Universidade de Brasília
Faculdade de Tecnologia

Scan-matching para Localização de Robôs Móveis em Ambientes Virtuais

Matheus Virgílio da Silva Ferreira

PROJETO FINAL DE CURSO

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Brasília

2024

Universidade de Brasília
Faculdade de Tecnologia

Scan-matching para Localização de Robôs Móveis em Ambientes Virtuais

Matheus Virgílio da Silva Ferreira

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Orientador: Prof. Dr. Daniel M. Muñoz Arboleda
Coorientador: MSc. Mario Andres Pastrana Triana

Brasília
2024

Silva Ferreira, Matheus Virgílio da.
S769s Scan-matching para Localização de Robôs Móveis em Ambientes Virtuais / Matheus Virgílio da Silva Ferreira; orientador Daniel M. Muñoz Arboleda; coorientador Mario Andres Pastrana Triana. -- Brasília, 2024.
61 p.

Projeto Final de Curso (Engenharia de Controle e Automação) -- Universidade de Brasília, 2024.

1. Robótica móvel. 2. Scan-matching. 3. Algoritmos bioinspirados. 4. Filtro de Kalman. I. Arboleda, Daniel M. Muñoz, orient. II. Triana, Mario Andres Pastrana, coorient. III. Título

Universidade de Brasília
Faculdade de Tecnologia

Scan-matching para Localização de Robôs Móveis em Ambientes Virtuais

Matheus Virgílio da Silva Ferreira

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Trabalho aprovado. Brasília, 20 de setembro de 2024:

Prof. Dr. Daniel Muñoz Arboleda
(FGA/UnB)
Orientador

Prof. Dr. Carlos H. Llanos Quintero
(FT/UnB)
Examinador interno

Prof. Dr. Janier Arias Garcia (UFMG)
Examinador externo

Brasília
2024

Este trabalho é dedicado a todes
pesquisadores neurodivergentes.

Agradecimentos

Agradeço a todos os familiares e amigos que me incentivaram nesta jornada. Agradecimento especial aos meus pais pelo apoio incondicional nas muitas decisões acadêmicas ousadas que tomei ao longo destes anos e à minha amada pelo incentivo final para a conclusão do curso e pela cuidadosa assistência na revisão do presente trabalho.

Agradeço também ao Prof. Dr. Walter de Britto Vidal Filho pela primeira oportunidade iniciação científica e pelos anos seguintes de colaboração.

“Even with the rest belated, everything is antiquated
Are you writing from the heart?
Are you writing from the heart?
Even in his heart the Devil has to know the water level
Are you writing from the heart?
Are you writing from the heart?”
(Sufjan Stevens)

Resumo

Robótica móvel refere-se ao campo da robótica que se concentra na concepção, construção e operação de robôs capazes de navegar autonomamente em seu ambiente. Esses robôs empregam sensores e outras tecnologias para perceber seu entorno, tomando decisões com base nessas informações. Dentro do amplo escopo do tema, um dos problemas clássicos é o de localização em ambientes fechados, onde, especificamente para o caso de rastreamento de posição, o filtro de Kalman se destaca como uma das metodologias mais proeminentes. Tratando-se de um algoritmo bastante estudado, ele deve lidar com inconsistências de medição ao longo do rastreamento - que, neste projeto, advém de um erro sistemático no balanceamento das rodas do robô no seu deslocamento -, e há diferentes implementações dele, buscando diversificar a sua abordagem. O *scan-matching*, ou correspondência de escaneamento, é uma técnica que consiste em comparar duas leituras de sensoriamento de posições próximas (no caso, leituras de distância medidas em 360°) e, a partir delas, descobrir o deslocamento (linear e angular) relativo entre as duas. Dessa forma, o *scan-matching* é adequado para a etapa de correspondência do filtro de Kalman, onde são comparadas a predição de posição e as observações dos sensores a fim de gerar um nova estimativa de posição mais confiável, além de compensar o desbalanceamento das rodas. No presente trabalho, todo esse processo é sustentado pelo algoritmo *Particle Swarm Optimization* (PSO), que, como-enquantoenquanto um algoritmo bioinspirado, consiste em simulações computacionais de comportamentos observados na natureza utilizados para otimização matemática. Com isso, um modelo meta-heurístico bioinspirado foi desenvolvido em ambiente virtual a partir da integração entre o software CoppeliaSim e a linguagem de programação Python. Tratando-se de um ambiente virtual, um erro sistemático de desbalanceamento das rodas do robô foi modelado para gerar as inconsistências de localização vistas em condições reais, emulando adversidades existentes para as quais o robô deverá se adaptar. Os resultados demonstraram que o algoritmo de *scan-matching* alcançou uma acurácia superior a *97,13%* em diversas simulações, realizadas em cenários distintos. Para testar a correção do desbalanceamento das rodas, foram realizadas simulações com diferentes entradas: (1,00, 1,00), (0,95, 1,00), (0,95, 0,95), (1,05, 1,05) e (1,05, 0,95), representando diferentes intensidades de desbalanceamento nas rodas. As saídas médias obtidas foram, respectivamente: (0,997, 0,997), (0,958, 1,00), (0,959, 0,958), (1,035, 1,036) e (1,038, 0,956), valores que se aproximam dos correspondentes de entrada, comprovando a eficiência do algoritmo de localização.

Palavras-chave: Robótica móvel. Scan-matching. Algoritmos bioinspirados. Filtro de Kalman.

Abstract

Mobile robotics refers to the field of robotics that focuses on designing, building, and operating robots capable of autonomously navigating their environment. These robots use sensors and other technologies to perceive their surroundings and make decisions based on that information. One of the classic challenges in this field is indoor localization, particularly for position tracking, where the Kalman filter stands out as one of the most prominent methodologies. As a well-studied algorithm, it must address measurement inconsistencies during tracking—specifically in this project, those caused by systematic errors in the robot's wheel balancing during movement. Various implementations of the Kalman filter exist, each seeking to diversify its approach. *Scan-matching* is a technique that involves comparing two sensor readings taken at nearby positions (in this case, 360° distance measurements) to estimate the relative displacement (both linear and angular) between them. Thus, scan-matching is well-suited for the Kalman filter's matching step, where predicted positions and sensor observations are compared to generate a new, more reliable position estimate, also compensating for wheel imbalances. In this project, the entire process is supported by the *Particle Swarm Optimization* (PSO) algorithm, a bioinspired technique that simulates behavior observed in nature to optimize mathematical solutions. A bioinspired metaheuristic model was developed in a virtual environment using the integration of CoppeliaSim software and Python.Since this is a virtual environment, a systematic wheel imbalance was modeled to generate localization inconsistencies similar to those encountered in real-world conditions, simulating challenges the robot must adapt to.The results showed that the scan-matching algorithm achieved accuracy higher than *97.13%* in various simulations conducted in different scenarios. To test the correction of wheel imbalance, simulations were performed with different inputs: (1.00, 1.00), (0.95, 1.00), (0.95, 0.95), (1.05, 1.05), and (1.05, 0.95), representing different intensities of wheel imbalance. The average outputs obtained were, respectively: (0.997, 0.997), (0.958, 1.00), (0.959, 0.958), (1.035, 1.036), and (1.038, 0.956), values close to the corresponding inputs, demonstrating the effectiveness of the localization algorithm.

Keywords: Mobile robotics. Scan-matching. Bioinspired algorithms. Kalman filter.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABC	Artificial Bee Colony	15
ABNT	Associação Brasileira de Normas Técnicas	16
COVID	Corona Vírus Decease	15
DE	Differential Evolution	15
EKF	Filtro de Kalman extendido	23
FDP	Função de Distribuição de Probabilidade	15
FGA-UnB	Faculdade do Gama - UnB	15
IA	Inteligência Artificial	15
KF	Filtro de Kalman	23
LiDAR	Light Detection and Ranging	15
LMS	Localização e Mapeamento Simultâneo	15
UnB	Universidade de Brasília	15
UV-C	Ultravioleta tipo C	15
WMR	Wheeled Mobile Robots	18
WOA	Whale Optimization Algorithm	15

Lista de símbolos

Símbolos romanos

Símbolos gregos

Sumário

1 Introdução

1.1 Contextualização

Em 2020, a entrada do mundo na pandemia da COVID-19 afetou a comunidade científica de muitas formas. A urgência criada pelas primeiras mortes e o número crescente de infectados pela SARS-CoV-2 mudou o paradigma científico (FERREIRA, J. P.; EPSTEIN; ZANNAD, 2021) e exigiu maior agilidade para o desenvolvimento de tecnologias que assistissem os enfermos, o corpo médico e todos aqueles susceptíveis a doença. Por isso, paralelo à criação das vacinas de imunização e às políticas públicas de isolamento social, ainda havia a necessidade de preencher as funções abandonadas pelo risco de infecção do Coronavírus, destacando-se soluções baseadas em robótica móvel.

No começo da pandemia, a falta de compreensão de como ocorria o contato pelo vírus dobrou a atenção do corpo médico de linha de frente e dos profissionais essenciais, obrigados a ter contato com os infectados. Visando a saúde desses, tecnologias assistidas por robótica móvel despontaram buscando tornar esse contato mais seguro. Robôs de telepresença para educação a distância (ULUER et al., 2021) e "robôs médicos" para atendimento de crianças doentes (FIORINI et al., 2022) são alguns exemplos que tentavam humanizar a distância do isolamento social. Na área de descontaminação de ambientes, também foram utilizados robôs móveis responsáveis por eliminar altos percentuais de carga viral em superfícies e no ar utilizando reagentes químicos (BLOCK; ROWAN, 2020) ou as amplamente aplicada lâmpadas de radiação ultravioleta tipo C (UV-C) (MOEZ; GHARBI; HAMZA, 2021). Foi nesse contexto, que se iniciou o presente trabalho na forma de um Projeto de Iniciação Científica (FERREIRA, M., 2022), seguido por um artigo publicado (FERREIRA, M.; VIDAL FILHO, 2022).

A proposta de pesquisa foi desenvolver um robô móvel de desinfecção UV-C que, quando deixado em qualquer posição de uma sala, navegasse-a completamente, desviando de obstáculos para emitir radiação ultravioleta em pontos críticos, eliminando a carga viral de SARS-CoV2 em superfícies. O projeto focou na simulação da navegação em ambiente virtual, onde se fez necessário o estudo e aplicação de técnicas de localização em robótica móvel. Enquanto a navegação era dependente da medição de odometria do robô, a localização utilizou um sistema de *scan-matching* (verificação de correspondência de escaneamentos, em tradução livre) a partir da leitura de um único sensor de distância a laser. O funcionamento é o seguinte: dado o escaneamento de distância partindo de duas posições próximas, o sistema era capaz de mensurar o deslocamento entre os dois pontos e sua diferença relativa de orientação.

Localização é um problema clássico de robótica móvel, processo pelo qual um robô determina sua posição e orientação em relação a um sistema de coordenadas conhecido, isso é, sua pose. Uma localização precisa é essencial para que o robô possa navegar em seu ambiente de forma autônoma e executar tarefas com eficiência (PANIGRAHI, P. K.; BISOY, S. K., 2022). Destaca-se a existência de uma gama de abordagens para esse problema com os mais diversos sensores, como por exemplo, a localização por modelos de Markov, por Filtro de Kalman, Gride ou Monte Carlo (THRUN; BURGARD; FOX, 2005), ou ainda a possibilidade de se utilizar Inteligência Artificial (IA) para a resolução do mesmo (CEBOLLADA et al., 2021). Entretanto, para a implementação das técnicas de localização, os robôs móveis comumente utilizam sistemas embarcados, os quais possuem restrições computacionais em termos de desempenho, memória disponível e consumo de energia. Tais restrições devem ser consideradas na escolha dos métodos de localização.

1.2 Justificativa

Levando em consideração a problemática anterior, propõe-se o desenvolvimento de um sistema de localização de robô móvel por filtro de Kalman auxiliado por *scan-matching*, a ser reproduzido em ambiente virtual. Será utilizado o modelo de simulação do robô de pequeno porte *Maria* (TRIANA, 2022), produzido pela FGA-UnB, o qual conta com tração diferencial, possuindo um único sensor de distância a laser que coletará escaneamentos em 360°, a partir do centro da estrutura. O simulador escolhido foi o software CoppeliaSim, versão educacional 4.5.1, que será integrado com a linguagem de programação Python para a coleta e subsequente análise de dados.

No intuito de aperfeiçoar o *scan-matching*, se utilizarão métodos de otimização mediante algoritmos bioinspirados para determinar a pose com base nas melhores estimativas de localização. Neste trabalho assume-se que a posição inicial e o mapeamento do ambiente são conhecidos pelo robô móvel. Nesse sentido, caracteriza-se o problema como um caso de rastreamento de posição em ambientes estáticos. Os algoritmos bioinspirados estudados foram: (a) por enxame de partículas - em inglês, *Particle Swarm Optimization* (PSO); (b) por colonia artificial de abelhas - em inglês, *Artificial Bee Colony* (ABC); (c) por evolução diferencial - em inglês, *Differential Evolution* (DE); (d) e, por último, o algoritmo baseado em baleias - em inglês, *Whale Optimization Algorithm* (WOA). Porém, de todos esses, na implementação final, apenas o PSO foi testado.

Objetivamente, aplicando o método de Kalman (PANIGRAHI, P.; BISOY, S., 2021), uma vez iniciada a simulação, o robô se encontrará em um ponto pré-determinado de um mapa conhecido, possuindo uma coordenada odométrica equivalente a um palpite inicial de onde ele mesmo está, além de ter os escaneamentos feitos do seu entorno. Estas são as etapas de observação e predição de medidas. Seguidamente, na fase de correspondência, o

escaneamento esperado com base na predição da pose é comparado com o escaneamento real, os quais devem estar defasados, tendo em vista a não-confiabilidade do sensoriamento odométrico (AQEL et al., 2016). Essa não-confiabilidade provém de um erro sistemático de desbalanceamento das rodas do robô, modelado para gerar as inconsistências de localização vistas em condições reais, emulando adversidades existentes para as quais o robô deve se adaptar. Por fim, na etapa de estimação, cabe ao algoritmo bioinspirado utilizado aferir uma nova predição de posição de acordo com essa defasagem, além de compensar o desbalanceamento das rodas. Todo o processo é repetido de forma iterativa enquanto o robô se movimenta, aumentando sucessivamente a confiança da pose estimada.

A implementação deste trabalho é relevante para diversas aplicações dentro da robótica móvel que se baseiam em localização, diminuindo a dependência do sensoriamento interno, muito sensível ao acúmulo de erro. Atividades de monitoramento e inspeção *in-door* para setores com ambientes complexos como instalações industriais (KAZEMINASAB et al., 2021), tubulações subterrâneas (SZREK et al., 2020) e usinas termonucleares (BIRD et al., 2019) são possíveis aplicações deste tipo de estudo.

1.3 Objetivos

1.3.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver, em simulação computacional, um método de localização de robôs móveis baseado em filtro de Kalman utilizando técnicas de *scan-matching* assistido pela utilização do algoritmo bioinspirado *Particle Swarm Optimization*.

1.3.2 Objetivos específicos

Para cumprir o objetivo geral deste trabalho, os seguintes objetivos devem ser alcançados:

a) Desenvolver algoritmo de *scan-matching* que afira a pose relativa dado dois escaneamentos de distância de posições próximas;

b) Compensação no movimento do robô do desbalanceamento do deslocamento das rodas;

c) Implementar uma simulação virtual do robô móvel com dispositivos que tenham as mesmas características daqueles esperados do modelo físico, tais como sensores LiDAR e *encoders* odométricos;

d) Implementar a comunicação entre o algoritmo de *scan-matching* e o simulador a fim de atualizar melhores estimativas de localização.

2 Fundamentação Teórica

Este capítulo apresenta os principais tópicos nos quais este trabalho se fundamenta. Se buscará desenvolver conceitos de robótica móvel, dissertando sobre as estratégias mais utilizadas na abordagem de localização, como princípios de odometria e métodos probabilísticos e meta-heurísticos. Em seguida, é explorado brevemente o conceito das ferramentas computacionais de simulação e, finalmente, faz-se uma revisão bibliográfica de trabalhos com temas correlatos, podendo-se apontar as diferenças do presente trabalho com as demais abordagens.

2.1 Robótica móvel

A robótica móvel refere-se ao campo da robótica que se concentra na concepção, construção e operação de robôs capazes autonomamente de navegar em seu ambiente. Esses robôs empregam sensores e outras tecnologias para perceber seu entorno, tomando decisões com base nessas informações. Os robôs móveis podem ser usados em uma ampla gama de aplicações (SILVA ORTIGOZA et al., 2012), desde a manufatura (DÖRFLER et al., 2022) a exploração mineradora (GREHL; MISCHO; JUNG, 2017) e marítima (MU et al., 2022), para o reconhecimento de terreno (BRUZZONE; QUAGLIA, 2012), em missões de busca e resgate de pessoas (MURPHY et al., 2009), na limpeza de resíduos perigosos (WANG et al., 2009), na assistência médica (RENTSCHLER et al., 2007) e mesmo na indústria do entretenimento (GRAF; SCHRAFT; NEUGEBAUER, 2000).

Robôs móveis são primariamente divididos pela sua locomoção, a qual impacta no seu design dependendo não apenas do meio onde o robô se move, mas também de critérios como técnicas de manobrabilidade, controlabilidade, condições do terreno, eficiência, estabilidade, entre outros aspectos (RUBIO; VALERO; LLOPIS-ALBERT, 2019). Em termos de locomoção, a classificação mais comum envolve robôs estacionários, terrestres, aquáticos e aéreos (DELMERICO et al., 2019). O presente trabalho propõe soluções de localização para robôs terrestres, mais especificamente os movidos por rodas, também conhecidos pela sigla WMR (*Wheeled Mobile Robots*, em inglês).

Assim como nos autoveículos, as vantagens do uso de rodas estão na baixa complexidade para programar seu deslocamento, além de causarem menor desgaste nas superfícies por onde se movem. Dentro da categoria WMR há diferentes configurações para o arranjo das rodas, se podendo citar: de tração diferencial, triciclos, omnidirecionais e de direção Ackerman (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Na Figura 2.1, é representado um WMR de tração diferencial traseira com um elemento omnidirecional frontal (não-tracionado), modelo implementado neste trabalho. A escolha dele impacta na cine-

mática da movimentação, que é caracterizada pelas suas equações de odometria durante o processo de navegação.

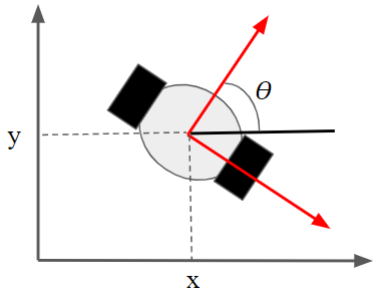


Figura 2.1 – Odometria de um robô móvel de tração diferencial

2.2 Odometria

Partindo de uma perspectiva humana, o problema de localização dentro de um espaço fechado é bastante trivial. Mover-se dentro do espaço é simplesmente verificar se há obstáculos no caminho e contorná-los para se deslocar entre dois pontos. Entretanto, um princípio inicial que faz essa trivialidade não existir para robótica móvel é a referência. Seres humanos ao se movimentar, costumeiramente, não precisam associar geometricamente o deslocamento que estão fazendo, pois o sensoriamento feito pelos seus sentidos já capta as informações necessárias para aquela ação. Entretanto, um robô móvel tende a iniciar com apenas uma referência, a posição inicial de partida e, por isso, é a referência que deve ser considerada para todas as suas ações subsequentes.

A odometria é um processo usado por robôs móveis para estimar sua pose $\mathbf{p}$ em relação à sua localização inicial (BEN-ARI; MONDADA, 2018), usando dados de sensores de movimento (como acelerômetro e giroscópio) para calcular as variações de translação e rotação na pose do robô (AQEL et al., 2016). Na odometria cartesiana, são utilizadas equações de posição x e y dependentes diretamente da velocidade e da orientação θ do robô ao longo do tempo t. Se tratando de um WMR de tração diferencial, consideram-se os deslocamentos lineares das rodas (ΔS_d e ΔS_e, direita e esquerda, respectivamente) e a distância b entre as mesmas (PANIGRAHI, P. K.; BISOY, S. K., 2022),

$$\begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} = \mathbf{p}(t-1) + \begin{bmatrix} \frac{\Delta S_d + \Delta S_e}{2} cos\left(\theta(t-1) + \frac{\Delta S_d - \Delta S_e}{2b}\right) \\ \frac{\Delta S_d + \Delta S_e}{2} sen\left(\theta(t-1) + \frac{\Delta S_d - \Delta S_e}{2b}\right) \\ \frac{\Delta S_d - \Delta S_e}{b} \end{bmatrix}, \tag{2.1}$$

$$\mathbf{p}(t) = \mathbf{p}(t-1) + \Delta\mathbf{p}.$$

É importante notar que as medições de ΔS_d e ΔS_e são feitas por *encoders*, mais comumente ópticos e rotacionais, que contam os passos de rotação das rodas para determinar o deslocamento total. Um mal funcionamento deles pode acarretar no acúmulo de erros sistemáticos, trazendo imprecisões significativas na estimativa de posição (BORENSTEIN; FENG, 1995). O erro sistemático refere-se a um tipo de erro consistente e previsível, que ocorre sempre da mesma maneira. No contexto de robôs móveis, eles são causados por imperfeições cinemáticas, como diâmetros de roda desiguais, incerteza sobre a distância efetiva entre eixos (BORENSTEIN; FENG, 1995), e por uma contagem inconstante dos passos.

Considerando tais imprecisões, a robótica móvel recorre a recursos adicionais para o problema de localização. Sensores exteroceptivos como sensor a laser, sensores de visão e sensores ultrassônicos são aplicados para fazer observações sobre seu ambiente, observações essas que podem ser combinadas com a odometria do robô para localizá-lo no ambiente (PANIGRAHI, P.; BISOY, S., 2021).

2.3 Abordagens de localização em robótica móvel

Embora o conceito de localização dentro da robótica móvel já tenha sido amplamente explorado (seção 1.1), é importante defini-lo propriamente:

> A localização do robô móvel é o problema de determinar a pose de um robô em relação a um determinado mapa do ambiente. Muitas vezes é chamado de estimativa de posição. A localização de robôs móveis é uma instância do problema geral de localização, que é o problema de percepção mais básico da robótica (THRUN; BURGARD; FOX, 2005, p.191, tradução livre) [1].

Na citação, é interessante observar o uso do termo "percepção". Trata-se de outro aspecto básico da robótica móvel que põe na forma de dados sensoriais as características do espaço no entorno e tenta interpretá-los. A interpretação, por sua vez, é feita por técnicas

[1] No original: *Mobile robot localization is the problem of determining the pose of a robot relative to a given map of the environment. It is often called position estimation. Mobile robot localization is an instance of the general localization problem, which is the most basic perceptual problem in robotics.*

adequadas à categoria do problema de localização. Para fazer uma abordagem geral deste tópico, a subseção 2.3.1 apresenta essas categorizações, enquanto as subseções 2.3.2 e 2.3.3 explicarão as técnicas desenvolvidas para lidar com elas.

2.3.1 Problemas de localização

Ao projetar um robô móvel, deve-se determinar o paradigma de localização em que ele está inserido a fim de traçar uma metodologia condizente para lidar com as funções desejadas. A localização pode ser classificada de diferentes maneiras, mais enfaticamente entre localização relativa e absoluta.

A localização relativa estima a posição e a orientação do robô móvel, combinando informações produzidas por diversos sensores, em relação a um ponto referencial (GONZÁLEZ et al., 2009). Em contrapartida, a localização absoluta usa de marcadores de navegação, análise de mapas e sistemas de sinais de satélite, como o Global Positioning System (GPS), para mensurar a posição independente do tempo ou do local (ALATISE; HANCKE, 2020). Em suma, enquanto a primeira tem maior erro de medição e uma resolução mais alta, funcionando melhor em curtas distâncias, a segunda é o justo oposto, com erros menores, porém resoluções mais baixas.

Dentro do escopo de localização relativa em aplicações *in-door*, há dois problemas clássicos: o rastreamento de posição e a localização global. No rastreamento de posição (do inglês, *position tracking*), é assumido que a pose inicial do robô é conhecida, ou seja, a tarefa de localizar o robô ao longo do tempo é acomodar seu ruído de movimento, com a incerteza podendo ser aproximada por uma distribuição unimodal centralizada próxima a pose real do robô (THRUN; BURGARD; FOX, 2005). Na localização global, a posição inicial é desconhecida, não sendo possível limitar a estimativa em torno de uma predição mais confiável, necessitando de técnicas mais desenvolvidas (SE; LOWE; LITTLE, 2001). Um caso especial de localização global é o "robô sequestrado", onde o robô pode ser movido manualmente entre posições, situação a qual ele deve se adaptar para se relocalizar no ambiente (CEN et al., 2008).

2.3.2 Método de localização de Markov

O método de Markov explora abordagens probabilísticas para resolver o problema de localização em robótica móvel. Considerando um espaço discreto, onde o número de posições é limitado, a ideia central desse método é ir aumentando gradativamente a confiança das estimativas de posição à medida que o robô se desloca, para, em um dado momento, ele ter um estimativa ótima (com maior probabilidade de acerto) (THRUN; BURGARD; FOX, 2005).

Nesse sentido, ao invés de manter uma única hipótese sobre onde um robô pode

estar no ambiente, a localização de Markov mantém uma distribuição de probabilidade no espaço de todas essas hipóteses (FOX; BURGARD; THRUN, 1999). Enquanto o rastreamento e técnicas de localização locais visam compensar erros de odometria que ocorrem durante a navegação do robô, os métodos Markovianos podem estimar a posição globalmente (GADEYNE; BRUYNINCKX, 2001).

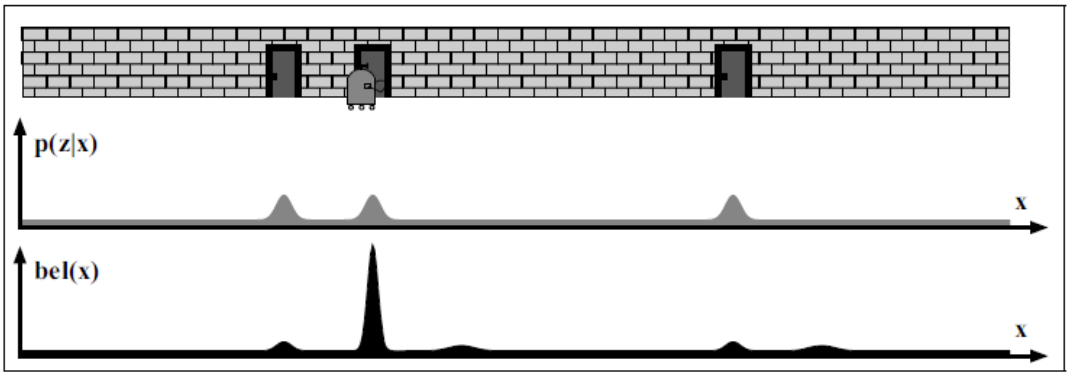


Figura 2.2 – Ilustração do algoritmo de localização de Markov

Fonte: Probabilistic Robotics (THRUN; BURGARD; FOX, 2005, p.199)

Replicando a demonstração teórica mais usual do algoritmo de Markov (THRUN; BURGARD; FOX, 2005), considera-se um robô que se desloca por um espaço 1-D (Figura 2.2). Inicialmente, o robô tem, para todos os pontos de um dado mapa, uma probabilidade uniforme de estar em qualquer ponto. Ao se deslocar e detectar uma porta, o robô passa a ter três hipóteses principais, incrementando a confiança nas referidas posições. Movendo-se para a segunda porta, o robô novamente avalia a possibilidade de estar na primeira, deslocando a confiança obtida na primeira iteração. Como a confiança é somada recursivamente, a crença do robô dele estar em frente a segunda porta se destaca, uma vez que ela foi considerada nas duas iterações.

As implementações de Markov presumem um ambiente estático, o que é conhecido como Suposição de Markov (do inglês, *Markov Assumption*) (FOX; BURGARD; THRUN; CREMERS, 1998). É possível afirmar que o algoritmo aumenta a confiança nos seus estados à medida que detecta objetos mais característicos de uma cena (FOX; BURGARD; THRUN, 1998), como as portas no exemplo. Na Equação 2.4, isso é expressado pelo termo $P(z_t|x_t, m)$, que se refere a probabilidade de estar sendo observado o conjunto de características z_t, dados a pose atual x_t e o mapa do ambiente m (FOX; BURGARD; THRUN, 1999).

E e $\overline{E}$ são o novo estado e o estado prévio do robô, respectivamente. $\overline{E}$ é calculado pela Equação 2.2 (ou, no caso discreto, pela Equação 2.3), a qual representa o aspecto recursivo do algoritmo de Markov, em que $P(x_t|u_t, x_{t-1})$ é a probabilidade da pose atual ser x_t dados a ação de controle u_t e a pose anterior x_{t-1} (PANIGRAHI, P.; BISOY, S., 2021). Na Equação 2.4, η indica que o resultado é normalizado a cada iteração.

$$\overline{E} = \int P(x_t|u_t, x_{t-1})E(x_{t-1})dx_{t-1}, \tag{2.2}$$

$$\overline{E} = \sum_{x_{t-1}} P(x_t|u_t, x_{t-1})E(x_{t-1}), \tag{2.3}$$

$$E = \eta P(z_t|x_t, m)\overline{E}. \tag{2.4}$$

2.3.3 Filtro de Kalman

O filtro de Kalman (KF) [2] é um tópico importante não apenas para a robótica, mas também no campo do controle de sistemas (LEE; RICKER, 1994) e em fusão sensorial. A fusão sensorial se refere à integração de múltiplas fontes de dados sensoriais para obter uma estimativa mais precisa e confiável de um estado ou variável de interesse (SASIADEK, J. Z., 2002). O KF é capaz de realizar essa fusão de forma otimizada, considerando as incertezas associadas a cada sensor e combinando suas medições de maneira ponderada, sendo um mecanismo eficiente para melhorar a precisão e a confiabilidade das estimativas obtidas a partir de diferentes sensores (SASIADEK, J., 2002).

Conceitualmente, o KF consiste em um conjunto de equações matemáticas para estimar recursivamente o estado de um processo, com o objetivo de minimizar a média do erro ao quadrado. Ele é capaz de realizar estimativas de estados passados, presentes e futuros, mesmo quando a natureza precisa do sistema modelado é desconhecida (WELCH, G.; BISHOP, 2006). O KF é amplamente utilizado em áreas como controle de processos (AUGER et al., 2013), navegação (COOPER; DURRANT-WHYTE, 1994), visão computacional (WELCH, G. F., 2020) e em outras disciplinas onde a precisão na estimativa do estado é essencial, por exemplo, em rastreamento de objetos (KIM; PARK, 2020).

O método de Markov e o KF são abordagens distintas para estimativa em sistemas dinâmicos. O KF, um caso especial da localização de Markov, utiliza distribuições de probabilidade gaussianas para representar as posições do robô, tornando-o eficiente para o problema de rastreamento. No entanto, ele também requer uma aproximação conhecida da posição inicial do robô, não conseguindo se reencontrar caso o robô se perca (PANIGRAHI, P.; BISOY, S., 2021). Tal restrição é inexistente nos métodos de localização markovianos, os quais são capazes de solucionar problemas de localização global.

O KF foi desenvolvido para lidar com sistemas lineares (para casos não-lineares pode-se utilizar o *Extended Kalman Filter* (EKF), ou filtro de Kalman extendido (RIBEIRO, 2004)), e é composto por quatro etapas principais: observação, previsão de medição/percepção, correspondência e estimação (NEGENBORN, 2003), conforme descrito a seguir.

a) Etapa de observação: nesta etapa são feitas as medições do sistema real por meio de sensores. Essas medições podem incluir dados como posição, velocidade,

[2] Sigla para *Kalman Filter*, nomenclatura mais utilizada.

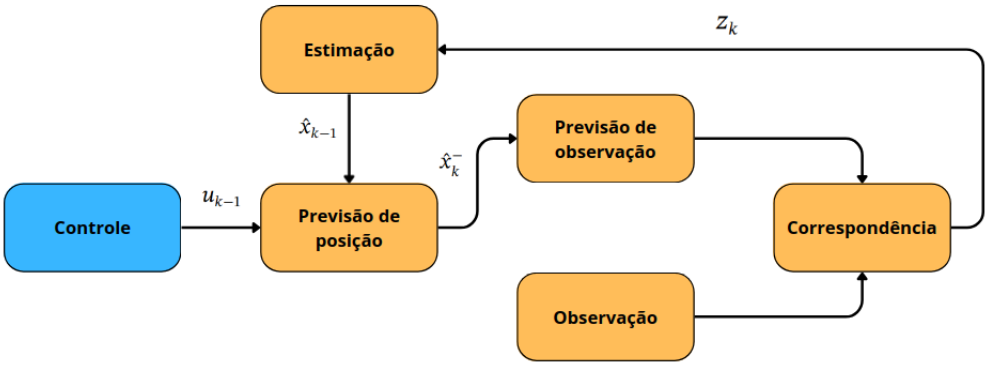


Figura 2.3 – Filtro de Kalman.

orientação, entre outros. As observações são usadas para atualizar o conhecimento sobre o estado atual do sistema, na iteração k;

b) Etapa de previsão da medição/percepção: com base no estado atual do sistema estimado na iteração anterior $\hat{x}_{k-1}$, é realizada uma previsão do estado $\hat{x}_k^-$ e das medidas que seriam esperadas dos sensores (Equação 2.5). Essa previsão é obtida por meio do modelo do sistema, representado pela variável de controle u_{k-1};

c) Etapa de correspondência: nesta etapa as medidas reais obtidas pelos sensores (z_k) são comparadas com as previsões feitas na etapa anterior. Uma previsão possui uma variância (ou covariância, caso seja multivariável) de erro P_k^-, composto pelo erro no instante anterior P_{k-1} e o erro de ruído do processo Q (Equação 2.6), enquanto a medição possui uma variância do ruído de medição R. Com elas, objetivo é identificar quais medidas correspondem às previsões e como elas se relacionam. Uma correspondência correta entre as medidas reais e as previsões é essencial para obter estimativas precisas do estado do sistema;

d) Etapa de estimação: com base nas correspondências encontradas, o KF conclui uma estimativa do estado, dada por $\hat{x}_k$. Na Equação 2.8, ele combina as informações das medições reais com as previsões feitas pelo modelo para atualizar seu conhecimento sobre o estado do sistema. Essa atualização é feita levando em consideração as incertezas das medições e demais informações do modelo na Equação 2.7, onde KG é o ganha de Kalman (*Kalman Gain*, em inglês).

$$\hat{x}_k^- = \hat{x}_{k-1} + u_{k-1} \tag{2.5}$$

$$P_k^- = P_{k-1} + Q \tag{2.6}$$

$$KG_k = \frac{P_k^-}{P_k^- + R} \tag{2.7}$$

$$\hat{x}_k = \hat{x}_k^- + KG_k\left(z_k - \hat{x}_k^-\right) \tag{2.8}$$

$$P_k = (1 - KG_k)P_k^- \tag{2.9}$$

As quatro etapas são repetidas em um processo recursivo em que, a cada iteração, o KF atualiza as estimativas de estado com base nas observações mais recentes. Isso permite acompanhar as mudanças do sistema ao longo do tempo, fornecendo estimativas cada vez mais precisas do estado atual, diminuindo a incerteza (Equação 2.9). Como dito anteriormente, as estimativas têm forma de função de densidade de probabilidade gaussiana, dada pela seguintes expressões (PANIGRAHI, P.; BISOY, S., 2021),

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.10}$$

$$P(\mathbf{X}) = \frac{1}{\sqrt[k]{2\pi}|\sqrt{\sum}|} \exp\left(-\frac{(\mathbf{X}-\mu)^T}{\sum(\mathbf{X}-\mu)}\right), \tag{2.11}$$

onde, para o caso unidimensional (vide Equação 2.10), x é uma variável escalar aleatória, μ e σ são a média e desvio padrão de x, respectivamente. Para o caso multivariável (vide Equação 2.11), k é a dimensão do vetor $\mathbf{X}$, μ é o vetor média e $\sum$ é a matrix de covariância.

2.4 Algoritmos de otimização bioinspirados

Os algoritmos bioinspirados são técnicas computacionais que se inspiram em princípios e processos encontrados na natureza para resolver problemas complexos (KAR, 2016). Eles buscam emular as estratégias utilizadas por organismos vivos como animais, plantas e insetos para lidar com desafios adaptativos. Esses algoritmos aproveitam os conceitos de evolução, comportamento coletivo, comunicação e aprendizado presentes nos sistemas biológicos. Ao aplicar esses princípios na computação, os algoritmos bioinspirados demonstram potencial para encontrar soluções eficientes e inovadoras para uma ampla gama de problemas como otimização (BINITHA; SATHYA et al., 2012), classificação (CHRISTO et al., 2019), reconhecimento de padrões (KUREICHIK et al., 2020), roteamento de redes (HAMRIOUI; LORENZ, 2017), entre outros (KAR, 2016).

2.4.1 Otimização por enxame de partículas

O algoritmo de otimização por enxame de partículas, o PSO (*Particle Swarm Optimization*), é uma técnica bioinspirada que se baseia no comportamento coletivo de enxames

na natureza para resolver problemas de otimização (JAIN, N.; NANGIA; JAIN, J., 2018). Com inspiração na movimentação de revoadas, isto é, em como manadas de pássaros e afins procuram a melhor corrente de ar para seu voo - sem usar necessariamente de uma comunicação verbal para guiá-las -, o PSO utiliza uma população de partículas que se movem pelo espaço de busca, visando encontrar a melhor solução. Cada partícula é influenciada pela sua melhor posição individual e pela melhor posição global encontradas a cada iteração. Essa interação entre as partículas com uma capacidade de comunicação e aprendizado coletivo permitem ao algoritmo explorar e convergir rapidamente para soluções ótimas. Assim, incorporando a ideia de cooperação e adaptação naturais de enxames, o PSO tem sido amplamente aplicado em problemas de otimização em diversas áreas, da engenharia (GUEDRIA, 2016) (RAHMAT-SAMII, 2003), ciência da computação (NGATMAN; SHARIF; NGADI, 2017) (MURUGANANDHAM; BANU, 2010) e finanças (CHIAM; TAN; MAMUN, 2009) (ZHOU et al., 2019).

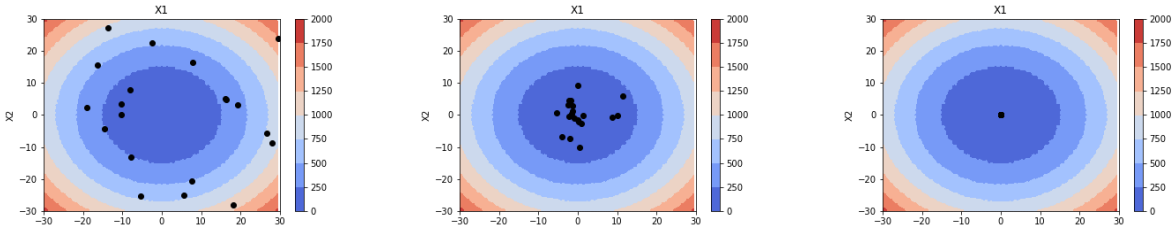
O PSO cria um conjunto de S soluções (ou partículas) N-dimensionais que são balanceadas por características individuais e coletivas chamadas de coeficiente cognitivo c_1 e social c_2, respectivamente. O quanto a partícula se moverá da posição atual $\vec{z}_i$ é dependente da velocidade $\vec{v}_i$ ($i = 0, 1, 2, ..., S-1$) de cada uma delas (EBERHART; KENNEDY, 1995),

$$\vec{v}_i[k] = w\vec{v}_i[k-1] + c_1 r_1(\vec{Z}_i - \vec{z}_i[k-1]) + c_2 r_2(\vec{Z}_S - \vec{z}_i[k-1]), \tag{2.12}$$

$$\vec{z}_i[k] = \vec{z}_i[k-1] + \vec{v}_i[k], \tag{2.13}$$

onde k é o número da iteração, w é o coeficiente de inércia que decai linearmente de 0,9 para 0,1 e r_1 e r_2 são números aleatórios com distribuição de probabilidade uniforme entre 0 e 1.

Cada partícula $\vec{z}_i$ possui um índice de performance na função de custo do seu problema, podendo essa ser de maximização ou minimização, onde se procura os maiores e menores *score*, respectivamente. A cada iteração são atualizados o melhor resultado individual de uma partícula ($\vec{Z}_i$) e o melhor resultado global entre todas elas ($\vec{Z}_S$), buscando encontrar um valor ótimo para todas convergirem Figura 2.4. O deslocamento individual v_i é calculado pela Equação 2.13, ponderando seu valor anterior por um dado coeficiente de inércia w, e em função de $\vec{Z}_i$ e $\vec{Z}_S$ que, ponderados pelos coeficientes cognitivo c_1 e social c_1, respectivamente, indicam o quanto cada partícula tende em direção a sua própria experiência e o quanto é levada pela experiência do enxame (VOIT, 2010).



(a) Estado inicial (b) Estado intermediário (c) Estado final

Figura 2.4 – Partículas do PSO convergindo

2.4.2 Otimização por colônia artificial de abelhas

O algoritmo ABC (sigla para *Artificial Bee Colony*) é um algoritmo bioinspirado que se baseia no comportamento das abelhas em busca de alimento para resolver problemas de otimização (KARABOGA, 2010) em diversas áreas do conhecimento (GAO et al., 2018). Inspirado na organização e comunicação das colônias de abelhas, o ABC possui uma população de abelhas artificiais que exploram um espaço de busca procurando pela melhor solução. Cada abelha é responsável por uma solução candidata e realiza buscas locais e globais para encontrar regiões promissoras. Simulando a comunicação real entre esses insetos através da dança, as abelhas artificiais também compartilham informações entre si a fim de encontrar soluções ótimas (LINDAUER; KERR, 1960).

Replicando o processo natural, o algoritmo ABC considera a estrutura hierárquica do sistema de organização na colmeia. Aqui são definidos três tipos de operações, que ocorrem sucessivamente. Inicialmente, há abelhas *onlookers* que estão constantemente procurando por novas fontes de alimentos (soluções) e, uma vez que realizam a primeira varredura (exploração do espaço de busca), passam a informação para as abelhas operárias, responsáveis por extrair o alimento (verificação da solução). As *onlookers*, então, verificam se em volta da sua solução atual há outra solução superior, se guiando para as regiões com maior concentração de fontes de alimentos. Essa concentração é a tendência social para as abelhas se moverem, retornando para posição anterior caso a hipótese seja falsa. Por fim, as abelhas restantes, as escoteiras, se direcionam para posições não-exploradas, garantindo a diversidade do algoritmo (NSEEF et al., 2016).

A atualização da posição $\vec{z}$ de uma abelha ocorre pela Equação 2.14, sempre feita na etapa onde as *onlookers* procuram por fontes de alimentos/soluções. Ela pode ser repetida na etapa das abelhas operárias, a depender da qualidade da solução atual dentre todas as soluções, definida pela probabilidade p (Equação 2.15). Um número aleatório com distribuição uniforme $R \in [0, 1]$ decide essa condição, logo, caso $R > p$, a posição é atualizada pela Equação 2.14. Dessa forma, uma abelha tem menor chance de se mover dependendo do quão boa for sua solução em relação ao conjunto.

$$\vec{z}_i[k+1] = \vec{z}_i[k] + r(\vec{Z}_S - \vec{z}_i[k]), \tag{2.14}$$

$$p_i = \frac{f_i}{\sum_{i=1}^{S} f_i}, \tag{2.15}$$

onde f é o resultado da função de objetivo e r é um número aleatório entre 0 e 1.

2.4.3 Algoritmo de otimização inspirado em baleias

A *Whale Optimization Algorithm* (WOA) é uma técnica de otimização bioinspirada que incorpora o comportamentos de busca e migração das baleias na natureza dentro de uma lógica computacional. O WOA imita o comportamento social das baleias jubarte, observando dois mecanismos principais: o mecanismo de encolhimento e o modelo espiral. As baleias jubarte nadam em torno de suas presas expelindo bolhas de ar que as fecham em um círculo cada vez menor, ao longo do caminho se movimentando em forma espiral (MIRJALILI; LEWIS, 2016). Trata-se de um algoritmo especialmente usado em problemas de processamento de imagens (ABD EL AZIZ; EWEES; HASSANIEN, 2017) (ABDEL-BASSET; CHANG; MOHAMED, 2020) (ABD ELAZIZ; LU; HE, 2021).

O algoritmo WOA assume as presas alvo das jubartes como as soluções $\vec{z}_i$ do problema. Essa "caça"pode ser dividida em duas estratégias, *exploration* e *exploitation*, que se referem aos comportamentos de procurar novos espaços de solução e explorar as soluções ótimas já conhecidas, respectivamente. Qual estratégia vai ser adotada a cada iteração é definida pela variável $\vec{A}$, um vetor de números aleatórios no intervalo $[-a, a]$, onde a é inicializado com $a = 2$, decaindo linearmente até $a = 0$. Se $|\vec{A}| \geq 1$, as partículas $\vec{X}$ tendem em direção a $\vec{z}^* = \vec{Z}_S$ (melhor solução do conjunto), caracterizando uma convergência; caso contrário ($|\vec{A}| < 1$), $\vec{z}^*$ assume os valores de uma partícula escolhida aleatoriamente, estendendo as buscas. A Equação 2.16 representa as estratégias descritas acima.

$$\begin{aligned} \vec{A} &= 2\vec{a} \cdot \vec{r} - \vec{a}, \\ \vec{C} &= 2\vec{r}, \\ \vec{D} &= |\vec{C}.\vec{z}^*[k] - \vec{z}_i[k]|, \\ \vec{D}' &= |\vec{z}^*[k] - \vec{z}_i[k]|. \end{aligned} \tag{2.16}$$

em que $\vec{C}$ é um vetor com constante no intervalo [0, 2], enquanto $\vec{D}$ e $\vec{D}'$ são vetores de deslocamento.

Independentemente da estratégia adotada em cada iteração, há duas formas possíveis de ser feito o deslocamento em torno de $\vec{z}^*$. Também escolhida de forma aleatória (definida por R, havendo chances iguais para cada uma), $\vec{z}$ pode se deslocar tanto linearmente quanto

em espiral com um fator $e^{bI} \cdot \cos 2\pi I$, onde b é uma constante e I um número aleatório com distribuição uniforme entre [-1, 1], indicando o quanto a partícula é rotacionada.

$$\vec{z}_i[k+1] = \begin{cases} \vec{z}^*[k] - \vec{A} \cdot \vec{D}, & \text{se } R < 0.5, \\ \vec{D}' \cdot e^{bI} \cdot \cos 2\pi I + \vec{z}_i[k], & \text{caso contrário.} \end{cases} \tag{2.17}$$

2.4.4 Otimização por evolução diferencial

O algoritmo *Differential Evolution* (DE), ou de evolução diferencial, diferentemente de outros algoritmos bioinspirados interessados nos padrões de movimento de animais, esta baseado no processo de evolução das espécies na natureza para sua construção (FEOKTISTOV, 2006). Conforme a seleção natural, o DE utiliza uma população de soluções candidatas que evoluem ao longo das gerações (iterações). Cada solução é representada como um vetor numérico e passa por operações de mutação, cruzamento e seleção para gerar novas soluções, processos que imitam a teoria evolutiva que dita que seres com melhor adaptação têm maior probabilidade de sobreviver e produzir descendentes.

O DE pode ser considerado o algoritmo mais randômico de todos os já citados (DENG et al., 2021) (SONG et al., 2023). Ele parte de uma etapa de inicialização aleatória, formando S posições $\vec{z}_i$ iniciais. No passo seguinte, de mutação (Equação 2.18), são criados o mesmo número de vetores ruidosos v_i, gerados pela combinação linear de posições com índices aleatórios diferentes de i (j_1, j_2 e j_3). Com $\vec{z}_i$ e $\vec{v}_i$, o cruzamento (Equação 2.19) consiste em combiná-los aleatoriamente em um conjunto de $\vec{u}_i$, com a proporção de cada um definida por uma taxa de recombinação constante CR (*crossover rate*). Encerrando, na seleção se comparam as soluções de $\vec{z}_i$ e $\vec{u}_i$, sendo mantidas para a interação seguinte aquela com melhor rendimento na função objetivo f (Equação 2.20, considerando um caso de maximização) (PRICE; STORN; LAMPINEN, 2005).

$$\vec{v}_i[k] = \vec{z}_{j_1}[k-1] + r(\vec{z}_{j_2}[k-1] - \vec{z}_{j_3}[k-1]), \tag{2.18}$$

$$\vec{u}_i[k] = \begin{cases} \vec{v}_i[k], & \text{se } R < CR, \\ \vec{z}_i[k-1], & \text{caso contrário,} \end{cases} \tag{2.19}$$

$$\vec{z}_i[k] = \begin{cases} \vec{u}_i[k], & \text{se } f(\vec{u}_i[k]) > f(\vec{z}_i[k-1]), \\ \vec{z}_i[k-1], & \text{caso contrário.} \end{cases} \tag{2.20}$$

onde r e R são números aleatórios com distribuição uniforme entre 0 e 1.

2.5 Ferramentas computacionais

Uma vez introduzida uma variedade de algoritmos a serem utilizados para as técnicas de localização de robôs móveis, é importante apresentar também algumas das ferramentas que permitirão a implementação da solução de localização em um ambiente simulado. No que se refere a estruturação desse ambiente e do próprio protótipo móvel, além da coleta das informações por via de sensores virtuais, todas as atribuições ficam por conta do CoppeliaSim (versão educacional 4.5.1), software de simulação 3-D desenvolvido pela CoppeliaRobotics (TURSYNBEK; SHINTEMIROV, 2020).

Antes conhecido como V-REP (*Virtual Robot Experimentation Platform*), ele conta com uma gama de recursos para modelagem, simulação e controle de sistemas robóticos. Pode ser usado para tarefas como verificação do sistema (TURSYNBEK; SHINTEMIROV, 2020), otimização de algoritmos (OMISORE et al., s.d.), simulação de cadeias de montagem complexas em aplicações de automação de fábrica (OSWAL; SARAVANAKUMAR, 2021) e planejamento de tarefas de robôs (MELCHIORRE et al., 2022). Está presente tanto na área acadêmica quanto na industrial, tornando-se um robusto e amplamente utilizado simulador e controlador de robôs (ROHMER; SINGH; FREESE, 2013).

O CoppeliaSim tem Lua com sua principal linguagem de programação, pela qual é possível manipular todos os elementos de uma dada cena 3-D simulada, além de incorporar outros scripts, extensões e plug-ins. Mais essencialmente, Lua permite a interação com clientes remotos de API (*Application Programming Interface* ou interface de programação de aplicativos), trazendo diferentes linguagens de programação para atuar em conjunto dela (JIMÉNEZ et al., 2020), a mais proeminente sendo Python.

Python é uma linguagem de programação de alto nível, bastante popular pela sua versatilidade e sintaxe simples. Ela conta com uma variedade de bibliotecas e *frameworks* que a tornam acessível para diferentes campos de estudo. Destacam-se para associação com este trabalho: *matplotlib*, uma biblioteca de visualização de dados, oferecendo recursos para criar gráficos e *plots* de alta qualidade; *numpy*, que fornece suporte para arrays multidimensionais e operações matemáticas eficientes, permitindo cálculos numéricos rápidos; e *pandas*, voltada para análise de dados e que possui estruturas de dados, chamada de *DataFrames*, flexíveis e eficientes para manipulação dos mesmos.

2.6 Trabalhos relacionados

Esta seção apresenta uma revisão de trabalhos anteriores que abordaram o problema de localização de robôs móveis usando filtro de Kalman mediantes técnicas de *scan-matching* e/ou algoritmos bioinspirados.

Na Tabela 2.1 apresentam-se dez referências que abordam diversos aspectos da

robótica móvel. Esses estudos foram selecionados com base em sua relevância e contribuições para o campo. Além disso, no final da tabela, destacam-se os aspectos distintivos deste trabalho em relação aos demais. Essa análise permite identificar as lacunas que este estudo pretende preencher.

Tabela 2.1 – Trabalhos correlatos

Referência	**Técnicas**	**Abordagem**	**Plataforma**	**Sensor utilizado**
Schiele & Crowley (SCHIELE; CROWLEY, 1994)	KF e EKF	Grides de ocupação com técnicas de *matching*	Ambiente físico	Sensor de distância ultrassônico
Gutmann & Schlegel (GUTMANN; SCHLEGEL, 1996)	Modelo de regressão extendida de Cox e função de correlação cruzada	Módulo de autolocalização baseado em *scan-matching*	Ambiente físico	Sensor de distância a laser
Vahdat et. al (VAHDAT; NOURASHRAFODDIN; GHIDARY, 2007)	Localização de Monte Carlo	Localização global baseada em DE e PSO	MATLAB 2006	Não especificado
Suliman et. al (SULIMAN; CRUCERU; MOLDOVEANU, 2010)	KF e EKF	Estudo comparativo das técnicas aplicadas em rastreamento de posição	Ambiente virtual	Não especificado
Chen et. al (CHEN; HU; MCDONALDMAIER, 2012)	EKF	Aplicado em um mapa de características com leituras de sensor em 180º	Ambiente físico	Sensor de distância a laser
Pinto et. al (PINTO; MOREIRA; COSTA, 2013)	EKF	*Map-matching* feito pelo PSO	Ambiente físico	Sensores de distância infravermelhos
Rajurkar et. al (RAJURKAR et al., 2016)	Controle *fuzzy* e algoritmo genético	Auxiliado por visão computacional	Ambiente físico	Sensor visual Kinect Xbox360
Neto et. al (NETO et al., 2019)	Algoritmo de localização bioinspirado em morcegos	Metodologia de "sequestro de robô"	Gazebo	Sensor a laser scanner SICK LMS-200
Ajeil et. al (AJEIL et al., 2020)	Algoritmo de busca local	Planejamento de trajetória usando uma técnica de PSO-MFB	MATLAB R2015a	Não especificado
Este estudo	KF	*Scan-matching* auxiliado por algoritmos bioinspirados	CoppeliaSim	Sensor infravermelho Sharp GP2Y0A21YK0F

Adicionalmente, com o intuito de verificar a especificidade das abordagens adotadas neste trabalho, foi elaborado um levantamento bibliográfico dos últimos dez anos com o auxílio do software *VOSviewer* e da base de dados *Scopus*.

Realizando a busca com os termos "robótica móvel"e "filtro de Kalman", obtiveram-se centenas de artigos em periódicos e conferências, demonstrando a ampla adoção do filtro de Kalman em aplicações de robótica móvel. Quando adicionado o termo "scan-matching", foram obtidas cinco saídas, evidenciando que a técnica tem sido relativamente pouco explorada em aplicações de localização ou navegação de robôs móveis. Finalmente, ao incluir

o termo "bioinspirados", não foram encontradas referências bibliográficas, o que reforça a lacuna existente nessa área específica de pesquisa.

3 Metodologia

Este capítulo desenvolverá sobre as técnicas e metodologias aplicadas para a realização dos objetivos estabelecidos para o presente trabalho. Primeiramente, na seção 3.1, será mostrado o ambiente virtual desenvolvido para a realização das simulações computacionais, bem como, a partir dele, são feitos os processamentos das medições do sensor empregado (seção 3.2). Em seguida, a seção 3.3 verifica como foi modelada a cinemática da movimentação do robô, fazendo isso em função de um erro de desbalanceamento das rodas. Tal erro, torna-se, na seção 3.6, o objeto que o algoritmo de localização deseja aferir, empregando assim duas técnicas importantes de robótica móvel: *scan-matching* e filtro de Kalman, as quais serão explicadas nas seções posteriores (3.5 e 3.7, respectivamente).

3.1 Ambiente virtual e especificações de aspectos da simulação

Para este trabalho, foi aproveitada parte da estrutura de simulação criada para o *Projeto Maria* (TRIANA, 2022). Nele, foi utilizado um chassi de robô móvel de 17 cm de diâmetro, de tração diferencial, equipado com seis sensores infravermelhos com referência Sharp GP2Y0A21YK0F. O objetivo final do robô *Maria* era fazer o controle da sua movimentação baseado no comportamento treinado por redes neurais artificiais.

O presente trabalho tem foco no rastreamento da posição do robô e balanceamento do deslocamento das rodas (fonte do erro de posição) usando *scan-matching* e filtro de Kalman, sendo o sensoriamento realizado por leituras de distância feitas a cada meio grau, com o sensor centralizado no chassi. Para um escaneamento completo são feitas 720 medições, completando uma circunferência. Uma vez feitas as leituras, elas são passadas para uma API em Python que faz o tratamento e análise dos dados. Para a execução dessas tarefas foram determinados os seguintes requisitos:

a) chassi do robô virtual com 17 cm de diâmetro e tração diferencial para atender as especificações do *Maria*, no qual o método de localização pode ser aplicado posteriormente;

b) único sensor infravermelho conectado a um eixo transversal ao robô e que faça uma rotação completa. Também foi mantida a referência Sharp GP2Y0A21YK0F, capaz de medir distâncias confiáveis de um metro;

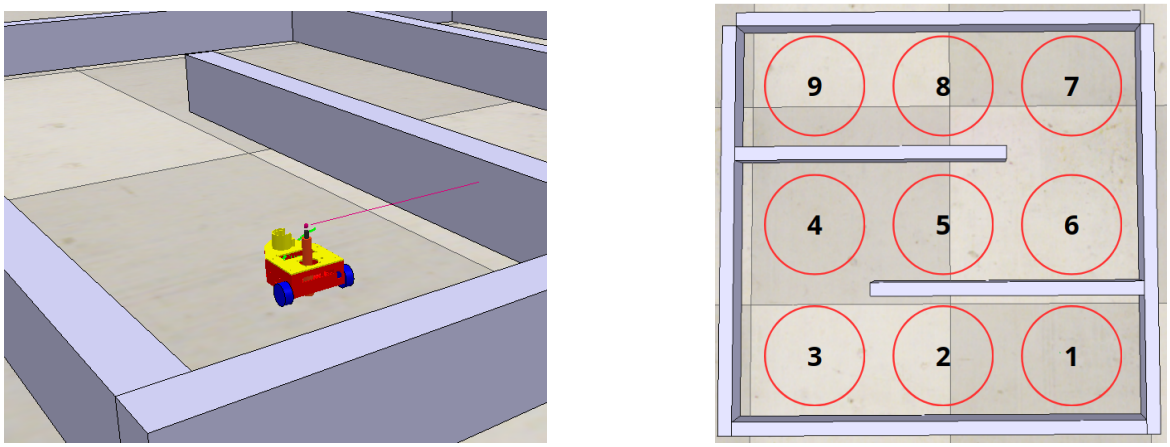
c) variedade nos ambientes virtuais, de forma que sejam simuladas as condições reais nas quais um robô móvel pode se encontrar;

d) interface que permita comunicação remota com aplicações executadas em Python;

e) a técnica de *scan-matching* aplicada deve ser escalável para a sistemas embarcados, devendo ter um custo computacional condizente com o processamento de microcontroladores/minicomputadores;

f) no problema de localização proposto, o robô conhece sua posição inicial e possui armazenado em memória o mapa do seu entorno, o qual é estático.

No software de simulação, então, foi prototipado um cenário estruturado 3-D (Figura 3.5a) em formato labiríntico 3x3 m, com paredes em volta do robô, de forma que seu sensor, disposto centralmente no chassi, pudesse escanear objetos ao seu alcance. Ainda no escopo do CoppeliaSim, foi programado em Lua uma interface de controle para o robô, de forma que ele pode ser movido interativamente pelas setas do teclado do computador onde o programa está sendo executado. O script também abre a conexão para aplicações externas.



(a) Vista panorâmica do robô

(b) Vista superior com posições numeradas

Figura 3.5 – Robô implementado no CoppeliaSim

3.2 Processamento das medições para *scan-matching*

Quando a simulação é iniciada, estabelece-se uma comunicação entre o CoppeliaSim e o código em Python. Em seguida, o eixo do sensor IR começa a rotacionar, capturando a cada meio grau uma medição de distância até o objeto mais próximo (d). As medições são armazenadas dentro de um *DataFrame*, associadas a seu ângulo (ϕ).

É importante salientar que os escaneamentos de distância serão a principal ferramenta de localização do robô, permitindo extrair as informações para aferimento da sua posição atual para um eventual balanceamento da velocidade das rodas. Ressalta-se que essa diferença de velocidade é uma das principais causas de erros entre a posição esperada e a posição real no deslocamento de um robô.

Utilizando as bibliotecas *numpy* e *matplotlib* é possível fazer uma representação visual das medições de distância realizadas. Na Figura 3.6, cada distância assume uma coordenada cartesiana em centímetros, conforme a Equação 3.1, e então é desenhada como um ponto. Distâncias superiores a 0,9 m e inferiores a 0,1 m não são consideradas por estarem fora de alcance do sensor.

DataFrame de distâncias	
Ângulo de medição (graus)	Distância medida (m)
0	0.4430156350
1	0.4430097639
2	0.4429304301
…	…
358	0.4430543482
359	0.4431045949

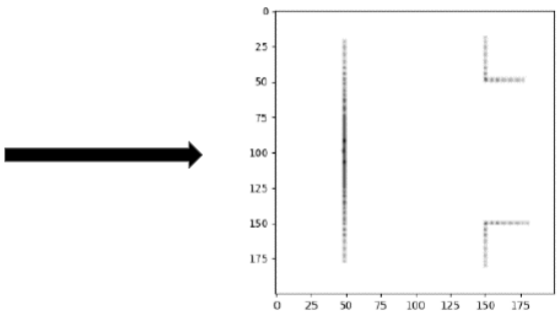


Figura 3.6 – *DataFrame* representado visualmente

$$(x, y) = (d cos\phi, d sen\phi). \tag{3.1}$$

3.3 Modelagem odométrica da movimentação do robô

Com essa ferramentas ao dispor, deseja-se navegar no mapa da Figura 3.5, indo da posição 1 a 9, realizando os escaneamentos ao longo do caminho. Para tal, será necessário operar apenas dois tipos de movimento básicos: translação para a frente e rotação de 90 graus (nos sentidos horário e anti-horário, efetuadas nas posições 3, 4, 6 e 7 para mudar a orientação do robô). Cada posição está a um metro da seguinte, distância que idealmente o robô deverá se deslocar a cada iteração.

Antes de explicar o funcionamento do algoritmo de *scan-matching*, é necessário definir as etapas que levarão ao seu uso. Tais etapas se referem a própria movimentação do robô, pois, se tratando de uma simulação em um ambiente ideal, deve haver um erro inserido manualmente para emular condições reais, afetando seu deslocamento e diferindo a posição em que o robô se encontra daquela esperada ao fim do movimento, situação a partir da qual pode ser aplicada a técnica de *scan-matching*.

Para modelar o erro em cada um dos tipos de movimento deve-se considerar as condições para executá-lo. No caso translacional, para se mover em linha reta, é preciso que as rodas do robô de tração diferencial realizem o mesmo deslocamento ($\Delta S_d = \Delta S_e = \Delta S$) por unidade de tempo, não tendo assim variação da orientação na Equação 2.1. Por outro

lado, considerar que esse deslocamento está desbalanceado, implica que as rodas estão se deslocando mais ou menos do que configurado pelo controlador, mudando assim a orientação do robô e sua posição final em (x, y).

$$\begin{aligned} \Delta S_d &= K_d \Delta S, \\ \Delta S_e &= K_e \Delta S. \end{aligned} \tag{3.2}$$

Seja o deslocamento das rodas modelado pela Equação 3.2, onde K_d e K_e são os fatores de desbalanceamento do deslocamento das rodas direita e esquerda, respectivamente, no intervalo $[0{,}95, 1{,}05]$. Dessa forma, K_d e K_e representam o desvio para mais ou para menos do valor de deslocamento esperado, podendo cada uma girar entre 95% a 105% do valor de ΔS. Nessa abordagem, o desbalanceamento será tratado como um erro sistemático, podendo ocorrer por uma variedade de fatores - como o acionamento mais rápido de um dos motores, o desgaste do pneu de uma das rodas, uma distribuição desigual da carga, etc. -, o que faz com que o robô realize um movimento curvilíneo ao invés de retilíneo. Deste modo, K_d e K_e funcionarão como valores de entrada da simulação, sendo seus intervalos escolhidos considerando as dimensões do próprio ambiente de simulação a fim de evitar que se extrapolem os limites do mapa. Com esses parâmetros definidos, pode-se rescrever a Equação 2.1 conforme abaixo.

$$\begin{bmatrix} x'_k \\ y'_k \\ \theta'_k \end{bmatrix} = \mathbf{p}_{k-1} + \begin{bmatrix} \frac{(K_d+K_e)\Delta S}{2} cos\left(\theta_{k-1} + \frac{(K_d-K_e)\Delta S}{2b}\right) \\ \frac{(K_d+K_e)\Delta S}{2} sen\left(\theta_{k-1} + \frac{(K_d-K_e)\Delta S}{2b}\right) \\ \frac{(K_d-K_e)\Delta S}{b} \end{bmatrix}, \tag{3.3}$$

$$\mathbf{p'_k} = \mathbf{p_{k-1}} + \Delta\mathbf{p'_k}(K_d, K_e).$$

Em uma rotação, para que o robô gire sem se modificar sua posição em (x, y), as rodas devem efetuar o mesmo deslocamento $(\Delta S = \theta\frac{b}{2})$ em sentidos opostos, isso é, $\Delta S_d = -\Delta S_e$. Na Equação 2.1, seja o deslocamento das rodas dado por $\Delta S_d = K_d \Delta S$ e $\Delta S_e = -K_e \Delta S$, obtém-se a seguinte modelagem de pose para o movimento rotacional.

$$\begin{bmatrix} x'_k \\ y'_k \\ \theta'_k \end{bmatrix} = \mathbf{p_{k-1}} + \begin{bmatrix} \frac{(K_d-K_e)\theta b}{4} cos\left(\theta_{k-1} + \frac{(K_d+K_e)\theta}{4}\right) \\ \frac{(K_d-K_e)\theta b}{4} sen\left(\theta_{k-1} + \frac{(K_d+K_e)\theta}{4}\right) \\ \frac{(K_d+K_e)\theta}{2} \end{bmatrix}, \tag{3.4}$$

$$\mathbf{p'_k} = \mathbf{p_{k-1}} + \Delta\mathbf{p'_k}(K_d, K_e).$$

Em ambos os casos, são observadas variações da pose esperada para a estimada, as quais podem ser mensuradas pela pose relativa entre elas (**D**), cuja notação será importante para as seções seguintes, onde $\mathbf{D} = [\Delta x, \Delta y, \Delta\theta]^T$.

$$\begin{aligned} \mathbf{D_k} &= \mathbf{p'_k} - \mathbf{p_k} \\ \mathbf{D_k} &= \mathbf{p_{k-1}} + \Delta\mathbf{p'_k} - (\mathbf{p_{k-1}} + \Delta\mathbf{p_k}) \\ \mathbf{D_k} &= \Delta\mathbf{p'_k} - \Delta\mathbf{p_k}. \end{aligned} \tag{3.5}$$

3.4 Algoritmo de localização

A partir deste momento, é necessário afirmar que K_d e K_e serão os objetos que busca-se estimar para que o robô funcione corretamente, realizando assim a trajetória da posição 1 até a 9 com o menor erro possível. Logo, em cada simulação, o robô será inicializado com um dado conjunto $\mathbf{K} = [K_d, K_e]^T$, desconhecido por ele e que afetarão seu deslocamento, condição que o algoritmo de *scan-matching* e o filtro de Kalman tentarão compensar, após cada iteração de movimento translacional.

Para obter K_d e K_e em função da variação de pose esperada ($\Delta\mathbf{p}$) e da estimada ($\Delta\mathbf{p'} = [\Delta x', \Delta y', \Delta\theta']^T$), deve-se isolar os fatores na Equação 3.3, obtendo assim as seguintes relações:

$$\Delta S' = \sqrt{\Delta x'^2 + \Delta y'^2} = \frac{(K_d + K_e)}{2}\Delta S, \tag{3.6}$$

$$\Delta\theta' = \frac{(K_d - K_e)}{b}\Delta S, \tag{3.7}$$

$$K_d + K_e = \frac{2\Delta S'}{\Delta S}, \tag{3.8}$$

$$K_d - K_e = \frac{\Delta\theta' b}{\Delta S}, \tag{3.9}$$

onde ΔS é o módulo do deslocamento de $\Delta\mathbf{p}$. Por fim, pelas equações 3.8 e 3.9,

$$K_d = \frac{\Delta S' + 0{,}5\Delta\theta' b}{\Delta S}, \tag{3.10}$$

$$K_e = \frac{\Delta S' - 0{,}5\Delta\theta' b}{\Delta S}, \tag{3.11}$$

$$\mathbf{K}(\Delta\mathbf{p'}) = \begin{bmatrix} K_d \\ K_e \end{bmatrix}. \tag{3.12}$$

Desta forma, é possível encontrar **K** a partir do conjunto $\Delta\mathbf{p'} = [\Delta x', \Delta y', \Delta\theta']^T$, o qual será obtido pelo algoritmo de *scan-matching*. Os detalhes da implementação da

técnica de *scan-matching* e do KF, serão explicitados nas seções seguintes; por enquanto, se descreverão os procedimentos da simulação utilizando seus resultados.

a) Com a pose inicial $\mathbf{p_1} = [0, 0, 90°]^T$ Figura 3.7a, a trajetória planejada indica que o robô deve chegar na posição 2 com pose $\mathbf{p_2} = [0, 1, 90°]^T$, deslocando-se com uma translação de 1 metro ($\Delta\mathbf{p_2}$). Entretanto, devido aos fatores K_d e K_e, pela Equação 3.3, ele chega a $\mathbf{p'_2} = [x'_2, y'_2, \theta'_2]^T$ (Figura 3.7b);

b) Nesse momento, $\mathbf{p_2}$ e $\mathbf{p'_2}$ representam a pose esperada e a estimada do robô no ambiente, respectivamente. Para evitar que na próxima iteração o robô se desloque para a posição 3 partindo de uma pose errônea (o que ampliaria o erro), fazem-se leituras de escaneamento do entorno de $\mathbf{p'_2}$, que comparadas com as do entorno de $\mathbf{p_2}$ (previamente armazenadas em memória), o *scan-matching* gera uma estimativa de pose relativa entre elas, isso é, $\mathbf{D_2} = \Delta\mathbf{p'_2} - \Delta\mathbf{p_2}$ (Equação 3.5), onde $\mathbf{D_2} = [\Delta x, \Delta y, \Delta\theta]^T$;

c) Seja $\Delta\mathbf{p'_2} = \Delta\mathbf{p_2} + \mathbf{D_2} = [\Delta x', \Delta y', \Delta\theta']^T$, calculam-se medições dos valores de K_d e K_e (Equação 3.10 e Equação 3.11), que são inseridas em um filtro de Kalman;

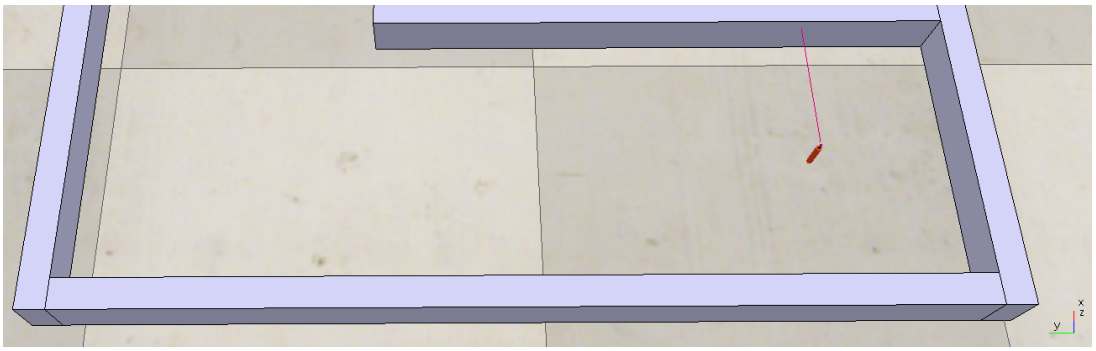
d) As estimativas $\widehat{x}_{K_d}$ e $\widehat{x}_{K_e}$ saídas do KF serão compensadas no movimento das rodas para a próxima iteração, fazendo com que elas andem mais uniformemente. Isso é, se a estimativa da roda direita for $\widehat{x}_{K_d} = 0{,}95$, no próximo deslocamento ela buscará se mover $\frac{\Delta S_d}{0{,}95} \approx 1.0526\Delta S_d$. É importante salientar que com $\widehat{x}_{K_d}$ e $\widehat{x}_{K_e}$ também se obtém uma estimativa da posição ($\widehat{x}_x$ e $\widehat{x}_y$);

e) Para a iteração seguinte, se considerará $\mathbf{p'_2}$ como pose de origem (Figura 3.7c) em direção à posição 3. Portanto, deve-se ajustar a orientação para $\theta'_2 - \Delta\theta - \alpha$ (Figura 3.7c), onde α é um ângulo de compensação para o robô chegar a próxima posição com um movimento linear;

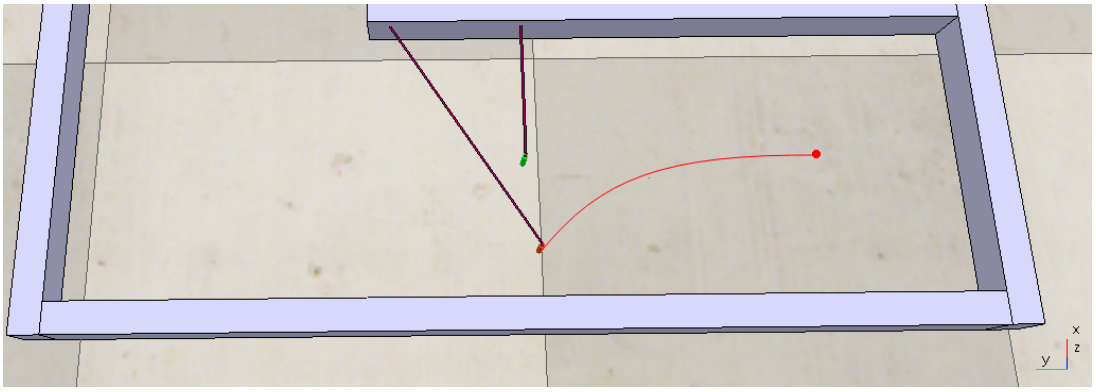
f) Os passos anteriores se repetem na posição 3 (Figura 3.7d). Nota-se que o *scan-matching* e o KF são aplicados apenas após movimentos de translação. Não se utiliza o processo descrito após rotações, pois o erro de pose obtido nesses casos é pequeno.

3.5 Algoritmo de *scan-matching* auxiliado por algoritmos bioinspirados

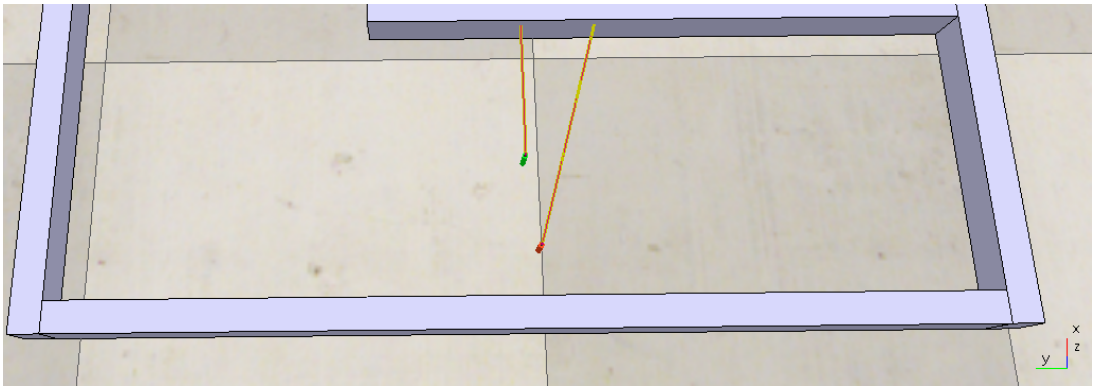
Como visto anteriormente, os escaneamentos dos entornos das posições de interesse são as ferramentas usadas para produzir estimativas que permitirão que o robô opere corrigindo o desbalanceamento do deslocamento de suas rodas e ajuste sua posição. Na seção 3.2, foi explorado como esses escaneamentos são capturados e processados em *DataFrames*, logo, busca-se entender como eles são aplicados dentro do escopo deste problema.



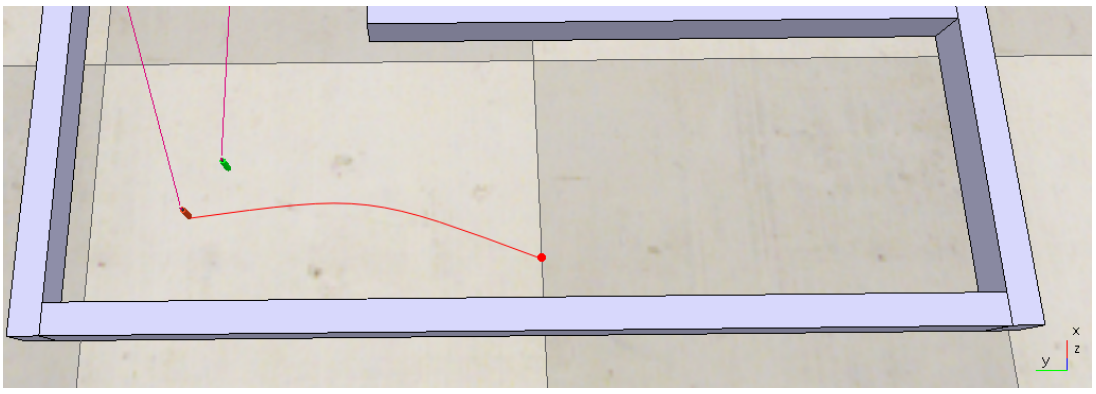
(a) Pose inicial $\mathbf{p_1}$



(b) $\mathbf{p_2}$ e $\mathbf{p'_2}$ em verde e vermelho, respectivamente



(c) $\mathbf{p'_2}$ ajustada para deslocamento para $\mathbf{p_3}$



(d) $\mathbf{p_3}$ e $\mathbf{p'_3}$ em verde e vermelho, respectivamente

Figura 3.7 – Movimento do robô entre as posições 1, 2 e 3

A técnica de *scan-matching* consiste em: dados dois *DataFrames* de diferentes poses, $\mathbf{p}$ e $\mathbf{p}'$, deve-se aferir a pose relativa ($\mathbf{D}$) entre elas em termos de um deslocamento horizontal, vertical e angular, notação dada por $(\Delta x, \Delta y, \Delta\theta)$. Na prática, ocorre uma mudança de base de coordenadas, sendo primeiro feita a rotação e, em seguida, a translação, conforme mostra a Figura 3.8 e a Equação 3.13.

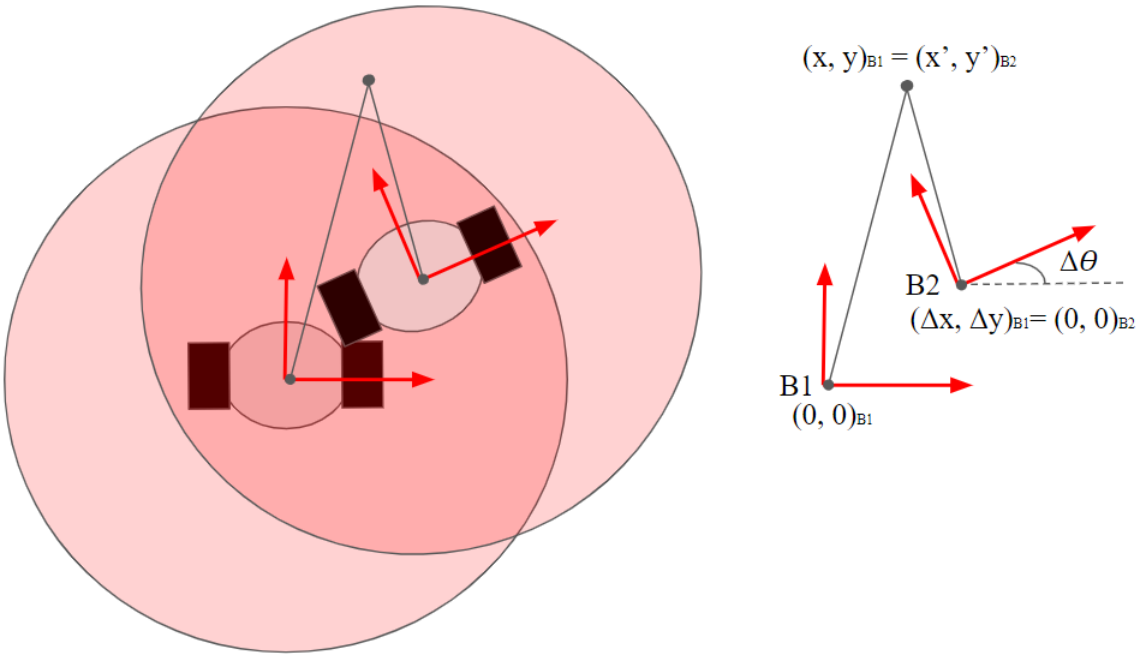


Figura 3.8 – Mudança de base das coordenadas.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) & \Delta x \\ \sin(\Delta\theta) & \cos(\Delta\theta) & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.13}$$

O *scan-matching* é auxiliado pelo algoritmo bioinspirado PSO, avaliando as soluções que a meta-heurística fornece. O processo segue os seguintes passos:

a) O sistema recebe dois *DataFrames* (M_1 e M_2) com bases de coordenadas, B_1 e B_2, transladadas e rotacionadas entre si pelo fator $(\Delta x, \Delta y, \Delta\theta)$;

b) O algoritmo bioinspirado é inicializado com S partículas $\vec{z}_i = (\Delta x_i, \Delta y_i, \Delta\theta_i)$. Para encontrar a solução que aproxime M_1 e M_2, as coordenadas de M_2 são modificadas pela Equação 3.13 com os valores de $\vec{z}_i$, gerando um novo *DataFrame* M_i na base B_1 (Equação 3.14);

c) Cada M_i terá um resultado de otimização associado, representando o quão similar ele está com o *DataFrame* M_1. Esse valor é definido pela função custo do problema de otimização, mostrado na Equação 3.15. A equação calcula o erro absoluto de distância termo a termo dos elementos dos *DataFrames*. Nota-se que, ao

fazer a mudança de base de B_2 para B_1, algumas informações serão perdidas, podendo estarem no escaneamento de B_2, mas quando transformadas na base B_1 a distância resultante torna-se superior a 0,9 m (limite de distância do sensor). Essas distâncias são desconsideras e a quantidade restante é a variável $N_{validas}$;

d) Conclui-se o *scan-matching* ao fim de K iterações. Nesta etapa, o algoritmo bioinspirado retorna a melhor solução Z_S.

$$\begin{aligned} d' &= \sqrt{x'^2 + y'^2} \\ \phi' &= tg^{-1}(\frac{y'}{x'}). \end{aligned} \tag{3.14}$$

$$minf_i = \frac{1}{N_{validas}} \sum_{n=1}^{N_{validas}} \sqrt{(x[n] - x'[n])^2 + (y[n] - y[n])^2} \tag{3.15}$$

3.6 Parâmetros do Algoritmo PSO

A parametrização é uma etapa importante de algoritmos de otimização pois faz o controle da resposta exigida do sistema. No caso do PSO, o número de partículas S impacta na diversidade das soluções, um maior número de iterações K atrasa a convergência, permitindo mais combinações serem feitas. Em contrapartida, o custo dessa maior variabilidade é a necessidade de um maior poder computacional para processar todas as operações.

Os coeficientes cognitivo (c_1) e social (c_2), por sua vez, indicam o quanto cada partícula tende em direção a sua própria experiência e o quanto é levada pela experiência do conjunto. A maior parte das implementações de PSO tendem a manter seus valores iguais, sendo $c_1 = c_2 = 2{,}05$ (CARLISLE; DOZIER et al., 2001).

A Tabela 3.2 apresenta os parâmetros escolhidos para o PSO. Os parâmetros foram calibrados para que o sistema tivesse uma saída otimizada e, ainda assim, rápida e computacionalmente acessível para atender o requisito de portabilidade da técnica de *scan-matching* para aplicações embarcadas.

Tabela 3.2 – Parâmetros do algoritmo PSO

Parâmetros	Valores adotados
Número de partículas S	10
Número de iterações K	50
Coeficiente cognitivo c_1	2,05
Coeficiente social c_2	2,05

3.7 Algoritmo de Localização por Filtro de Kalman

3.7.1 Filtro de Kalman adaptado para rastreamento de posição

O filtro de Kalman é uma técnica clássica de rastreamento de posição, conseguindo incrementar a confiança de suas previsões ao longo do tempo, representando essa confiança por meio de distribuições de probabilidade gaussianas. No caso deste estudo, o filtro de Kalman implementado é unidimensional, sendo aplicado individualmente para cada elemento de $\mathbf{K}$, os tratando como um problema linear. Para tal implementação, é importante revisar algumas especificações do problema.

Conforme mencionado na subseção 2.3.3, o robô conhece a posição inicial, além de dispor do mapa do ambiente em que está presente e de sensores odométricos que permitem estimar o possível deslocamento realizado. Portanto, dadas tais características, na etapa de previsão, estabelece-se que o robô sempre possui uma predição inicial da sua pose, denotada por $\mathbf{p_{pred}}$ (previsão de posição).

De acordo com a Figura 3.9, a cada movimentação no ambiente real, a pose obtida será potencialmente diferente de $\mathbf{p_{pred}}$, uma vez que sensores físicos acumulam erros aleatórios e sistemáticos. Com isso, o robô realiza o escaneamento do seu entorno para extrair as informações no *DataFrame* M_1 (previsão de observação), enquanto, na etapa de observação, o mapa do ambiente (presente na memória do robô) é escaneado internamente centralizado em $\mathbf{p_{pred}}$, gerando M_2. Assim, na etapa de correspondência o *scan-matching* utiliza os dois escaneamentos de posição que podem ser comparados para encontrar a pose relativa entre eles ($\mathbf{D} = [\Delta x, \Delta y, \Delta\theta]^T$), e, com isso, o deslocamento de fato obtido $\Delta\mathbf{p_{med}} = \Delta\mathbf{p'}$ (Equação 3.5).

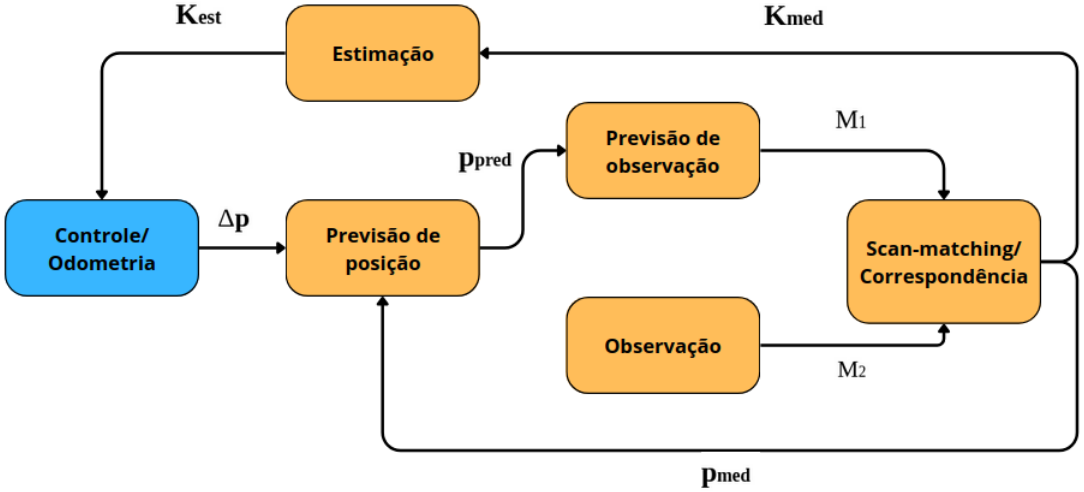


Figura 3.9 – Filtro de Kalman representado em blocos

Como propõe a metodologia deste trabalho, além de aprimorar a estimativa de posição do robô por meio das medições, também é possível compensar o fator de desbalanceamento das rodas, denotado por $\mathbf{K_{med_k}} = [K_d, K_e]^T$ (Figura 3.9). Logo, forão feitas adaptações ao KF

da Figura 2.3. Primeiramente, é usada a medição $\mathbf{p}_{\mathbf{med}_k}$ para corrigir a pose $\mathbf{p}_{\mathbf{pred}_k}$ ao invés da estimativa. Isso só é possível pois o algoritmo de *scan-matching* fornece uma acurácia (como será visto na seção 4.1) que permite usar a pose medida de referência para a operação do robô. Em segundo lugar, a etapa de estimação será destinada para gerar uma estimativa $\mathbf{K}_{\mathbf{est}_k}$ que compensará o funcionamento desigual das rodas para o movimento seguinte. Apenas a partir de $\mathbf{K}_{\mathbf{est}_k}$ se calculará $\mathbf{p}_{\mathbf{est}_k}$ (Equação 3.3), a informação de saída desejada do problema de localização. Por fim, $\mathbf{p}_{\mathbf{med}_k}$ é somada novamente com deslocamento $\Delta\mathbf{p}_{\mathbf{k+1}}$, dando vez a uma nova predição inicial $\mathbf{p}_{\mathbf{pred}_{k+1}}$.

3.7.2 Propagação de incertezas no filtro de Kalman

O KF dita que $\mathbf{p}_{\mathbf{est}}$ é sempre uma estimativa da pose real, ou seja, uma distribuição gaussiana multivariável (Equação 2.11) indicando a confiança do robô estar de fato naquela pose. Dentro de um espaço discreto, cada posição terá uma distribuição de probabilidade $P(\mathbf{p}_{\mathbf{est}})$ em que, sendo somada recursivamente, incrementa a confiança das posições estimadas com o auxílio das medições do *scan-matching*.

Todas as medições $\Delta\mathbf{p}_{\mathbf{med}}$ possibilitam calcular $\mathbf{K}_{\mathbf{med}}$ (Equação 3.12), tendo K_d e K_e valores médios de medição e variâncias de erro σ^2, onde σ é o desvio padrão da medição. Na etapa de estimação do KF, um novo $\mathbf{K}_{\mathbf{med}}$ ajuda a compor a estimativa, atualizando o estado de $\mathbf{K}_{\mathbf{est}} = [K_d, K_e]^T$ pela Equação 2.8, e sua incerteza pela Equação 2.9 (para a simulação, se considerará o ruído de processo R nulo).

$$\mathbf{K}_{\mathbf{est}} = \begin{bmatrix} K_d \\ \\ K_e \end{bmatrix} = \begin{bmatrix} \hat{x}_{K_d} \pm \sigma_{K_d} \\ \\ \hat{x}_{K_e} \pm \sigma_{K_e} \end{bmatrix} \tag{3.16}$$

Com o estado atualizado de $\mathbf{K}_{\mathbf{est}}$ na Equação 3.16, é calculada a variação de pose $\Delta\mathbf{p}_{\mathbf{est}}$ (Equação 3.3). Por fim, para estimar a pose na iteração k, temos:

$$\mathbf{p}_{\mathbf{est}_k} = \mathbf{p}_{\mathbf{est}_{k-1}} + \Delta\mathbf{p}_{\mathbf{est}_k}. \tag{3.17}$$

Note que assim como $\mathbf{K}_{\mathbf{est}}$ possui desvios padrão para K_d e K_e, $\mathbf{p}_{\mathbf{est}}$ também deve os ter para si. Seja:

$$\mathbf{p}_{\mathbf{est}_k} = \begin{bmatrix} x_k \\ \\ y_k \\ \\ \theta_k \end{bmatrix} = \begin{bmatrix} \hat{x}_{x_k} \pm \sigma_{x_k} \\ \\ \hat{x}_{y_k} \pm \sigma_{y_k} \\ \\ \hat{x}_{\theta_k} \pm \sigma_{\theta_k} \end{bmatrix}. \tag{3.18}$$

Para calcular os desvios padrões σ_x, σ_y e σ_θ, faz-se a propagação do erro de $\mathbf{K_{est}}$. Usando a Equação 3.7, tem-se:

$$\begin{aligned} \hat{x}_{\theta_k} &= \frac{(\hat{x}_{K_d} + \hat{x}_{K_e})\Delta S}{b}, \\ \sigma_{\theta_k} &= \frac{\Delta S}{b}\sqrt{\sigma_{K_d}^2 + \sigma_{K_e}^2}. \end{aligned} \tag{3.19}$$

Por sua vez, σ_x, σ_y possuem expressões mais complexas, de forma que, para demonstração, seus fatores serão decompostos. Assim, através da Equação 3.6, obtém-se a seguinte expressão:

$$\begin{aligned} \Delta S' &= \frac{(\hat{x}_{K_d} + \hat{x}_{K_e})\Delta S}{2}, \\ \sigma_{\Delta S'} &= \frac{\Delta S}{2}\sqrt{(\sigma_{K_d}^2 + \sigma_{K_e}^2)}, \end{aligned} \tag{3.20}$$

onde $\Delta S'$ é a estimativa do deslocamento. Pela Equação 3.3 calculam-se as expressões de incerteza para as coordenadas x e y,

$$\begin{aligned} A &= \cos\left(\hat{x}_{\theta_{k-1}} + \frac{\hat{x}_{\theta_k}}{2}\right) \\ \sigma_A &= \left|\sin\left(\hat{x}_{\theta_{k-1}} + \frac{\hat{x}_{\theta_k}}{2}\right)\right| \sqrt{\sigma_{\theta_{k-1}}^2 + \left(\frac{1}{2}\sigma_{\theta_k}\right)^2} \\ \hat{x}_{x_k} &= \hat{x}_{x_{k-1}} + \Delta S' \cdot A \\ \sigma_{x_k} &= \sqrt{\sigma_{x_{k-1}}^2 + (A \cdot \sigma_{\Delta S'})^2 + (\Delta S' \cdot \sigma_A)^2} \end{aligned} \tag{3.21}$$

$$\begin{aligned} B &= \sin\left(\hat{x}_{\theta_{k-1}} + \frac{\hat{x}_{\theta_k}}{2}\right) \\ \sigma_B &= \left|\cos\left(\hat{x}_{\theta_{k-1}} + \frac{\hat{x}_{\theta_k}}{2}\right)\right| \sqrt{\sigma_{\theta_{k-1}}^2 + \left(\frac{1}{2}\sigma_{\theta_k}\right)^2} \\ \hat{x}_{y_k} &= \hat{x}_{y_k} + \Delta S' \cdot B \\ \sigma_{y_k} &= \sqrt{\sigma_{y_{k-1}}^2 + (B \cdot \sigma_{\Delta S'})^2 + (\Delta S' \cdot \sigma_B)^2} \end{aligned} \tag{3.22}$$

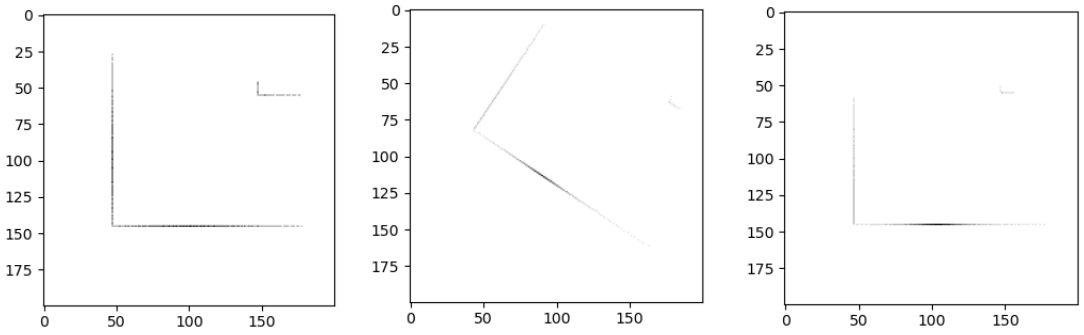
É importante ressaltar que a estimativa de pose é calculada retroativamente a cada iteração, ou seja, quando $\mathbf{K_{est_k}}$ é atualizado, as poses anteriores à iteração k são recalculadas em função da nova estimativa.

4 Análise de resultados

4.1 Avaliação do algoritmo de *scan-matching*

Para demonstração visual, um teste consiste em, inicialmente, realizar um escaneamento em uma posição de referência, coletando o *DataFrame* M_1 (vide Figura 4.10a). Em seguida, as distâncias coletadas de M_1 tem sua posição modificada pelo termo $(\Delta x, \Delta y, \Delta\theta)$, conforme o conjunto de equações 3.13 e 3.14, gerando um novo *DataFrame* M_{exp}, como mostrado na Figura 4.10b. Por fim, M_1 e M_{exp} são inseridas no algoritmo *scan-matching*, no qual o PSO retornou a melhor restauração (*DataFrame* M_S) encontrada após sua execução.

Nos testes, (K_d, K_e) é o conjunto de variáveis de entrada da simulação, capaz de gerar variação na pose nos intervalos $\Delta x \in [-28{,}99\text{ cm}, 28{,}99\text{ cm}]$, $\Delta y \in [-5\text{ cm}, 5\text{ cm}]$ e $\Delta\theta \in [-34{,}32°, 34{,}32°]$. Na Figura 4.10c, é mostrada a representação visual de M_S, gerada a partir de uma solução encontrada pelo PSO, levando em conta que a variação forçada para M_{exp} foi de $(\Delta x = -0.2898955, \Delta y = -0.04294181, \Delta\theta = 33.70340112)$ na posição 3 da Figura 3.5. Dessa forma, a saída do PSO corrige M_{exp} aplicando o resultado simetricamente negativo em M_S.



(a) *DataFrame* M_1 (b) *DataFrame* M_{exp} (c) *DataFrame* M_S

Figura 4.10 – Representação visual dos *DataFrames*

Para uma análise mais geral do algoritmo, foram escolhidas amostras de $(K_d, K_e)_{entrada}$, as quais tiveram o processo de *scan-matching* repetido 6 vezes. As amostras foram combinações de valores de K_d e K_e entre 0,95 e 1,05, com intervalo de 0,01 entre si, totalizando 121 amostras. Para cada amostra, obteve-se as saídas em termos de $(K_d, K_e, \Delta x, \Delta y, \Delta\theta)_{saida}$. Normalizando os valores, calculou-se a acurácia dos resultados, que estão reportados na Tabela 4.3.

As medições foram feitas para as posições 1, 2 e 3 do ambiente da Figura 3.5. É possível verificar que as demais posições têm simetria com as escolhidas, sendo as posições 5 e 8 simétricas a 2; 4, 6 e 7 simétricas a 3; e a posição 9 simétrica a 1. Dessa forma, calculou-se

Tabela 4.3 – Tabela de resultados do *scan-matching*

Posições	$K_d(\%)$	$K_e(\%)$	$x(\%)$	$y(\%)$	$\theta(\%)$
1	$99{,}80 \pm 2{,}68$	$99{,}80 \pm 2{,}70$	$99{,}78 \pm 0{,}17$	$99{,}45 \pm 7{,}66$	$99{,}91 \pm 0{,}08$
2	$91{,}87 \pm 7{,}20$	$91{,}88 \pm 7{,}27$	$99{,}64 \pm 7{,}47$	$79{,}77 \pm 18{,}21$	$99{,}94 \pm 0{,}47$
3	$99{,}90 \pm 0{,}05$	$99{,}89 \pm 0{,}05$	$99{,}93 \pm 0{,}11$	$99{,}73 \pm 0{,}11$	$99{,}84 \pm 0{,}20$
Resultante	$97{,}13 \pm 5{,}80$	$97{,}13 \pm 5{,}84$	$99{,}80 \pm 4{,}37$	$92{,}84 \pm 14{,}68$	$99{,}89 \pm 0{,}31$

a acurácia resultante pela média ponderada das acurácias por posição, levando em conta as ocorrências de cada simetria. Os desvios padrão resultantes obtidos de K_d e K_e passarão a ser as incertezas de medição σ_{K_d} e σ_{K_e}, necessários para aplicação do KF.

Na Tabela 4.3, é necessário notar que os resultados de medição da coordenada y na posição 2 destoam das demais métricas. Esse comportamento também é observado quando, durante a simulação, o robô passa pelas posições 5 e 8. Isso ocorre devido aos escaneamentos dessas posições terem informações apenas na suas laterais, não havendo nenhuma detecção frontalmente ou atrás. Isso faz com que o algoritmo de *scan-matching* não consiga diferenciar com eficiência medições com Δx e $\Delta\theta$ próximos, o que prejudica a otimização pelo PSO e torna os resultados de Δy inconsistentes.

4.2 Avaliação do algoritmo de localização

Para avaliar o algoritmo de localização, deve-se testá-lo sob determinadas condições repetidas vezes, buscando "estressá-lo"de forma a verificar sua eficiência. Como visto anteriormente, o fator de desbalanceamento das rodas **K** dita o comportamento das mesmas, sendo a variável de entrada da simulação, cujos valores K_d e K_e buscam ser estimados e corrigidos para funcionamento ideal do robô. Ou seja, a avaliação do algoritmo proposto passa por submetê-lo a conjuntos de K_d e K_e em uma trajetória pré-definida, e ao fim dela, verificar se o robô foi capaz de percorrê-la com o menor erro possível. Para tal tarefa, escolheram-se os seguintes fatores:

- $(K_d, K_e) = (100\%, 100\%)$: comportamento ideal das rodas, sem deslizamentos ou diferenças no acionamento dos motores, proporcionando movimentos retilíneos. Busca-se por essas condições verificar se, dado um funcionamento perfeito do robô, o algoritmo de localização implementado pode comprometer a realização da trajetória;
- $(K_d, K_e) = (95\%, 100\%)$: roda direita se desloca 5% a menos que a esquerda, fazendo com que o robô penda levemente para sua direita;
- $(K_d, K_e) = (95\%, 95\%)$: as duas rodas estão se desloquem retilineamente, porém 5% a menos que o devido, ficando atrasado em relação a posição esperada;
- $(K_d, K_e) = (105\%, 105\%)$: as duas rodas também deslocam retilineamente, mas, nesse caso, 5% a mais que o devido, ficando adiantado em relação a posição esperada;

- $(K_d, K_e) = (105\%, 95\%)$: trata-se do caso extremo modelado para o algoritmo, sendo aquele que gera a maior variação angular entre a pose esperada e a obtida.

4.2.1 Avaliação do algoritmo no caso extremo

Os maiores esforços de simulação foram direcionados ao caso extremo de desbalanceamento das rodas $(K_d, K_e) = (105\%, 95\%)$, pelo entendimento de que, se o algoritmo de localização consegue se adaptar a ele, também se adaptará aos casos intermediários.

Durante a execução da simulação, foram coletados os pontos da trajetória esperada pelo robô, (x, y), e aquela realizada de fato, (Rx, Ry) (vide Figura 4.11). Pode-se observar que da posição 1 nas coordenadas $(0, 0)$ até a posição 9, coordenadas $(2, 2)$, o algoritmo proposto fez com que o robô, mesmo com o desbalanceamento em suas rodas, tente retornar para seu trajeto original, compensando o erro para as próximas iterações. Na Figura 4.11, observa-se que gradativamente o erro entre a posições é minimizado [1], indicando que o algoritmo de *scan-matching* forneceu medições acuradas que possibilitaram o robô ajustar sua posição e o erro sistemático de deslocamento das suas rodas.

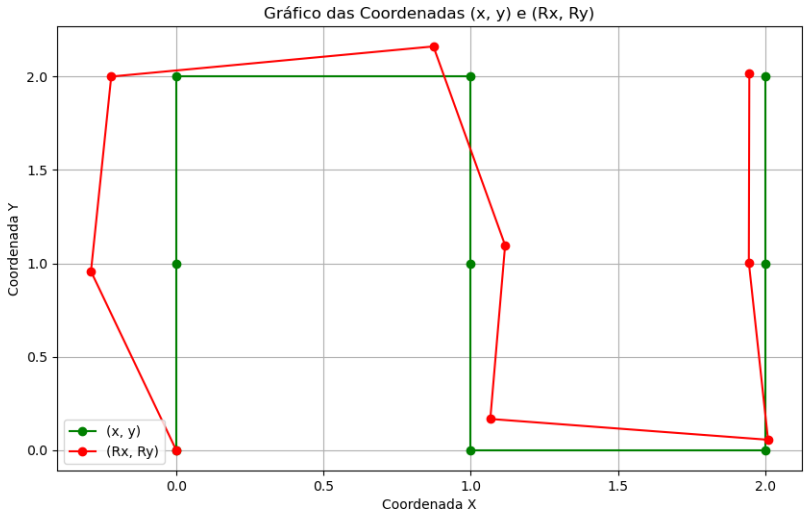


Figura 4.11 – Gráfico de coordenadas. Em verde, a trajetória de pontos (x, y) que o robô deveria seguir. Em vermelho, a trajetória de pontos (Rx, Ry) que o robô percorreu

Ao longo do percurso, a cada iteração estimam-se os valores de K_d e K_e por meio do KF, aproximando-se da entrada inserida na simulação, (1,05, 0,95). Isso pode ser observado na Figura 4.12 pela progressão das estimativas, cujas incertezas também vão decaindo dos seus estados iniciais em $1{,}00 \pm 0{,}05^2$. Nota-se que na quarta iteração, as curvas têm um comportamento mais brusco do que no restante da simulação, o que se deve à passagem

[1] A simulação pode ser vista neste link.

pela posição 5 que produz medições inconsistentes. Entretanto, uma vantagem do filtro de Kalman reside em conseguir reduzir o impacto de medidas ruins ao longo da sua execução.

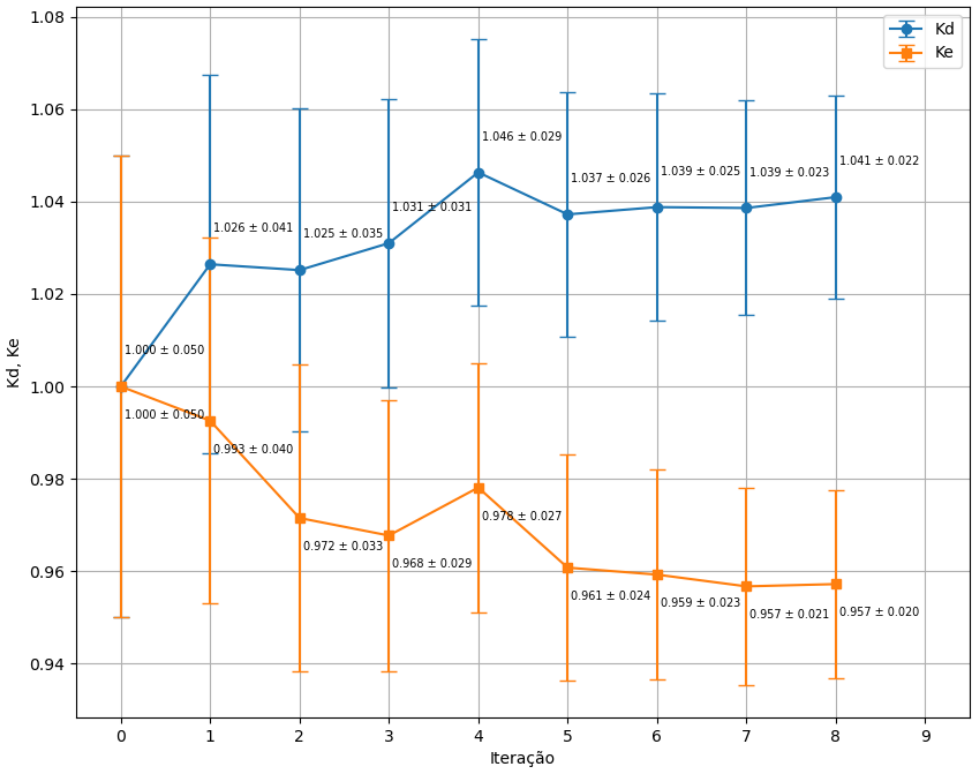


Figura 4.12 – Evolução das estimativas de K_d e K_e ao longo do percurso

É importante lembrar que a pose do robô também está sendo estimada a partir $\mathbf{K_{est}}$, sendo atualizada retroativamente a cada iteração. Na Figura 4.14, percebe-se a evolução da estimativa de posição em $(Estx, Esty)$, junto com o avanço real do robô em (Rx, Ry). Para tais gráficos, deseja-se observar os pontos amarelos se aproximando dos vermelhos, indicando a aferição correta da trajetória que foi realizada.

Tendo sido analisado o funcionamento do algoritmo para uma simulação, deve-se averiguar a sua repetibilidade. Para tal, foram realizados 30 testes com os demais casos e a Tabela 4.4 reporta a média das estimativas finais. Na Figura 4.13 apresenta-se um histograma com a variabilidade dos valores de K_d e K_e. Observa-se que em quase todos os casos, as estimativas estão próximas de seus valores reais. O fato das estimativas de K_d e K_e não chegarem aos valores exatos, pode se atribuir aos limites estipulados no algoritmo, que não permitem medições fora do intervalo $[0{,}95, 1{,}05]$, e ao baixo número de iterações. Espera-se que uma maior quantidade de iterações aprimore as estimativas.

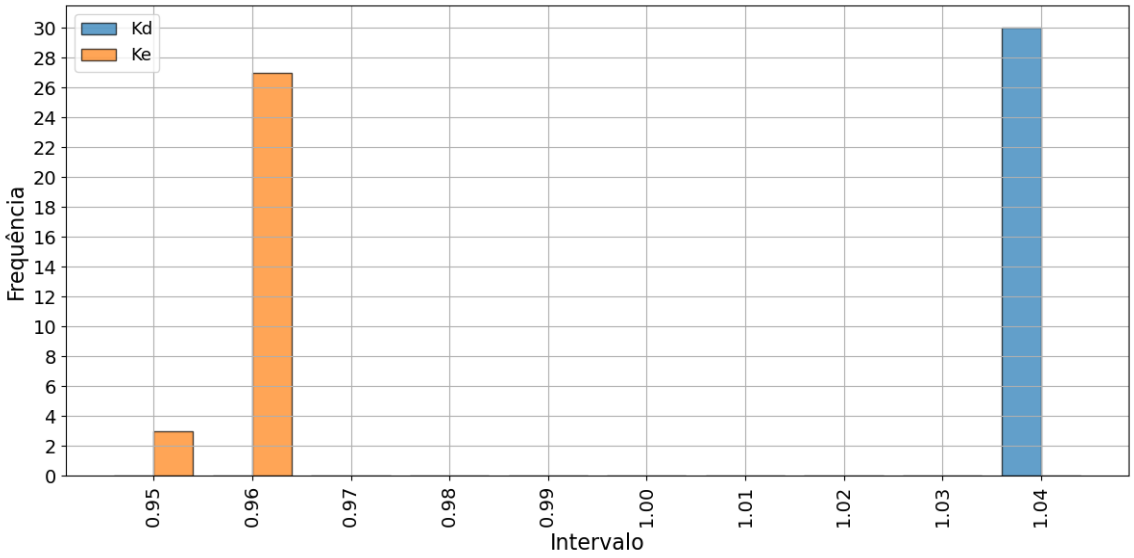


Figura 4.13 – Histograma de K_d e K_e obtidos após 30 simulações

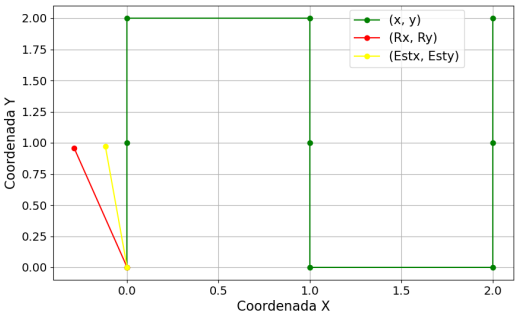
4.2.2 Avaliação do algoritmo nos demais casos

Os mesmos procedimentos da seção anterior foram atribuídos aos casos (100%, 100%), (95%, 100%), (95%, 95%) e (105%, 105%). Na Figura 4.15, observa-se a trajetória seguida pelo robô nas diferentes condições de desbalanceamento das rodas. Em todos elas, nota-se que o algoritmo é capaz de retornar para a trajetória desejada; em especial, é perceptível como na Figura 4.15a, um cenário ideal onde não há desbalanceamento por K_d e K_e, que o algoritmo não atrapalha o funcionamento do robô.

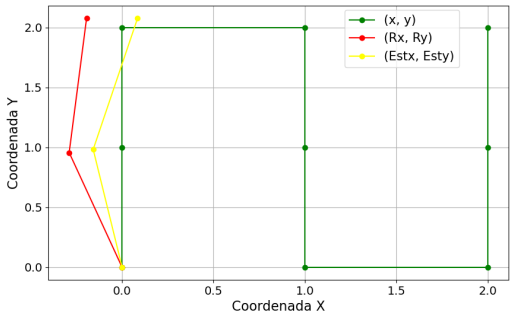
Também foi verificada a repetibilidade do algoritmo para essas condições, gerando outros histogramas. Neles, é possível ver novamente a dificuldade do algoritmo de chegar aos valores extremos (0,95 e 1,05), porém se aproximando bastante do esperado nos múltiplos testes, demonstrando sua eficiência. Na Tabela 4.4, encontram-se as médias das estimativas finais do algoritmo em cada um dos casos testados.

Tabela 4.4 – Tabela de média de K_d e K_e

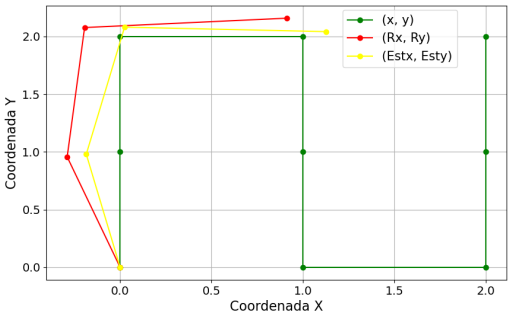
Casos	K_d	K_e
(1,00, 1,00)	0,997	0,997
(0,95, 1,00)	0,958	1,000
(0,95, 0,95)	0,959	0,958
(1,05, 1,05)	1,035	1,036
(1,05, 0,95)	1,038	0,956



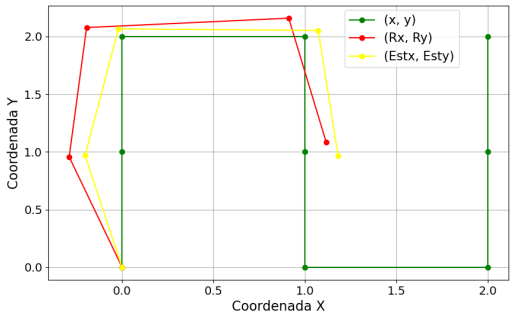
(a) Primeira iteração



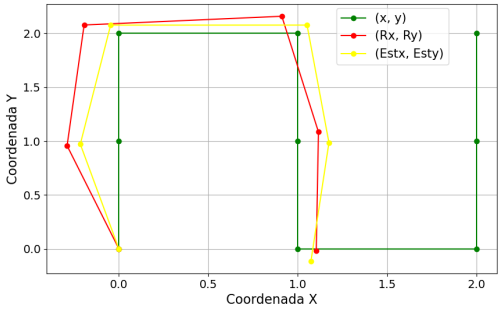
(b) Segunda iteração



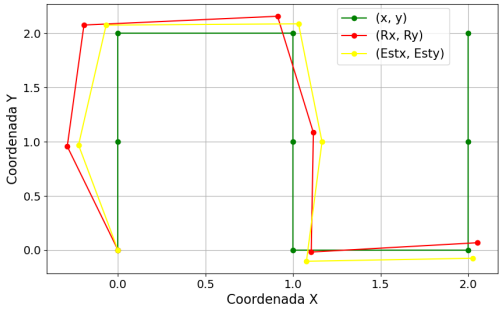
(c) Terceira iteração



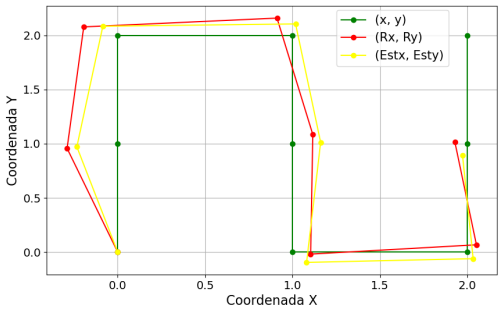
(d) Quarta iteração



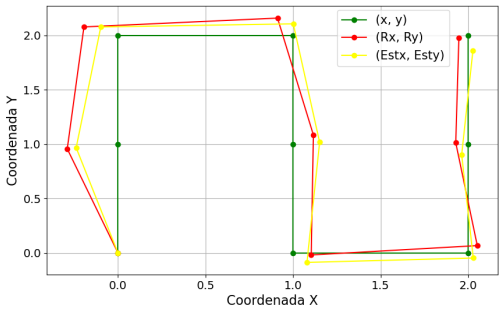
(e) Quinta iteração



(f) Sexta iteração

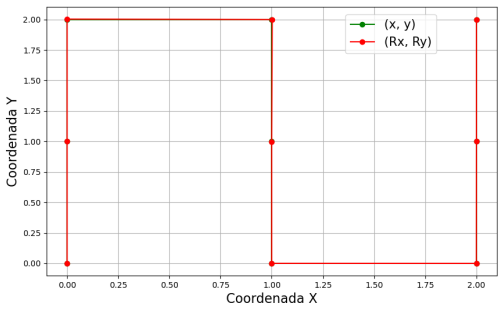


(g) Sétima iteração

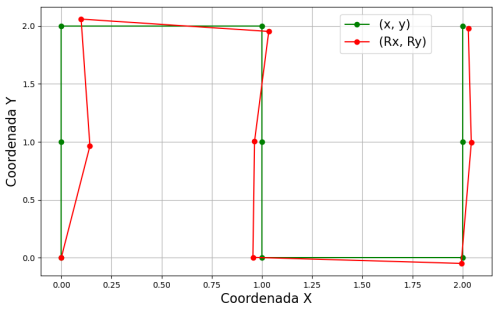


(h) Oitava iteração

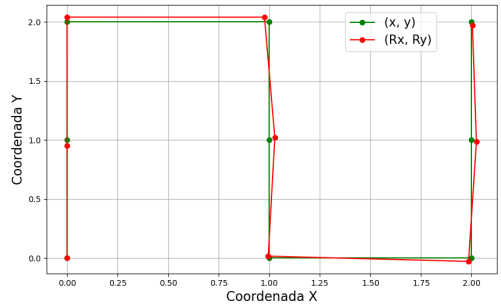
Figura 4.14 – Evolução da estimativa de posição, em amarelo, em torno da trajetória realizada, em vermelho



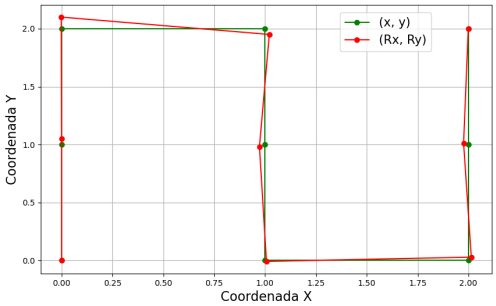
(a) $(K_d, K_e) = (100\%, 100\%)$



(b) $(K_d, K_e) = (95\%, 100\%)$

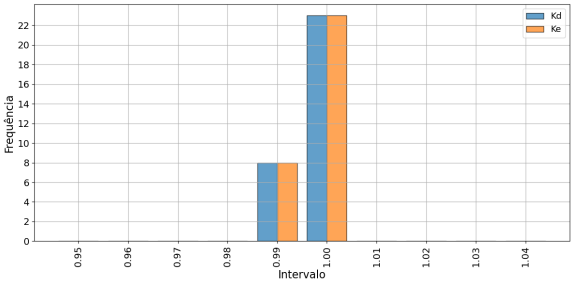


(c) $(K_d, K_e) = (95\%, 95\%)$

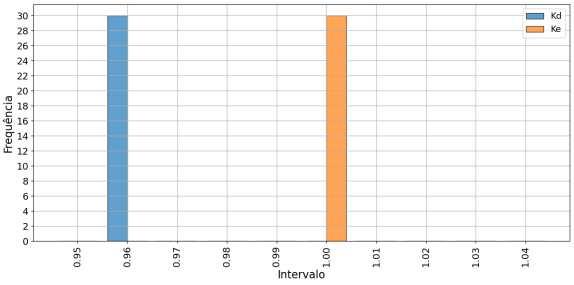


(d) $(K_d, K_e) = (105\%, 105\%)$

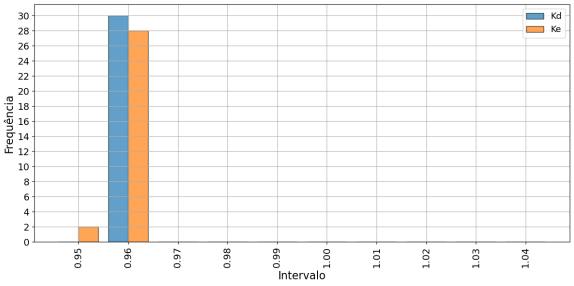
Figura 4.15 – Gráfico de coordenadas em diferentes casos



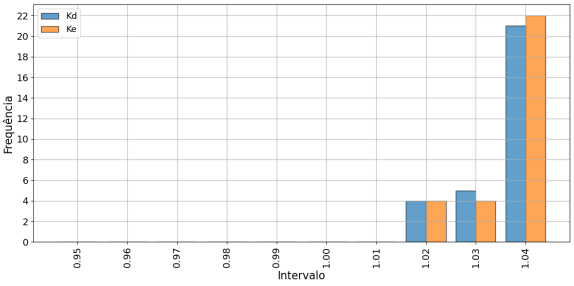
(a) $(K_d, K_e) = (100\%, 100\%)$



(b) $(K_d, K_e) = (95\%, 100\%)$



(c) $(K_d, K_e) = (95\%, 95\%)$



(d) $(K_d, K_e) = (105\%, 105\%)$

Figura 4.16 – Gráfico de coordenadas dos demais casos

5 Conclusões

Neste trabalho, foram apresentadas e discutidas abordagens teóricas no campo da robótica móvel, abrangendo desde algoritmos bioinspirados até técnicas como odometria e localização baseada no filtro de Kalman. Essas abordagens representam um conjunto diversificado de ferramentas e conceitos que podem ser aplicados para melhorar a eficiência, precisão e autonomia dos robôs móveis. Buscou-se sintetizar as principais contribuições teóricas e discutir possíveis direções futuras para o desenvolvimento e aprimoramento dessas abordagens na prática e em ambiente de simulação.

Foi mostrado o desenvolvimento e a avaliação de um método de localização para robôs móveis, utilizando o filtro de Kalman, combinado com técnicas de *scan-matching* e o algoritmo PSO. Os resultados das simulações mostraram que o algoritmo de localização conseguiu reduzir os erros de posicionamento ao longo das iterações. Em particular, a compensação de erros sistemáticos provocados pelo desbalanceamento das rodas (K_d, K_e) se mostrou eficiente, com o robô retornando à trajetória predefinida em todas as condições testadas.

Além disso, a análise das estimativas dos parâmetros K_d e K_e demonstrou que o algoritmo foi capaz de aproximar esses valores aos reais, ainda que haja limitações inerentes ao intervalo de valores permitido pela simulação e ao número reduzido de iterações. Contudo, o comportamento geral do sistema sugere que um aumento no número de iterações poderia melhorar a precisão dessas estimativas, o que abre possibilidades para aprimoramentos futuros. Outro ponto a ser aprimorado refere-se às inconsistências observadas nos resultados de medição da coordenada y na posição 2 e em posições simétricas como 5 e 8, onde os escaneamentos laterais limitam a diferenciação das medições de Δx e $\Delta\theta$, prejudicando a otimização pelo PSO. Isso pode ser feito limitando o espaço de busca das partículas do PSO, reduzindo a ambiguidade para escaneamentos similares.

O método proposto também mostrou alta repetibilidade nos testes de *scan-matching* realizados, alcançando uma acurácia superior a 97,13% em diversas simulações, demonstrando uma precisão adequada em cenários distintos, mesmo aqueles com desbalanceamento severo das rodas.

Para trabalhos futuros, recomenda-se a aplicação do algoritmo em ambientes reais e dinâmicos, onde novas incertezas podem surgir. Adicionalmente, a extensão das iterações e a implementação de técnicas mais avançadas de ajuste de parâmetros poderão aumentar ainda mais a precisão e robustez do sistema. Também deseja-se testar o algoritmo de localização com outras técnicas bioinspiradas, explorando alternativas ao PSO para avaliar seu desempenho em comparação com diferentes abordagens de otimização.

Referências

ABD EL AZIZ, M.; EWEES, A. A.; HASSANIEN, A. E. Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. **Expert Systems with Applications**, Elsevier, v. 83, p. 242–256, 2017. Citado na p. 28.

ABD ELAZIZ, M.; LU, S.; HE, S. A multi-leader whale optimization algorithm for global optimization and image segmentation. **Expert Systems with Applications**, Elsevier, v. 175, p. 114841, 2021. Citado na p. 28.

ABDEL-BASSET, M.; CHANG, V.; MOHAMED, R. HSMA_WOA: A hybrid novel Slime mould algorithm with whale optimization algorithm for tackling the image segmentation problem of chest X-ray images. **Applied soft computing**, Elsevier, v. 95, p. 106642, 2020. Citado na p. 28.

AJEIL, F. H.; IBRAHEEM, I. K.; SAHIB, M. A.; HUMAIDI, A. J. Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm. **Applied Soft Computing**, v. 89, p. 106076, 2020. ISSN 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2020.106076. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494620300168>. Citado na p. 31.

ALATISE, M. B.; HANCKE, G. P. A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods. **IEEE Access**, v. 8, p. 39830–39846, 2020. DOI: 10.1109/ACCESS.2020.2975643. Citado na p. 21.

AQEL, M. O. A.; MARHABAN, M. H.; SARIPAN, M. I.; ISMAIL, N. Review of visual odometry: types, approaches, challenges, and applications. **SpringerPlus**, v. 5, 2016. Citado nas pp. 17, 19.

AUGER, F.; HILAIRET, M.; GUERRERO, J. M.; MONMASSON, E.; ORLOWSKA-KOWALSKA, T.; KATSURA, S. Industrial applications of the Kalman filter: A review. **IEEE Transactions on Industrial Electronics**, IEEE, v. 60, n. 12, p. 5458–5471, 2013. Citado na p. 23.

BEN-ARI, M.; MONDADA, F. Robotic Motion and Odometry. In: jan. 2018. P. 63–93. ISBN 978-3-319-62532-4. DOI: 10.1007/978-3-319-62533-1_5. Citado na p. 19.

BINITHA, S.; SATHYA, S. S. et al. A survey of bio inspired optimization algorithms. **International journal of soft computing and engineering**, Citeseer, v. 2, n. 2, p. 137–151, 2012. Citado na p. 25.

BIRD, B.; GRIFFITHS, A.; MARTIN, H.; CODRES, E.; JONES, J.; STANCU, A.; LENNOX, B.; WATSON, S.; POTEAU, X. A Robot to Monitor Nuclear Facilities: Using Autonomous Radiation-Monitoring Assistance to Reduce Risk and Cost. **IEEE Robotics & Automation Magazine**, v. 26, n. 1, p. 35–43, 2019. DOI: 10.1109/MRA.2018.2879755. Citado na p. 17.

BLOCK, M. S.; ROWAN, B. G. Hypochlorous Acid: A Review. **Journal of Oral and Maxillofacial Surgery**, v. 78, n. 9, p. 1461–1466, 2020. ISSN 0278-2391. DOI: https://doi.org/10.1016/j.joms.2020.06.029. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0278239120306728>. Citado na p. 15.

BORENSTEIN, J.; FENG, L. Correction of systematic odometry errors in mobile robots. In: PROCEEDINGS 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots. 1995. v. 3, 569–574 vol.3. DOI: 10.1109/IROS.1995.525942. Citado na p. 20.

BRUZZONE, L.; QUAGLIA, G. Locomotion systems for ground mobile robots in unstructured environments. **Mechanical sciences**, Copernicus Publications Göttingen, Germany, v. 3, n. 2, p. 49–62, 2012. Citado na p. 18.

CARLISLE, A.; DOZIER, G. et al. An off-the-shelf PSO. In: INDIANAPOLIS, USA. PROCEEDINGS of the workshop on particle swarm optimization. 2001. v. 1, p. 1–6. Citado na p. 41.

CEBOLLADA, S.; PAYÁ, L.; FLORES, M.; PEIDRÓ, A.; REINOSO, O. A state-of-the-art review on mobile robotics tasks using artificial intelligence and visual data. **Expert Systems with Applications**, v. 167, p. 114195, 2021. ISSN 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.114195. Disponível em: <https://www.sciencedirect.com/science/article/pii/S095741742030926X>. Citado na p. 16.

CEN, G.; MATSUHIRA, N.; HIROKAWA, J.; OGAWA, H.; HAGIWARA, I. Mobile robot global localization using particle filters. In: 2008 International Conference on Control, Automation and Systems. 2008. P. 710–713. DOI: 10.1109/ICCAS.2008.4694593. Citado na p. 21.

CHEN, L.; HU, H.; MCDONALD-MAIER, K. EKF Based Mobile Robot Localization. In: 2012 Third International Conference on Emerging Security Technologies. 2012. P. 149–154. DOI: 10.1109/EST.2012.19. Citado na p. 31.

CHIAM, S. C.; TAN, K. C.; MAMUN, A. A. A memetic model of evolutionary PSO for computational finance applications. **Expert Systems with Applications**, Elsevier, v. 36, n. 2, p. 3695–3711, 2009. Citado na p. 26.

CHRISTO, V. E.; NEHEMIAH, H. K.; MINU, B.; KANNAN, A. Correlation-based ensemble feature selection using bioinspired algorithms and classification using backpropagation neural network. **Computational and mathematical methods in medicine**, Hindawi Limited, v. 2019, 2019. Citado na p. 25.

COOPER, S.; DURRANT-WHYTE, H. A Kalman filter model for GPS navigation of land vehicles. In: IEEE. PROCEEDINGS of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94). 1994. v. 1, p. 157–163. Citado na p. 23.

DELMERICO, J.; MINTCHEV, S.; GIUSTI, A.; GROMOV, B.; MELO, K.; HORVAT, T.; CADENA, C.; HUTTER, M.; IJSPEERT, A.; FLOREANO, D. et al. The current state and future outlook of rescue robotics. **Journal of Field Robotics**, Wiley Online Library, v. 36, n. 7, p. 1171–1191, 2019. Citado na p. 18.

DENG, W.; SHANG, S.; CAI, X.; ZHAO, H.; SONG, Y.; XU, J. An improved differential evolution algorithm and its application in optimization problem. **Soft Computing**, Springer, v. 25, p. 5277–5298, 2021. Citado na p. 29.

DÖRFLER, K.; DIELEMANS, G.; LACHMAYER, L.; RECKER, T.; RAATZ, A.; LOWKE, D.; GERKE, M. Additive Manufacturing using mobile robots: Opportunities and challenges for building construction. **Cement and Concrete Research**, v. 158, p. 106772, 2022. ISSN 0008-8846. DOI: https://doi.org/10.1016/j.cemconres.2022.106772. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0008884622000631>. Citado na p. 18.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. 1995. P. 39–43. DOI: 10.1109/MHS.1995.494215. Citado na p. 26.

FEOKTISTOV, V. **Differential evolution**. Springer, 2006. Citado na p. 29.

FERREIRA, J. P.; EPSTEIN, M.; ZANNAD, F. The Decline of the Experimental Paradigm During the COVID-19 Pandemic: A Template for the Future. **The American Journal of Medicine**, v. 134, n. 2, p. 166–175, 2021. ISSN 0002-9343. DOI: https://doi.org/10.1016/j.amjmed.2020.08.021. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0002934320307944>. Citado na p. 15.

FERREIRA, M. "Desenvolvimento de robô móvel para descontaminação de ambiente hospitalar". In: 27° Congresso de Iniciação Científica da Unb e 18° Congresso de Iniciação Científica do DF (2022). 2022. Disponível em: <https://conferencias.unb.br/index.php/iniciacaocientifica/27CICUnB18df/paper/view/39937>. Citado na p. 15.

FERREIRA, M.; VIDAL FILHO, W. Projeto de um robô móvel para descontaminação hospitalar. In: IX Congresso Nacional de Engenharia Mecânica (CONEM 2022). 2022. Citado na p. 15.

FIORINI, L.; ROVINI, E.; RUSSO, S.; TOCCAFONDI, L.; D'ONOFRIO, G.; CORNACCHIA LOIZZO, F. G.; BONACCORSI, M.; GIULIANI, F.; VIGNANI, G.; SANCARLO, D.; GRECO, A.; CAVALLO, F. On the Use of Assistive Technology during the COVID-19 Outbreak: Results and Lessons Learned from Pilot Studies. **Sensors**, v. 22, n. 17, 2022. ISSN 1424-8220. DOI: 10.3390/s22176631. Disponível em: <https://www.mdpi.com/1424-8220/22/17/6631>. Citado na p. 15.

FOX, D.; BURGARD, W.; THRUN, S. Markov Localization for Mobile Robots in Dynamic Environments. **Journal of Artificial Intelligence Research**, AI Access Foundation, v. 11, p. 391–427, nov. 1999. DOI: 10.1613/jair.616. Disponível em: <https://doi.org/10.1613%5C%2Fjair.616>. Citado na p. 22.

FOX, D.; BURGARD, W.; THRUN, S. Active Markov localization for mobile robots. **Robotics and Autonomous Systems**, v. 25, n. 3, p. 195–207, 1998. Autonomous Mobile Robots. ISSN 0921-8890. DOI: https://doi.org/10.1016/S0921-8890(98)00049-9. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0921889098000499>. Citado na p. 22.

FOX, D.; BURGARD, W.; THRUN, S.; CREMERS, A. B. Position estimation for mobile robots in dynamic environments. **AAAI/IAAI**, v. 1998, p. 983–988, 1998. Citado na p. 22.

GADEYNE, K.; BRUYNINCKX, H. Markov techniques for object localization with force-controlled robots. In: 10TH Int'l Conf. on Advanced Robotics. 2001. P. 91–96. Citado na p. 22.

GAO, H.; SHI, Y.; PUN, C.-M.; KWONG, S. An improved artificial bee colony algorithm with its application. **IEEE Transactions on Industrial Informatics**, IEEE, v. 15, n. 4, p. 1853–1865, 2018. Citado na p. 27.

GONZÁLEZ, R.; RODRIGUEZ, F.; GUZMAN, J.; BERENGUEL, M. Comparative study of localization techniques for mobile robots based on indirect kalman filter. In: CITESEER. PROCEEDINGS of IFR Int. Symposium on Robotics. 2009. P. 253–258. Citado na p. 21.

GRAF, B.; SCHRAFT, R. D.; NEUGEBAUER, J. A mobile robot platform for assistance and entertainment. In: CITESEER. INTERNATIONAL Symposium on Robotics, Montreal, Canada. 2000. P. i4. Citado na p. 18.

GREHL, S.; MISCHO, H.; JUNG, B. Research perspective – mobile robots in underground mining: Using robots to accelerate mine mapping, create virtual models, assist workers and increase safety. **AusIMM Bulletin**, v. 2017, p. 44–47, fev. 2017. Citado na p. 18.

GUEDRIA, N. B. Improved accelerated PSO algorithm for mechanical engineering optimization problems. **Applied Soft Computing**, Elsevier, v. 40, p. 455–467, 2016. Citado na p. 26.

GUTMANN, J.-S.; SCHLEGEL, C. AMOS: comparison of scan matching approaches for self-localization in indoor environments. In: PROCEEDINGS of the First Euromicro Workshop on Advanced Mobile Robots (EUROBOT '96). 1996. P. 61–67. DOI: 10.1109/EURBOT.1996.551882. Citado na p. 31.

HAMRIOUI, S.; LORENZ, P. Bio inspired routing algorithm and efficient communications within IoT. **IEEE Network**, IEEE, v. 31, n. 5, p. 74–79, 2017. Citado na p. 25.

JAIN, N.; NANGIA, U.; JAIN, J. A review of particle swarm optimization. **Journal of The Institution of Engineers (India): Series B**, Springer, v. 99, p. 407–411, 2018. Citado na p. 26.

JIMÉNEZ, A. C.; ANZOLA, J. P. A.; DÍAZ, V. G.; CRESPO, R. G.; ZHAO, L. PyDSLRep: A domain-specific language for robotic simulation in V-Rep. **PLoS ONE**, v. 15, 2020. Citado na p. 30.

KAR, A. K. Bio inspired computing–a review of algorithms and scope of applications. **Expert Systems with Applications**, Elsevier, v. 59, p. 20–32, 2016. Citado na p. 25.

KARABOGA, D. Artificial bee colony algorithm. **scholarpedia**, v. 5, n. 3, p. 6915, 2010. Citado na p. 27.

KAZEMINASAB, S.; SADEGHI, N.; JANFAZA, V.; RAZAVI, M.; ZIYADIDEGAN, S.; BANKS, M. K. Localization, Mapping, Navigation, and Inspection Methods in In-Pipe Robots: A Review. **IEEE Access**, v. 9, p. 162035–162058, 2021. DOI: 10.1109/ACCESS.2021.3130233. Citado na p. 17.

KIM, T.; PARK, T.-H. Extended Kalman filter (EKF) design for vehicle position tracking using reliability function of radar and lidar. **Sensors**, MDPI, v. 20, n. 15, p. 4126, 2020. Citado na p. 23.

KUREICHIK, V.; KURSITYS, I.; KULIEV, E.; GERASIMENKO, E. Application of bioinspired algorithms for solving transcomputational tasks. In: IOP PUBLISHING, 1. JOURNAL of Physics: Conference Series. 2020. v. 1703, p. 012021. Citado na p. 25.

LEE, J. H.; RICKER, N. L. Extended Kalman filter based nonlinear model predictive control. **Industrial & Engineering Chemistry Research**, ACS Publications, v. 33, n. 6, p. 1530–1541, 1994. Citado na p. 23.

LINDAUER, M.; KERR, W. E. Communication between the workers of stingless bees. **Bee world**, Taylor & Francis, v. 41, n. 2, p. 29–41, 1960. Citado na p. 27.

MELCHIORRE, M.; SCIMMI, L. S.; MAURO, S.; PASTORELLI, S. A Novel Constrained Trajectory Planner for Safe Human-robot Collaboration. In: PROCEEDINGS of the ICINCO. 2022. Citado na p. 30.

MIRJALILI, S.; LEWIS, A. The Whale Optimization Algorithm. **Advances in Engineering Software**, v. 95, p. 51–67, 2016. ISSN 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2016.01.008. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0965997816300163>. Citado na p. 28.

MOEZ, G.; GHARBI, I.; HAMZA, S. UVC disinfection robot. **Environmental Science and Pollution Research**, v. 98, p. 434–441, ago. 2021. DOI: 10.1007/s11356-020-11184-2. Citado na p. 15.

MU, X.; YUE, G.; ZHOU, N.; CHEN, C. Occupancy Grid-Based AUV SLAM Method with Forward-Looking Sonar. **Journal of Marine Science and Engineering**, v. 10, p. 1056, jul. 2022. DOI: 10.3390/jmse10081056. Citado na p. 18.

MURPHY, R. R.; KRAVITZ, J.; STOVER, S. L.; SHOURESHI, R. Mobile robots in mine rescue and recovery. **IEEE Robotics & Automation Magazine**, IEEE, v. 16, n. 2, p. 91–103, 2009. Citado na p. 18.

MURUGANANDHAM, A.; BANU, R. W. Adaptive fractal image compression using PSO. **Procedia Computer Science**, Elsevier, v. 2, p. 338–344, 2010. Citado na p. 26.

NEGENBORN, R. **Robot Localization and Kalman Filters. On finding your position in a noisy world**. Jan. 2003. Tese (Doutorado). Citado na p. 23.

NETO, W.; FARIA PINTO, M.; MARCATO, A.; SILVA, I.; FERNANDES, D. Mobile Robot Localization Based on the Novel Leader-Based Bat Algorithm, p. 10, fev. 2019. DOI: 10.1007/s40313-019-00453-2. Citado na p. 31.

NGATMAN, M. F.; SHARIF, J. M.; NGADI, M. A. A study on modified PSO algorithm in cloud computing. In: IEEE. 2017 6th ICT international student project conference (ICT-ISPC). 2017. P. 1–4. Citado na p. 26.

NSEEF, S. K.; ABDULLAH, S.; TURKY, A.; KENDALL, G. An adaptive multi-population artificial bee colony algorithm for dynamic optimisation problems. **Knowledge-Based Systems**, v. 104, p. 14–23, 2016. ISSN 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2016.04.005. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705116300363>. Citado na p. 27.

OMISORE, O.; AKINYEMI, T.; DUAN, W.; DU, W.; CHEN, X.; WANG, L. A Novel Sample-efficient Deep Reinforcement Learning with Episodic Policy Transfer for PID-Based Control in Robotic Catheter System. Citado na p. 30.

OSWAL, S.; SARAVANAKUMAR, D. Line following robots on factory floors: Significance and Simulation study using CoppeliaSim. In: IOP PUBLISHING, 1. IOP Conference Series: Materials Science and Engineering. 2021. v. 1012, p. 012008. Citado na p. 30.

PANIGRAHI, P.; BISOY, S. Localization Strategies for Autonomous Mobile Robots: A review. **Journal of King Saud University - Computer and Information Sciences**, v. 34, mar. 2021. DOI: 10.1016/j.jksuci.2021.02.015. Citado nas pp. 16, 20, 22, 23, 25.

PANIGRAHI, P. K.; BISOY, S. K. Localization strategies for autonomous mobile robots: A review. **Journal of King Saud University - Computer and Information Sciences**, v. 34, 8, Part B, p. 6019–6039, 2022. ISSN 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2021.02.015. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1319157821000550>. Citado nas pp. 16, 19.

PINTO, A.; MOREIRA, A.; COSTA, P. A Localization Method Based on Map-Matching and Particle Swarm Optimization. **Journal of Intelligent and Robotic Systems**, dez. 2013. DOI: 10.1007/s10846-013-0009-2. Citado na p. 31.

PRICE, K. V.; STORN, R. M.; LAMPINEN, J. A. The differential evolution algorithm. **Differential evolution: a practical approach to global optimization**, Springer, p. 37–134, 2005. Citado na p. 29.

RAHMAT-SAMII, Y. Genetic algorithm (GA) and particle swarm optimization (PSO) in engineering electromagnetics. In: IEEE. 17TH International Conference on Applied Electromagnetics and Communications, 2003. ICECom 2003. 2003. P. 1–5. Citado na p. 26.

RAJURKAR, S. D.; KAR, A. K.; GOSWAMI, S.; VERMA, N. K. Optimal path estimation and tracking for an automated vehicle using GA optimized fuzzy controller. In: 2016 11th International Conference on Industrial and Information Systems (ICIIS). 2016. P. 365–370. DOI: 10.1109/ICIINFS.2016.8262967. Citado na p. 31.

RENTSCHLER, M. E.; DUMPERT, J.; PLATT, S. R.; IAGNEMMA, K.; OLEYNIKOV, D.; FARRITOR, S. M. An in vivo mobile robot for surgical vision and task assistance, 2007. Citado na p. 18.

RIBEIRO, M. I. Kalman and extended kalman filters: Concept, derivation and properties. **Institute for Systems and Robotics**, v. 43, n. 46, p. 3736–3741, 2004. Citado na p. 23.

ROHMER, E.; SINGH, S. P. N.; FREESE, M. V-REP: A versatile and scalable robot simulation framework. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2013. P. 1321–1326. DOI: 10.1109/IROS.2013.6696520. Citado na p. 30.

RUBIO, F.; VALERO, F.; LLOPIS-ALBERT, C. A review of mobile robots: Concepts, methods, theoretical framework, and applications. **International Journal of Advanced Robotic Systems**, v. 16, p. 172988141983959, mar. 2019. DOI: 10.1177/1729881419839596. Citado na p. 18.

SASIADEK, J. Sensor fusion. **Annual Reviews in Control**, v. 26, n. 2, p. 203–228, 2002. ISSN 1367-5788. DOI: https://doi.org/10.1016/S1367-5788(02)00045-7. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1367578802000457>. Citado na p. 23.

SASIADEK, J. Z. Sensor fusion. **Annual Reviews in Control**, Elsevier, v. 26, n. 2, p. 203–228, 2002. Citado na p. 23.

SCHIELE, B.; CROWLEY, J. L. A comparison of position estimation techniques using occupancy grids. **Robotics and Autonomous Systems**, v. 12, n. 3, p. 163–171, 1994. ISSN 0921-8890. DOI: https://doi.org/10.1016/0921-8890(94)90023-X. Disponível em: <https://www.sciencedirect.com/science/article/pii/092188909490023X>. Citado na p. 31.

SE, S.; LOWE, D.; LITTLE, J. Local and global localization for mobile robots using visual landmarks. In: PROCEEDINGS 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180). 2001. v. 1, 414–420 vol.1. DOI: 10.1109/IROS.2001.973392. Citado na p. 21.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Introduction to Autonomous Mobile Robots**. 2nd: The MIT Press, 2011. ISBN 0262015358. Citado na p. 18.

SILVA ORTIGOZA, R.; MARCELINO-ARANDA, M.; SILVA ORTIGOZA, G.; HERNANDEZ GUZMAN, V. M.; MOLINA-VILCHIS, M. A.; SALDANA-GONZALEZ, G.; HERRERA-LOZADA, J. C.; OLGUIN-CARBAJAL, M. Wheeled Mobile Robots: A review. **IEEE Latin America Transactions**, v. 10, n. 6, p. 2209–2217, 2012. DOI: 10.1109/TLA.2012.6418124. Citado na p. 18.

SONG, Y.; CAI, X.; ZHOU, X.; ZHANG, B.; CHEN, H.; LI, Y.; DENG, W.; DENG, W. Dynamic hybrid mechanism-based differential evolution algorithm and its application. **Expert Systems with Applications**, Elsevier, v. 213, p. 118834, 2023. Citado na p. 29.

SULIMAN, C.; CRUCERU, C.; MOLDOVEANU, F. Mobile Robot Position Estimation Using the Kalman Filter. In. Citado na p. 31.

SZREK, J.; WODECKI, J.; BŁAŻEJ, R.; ZIMROZ, R. An Inspection Robot for Belt Conveyor Maintenance in Underground Mine—Infrared Thermography for Overheated Idlers Detection. **Applied Sciences**, v. 10, n. 14, 2020. ISSN 2076-3417. DOI: 10.3390/app10144984. Disponível em: <https://www.mdpi.com/2076-3417/10/14/4984>. Citado na p. 17.

THRUN, S.; BURGARD, W.; FOX, D. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). In: jan. 2005. ISBN 0-262-20162-3. Citado nas pp. 16, 20–22.

TRIANA, M. A. P. **Controle baseado em comportamento aplicado a robótica móvel usando hardware reconfigurável**. Dez. 2022. Diss. (Mestrado) – Universidade de Brasília, Brasília. Citado nas pp. 16, 33.

TURSYNBEK, I.; SHINTEMIROV, A. Modeling and simulation of spherical parallel manipulators in CoppeliaSim (V-REP) robot simulator software. In: IEEE. 2020 International Conference Nonlinearity, Information and Robotics (NIR). 2020. P. 1–6. Citado na p. 30.

ULUER, P.; KOSE, H.; LANDOWSKA, A.; ZORCEC, T.; ROBINS, B.; BARKANA, D. E. **Child-Robot Interaction Studies During COVID-19 Pandemic**. 2021. arXiv: 2105.00215 [cs.HC]. Citado na p. 15.

VAHDAT, A. R.; NOURASHRAFODDIN, N.; GHIDARY, S. S. Mobile robot global localization using differential evolution and particle swarm optimization. In: 2007 IEEE Congress on Evolutionary Computation. 2007. P. 1527–1534. DOI: 10.1109/CEC.2007.4424654. Citado na p. 31.

VOIT, J. **Otimização por enxame de partículas com congregação passiva seletiva**. 2010. Tese (Doutorado) – Universidade Federal do Rio de Janeiro, Brasil. Citado na p. 26.

WANG, J.; WEI, B.-y.; ZHANG, Y.; CHEN, H. Design of an autonomous mobile robot for hospital. In: IEEE. 2009 IEEE International Symposium on IT in Medicine & Education. 2009. v. 1, p. 1181–1186. Citado na p. 18.

WELCH, G.; BISHOP, G. An Introduction to the Kalman Filter. **Proc. Siggraph Course**, v. 8, jan. 2006. Citado na p. 23.

WELCH, G. F. Kalman filter. **Computer Vision: A Reference Guide**, Springer, p. 1–3, 2020. Citado na p. 23.

ZHOU, H.; SUN, G.; FU, S.; LIU, J.; ZHOU, X.; ZHOU, J. A big data mining approach of PSO-based BP neural network for financial risk management with IoT. **IEEE Access**, IEEE, v. 7, p. 154035–154043, 2019. Citado na p. 26.