exemplo, é um método que busca localizar o robô em um ambiente desconhecido, a base móvel começa a se mover de uma posição desconhecida e ao longo do processo de movimentação, ela é posicionada de acordo com a estimativa de posição. Ao mesmo tempo, um mapa incremental é construído sobre as bases de seu próprio posicionamento, auxiliando a realizar a sua localização e navegação [45].

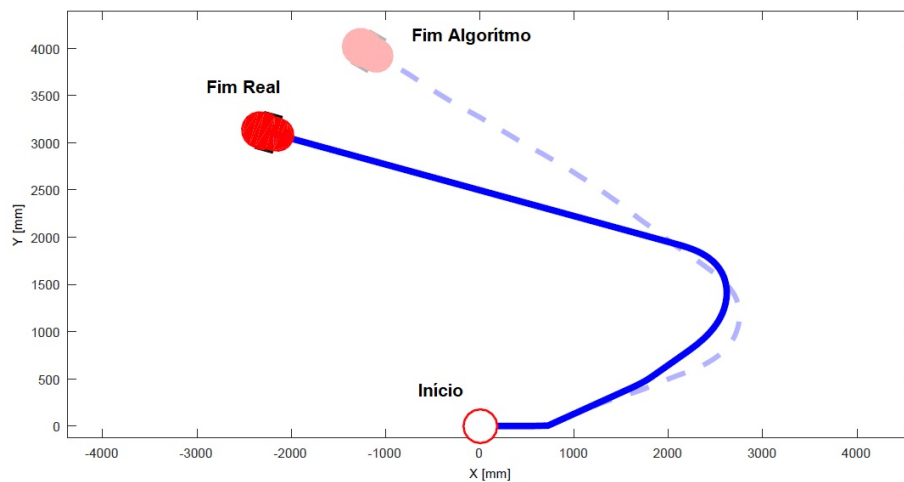
Existem diversas fontes de erros aleatórios que prejudicam a estimação da pose de um robô, tais como erros oriundos de sensores e atuadores; desníveis no solo; escorregamentos, dentre outras. A escassez de informação, erros sistemáticos dos sensores e a preocupação com a normalização dos dados são trabalhadas pela sociedade acadêmica desde 1985 [46], além de explosões combinatórias e restrições de dados [47], sonares já eram utilizados para se determinar a posição bidimensional e a orientação de um robô móvel dentro de uma sala [48] no século passado.

Apesar de algumas técnicas como o mapeamento simultâneo se mostrarem promissoras, necessita-se em determinadas situação de aplicação de mais de um robô, no intuito de evitar que obstáculos, no ponto cego dos sensores, colocassem em risco a integridade do robô [49]. Embora um mapa da área onde será realizada a missão autônoma faça parte do conhecimento prévio do sistema robótico, se faz necessário, posteriormente, a utilização de um algoritmo de localização, para indicar as localizações estimadas dos robôs ao longo de sua movimentação.

Ter uma boa localização é uns dos principais requisitos para garantir o bom desempenho do robô em suas missões de navegação, obtida pela estimativa probabilística da pose de um robô com relação às posições conhecidas de pontos de interesse (*landmarks*) disponíveis. Em alguns casos, são utilizadas bordas verticais no ambiente [50] ou características obtidas através de um mapa definido e conhecido, utilizando de correspondências para localização do robô [14, 15, 51].

Os diversos tipos de erros aleatórios deterioram sobremaneira a pose do robô conforme este se movimenta, de tal forma que o cumprimento da missão torne-se inviável, como exemplificado na Figura 3 onde, por motivos de deslizamento das rodas em relação ao piso ou o fato do piso ser irregular pode atrapalhar o sensor de odometria. A figura exhibe o robô móvel P3DX se deslocando com o objetivo de realizar uma missão, ele parte do ponto inicial (0 mm) e realiza uma curva para a esquerda, nesse momento há um aumento de erro de odometria evidenciado pela rota que o P3DX acha que fez (pontilhada) e pelo real caminho percorrido pelo robô (reta contínua). Dentre as técnicas probabilísticas de localização encontradas na literatura, considera-se as técnicas unimodais, como os filtros de Bayes e Kalman, que depositam sua crença estatística em uma única hipótese Gaussiana. Técnicas unimodais podem não recuperar a posição correta do robô após um determinado distanciamento limite [14].

Figura 3 – Acúmulo de erro de odometria durante a realização de uma missão.



Fonte: Elaborada pelo autor

O Filtro de Partículas (FP) possui múltiplas hipóteses para o estado que se deseja monitorar. Entretanto, a sua desvantagem é o elevado número de cálculos necessários a medida que é necessário utilizar mais partículas, se comparado com as unimodais [15]. Por ser um processo contínuo, sua limitação se dá quando o robô não possui variabilidade nas medidas de seus sensores (por falta de movimentação ou recebimento de novos dados). O FP é proveniente do algoritmo *Monte Carlo Localization* (MCL), que foi apresentado como um eficiente estimador de estados para ser aplicado em localização de robôs móveis. O MCL é uma versão da localização de Markov, o qual aplica métodos baseados na amostragem para aproximar as distribuições de probabilidade, de forma a empregar o método onde for necessário [52].

Algumas vezes o usuário/robô possui o mapa de onde será realizado a missão, em outras ocasiões só se sabe algumas características do cenário e, em determinadas circunstâncias, se torna necessário mapear e se localizar ao mesmo tempo para realizar a

missão. O algoritmo *Adaptive Monte Carlo Localization* (AMCL) pode ser empregado para este tipo de problema. Sua proposta é utilizar a média de curto prazo da probabilidade de medição e a relacionar com uma média de longo prazo para determinar o número de amostras aleatórias. A probabilidade de adição de uma amostra aleatória leva em consideração a divergência entre o curto, médio e longo prazo da probabilidade de medição [53]. Se o valor das funções objetivos de curto prazo (daquela iteração) é melhor ou igual ao valor de longo prazo (encontrado ao longo da otimização), não se adicionam amostras aleatórias. No entanto, se o risco de curto prazo é pior do que a longo prazo, as amostras aleatórias são adicionadas (sorteia-se novas posições) na proporção para o quociente destes valores. Desta forma, uma deterioração repentina na probabilidade de medição induz um aumento do número de amostras aleatórias.

Algumas linhas de pesquisa no intuito de diminuir o esforço computacional gasto na busca de encontrar o local correto do robô, optam por diminuir os espaços de busca. Ou seja, explorar uma pequena parcela dos dados que se possui do mapa, para que a localização seja mais eficiente e menos custosa computacionalmente. Por exemplo, localizar um robô em um prédio, composto por várias cômodos (quase idênticos) e que em cada região há um sinal proveniente da rede sem fio vindo de diferentes roteadores. Logo, é possível utilizar a WLAN para determinar a região de interesse dentro do universo de informações que se tem do mapa, auxiliando assim no uso de outros algoritmos probabilísticos de busca [54].

Filtrar dados para conseguir um melhor desempenho de localização nem sempre se limita aos mapas. Nos casos que se deseja localizar um objeto em uma imagem se faz necessário trabalhar o conjunto de informações para que o algoritmo de localização tenha um desempenho eficiente. Wu (2014) demonstra em seu trabalho como essas técnicas podem ser utilizadas, por exemplo, para prever e atualizar as posições dos veículos em câmeras de vigilância, demonstrando a robustez de algoritmos como o FP aplicados inclusive na localização através de visão computacional.

Garantir uma boa localização pode envolver grandes investimentos. Alguns sensores podem ser demasiadamente caros para serem empregados em situações do cotidiano. Pensando nisso, algumas pesquisas visam não só garantir a eficiência da localização como também apresentar sensores mais baratos (câmera estereoscópica, por exemplo) se comparados aos tradicionais *rangefinders lasers*. Considere a diminuição do custo de uma cadeira de rodas semi-autônoma ao substituir um sensor laser por um equipamento mais acessível como o kinect[®] (ambos exercendo a mesma função na localização). Isso auxilia não só os usuários finais desta nova tecnologia, como proporciona também uma série de novas oportunidades de pesquisa na área.

Existem muitos desafios relacionados à robótica móvel, não só a localização em um mapa conhecido como também quando se torna necessário o mapeamento do ambiente para realizar a localização de um robô posteriormente. Barcelos, Vidal e Rosa (2014), por

exemplo, em seu trabalho apresentam os desafios de se construir uma grade de ocupação para o reconhecimento de obstáculos e geração de um mapa, além da complexidade quando se trata da disponibilidade de energia e custos para realizar o que se chama de SLAM, o qual foi proposto por Sebastian Thrun após uma pesquisa patrocinada pelo Programa *Mobile Autonomous Robot Software* (MARS) da *Defense Advanced Research Projects Agency* (DARPA) [55].

A utilização de câmeras ajudou a propor abordagens 3D de mapeamento, as partículas de algoritmos como FP passaram a ser substituídas por modelos gaussianos mais complexos em um mapa de *grids* e conseguindo resultados mais precisos em determinadas situações [56]. Modificar um método na busca de simplificar o SLAM é um tema bastante atual, um cenário com característica esparsa (*landmarks* bem separados, dispersos, e/ou escassos), por exemplo, pode significar um custo considerável para um *Factored Solution to the Simultaneous Localization and Mapping* (FastSLAM) tradicional, onde cada partícula é uma amostra de um caminho, enquanto cada Filtro de Kalman Estendido (EKF) é apenas um estimador de característica.

Um trabalho recente o qual propõem uma significativa redução computacional, apresenta o algoritmo FP adaptativo baseado em dispersão das partículas utilizando sensor Kinect (equipamento de baixo custo) chama atenção para a necessidade de aliar otimização na busca da uma localização eficiente com um bom custo benefício [57]. A técnica responsável por dar característica adaptativa à proposta do artigo é denominada Distância de *Kullback-Leiber* (KLD), buscando obter o número eficaz de partículas compensando velocidade e precisão para diferentes cenários durante a localização (uma vantagem aos algoritmos de quantidade de partículas fixo), ajustando adequadamente o número de partículas ao longo do tempo (estado de busca e estado de monitoramento) permitindo a propagação de uma pequena quantidade de partículas em áreas de baixa incerteza e um elevado número em áreas de alta incerteza. Apesar do trabalho buscar uma economia computacional, a técnica KLD têm a desvantagem de precisar dividir o mapa em “caixas”, que pode ser altamente ineficiente para locais de grandes dimensões.

Ambientes complexos e de difícil aquisição dos dados são numerosos quando se faz um experimento em um local que não seja controlado, logo um método capaz de se localizar e/ou mapear com recursos dispersos se faz necessário. O trabalho realizado por pesquisadores da *National Chi-Nan University* adota um conceito de filtro de partículas com múltiplos filtros de Kalman. Comparado com o FastSLAM comum, na qual cada hipótese é uma amostra de um caminho do robô, enquanto que cada EKF é um estimador de características, o algoritmo proposto no trabalho de Jian-Hua Chen e Kai-Yew Lum acopla a estimativa da posição do robô em cada EKF, onde a partícula/hipótese é uma amostra do movimento [58]. Apesar de mostrar uma aplicação somente no universo simulado, o método proposto se mostra eficiente na construção do mapa, baseando-se

em um ambiente de cinco pontos de referência e transformando em linhas os obstáculos cenário proposto, atingindo bom desempenho e viabilidade.

Em 2015 objetivando minimizar os erros de hometria Furtado (2015) defende a utilização de um algoritmo bio-inspirado no comportamento de morcegos que corrige a localização de um robô móvel ao longo de uma missão de deslocamento. O Algoritmo de Morcegos foi comparado com Filtro de Partículas devido a necessidade de substituir o filtro como algoritmo de localização devido à complexidade computacional exponencial da teoria. O algoritmo bioinspirado apresentou melhores resultados tanto para o erro entra a pose real e a apresentada pelo robô, quanto para o número de iterações e tempo de convergência. Um estudo mais recente busca agregar precisão na localização com baixos custos investidos em sensores, propondo uma localização fundamentada em odometria visual - *Monocular Visual Odometry* (MVO) - através de uma abordagem geométrica de localização assistida por mapa para robôs móveis equipados com MVO. Sendo a medida obtida através da odometria visual modelada como grupos de partículas, obedecendo uma distribuição Gaussiana uniforme e incertezas de medição [59]. O trabalho propõe um ajuste de parâmetros para reduzir as ambiguidades do método.

A busca por melhores resultados em precisão e eficiência computacional dos algoritmos de localização é um tema ainda discutido, devido a sua relevância para a robótica móvel. Sobreira (2018) apresenta em seu artigo uma comparação entre algoritmos, considerando a precisão da estimativa de pose, a eficiência computacional, e a velocidade de convergência métricas de comparação. É importante levantar as características do local em que se deseja realizar a missão, visto que a particularidade de cada ambiente vai exigir um sensor mais apto e demais características que compõem um robô, por exemplo, como será feita sua locomoção (pernas, rodas, esteiras).

Sanguino (2016) fez uma comparação de sensores em seu trabalho, compilada na Tabela 1, evidenciando de forma clara a importância da utilização do sensor Kinect em relação a outros sensores também utilizados na literatura. Entretanto, a precisão ficou comprometida em alguns casos de localização global devido à maior complexidade para extrair dados relevantes dos cenários para comparação em relação à simplicidade do rastreamento através apenas de cores (experimento realizado no trabalho).

Neste presente trabalho os ambientes de testes são locais fechados e controlados, entretanto, o conhecimento a priori contido nos mapas utilizados pelos algoritmos contém informações não tão fiéis em sua totalidade com a estrutura real. Para isso se fez necessário utilizar um sensor com alta precisão de escaneamento e alta velocidade, capaz de entregar dados confiáveis e que sua representação, através da sua modelagem, durante as simulações não comprometesse o processamento do algoritmo.

Tabela 1 – Comparação de características e capacidades de diversos sensores

Sensor	Alcance	Custo	Resolução	Medida	Vantagens	Desvantagens
Sonar	Grande	Médio	Baixo	1-D/3-D	Menos atenuação das ondas no ambiente aquático, tecnologia melhorada constantemente desde sua invenção	Os falsos ecos, as reflexões, a resolução da imagem dependem da frequência de trabalho
Inf.V.	Pequeno	Baixo	Médio	1-D	Requisitos de baixo custo e de baixa potência, implementação de projeto simples	Curto alcance, reflexões, linha de visão direta, leve e sensível ao clima
Laser (SICK)	Pequeno	Baixo	Médio	1-D/2-D	Alta precisão de escaneamento, modo de varredura até 360°, sensor de alta velocidade	Reflexões, a maior parte das vezes o laser fornece detecção planar, comprimento de onda limitado do sinal transmitido
LIDAR	Grande	Muito alto	Alta	2-D/3-D	Dados georreferenciados, alto nível de precisão, abrange grandes áreas	Consumo de alta potência, incapacidade de penetrar em um obstáculo muito denso, sem padrão de protocolos, conjuntos de dados muito grandes difíceis de interpretar e processar
RADAR	Grande	Alto	Alta	2-D	Capacidade de penetração superior através de qualquer tipo de condição climática	Consumo de alta potência, resposta de baixa velocidade, ampla propagação do feixe, grandes alvos fechados saturam o receptor, sensível a fontes de interferência
Câmera Digital	Médio	Baixo	Alto	Somente vídeo RGB	Grande espaço de armazenamento, imagens de alta resolução, amplos canais ópticos	A distância deve ser deduzida da transmissão de vídeo, incapacidade de ser usada à noite, consumo de bateria
Kinect	Médio	Baixo	Médio	RGB-D	Sensor de profundidade de baixo custo permite tarefas multiuso, FOV vertical	Precisão de profundidade versus distância, FOV horizontal reduzido, fonte de energia externa necessária

2.1 CONCLUSÃO

A utilização de diferentes sensores em alguns artigos que compõem esta revisão bibliográfica demonstra a preocupação de sanar problemas com características diversas (Tabela 2). Observa-se a busca de diminuição dos custos de localização ao longo do tempo a medida que novas tecnologias foram sendo desenvolvidas. O módulo Kinect chegou ao mercado no ano de 2010 com um custo inferior em relação a alguns outros sensores além de oferecer um conjunto de recursos (Câmera RGB, Infra Vermelho, Microfone, e até mesmo IMU integrada).

Tabela 2 – Utilização de Sensores

Autor/Ano	Sonar	Câmera	Laser	Kinect	Pré-Requisitos
Drumheller (1987) [48]	X				Mapa conhecido
Atiya (1993)[50]		X			Mapa conhecido
Stein (1995) [60]		X			Mapa conhecido
Fox (1999) [52]			X		Mapa conhecido
Thrun (2002) [14]			X		Mapa conhecido
Araujo (2002) [16]	X		X		Mapa conhecido
Woo (2006) [51]	X				Mapa conhecido
Liang (2007) [17]		X	X		Mapa conhecido
Djehaich (2013) [19]			X		-
zhang (2013)[53]				X	-
Lee (2014)[54]			X		Mapa conhecido
wu (2014)[61]		X			-
Nguyen (2014) [62]		X		X	Detecção Landmarks
Sanguino (2016) [57]				X	Mapa conhecido
Jiang (2017) [59]		X			Mapa conhecido

Nesse trabalho foi utilizado o sensor laser Sick devido a proposta de desenvolver uma missão *indoor*, que não se dispusesse de muitos recursos nos mapas, no intuito de simular situações críticas de localização, por exemplo, uma usina nuclear ou uma mina de mineração em que uma situação de risco ocorrera e não possuísse energia elétrica. Mesmo sem a luz das lâmpadas e sem sinal de wi-fi, o robô deveria ser capaz de se localizar em um ambiente previamente conhecido.

O sensor laser é utilizado em diversas aplicações de localização em que se tem um conhecimento a priori do ambiente, só que condições desfavoráveis. Uma missão de salvamento dentro de uma mina por exemplo [63], que poderia ser comprometida o uso da teleoperação, poderia ser utilizado em conjunto para realizar uma missão autônoma [64].

3 FUNDAMENTOS TEÓRICOS

3.1 INTRODUÇÃO

A metodologia elaborada neste trabalho busca mostrar a eficiência do algoritmo proposto. Neste capítulo será apresentado de forma mais precisa a organização, disposição e ordem dos elementos essenciais que compõem a metodologia proposta e experimentos realizados, com a mesma, para corroborar sua eficácia.

O sensor utilizado em testes simulados e mapas reais, com robô real, foi o laser scan SICK LMS-200, do fabricante SICK™, o qual está instalado em um robô Pioneer 3-DX (Figura 4) como dispositivo de medição de distâncias o qual irá auxiliar na localização do robô.

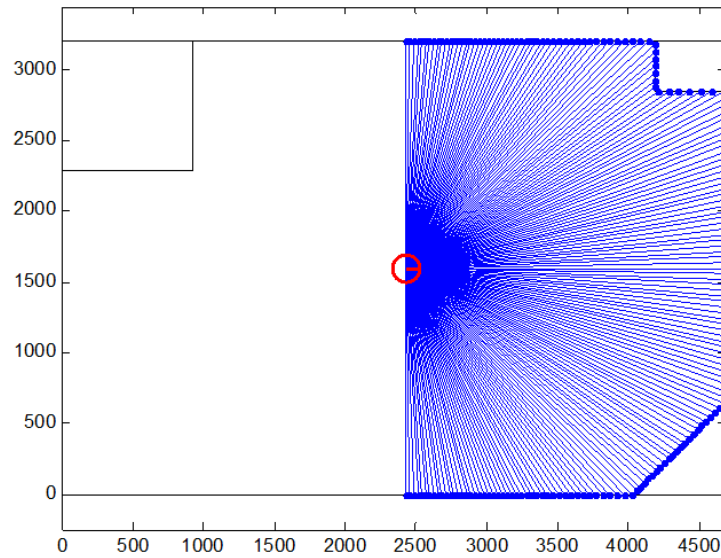
Figura 4 – SICK LMS-200.



Fonte: Elaborada pelo autor

O sensor Sick LMS-200 é um sensor exteroceptivo (empregado para medir as informações do ambiente) capaz de realizar 181 medidas de distância num ângulo de 180 graus como retratado na Figura 5. Este sensor emite um feixe laser a cada um grau aproximadamente, ao colidir com um obstáculo o sensor é capaz de entregar como saída da leitura a distância do laser em relação ao obstáculo. Quanto mais leituras utilizadas para analisar um ambiente que se deseja localizar, mais preciso é o sensoriamento.

Figura 5 – Modelo esquemático das medidas de distância.

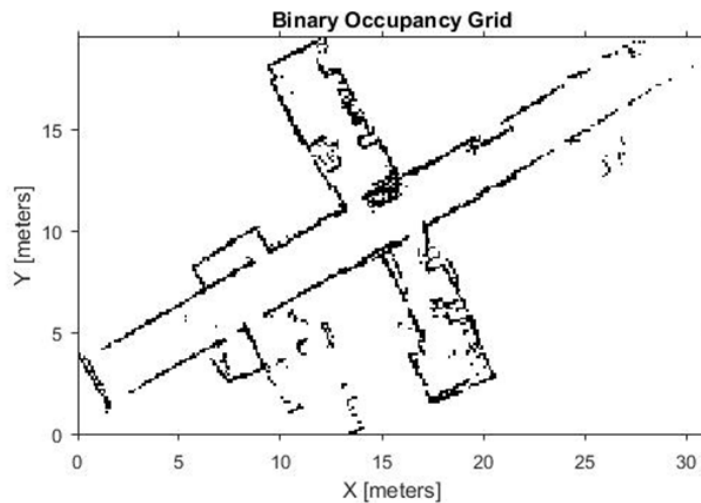


Fonte: Elaborada pelo autor

A escolha do sensor é de extrema importância, um vez que é necessário buscar a aplicabilidade da metodologia em diversos tipos de problemas e representações. No contexto da localização pretende-se empregar o algoritmo proposto em diferentes tipos de mapas e *landmarks*, incluindo nos testes mapas apenas simulados e mapas de ambientes reais, por exemplo, mapa do prédio do Programa de Pós-Graduação de Engenharia Elétrica da Universidade Federal de Juiz de Fora (PPEE-UFJF):

- **Occupancy Grid:** O mapa do ambiente é representado como um campo uniformemente espaçado de variáveis aleatórias binárias, as quais representam a presença de um obstáculo como na Figura 6 [65].

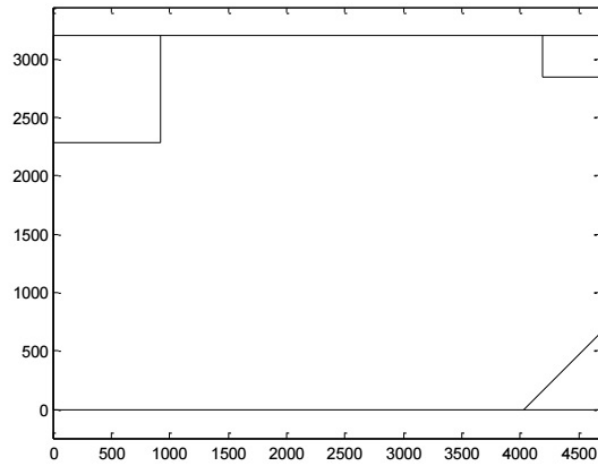
Figura 6 – Mapa Occupancy Grid PPEE-UFJF.



Fonte: Elaborada pelo autor

- **Retas:** O ambiente é retratado através de um mapa topológico, sendo os obstáculos presentes descritos na forma de retas (Figura 7).

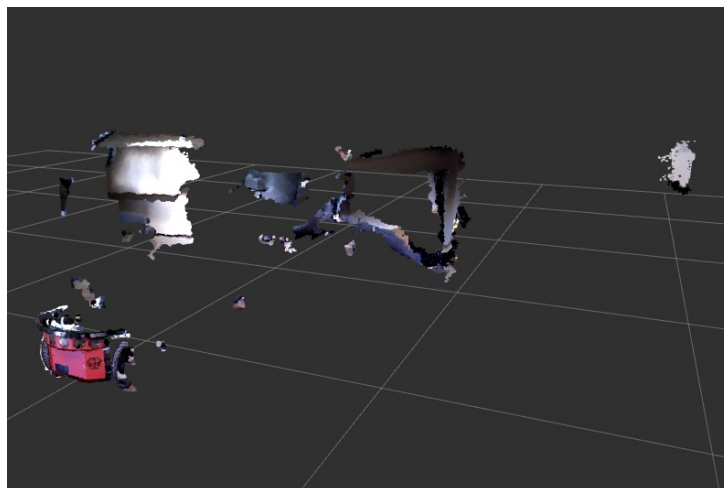
Figura 7 – Mapa de Retas.



Fonte: Elaborada pelo autor

- **Visuais:** Cenário referente ao mapa, obstáculos e *landmarks* são representados por Nuvem de pontos ou imagens (Figura 8).

Figura 8 – Mapa de nuvem de pontos



Fonte: Elaborada pelo autor

3.2 LOCALIZAÇÃO DE MONTE CARLO ADAPTATIVA - AMCL

O Filtro de Partículas é um método de Monte Carlo onde as partículas são simulações contínuas do sistema. Cada partícula representa uma solução e um grupo de partículas formam um sistema de soluções múltiplas que pode estimar a probabilidade total posterior do estado que o filtro deseja rastrear. Este algoritmo usa uma quantidade predefinida de partículas, que é escolhida pelo usuário. À medida que o número de

partículas cresce, melhor é a probabilidade posterior estimada para o sistema, no entanto, sua complexidade computacional cresce exponencialmente [66]. Cada partícula representa a “simulação” do robô e seus sensores em uma posição fictícia do mapa real.

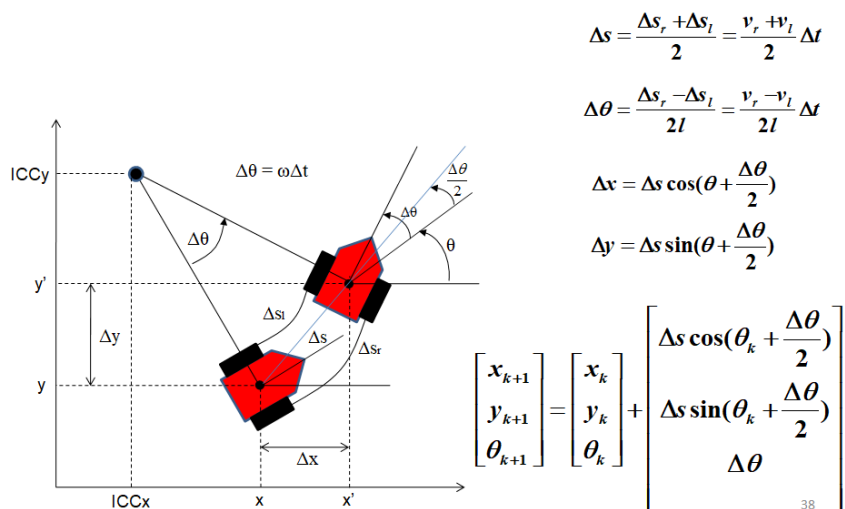
O FP (em sua forma mais simplificada) tem quatro etapas principais, como mostrado em Algoritmo 1. A primeira requer mover o robô real e as partículas exatamente da mesma maneira, a fim de obter variabilidade nas medidas, evitando problemas de mínimos locais. Esta movimentação é composta por uma translação e uma rotação $(\Delta S, \Delta\theta)$, a Figura 9 apresenta como o modelo incremental computa a nova pose do robô a cada Δt em função das velocidades das rodas.

As linhas 1 e 2 do algoritmo aproximado estão inclusas nessa primeira etapa. O desvio padrão é representado pelos parâmetros σ ($\sigma_x, \sigma_y, \sigma_\theta$), o parâmetro N significa a quantidade de partículas utilizadas que serão distribuídas inicialmente de maneira aleatória e uniforme pelo ambiente para realizar a otimização.

Algoritmo 1 Algoritmo Filtro de Partículas

- 1 **inicialize os parâmetros:** $\sigma_x, \sigma_y, \sigma_\theta, N, \Delta S, \Delta\theta$;
 - 2 **inicialize as partículas:** x_i , onde $i = 1, \dots, N$;
 - 3 **evolução das partículas:** obter os pesos iniciais das partículas ω_i ;
 - 4 **while** condição de parada **do**
 - 5 **desloca** robô e partículas com ΔS e $\Delta\theta$;
 - 6 $\Delta X = \Delta S \cdot \text{Cos}(\theta + (\Delta\theta/2))$;
 - 7 $\Delta Y = \Delta S \cdot \text{Sen}(\theta + (\Delta\theta/2))$;
 - 8 **leitura** dos sensores do robô e partículas ;
 - 9 **cálculo do peso** de cada partícula i com a máxima verossimilhança: ω_i ;
 - 10 **reamostragem** das partículas utilizando o algoritmo de roleta ;
 - end**
 - 11 **return** *media(partículas)*
-

Figura 9 – Modelo incremental da movimentação do robô.



Fonte: Elaborada pelo autor

Na linha 3 define-se a função objetivo (FOB) para a avaliação das partículas, responsável por determinar os pesos (ω_i) das mesmas.

A segunda etapa, configurada na linha 8, adquire as leituras dos sensores (tanto as medidas do robô real quanto as medidas simuladas das partículas). Já na terceira etapa estima-se o peso (também conhecido como *fitness*) de cada partícula calculando a probabilidade máxima verossimilhança entre as medidas do sensor do robô e as medidas simuladas dos sensores de cada partícula.

A quarta etapa redimensiona as partículas com menor peso (*fitness*) em regiões próximas daquelas com probabilidade de estarem em uma posição próxima do robô real (partículas com melhor peso), esse redimensionamento é feito de maneira justa através do algoritmo de roleta de cassino que considera um desvio padrão (σ) em torno da provável posição original.

Então, o algoritmo repete esses quatro passos até que as condições de parada estejam satisfeitas. Nesta abordagem, as partículas são circulares em relação aos limites do ambiente [66].

A probabilidade utilizada na terceira etapa de peso das partículas é dada por:

$$\text{maximize } L_i = \exp\left(-\frac{1}{2} \left(L_p^i - L_r\right)^T \Sigma^{-1} \left(L_p^i - L_r\right)\right) \quad (3.1)$$

onde L_i é a verossimilhança da i -ésima, partícula, que está compreendida no intervalo $[0, 1]$, L_p^i e L_r são os vetores de marcos detectados de, respectivamente, a partícula i -ésima e o robô na instante atual, e Σ é a matriz de covariância do sensor usado para detectar pontos de referência.

Na robótica móvel, tanto Filtro de Partículas como o AMCL são comumente usados para obter a posição bidimensional do robô, dado por $[x, y, \theta]^T$. O AMCL difere do Filtro de Partículas na implementação, sua abordagem de localização é adaptativa, tendo como método de análise para o aumento ou redução do número de partículas a distância de Kullback-Leibler (*KLD-sampling*). Para este fim, N partículas são distribuídas aleatoriamente através do ambiente conhecido em uma distribuição uniforme. O algoritmo compara as medidas obtidas do robô real com as medidas virtuais obtidas das partículas e analisa se deve aumentar ou diminuir o número de partículas na busca do objetivo.

O método AMCL é uma melhoria no algoritmo FP, através da variação no número de amostras necessárias no decorrer do processo iterativo. Durante a localização global o robô é completamente ignorante em relação a sua posição real e com isso tem-se a probabilidade de localização igual por todo o mapa de forma uniforme. O sistema de localização do robô pode ser definido da seguinte forma: 1) O robô realiza sua missão em um ambiente bidimensional, que pode ser descrito, por exemplo, na fórmula de células (*grids*); 2) O robô é equipado com sensores para obter leituras e medições em tempo real

do ambiente, e com base nessas medidas, os dados do robô são confrontados com leituras simuladas das partículas, o que permite determinar uma estimaco de localizaco [67].

O Algoritmo 2 descreve o AMCL proposto por Zhang (2013). Inicializam-se variveis estticas w_{slow} e w_{fast} (mdias de longo e curto prazo, respectivamente), e um nmero mximo de partculas N_{kld} . Conjuntos de partculas, que sero compostos por suas posices x_t^i e pesos w_t^i , so inicializados vazios: $X_t = \emptyset$, $\bar{X}_{t-1} = \emptyset$. Na linha 3 do Algoritmo, as $i = 1, 2, \dots, N_{t-1}$ partculas da iterao interior passam por uma varredura, que poder inserir mais partculas conjunto de posices anteriores \bar{X}_{t-1} . Caso um nmero randmico ($rand()$) seja menor que o mximo da funo $\max(0, 1 - w_{fast}/w_{slow})$,  adicionado de forma aleatria mais uma partcula com posico no conjunto X_{t-1} e incrementa o contador de partculas acrescentadas. Quanto maior a relao w_{fast}/w_{slow} , maior a probabilidade de incremento de uma nova partcula, visto que essa condio aponta para diminuico da qualidade da estimaco.

Cada *bin* b pertencente a um histograma G  igualado a zero. Subsequentemente,  realizada uma amostragem no conjunto de partculas, acarretando em um conjunto das partculas amostradas \bar{X}_{t-1} , de tamanho N_t . Cada um dos $i = 1, 2, \dots, N_t$ elementos tem receber a mesma entrada u_t do rob real, gerando novas posices x_t^i , e com suas medidas simuladas so calculados seus peso w_t^i , atravs das entradas F (medidas do sensor real) e MAP o mapa. Todas essas novas posices e pesos so colocadas no conjunto de partculas X_t , para uso na prxima iterao. A mdia dos pesos das partculas w_{avg}  calculada, e ento  analisado o conjunto de *bins*.  alocado um *bin* para cada posico das partculas existente, alm de acrescentar uma unidade no contador de *bins* N_b .

Ao fim do Algoritmo, so calculados os pesos w_{slow} e w_{fast} , alm do nmero de partculas mxima N_{kld} , para a prxima iterao. O Algoritmo retorna o conjunto X_t de partculas, de onde pode-se estimar a melhor posico.

Li (2010) afirma em seu artigo que como algumas partculas podem ter pesos relativamente grandes, o problema da degeneraco de peso pode ocorrer no processo de fuso de partculas. Para evitar isso, faz-se necessrio controlar o aumento dos pesos das partculas. Por fim,  recalculado o limite de partculas utilizado atravs da equao proposta por Zhang (2013) na linha 19.

 possvel constatar na literatura e atravs de testes prticos que o nmero de amostras necessrias para a estimativa da posico influencia no tempo de execuo do algoritmo devido a quantidade de cculos e comparaes a serem feitas, entretanto, novas propostas foram pesquisadas e desenvolvidas buscando diminuir a quantidade de cculos sem perder a eficincia. Neste intuito, a proposta do AMCL  introduzir uma abordagem para aumentar a eficincia do filtro de partculas adaptando o nmero de amostras  incerteza do estado [68]. Ou seja, um nmero fixo de amostras, exige na maioria dos casos utilizar grandes conjuntos de amostras de modo que atenda tanto a localizaco global de um

Algoritmo 2 Adaptive Monte Carlo Localization

```

1 Variáveis Estáticas:  $w_{slow}$ ,  $w_{fast}$ ,  $N_{kld}$  ;
2  $X_t = \emptyset$ ,  $\bar{X}_{t-1} = \emptyset$ ,  $N_{t-1} = |X_{t-1}|$ ,  $N_r = 0$ ,  $N_{b=0}$ ,  $w_{avg} = 0$  ;
3 for  $i = 1$  to  $N_{t-1}$  do

4   if  $rand() < \max\{0, 1 - w_{fast}/w_{slow}\}$  then
      adiciona de forma aleatória uma pose em  $\bar{X}_{t-1}$  ;
6      $N_r = N_r + 1$ 
   end
end
;
7 foreach  $b \in G$  do
    $b = 0$ 
end
;
8  $\bar{X}_{t-1} = \text{sampler}(X_{t-1}, \max\{0, N_{kld} - N_r\}) \cup \bar{X}_{t-1}$  ;
9  $N_t = |\bar{X}_{t-1}|$  ;
10 for  $i = 1$  ate  $N_t$  do

11    $x_t^i = \text{sampleMotionModel}(u_t, \bar{x}_{t-1}^i)$  ;
12    $w_t^i = \text{measurementModel}(x_t^i, F, MAP)$  ;
13    $X_t = X_t \cup \langle x_t^i, w_t^i \rangle$  ;
14    $w_{avg} = w_{avg} + \frac{w_t^i}{N_t}$  ;
15   if  $bin(x_t^i) = 0$  then

16      $bin(x_t^i) = 1$  ;
17      $N_b = N_b + 1$ 
   end
end
;
18  $w_{slow} = w_{slow} + \alpha_{slow}(w_{avg} - w_{slow})$  ;
19  $w_{fast} = w_{fast} + \alpha_{fast}(w_{avg} - w_{fast})$  ;
20 if  $N_b > 1$  then
    $N_{kld} = \frac{N_b - 1}{2\varepsilon} \{1 - \frac{2}{9(N_b - 1)} + \sqrt{\frac{2}{9(N_b - 1)}} z_{1-\delta}\}^3$  ;
21 else ;
22    $N_{kld} = 1$ 
end
;
23 retorna  $X_t$ 

```

robô quanto o problema de rastreamento de posição durante uma missão, em contraste, o AMCL adapta o número de amostras durante o processo de localização, escolhendo grandes conjuntos de amostras durante a localização global e pequenos conjuntos de amostras para o rastreamento de posição, uma abordagem que demanda de um esforço computacional significativamente menor [69].

3.3 ALGORITMO DE MORCEGOS - *Bat Algorithm*

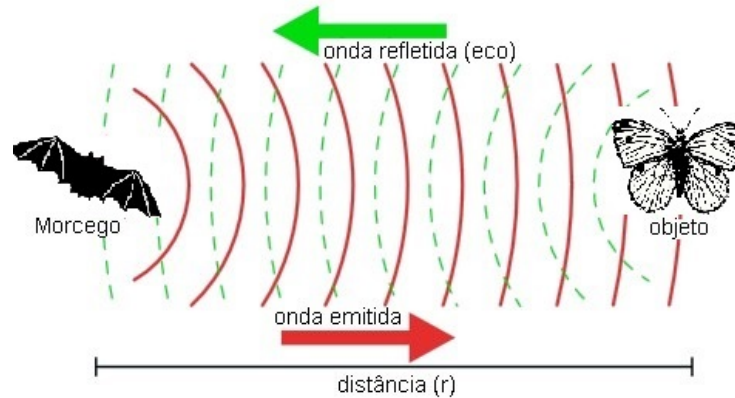
O comportamento de alguns seres vivos existentes na natureza tem sido uma inspiração para o desenvolvimento de novas metodologias no campo da otimização e da pesquisa operacional. A maneira inteligente de coordenar vários indivíduos para lidar com os problemas é o conceito de inteligência de enxame [31]. Desde a primeira proposta baseada em otimização de enxame de partículas (PSO), muitos outros algoritmos com a mesma estratégia de comportamento foram pesquisados, como a colônia de formigas, colônia de abelhas, morcegos e outros [20].

O *Bat Algorithm* (BA), desenvolvido por Yang (2010), é um algoritmo baseado no comportamento individual e coletivo dos morcegos, o qual se baseia em: (i) a capacidade desses mamíferos para determinar a distância e posição das presas (objetivo) pela emissão de ondas de ultra-som; (ii) cada morcego analisa o tempo de voo dessas ondas de som e relaciona o eco para construir um modelo d -dimensional do ambiente (eco-localização).

Quando um local em potencial é identificado, ou seja, a possível localização correta do robô, cada morcego modifica a própria transmissão das ondas, tanto na emissão de pulso quanto na amplitude da onda (Figura 10). Os morcegos mudam sua posição x_i com uma velocidade v_i em um espaço de busca d -dimensional. O método proposto é mostrado no Algoritmo 3, onde para cada morcego i , x_i é a posição atual; v_i é a velocidade; f_i é a frequência relacionada ao fator de escala do comprimento de onda que está no intervalo $[f_{\min}, f_{\max}]$; ξ_1 , ξ_2 e ξ_3 são números randômicos uniformes normalizados; ϵ é um número randômico normal; α , β , γ e r_0 são fatores de escala, sendo que as variáveis α , β estão compreendidas numa faixa de valores entre [0-1], pois são responsáveis por influenciar a busca local (valor de α decresce) e global (valor de β cresce) respectivamente; r é a taxa de emissão de pulso; A é a intensidade dos pulsos; $f(x_i)$ é o peso (*fitness*); \hat{x}_i é a posição estimada do morcego i e $f(\hat{x}_i)$ são suas *fitness*; x^* é o melhor morcego atual e $f(x^*)$ é o seu peso.

Todos os N morcegos são inicializados com os seguintes parâmetros: taxa de pulso ($r_i = 0$), velocidade ($v_i = 0$), amplitude ($A_i = 1$), frequência ($f_i = 0$), posição aleatória (x_i). Após definido as parâmetros mencionados, parte-se para a avaliação da população de morcegos (linha 3). Esta avaliação corresponde ao valor numérico da função objetivo para cada um dos morcegos (soluções do problema em análise). Diante da avaliação de toda a população, é possível determinar a posição do melhor morcego, melhor solução (linha 4).

Figura 10 – Eco-localização de morcegos.



Fonte: Elaborada pelo autor

Algoritmo 3 Algoritmo de morcegos para localização.

```

1 Inicializar a população:  $x_i$  e  $v_i$ , onde  $i = 1, \dots, N$  ;
2 Inicializar amplitudes e frequências:  $r_i$  e  $A_i$  ;
3 peso dos morcegos:  $f(x_i)$  ;
4 obtenha o melhor morcego com a melhor fitness:  $x^*$  e  $f(x^*)$  ;
5 while condição de parada do
6   for  $i = 1, \dots, N$  do
7      $f_i = f_{\min} + (f_{\max} - f_{\min}) \xi_1$ 
8      $v_i^{t+1} = v_i^t + (x_i^t - x^*) f_i$ 
9      $\hat{x}_i = x_i + v_i^{t+1}$ 
10    if  $\xi_2 > r_i$  then
11       $\hat{x}_i = x^* + \alpha \epsilon$ ;
12    end
13    pesos estimados dos morcegos:  $f(\hat{x}_i)$  ;
14    if  $f(\hat{x}_i) \leq f(x_i)$  e  $\xi_3 < A_i$  then
15       $x_i = \hat{x}_i$  ;
16       $f(x_i) = f(\hat{x}_i)$ 
17       $r_i = r_0 (1 - \exp(-\beta t))$ 
18       $A_i^{t+1} = \gamma A_i^t$ 
19    end
20    if  $f(\hat{x}_i) < f(x^*)$  then
21       $x^* = \hat{x}_i$  ;
22       $f(x^*) = f(\hat{x}_i)$ 
23    end
24  end
25 end
26 return  $x^*$ 

```

Após esta etapa, parte-se efetivamente para o processo de busca bio-inspirado. Para tanto, a frequência de cada morcego é atualizada (linha 7), sendo a mesma utilizada para determinar a nova velocidade (linha 8) e conseqüentemente a nova posição (solução)

temporária de cada morcego (linha 9). Destaca-se que a posição temporária é função da melhor posição encontrada até o instante de tempo (discreto) t .

Determinada a posição temporária, parte-se para a etapa de busca local (linha 10). Nesta etapa, uma componente aleatória é inserida podendo ser usada tanto para exploração quanto para intensificação, dependendo do tamanho do passo.

Conhecida a solução temporária, seja obtida pela atualização da posição e velocidade ou pelo processo de busca local surge uma pergunta: deve-se aceitar ou não esta solução? Se a condição (linha 13) for verdadeira, a solução temporária é aceita de forma circular.

Além disso, um aumento da taxa de pulso é considerado, sendo que para t tendendo ao infinito a taxa de emissão de pulsos tende ao valor unitário (linha 14). Ou seja, com o passar do tempo a busca local se intensifica. Outro parâmetro atualizado é a amplitude, onde a mesma decresce através de uma taxa de diminuição α (linha 15). Para valores altos de amplitude tem-se uma probabilidade maior de aceitar novas soluções. Para valores de amplitudes baixos, uma solução de qualidade ruim é raramente aceita.

Em seu livro Yang (2010) afirma que ao implementar e comparar o algoritmo de morcegos com outros algoritmos é possível perceber o quanto é promissor. O autor afirma que o BA se destacava dentre os algoritmos de enxames de partículas, algoritmos genéticos, e busca harmônica. Tudo depende do ajuste fino dos parâmetros (frequência e taxa de emissão de pulso) que estão relacionados diretamente a proximidade ou aptidão dos locais que apontam a solução global otimizada.

O Grupo de Otimização Heurística e Bioinspirada (GOHB) da UFJF propôs a utilização do BA [36] para a localização de robôs em 2015 [70], e foi possível verificar o promissor horizonte de atuação na robótica quando comparado com o algoritmo clássico do FP. Entretanto, neste trabalho, não se comparou resultados com AMCL pois o foco era obter resultados iniciais analisando os modelos clássicos de cada uma das proposta dos algoritmos.

3.4 ROS MATLAB®

O conceito de um *framework* operacional de robôs iniciado na Universidade de Stanford, evoluiu através da *Willow Garage* e agora residente na *Open Robotics* influenciou grande parte da pesquisa e indústria de robótica nos últimos anos. Em 2007 quando o engenheiro do laboratório de robótica do *Silicon Valley* (Willow Garage) publicou um novo repositório de código no SourceForge (disponibilizado publicamente a qualquer pessoa no mundo), ajudou a criar à base de código para o projeto em que Willow estava trabalhando. O repositório do código ROS, criado por Ken Conley em 7 de novembro de 2007 foi a primeira vez que o termo ROS foi usado como uma designação pública formal do projeto [71].

O ROS é constituído por ferramentas, bibliotecas e algoritmos para auxiliar no desenvolvimento de softwares para robô, que o torna flexível para programação de robôs e controle de plataformas robóticas. Desde o ano de 2015 o MATLAB® vem trabalhando na integração com o ROS, utilizando a *toolbox Robotics System Toolbox™*, através do qual muitos usuários podem desenvolver aplicações de robótica autônoma para veículos terrestres, manipuladores e robôs humanoides por exemplo. Na documentação da *toolbox* de Peter Corke é possível ter acesso a parte dedicada ao ROS (Figura 11). Além de tutoriais é possível encontrar quais funcionalidades já estão implementados no *software*.

Figura 11 – Robot Operating System (ROS).

Robot Operating System (ROS)
Access ROS networks, robots, and simulators

Robotics System Toolbox™ enables you to interface with ROS and use ROS functionality in MATLAB® and Simulink®. You can connect to a ROS network, collect data, send and receive your own messages, and deploy code to a standalone system. To get started, you must first connect to a ROS network. See [Get Started with ROS](#).

Frequently Viewed Topics

- [Connect to a ROS Network](#)
- [Work with rosbag Logfiles](#)
- [Access the TF Transformation Tree in ROS](#)
- [Communicate with the TurtleBot](#)
- [Built-In Message Support](#)
- [Generate a standalone ROS node from Simulink®](#)

Network Connection and Exploration
Set up and interact with ROS networks

Publishers, Subscribers, and Services
Create and receive messages, topics, and network information

Log Files and Transformations
Analyze rosbags, transformation trees, and time series data

Specialized Messages
Access messages from specialized sensors and inputs

Custom Message Support
Create ROS custom messages

Robots and Simulators
Simulate TurtleBot® and Gazebo applications, connect to TurtleBot hardware

ROS Access with Simulink
Access ROS networks and messages using Simulink

Fonte: Elaborada pelo autor

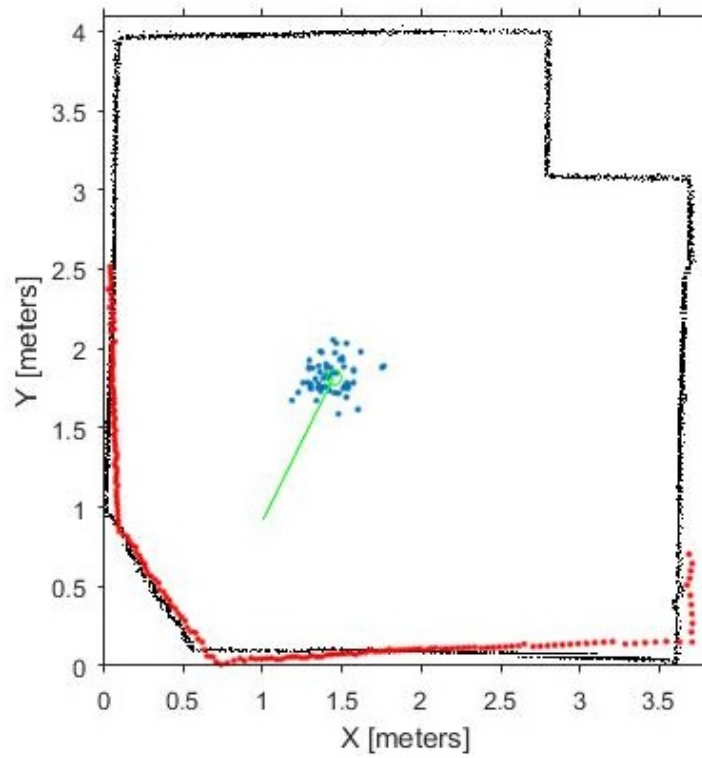
Neste trabalho o MATLAB® comunica-se com o robô utilizando o protocolo do ROS, esta comunicação é assimilada pelo *framework* como um nó. Os *scripts* desenvolvidos no *software* foram responsáveis tanto para parte de localização, utilizando os algoritmos de otimização para determinar a posição real do robô no mapa (real/simulado), como para o controle do P3DX durante a realização da missão (seja ela autônoma ou teleguiada).

Para realizar a localização do P3DX no mapa é necessário passar para os algoritmos informações dos sensores de odometria e laser, essas informações são adquiridas através dos *subscribers* das mensagens dos tópicos *pose* (odometria) e *scan* (laser), sensores presentes no robô móvel. Como a localização proposta nesse trabalho depende de um conhecimento prévio do mapa que se deseja executar a de localização do robô, é imprescindível que ao longo dos *scripts* se carregue os dados do ambiente que será realizado a missão, seja um mapa representado na forma de retas ou de células de ocupação, os dados do cenário em que se deseja atuar devem buscar representar de forma mais fiel possível o ambiente.

Durante a localização o usuário consegue observar a evolução da busca pelo ótimo global. Ambos algoritmos apresentam a posição das partículas no mapa, a melhor partícula, e compara a posição desta com a informação do robô real. Sendo os pontos vermelhos

indicadores das leituras reais vistas do ponto indicado pelo algoritmo como posição ótima (Figura 12).

Figura 12 – Demonstrativo de localização no mapa Occupancy Grid.



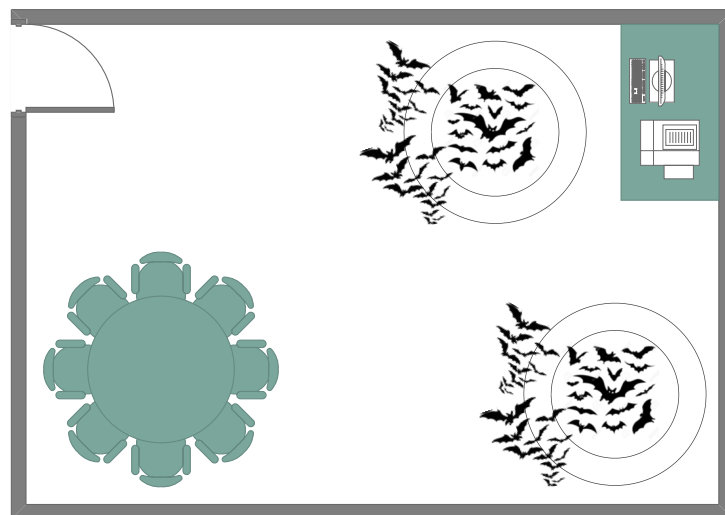
Fonte: Elaborada pelo autor

4 ALGORITMO PROPOSTO - LBBA

Este capítulo tem como objetivo apresentar o Algoritmo de Morcego Baseado em Líderes – *Leader-based Bats Algorithm*, que é uma modificação do algoritmo baseado em ecolocalização de morcegos original. Em seu artigo, Yang (2013) menciona as aplicações do algoritmo de morcegos desde sua proposta inicial em 2010, como em processamento de imagem [72], classificadores [73] e etc. O algoritmo de morcego proposto por Yang (2010) é uma abstração do comportamento social destes animais, demonstrando promissora eficiência para otimização, contudo apresentando dificuldades em lidar com ótimos locais.

Pressupõe-se que o algoritmo original não reflete a capacidade auto-adaptativa dos morcegos em relação ao meio ambiente, por ser uma simplificação muito restrita. Algumas modificações levam em conta modelagens mais completas, por exemplo como a consideração do efeito Doppler na ecolocalização dos morcegos, apresentada por Meng (2015). Nesta nova abordagem proposta, um certo número de indivíduos líderes influenciam no comportamento do grupo, e não apenas um único líder. Isso equivale a introduzir uma heurística para estabilizar o movimento das partículas em ótimos locais e, assim, o algoritmo deve ter facilidade de convergir para o ótimo global em cenários mais complexos. A Figura 13 apresenta um diagrama esquemático do algoritmo proposto, onde a população é dividida em dois grupos influenciados por dois líderes, indivíduos cujas localizações são consideradas mais aptas.

Figura 13 – Líderes influenciando na busca do ótimo global



Fonte: Elaborada pelo autor

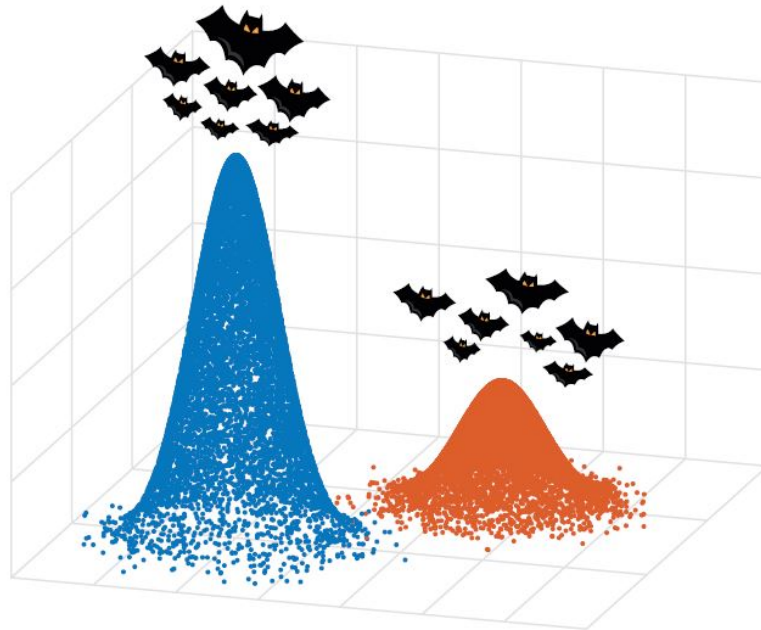
4.1 ALGORITMO DE MORCEGO BASEADO EM LÍDERES

O LBBA é um filtro não paramétrico baseado no BA, que se utiliza de um número finito de indivíduos (morcegos), cada um destes correspondendo a aproximadamente a uma região no espaço de busca. Thrun (2005) defendem que os filtros não paramétricos

são bem adaptados para representar crenças multimodais complexas, como no caso da localização de um robô.

Assim como outros algoritmos bio-inspirados, o LBBA representa o estado dos indivíduos *a priori* por um conjunto de amostras aleatórias. O algoritmo de morcego baseado em líderes tem por característica a evolução dos estados dos morcegos, que devem se aproximar dos indivíduos mais aptos. A distribuição do conjunto de amostras tende a se aproximar de uma distribuição normal, mesmo com o caráter não paramétrico do filtro. Considerando essa característica de representação de ótimos locais, o algoritmo proposto representa um espaço muito mais amplo de distribuições do que modelos gaussianos baseados no método original [65]. A Figura 14 ilustra a representação aproximada da distribuição normal dos indivíduos, influenciados por dois líderes na colônia, que podem significar posições ambíguas de um robô no mapa.

Figura 14 – Representação Aproximada da Distribuição Normal



Fonte: Elaborada pelo autor

Embora eficiente, o BA mantém uma representação restritiva da crença sobre a pose do robô, devido ao fato dos indivíduos se dirigirem a um único líder. O algoritmo não se torna inviável para alguns tipos de mapas complexos, entretanto, possui a desvantagem de um excessivo custo computacional quanto ao número de morcegos necessários para encontrar o ótimo global.

O processo de otimização do algoritmo de morcegos clássico usa tanto o melhor global, g^* , como o melhor individual, x^* . A razão de usar x^* é aumentar a diversidade nas soluções, no entanto essa diversidade pode ser obtida usando artifícios de busca randômica. A proposta de utilizar diversos líderes, estimulando o desenvolvimento de colônias, busca

diminuir a aleatoriedade do voo dos morcegos na busca do ótimo global, auxiliando uma análise mais eficiente do espaço de solução. No Algoritmo 4 é explicitada a estrutura base do LBBA:

Algoritmo 4 Algoritmo de Morcegos Baseado em Líderes - LBBA.

```

1 Inicializar a população:  $x_i$  e  $v_i$ , onde  $i = 1, \dots, N$  ;
2 Define quantidade de líderes:  $l = 1, \dots, L$  ;
3 Inicializar frequências e amplitudes:  $r_i$  e  $A_i$  ;
4 Avalia o peso dos morcegos:  $f(x_i)$  ;
5 Obtenha os líderes:  $x_L^*$  e  $f(x_L^*)$  ;
6 Define o melhor morcego com a melhor fitness:  $x^*$  e  $f(x^*)$  ;
7 while condição de parada do

8   for  $i = 1, \dots, N$  do

9      $f_i = f_{\min} + (f_{\max} - f_{\min}) \xi_1$ 
10     $v_i^{t+1} = v_i^t + (x_L^* - x_i^t) f_i$ 
11     $\hat{x}_i = x_i + v_i^{t+1}$ 
12    if  $\xi_2 > r_i$  then

13       $\hat{x}_i = x^* + \alpha \epsilon$ ;
14    end
15    pesos estimados dos morcegos:  $f(\hat{x}_i)$  ;
16    novos líderes:  $f(\hat{x}_L^*)$  ;
17    if  $f(\hat{x}_i) \leq f(x_i)$  e  $\xi_3 < A_i$  then

18       $x_i = \hat{x}_i$  ;
19       $f(x_i) = f(\hat{x}_i)$ 
20       $r_i = r_0 (1 - \exp(-\beta t))$ 
21       $A_i^{t+1} = \gamma A_i^t$ 
22    end
23    if  $f(\hat{x}_i) < f(x^*)$  then

24       $x^* = \hat{x}_i$  ;
25       $f(x^*) = f(\hat{x}_i)$ 
26    end
27  end
28 end
29 return  $x^*$ 

```

Inicialmente, o LBBA estabelece o número de morcegos N que é usado no processo de otimização, como mostrado em 4.1. A variável M representa o conjunto de indivíduos n_i . Cada indivíduo tem dois parâmetros de caracterização, a posição x_i e a velocidade v_i .

$$M = \{N \in \mathbb{N}^+ \mid \forall n_i, \exists (x_i, v_i), 0 < i < N\} \quad (4.1)$$

Entre os N morcegos, alguns são escolhidos como líderes durante a execução do algoritmo. A quantidade de líderes é definida de acordo com a complexidade do ambiente. Os morcegos voam aleatoriamente com uma velocidade v_i em uma determinada posição

x_i , variando a amplitude A_i para procurar presas e a frequência de seus pulsos emitidos r_i em um intervalo de $[0,1]$, dependendo da proximidade do alvo. Supõe-se que a amplitude varie de um A_o positivo máximo para um valor constante A_{min} mínimo. Para cada posição individual, é necessário calcular uma função de objetivo $O(x_i)$, que é determinada por uma função de desvio mínimo, como mostrado em 4.2.

$$O(x_i) = (\mathbf{m}_i - \mathbf{m})/\mathbf{m} \quad (4.2)$$

O parâmetro m_i representa a medição realizada pelo sensor simulado do i -th individual e m é a medição realizada pelo sensor real do robô. Os morcegos com os melhores pesos tornam-se os líderes. A função de valor ideal para o i -th indivíduo f_i , sua velocidade e posição são atualizadas dinamicamente pelo algoritmo. Os parâmetros (f_{max} e f_{min}) representam as frequências máxima e mínima da taxa de emissão de pulso, respectivamente. Além disso, são variáveis pré-definidas. Os vetores de velocidade e posição são gerados por 4.6 O parâmetro β representa a mudança da taxa de pulso.

A equação acima funciona com a estratégia de que o líder mais próximo de cada colônia influencia a busca pela melhor posição. A equação de distância de cada morcego em relação aos líderes é dada por

$$dist = \sqrt{X^2 + Y^2} \quad (4.3)$$

onde

$$\begin{cases} X = \text{Lider.x} - \text{Morcego.x} \\ Y = \text{Lider.y} - \text{Morcego.y} \end{cases} \quad (4.4)$$

O cálculo da distância euclidiana dos líderes para cada outro morcego é importante para determinar o quão distantes estão algumas possíveis soluções (morcegos) daquelas posições candidatas a serem a correta (líderes). O intuito é calcular de forma rápida e eficiente a região de influência de cada um dos líderes.

Abaixo apresenta-se o passo a passo do algoritmo, observe que os parâmetros ξ_1 , ξ_2 e ξ_3 são responsáveis pela aleatorização da frequência, posição e seleção natural, respectivamente. O parâmetro α é uma variável de deslocamento aleatório e é multiplicado por ϵ , que é responsável por garantir que a variável α seja diferente de zero. Finalmente, γ é a taxa de diminuição de amplitude.

Passo 1: Estabelecer o número de morcegos a serem utilizados na otimização N , sendo x a posição aleatória de cada morcego no ambiente, e v a velocidade desses morcegos, inicialmente nulas – Linha 1 do pseudocódigo

Passo 2: Estipular a quantidade de líderes de acordo com a complexidade do ambiente – Linha 2. No caso da localização de um robô, é importante observar que a complexidade de um mapa está ligado tanto ao número de ambiguidades (ótimos locais) quanto à sua extensão.

Passo 3: Assim como a velocidade v , a amplitude A e a frequência r da taxa de emissão dos pulsos são inicializadas com um valor pré determinado – Linha 3. Por análise empírica escolheu-se os valores arbitrários para amplitude e frequência.

Passo 4: Através da função de mínimo desvio determina-se o valor da função objetivo para cada partícula – Linha 4. Conforme a seguinte expressão.

$$FOB = \frac{medida_{sensorParticula} - medida_{sensorRobo}}{medida_{sensorRobo}} \quad (4.5)$$

Passo 5: Líderes são morcegos com melhores pesos (FOB), estabelece o número de líderes proporcional a quantidade estipulada de morcegos no Passo 2 – Linha 5. Dentre os líderes há aquele morcego com a melhor *fitness* que temporariamente é a solução da otimização – Linha 6.

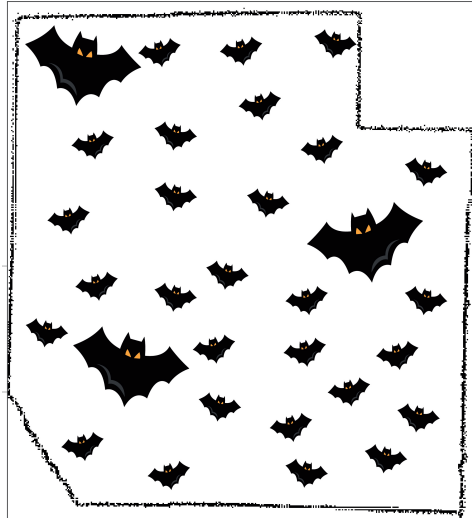
Passo 6: Apesar de ter uma estrutura bem próxima ao algoritmo BA, o LBBA vai se diferenciar da proposta padrão utilizando a estratégia de líderes (x_L^*). Na versão simplificada a convergência do algoritmo é usar apenas o melhor global. Assim, na otimização baseada apenas na inercia de partículas aceleradas, o vetor de velocidade é gerado pela Equação 4.6 das linhas 9 a 11:

$$\begin{aligned} f_i &= f_{\min} + (f_{\max} - f_{\min})\beta \\ v_i^t &= v_i^{t-1} + (x_i^t - x_*)f_i, \\ x_i^t &= x_i^{t-1} + v_i^t \end{aligned} \quad (4.6)$$

A equação da Linha 10 trabalha com a estratégia de que o líder mais perto de cada morcego influência na busca da melhor posição, e não o melhor morcego de toda população. Para elucidar melhor o método proposto pelo LBBA, as Figuras 15 a 18 explicam a contribuição do algoritmo.

Na Figura 15 é apresentado um mapa na estrutura de *grid*, a população de morcegos é espalhada de forma aleatória por todo o universo de probabilidade obedecendo os limites do mesmo. Os morcegos estampados com um tamanho maior que os demais são os considerados líderes.

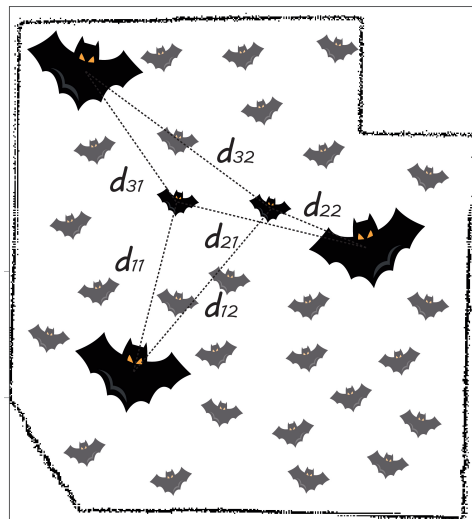
Figura 15 – Determinando os melhores morcegos da colônia - Líderes



Fonte: Elaborada pelo autor

Os líderes provavelmente se encontram em posições ambíguas ou muito próximas da posição real do robô que se deseja localizar. A próxima etapa representada pela Figura 16 se resume em analisar a posição de cada morcego em relação aos melhores da colônia:

Figura 16 – Cálculo da distância de cada morcego em relação aos Líderes



Fonte: Elaborada pelo autor

A Figura 17 denota aqueles morcegos que estão na região de influência, ou seja, localizados a uma distância menor de um morcego considerado líder em relação aos outros líderes.

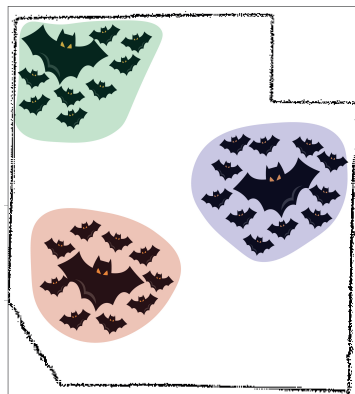
Aqueles que se encontram na micro-região de interesse do espaço de busca (mapa) são instigados a seguir o líder daquela região como apresentado na Figura 18, essa característica de busca ajuda a lidar com cenários complexos e fugir da “armadilha” do ótimo local em que o BA poderia ficar estagnado.

Figura 17 – Região de Influência dos Líderes



Fonte: Elaborada pelo autor

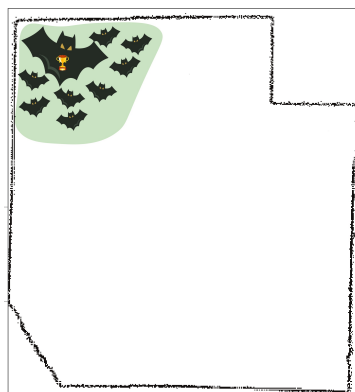
Figura 18 – Líderes influenciando na busca da otimização



Fonte: Elaborada pelo autor

O processo iterativo continua até que ocorra a condição de parada, que pode ser caracterizada pelo fato de atingir a tolerância determinada ou que o tempo de otimização máxima seja alcançado durante a localização do robô. A Figura 19 representa um cenário caso o critério de parada ocorrido tenha sido a tolerância, indicando assim a real posição do robô no mapa.

Figura 19 – Ótimo Global é encontrado pelo melhor morcego da colônia



Fonte: Elaborada pelo autor

4.2 CONCLUSÃO

O LBBA explicita um comportamento de enxame. Na resolução do problema de otimização deste presente trabalho cada morcego da colônia representa uma solução candidata. Semelhante às metodologias de Filtro de Partículas e Algoritmos Genéticos, o processo de busca é orientado pelo sucesso de algum indivíduo da população. A motivação para o voo é a busca por alimento, que no algoritmo corresponde à busca pela localização ótima. O conceito de alimento está relacionado com a função objetivo do problema (encontrar medidas iguais ou próximas ao sensor real). Nesse estudo temos um problema de minimização, logo a quantidade de alimento em uma região é inversamente proporcional ao valor da função objetivo neste ponto.

Da mesma forma que em um ambiente real há diferentes lugares com alimentos, é possível encontrar soluções com leituras ambíguas em um mapa, dificultando a busca pelo melhor lugar. Os líderes, metaforicamente, correspondem aos melhores indivíduos no espaço de busca por onde os morcegos podem voar, e influenciam apenas aqueles ao seu redor e não todos os morcegos do universo de busca.

Ao longo das iterações, a possibilidade de surgir diferentes líderes e novas regiões contribui na busca pela solução verdadeira, proporcionando a capacidade de fugir de convergências ruins, quando a busca tende a se deslocar para posições ambíguas que não correspondem a posição verdadeira do robô no mapa. Alguns outros algoritmos usam da estratégia de amostragem aleatória ao longo das iterações para tentar conseguir o mesmo efeito, entretanto, sem nenhuma inteligência relacionada.

A contribuição do *Leader-based Bats Algorithm* auxilia em uma averiguação mais extensa do universo de possibilidades não ignorando outros possíveis bons locais, prevenindo-se de falsos positivos e apresentando a expectativa de ser capaz de indicar a melhor localização de uma forma mais assertiva.

5 RESULTADOS

Neste Capítulo serão apresentados resultados comparativos do método proposto em relação a outros algoritmos de otimização presentes na literatura. Na Seção 5.1 serão apresentados testes de otimização para funções multimodais clássicas, comparando o algoritmo LBBA com BA, FP e PSO. O objetivo desses testes é analisar o desempenho dos algoritmos estatisticamente, em igualdade de condições iniciais, variando o número de partículas utilizadas para obter o ótimo global. Os testes foram realizados repetidas vezes, com diferentes sementes de números aleatórios para que fosse possível uma análise estatística dos resultados.

Na Seção 5.2 são apresentados os resultados da utilização do algoritmo LBBA para localização de um robô móvel. Para comparar seu desempenho são apresentados os resultados de simulação de outros algoritmos da literatura – BA, PSO, FP e AMCL. Os cenários de localização propostos foram implementados com robô real para atestar sua eficiência. Também são apresentados resultados de localização para o contexto de robô sequestrado: situação na qual o robô, por algum motivo (falha de sensor, ou mudança abrupta de posição) deve conseguir obter sua localização após a retomada das leituras dos sensores.

5.1 Funções de Teste Multimodais e Análise Estatística

Com o propósito de testar a eficiência do algoritmo LBBA, é realizado uma comparação de desempenho dele com outros algoritmos (BA, FP, PSO) em algumas funções clássicas de otimização que são utilizadas na literatura – Rastrigin (Equação 5.1), Eggcrate (Equação 5.2) e Ackley (Equação 5.3). A Função Rastrigin (Equação 5.1), assim como as outras duas utilizadas, é uma função não convexa bastante utilizada como teste de desempenho para algoritmos de otimização. O espaço de busca da Rastrigin pode ser verificado na Figura 20.

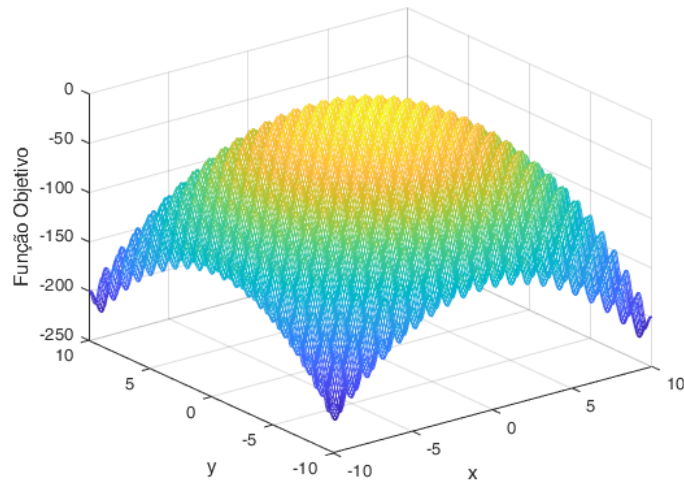
$$f(x, y) = 20 + x^2 + y^2 - 10 \cdot \cos(2\pi x) - 10 \cdot \cos(2\pi y) \quad (5.1)$$

O espaço de busca da Função Eggcrate (Equação 5.2) é apresentado na Figura 21, que devido a sua característica de ter curvas mais suavizadas, ilustra bem o cenário que os algoritmos precisaram lidar para otimizar e encontrar o máximo global.

$$f(x, y) = x^2 + y^2 + 25 \cdot (\sin(x)^2 + \sin(y)^2) \quad (5.2)$$

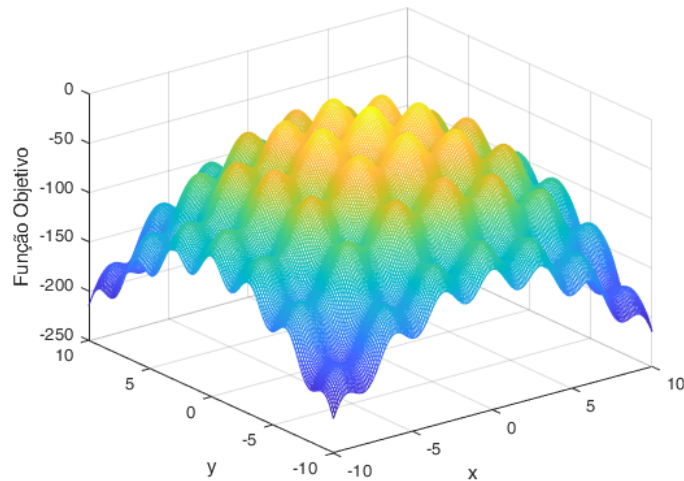
Encontrar o máximo dessas funções se torna um problema bastante difícil devido ao grande espaço de pesquisa e ao grande número de máximos locais. A Função Ackley

Figura 20 – Função Rastrigin.



Fonte: Elaborada pelo autor

Figura 21 – Função Eggcrate.



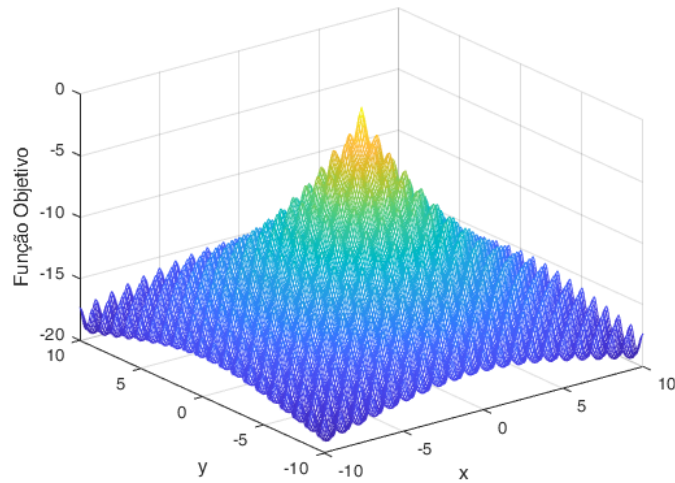
Fonte: Elaborada pelo autor

(Equação 5.3) é um excelente exemplo de como o algoritmo precisa ser robusto para conseguir escapar dos máximos locais durante as iterações. Seu espaço de busca é apresentado na Figura 22). Esta característica é análoga ao problema de localização de robôs, onde os máximos locais poderiam ser considerados posições ambíguas no mapa.

$$f(x, y) = -20 \cdot e^{-0.2 \cdot \sqrt{1/2 \cdot (x^2 + y^2)}} - e^{1/2 \cdot (\cos(2\pi x) + \cos(2\pi y))} + 20 + e^1; \quad (5.3)$$

É importante salientar que todos os algoritmos foram implementados segundo uma mesma filosofia de programação, para que se obtivesse uma análise justa do desempenho, utilizando inclusive a mesma semente para todos os algoritmos, mesmo número

Figura 22 – Função Ackley.

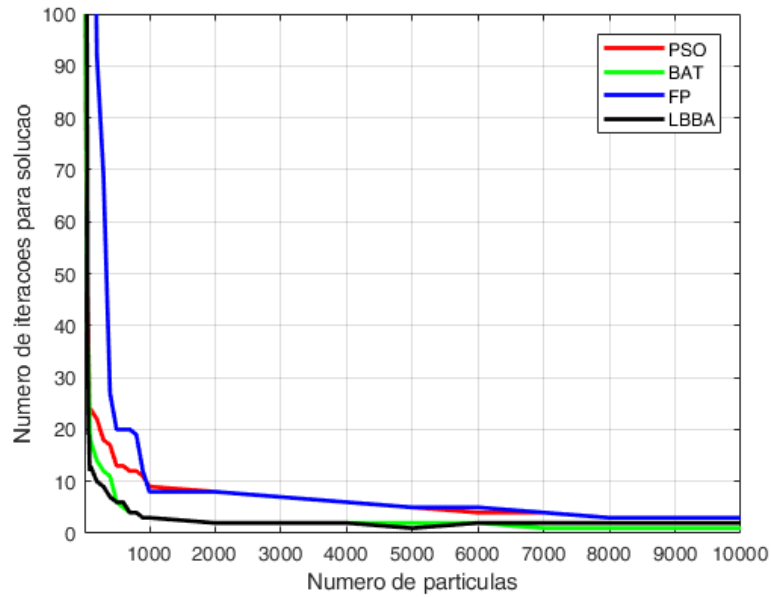


Fonte: Elaborada pelo autor

de repetições, e igualdade de condições iniciais. O espaço de busca dos algoritmos é apresentado nas Figuras 20, 21 e 22, onde o objetivo é encontrar o máximo global de cada uma das funções utilizadas. O número de partículas dos algoritmos foi confrontado com o número de iterações necessárias para convergência para as três funções, como forma de comparar o desempenho do algoritmo proposto frente aos da literatura. Para geração destes resultados, 50 otimizações das três funções foram realizadas para cada algoritmo, e a média de iterações foi calculada para cada número de partículas. Para garantir que os algoritmos fossem comparados em condição de igualdade, o conjunto inicial de partículas foi o mesmo para os algoritmos, gerados com 50 sementes de números pseudo-aleatórios diferentes.

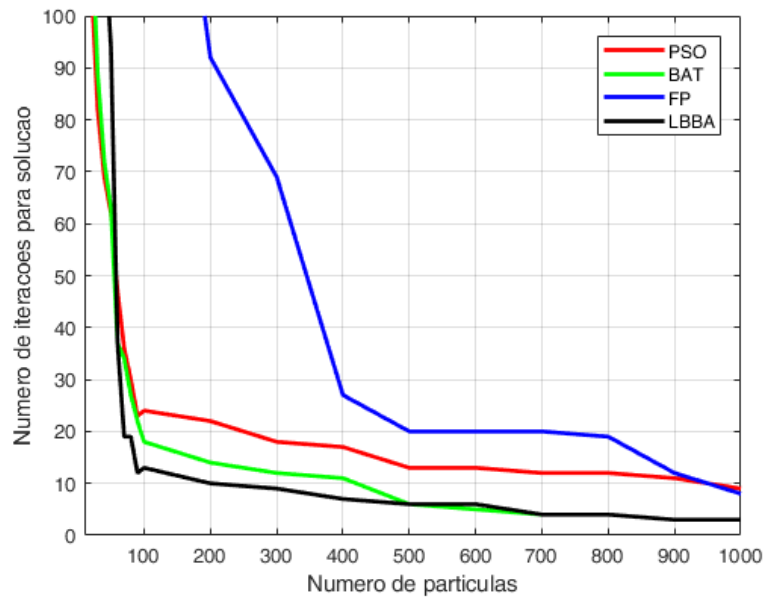
A Figura 23 apresenta o desempenho dos algoritmos para a Função Rastrigin, para um máximo de 10000 partículas. Os número de iterações dos algoritmos tendem a um mesmo comportamento com o aumento do número de partículas. A Figura 24 ressalta o intervalo de até 1000 partículas, quando a diferença de desempenho é mais relevante. O algoritmo Filtro de Partículas apresentou os piores resultados como esperado, devido ao fato dos outros algoritmos terem sido propostos como desenvolvimentos do FP. O desempenho do PSO é superior ao FP para um número baixo de partículas, mas inferior ao BA, baseado no PSO. O LBBA tem um desempenho similar ao BA, com desempenho superior para um número baixo de partículas. A diminuição abrupta do número de iterações para um número de partículas menor que 500 mostra que BA e LBBA tem desempenhos semelhantes, porém a estratégia baseada em líderes se mostra adequada para diminuição no número de iterações. Contudo, algoritmo de Morcegos clássico conseguiu resultados tão bons quanto o algoritmo proposto, a partir de um determinado número de partículas.

Figura 23 – Função Rastrigin: Número de iterações × Número de partículas



Fonte: Elaborada pelo autor

Figura 24 – Resultado da Função Rastrigin para o intervalo de 0 a 1000 partículas.

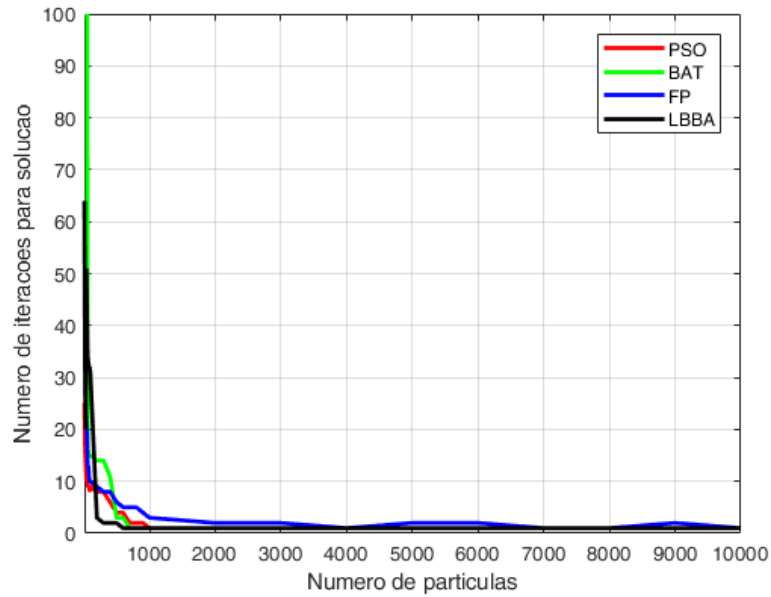


Fonte: Elaborada pelo autor

As Figuras 25 e 26 apresentam o desempenho dos algoritmos para a Função Eggcrate. Para um número abaixo de 50 partículas, os algoritmos que utilizam a meta-heurística dos morcegos não obtiveram bom desempenho. A partir de 200 partículas o desempenho do LBBA é superior os demais algoritmos, tendendo ao mesmo desempenho com o aumento do número de partículas, de modo similar ao verificado para a Função Rastrigin.

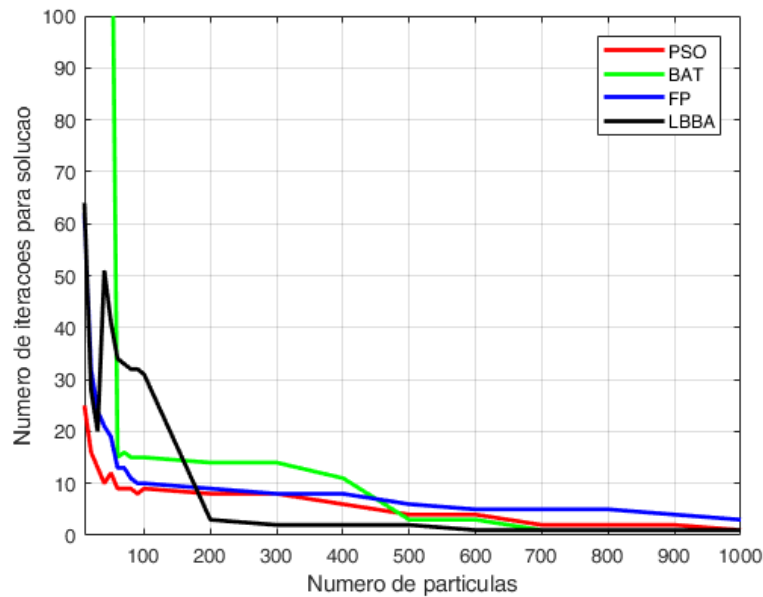
Para a Função Ackley, o FP apresenta resultados significativamente piores que os

Figura 25 – Função Eggcrate: : Número de iterações × Número de partículas



Fonte: Elaborada pelo autor

Figura 26 – Resultado da Função Eggcrate para o intervalo de 0 a 1000 partículas.

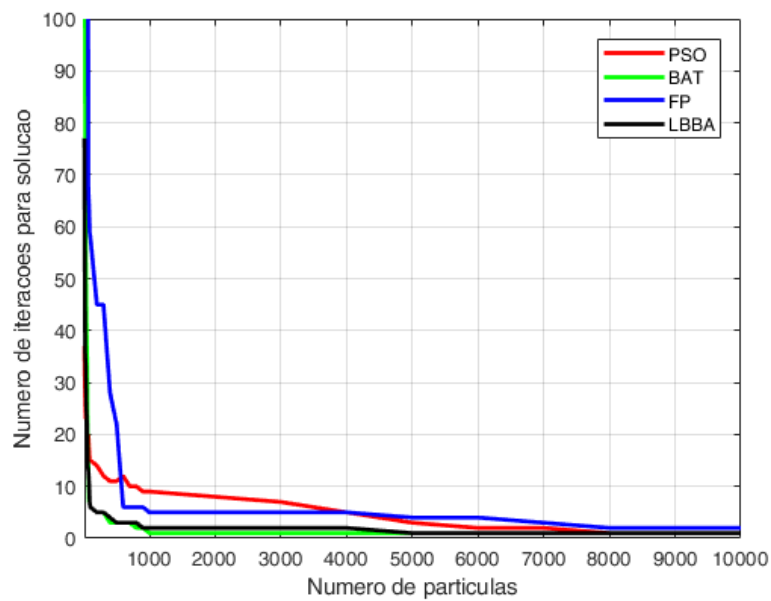


Fonte: Elaborada pelo autor

demais, como apresentado nas Figuras 27 e 28. Para um número inferior a 30 partículas o PSO obteve os melhores resultados, mas com decréscimo de iterações lento, tendo o pior desempenho na faixa de 600 a 4000 partículas. O algoritmos BA e LBBA tiveram desempenhos equivalentes, com pequena vantagem para o BA. Ambos demonstraram elevada queda de iterações para um pequeno aumento de partículas, característica favorável para este tipo de algoritmo.

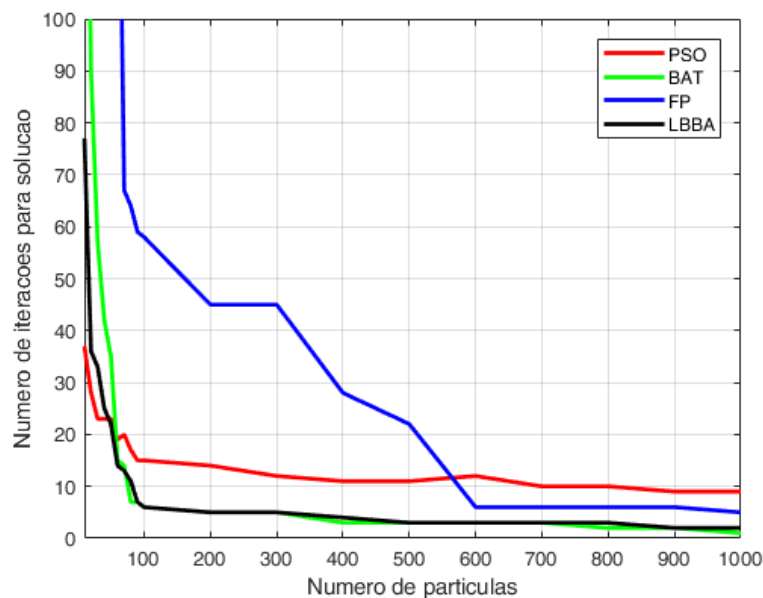
Com o bom desempenho verificado por estes testes, o LBBA demonstrou vantagens em relação aos demais, obtendo o melhor resultado ou comportamento similar ao algoritmo de melhor resultado para todas as funções analisadas. Os testes desenvolvidos na Seção 5.2 tem por objetivo ratificar se o desempenho obtido para essas funções podem ser verificados para o problema de localização de bases móveis, visto que este problema apresenta desafios multimodais.

Figura 27 – Função Ackley: : Número de iterações \times Número de partículas



Fonte: Elaborada pelo autor

Figura 28 – Resultado da Função Ackley para o intervalo de 0 a 1000 partículas.



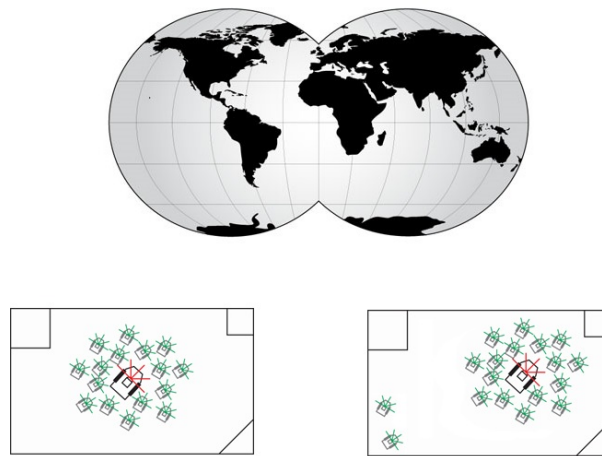
Fonte: Elaborada pelo autor

5.2 Localização de Robôs Móveis utilizando o LBBA

Nesta sessão serão apresentados os resultados da comparação do desempenho do LBBA entre: i) o algoritmo padrão de morcegos; ii) dois algoritmos cujas populações de partículas são invariantes (PSO e FP); iii) um algoritmo no qual a população de partículas é variável em cada iteração (AMCL). O hardware usado para os testes é um Intel (R) Core (TM) 2 Duo T6600 de 2,20 GHz e 6 GB de RAM. Neste processo, todos os algoritmos utilizam as mesmas condições iniciais que incluem a mesma posição inicial e representam a mesma quantidade de partículas distribuídas aleatoriamente e uniformemente através do ambiente simulado. Isto é exemplificado na Figura 30. Observe que a parte esquerda desta figura apresenta o mapa simulado e a parte direita mostra a distribuição de partículas no ambiente.

O ambiente simulado usado nesse trabalho dispõe de característica circular, ou seja, se o robô ultrapassar os limites dos extremos do mapa, aparece do lado oposto do limite transposto (Figura 29). Tal artifício auxilia na busca dos ótimos globais do problema, além de garantir uma varredura mais eficiente do cenário durante o processo de localização.

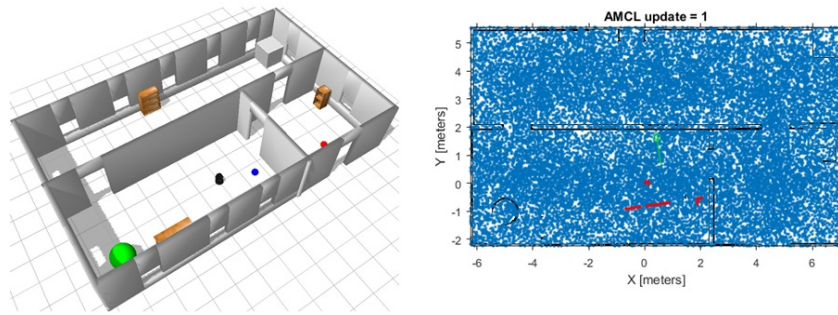
Figura 29 – Mapa Circular.



Fonte: Elaborada pelo autor

Simulações foram realizadas no simulador Gazebo [74] e avaliadas em ambiente real para validar a metodologia proposta. Todo o código foi implementado em ambiente ROS usando a linguagem de programação C++. O robô móvel Pioneer 3-DX foi usado junto com o scanner a laser SICK LMS-200 para extrair as informações do ambiente. As partículas simuladas fazem leituras semelhantes quando comparadas ao sensor. Através dessas leituras, foi feita uma comparação entre o sensor de partículas e o robô real para determinar o peso de cada partícula. Os experimentos com o P3DX foram realizados nos ambientes mostrados nas Figuras 33-34.

Figura 30 – Partículas espalhadas uniformemente pelo mapa.



Fonte: Elaborada pelo autor

5.2.1 ALGORITMOS SEM VARIAÇÃO DE QUANTIDADE DE PARTÍCULAS

Quando comparado com algoritmos que não variam o número de “população” das partículas, o algoritmo de morcegos consegue bons resultados, tanto em simulações quanto em ambiente real, onde o algoritmo proposto se mostrou mais eficiente que os demais em razão das suas características fundamentais.

Este primeiro experimento compara o desempenho dos métodos LBBA, FP e *Particle Swarm Optimization* (PSO) propostos no primeiro ambiente real. A população de partículas dos dois últimos algoritmos permanece invariável durante todo o experimento e a primeira localização deve ser necessariamente global, uma vez que o P3DX não conhece sua pose inicial. A pose inicial do sistema robótico e os respectivos caminhos realizados pelos diferentes algoritmos de localização são mostrados na Figura 31. Como esperado, os três algoritmos determinaram a mesma pose inicial para o robô móvel. A Tabela 3 mostra alguns resultados quantitativos para este primeiro experimento, onde ΔE representa o erro relativo da melhor posição indicada pelos algoritmos para a posição real do robô e σ é sua variância.

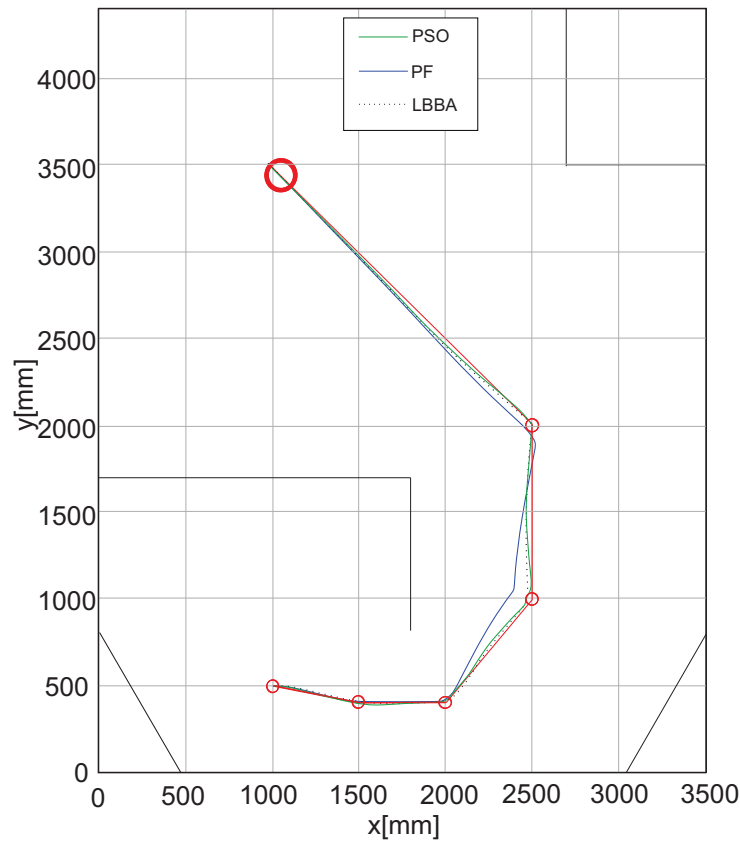
O LBBA e o PSO apresentaram melhor desempenho comparado ao FP em toda a análise estatística. Como critério de comparação, foi utilizado o intervalo de confiança de 99%, que representa a faixa de valores onde a posição real tem a probabilidade de 99% de ocorrência. Para a obtenção do $IC_{99\%}$ foi usada a seguinte relação [75]:

$$IC_{99\%} = 2,58 \cdot \frac{\sigma}{\sqrt{30}} \quad (5.4)$$

onde σ é o desvio padrão das amostras, e 30 o número de amostras utilizadas, sendo esse o número mínimo para relevância estatística.

A localização realizada pelo LBBA e pelo PSO é muito similar, o que também pode ser observado na Figura 31. A localização alcançada pelo FP teve problemas durante os movimentos rotacionais.

Figura 31 – Localização P3DX realizada pelos diferentes algoritmos de otimização e testada no primeiro ambiente.



Fonte: Elaborada pelo autor

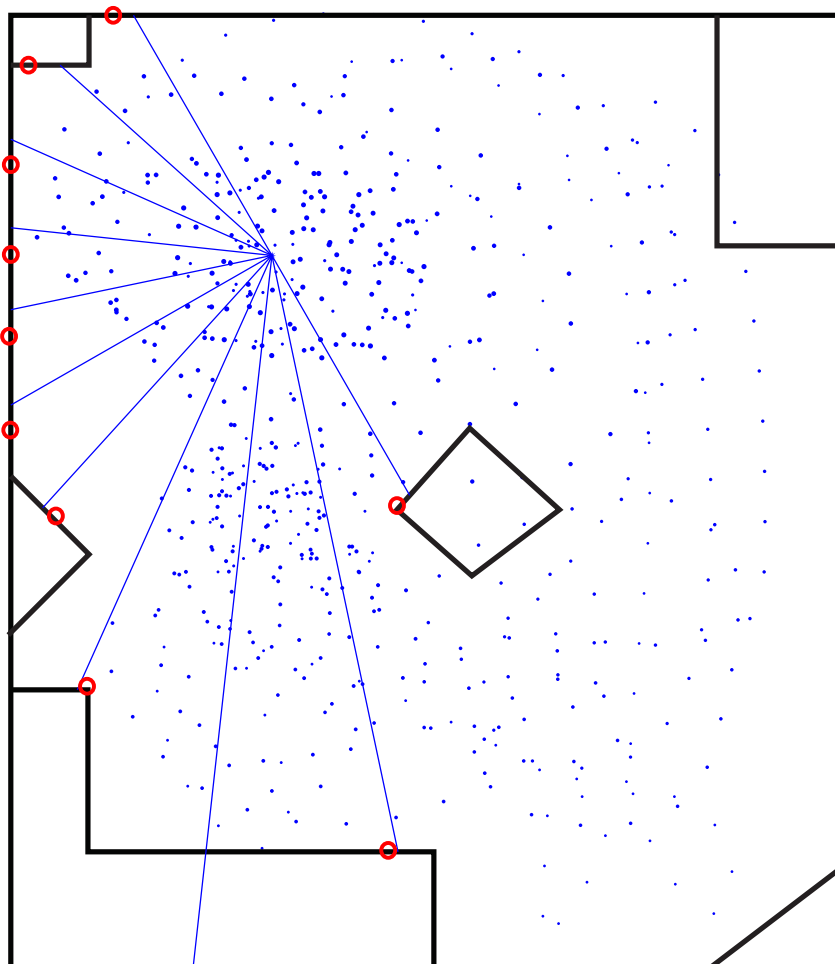
É possível perceber que o LBBA apresentou um desempenho superior BA, como mostra a Tabela 3, onde são apresentados a média dos erros de posição \bar{E} , o intervalo de confiança $IC_{99\%}$, e o desvio padrão σ de cada método analisado. A característica LBBA de múltiplos líderes influencia a busca de regiões potenciais. Observe que o LBBA trabalha com mais de um líder e essa abordagem aumenta a probabilidade de encontrar o global ideal. Isso é explicado pelo fato de que os múltiplos líderes disseminaram sua busca por mais posições, lidando satisfatoriamente com ambiguidades e possíveis medições ruins. De modo que o algoritmo proposto não tenha um aumento de parâmetros a serem ajustados, determinou-se por forma empírica que a cada 300 indivíduos teria-se 1 líder, ou seja, em um experimento com 1000 morcegos existiriam 3 alfas (correspondendo as 3 melhores posições).

Tabela 3 – Comparação de desempenho entre os algoritmos

Algoritmo	\bar{E}	$IC_{99\%}$	σ
LBBA	13,6446 mm	4,7003 mm	9,9785 mm
BA	14,9475 mm	4,8844 mm	10,3694 mm
PSO	15,8803 mm	4,9500 mm	10,5087 mm
FP	35,0151 mm	11,4335 mm	24,2728 mm

Em um mapa representado por *grids*, o LBBA obteve resultados ainda melhores (Figura 34). Como pode ser visto, a Figura 32 ilustra uma simulação usando as medidas de onze feixes de laser, dois mil morcegos e apenas três líderes. O algoritmo poderia localizar o robô com um erro de 2,3098 mm com apenas duas iterações e um tempo estimado de 27 segundos. O LBBA foi avaliado com o robô em diferentes situações. Testando a capacidade de localização global inicial, mudança de posição, redução de velocidade e localização quando o robô se move através de todos os *waypoints*. Finalmente, também foi testada a capacidade de localizar o P3DX que está se movendo aleatoriamente com uma velocidade de 0,4 m/s para verificar o desempenho da localização online durante um deslocamento contínuo, ou seja, se o robô está “perdido” e quer saber sua posição durante a missão.

Figura 32 – Experimento no segundo ambiente usando LBBA no mapa da grade. Os círculos vermelhos representam as medidas do sensor laser e a linha azul são as medições simuladas. Os pontos azuis representam os morcegos.



Fonte: Elaborada pelo autor

A primeira localização deve ser global porque o robô não conhece sua pose, logo se torna necessário o artifício de espalhar partículas por todo o mapa. A partir da localização global, o robô começa a realizar sua missão e, conforme se desloca, aumenta a probabilidade

de erros de odometria, representados na covariância crescente como mostrado na Equação 5.5. A localização local ocorre quando a covariância é considerada suficientemente grande a ponto de colocar em risco a integridade do robô durante a missão.

$$\Sigma = \begin{bmatrix} \delta_r |\Delta S_r| & 0 \\ 0 & \delta_l |\Delta S_l| \end{bmatrix}_k \quad (5.5)$$

Os experimentos realizados em simulações foram testados em dois ambientes reais (Figuras 33 e 34.), aos quais foram configurados para apresentar uma quantidade considerável de mínimos locais. Tal característica é importante para verificar se os algoritmos de otimização estão realmente filtrando corretamente a pose do robô e também, para verificar o desempenho dos algoritmo de inteligência artificial proposto sobre os demais algoritmos.

Figura 33 – Primeiro ambiente.



Fonte: Elaborada pelo autor

Figura 34 – Segundo ambiente.

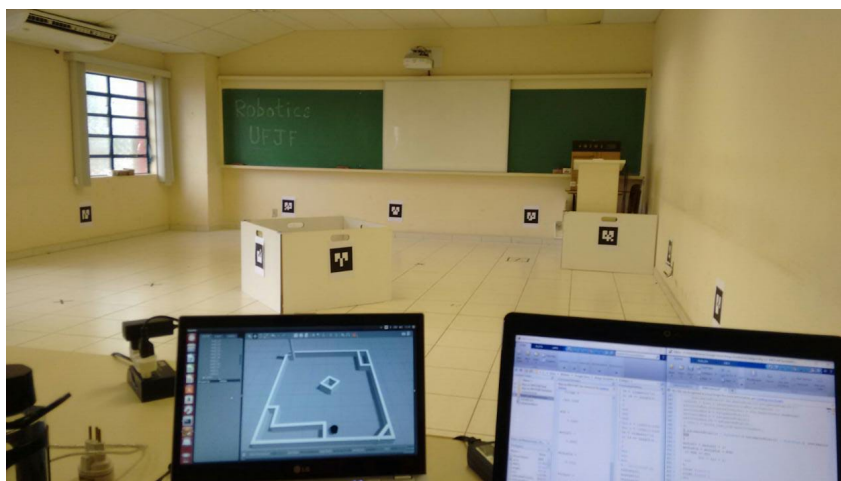


Fonte: Elaborada pelo autor

O caminho determinado para o robô autônomo realizar sua missão é indicado pelo usuário, através de *waypoints* que deveriam ser visitados e o P3DX de forma autônoma,

controlado por um controlador PID sintonizado com o método Twiddle [66], deveria percorrer todos os objetivos da forma mais precisa possível (Figura 35).

Figura 35 – Usuário informa ao programa quando os *waypoints* devem ser visitados pelo robô durante a realização da missão.

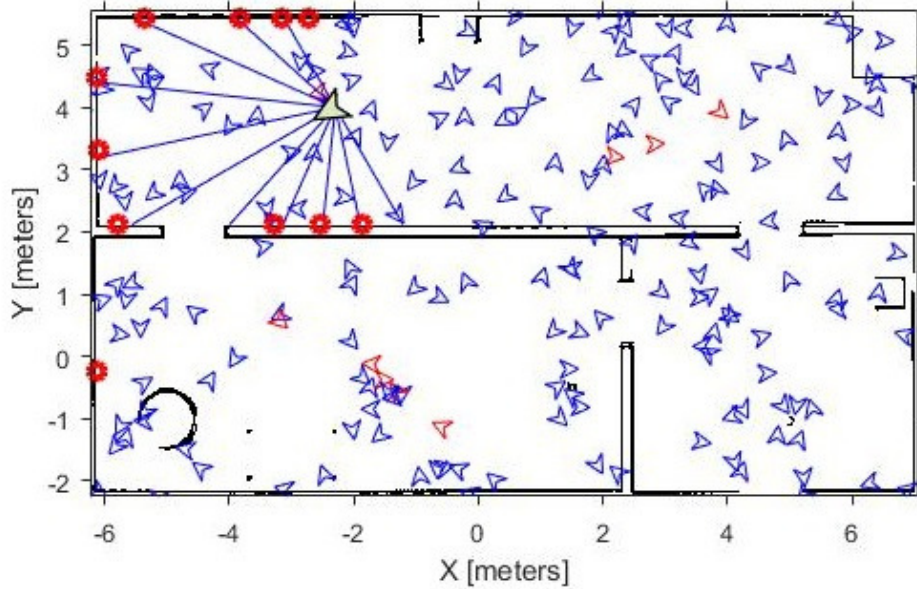


Fonte: Elaborada pelo autor

5.2.2 ALGORITMOS COM VARIAÇÃO DE QUANTIDADE DE PARTÍCULAS

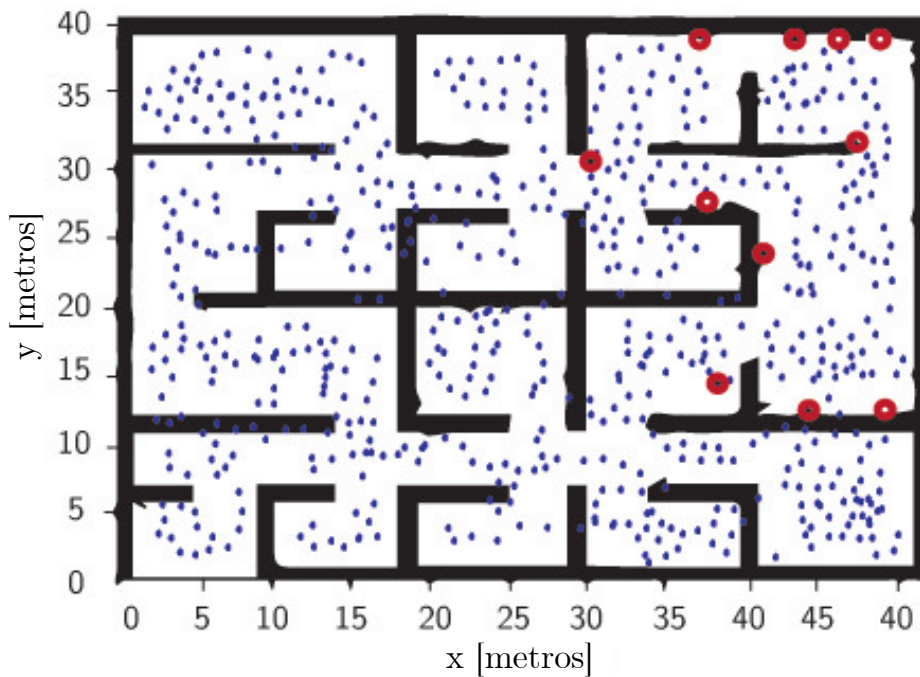
Para validar o LBBA proposto, mais experimentos simulados foram realizados utilizando a plataforma MATLAB[®] e o robô P3DX com a estrutura ROS para controle e gerenciamento do robô. Os experimentos foram configurados com 400 a 1600 partículas para localizar globalmente o robô de acordo com o tamanho e a complexidade do ambiente. O algoritmo proposto foi testado usando um banco de dados MATLAB[®]. Os resultados podem ser observados nas Figuras 36 e 37. Estes dois ambientes (isto é, simples e complexos) apresentam características e complexidades diferentes. Além disso, seus tamanhos mudam de doze para cinquenta metros. Os melhores morcegos não foram enfatizados nesta figura. A figura apresentaria uma poluição da informação caso os melhores morcegos fossem adicionados ao mapa. Portanto, a intenção era apresentar o ambiente do complexo mapa. A informação detalhada é dada na Tabela 6.

Figura 36 – Simulação do mapa simples com os melhores morcegos representados por pontos azuis e leituras de laser representadas por círculos vermelhos. A localização foi realizada em um mapa disponível no MATLAB®.



Fonte: Elaborada pelo autor

Figura 37 – Simulação do mapa complexo com os melhores morcegos e as melhores leituras a laser. Localização realizada em um mapa disponível no MATLAB®.



Fonte: Elaborada pelo autor

Durante os experimentos, o LBBA foi capaz de convergir em diferentes ambientes e condições com muitas e poucas partículas, enquanto AMCL não conseguiu. O AMCL foi capaz de manter uma alta taxa de resposta usando um grande número de partículas, mas

não mostrou resultados satisfatórios em situações críticas, como o LBBA. Observe que o LBBA também supera satisfatoriamente o BA padrão em ambos os mapas, apresentando menores erros e interações. Os resultados usando mapas simples e complexos são mostrados nas Tabelas 4 e 5.

Tabela 4 – Comparação entre algoritmos no mapa simples.

Variáveis	Mapa	Número de partículas			
		400	800	1200	1600
Erro Médio (%)	LBBA	0,013	0,012	0,011	0,011
	BA	0,72	0,700	0,700	0,039
	AMCL	0,28	0,279	0,003	0,020
Iterações	LBBA	8	6	5	4
	BA	500	500	500	354
	AMCL	500	500	46	43

Tabela 5 – Comparação entre algoritmos no mapa complexo.

Variáveis	Mapa	Número de partículas			
		400	800	1200	1600
Erro Médio (%)	LBBA	0,013	0,015	0,012	0,011
	BA	0,563	0,118	0,275	0,234
	AMCL	1	0,275	0,328	0,325
Iterações	LBBA	14	6	3	5
	BA	500	500	500	500
	AMCL	500	500	500	500

Um exemplo ideal de eficiência LBBA para determinar a localização com poucas partículas é mostrado no ambiente proposto do cenário 1. Apesar de a velocidade do AMCL chegar a 0,8 segundos no decorrer da otimização em alguns momentos, esta não foi capaz de encontrar o robô durante toda a busca da localização. O LBBA chegou a alcançar 3 segundos para executar a localização sem qualquer estratégia de adaptação (diminuição do número de indivíduos ao longo das iterações). O número máximo de iterações foi definido como 500 com 3% de tolerância a erros entre as medidas do robô real e o robô mais bem-simulado. O AMCL não conseguiu localizar o robô móvel usando menos partículas (20 unidades), como o LBBA. Este resultado é apresentado na Tabela 6. Percebe-se nesta tabela que o BA utiliza mais interações e tem um erro maior em comparação com o LBBA e o AMCL.

5.2.3 SEQUESTRO DO ROBÔ

Como mencionado anteriormente, a localização do robô é suscetível a diversos problemas, como detecção de imprecisão, determinação inicial de pose e sequestro. O sequestro robótico é uma situação em que um robô móvel bem localizado é inesperadamente

Tabela 6 – Comparação entre AMCL e LBBA no cenário 1 usando o P3DX.

		Número de partículas			
		400	800	1200	1600
Erro Médio (%)	LBBA	0,0238	0,0214	0,0223	0,0221
	BA	0,693	0,651	0,5631	0,1181
	AMCL	0,0433	0,0210	0,0302	0,0269
Iterações	LBBA	7	5	2	2
	BA	500	500	500	500
	AMCL	100	54	92	23

transferido para outro local. Neste problema, o robô móvel acredita estar em outro lugar no momento do sequestro [76]. Outras razões podem causar o mesmo efeito que uma falha do sensor de varredura a laser durante um período de tempo.

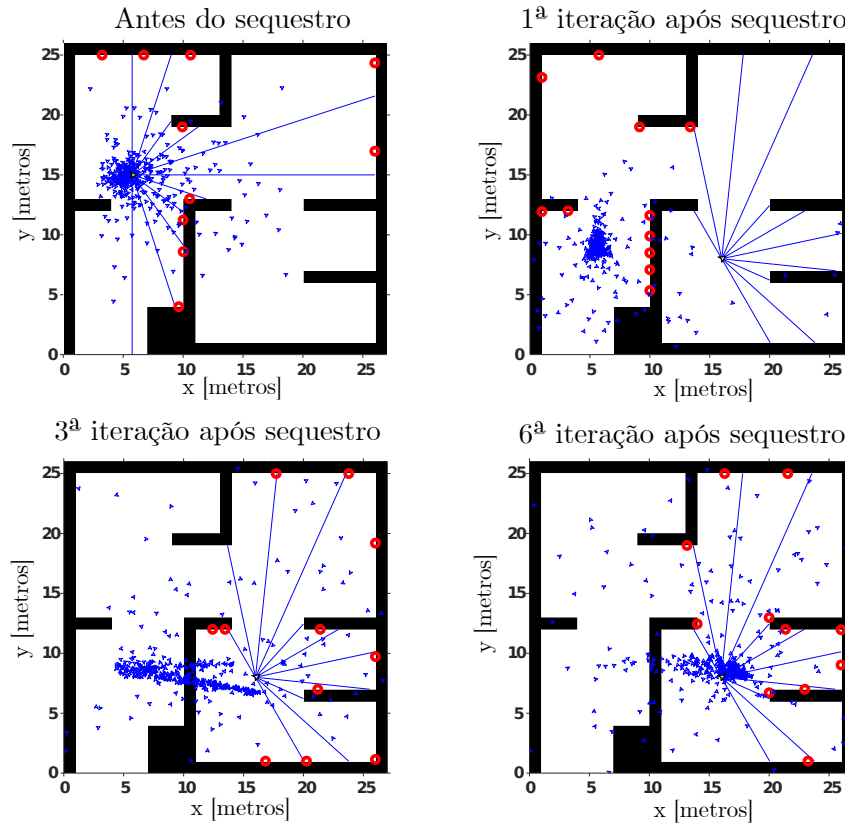
O problema do sequestro pode ser usado para avaliar a capacidade de recuperação de robôs móveis quando ocorre falha de localização. Assim, dados os resultados anteriores, este último caso é responsável por testar a habilidade do robô no problema de sequestro. O primeiro conjunto de resultados foi realizado usando a plataforma MATLAB[®], o robô começa a navegar no ponto X_i e o sistema robótico é deslocado instantaneamente para a posição X_s . Após essa posição alterada, o algoritmo exigirá poucas iterações para identificar a nova posição do robô e para convergir o robô para ele.

Inicialmente, o robô é colocado na posição inicial do mapa ($X_0 = (6, 15)$). Depois que o algoritmo encontra a posição, o robô é sequestrado. Então, o algoritmo LBBA é aplicado para encontrar a posição do novo robô. Esse comportamento é ilustrado na Figura 38. Note que antes do sequestro, os morcegos estão concentrados mais perto da posição do robô. Na primeira iteração, os morcegos começam a se mover na direção do robô. No entanto, quando o robô é deliberadamente colocado em uma sala diferente, há um erro devido a essa condição na primeira iteração. As iterações subsequentes mostram a convergência do algoritmo e a melhor evolução da posição estimada.

Este problema foi simulado para um número diferente de morcegos, conforme exposto anteriormente. A figura 39 mostra a posição estimada pelo algoritmo desde a posição inicial até um erro de 0,1 da verdade do terreno. Observe que a convergência não possui um caminho preferencial. Não é possível determinar na Figura 39 o comportamento do erro. A Figura 40 apresenta como a evolução da convergência varia de acordo com o número de partículas. Como esperado, o número de partículas está correlacionado com o número de iterações necessárias para determinar a posição correta do robô. Além disso, o erro é rapidamente reduzido nas primeiras iterações assim que o algoritmo é iniciado. É importante mencionar que essa figura representa apenas uma execução do algoritmo e o número de iterações pode mudar um pouco de uma para outra.

Os resultados seguintes foram realizados no *software* Gazebo e gerenciados pelo

Figura 38 – Comportamento do algoritmo LBBA no problema de sequestro do robô.



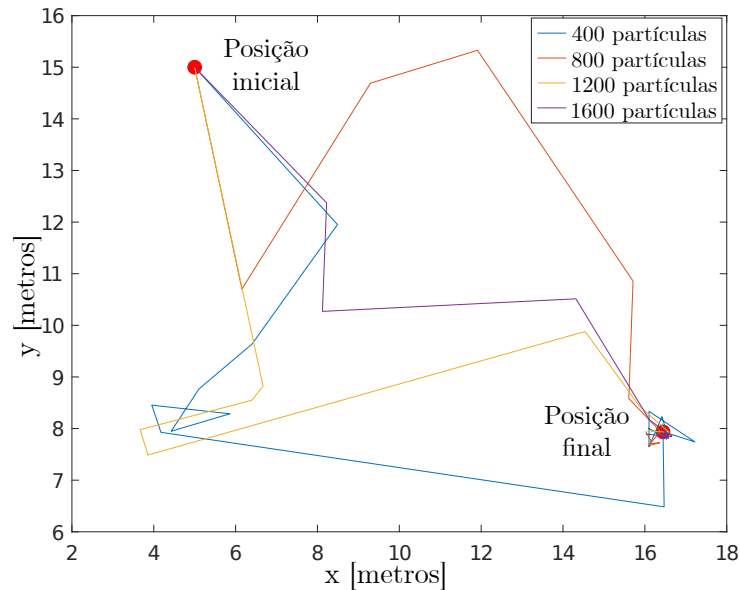
Fonte: Elaborada pelo autor

framework ROS. Este experimento utilizou 2000 partículas para obter a convergência com o menor número de possíveis iterações. A velocidade do robô foi escolhida para permitir uma estimativa precisa da posição pelo algoritmo. Neste experimento, o robô executa um caminho em torno de um obstáculo mostrado na Figura 32. Enquanto executa seu caminho, o robô é subitamente movido para outra posição e orientação. A figura 41 apresenta o caminho do robô em vermelho e o local onde o sequestro ocorreu como um X vermelho, bem como o ponto em que o robô retornou.

A linha tracejada azul representa a posição estimada do LBBA. A mudança da posição é instantânea. Nesse sentido, não há caminho representado após o X marcado. O tracejado azul também se mostra como descontinuidade uma vez que isso nos permite perceber a evolução do erro de posição estimado. Observe que, após o problema de sequestro de robô, o algoritmo LBBA tenta encontrar a posição correta e a convergência rápida é notada. A Figura 42 mostra o erro de acordo com a iteração. Cada iteração contém 3 etapas do algoritmo LBBA para uma determinada posição. O erro de posição foi calculado usando a distância euclidiana para a posição real. Nota-se nesta figura que após o sequestro, os picos de erro aumentam. No entanto, ele é reduzido rapidamente na próxima iteração.

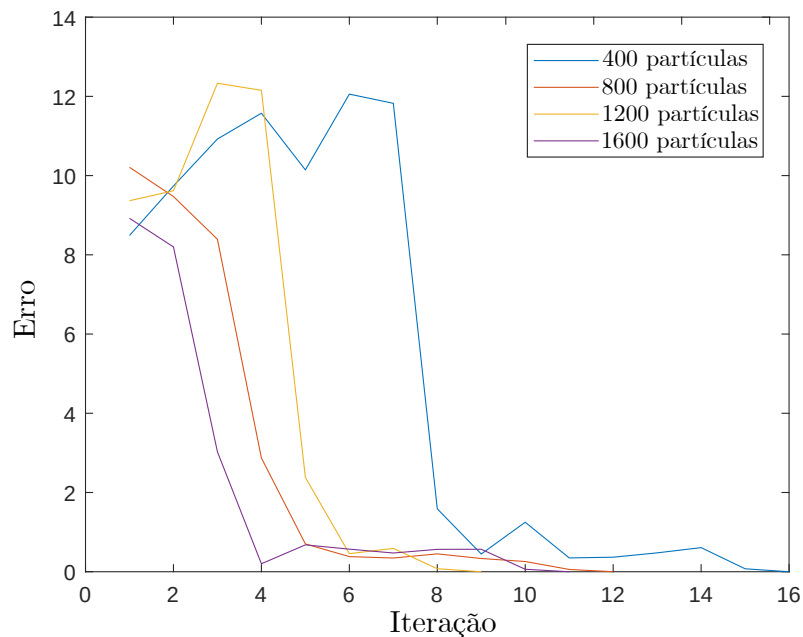
Isso mostra que o algoritmo pode determinar a posição do robô com precisão rápida

Figura 39 – Evolução da posição estimada para diferentes números de partículas na simulação de sequestro do robô.



Fonte: Elaborada pelo autor

Figura 40 – Evolução do erro para diferentes números de partículas na simulação de sequestro do robô.

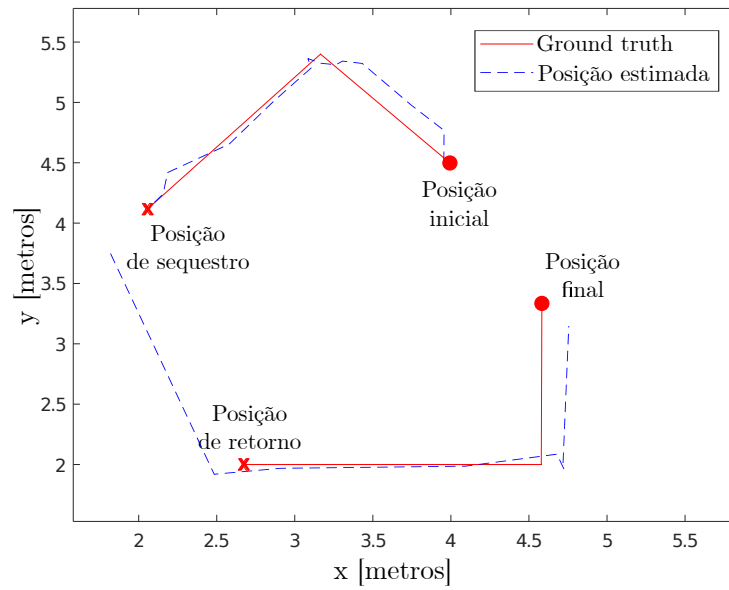


Fonte: Elaborada pelo autor

e logo após um evento de falha. A partir das simulações e experimentos, o algoritmo mostrou robustez aos erros limitados. Além disso, a partir de 39, é possível notar que o algoritmo não converge prematuramente e leva apenas alguns passos para recuperar a posição correta.

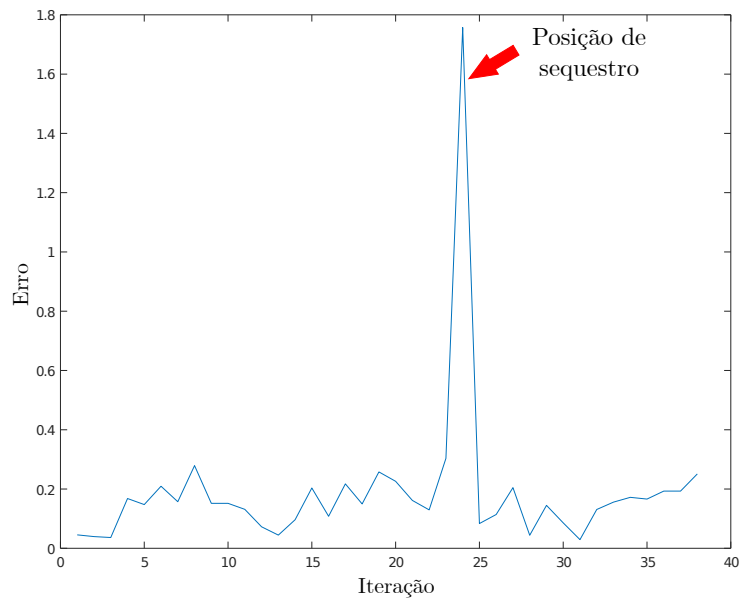
A análise anterior também foi realizada com o robô real e os resultados podem ser vistos na Figura 43 e Figura 44. Note que em determinados momentos, o algoritmo

Figura 41 – Comportamento do robô no experimento de sequestro.



Fonte: Elaborada pelo autor

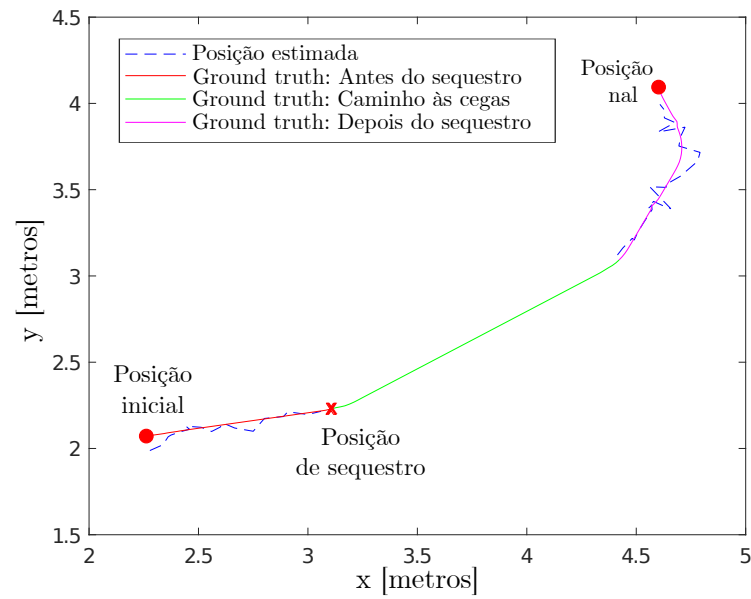
Figura 42 – Evolução do erro de acordo com o passar do tempo.



Fonte: Elaborada pelo autor

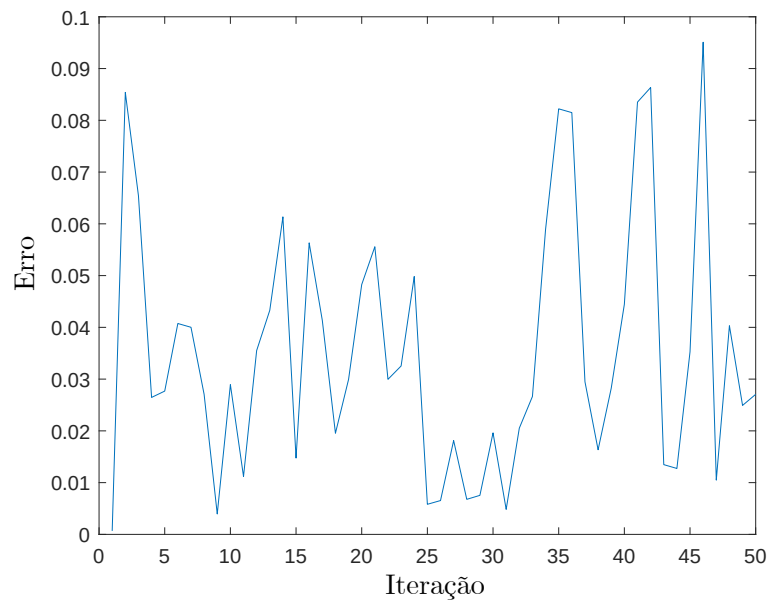
encontra uma posição próxima ao P3DX com um certo valor de erro indesejado. No entanto, isso é explicado pelo fato de que o mapa real não pode ser totalmente idêntico ao mapa usado pelo algoritmo e as ambiguidades presentes durante a missão implicam em maior dificuldade de localização.

Figura 43 – Comportamento do P3DX no cenário de sequestro.



Fonte: Elaborada pelo autor

Figura 44 – Evolução do erro do P3DX com o passar das iterações no cenário de sequestro.



Fonte: Elaborada pelo autor

5.3 Conclusão

Este Capítulo apresentou os testes comparativos do algoritmo proposto em funções multimodais, no intuito de investigar o seu desempenho perante a variantes da literatura. Os algoritmos foram implementados segundo uma mesma filosofia de programação, e testados estatisticamente em igualdade semente e condições iniciais. O LBBA se mostrou eficaz como algoritmo de otimização, indicando uma melhora de desempenho após as

modificações propostas no BA, e eventualmente, um comportamento similar ao algoritmo clássico nos cenários que não obteve a melhor performance.

Após autenticar a melhora do algoritmo proposto em relação ao algoritmo clássico, através dos testes estatísticos nas funções multimodais de otimização, o próximo passo foi aplicar o LBBA no problema de localização de robôs. Para isso foram realizados testes simulados com o objetivo de comparar o comportamento do novo algoritmo frente aos demais (BA, PSO e FP), agora em um problema de localização de bases móveis. Diante de resultados promissores, testes com robô real foram realizados em mapas desenvolvidos dentro da Universidade Federal de Juiz de Fora, em diferentes condições, por exemplo, dentro de salas fechadas e corredores propícios a interferência luminosa nos sensores.

Os testes indicaram novamente uma vantagem do algoritmo para com os demais, logo, buscou-se comparar o mesmo com um algoritmo de localização que utiliza-se da estratégia de variação do número de partículas (AMCL), onde o LBBA também obteve um bom desempenho em mapas com elevado número de ambiguidades (possíveis ótimos locais).

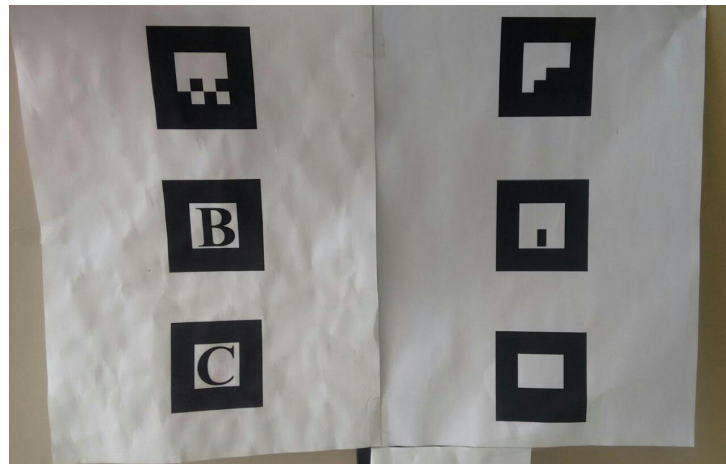
Por fim, ensaiou-se uma situação onde o P3DX se encontrava em um problema de robô sequestrado, de modo que o algoritmo baseado em líderes tivesse que se localizar após uma perda de comunicação com os sensores, ou uma mudança abrupta de posição ao longo de uma missão. Frente a esse cenário, o algoritmo proposto obteve sucesso em relocalizar o robô após o evento de sequestro, convergindo corretamente para a nova posição.

6 TRABALHOS FUTUROS

O trabalho apresentado mostrou resultados esperados e satisfatórios no proposto à pesquisa. No entanto, existem possibilidades a serem exploradas em futuros experimentos de modo a melhorar a acurácia do algoritmo, ou então aliviar o peso computacional do mesmo. Este capítulo descreve algumas das abordagens possíveis.

O sensor utilizado neste trabalho é o *laser rangefinder* do fabricante SICK, o qual fornece pontos bidimensionais de distância com relação ao mesmo. A extração de características (*features*) é uma etapa que tem um custo computacional, bem como é sujeita a possibilidades de ambiguidades que podem vir a tornar uma rodada de leitura de sensores inutilizada. Uma próxima etapa para este trabalho seria a utilização de *landmarks* visuais, como da Figura 45, conforme se encontra, por exemplo, no trabalho de Chen (2014). Diversas bibliotecas disponíveis nos principais *softwares* de robótica atuais oferecem suporte para esse tipo de atividade. Uma das vantagens do *landmark* visual é a eliminação das ambiguidades presentes no laser, bem como na precisão na informação.

Figura 45 – Marcadores - *Landmarks*



Fonte: Elaborada pelo autor

Para o caso de se continuar utilizando o *laser rangefinder* como fonte de leituras do ambiente, é necessário adotar estratégias mais robustas na redução das ambiguidades. Para isso, uma possível atitude seria remodelar os mapas como sendo segmentos de retas e calcular o local geométrico dos cruzamentos entre as retas do sensor e do ambiente. Ainda assim, uma política de seleção dessas características deveria ser implementada. Ou ainda, a fusão entre os sensores laser e câmera, uma vez que podem existir situações em que a luminosidade local não favoreça o uso de *landmarks* visuais.

É importante lembrar que o método desenvolvido neste trabalho é agnóstico com relação ao tipo dos *landmarks* ou *features* utilizados, mas é sempre importante ressaltar que a qualidade desses influencia significativamente na rapidez de convergência ou mesmo

na acurácia da posterior final, não somente deste mas de todos os métodos de filtragem.

Ainda com relação ao emprego do *laser rangefinder* neste trabalho, ressalta-se que não foram empregadas todas as medidas bidimensionais fornecidas pelo sensor, no caso, 181 medidas. Esta escolha refletiu a decisão de dificultar a localização para o método por meio do empobrecimento da informação disponível. Uma vez que a informação é pobre, os métodos tendem, geralmente, a levar um tempo maior para a convergência, pois as ambiguidades e ruídos passam a predominar. Mesmo com este desafio, o método prevaleceu e a localização obteve sucesso, conforme o capítulo de resultados demonstrou. Como próximos trabalhos, pode-se utilizar todo o potencial do sensor, oferecendo *features* mais ricas em informação, possivelmente melhorando ainda mais a acurácia dos resultados.

Além de um método de localização, o algoritmo proposto (LBBA) pode ser evoluído para funcionar como um método SLAM, o qual é uma técnica para construção e atualização de mapas sem o conhecimento a priori acerca do ambiente, o qual será utilizado concomitantemente pelo robô para corrigir sua localização, a medida que a confiança em sua localização vai se deteriorando. Para este fim, sugere-se a inserção de um Filtro de Kalman Estendido (EKF) em cada uma das partículas utilizadas, formando um SLAM de morcegos. Como o algoritmo de morcegos é baseado em parâmetros, o *Bat-Simultaneous Localization And Mapping* (BatSLAM) precisaria de ajustes dos mesmos para convergir, ou seja, calibrar a amplitude e a frequência dos possíveis ambientes diferentes, como mapas esparsos, ambientes extensos e pequenos, missões autônomas, controle compartilhado, entre outros. Esta seria, portanto, uma extensão interessante do algoritmo apresentado.

Na Figura 46 é proposto um cenário onde se tem um robô que vai construindo um mapa com o auxílio de um sensor *rangefinder* (laser). Enquanto constrói o mapa, o robô identifica os *landmarks*, entretanto, durante sua navegação a hodiometria vai se deteriorando e prejudicando sua localização. Ao retornar a um ponto já mapeado, o P3DX identificaria a distância da posição onde acredita estar (hodiometria) até a posição de onde o *landmark* deveria estar não é coerente com as informações que está recebendo do sensor, logo iria utiliza-se do algoritmo para corrigir sua posição e conseqüentemente o mapa.

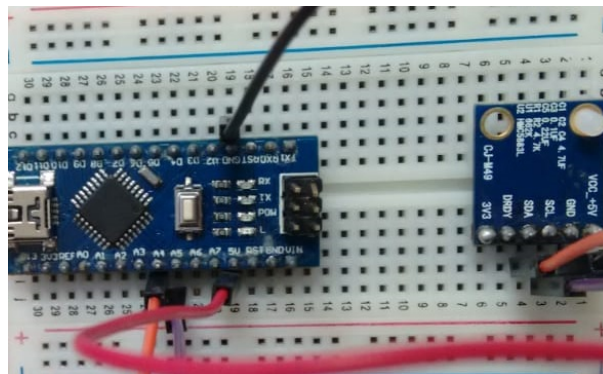
Figura 46 – *Landmarks* ao longo do mapa para serem utilizados na detecção.



Fonte: Elaborada pelo autor

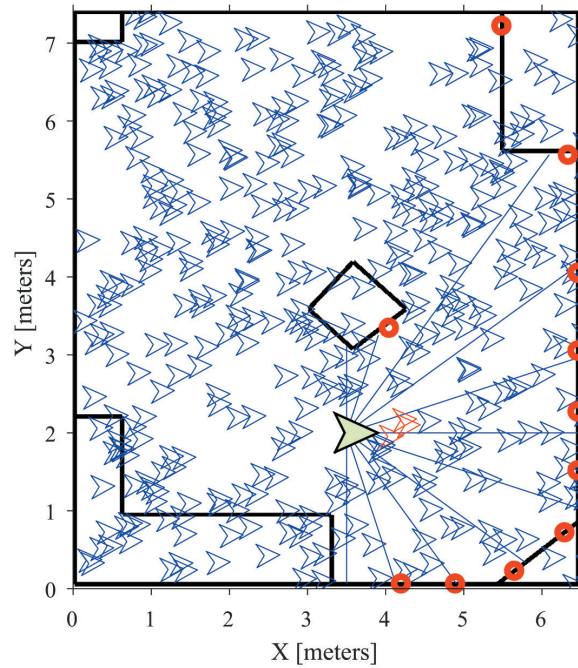
Outro experimento iniciado e que pode ser finalizado é a utilização de uma bússola (Figura 47), para auxiliar o chute inicial do robô em uma provável primeira localização global (Figura 48). A adição desse novo sensor ajudaria eliminar também algumas ambiguidades de leituras existentes nos cenários, visto que seria mais uma informação utilizada na simulação das partículas. Em testes simulados houve um ganho de velocidade de convergência em relação a utilização de apenas o sensor *rangefinder*, nos testes reais o protótipo chegou a sofrer interferência em alguns momentos, prejudicando a localização durante as iterações.

Figura 47 – Protótipo de um sensor Bússola utilizado em testes.



Fonte: Elaborada pelo autor

Figura 48 – Localização Global utilizando o sensor bussola proposto durante a simulação.



Fonte: Elaborada pelo autor

Abaixo na Figura 49 está o robô utilizado para os testes reais, nele estão contidos uma Raspberry Pi 3 responsável por fazer a comunicação dos comandos de controle de velocidade e aquisição da dados dos sensores, uma bateria para alimentação da Raspberry, o protótipo da bussola utilizando arduino, e o sensor laser Sick.

Figura 49 – P3DX utilizado para testes, contendo o protótipo do sensor bussola.



Fonte: Elaborada pelo autor

7 CONCLUSÃO

Este trabalho apresentou uma nova metodologia de localização de robôs móveis. A modificação proposta no algoritmo padrão de morcegos é baseado na utilização de líderes, LBBA, durante as iterações de busca, as quais tem como objetivo a minimização da diferença entre as leituras dos sensores reais com as leituras dos sensores simulados por cada morcego. Para comparar o desempenho do algoritmo proposto, foi executada uma análise estatística com funções multimodais da literatura – Rastrigin, Eggcrate e Ackley. Nestes testes, o número de iterações médio de cada algoritmo em função do número de partículas foi analisado. O desempenho de todos os métodos foi semelhante para um número suficientemente grande partículas. A partir dos dados foi possível notar que o algoritmo apresentou elevado decréscimo do número de iterações com pequena variação das partículas, desempenho também observado no BA. Em todos os cenários, o desempenho do método proposto foi superior ou similar ao melhor método, demonstrando que o método é competitivo perante aos apresentados na literatura.

Dados os resultados promissores, foi realizado uma comparação de algoritmos para o problema de localização de um robô P3DX – os algoritmos de Filtro de Partícula, Enxame de Partículas, além do algoritmo adaptativo de monte carlo (AMCL). A utilização do LBBA trouxe vantagens ao processo de otimização, com aumento da velocidade de convergência em determinadas situações, erros iguais ou menores ao algoritmos comparados, e menor ocorrência de falsos positivos.

Dessemelhante ao BA, o LBBA tem como estratégia de busca a regionalização de bons locais de alimento (mínimos ou máximos locais) que são correspondentes a colônias de morcegos. Portanto, os demais indivíduos não vão ser influenciados pelo melhor morcego de uma geração, e sim pelo número de líderes existentes em toda população. Os parâmetros de entrada para este novo algoritmo proposto são baseados nos parâmetros propostos na literatura, com o acréscimo do número de líderes presente. Objetivando desenvolver uma proposta genérica capaz de entregar um algoritmo sem o aumento de parâmetros a serem ajustados, determinou-se por forma empírica que a cada 300 indivíduos teria-se 1 líder, ou seja, em um experimento com 1000 morcegos existiriam 3 alfas (correspondendo as 3 melhores posições).

Diferentes experimentos foram realizados, visto a grande variedade de cenários possíveis de testes. O robô diferencial, P3DX, foi utilizado em todos estes testes, inclusive a modelagem do modelo cinemático utilizado em simulações foi baseado em suas estruturas e limitações, respeitando inclusive a limitação de alcance do sensor. O LBBA foi aplicado em um modelo simplificado de mapa, contendo uma pequena área de busca, e também foi testado em mapas com tamanhos significativos, contendo locais ambíguos que dificultavam a localização por oferecerem medidas muito próximas ao local verdadeiro em que o robô

estava. Os testes foram capazes de demonstrar o bom desempenho do novo algoritmo, como esperado o LBBA foi capaz de localizar o robô real de modo mais eficiente que o BA, tendo em vista a possibilidade de diminuição do número de partículas, e a capacidade de sair de situações de falso positivos.

A metodologia proposta obteve resultados contundentes mesmo na presença de ruído dos sensores do robô real, e na dificuldade de conseguir um mapa simulado fiel ao mapa real nos testes práticos. O que dificultou obter uma precisão melhor da localização em alguns cenários propostos. Haja vista que o algoritmo proposto pode enfrentar situações em que se exija uma quantidade elevada de partículas e prejudique a velocidade convergência da otimização, em algumas situações durante os testes foi necessário utilizar menos medidas de *lasers* para que não ocorresse uma explosão combinacional e prejudicasse a eficiência do mesmo.

Quando comparado com um algoritmo adaptativo, a velocidade de convergência não foi superior, entretanto, em alguns cenários se demonstrou tão eficiente quanto. Esta característica foi observada tanto para localização global (quando não se tem ideia da posição do robô em todo o ambiente possível), quanto para a localização local durante o prosseguimento de uma missão. Ademais o algoritmo de morcegos baseado em líderes foi colocado a prova em uma situação de sequestro de robô, cenário em que por algum motivo não se tem informação dos sensores do robô durante a missão, e obteve resultados interessantes, demonstrando um aspecto de robustez e promissor em futuras aplicações.

REFERÊNCIAS

- [1] SCHWAB, K. A quarta revolução industrial. *São Paulo: Edipro*, v. 27, 2016.
- [2] FORLIZZI, J.; DISALVO, C. Service robots in the domestic environment: a study of the roomba vacuum in the home. In: ACM. *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*. [S.l.], 2006. p. 258–265.
- [3] BROEKENS, J. et al. Assistive social robots in elderly care: a review. *Gerontechnology*, v. 8, n. 2, p. 94–103, 2009.
- [4] MUBIN, O. et al. A review of the applicability of robots in education. *Journal of Technology in Education and Learning*, v. 1, p. 209–0015, 2013.
- [5] SICILIANO, B. et al. *Robotics: modelling, planning and control, ser. Advanced Textbooks in Control and Signal Processing*. [S.l.]: Springer, 2009.
- [6] ROMERO, R. A. F. et al. *Robotica Movei*. [S.l.]: LTC, 2014.
- [7] AUTORES, V. *Research Report: Investing in Robotics*. 2019. URL: <http://www.roboticsbusinessreview.com/download/12684/>. Acessado 13-04-2019.
- [8] SOOHOO, M. S. S. *Worldwide Spending on Robotics Systems and Drones Forecast to Total 115.7 Billion in 2019, According to New IDC Spending Guide*. 2018. Disponível em: <<https://www.idc.com/getdoc.jsp?containerId=prUS44505618>>. Acesso em: 13-04-2019.
- [9] Y., C. *Mercado de robôs está em alta e deve crescer 21 na América Latina*. 2019. Disponível em: <<https://www.tecmundo.com.br/mercado/140311-mercado-robos-alta-deve-crescer-21-america-latina.htm>>. Acesso em: 13-04-2019.
- [10] QUIGLEY, M. et al. Ros: an open-source robot operating system. In: KOBE. *ICRA workshop on open source software*. [S.l.], 2009. v. 3, n. 3.2, p. 5.
- [11] SOBREIRA, H. et al. Map-matching algorithms for robot self-localization: A comparison between perfect match, iterative closest point and normal distributions transform. *Journal of Intelligent & Robotic Systems*, Springer, p. 1–14, 2018.
- [12] DELLAERT, F. et al. Monte carlo localization for mobile robots. In: *ICRA*. [S.l.: s.n.], 1999. v. 2, p. 1322–1328.
- [13] HAYKIN, S. *Kalman filtering and neural networks*. [S.l.]: John Wiley & Sons, 2004. v. 47.
- [14] THRUN, S. Particle filters in robotics. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*. [S.l.], 2002. p. 511–518.
- [15] CHOSET, H. M. *Principles of robot motion: theory, algorithms, and implementation*. [S.l.]: MIT press, 2005.
- [16] ARAÚJO, R.; GOUVEIA, G.; SANTOS, N. Mobile robot localization using a fuzzy art world model. In: IEEE. *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*. [S.l.], 2002. v. 3, p. 2409–2414.

- [17] LIANG, Z.; MA, X.; DAI, X. Extended monte carlo algorithm to collaborate distributed sensors for mobile robot localization. In: IEEE. *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*. [S.l.], 2007. p. 1647–1652.
- [18] KIA, S. S.; ROUNDS, S.; MARTÍNEZ, S. Cooperative localization under message dropouts via a partially decentralized ekf scheme. In: IEEE. *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. [S.l.], 2015. p. 5977–5982.
- [19] DJEHAICH, M. et al. Slam-icp with a boolean method applied on a car-like robot. In: IEEE. *Programming and Systems (ISPS), 2013 11th International Symposium on*. [S.l.], 2013. p. 116–121.
- [20] YANG, X.-S. *Nature inspired metaheuristic algorithms*. [S.l.]: Luniver press, 2010.
- [21] CARPENTER, G. A. et al. Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on neural networks*, v. 3, n. 5, p. 698–713, 1992.
- [22] KOHONEN, T. The self-organizing map. *Proceedings of the IEEE*, IEEE, v. 78, n. 9, p. 1464–1480, 1990.
- [23] GARDNER, M. W.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, Elsevier, v. 32, n. 14-15, p. 2627–2636, 1998.
- [24] YINGWEI, L.; SUNDARARAJAN, N.; SARATCHANDRAN, P. Performance evaluation of a sequential minimal radial basis function (rbf) neural network learning algorithm. *IEEE Transactions on neural networks*, IEEE, v. 9, n. 2, p. 308–318, 1998.
- [25] DAVIS, L. *Handbook of genetic algorithms*. CUMINCAD, 1991.
- [26] BACK, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. [S.l.]: Oxford university press, 1996.
- [27] KINNEAR, K. E. et al. *Advances in genetic programming*. [S.l.]: MIT press, 1999. v. 3.
- [28] AFSHINMANESH, F.; MARANDI, A.; RAHIMI-KIAN, A. A novel binary particle swarm optimization method using artificial immune system. In: IEEE. *EUROCON 2005-The International Conference on "Computer as a Tool"*. [S.l.], 2005. v. 1, p. 217–220.
- [29] KHILWANI, N. et al. Fast clonal algorithm. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 21, n. 1, p. 106–128, 2008.
- [30] YANG, X.-S. Bat algorithm: literature review and applications. *arXiv preprint arXiv:1308.3900*, 2013.
- [31] KENNEDY, J. Particle swarm optimization. In: *Encyclopedia of machine learning*. [S.l.]: Springer, 2011. p. 760–766.
- [32] DJURIC, P. M. et al. Particle filtering. *IEEE signal processing magazine*, IEEE, v. 20, n. 5, p. 19–38, 2003.

- [33] ZHANG, L.; ZAPATA, R.; LEPINAY, P. Self-adaptive monte carlo localization for mobile robots using range finders. *Robotica*, Cambridge University Press, v. 30, n. 2, p. 229–244, 2012.
- [34] PAPPA, G. L. Algoritmos bio-inspirados. 2017. Disponível em: <<http://homepages.dcc.ufmg.br/~glpappa/cverao/CursoVerao-Parte1.pdf>>.
- [35] JONES, G.; TEELING, E. C. The evolution of echolocation in bats. *Trends in Ecology & Evolution*, Elsevier, v. 21, n. 3, p. 149–156, 2006.
- [36] YANG, X.-S. A new metaheuristic bat-inspired algorithm, in: Nature inspired cooperative strategies for optimization. In: *Nature Inspired Cooperative Strategies for Optimization*. [S.l.: s.n.], 2010. p. 65–74.
- [37] JIANG, Z. Discrete bat algorithm for traveling salesman problem. In: IEEE. *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*. [S.l.], 2016. p. 343–347.
- [38] KUMAR, G. R. et al. Multi-objective optimal economic emission power dispatch using bat algorithm. In: IEEE. *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*. [S.l.], 2017. p. 1–5.
- [39] SINGH, D.; SALGOTRA, R.; SINGH, U. A novel modified bat algorithm for global optimization. In: IEEE. *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. [S.l.], 2017. p. 1–5.
- [40] AZLAN, N.; YAHYA, N. Modified adaptive bats sonar algorithm with doppler effect mechanism for solving single objective unconstrained optimization problems. In: IEEE. *2019 IEEE 15th International Colloquium on Signal Processing & Its Applications (CSPA)*. [S.l.], 2019. p. 27–30.
- [41] GAO, M.-L. et al. A bat-inspired particle filter for visual tracking. In: IEEE. *2016 Chinese Control and Decision Conference (CCDC)*. [S.l.], 2016. p. 3810–3815.
- [42] SRIDEVIPONMALAR, P.; HARIKRISHNAN, R. et al. Bat-firefly localization algorithm for wireless sensor networks. In: IEEE. *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICIC)*. [S.l.], 2017. p. 1–4.
- [43] NETO, W. A. et al. Mobile robot localization based on the novel leader-based bat algorithm. *Journal of Control, Automation and Electrical Systems*, Springer, p. 1–10, 2019.
- [44] ZHI, L.; XUESONG, M. Navigation and control system of mobile robot based on ros. In: IEEE. *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. [S.l.], 2018. p. 368–372.
- [45] SHIM, J. H.; CHO, Y. I. A visual localization technique for unmanned ground and aerial robots. In: IEEE. *2017 First IEEE International Conference on Robotic Computing (IRC)*. [S.l.], 2017. p. 399–403.
- [46] GRIMSON, W.; LOZANO-PEREZ, T. Recognition and localization of overlapping parts from sparse data in two and three dimensions. In: IEEE. *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*. [S.l.], 1985. v. 2, p. 61–66.

- [47] GRIMSON, W. E. L. The combinatorics of local constraints in model-based recognition and localization from sparse data. *Journal of the ACM (JACM)*, ACM, v. 33, n. 4, p. 658–686, 1986.
- [48] DRUMHELLER, M. Mobile robot localization using sonar. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, n. 2, p. 325–332, 1987.
- [49] PARK, S.; LEE, G. Mapping and localization of cooperative robots by ros and slam in unknown working area. In: IEEE. *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. [S.l.], 2017. p. 858–861.
- [50] ATIYA, S.; HAGER, G. D. Real time vision based robot localization. *IEEE Transactions on Robotics and Automation*, IEEE, v. 9, n. 6, p. 785–800, 1993.
- [51] WOO, J. et al. Localization of mobile robot using particle filter. In: IEEE. *SICE-ICASE, 2006. International Joint Conference*. [S.l.], 2006. p. 3031–3034.
- [52] FOX, D. et al. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, v. 1999, n. 343-349, p. 2–2, 1999.
- [53] ZHANG, B.; LIU, J.; CHEN, H. Amcl based map fusion for multi-robot slam with heterogenous sensors. In: IEEE. *Information and Automation (ICIA), 2013 IEEE International Conference on*. [S.l.], 2013. p. 822–827.
- [54] LEE, Y.-C.; PARK, B.; PARK, S. Coarse-to-fine robot localization method using radio fingerprint and particle filter. In: IEEE. *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*. [S.l.], 2014. p. 290–296.
- [55] MONTEMERLO, M. et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In: *Aaai/iaai*. [S.l.: s.n.], 2002. p. 593–598.
- [56] TURNOWICZ, M. R.; CHENG, Y. Gaussian mixture model based high dimensional slam utilizing sparse grid quadrature. In: IEEE. *American Control Conference (ACC), 2014*. [S.l.], 2014. p. 1102–1107.
- [57] SANGUINO, T. d. J. M.; GÓMEZ, F. P. Toward simple strategy for optimal tracking and localization of robots with adaptive particle filtering. *IEEE/ASME Transactions on Mechatronics*, IEEE, v. 21, n. 6, p. 2793–2804, 2016.
- [58] CHEN, J.-H.; LUM, K.-Y. Simultaneous localization and mapping based on particle filter for sparse environment. In: IEEE. *Mechatronics and Control (ICMC), 2014 International Conference on*. [S.l.], 2014. p. 1869–1873.
- [59] JIANG, R. et al. Geometric map-assisted localization for mobile robots based on uniform-gaussian distribution. *IEEE Robotics and Automation Letters*, IEEE, v. 2, n. 2, p. 789–795, 2017.
- [60] STEIN, F.; MEDIONI, G. Map based localization using the panoramic horizon. *IEEE Transactions on Robotics and Automation*, IEEE, v. 11, n. 6, p. 892–896, 1995.
- [61] WU, B.-F. et al. A relative-discriminative-histogram-of-oriented-gradients-based particle filter approach to vehicle occlusion handling and tracking. *IEEE Transactions on Industrial Electronics*, IEEE, v. 61, n. 8, p. 4228–4237, 2014.

- [62] NGUYEN, Q.-H. et al. A visual slam system on mobile robot supporting localization services to visually impaired people. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2014. p. 716–729.
- [63] XUHUI, Z.; RUNLIN, D.; YONGWEI, L. Vr-based remote control system for rescue detection robot in coal mine. In: IEEE. *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. [S.l.], 2017. p. 863–867.
- [64] GUO, T. et al. Research on localization method of mine rescue robot. In: IEEE. *2010 2nd International Conference on Industrial and Information Systems*. [S.l.], 2010. v. 2, p. 503–506.
- [65] THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic robotics*. [S.l.]: MIT press, 2005.
- [66] THRUN, S. Probabilistic robotics. *Communications of the ACM*, ACM, v. 45, n. 3, p. 52–57, 2002.
- [67] LI, T.; SUN, S.; DUAN, J. Monte carlo localization for mobile robot using adaptive particle merging and splitting technique. In: IEEE. *The 2010 IEEE International Conference on Information and Automation*. [S.l.], 2010. p. 1913–1918.
- [68] FOX, D. Kld-sampling: Adaptive particle filters. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2002. p. 713–720.
- [69] FOX, D. Adapting the sample size in particle filters through kld-sampling. *The international Journal of robotics research*, SAGE Publications, v. 22, n. 12, p. 985–1003, 2003.
- [70] FURTADO, L. et al. Bat search algorithm aplicado na localizacao de robos moveis. In: *Simposio Brasileiro de Automacao Inteligente*. [S.l.: s.n.], 2015.
- [71] ACKERMAN, E.; GUIZZO, E. *Wizards of ROS: Willow Garage and the Making of the Robot Operating System*. 2017. Disponível em: <<https://spectrum.ieee.org/automaton/robotics/robotics-software/wizards-of-ros-willow-garage-and-the-making-of-the-robot-operating-system>>.
- [72] ZHANG, J. W.; WANG, G. G. Image matching using a bat algorithm with mutation. In: TRANS TECH PUBL. *Applied Mechanics and Materials*. [S.l.], 2012. v. 203, p. 88–93.
- [73] MISHRA, S.; SHAW, K.; MISHRA, D. A new meta-heuristic bat inspired classification approach for microarray data. *Procedia Technology*, Elsevier, v. 4, p. 802–806, 2012.
- [74] KOENIG, N.; HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IEEE. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. [S.l.], 2004. v. 3, p. 2149–2154.
- [75] WONNACOTT, T. H.; WONNACOTT, R. J. *Introductory statistics*. [S.l.]: Wiley New York, 1990. v. 5.
- [76] SEOW, Y. et al. Detecting and solving the kidnapped robot problem using laser range finder and wifi signal. In: *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. [S.l.: s.n.], 2017. p. 303–308.