Universidade Federal de Juiz de Fora

Engenharia Elétrica

Programa de Graduação em Engenharia Elétrica com habilitação em Robótica e Automação Industrial

Filipe Póvoa de Paiva

Filtro de Kalman aplicado à Localização de Robôs Móveis utilizando o sensor Laser Rangefinder 2D

Juiz de Fora

2017

Filipe Póvoa de Paiva

Filtro de Kalman aplicado à Localização de Robôs Móveis utilizando o sensor Laser Rangefinder 2D

Trabalho de Conclusão de Curso apresentado ao Programa de Graduação em Engenharia Elétrica com habilitação em Robótica e Automação Industrial da Universidade Federal de Juiz de Fora, na área de concentração em Localização de Robôs Móveis , como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica.

Orientadora: Prof[a]. Ana Sophia Cavalcanti Alves Vilas Boas, D. Eng.

Coorientador: Prof. Leonardo Rocha Olivi, D. Eng.

Juiz de Fora

2017

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

Póvoa de Paiva, Filipe.
Filtro de Kalman aplicado à Localização de Robôs Móveis utilizando o sensor Laser Rangefinder 2D / Filipe Póvoa de Paiva. – 2017.
79 f. : il.

Orientadora: Profª. Ana Sophia Cavalcanti Alves Vilas Boas, D. Eng.
Coorientador: Prof. Leonardo Rocha Olivi, D. Eng.
Trabalho de Conclusão de Curso (Graduação) – Universidade Federal de Juiz de Fora, Engenharia Elétrica. Programa de Graduação em Engenharia Elétrica com habilitação em Robótica e Automação Industrial, 2017.

1. Palavra-chave. 2. Palavra-chave. 3. Palavra-chave. I. Cavalcanti Alves Vilas Boas, Ana Sophia, orient. II. Rocha Olivi, Leonardo, coorient. III. Título

Filipe Póvoa de Paiva

Filtro de Kalman aplicado à Localização de Robôs Móveis utilizando o sensor Laser Rangefinder 2D

Trabalho de Conclusão de Curso apresentado ao Programa de Graduação em Engenharia Elétrica com habilitação em Robótica e Automação Industrial da Universidade Federal de Juiz de Fora, na área de concentração em Localização de Robôs Móveis , como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica.

Aprovada em:

BANCA EXAMINADORA

Profª. Ana Sophia Cavalcanti Alves Vilas Boas, D. Eng. - Orientadora
Universidade Federal de Juiz de Fora

Prof. Leonardo Rocha Olivi, D. Eng. - Coorientador
Universidade Federal de Juiz de Fora

Elias Ramos Vilas Boas, M. Eng.
Universidade Federal de Juiz de Fora

Prof. Exuperry Barros Costa, M. Eng.
Universidade Federal de Juiz de Fora

AGRADECIMENTOS

Meus primeiros agradecimetos vão à minha família que é o meu suporte em tudo na vida, especialmente os meus pais que sempre me apoiaram nesta longa caminhada que é graduar-se em Engenharia Elétrica. Ao meu pai por sempre acreditar em mim e ser base de minha inspiração. À minha mãe que está sempre presente. À minha irmã que, apesar da distância, sempre se interessou pelos meu trabalhos na faculdade. E a Deus, em quem sempre confiei.

Gostaria, também, de agradecer a todos os meus colegas de turma que sempre foram excelentes parceiros na realização de tarefas árduas, responsáveis pelo meu crescimento pessoal e profissional, além de suas grandes contribuições e ideias.

À minha professora e orientadora, Ana Sophia, por disponibilizar do seu tempo, mesmo durante as férias, e me ajudar na realização deste trabalho. Ao professor e coordenador do curso de Robótica e Automação Industrial, Leonardo Olivi, pelos seus conselhos para a realização deste trabalho, além de vários outros conselhos e por ser uma fonte de inspiração.

“Se vi mais longe foi por estar de pé sobre ombros de gigantes”
(Isaac Newton)

RESUMO

Este trabalho aborda a Localização de Robôs Móveis, um tema bem estudado na área da Robótica. O maior desafio encontrado até então é o de se obter uma localização que seja o suficientemente precisa e mais próxima possível da realidade. Na literatura são encontrados diversos métodos distintos para a solução do problema, e cabe ao projetista saber qual é aquele mais adequado aos seus requisitos. O presente trabalho utiliza-se de dois sensores, o *encoder* e o *Laser Rangefinder* 2D, juntamente com o Filtro de Kalman, uma ferramenta estatística que serve para filtrar e entregar informações que são mais confiáveis do que aquelas dadas pelos sensores. O desenvolvimento do trabalho se dá através de uma plataforma educacional de simulação fornecida pela Coppelia Robotics chamada V-Rep, sendo utilizado, como robô móvel, o *p3dx*. Ele é submetido a alguns testes onde deve percorrer trajetórias diferentes ao longo de mapas distintos que não foram escolhidos especificamente, comprovando, assim, que o algoritmo funciona para qualquer mapa. O objetivo, que era atingir resultados melhores do que aqueles apresentados somente pela hodometria do robô, foi alcançado de modo satisfatório. Uma futura proposta seria a implementação do algoritmo em um robô real.

Palavras-chave: Filtro de Kalman. Localização de Robôs Móveis. Hodometria. *Laser Rangefinder* 2D. Simulador V-REP.

ABSTRACT

This work approaches the Mobile Robot Localization problem, a well studied area within Robotics field. The biggest challenge so far is to obtain a localization precise enough and close to the reality. In the literature there are several different methods of resolution and it is up to the designer decide which one is the best for their purposes. The present work uses two sensors, the encoder and the Laser Rangefinder 2D, together with the Kalman Filter, a statistical tool which filters and delivers information that are more reliable than those given by the sensors individually. It is developed in a educational simulation platform called V-Rep provided by Coppelia Robotics, and the mobile robot used is the p3dx. The robot undergoes various tests where it goes through different paths along some maps. These maps are not specifically chosen, showing that the algorithm works for any map. The goal, which was to achieve better results than those given only by the odometry, is successfully reached. A future work would be implementing the algorithm in a real robot.

Key-words: Kalman Filter. Mobile Robot Localization. Odometry. Laser Rangefinder 2D. V-REP Simulator.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UFJF	Universidade Federal de Juiz de Fora
GPS	Global Positioning System
EKF	Filtro de Kalman Estendido
EMQ	Erro Médio Quadrático
SLAM	Simultaneous Localization And Mapping

LISTA DE SÍMBOLOS

v_{r}	Velocidade linear da roda direita do robô
v_{l}	Velocidade linear da roda esquerda do robô
φ_{r}	Velocidade angular da roda direita do robô
φ_{l}	Velocidade angular da roda esquerda do robô
r	Raio da roda do robô
v	Velocidade linear do robô
w	Velocidade angular do robô
l	Distância entre a roda do robô e o ponto médio entre as duas rodas
P	Posição do robô em relação ao frame inercial
Δs	Deslocamento linear do robô
$\Delta\theta$	Deslocamento angular do robô
Δt	Variação do instante de tempo entre $t_{\mathrm{k\text{-}1}}$ e t_{k}
x_{k}	Posição exata do robô no instante k
u_{k}	Velocidade do robô no instante k
v_{k}	Ruído que representa os erros nas medidas de velocidade e deslocamento no instante k
z_{k}	Medida exata do sensor no instante k
w_{k}	Ruído que representa os erros nas medidas do sensor no instante k
$\hat{x_{\mathrm{k}}}^{-}$	Posição estimada do robô no instante k a Priori
${P_{\mathrm{k}}}^{-}$	Covariância da posição do robô no instante k a Priori
ν_{k}	Inovação no instante k
K_{k}	Ganho de Kalman no instante k
${P_{\mathrm{k}}}^{+}$	Covariância da posição do robô no instante k a Posteriori
$\hat{x_{\mathrm{k}}}^{+}$	Posição estimada do robô no instante k a Posteriori
k_{r}	Constante de multiplicação do deslocamento da roda direita

k_{l} Constante de multiplicação do deslocamento da roda esquerda

dsr Deslocamento da roda direita

dsl Deslocamento da roda esquerda

x_{oi} Coordenada x do obstáculo i identificado pelo *laser*

y_{oi} Coordenada y do obstáculo i identificado pelo *laser*

x_{R} Coordenada x do robô

y_{R} Coordenada y do robô

θ_{R} Ângulo do robô no frame inercial

${\sigma_{\mathrm{i}}}^2$ Variância da medida do sensor i

$dist_{\mathrm{i}}$ Distância medida ao obstáculo medida pelo sensor i

α_{i} Ângulo ao obstáculo i no frame do robô

${EMQ_{\mathrm{x}}}^{priori}$ Erro Médio Quadrático em relação à coordenada x das posições a Priori

${EMQ_{\mathrm{y}}}^{priori}$ Erro Médio Quadrático em relação à coordenada y das posições a Priori

${EMQ_{\mathrm{theta}}}^{priori}$ Erro Médio Quadrático em relação à orientação θ do robô a Priori

${EMQ_{\mathrm{x}}}^{posteriori}$ Erro Médio Quadrático em relação à coordenada x das posições a Posteriori

${EMQ_{\mathrm{y}}}^{posteriori}$ Erro Médio Quadrático em relação à coordenada y das posições a Posteriori

${EMQ_{\mathrm{theta}}}^{posteriori}$ Erro Médio Quadrático em relação à orientação θ do robô a Posteriori

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO E MOTIVAÇÃO

O estudo da robótica móvel é de grande importância e atual, uma vez que há crescente demanda e investimentos relacionados à área em diversos setores da economia. Os estudos, pesquisas e desenvolvimento fizeram com que houvesse um grande salto nas tecnologias envolvendo a aplicação de robôs móveis. Atualmente, existem aplicações domésticas, industriais, urbanas, militares e de segurança e defesa civil e militar [14], exemplificando a grande importância dos robôs móveis nos dias de hoje.

Existem duas possibilidades para os robôs móveis: serem teleoperados, ou seja, serem controlados por um operador externo (ser humano) ou serem autônomos, isto é, navegarem pelo espaço sem nenhum auxílio externo. O interesse por robôs móveis autônomos tem ganhado uma maior importância, uma vez que eles podem ser extremamente vantajosos em várias aplicações, diminuindo vários gastos e aumentando a confiabilidade. A Figura 1 dá dois exemplos de robôs móveis autônomos muito utilizados.

A capacidade de um robô móvel se autoguiar é uma das competências mais desafiadoras exigidas [5]. Para uma navegação bem sucedida, o robô precisa utilizar-se de sensores que sejam altamente confiáveis para que lhe deem uma informação relevante acerca de sua posição. A partir daí, o robô pode decidir qual decisão deve tomar. Durante a navegação, o robô realiza três perguntas: "Onde estou?", "Onde quero chegar?"e "Como posso chegar lá?"[15]. O presente trabalho foca apenas na primeira pergunta "Onde estou?".

Figura 1 – Exemplos de robôs móveis autônomos terrestres. À esquerda: robô *p3dx*. À direita: robô *Husky*.

Fonte: https://www.researchgate.net/figure/283284344_fig1_Fig-1-Intelligent-mobile-mapping-systems-Left-autonomous-mobile-robot-PIONEER-3DX

Deste modo, fica evidente que a autonomia dos robôs móveis está fortemente dependente da sua capacidade de se localizar [16]. Existem diversos sensores diferentes no mercado que auxiliam nisso, como GPS, sensores a *laser*, sensores ultrassônicos, câmeras e etc., porém muitos deles podem não ser precisos e acurados o suficiente para determinadas

aplicações. Um exemplo disso é o carro autônomo que tem que saber a sua posição com alta exatidão para não causar acidentes. Ele utiliza-se de diversos sensores e de algumas ferramentas estatísticas para que tenha uma alta confiabilidade.

Este trabalho utiliza um robô móvel autônomo, juntamente com o sensor *laser*, em uma plataforma de simulação chamada V-Rep para poder determinar a localização deste ao longo do tempo e do espaço utilizando uma ferramenta estatística chamada Filtro de Kalman.

1.2 OBJETIVOS

O objetivo deste trabalho é realizar a localização de um robô móvel no espaço bidimensional utilizando, para isso, dois sensores diferentes, um *encoder* para cada roda e o sensor *laser scanner* 2D, e o Filtro de Kalman que permite reunir as informações dos sensores de maneira a obter uma posição estimada o mais próximo possível da posição real, minimizando, assim, erros causados por ruídos e interferências a que estão sujeitos os sensores.

O trabalho tem também, por finalidade, apresentar uma nova ferramenta de simulação (V-Rep) para o estudo de assuntos relacionados à Robótica dentro da Faculdade de Engenharia da UFJF, além de estimular e divulgar o conhecimento acerca do tema estudado.

1.3 ESTRUTURA DO TRABALHO

O trabalho é dividido em 5 capítulos e 3 apêndices.

A Introdução encontra-se no Capítulo 1 e dá uma visão geral acerca do assunto abordado. O problema e os seus desafios são tratados nele.

No Capítulo 2 é descrito um pouco da História acerca da Teoria da Estimação em estatística até elaboração do Filtro de Kalman por R. E. Kalman que é a ferramenta utilizada para obter uma localização ótima do robô no trabalho.

O Capítulo 3 apresenta todas as ferramentas e estratégias utilizadas neste trabalho para alcançar os objetivos propostos.

No Capítulo 4 estão apresentados os diversos resultados obtidos que variam de acordo com os parâmetros de entrada impostos.

No Capítulo 5 apresentam-se as conclusões finais e algumas sugestões de trabalhos futuros.

Nos Apêndices A, B e C estão contidos os códigos fontes utilizados ao longo do trabalho.

2 REVISÃO BIBLIOGRÁFICA

Para entender sobre o Filtro de Kalman é preciso conhecer um pouco o contexto histórico da Teoria da Estimação em Estatística e, de Karl Friedrich Gauss a Rudolf Emil Kalman, o interesse e as evoluções no assunto foram notáveis [2].

Os primeiros estímulos para o desenvolvimento da Teoria da Estimação vieram da astronomia, quando os estudiosos da época tentavam calcular e prever a trajetória dos planetas no Sistema Solar. Para resolver esse problema Karl Friedrich Gauss inventou o Método dos Mínimos Quadrados, um método que possibilita o conhecimento do sistema com apenas alguns dados de entrada e saída, mesmo que estes estejam sujeitos a interferências e ruídos.

De acordo com Sorenson [2], esse método proposto por Gauss toma como base algumas suposições necessárias:

1) Gauss propõe que exista um número mínimo de observações para a determinação do sistema.

2) Gauss propõe que sejam tomadas mais observações do que o mínimo necessário para diminuir os erros nas medidas.

3) Gauss diz que é necessário modelar a dinâmica do sistema com o intuito de obter uma descrição exata do sistema.

4) Gauss requer um conhecimento aproximado da órbita (do corpo celestial) disponível. No Filtro de Kalman esse conhecimento prévio é essencial e, em alguns casos, requer um procedimento de linearização do sistema.

5) Gauss afirma que as estimativas devem satisfazer as observações da maneira mais exata possível. Isso significa que a diferença entre os valores observados e os valores preditos deve ser a menor possível.

6) Gauss indica que os erros nas observações são desconhecidos, abrindo espaço para os estudos em probabilidade nos problemas de estimação.

7) Por último, Gauss refere-se a um combinação adequada das observações que darão as estimativas mais acuradas. Isto está relacionado à estrutura do procedimento de estimação (filtro linear ou não-linear) e à definição de um critério de desempenho. Estas considerações são de extrema importância nas discussões atuais dos problemas de estimação.

O único problema do método apresentado por Gauss é que este considera apenas sistemas que sejam estaciónarios, como o movimento de planetas que nunca mudam a sua órbita. Com isso, cientistas passaram a estudar cada vez mais os sistemas que fossem não-estacionários à medida em que uma solução para este problema se fizesse cada vez

mais necessária.

Posteriormente, um filtro que poderia lidar com sistemas estacionários e sistemas não-estacionários foi desenvolvido por Wiener e Kolmogorov. A solução envolvia a equação de Wiener-Hopf que não era muito fácil de ser resolvida, nem computacionalmente e, portanto, pouco prática.

Baseando-se nos fundamentos propostos por Gauss e por Wiener-Kolmogorov, Rudolf Emil Kalman foi o primeiro a desenvolver uma solução que fosse fácil e prática e que pudesse resolver os sistemas, tanto estacionários, quanto não-estacionários [1]. Kalman utilizou a abordagem dos espaços de estados que estava começando a ser bastante utilizada no meio científico. Este trabalho de Kalman teve grande impacto no controle de sistemas dinâmicos, e inspirou diversos trabalhos na área de Engenharia, permitindo que o Homem pudesse ir à Lua em 1969.

Devido ao fato de ser um estimador que funciona em tempo real, rápido, eficiente e prático, o Filtro de Kalman é aplicado em cálculos de órbitas [6], rastreamento e navegação [7], posicionamento de GPS e outros. Além disso, é utilizado nas áreas de localização de robôs móveis [8]-[11] (tema do presente trabalho), microeconomia, processamento digital de imagens, reconhecimento de padrões, segmentação de imagem e etc [12].

Na área da Robótica, além do Filtro de Kalman, que é o mais utilizado, outros filtros como o Filtro de Bayes [17], o Filtro de Markov [18] e o Filtro de Partículas [19] são abordados para a localização de robôs móveis.

O simulador de robôs V-Rep é uma plataforma de experimentação de robôs virtuais onde cada objeto/modelo pode ser controlado por um *script* embarcado, por ROS, por um cliente remoto ou por uma solução customizada. Ele pode ser controlado através de linguagens de programação como C/C++, Python, Java, Lua, Matlab ou Octave [22]. O robô *p3dx* utilizado no trabalho foi desenvolvido por Eric Rohmer.

3 MATERIAIS E MÉTODOS

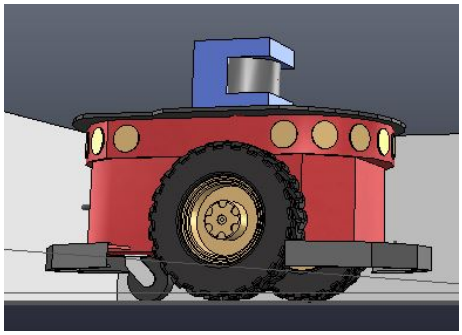
3.1 AMBIENTE DE SIMULAÇÃO V-REP

O presente trabalho foi desenvolvido na plataforma de simulação V-Rep da Coppelia Robotics [22]. Utilizou-se da versão educacional gratuita que tem algumas funcionalidades importantes.

O robô utilizado foi o *Pioneer p3dx* que é extensamente aplicado no meio acadêmico. Ele é um robô diferencial que possui *encoders* em suas duas rodas motorizadas. A outra roda é uma roda caster que proporciona estabilidade estática ao robô.

O *p3dx* possui 16 sensores ultrasônicos já embutidos, porém estes não foram utilizados. O sensor utilizado foi o *laser scanner* 2D que faz leituras de distância, assim como o ultrassônico, porém menos sujeito a interferências e erros. Este sensor foi colocado em cima do robô como mostrado na Figura 2. Os sensores ultrassônicos estão representados pelos círculos em amarelo, enquanto o sensor *laser scanner* 2D está no topo do robô em azul.

Figura 2 – Robô Móvel *p3dx* no ambiente de simulação V-Rep com o sensor *laser scanner* 2D.



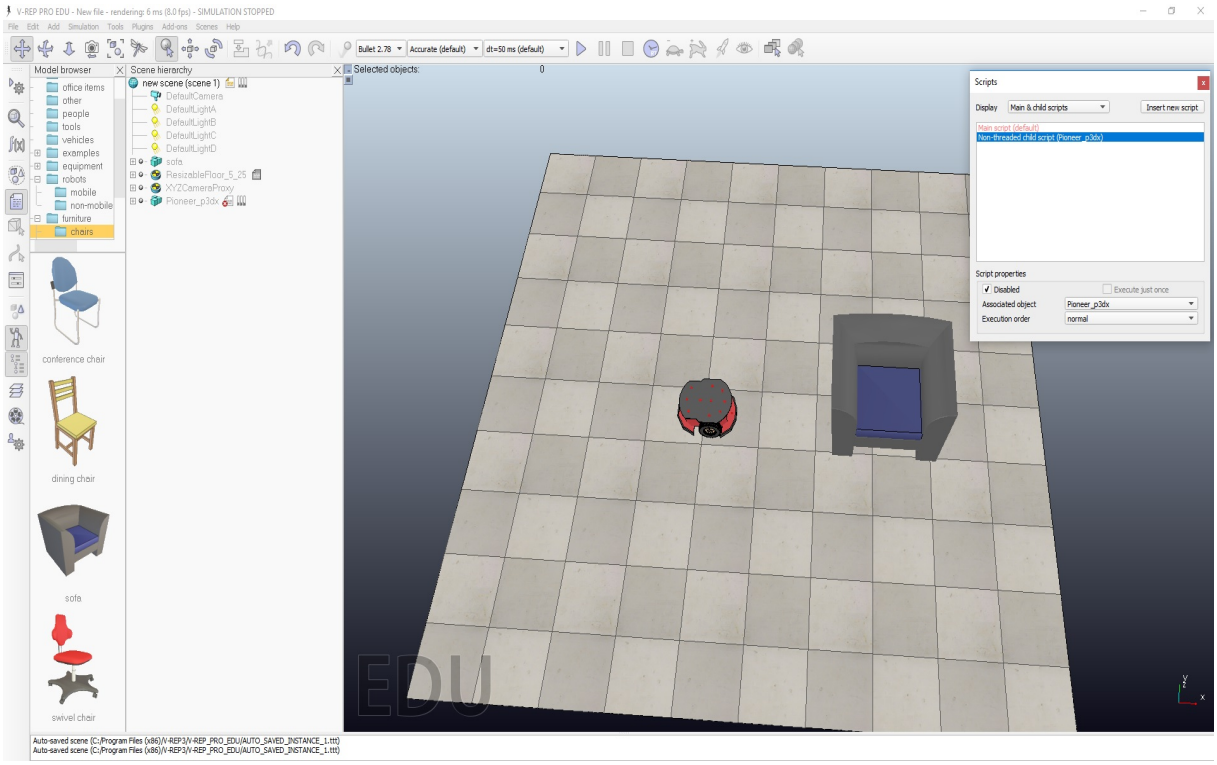
Fonte: elaborada pelo autor.

Inicialmente o robô *p3dx* já vem programado utilizando o algoritmo Braitenberg [13] para desvio de obstáculos. O *script* do código está incluído no próprio simulador. A linguagem de programação utilizada dentro do V-Rep é Lua. Entretanto, é possível utilizar vários tipos de linguagem de programação (C/C++, Python, Java, Lua, Matlab ou Octave) fora do ambiente de simulação para programar dentro do V-Rep. Neste trabalho foi utilizado o Matlab.

Para desabilitar o *script* embutido no robô é necessário seguir as seguintes etapas: *Scripts > Non-Threaded child script (Pioneer_p3dx) > checkbox Disabled.* Isso pode ser visto na Figura 3.

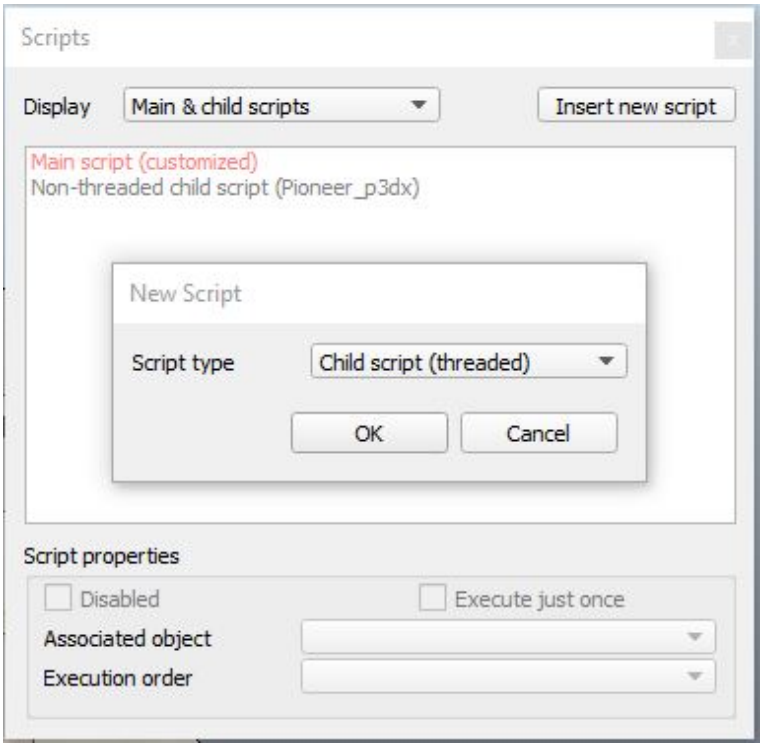
Agora, para utilizar um código exterior ao ambiente de simulação no *p3dx*, é preciso criar um *Threaded child script* (Figura 4) e associá-lo ao robô (Figura 5).

Figura 3 – Passos para desabilitar o algoritmo Braitenberg do robô.



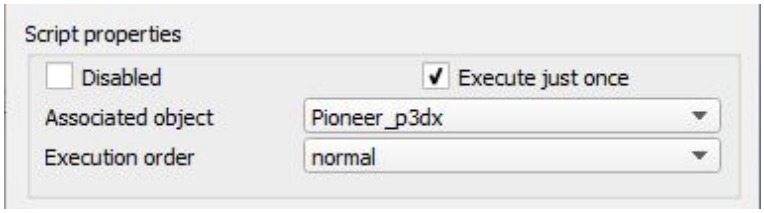
Fonte: elaborada pelo autor.

Figura 4 – Criação de um *Threaded child script*.



Fonte: elaborada pelo autor.

Figura 5 – Associação do *Threaded child script* ao *Pioneer_p3dx*



Fonte: elaborada pelo autor.

Para criar um *Threaded child script* as seguintes etapas fazem-se necessárias: *Insert new script* > *Script type: Child script (threaded)* > *Ok*.

Para desassociar o *p3dx* do *Non-threaded child script* existente e associá-lo ao *Threaded child script* criado, é preciso mudar o objeto escolhido no local onde está escrito *Associated object*, como na Figura 5.

Além disso, é necessário fazer uma comunicação em rede, utilizando uma porta e um IP (por exemplo, a porta 19999 e o IP 127.0.0.1). É necessário acrescentar *simExtRemoteApiStart(19999)* no *Threaded child script* criado, como mostrado na Figura 6.

Figura 6 – Estabelecimento de conexão pela porta 19999 no V-Rep.

```
40 -- Put some initialization code here:
41 simSetThreadSwitchTiming(2) -- Default timing for automatic thread switching
42 simExtRemoteApiStart(19999)
```

Fonte: elaborada pelo autor.

Ao adicionar o *laser scanner* 2D, associa-se este ao *p3dx* simplesmente arrastando o ícone do *laser* ao ícone do robô em *Scene hierarchy*. Um novo *Non-threaded child script* foi criado e este está associado ao *laser scanner* 2D. Para que os dados coletados pelo *laser* sejam enviados ao Matlab, é preciso alterar o código do *laser* da maneira mostrada na Figura 7.

É necessário, também, garantir que as coordenadas $[x_{\text{laser}}, y_{\text{laser}}, \theta_{\text{laser}}]^T$ do centro do *laser* coincidam com as coordenadas $[x_{\text{robô}}, y_{\text{robô}}, \theta_{\text{robô}}]^T$ do centro do robô para que não ocorram erros. Para mudar a posição de um objeto basta ir no ícone *Object/item shift* e, a partir de lá, alterar a posição e a orientação para os valores desejados.

Existe a possibilidade de alterar a amplitude de variação das medidas do *laser scanner* 2D (amplitude máxima de 0^o a 180^o) e a densidade de medidas que é dada pela quantidade por grau (mínimo de $0{,}1/^o$ e máximo de $5/^o$). A alteração desses parâmetros pode ser realizada da seguinte maneira: *Scene hierarchy* > ícone *LaserScanner_2D* >

Script Parameters > *scanningAngle* ou *scanningDensity*. Isso é ilustrado na Figura 8.

Figura 7 – Código do *Laser Scanner* 2D alterado.

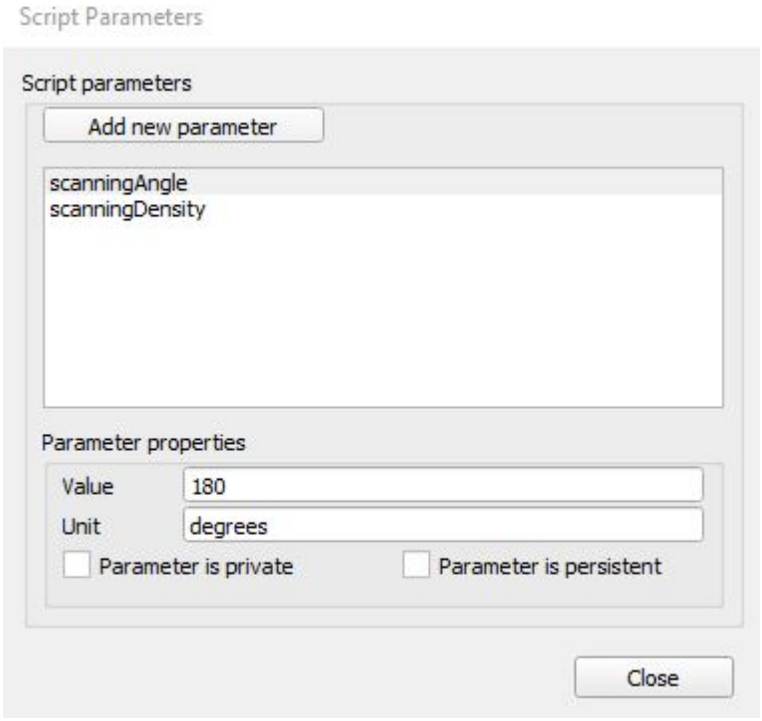
```
simResetGraph(graphHandle)
pts=scanningAngle*scanningDensity+1
p=-scanningAngle*math.pi/360
stepSize=math.pi/(scanningDensity*180)
points={}
modelInverseMatrix=simGetInvertedMatrix(simGetObjectMatrix(modelHandle,-1))
distance={}
angle={}
for i=0,pts,1 do
    simSetJointPosition(jointHandle,p)
    p=p+stepSize
    r,dist,pt=simHandleProximitySensor(laserHandle) -- pt is RELATIVE to te rotating laser beam!
    distance[i]=dist
    angle[i]=(1/scanningDensity)*(i-1)
    if r>0 then
        -- We put the RELATIVE coordinate of that point into the table that we will return:
        m=simGetObjectMatrix(laserHandle,-1)
        pt=simMultiplyVector(m,pt)
        pt=simMultiplyVector(modelInverseMatrix,pt) -- after this instruction, pt will be relative to the model base!
        table.insert(points,pt[1])
        table.insert(points,pt[2])
        table.insert(points,pt[3])
    end
    simHandleGraph(graphHandle,0.0)
end

-- Now send the data:
if #points>0 then
    simTubeWrite(communicationTube,simPackFloats(points))
end

local Distance_string=simPackFloats(distance)
local Angle_string=simPackFloats(angle)
simSetStringSignal('Distance_table',Distance_string)
simSetStringSignal('Angle_table',Angle_string)
```

Fonte: elaborada pelo autor.

Figura 8 – Aba onde podem ser alterados os parâmetros do *laser scanner* 2D.



Fonte: elaborada pelo autor.

3.2 MATLAB

A linguagem de programação escolhida para o trabalho foi a do Matlab devido à sua interface gráfica, à sua fácil manipulação e ao seu vasto material de consulta existente. É uma linguagem compatível com o V-Rep e são necessários poucos comandos para estabelecer uma conexão, como é mostrado nos comandos a seguir.

```
% Conexao inicial com o vrep
vrep=remApi('remoteApi');
vrep.simxFinish(-1); % Finaliza a comunicacao com o V-Rep caso alguma conexao tenha ficado aberta.
robot=vrep.simxStart('127.0.0.1',19999,true,true,5000,5); % Estabelece uma conexao com o V-Rep atraves do endereco de IP 127.0.0.1 e porta 19999 citados na secao 3.1.

if (robot~=-1)
    disp('Connected to remote API server') % Se a conexao foi bem sucedida
else
    disp('Connection not succesful') % Se a conexao nao foi bem sucedida
end
```

É importante destacar que os arquivos *remApi.m* e *remoteApi.dll* presentes nas pastas *C:\Program Files (x86)\V-REP3\V-REP_PRO_EDU\programming \remoteApiBindings\matlab\matlab* e *C:\Program Files (x86)\V-REP3\V-REP_PRO_EDU\programming \remoteApiBindings\lib\lib\64Bit*, respectivamente, estejam presentes na mesma pasta do código a ser utilizado.

A partir daqui, é possível manipular o robô *p3dx* da maneira que for desejada. Se, por exemplo, deseja-se fazer o robô seguir uma trajetória linear com velocidade linear de 0,2 m/s, é necessário enviar as velocidades de 2 dm/s (ou 0,2 m/s) para as rodas direita e esquerda. Isso pode ser exemplificado no código a seguir.

```
% Obtem o handle dos motores das rodas direita e esquerda
[errorCode,left_motor_handle]=vrep.simxGetObjectHandle(robot,'Pioneer_p3dx_leftMotor',vrep.simx_opmode_oneshot_wait);
[errorCode,right_motor_handle]=vrep.simxGetObjectHandle(robot,'Pioneer_p3dx_rightMotor',vrep.simx_opmode_oneshot_wait);
% Estabelece as velocidades das rodas direita e esquerda do robo e as envia para o mesmo
vl=2; % [dm/s]
```

```
6  vr=2; % [dm/s]
7  errorCode=vrep.simxSetJointTargetVelocity(robot,
       left_motor_handle,vl,vrep.simx_opmode_streaming);
8  errorCode=vrep.simxSetJointTargetVelocity(robot,
       right_motor_handle,vr,vrep.simx_opmode_streaming);
```

É importante, também, mostrar a maneira como são obtidas as informações sobre distâncias e ângulos dadas por cada medida do *laser scanner* 2D. O *laser* realiza medidas de distâncias nos ângulos de 0^o a 180^o como foi estabelecido. A quantidade de medidas varia de acordo com a densidade de pontos medidos a cada grau.

```
1  % Obtem o vetor de distancias
2  [errorCode,Distance_Value]=vrep.simxGetStringSignal(robot,'
       Distance_table',vrep.simx_opmode_oneshot_wait);
3  if(errorCode==vrep.simx_return_ok)
4      Distance=vrep.simxUnpackFloats(Distance_Value); % Distancia
           dada em metros
5  end
6  % Obtem o vetor de angulos
7  [errorCode,Angle_Value]=vrep.simxGetStringSignal(robot,'
       Angle_table',vrep.simx_opmode_oneshot_wait);
8  if(errorCode==vrep.simx_return_ok)
9      Angle=vrep.simxUnpackFloats(Angle_Value); % Angulo dado em
           graus
10 end
```

3.3 MODELO CINEMÁTICO DO ROBÔ

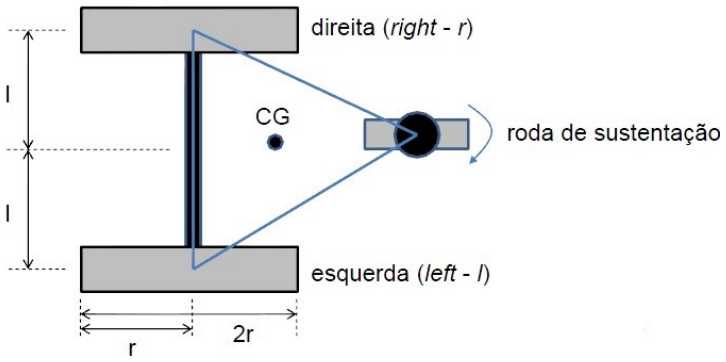
3.3.1 **Modelo do Robô**

Um robô diferencial, por possuir um motor independente para cada roda, consegue aplicar velocidades diferentes para cada uma delas.

Cada robô possui as suas características físicas já conhecidas como, por exemplo, a distância entre as rodas ($2l$) e o raio das rodas (r), já que são características de construção do mesmo. Na Figura 9 é possível observar a configuração do robô diferencial onde as duas rodas laterais são as rodas motorizadas e a outra é uma roda de sustentação para estabilidade estática.

Para o modelo cinemático do robô, além de conhecer os valores de distância entre as rodas e o raio das rodas dados anteriormente, é necessário obter os valores de velocidade de rotação das rodas direita (φ_r) e esquerda (φ_l) e o ângulo de orientação do robô (θ)

Figura 9 – Robô diferencial com suas duas rodas motorizadas e uma roda de sustentação.



Fonte: OLIVI, Leonardo - Material de Robótica Móvel, Cinemática-pt2 [20].

em relação ao frame inercial. Com estas informações em mão, é possível obter os valores de velocidade linear (v) e velocidade angular (w) do robô. No modelo cinemático, a localização do robô é representada por um ponto que fica no ponto médio entre as suas duas rodas.

As velocidades lineares das rodas direita (v_r) e esquerda (v_l) são dadas pelas seguintes equações:

$$v_\mathrm{r} = r\varphi_\mathrm{r} \tag{3.1}$$

$$v_\mathrm{l} = r\varphi_\mathrm{l} \tag{3.2}$$

onde:

v_r é a velocidade linear da roda direita do robô

v_l é a velocidade linear da roda esquerda do robô

φ_r é a velocidade angular da roda direita do robô

φ_l é a velocidade angular da roda esquerda do robô

r é o raio da roda do robô

Cada uma das rodas contribui com parte da velocidade linear do ponto médio entre elas e a soma das velocidades de cada roda é o dobro da velocidade linear deste ponto [20]. Logo, a velocidade linear do robô (v) é:

$$v = \frac{v_\mathrm{r} + v_\mathrm{l}}{2} \tag{3.3}$$

onde:

v é a velocidade linear do robô

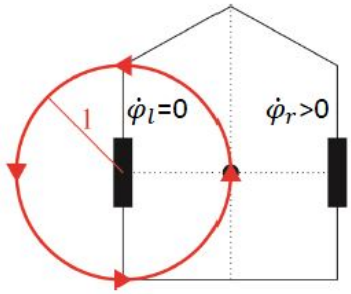
v_{r} é a velocidade linear da roda direita do robô

v_{l} é a velocidade linear da roda esquerda do robô

Para determinar a velocidade angular do robô para qualquer velocidade das rodas é necessário passar por duas etapas. Em uma primeira etapa, calcula-se a velocidade angular do robô quando a velocidade de uma roda é igual a zero e, em outra etapa, calcula-se também a velocidade angular do robô quando a velocidade da outra roda é igual a zero. A velocidade angular do robô será metade da soma destas duas velocidades encontradas [20].

Quando a velocidade da roda esquerda for igual a zero, o robô irá girar em torno desta roda no sentido anti-horário com o raio de rotação sendo a distância entre a roda e o ponto médio (l), assim como mostra a Figura 10.

Figura 10 – Robô diferencial cuja velocidade da roda esquerda é igual a zero e a velocidade da roda direita é maior do que zero.



Fonte: OLIVI, Leonardo - Material de Robótica Móvel, Cinemática-pt2 [20].

A velocidade angular, então, será dada por:

$$w = \frac{r\varphi_{\mathrm{r}}}{l} \tag{3.4}$$

onde:

w é a velocidade angular do robô

r é o raio da roda do robô

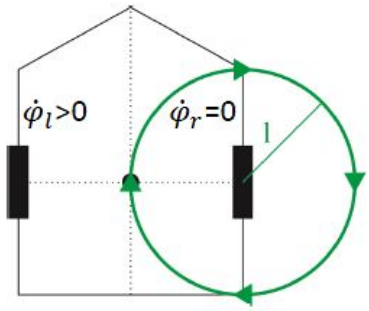
φ_{r} é a velocidade angular da roda direita do robô

l é a distância entre a roda do robô e o ponto médio entre as suas duas rodas

Quando a velocidade da roda direita for igual a zero, o robô irá girar em torno desta roda no sentido horário com o raio de rotação sendo a distância entre a roda e o

ponto médio (l), como mostrado na Figura 11.

Figura 11 – Robô diferencial cuja velocidade da roda direita é igual a zero e a velocidade da roda esquerda é maior do que zero.



Fonte: OLIVI, Leonardo - Material de Robótica Móvel, Cinemática-pt2 [20].

A velocidade angular será:

$$w = -\frac{r\varphi_{\mathrm{l}}}{l} \tag{3.5}$$

onde:

w é a velocidade angular do robô

r é o raio da roda do robô

φ_{l} é a velocidade angular da roda esquerda do robô

l é a distância entre a roda do robô e o ponto médio entre as suas duas rodas

Utilizando, assim, as equações (3.4) e (3.5), é possível encontrar a velocidade angular do robô para qualquer velocidade das rodas, resultado encontrado na equação (3.6).

$$w = \frac{r\varphi_{\mathrm{r}} - r\varphi_{\mathrm{l}}}{2l}$$

ou

$$w = \frac{v_{\mathrm{r}} - v_{\mathrm{l}}}{2l} \tag{3.6}$$

onde:

w é a velocidade angular do robô

r é o raio da roda do robô

φ_{r} é a velocidade angular da roda direita do robô

φ_{l} é a velocidade angular da roda esquerda do robô

l é a distância entre a roda do robô e o ponto médio entre as suas duas rodas

v_{r} é a velocidade linear da roda direita do robô

v_{l} é a velocidade linear da roda esquerda do robô

3.3.2 **Modelo Cinemático Incremental Direto**

Utilizando-se do Modelo do Robô visto na seção 3.3.1, é possível, agora, implementar o Modelo Cinemático Incremental Direto para definir o movimento do robô e prever, a cada instante de tempo Δt, a posição do mesmo em relação ao frame inercial.

A posição do robô em relação ao frame inercial é dado por:

$$P = \begin{bmatrix} {X_{\mathrm{R}}}^{\{I\}} \\ {Y_{\mathrm{R}}}^{\{I\}} \\ {\theta_{\mathrm{R}}}^{\{I\}} \end{bmatrix} \tag{3.7}$$

onde:

${X_{\mathrm{R}}}^{\{I\}}$ é a posição x do robô no frame inercial

${Y_{\mathrm{R}}}^{\{I\}}$ é a posição y do robô no frame inercial

${\theta_{\mathrm{R}}}^{\{I\}}$ é a posição θ do robô no frame inercial

Quando o robô se movimenta por uma trajetória em um determinado intervalo de tempo Δt, ele sai de uma determinada posição $P_{\mathrm{k\text{-}1}}$ para uma nova posição P_{k}. O robô se desloca de uma quantidade linear Δs e de uma quantidade angular $\Delta\theta$ como visto na Figura 12.

Estes valores de deslocamente linear (Δs) e angular ($\Delta\theta$) podem ser encontrados a partir das seguintes equações:

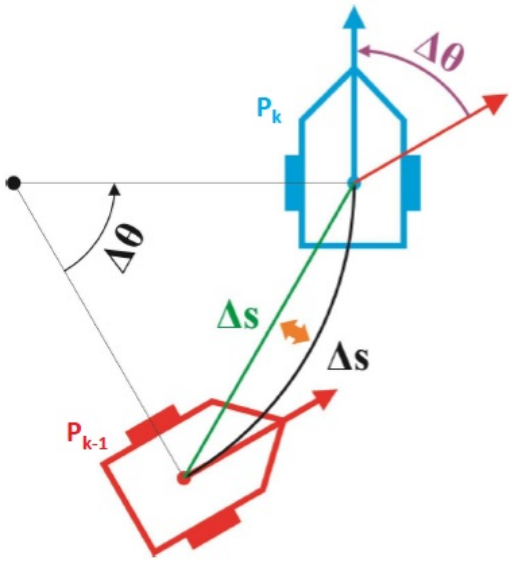
$$\Delta s = v\Delta t \tag{3.8}$$

$$\Delta\theta = w\Delta t \tag{3.9}$$

onde:

Δs é o deslocamento linear do robô

Figura 12 – Robô diferencial se deslocando por uma pequena trajetória circular entre os instantes de tempo $k-1$ e k.



Fonte: OLIVI, Leonardo - Material de Robótica Móvel, Cinemática-pt2 [20].

$\Delta\theta$ é o deslocamento do robô em θ

v é a velocidade linear do robô

w é a velocidade angular do robô

Δt é o intervalo de tempo decorrido entre os instantes $k-1$ e k

Assim, o Modelo Cinemático Incremental Direto é definido da seguinte maneira:

$$P_{\mathrm{k}} = P_{\mathrm{k\text{-}1}} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix}$$

ou

$$\begin{bmatrix} x_{\mathrm{k}} \\ y_{\mathrm{k}} \\ \theta_{\mathrm{k}} \end{bmatrix} = \begin{bmatrix} x_{\mathrm{k\text{-}1}} \\ y_{\mathrm{k\text{-}1}} \\ \theta_{\mathrm{k\text{-}1}} \end{bmatrix} + \begin{bmatrix} \Delta s.cos(\theta_{\mathrm{k\text{-}1}} + \frac{\Delta\theta}{2}) \\ \Delta s.sen(\theta_{\mathrm{k\text{-}1}} + \frac{\Delta\theta}{2}) \\ \Delta\theta \end{bmatrix} \tag{3.10}$$

onde:

P_{k} é a posição do robô no instante k

$P_{\mathrm{k\text{-}1}}$ é a posição do robô no instante $k-1$

Δx é o deslocamento do robô em x

Δy é o deslocamento do robô em y

$\Delta\theta$ é o deslocamento do robô em θ

Δs é o deslocamento linear do robô

x_{k} é a posição x do robô no instante k

y_{k} é a posição y do robô no instante k

θ_{k} é a posição θ do robô no instante k

$x_{\mathrm{k\text{-}1}}$ é a posição x do robô no instante $k-1$

$y_{\mathrm{k\text{-}1}}$ é a posição y do robô no instante $k-1$

$\theta_{\mathrm{k\text{-}1}}$ é a posição θ do robô no instante $k-1$

3.4 FILTRO DE KALMAN

O Filtro de Kalman é um estimador ótimo que coleta informações ruidosas e entrega informações mais confiáveis. É um estimador linear que trabalha com variáveis aleatórias Gaussianas.

3.4.1 **Variáveis aleatórias Gaussianas**

Para a utilização e compreensão do Filtro de Kalman, é preciso entender um pouco acerca das variáveis aleatórias Gaussianas.

3.4.1.1 *Variáveis aleatórias Gaussianas unidimensionais*

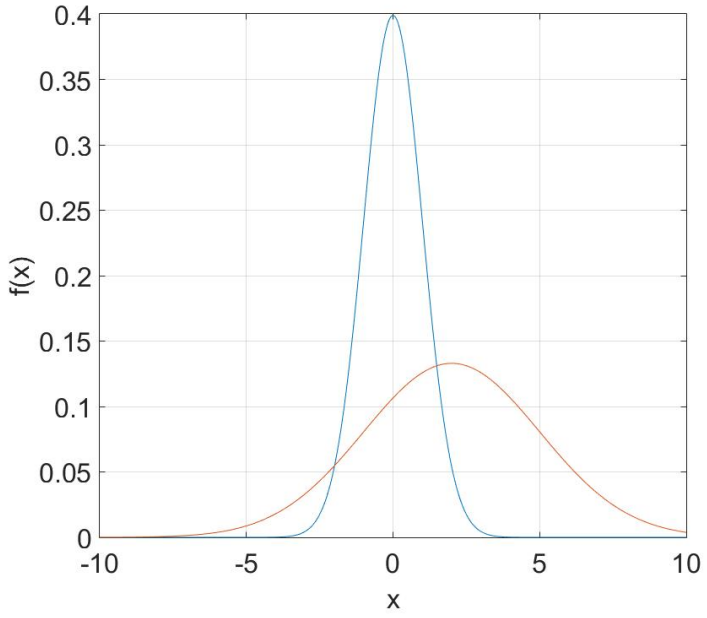
A função Gaussiana unidimensional é escrita como:

$$f(x) = \frac{1}{\sqrt{\sigma^2 2\pi}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \tag{3.11}$$

Esta é a função de distribuição de probabilidades Gaussiana para apenas uma variável aleatória x, com a média μ e desvio padrão σ. A média diz acerca da crença que tem-se sobre aquela variável, ou seja, o seu valor mais provável. O desvio padrão informa sobre o quanto de incerteza tem-se sobre aquela variável, ou seja, o quão distante ela pode estar do seu valor médio. Uma distribuição Gaussiana ou distribuição Normal, como é mais conhecida, pode ser escrita em torno destes dois parâmetros como $X \sim N(\mu, \sigma^2)$. O termo σ^2 é usualmente denominado de variância. Na Figura 13, observam-se duas distribuições normais $X_1 \sim N(0, 1^2)$ e $X_2 \sim N(2, 3^2)$.

A partir da Figura 13, é possível dizer que, para a variável aleatória em azul, é mais provável que ela esteja em 0, equanto que, para a variável aleatória em vermelho, é

Figura 13 – Duas distribuições Normais. Em azul $\mu = 0$ e $\sigma = 1$. Em vermelho $\mu = 2$ e $\sigma = 3$.



Fonte: elaborada pelo autor.

mais provável que ela esteja em 2. Porém, a certeza que a variável em azul esteja em 0 é maior do que a certeza que a variável em vermelho esteja em 2.

3.4.1.2 *Variáveis aleatórias Gaussianas multidimensionais*

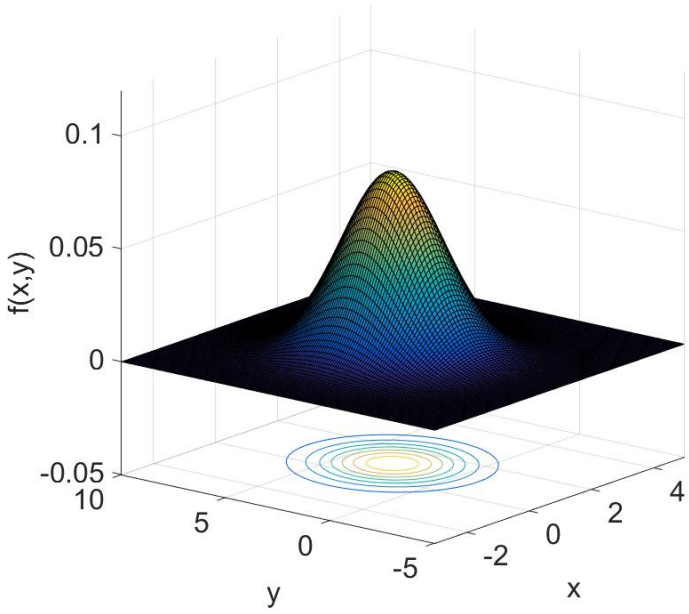
Para robôs móveis que se deslocam em espaços bidimensionais ou tridimensionais, existe uma necessidade de se estudar as variáveis aleatórias multidimensionais. O presente trabalho, se tomado em consideração a orientação θ do robô, trabalha com variáveis aleatórias tridimensionais. Uma função Gaussiana multidimensional (n dimensões) é escrita como:

$$f(x) = \frac{1}{\sqrt{\det(C)(2\pi)^{n/2}}} e^{-\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)} \tag{3.12}$$

As variáveis x e μ tornaram-se vetores de dimensão n e a matriz C é denominada de matriz de covariância. Os elementos da diagonal principal de C são as variâncias dos elementos x_i, e os elementos fora da diagonal principal são as covariâncias entre x_i e x_j. Se dois elementos x_i e x_j não têm relação entre si, a matriz C será diagonal.

Ao considerar robôs móveis que estejam situados em ambientes bidimensionais, e não for importante considerar a sua orientação, apenas a sua localização no espaço 2-D, a função de distribuição Gaussiana torna-se como na Figura 14, onde $\mu = [1\ 3]^T$ e $C = diag(1^2\ 2^2)$.

Figura 14 – Posição mais provável do robô móvel em $p = [1\ 3]^T$ e incerteza maior em y do que em x.



Fonte: elaborada pelo autor.

3.4.2 **Conceitos sobre o Filtro de Kalman**

3.4.2.1 *Modelos*

Com o objetivo de obter a posição do robô à medida em que este se desloca no espaço, é necessário que um modelo do sistema seja elaborado. Este modelo leva em consideração que a posição atual do robô depende da posição anterior mais o quanto e como ele se deslocou. Além disto, pode haver um ruído no processo, que é considerado como sendo um ruído Gaussiano de média zero e covariância V ($v_{\mathrm{k}} \sim N(0, V)$). A equação 3.13 mostra o modelo do sistema.

$$x_{\mathrm{k}} = F x_{\mathrm{k\text{-}1}} + G u_{\mathrm{k\text{-}1}} + v_{\mathrm{k\text{-}1}} \tag{3.13}$$

O vetor x representa a posição do robô, o vetor u representa a entrada do sistema que pode ser velocidade, a matriz F representa a dinâmica do sistema e a matriz G descreve como que a entrada influencia no vetor posição. A variável aleatória Gaussiana v representa o ruído.

Utilizando um sensor para identificar a posição do robô, é necessário, também, elaborar um modelo para as medidas. Para cada posição do robô no espaço, são adquiridas determinadas medidas. Estas medidas estão sujeitas a um ruído Gaussiano de média zero

e covariância W (w_{k}~$N(0, W)$). A equação 3.14 exemplifica este modelo.

$$z_{\mathrm{k}} = Hx_{\mathrm{k}} + w_{\mathrm{k}} \tag{3.14}$$

O vetor z representa a saída do sistema que são as medidas dadas pelo sensor, o vetor x é a posição do robô e a matriz H é aquela que mapeia a posição do robô para as medidas observadas. A variável aleatória Gaussiana w representa o ruído do sensor.

3.4.2.2 *Predição*

Na etapa de predição do Filtro de Kalman, o modelo do sistema é necessário para poder prever em qual estado o sistema estará após uma determinada ação ser tomada durante um determinado intervalo de tempo. Como o ruído do sistema tem média zero, o mesmo é desconsiderado quando se analisa a média da posição do robô no instante k, e isto pode ser visto na equação 3.15. A equação 3.16 determina a covariância da posição do robô. Logo, a posição do robô pode ser descrita como x_{k}~$N(\hat{x}_{\mathrm{k}}, P_{\mathrm{k}})$.

$$\hat{x}_{\mathrm{k}}{}^{-} = F\hat{x}_{\mathrm{k\text{-}1}}{}^{+} + Gu_{\mathrm{k\text{-}1}} \tag{3.15}$$

$$P_{\mathrm{k}}{}^{-} = FP_{\mathrm{k\text{-}1}}{}^{+}F^{T} + \hat{V} \tag{3.16}$$

O índice $^{-}$ representa o estado a Priori, ou seja, representa o estado antes da etapa de correção. O índice $^{+}$ representa o estado a Posteriori, ou seja, o estado após a etapa de correção.

3.4.2.3 *Correção*

Na etapa de correção do Filtro de Kalman, as medidas feitas pelo sensor em questão são utilizadas para se extrair a posição a Posteriori do robô. Se não existisse esta etapa, apenas a etapa anterior, a incerteza da posição do robô aumentaria indefinidamente, uma vez que a covariância a Priori é a soma de duas matrizes. A nova informação adquirida através do sensor resulta no que se chama inovação dada em 3.17.

$$\nu_{\mathrm{k}} = z_{\mathrm{k}} - H\hat{x}_{\mathrm{k}}{}^{-} \tag{3.17}$$

A inovação pode ser traduzida como a diferença entre o que o sensor mediu z_{k} e o que o sensor deveria ter medido pela etapa de predição $H\hat{x}_{\mathrm{k}}{}^{-}$.

Após o cálculo da inovação, é necessário calcular o ganho de Kalman para que se faça uma correção do estado a Priori.

$$K_{\mathrm{k}} = P_{\mathrm{k}}{}^{-} H^T (H P_{\mathrm{k}}{}^{-} H^T + \hat{W})^{-1} \tag{3.18}$$

$$P_{\mathrm{k}}{}^{+} = P_{\mathrm{k}}{}^{-} - K_{\mathrm{k}} H P_{\mathrm{k}}{}^{-} \tag{3.19}$$

$$\hat{x}_{\mathrm{k}}{}^{+} = \hat{x}_{\mathrm{k}}{}^{-} + K_{\mathrm{k}} \nu_{\mathrm{k}} \tag{3.20}$$

A matriz K é o ganho de Kalman, a matriz $\hat{W}$ é a covariância das medidas do sensor, a matriz P^+ é a covariância do estado a Posteriori e o vetor $\hat{x}^+$ é a estimativa do estado a Posteriori. É importante observar que, após a estapa de correção, a incerteza acerca da posição do robô diminui, como pode ser visto na equação 3.19, uma vez que há uma substração envolvida.

O ganho de Kalman K funciona como um peso em uma balança, isto é, ele indica, no instante k, se as medidas do sensor foram boas o suficiente para fazer uma correção da posição do robô ou não. Quanto mais próximo de zero for o ganho de Kalman, menos confiáveis são as medidas do sensor e, logo, a estimativa a Posteriori será aproximadamente igual a estimativa a Priori. Quanto mais próximo da unidade for o ganho de Kalman, mais confiáveis são as medidas do sensor e, logo, a estimativa da posição do robô sofrerá uma correção significativa.

3.4.3 **Filtro de Kalman Estendido**

O Filtro de Kalman, como dito anteriormente, é um estimador linear e, portanto, as equações 3.13 e 3.14 são lineares. Para os casos em que as equações não são lineares, o Filtro de Kalman Estendido (EKF) é utilizado. Seja o sistema não-linear:

$$x_{\mathrm{k}} = f(x_{\mathrm{k\text{-}1}}, u_{\mathrm{k\text{-}1}}) + v_{\mathrm{k\text{-}1}} \tag{3.21}$$

$$z_{\mathrm{k}} = h(x_{\mathrm{k}}) + w_{\mathrm{k}} \tag{3.22}$$

As funções f e h são não-lineares. Aproximações lineares locais são tomadas e, depois de uma certa álgebra dada por Corke, P. [3], o resultado a seguir é obtido.

A etapa de predição:

$$\hat{x}_{\mathrm{k}}{}^{-} = f(\hat{x}_{\mathrm{k\text{-}1}}{}^{+}, u_{\mathrm{k\text{-}1}}) \tag{3.23}$$

$$P_{\mathrm{k}}{}^{-} = F_{\mathrm{p}} P_{\mathrm{k\text{-}1}}{}^{+} F_{\mathrm{p}}{}^{T} + F_v \hat{V} F_v{}^{T} \tag{3.24}$$

a inovação:

$$\nu_{\mathrm{k}} = z_{\mathrm{k}} - h(\hat{x}_{\mathrm{k}}{}^{-}) \tag{3.25}$$

e a etapa de correção:

$$K_{\mathrm{k}} = P_{\mathrm{k}}{}^{-} H^T (H P_{\mathrm{k}}{}^{-} H^T + \hat{W})^{-1} \tag{3.26}$$

$$\hat{x}_{\mathrm{k}}{}^{+} = \hat{x}_{\mathrm{k}}{}^{-} + K_{\mathrm{k}} \nu_{\mathrm{k}} \tag{3.27}$$

$$P_{\mathrm{k}}{}^{+} = P_{\mathrm{k}}{}^{-} - K_{\mathrm{k}} H P_{\mathrm{k}}{}^{-} \tag{3.28}$$

As matrizes F_{p}, F_v e H são os Jacobianos das funções f e h avaliadas a cada instante k. As matrizes $\hat{V}$ e $\hat{W}$ são estimativas das covariâncias que mais se aproximam das covariâncias V e W.

3.4.3.1 *Filtro de Kalman Estendido para um Robô Diferencial utilizando Laser Scanner 2D*

Neste trabalho, o Filtro de Kalman Estendido proposto foi utilizado, porque as funções de dinâmica do robô f (equação 3.10) e de mapeamento da posição do robô para as medidas h não são lineares. Para o robô diferencial, os Jacobianos F_{p} e F_v são dados pelas equações 3.29 e 3.30, respectivamente.

$$f = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} x_{\mathrm{k\text{-}1}} + \Delta s.cos(\theta_{\mathrm{k\text{-}1}} + \frac{\Delta\theta}{2}) \\ y_{\mathrm{k\text{-}1}} + \Delta s.sen(\theta_{\mathrm{k\text{-}1}} + \frac{\Delta\theta}{2}) \\ \theta_{\mathrm{k\text{-}1}} + \Delta\theta \end{bmatrix}$$

$$F_{\mathrm{p}} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial \theta} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial \theta} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial \theta} \end{bmatrix}$$

$$F_{\mathrm{p}} = \begin{bmatrix} 1 & 0 & -\Delta s.sen(\theta_{\mathrm{k\text{-}1}} + \frac{\Delta\theta}{2}) \\ 0 & 1 & \Delta s.cos(\theta_{\mathrm{k\text{-}1}} + \frac{\Delta\theta}{2}) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.29}$$

$$F_v = \begin{bmatrix} \frac{\partial f_1}{\partial dsr} & \frac{\partial f_1}{\partial dsl} \\ \frac{\partial f_2}{\partial dsr} & \frac{\partial f_2}{\partial dsl} \\ \frac{\partial f_3}{\partial dsr} & \frac{\partial f_3}{\partial dsl} \end{bmatrix}$$

$$F_v = \begin{bmatrix} \frac{1}{2}cos(\theta_{\text{k-1}} + \frac{\Delta\theta}{2}) - \frac{\Delta s}{4l}sen(\theta_{\text{k-1}} + \frac{\Delta\theta}{2}) & \frac{1}{2}cos(\theta_{\text{k-1}} + \frac{\Delta\theta}{2}) + \frac{\Delta s}{4l}sen(\theta_{\text{k-1}} + \frac{\Delta\theta}{2}) \\ \frac{1}{2}sen(\theta_{\text{k-1}} + \frac{\Delta\theta}{2}) + \frac{\Delta s}{4l}cos(\theta_{\text{k-1}} + \frac{\Delta\theta}{2}) & \frac{1}{2}sen(\theta_{\text{k-1}} + \frac{\Delta\theta}{2}) - \frac{\Delta s}{4l}cos(\theta_{\text{k-1}} + \frac{\Delta\theta}{2}) \\ \frac{1}{2l} & -\frac{1}{2l} \end{bmatrix} \quad (3.30)$$

As variáveis *dsr* e *dsl* são os deslocamentos das rodas direita e esquerda, respectivamente. A matriz $\hat{V}$ é dada na equação 3.31.

$$\hat{V} = \begin{bmatrix} k_{\text{r}}|dsr| & 0 \\ 0 & k_{\text{l}}|dsl| \end{bmatrix} \quad (3.31)$$

As variáveis k_{r} e k_{l} são as constantes de multiplicação dos deslocamentos das rodas direita e esquerda, respectivamente. A matriz $\hat{V}$, então, dá informações acerca da incerteza em relação ao deslocamento das rodas.

Suponha, agora, um *laser* que realiza N medidas. Essas medidas são as distâncias euclidianas entre o robô e os seus obstáculos. Logo, a função h pode ser descrita como na equação 3.32.

$$h = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_{\text{N}} \end{bmatrix} = \begin{bmatrix} \sqrt{(x_{\text{o1}} - x_{\text{R}})^2 + (y_{\text{o1}} - y_{\text{R}})^2} \\ \sqrt{(x_{\text{o2}} - x_{\text{R}})^2 + (y_{\text{o2}} - y_{\text{R}})^2} \\ \vdots \\ \sqrt{(x_{\text{oN}} - x_{\text{R}})^2 + (y_{\text{oN}} - y_{\text{R}})^2} \end{bmatrix} \quad (3.32)$$

A matriz H será, então:

$$H = \begin{bmatrix} \frac{\partial h_1}{\partial x_{\text{R}}} & \frac{\partial h_1}{\partial y_{\text{R}}} & \frac{\partial h_1}{\partial \theta_{\text{R}}} \\ \frac{\partial h_2}{\partial x_{\text{R}}} & \frac{\partial h_2}{\partial y_{\text{R}}} & \frac{\partial h_2}{\partial \theta_{\text{R}}} \\ \vdots & \vdots & \vdots \\ \frac{\partial h_{\text{N}}}{\partial x_{\text{R}}} & \frac{\partial h_{\text{N}}}{\partial y_{\text{R}}} & \frac{\partial h_{\text{N}}}{\partial \theta_{\text{R}}} \end{bmatrix}$$

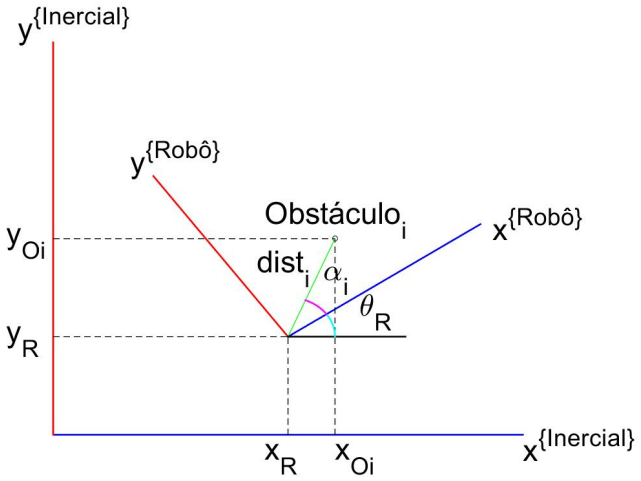
$$H = \begin{bmatrix} -\frac{x_{o1}-x_R}{\sqrt{(x_{o1}-x_R)^2+(y_{o1}-y_R)^2}} & -\frac{y_{o1}-y_R}{\sqrt{(x_{o1}-x_R)^2+(y_{o1}-y_R)^2}} & 0 \\ -\frac{x_{o2}-x_R}{\sqrt{(x_{o2}-x_R)^2+(y_{o2}-y_R)^2}} & -\frac{y_{o2}-y_R}{\sqrt{(x_{o2}-x_R)^2+(y_{o2}-y_R)^2}} & 0 \\ \vdots & \vdots & \vdots \\ -\frac{x_{oN}-x_R}{\sqrt{(x_{oN}-x_R)^2+(y_{oN}-y_R)^2}} & -\frac{y_{oN}-y_R}{\sqrt{(x_{oN}-x_R)^2+(y_{oN}-y_R)^2}} & 0 \end{bmatrix} \tag{3.33}$$

A matriz de covariância $\hat{W}$ do sensor é diagonal com os seus elementos sendo as variâncias das medidas individuais, que são valores estimados. Logo, a matriz tem dimensão NxN.

$$\hat{W} = \begin{bmatrix} {\sigma_1}^2 & 0 & \cdots & 0 \\ 0 & {\sigma_2}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & {\sigma_N}^2 \end{bmatrix} \tag{3.34}$$

As variáveis x_{oi} e y_{oi} na equação 3.33 são as coordenadas dos obstáculos do robô. Elas são obtidas a partir do esquema da Figura 15.

Figura 15 – Esquema para obtenção das coordenadas x_{oi} e y_{oi} do obtáculo$_i$.



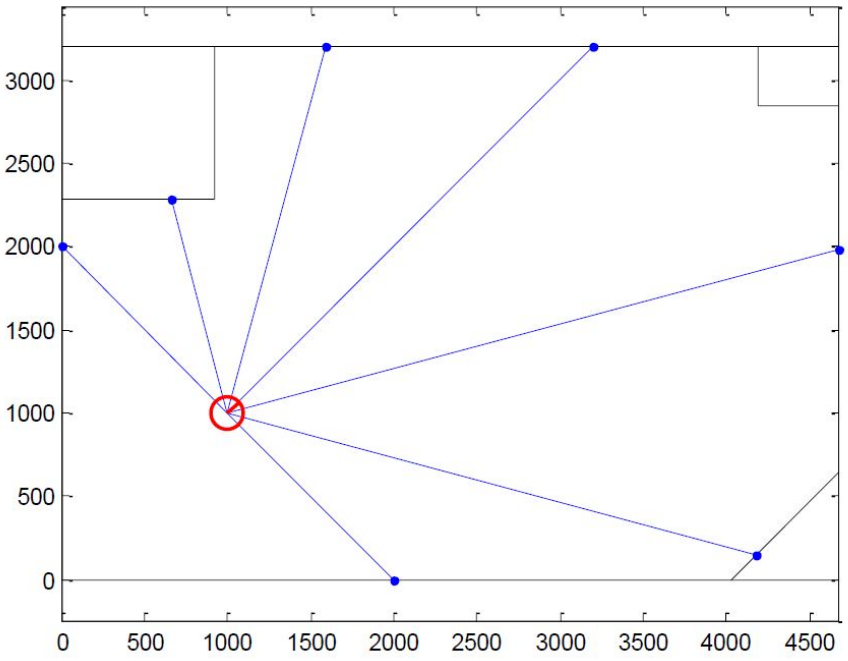
Fonte: elaborada pelo autor.

$$\begin{aligned} x_{oi} &= x_R + dist_i.cos(\theta_R + \alpha_i) \\ y_{oi} &= y_R + dist_i.sen(\theta_R + \alpha_i) \end{aligned} \tag{3.35}$$

3.5 LASER SCANNER 2D VIRTUAL

Com o intuito de se prever quais serão as medidas do sensor dada a posição a Priori do robô $\hat{x_k}^-$, foi implementado o *laser scanner* 2D virtual. Esse sensor é um modelo matemático que utiliza-se da geometria do ambiente para conseguir descobrir o termo $h(\hat{x_k}^-)$ dado na equação 3.25. O modelo mostrado na Figura 16 é um exemplo do que foi utilizado. O código que implementa o modelo encontra-se no Apêndice B.

Figura 16 – Modelo do sensor *laser scanner* 2D virtual.



Fonte: OLIVI, Leonardo - Material de Robótica Móvel - Sensores-pt2 [20].

4 RESULTADOS

Com o intuito de comprovar e validar as proposições dadas no Capítulo 3, fez-se com que o robô percorresse algumas trajetórias diferentes em mapas distintos.

No simulador, como as medidas dadas pelo *laser* são precisas e exatas, foi necessário acrescentar um ruído Gaussiano aos dados no Matlab, atendendo aos pré-requisitos necessários para a utilização do filtro, e criando um ambiente mais próximo da realidade.

No primeiro mapa, o robô percorreu uma mesma trajetória circular em três experimentos. Em cada um deles foi utilizado um ruído Gaussiano com variâncias diferentes para o sensor a *laser*: $\sigma^2 = 0,05$ no primeiro, $\sigma^2 = 0,1$ no segundo e $\sigma^2 = 0,01$ no terceiro. As constantes de multiplicação foram mantidas como $k_\mathrm{r} = 0,01$ e $k_\mathrm{l} = 0,01$ e 16 feixes de *laser* foram utilizados para o sensor *laser scanner* 2D nos três experimentos.

No segundo mapa, o robô também percorreu uma trajetória circular em três experimentos. Neste caso, o que mudou de um para o outro foram as constantes de multiplicação dos deslocamentos das rodas: $k_\mathrm{r} = 0,01$ e $k_\mathrm{l} = 0,01$ no primeiro, $k_\mathrm{r} = 0,05$ e $k_\mathrm{l} = 0,05$ no segundo e $k_\mathrm{r} = 0,1$ e $k_\mathrm{l} = 0,1$ no terceiro. As variâncias para os ruídos foram de $\sigma^2 = 0,01$ e 21 feixes de *laser* foram utilizados para o sensor *laser scanner* 2D nos três experimentos.

No terceiro mapa, houve apenas um experimento onde foi feito um controle proporcional de posição do robô em malha fechada, cujo objetivo era fazer com que ele alcançasse um ponto específico dado. A posição realimentada na malha de controle foi aquela dada após a etapa de correção do Filtro de Kalman. A variância do ruído foi de $\sigma^2 = 0,0025$ e as constantes de multiplicação dos deslocamentos das rodas foram de $k_\mathrm{r} = 0,01$ e $k_\mathrm{l} = 0,01$. O robô chegou, dentro dos limites estabelecidos, com sucesso ao ponto.

Com os resultados a expectativa é que os Erros Médios Quadráticos em x, y e θ reduzam da etapa de predição (a priori) para a etapa de correção (a posteriori).

4.1 MAPA 1

4.1.1 **Experimento 1**

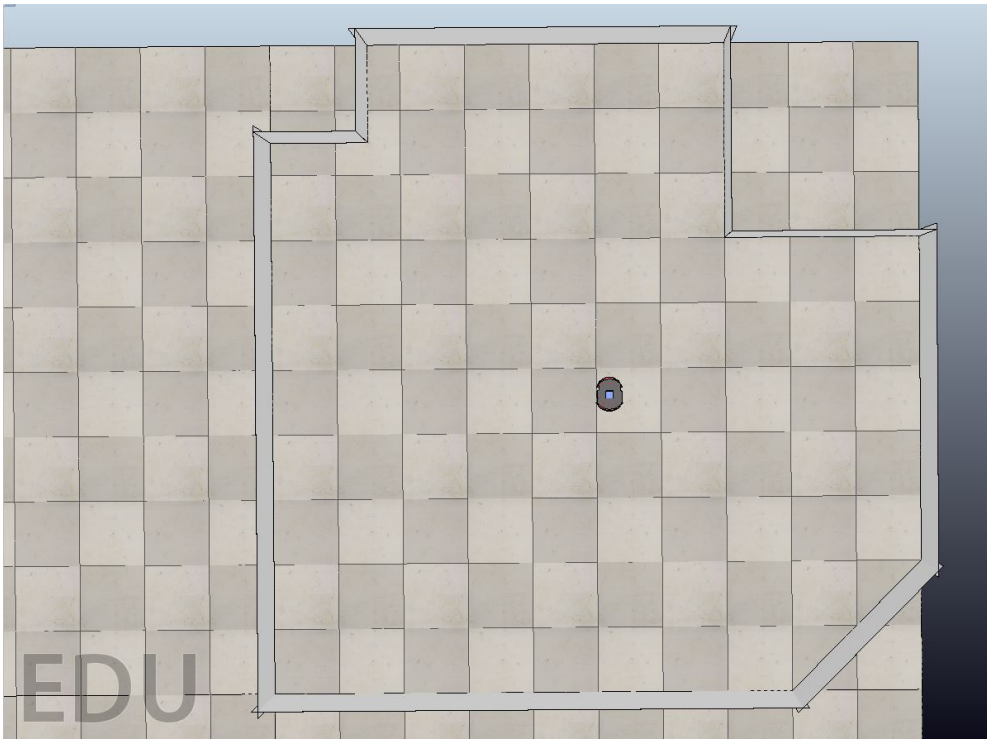
Valores considerados:

$$\text{Posição inicial do robô } p = \begin{bmatrix} 2,625\ m \\ 4,050\ m \\ -90^o \end{bmatrix}$$

16 feixes de *laser*, um a cada 12^o;

1500 iterações;

Figura 17 – Mapa 1.



Fonte: elaborada pelo autor.

$$\text{Ruído Gaussiano } W{\sim}N(0;0,0025);$$

$$\hat{W} = diag(0,0024;0,0024;0,0027;0,0025;0,0028;0,0025;0,0025;0,0023;$$

$$0,0026;0,0025;0,0024;0,0026;0,0027;0,0027;0,0024;0,0023);$$

$$k_{\mathrm{r}} = 0,01;$$

$$k_{\mathrm{l}} = 0,01;$$

$$v_{\mathrm{r}} = 0,5\ dm/s;$$

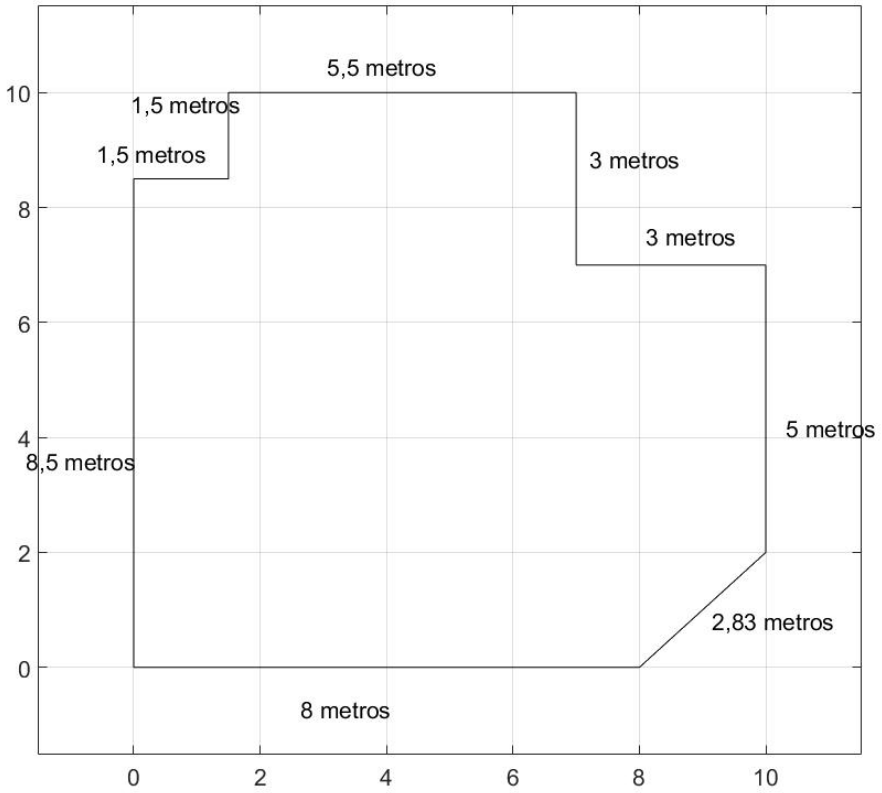
$$v_{\mathrm{l}} = 0,4\ dm/s;$$

Resultados obtidos:

$$\text{Posição final a Priori: } pf_{\text{priori}} = \begin{bmatrix} 5,3979\ m \\ 3,3340\ m \\ 61,1632^{o} \end{bmatrix}$$

$$\text{Posição final a Posteriori: } pf_{\text{posteriori}} = \begin{bmatrix} 5,3907\ m \\ 3,3063\ m \\ 54,9868^{o} \end{bmatrix}$$

Figura 18 – Mapa 1 e suas dimensões.



Fonte: elaborada pelo autor.

$$\text{Posição final real: } pf_{\text{real}} = \begin{bmatrix} 5,3758\ m \\ 3,2541\ m \\ 61,2263^{o} \end{bmatrix}$$

Para analisar os dados obtidos na Figura 19 e na Figura 20, foram computados os Erros Médios Quadráticos (EMQs) das posições a Priori e a Posteriori em relação a x, y e θ.

Dados a Priori:

$$EMQ_{\text{x}}{}^{priori} = 0.0011$$

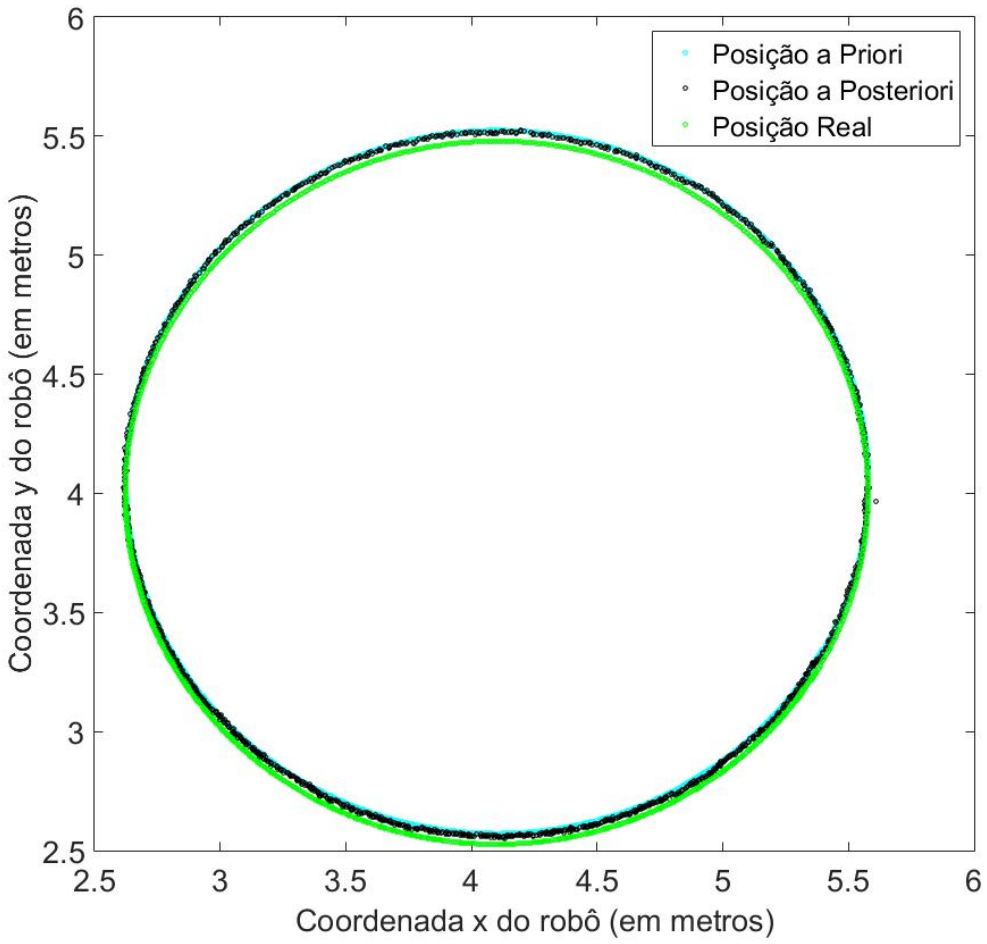
$$EMQ_{\text{y}}{}^{priori} = 0.0028$$

$$EMQ_{\text{theta}}{}^{priori} = 1.5952e^{-06}$$

Dados a Posteriori:

$$EMQ_{\text{x}}{}^{posteriori} = 4.2069e^{-04}$$

Figura 19 – Posições dadas a Priori, a Posteriori e Real do robô no plano 2D.

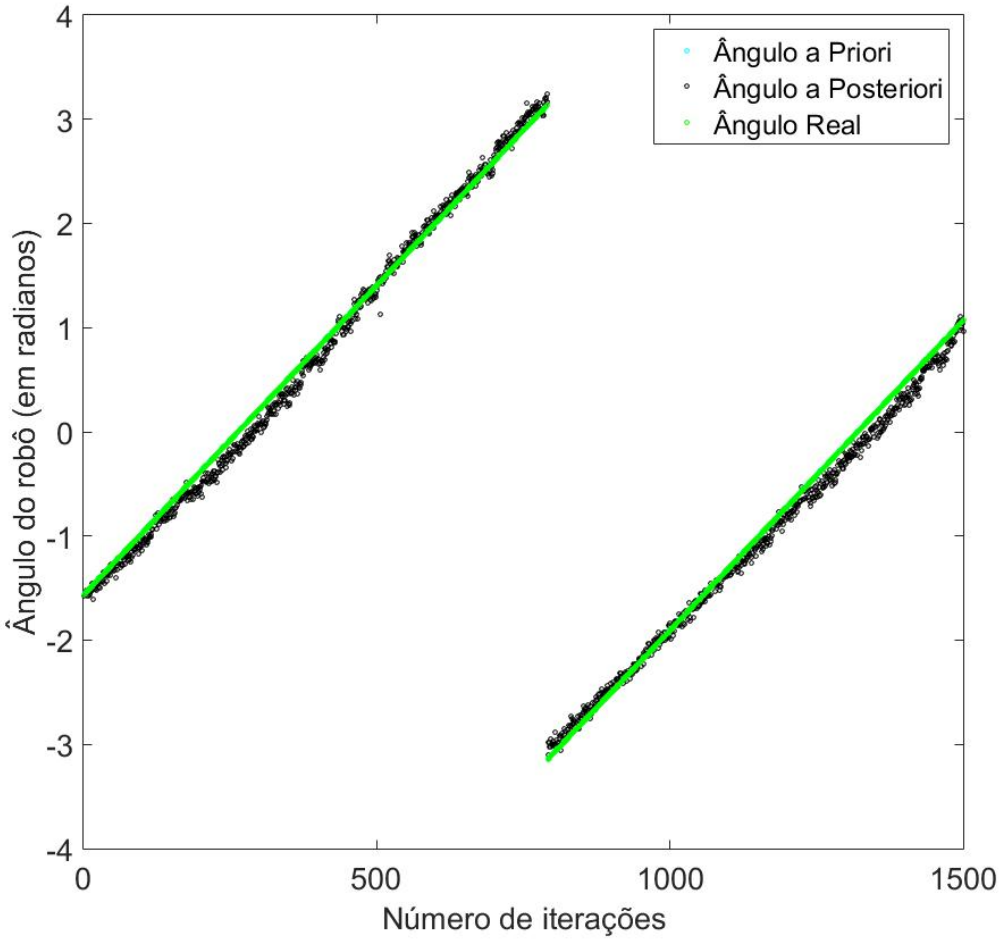


Fonte: elaborada pelo autor.

$$EMQ_{\mathrm{y}}{}^{posteriori} = 0.0015$$

$$EMQ_{\mathrm{theta}}{}^{posteriori} = 0.0112$$

Figura 20 – Ângulos a Priori, a Posteriori e Real do robô para todas as iterações.



Fonte: elaborada pelo autor.

4.1.2 **Experimento 2**

Valores considerados:

$$\text{Posição inicial do robô } p = \begin{bmatrix} 2,625\ m \\ 4,050\ m \\ -90^o \end{bmatrix}$$

16 feixes de *laser*, um a cada 12^o;

1500 iterações;

Ruído Gaussiano $W{\sim}N(0; 0,01)$;

$$\hat{W} = diag(0,0095; 0,0094; 0,0109; 0,0101; 0,0111; 0,0098; 0,0102; 0,0093;$$

$$0,0104; 0,0098; 0,0093; 0,0103; 0,0108; 0,0108; 0,0096; 0,0094);$$

$$k_{\mathrm{r}} = 0,01;$$

$$k_{\mathrm{l}} = 0,01;$$

$$v_{\mathrm{r}} = 0,5\ dm/s;$$

$$v_{\mathrm{l}} = 0,4\ dm/s;$$

Resultados obtidos:

$$\text{Posição final a Priori: } pf_{\text{priori}} = \begin{bmatrix} 5,3906\ m \\ 3,3421\ m \\ 61,3179^o \end{bmatrix}$$

$$\text{Posição final a Posteriori: } pf_{\text{posteriori}} = \begin{bmatrix} 5,3795\ m \\ 3,3282\ m \\ 62,2118^o \end{bmatrix}$$

$$\text{Posição final real: } pf_{\text{real}} = \begin{bmatrix} 5,3672\ m \\ 3,2582\ m \\ 61,2664^o \end{bmatrix}$$

Agora, como anteriormente, obtém-se os Erros Médios Quadráticos das posições a Priori e a Posteriori em relação a x, y e θ para analisar a Figura 21 e a Figura 22.

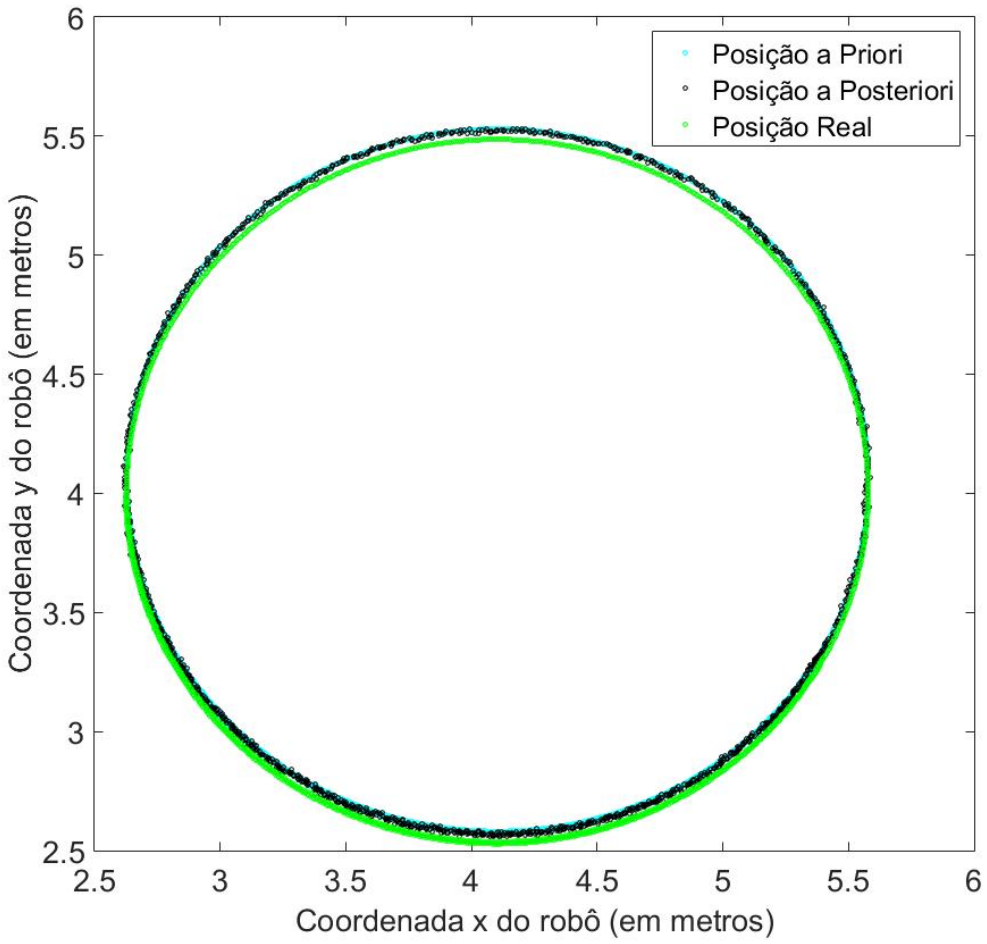
Dados a Priori:

$$EMQ_{\mathrm{x}}{}^{priori} = 0.0011$$

$$EMQ_{\mathrm{y}}{}^{priori} = 0.0027$$

$$EMQ_{\mathrm{theta}}{}^{priori} = 0.0263$$

Figura 21 – Posições dadas a Priori, a Posteriori e Real do robô no plano 2D.



Fonte: elaborada pelo autor.

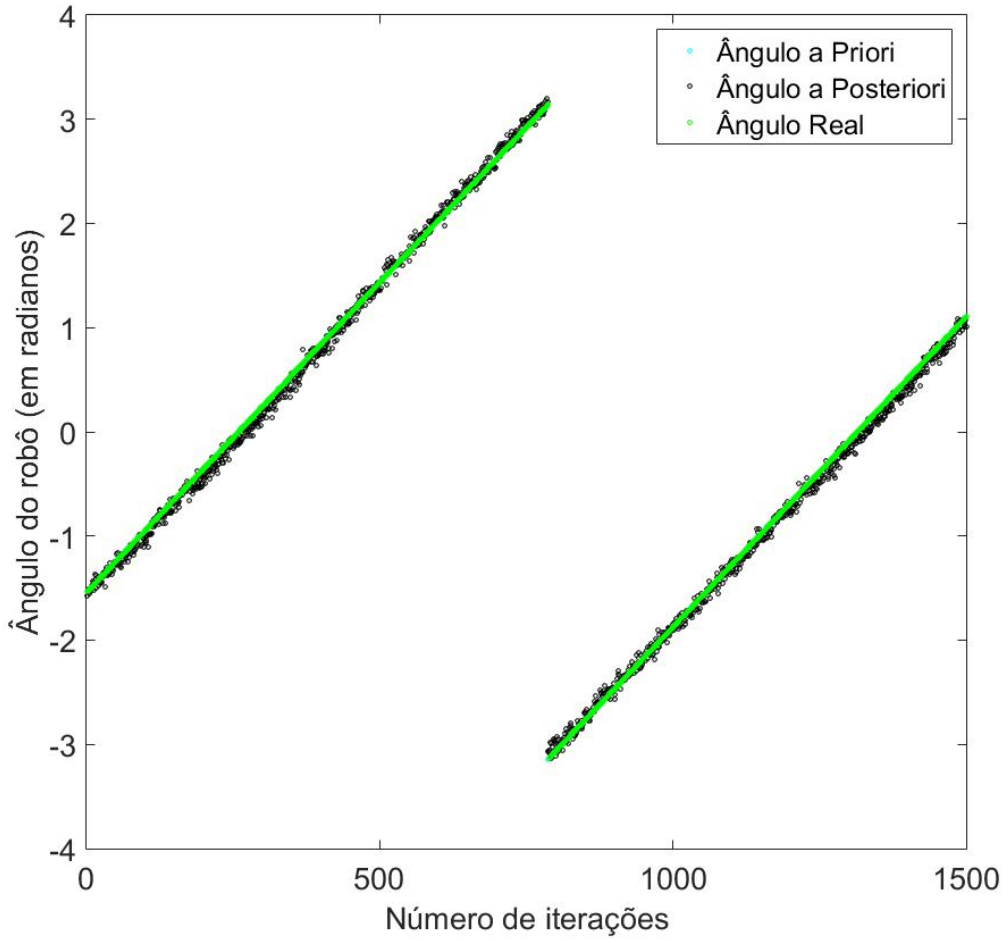
Dados a Posteriori:

$$EMQ_{\mathrm{x}}{}^{posteriori} = 7.0454e^{-04}$$

$$EMQ_{\mathrm{y}}{}^{posteriori} = 0.0019$$

$$EMQ_{\mathrm{theta}}{}^{posteriori} = 0.0301$$

Figura 22 – Ângulos a Priori, a Posteriori e Real do robô para todas as iterações.



Fonte: elaborada pelo autor.

4.1.3 **Experimento 3**

Valores considerados:

$$\text{Posição inicial do robô } p = \begin{bmatrix} 2,625\ m \\ 4,050\ m \\ -90^o \end{bmatrix}$$

16 feixes de *laser*, um a cada 12^o;

1500 iterações;

Ruído Gaussiano $W{\sim}N(0; 0,0001)$;

$$\hat{W} = 1.0e^{-3}diag(0,1010; 0,1066; 0,1030; 0,0922; 0,0996; 0,1037; 0,1065; 0,0968;$$

$$0,1042; 0,1065; 0,1059; 0,1157; 0,1092; 0,0940; 0,1085; 0,0928);$$

$$k_{\mathrm{r}} = 0,01;$$

$$k_{\mathrm{l}} = 0,01;$$

$$v_{\mathrm{r}} = 0,5\ dm/s;$$

$$v_{\mathrm{l}} = 0,4\ dm/s;$$

Resultados obtidos:

$$\text{Posição final a Priori: } pf_{\text{priori}} = \begin{bmatrix} 5,3920\ m \\ 3,3383\ m \\ 61,0658^o \end{bmatrix}$$

$$\text{Posição final a Posteriori: } pf_{\text{posteriori}} = \begin{bmatrix} 5,3603\ m \\ 3,2620\ m \\ 43,4245^o \end{bmatrix}$$

$$\text{Posição final real: } pf_{\text{real}} = \begin{bmatrix} 5,3702\ m \\ 3,2596\ m \\ 61,1461^o \end{bmatrix}$$

Os Erros Médios Quadráticos (EQMs) (Figura 23 e Figura 24).

Dados a Priori:

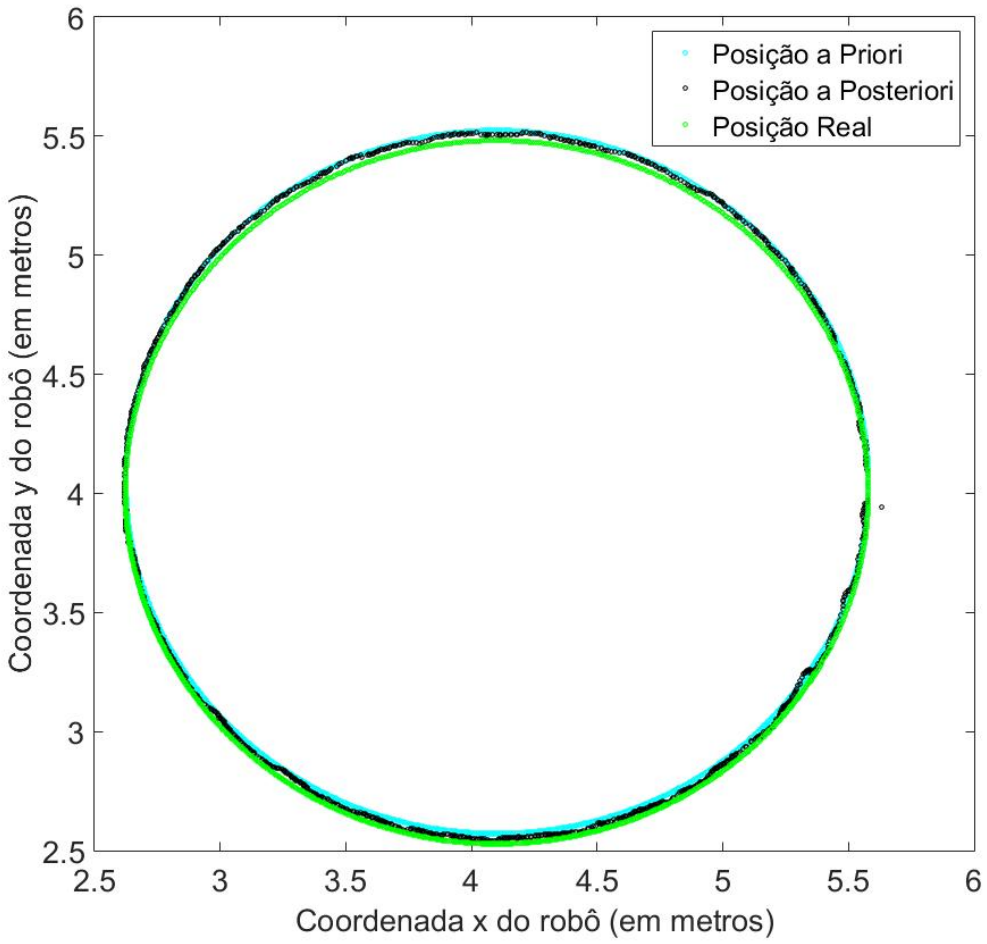
$$EMQ_{\mathrm{x}}{}^{priori} = 9.8851e^{-04}$$

$$EMQ_{\mathrm{y}}{}^{priori} = 0.0025$$

$$EMQ_{\mathrm{theta}}{}^{priori} = 1.0135e^{-06}$$

Dados a Posteriori:

Figura 23 – Posições dadas a Priori, a Posteriori e Real do robô no plano 2D.



Fonte: elaborada pelo autor.

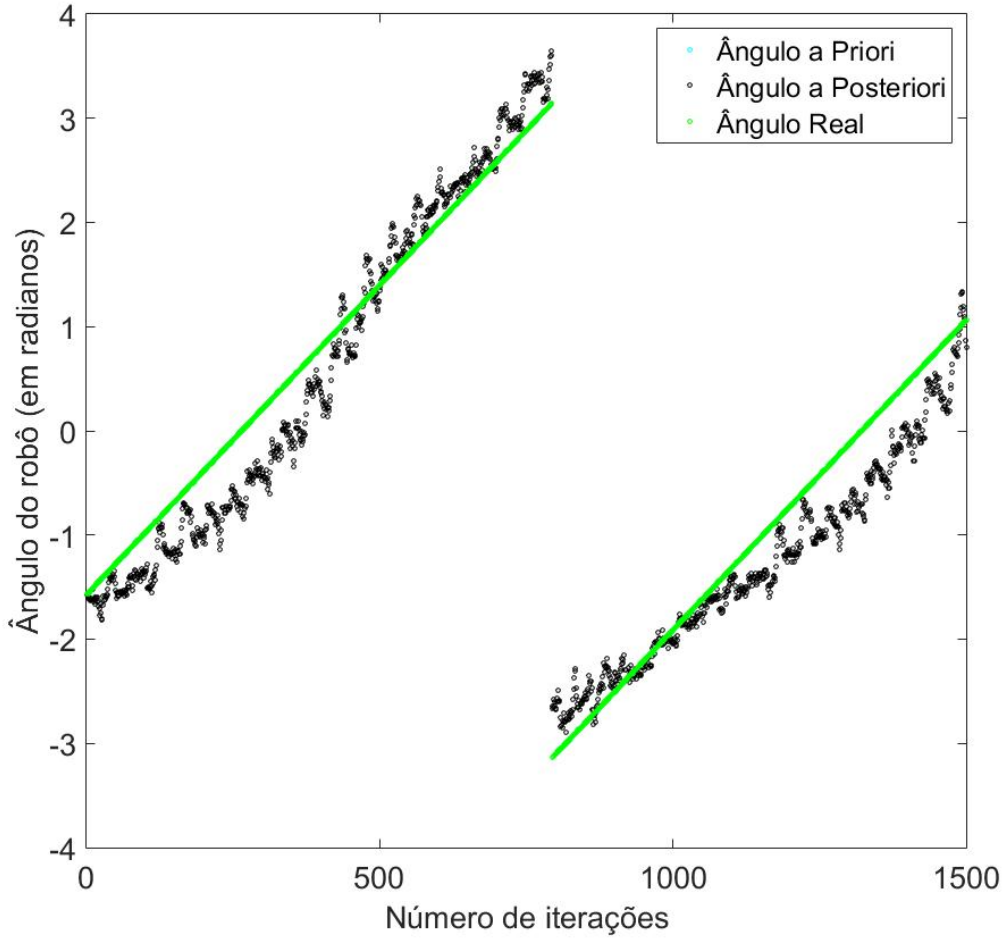
$$EMQ_{\mathrm{x}}{}^{posteriori} = 1.2950e^{-04}$$

$$EMQ_{\mathrm{y}}{}^{posteriori} = 7.7468e^{-04}$$

$$EMQ_{\mathrm{theta}}{}^{posteriori} = 0.1560$$

A Tabela 1 resume os valores encontrados para os erros nos três experimentos do mapa 1. Houve uma redução do EMQ em x e em y da etapa de predição para a etapa de correção, enquanto o EMQ em θ aumentou. Na medida em que o ruído Gaussiano aumentou, os erros também aumentaram.

Figura 24 – Ângulos a Priori, a Posteriori e Real do robô para todas as iterações.



Fonte: elaborada pelo autor.

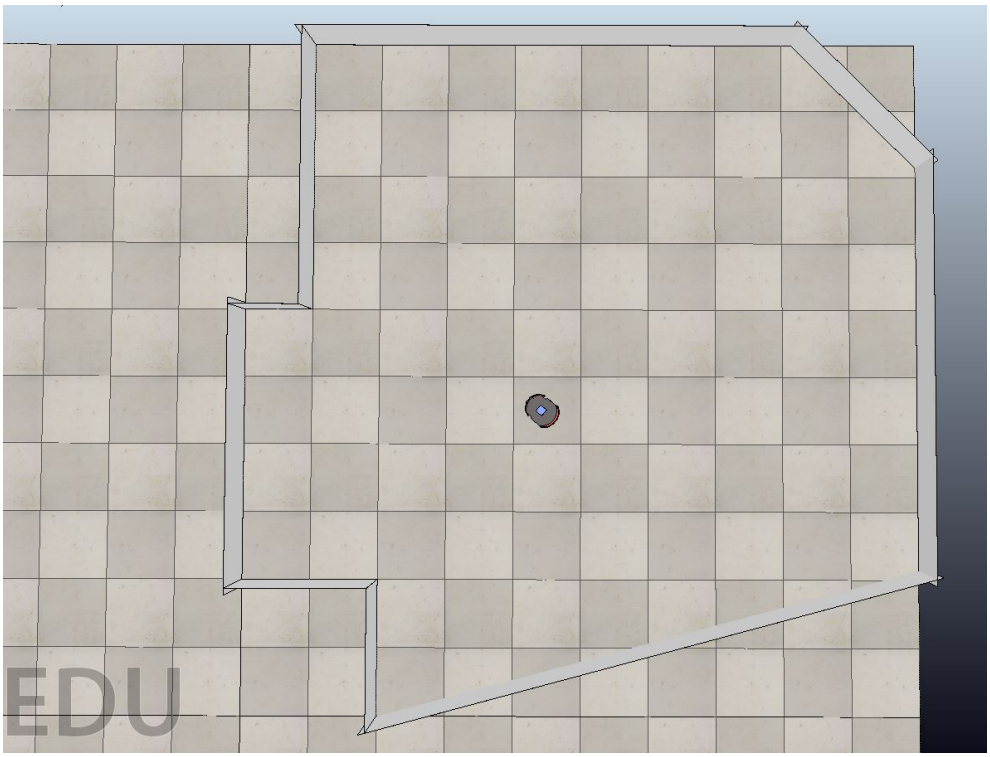
Tabela 1 – Erros Médios Quadráticos para os três experimentos do mapa 1.

	Priori	Posteriori	Priori	Posteriori	Priori	Posteriori
	EMQ_{x}		EMQ_{y}		EMQ_{theta}	
${\sigma_{\text{i}}}^2 = 0,0001$	0,0010	0,0001	0,0025	0,0008	0,0000	0,1560
${\sigma_{\text{i}}}^2 = 0,0025$	0,0011	0,0004	0,0028	0,0015	0,0000	0,0112
${\sigma_{\text{i}}}^2 = 0,0100$	0,0011	0,0007	0,0027	0,0019	0,0263	0,0301

Fonte: elaborada pelo autor.

4.2 MAPA 2

Figura 25 – Mapa 2.



Fonte: elaborada pelo autor.

4.2.1 **Experimento 1**

Valores considerados:

$$\text{Posição inicial do robô } p = \begin{bmatrix} 4,425\ m \\ 4,500\ m \\ -40^o \end{bmatrix}$$

21 feixes de *laser*, um a cada 9^o;

1500 iterações;

Ruído Gaussiano $W{\sim}N(0;0,01)$;

$\hat{W} = 0,0092; 0,0099; 0,0106; 0,0101; 0,0104; 0,0103; 0,0098; 0,0113; 0,0093; 0,0092;$

$0,0098; 0,0093; 0,0097; 0,0097; 0,0099; 0,0099; 0,0096; 0,0104; 0,0088; 0,0101; 0,0084);$

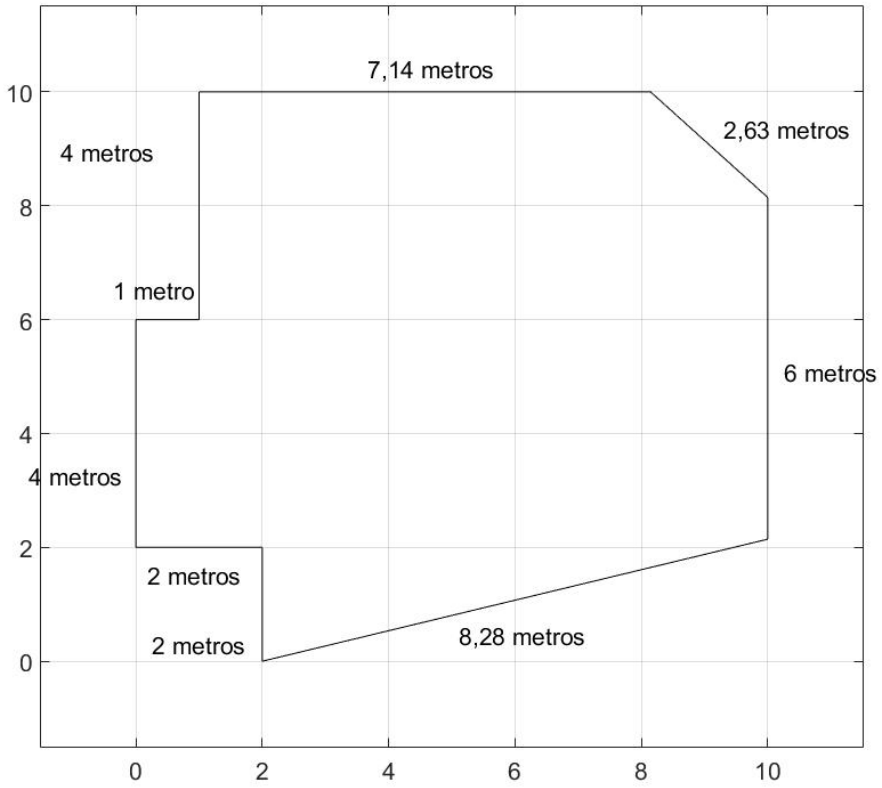
$k_\mathrm{r} = 0,01;$

$k_\mathrm{l} = 0,01;$

$v_\mathrm{r} = 0,5\ dm/s;$

$v_\mathrm{l} = 0,4\ dm/s;$

Figura 26 – Mapa 2 e suas dimensões.



Fonte: elaborada pelo autor.

Resultados obtidos:

$$\text{Posição final a Priori: } pf_{\text{priori}} = \begin{bmatrix} 6,7310\ m \\ 6,2013\ m \\ 112,5518^{o} \end{bmatrix}$$

$$\text{Posição final a Posteriori: } pf_{\text{posteriori}} = \begin{bmatrix} 6,7391\ m \\ 6,1840\ m \\ 111,3543^{o} \end{bmatrix}$$

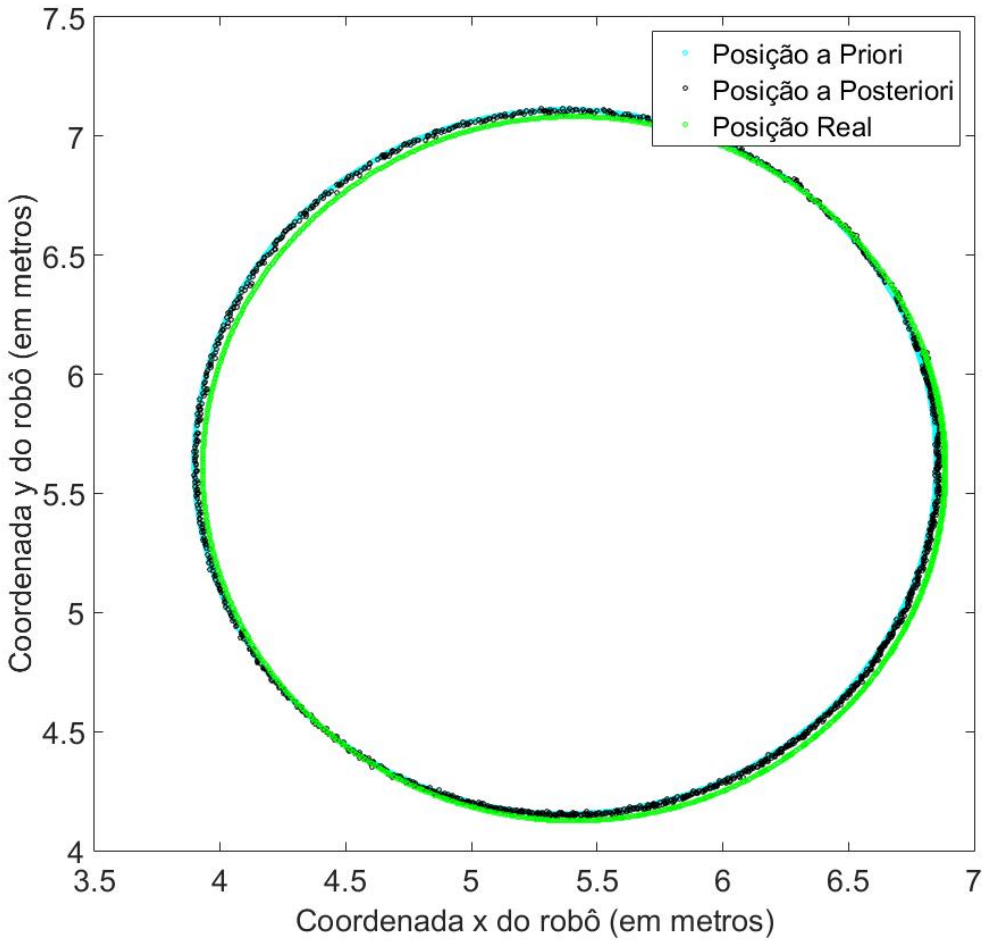
$$\text{Posição final real: } pf_{\text{real}} = \begin{bmatrix} 6,7816\ m \\ 6,1294\ m \\ 112,5060^{o} \end{bmatrix}$$

Os Erros Médios Quadráticos (EQMs) (Figura 27 e Figura 28).

Dados a Priori:

$$EMQ_{\text{x}}{}^{priori} = 0.0016$$

Figura 27 – Posições dadas a Priori, a Posteriori e Real do robô no plano 2D.



Fonte: elaborada pelo autor.

$$EMQ_{\mathrm{y}}{}^{priori} = 0.0022$$

$$EMQ_{\mathrm{theta}}{}^{priori} = 7.1962e^{-07}$$

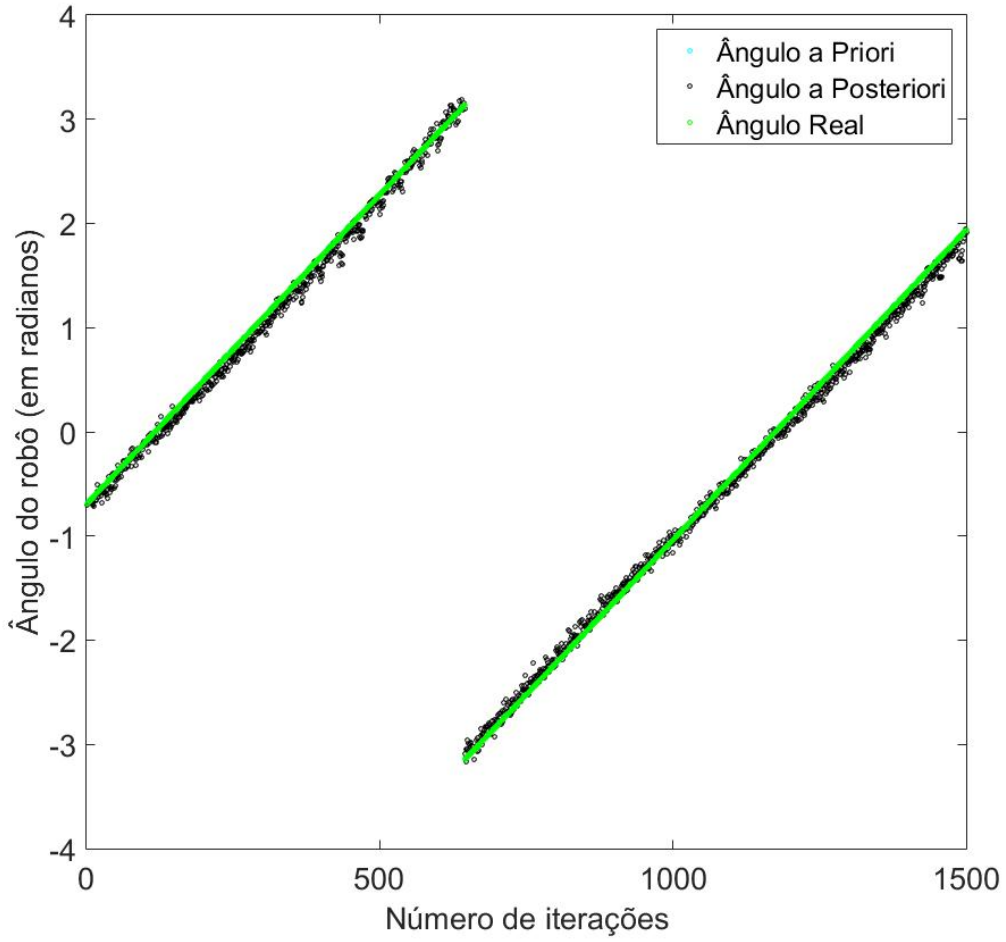
Dados a Posteriori:

$$EMQ_{\mathrm{x}}{}^{posteriori} = 9.3612e^{-04}$$

$$EMQ_{\mathrm{y}}{}^{posteriori} = 0.0013$$

$$EMQ_{\mathrm{theta}}{}^{posteriori} = 0.0067$$

Figura 28 – Ângulos a Priori, a Posteriori e Real do robô para todas as iterações.



Fonte: elaborada pelo autor.

4.2.2 **Experimento 2**

Valores considerados:

$$\text{Posição inicial do robô } p = \begin{bmatrix} 4,425\ m \\ 4,500\ m \\ -40^o \end{bmatrix}$$

21 feixes de *laser*, um a cada 9^o;

1500 iterações;

Ruído Gaussiano $W{\sim}N(0;0,01)$;

$$\hat{W} = 0,0092; 0,0099; 0,0106; 0,0101; 0,0104; 0,0103; 0,0098; 0,0113; 0,0093; 0,0092;$$

$$0,0098; 0,0093; 0,0097; 0,0097; 0,0099; 0,0099; 0,0096; 0,0104; 0,0088; 0,0101; 0,0084);$$

$$k_{\text{r}} = 0,05;$$

$$k_{\text{l}} = 0,05;$$

$$v_{\text{r}} = 0,5\ dm/s;$$

$$v_{\text{l}} = 0,4\ dm/s;$$

Resultados obtidos:

$$\text{Posição final a Priori: } pf_{\text{priori}} = \begin{bmatrix} 6,7552\ m \\ 6,1432\ m \\ 110,4720^o \end{bmatrix}$$

$$\text{Posição final a Posteriori: } pf_{\text{posteriori}} = \begin{bmatrix} 6,7726\ m \\ 6,1096\ m \\ 105,3383^o \end{bmatrix}$$

$$\text{Posição final real: } pf_{\text{real}} = \begin{bmatrix} 6,8011\ m \\ 6,0759\ m \\ 110,5522^o \end{bmatrix}$$

Os Erros Médios Quadráticos (EQMs) (Figura 29 e Figura 30).

Dados a Priori:

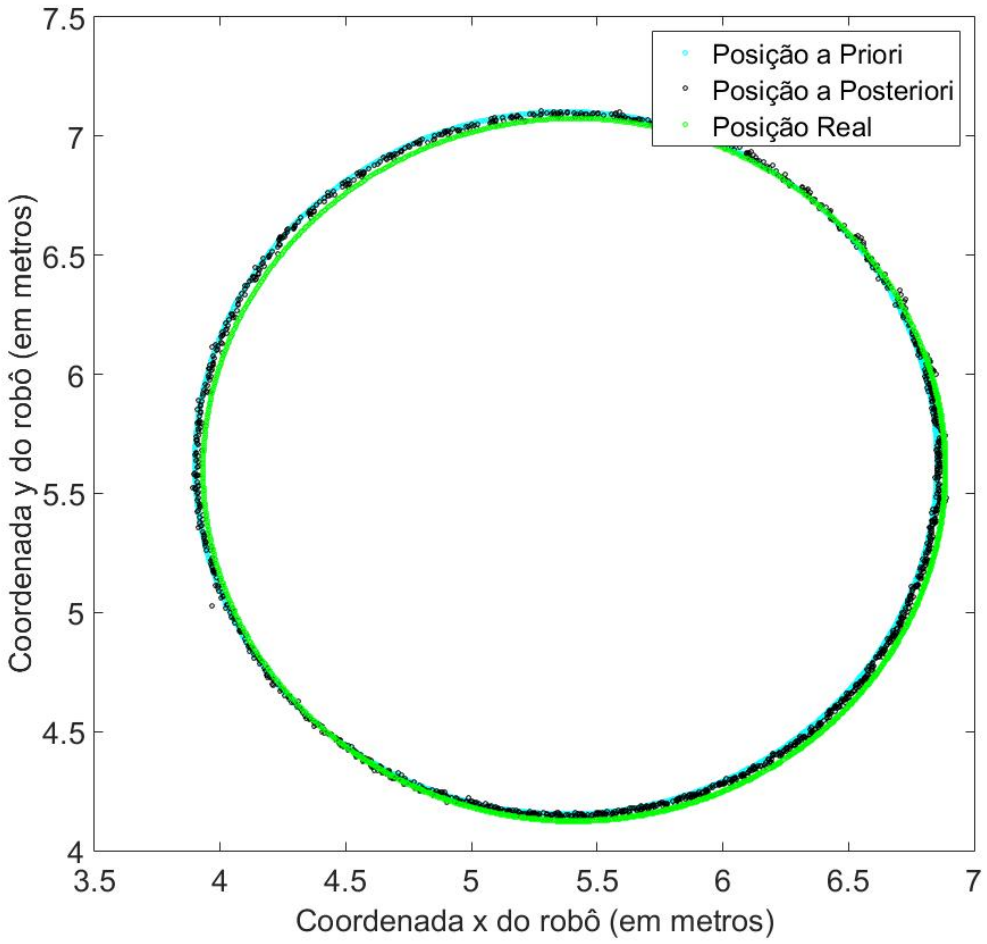
$$EMQ_{\text{x}}{}^{priori} = 0.0015$$

$$EMQ_{\text{y}}{}^{priori} = 0.0020$$

$$EMQ_{\text{theta}}{}^{priori} = 5.8669e^{-07}$$

Dados a Posteriori:

Figura 29 – Posições dadas a Priori, a Posteriori e Real do robô no plano 2D.



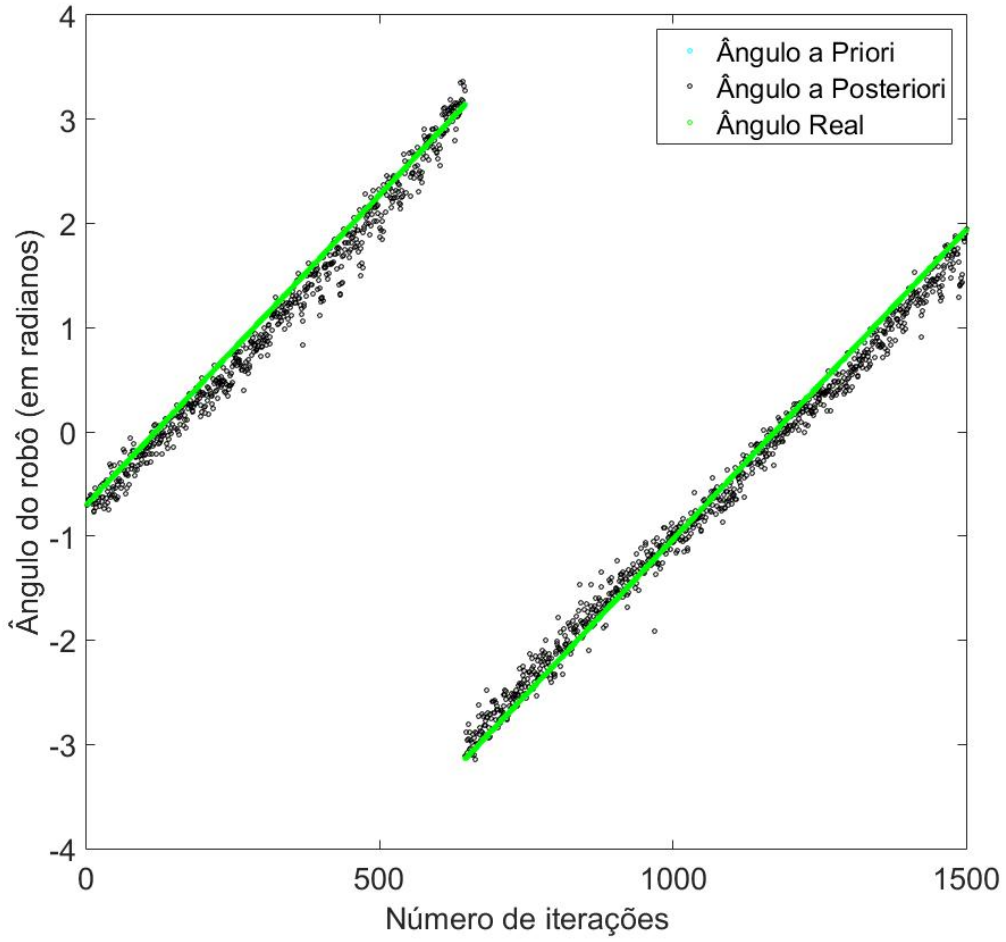
Fonte: elaborada pelo autor.

$$EMQ_{\mathrm{x}}{}^{posteriori} = 5.7499e^{-04}$$

$$EMQ_{\mathrm{y}}{}^{posteriori} = 7.7317e^{-04}$$

$$EMQ_{\mathrm{theta}}{}^{posteriori} = 0.0289$$

Figura 30 – Ângulos a Priori, a Posteriori e Real do robô para todas as iterações.



Fonte: elaborada pelo autor.

4.2.3 **Experimento 3**

Valores considerados:

$$\text{Posição inicial do robô } p = \begin{bmatrix} 4,425\ m \\ 4,500\ m \\ -40^o \end{bmatrix}$$

21 feixes de *laser*, um a cada 9^o;

1500 iterações;

Ruído Gaussiano $W{\sim}N(0;0,01)$;

$$\hat{W} = 0,0092; 0,0099; 0,0106; 0,0101; 0,0104; 0,0103; 0,0098; 0,0113; 0,0093; 0,0092;$$

$$0,0098; 0,0093; 0,0097; 0,0097; 0,0099; 0,0099; 0,0096; 0,0104; 0,0088; 0,0101; 0,0084);$$

$$k_{\mathrm{r}} = 0,1;$$

$$k_{\mathrm{l}} = 0,1;$$

$$v_{\mathrm{r}} = 0,5\ dm/s;$$

$$v_{\mathrm{l}} = 0,4\ dm/s;$$

Resultados obtidos:

$$\text{Posição final a Priori: } pf_{\text{priori}} = \begin{bmatrix} 6,7471\ m \\ 6,1747\ m \\ 111,1996^o \end{bmatrix}$$

$$\text{Posição final a Posteriori: } pf_{\text{posteriori}} = \begin{bmatrix} 6,7503\ m \\ 6,1479\ m \\ 116,6198^o \end{bmatrix}$$

$$\text{Posição final real: } pf_{\text{real}} = \begin{bmatrix} 6,7955\ m \\ 6,1049\ m \\ 111,2111^o \end{bmatrix}$$

Os Erros Médios Quadráticos (EQMs) (Figura 31 e Figura 32).

Dados a Priori:

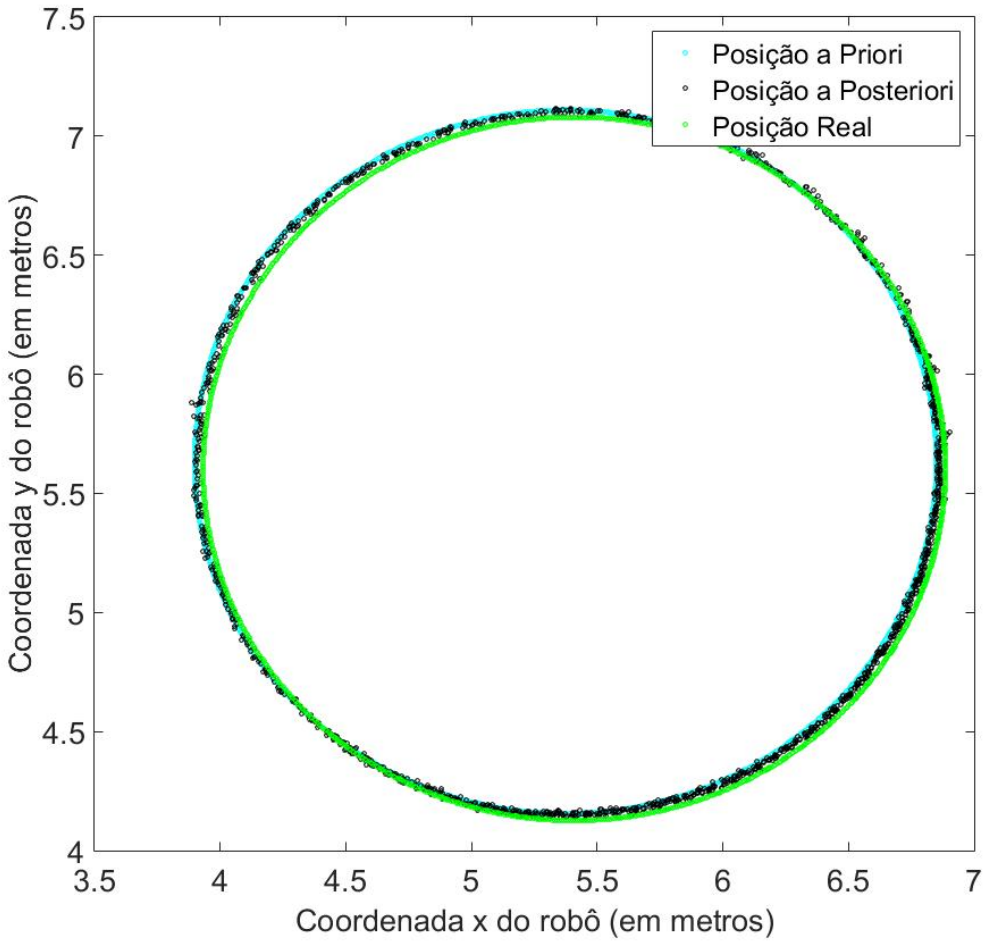
$$EMQ_{\mathrm{x}}{}^{priori} = 0.0017$$

$$EMQ_{\mathrm{y}}{}^{priori} = 0.0023$$

$$EMQ_{\mathrm{theta}}{}^{priori} = 0.0263$$

Dados a Posteriori:

Figura 31 – Posições dadas a Priori, a Posteriori e Real do robô no plano 2D.



Fonte: elaborada pelo autor.

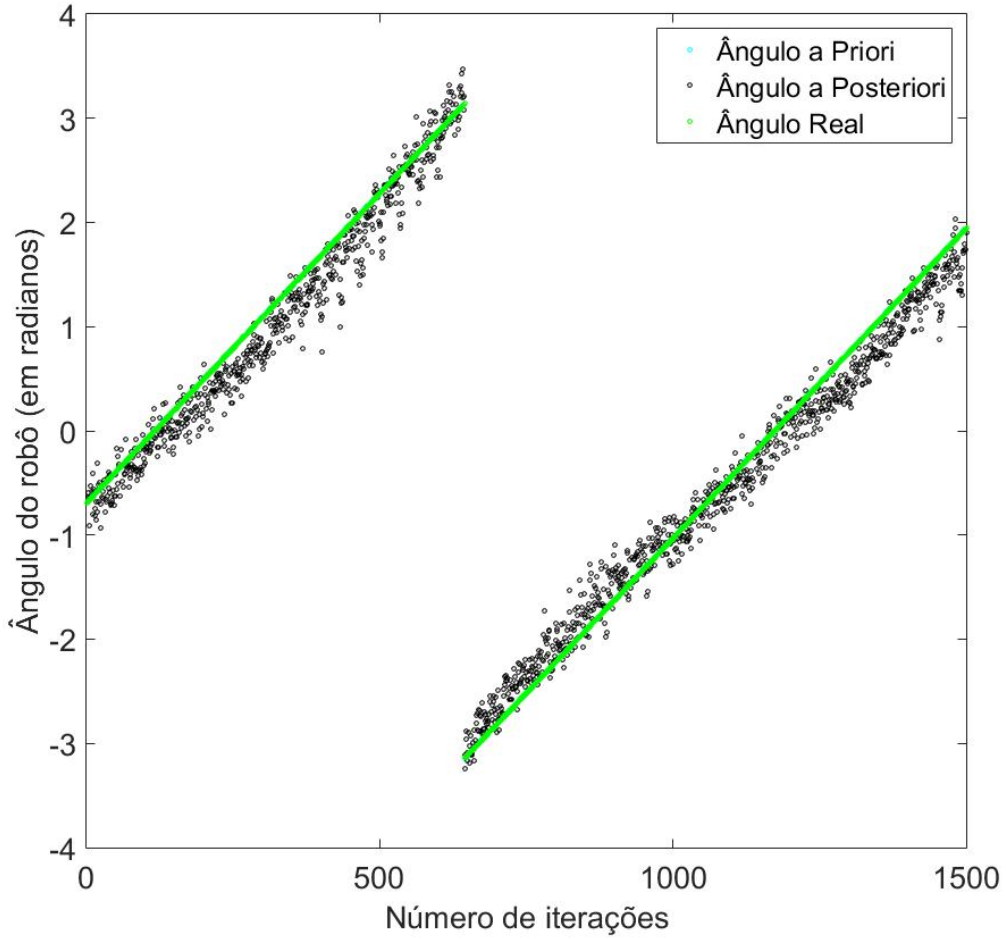
$$EMQ_{\mathrm{x}}{}^{posteriori} = 5.6850e^{-04}$$

$$EMQ_{\mathrm{y}}{}^{posteriori} = 7.5033e^{-04}$$

$$EMQ_{\mathrm{theta}}{}^{posteriori} = 0.0794$$

A Tabela 2 resume os valores encontrados para os erros nos três experimentos do mapa 2. Houve uma redução do EMQ em x e em y da etapa de predição para a etapa de correção, enquanto o EMQ em θ aumentou. Na medida em que as constantes de multiplicação dos deslocamentos das rodas direita e esquerda aumentaram, os erros diminuíram.

Figura 32 – Ângulos a Priori, a Posteriori e Real do robô para todas as iterações.



Fonte: elaborada pelo autor.

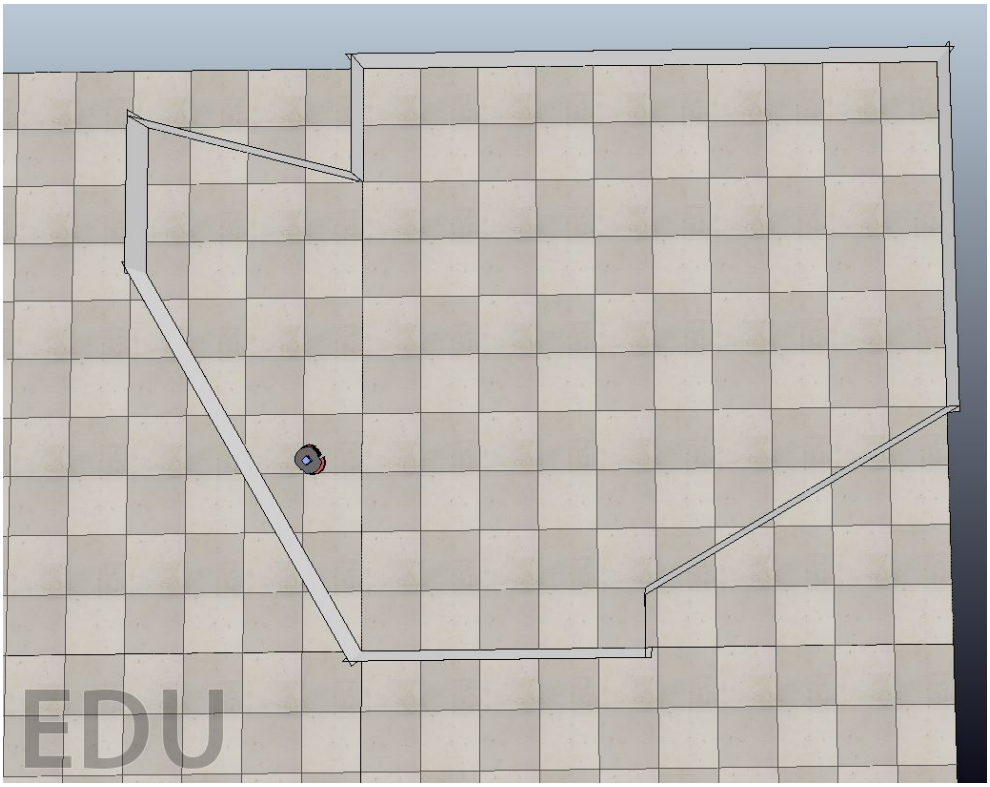
Tabela 2 – Erros Médios Quadráticos para os três experimentos do mapa 2.

	Priori	Posteriori	Priori	Posteriori	Priori	Posteriori
	EMQ_{x}		EMQ_{y}		EMQ_{theta}	
$k_{\mathrm{r}} = 0,01$ e $k_{\mathrm{l}} = 0,01$	0,0016	0,0009	0,0022	0,0013	0,0000	0,0067
$k_{\mathrm{r}} = 0,05$ e $k_{\mathrm{l}} = 0,05$	0,0015	0,0006	0,0020	0,0008	0,0000	0,0289
$k_{\mathrm{r}} = 0,10$ e $k_{\mathrm{l}} = 0,10$	0,0017	0,0006	0,0023	0,0008	0,0263	0,0794

Fonte: elaborada pelo autor.

4.3 MAPA 3

Figura 33 – Mapa 3.



Fonte: elaborada pelo autor.

4.3.1 **Experimento 1**

A partir do mapa acima, foi feito um controle de posição em malha fechada do robô, utilizando, na realimentação, a posição a Posteriori dada pelo Filtro de Kalman implementado.

Valores considerados:

$$\text{Posição inicial do robô } p = \begin{bmatrix} -0,850\ m \\ 3,225\ m \\ 130^o \end{bmatrix}$$

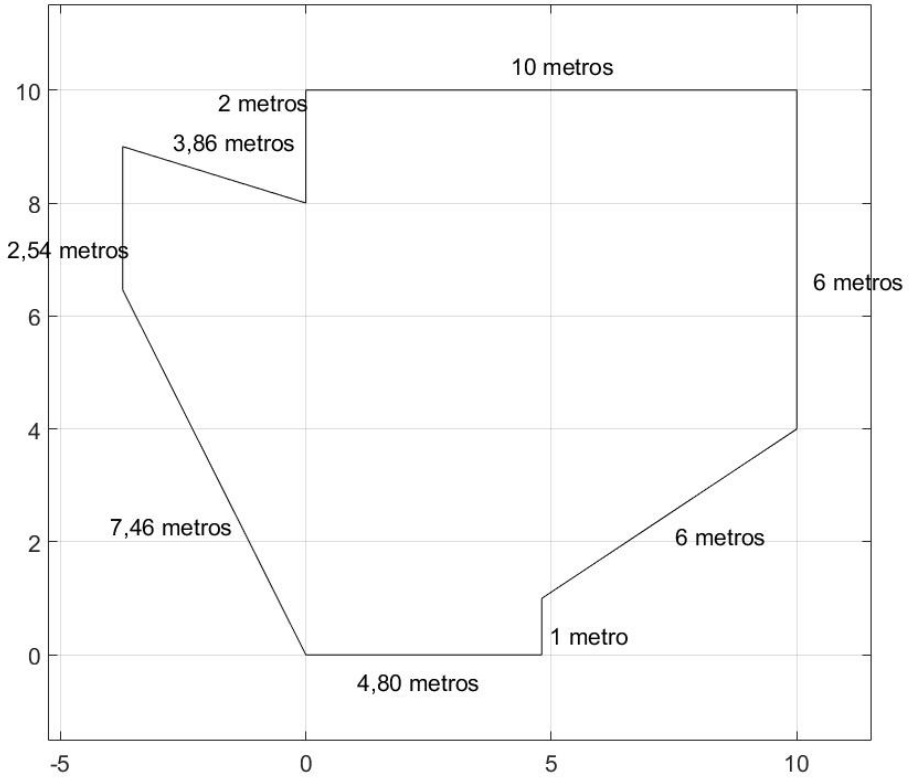
$$\text{Posição objetivo } pg = \begin{bmatrix} 8\ m \\ 8\ m \\ 0^o \end{bmatrix}$$

21 feixes de *laser*, um a cada 9^o;

Ruído Gaussiano $W{\sim}N(0; 0,0025)$;

$$\hat{W} = 0,0023; 0,0029; 0,0023; 0,0024; 0,0027; 0,0028; 0,0024; 0,0028; 0,0027; 0,0025;$$

Figura 34 – Mapa 3 e suas dimensões.



Fonte: elaborada pelo autor.

$$0,0029; 0,0028; 0,0023; 0,0024; 0,0025; 0,0024; 0,0025; 0,0022; 0,0027; 0,0023; 0,0026);$$

$$k_{\mathrm{r}} = 0,01;$$

$$k_{\mathrm{l}} = 0,01;$$

Erros máximos aceitáveis: $0,5\ m$ e 3^o.

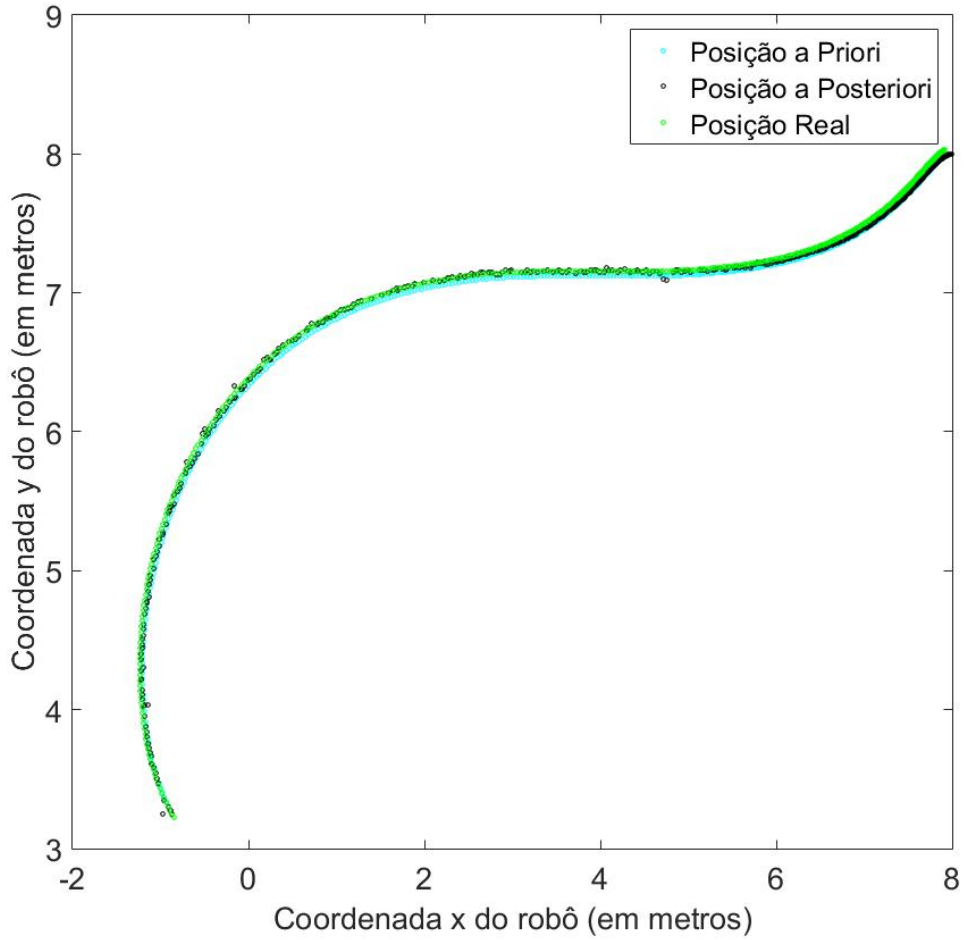
Resultados obtidos:

$$\text{Posição final a Priori: } pf_{\text{priori}} = \begin{bmatrix} 7,9915\ m \\ 7,9994\ m \\ 1,5814^o \end{bmatrix}$$

$$\text{Posição final a Posteriori: } pf_{\text{posteriori}} = \begin{bmatrix} 7,9893\ m \\ 7,9996\ m \\ 2,7044^o \end{bmatrix}$$

$$\text{Posição final real: } pf_{\text{real}} = \begin{bmatrix} 7,9161\ m \\ 8,0309\ m \\ 1,5470^{o} \end{bmatrix}$$

Figura 35 – Posições dadas a Priori, a Posteriori e Real do robô no plano 2D.



Fonte: elaborada pelo autor.

Os Erros Médios Quadráticos (EQMs) (Figura 35 e Figura 36).

Dados a Priori:

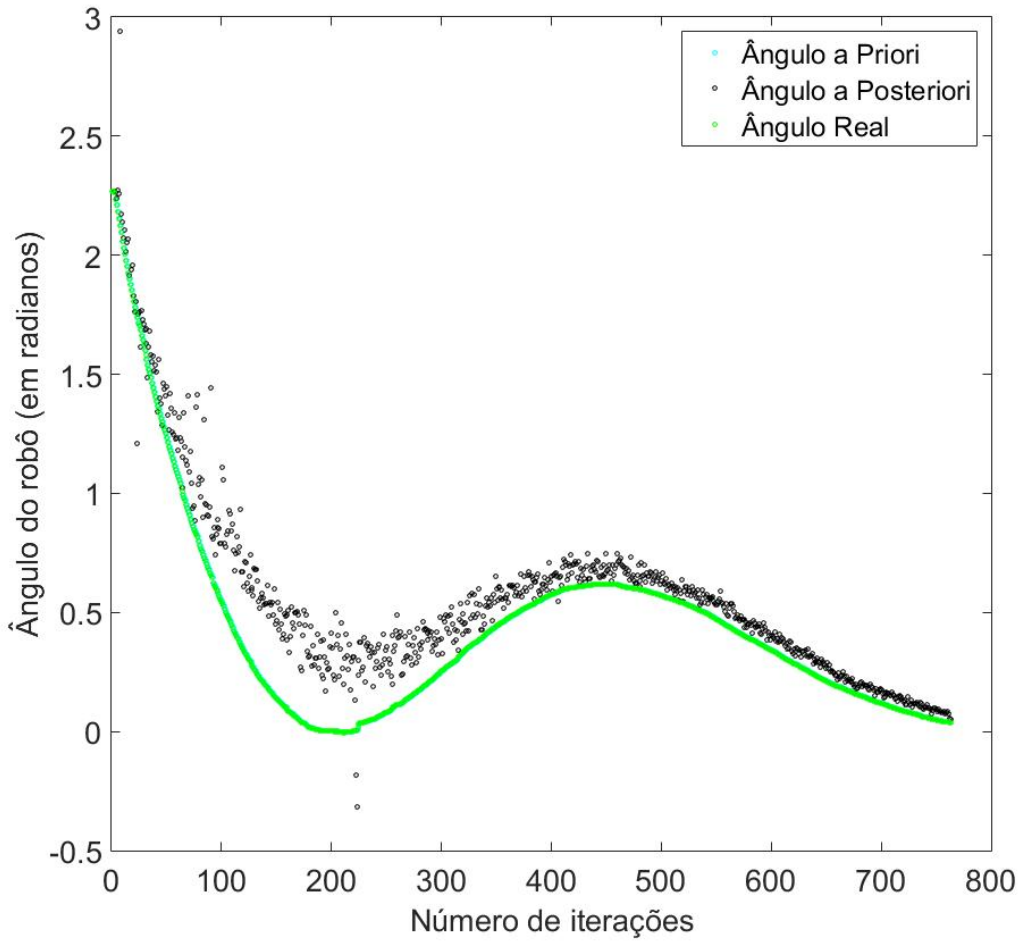
$$EMQ_{\text{x}}{}^{priori} = 0.0033$$

$$EMQ_{\text{y}}{}^{priori} = 6.0642e^{-04}$$

$$EMQ_{\text{theta}}{}^{priori} = 5.4419e^{-05}$$

Dados a Posteriori:

Figura 36 – Ângulos a Priori, a Posteriori e Real do robô para todas as iterações.



Fonte: elaborada pelo autor.

$$EMQ_{\mathrm{x}}{}^{posteriori} = 0.0027$$

$$EMQ_{\mathrm{y}}{}^{posteriori} = 4.5501e^{-04}$$

$$EMQ_{\mathrm{theta}}{}^{posteriori} = 0.0364$$

Observa-se que, em todos os casos, não importando quais são os valores de entrada utilizados no sistema, os Erros Médios Quadráticos para as posições x e y do robô diminuem da etapa de predição (priori) para a etapa de correção (posteriori). Já o Erro Médio Quadrático da orientação θ do robô, opostamente ao que ocorre com x e y, aumenta da etapa de predição para a etapa de correção.

5 CONCLUSÃO

O trabalho teve o objetivo de obter a localização no espaço bidimensional do robô diferencial *p3dx* utilizando dois sensores diferentes (*encoder* e *laser scanner* 2D), além da ferramenta estatística, o Filtro de Kalman. Este filtro é uma ferramenta que consegue reunir os dados coletados por ambos sensores e entrega, portanto, informações que são mais precisas do que aquelas entregues pelos dois sensores de formas individuais.

A hodometria, de uma simples maneira, já dá informações suficientes para obter-se a localização do robô. Entretanto, à medida que o tempo passa e o robô se desloca no espaço, erros na hodometria são acumulados por diversos fatores como, por exemplo, o escorregamento das rodas direita e esquerda na superfície em que ele se encontra. Por esse motivo que, neste trabalho, foi utilizado o Filtro de Kalman em conjunto com o sensor *laser rangefinder*, com o propósito de não permitir que tais erros de posição possam se acumular e, assim, manter uma informação precisa acerca da localização do mesmo.

Como a cinemática do robô e as medidas dadas pelo *laser* são funções não-lineares, foi necessário, dessa forma, linearizar o sistema utilizando o Filtro de Kalman Estendido (EKF), obtendo respostas que foram satisfatórias.

Além dos sensores *encoder* e *laser scanner* 2D foi necessário, também, implementar o sensor *laser* virtual com o intuito de chegar a um modelo das medidas para comparação com as medidas reais e, desta forma, corrigir a posição dada na etapa de predição. Essa simples ferramenta se mostrou extremamente poderosa e eficaz.

Os resultados foram suficientemente satisfatórios, uma vez que os erros em relação às posições x e y do robô diminuíram da etapa de predição para a etapa de correção. De um outro lado, o erro em relação à orientação θ do robô aumentou da etapa de predição para a etapa de correção.

5.1 TRABALHOS FUTUROS

De forma a dar procedência a este trabalho é possível, com o conhecimento adquirido e códigos já utilizados, implementar a ideia em um robô *p3dx* real, ou qualquer outro robô diferencial, desde que se assuma que o ruído do sensor *laser* seja Gaussiano, ou aproximadamente Gaussiano.

Pode-se também, utilizar um outro sensor, como o sensor ultrassônico que é mais barato do que o sensor *laser*, porém menos robusto. A ideia por trás do sensor ultrassônico é a mesma do que a do *laser*.

Uma outra maneira de melhorar os resultados é utilizar o robô em alguma superfície cujo coeficiente de atrito seja o suficiente para evitar que ocorra o deslizamento das rodas direita e esquerda, minimizando, assim, os erros na hodometria.

Este trabalho pode ser futuramente desenvolvido de maneira a implementar um algoritmo de mapeamento e localização simultâneos (SLAM) no robô.

REFERÊNCIAS

[1] KALMAN, R. E. *A New Approach to Linear Filtering and Prediction Problems.* Research Institute for Advanced Study, Baltimore, Md. 1960.

[2] SORENSON, H. W. *Least-squares estimation: from Gauss to Kalman.* University of California, San Diego. July 1970.

[3] CORKE, P. *Robotics, Vision and Control: Fundamental Algorithms in Matlab.* Springer, 2011.

[4] THRUN, S. *Probabilistic Robotics.* 1999-2000.

[5] SIEGWART, R. and NOURBAKHSH, I. R. *Introduction to Autonomous Mobile Robots.* The MIT Press, Cambridge, Massachusetts, 2004.

[6] BUSSE, Franz D.; HOW, Johnathan P. and SIMPSON, James. *Demonstration of Adaptive Extended Kalman Filter for Low-Earth-Orbit Formation Estimation Using CDGPS.* Journal of The Institute of Navigation, 2003.

[7] PEARSON III, J. B. and STEAR, E. B. *Kalman Filter Applications In Airborne Radar Tracking.* May 1974.

[8] PINTO, F. P. V. F. *The Cleaning Robot Project: Aplicação do Filtro de Kalman na Auto-Localização de um Sistema Robótico Autónomo.* Universidade do Porto, Abril 2008.

[9] JETTO, L.; LONGHI, S. and VENTURINI, G. *Development and Experimental Validation of an Adaptive Extended Kalman Filter for the Localization of Mobile Robots.* IEEE Transactions on Robotics and Automation, April 1999.

[10] NEGENBORN, R. *Robot Localization and Kalman Filters: On finding your position in a noisy world.* Utrecht University. September 1, 2003.

[11] HOFSTATTER, G. *Filtro de Kalman aplicado à localização de robôs móveis.* Faculdade de Engenharia, UFJF.

[12] LI, Qiang; LI, Ranyang; JI, Kaifan and DAI, Wei. *Kalman Filter and Its Application.* Kunming University of Science and Technology, China.

[13] VIET, Chau N. and MARSHALL, Ian. *Vision-Based Obstacle Avoidance for a Small, Low-Cost Robot.* Computer Science Department, University of Kent, Canterbury, United Kingdom.

[14] WOLF, Denis F.; OSÓRIO, Fernando S.; SIMÕES, Eduardo; TRINDADE JR., Onofre. *Intelligent Robotics: From Simulation to Real World Applications* Mobile Robotics Laboratory - LRM, Universidade de São Paulo, São Carlos-SP, Brasil.

[15] BORENSTEIN, J.; EVERETT, H. R. and FENG, L. *Where am I? - Systems and Methods for Mobile Robot Positioning.* The University of Michigan, 1996.

[16] LIMA, David da Silva. *Localização Absoluta de Robôs Móveis em Ambientes Industriais.* Faculdade de Engenharia da Universidade do Porto.

[17] ROUMELIOTIS, Stergios I. and BEKEY, George A. *Bayesian estimation and Kalman filtering: A unified framework for Mobile Robot Localization.* University of Southern California, Los Angeles, CA.

[18] FOX, D.; WOLFRAM, B. and THRUN, S. *Markov Localization for Mobile Robots in Dynamic Environments.* Journal of Artificial Intelligence Research.

[19] MAZZOTTI, Bruno F. *Co-projeto de Hardware/Software do Filtro de Partículas para Localização em Tempo Real de Robôs Móveis.* Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos-SP.

[20] OLIVI, L. R. *Material de aula da disciplina de Robótica Móvel.* Faculdade de Engenharia, UFJF.

[21] *https://www.mathworks.com/help/matlab*

[22] *https://www.coppeliarobotics.com*

APÊNDICE A – Algoritmo para calcular a matriz $\hat{W}$

```
1  ths=-90:9:90; % Define os angulos de cada sensor no frame do
       robo
2
3  vrep=remApi('remoteApi');
4  vrep.simxFinish(-1);
5  robot=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
6  if (robot~=-1)
7      disp('Connected to remote API server')
8  else
9      disp('Connection not succesful')
10 end % Estabelece conexao inicial com o V-Rep
11
12 inter=500; % Estabelece o numero de amostras a serem colhidas
       para se obter a matriz de covariancia W
13 for k=1:inter
14     [errorCode,Distance_Value]=vrep.simxGetStringSignal(robot,'
           Distance_table',vrep.simx_opmode_oneshot_wait);
15     if(errorCode==vrep.simx_return_ok)
16         Dist=vrep.simxUnpackFloats(Distance_Value); % Colhe os
               dados de distancias dadas pelo sensor laser
17     end
18     for i=1:length(Dist)
19         Dist(i)=Dist(i)+0.05*randn; % Acrescenta um ruido
               Gaussiano aos valores de distancia
20     end
21     for i=1:length(ths)
22         D{k}(i)=Dist((abs(ths(1)-ths(2))*(i-1))+1); % Armazena
               apenas as distancias nos angulos de interesse. Exemplo:
                -90,-45,0,45 e 90.
23     end
24 end
25
26 for i=1:length(ths)
27     sum=0;
28     for k=1:inter
29         sum=sum+D{k}(i);
30     end
```

```
31      mu(i)=sum/inter; % Encontra a media aritmetica de todas as
            medidas
32  end
33
34  for i=1:length(ths)
35      sum=0;
36      for k=1:inter
37          sum=sum+(D{k}(i)-mu(i))^2;
38      end
39      var(i)=sum/inter; % Encontra a variancia de todas as medidas
40  end
41
42  for i=1:length(ths)
43      for j=1:length(ths)
44          if i==j
45              W(i,j)=var(i); % Elementos da diagonal principal da
                    matriz de covariancia
46          else
47              W(i,j)=0; % Elementos fora da diagonal principal da
                    matriz de covariancia
48          end
49      end
50  end
```

APÊNDICE B – Função que implementa o sensor laser virtual

```
1  function Dist=SensorLaserVirtual(R,ths,L) % R -> posicao do robo
       ; ths -> angulos de leitura do laser; L -> mapa
2
3      R.th=rad2deg(R.th); % Converte o angulo da posicao do robo de
           radianos para graus
4
5      for j=1:size(ths,2)
6          Dist(j)=inf; % Inicializa todas as distancias com os
               valores sendo infinitos
7      end
8
9      for j=1:size(ths,2)
10         % Obtencao da reta que direciona onde o sensor esta
               medindo
11
12         x1=R.x;
13         y1=R.y; % Obtem o primeiro ponto da reta
14         thS=ths(j); % Obtem o angulo do sensor j
15         x2=x1+cosd(R.th+thS);
16         y2=y1+sind(R.th+thS); % Obtem o segundo ponto da reta
17         a=y1-y2;
18         b=x2-x1;
19         c=x1*y2-x2*y1; % Considerando a equacao da reta como sendo
               : ax+by+c=0
20
21         % Algoritmo para achar o ponto de intersecao entre a reta
               e o obstaculo
22
23         n=0; % Inicializa um contador
24         for i=1:size(L,1)
25             x3=L(i,1);
26             y3=L(i,2); % Primeiro ponto da reta que forma a parede
                   i
27             x4=L(i,3);
28             y4=L(i,4); % Segundo ponto da reta que forma a parede i
29             den=(x1-x2)*(y3-y4)-(y1-y2)*(x3-x4);
30
```

```
31          if den~=0 % Verifica se as retas do feixe de luz do
                laser e a da parede sao paralelas ou nao
32              Px=((x1*y2-y1*x2)*(x3-x4)-(x1-x2)*(x3*y4-y3*x4))/den
                    ;
33              Py=((x1*y2-y1*x2)*(y3-y4)-(y1-y2)*(x3*y4-y3*x4))/den
                    ; % Encontra o ponto de interseccao entre a reta
                    do feixe de luz do laser e a reta que forma a
                    parede
34
35              thP=mod(rad2deg(atan2(Py-R.y,Px-R.x))-R.th,360); %
                    Encontra o angulo do ponto de interseccao obtido
                    anteriormente
36              if thP>180
37                  thP=thP-360;
38              end
39
40              epsilon=0.01;
41              if abs(thS-thP)<epsilon % Verifica se os angulos thS
                     e thP sao aproximadamente iguais (pequenos erros
                     computacionais)
42                  if ((min(x3,x4)-0.01<=Px) && (Px<=max(x3,x4)
                        +0.01) && (min(y3,y4)-0.01<=Py) && (Py<=max(y3
                        ,y4)+0.01)) % Verifica se o ponto de
                        interseccao esta contido dentro dos limites
                        das paredes
43                      n=n+1;
44                      dist(n)=sqrt((R.x-Px)^2+(R.y-Py)^2); % Calcula
                             a distancia entre o robo e o ponto de
                            interseccao e armazena-a em um vetor
45                  end
46              end
47          end
48      end
49
50      Dist(j)=min(dist); % Obtem a distancia minima entre
            aqueles pontos de interseccao calculados para saber
            qual e o ponto de interseccao valido
51      px(j)=R.x+Dist(j)*cosd(R.th+thS);
52      py(j)=R.y+Dist(j)*sind(R.th+thS); % Obtem o ponto de
```

```
            interseccao valido para o sensor j
53
54          clearvars dist
55      end
56  end
```

APÊNDICE C – Implementação do Filtro de Kalman

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Algoritmo para implementacao do Filtro de Kalman %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load 'W' % Carrega a matriz de covariancia obtida anteriormente

% Inicializacao de alguns parametros
% Mapa 2
L=[2 0 10 2.1436
    10 2.1436 10 8.1436
    10 8.1436 8.1436 10
    8.1436 10 1 10
    1 10 1 6
    1 6 0 6
    0 6 0 2
    0 2 2 2
    2 2 2 0];

% Range de angulos do robo
ths=-90:9:90; % 21 feixes, um a cada 9

l=0.14; % Distancia entre a roda e o centro do robo em metros
kr=0.1; % Constante de deslizamento da roda direita
kl=0.1 % Constante de deslizamento da roda esquerda

% Inicia-se conexao com vrep
vrep=remApi('remoteApi');
vrep.simxFinish(-1);
robot=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);

if (robot~=-1)
    disp('Connected to remote API server')
else
    disp('Connection not succesful')
end

% Obtem o handle do robo e dos motores das rodas direita e
```

```
    esquerda
38 [errorCode,robot_handle]=vrep.simxGetObjectHandle(robot,'
    Pioneer_p3dx',vrep.simx_opmode_oneshot_wait);
39 [errorCode,left_motor_handle]=vrep.simxGetObjectHandle(robot,'
    Pioneer_p3dx_leftMotor',vrep.simx_opmode_oneshot_wait);
40 [errorCode,right_motor_handle]=vrep.simxGetObjectHandle(robot,'
    Pioneer_p3dx_rightMotor',vrep.simx_opmode_oneshot_wait);
41 % Primeira chamada das funcoes que obtem a posicao e as
    velocidades do robo para inicializacao das mesmas
42 [errorCode,Position]=vrep.simxGetObjectPosition(robot,
    robot_handle,-1,vrep.simx_opmode_streaming);
43 [errorCode,Orientation]=vrep.simxGetObjectOrientation(robot,
    robot_handle,-1,vrep.simx_opmode_streaming);
44 [erroCode,linearVelocity,angularVelocity]=vrep.
    simxGetObjectVelocity(robot,robot_handle,vrep.
    simx_opmode_streaming);
45
46 % Uma parada no codigo para que se execute as funcoes anteriores
     com sucesso
47 pause(2)
48
49 % Estabelece as velocidades das rodas direita e esquerda do robo
     e as envia para o mesmo
50 vl=0.4; % [dm/s]
51 vr=0.5; % [dm/s]
52 errorCode=vrep.simxSetJointTargetVelocity(robot,
    left_motor_handle,vl,vrep.simx_opmode_streaming);
53 errorCode=vrep.simxSetJointTargetVelocity(robot,
    right_motor_handle,vr,vrep.simx_opmode_streaming);
54
55 % Inicia o cronometro
56 t0=vrep.simxGetLastCmdTime(robot);
57
58 % Obtem-se a posicao inicial do robo
59 [errorCode,Position]=vrep.simxGetObjectPosition(robot,
    robot_handle,-1,vrep.simx_opmode_buffer);
60 [errorCode,Orientation]=vrep.simxGetObjectOrientation(robot,
    robot_handle,-1,vrep.simx_opmode_buffer);
61 r.th=Orientation(3);
```

```
62 r.x=Position(1);
63 r.y=Position(2);
64
65 % Estabelece o numero de iteracoes para coleta de dados com o
       robo em movimento
66 inter=1500;
67 for k=1:inter
68     % Calcula o dt
69     t=vrep.simxGetLastCmdTime(robot);
70     dt=0.001*(t-t0); % Tempo dado em milisegundos (converte para
           segundos)
71     t0=vrep.simxGetLastCmdTime(robot);
72
73     % Obtem diretamente as velocidades linear e angular do robo
74     [erroCode,linearVelocity,angularVelocity]=vrep.
           simxGetObjectVelocity(robot,robot_handle,vrep.
           simx_opmode_buffer);
75     v=sqrt(linearVelocity(1)^2+linearVelocity(2)^2); % [m/s]
76     w=angularVelocity(3); % [rad/s]
77
78     % Calcula ds (deslocamento linear) e dth (deslocamento
           angular)
79     ds=v*dt;
80     dth=w*dt;
81     dsr=ds+l*dth; % Deslocamento da roda direita
82     dsl=ds-l*dth; % Deslocamento da roda esquerda
83
84     % Calculo da covariancia do estado a Priori
85     Fp=[1  0  -ds*sin(r.th+dth/2)
86         0  1  ds*cos(r.th+dth/2)
87         0  0  1];
88     Q=[kr*abs(dsr)  0
89         0  kl*abs(dsl)];
90     Fdrl=[(1/2)*cos(r.th+dth/2)-(ds/(4*l))*sin(r.th+dth/2)  (1/2)*
           cos(r.th+dth/2)+(ds/(4*l))*sin(r.th+dth/2)
91               (1/2)*sin(r.th+dth/2)+(ds/(4*l))*cos(r.th+dth/2)  (1/2)*
                   sin(r.th+dth/2)-(ds/(4*l))*cos(r.th+dth/2)
92               1/(2*l)  -1/(2*l)];
93     if(k==1)
```

```
94          P_minus{k}=Fdrl*Q*Fdrl';
95      else
96          P_minus{k}=Fp*P_plus{k-1}*Fp'+Fdrl*Q*Fdrl';
97      end
98
99      % Calculo do estado a Priori
100     r.x=r.x+ds*cos(r.th+dth/2);
101     r.y=r.y+ds*sin(r.th+dth/2);
102     r.th=r.th+dth;
103     r.th=mod(r.th,2*pi);
104     if (r.th>pi)
105         r.th=r.th-2*pi;
106     end
107     x_minus{k}=[r.x
108                 r.y
109                 r.th]; % angulo em radianos
110
111     % Calculo do ganho de Kalman e estado a Posteriori
112     boolean=true;
113     while boolean
114         [errorCode,Distance_Value]=vrep.simxGetStringSignal(robot,
                'Distance_table',vrep.simx_opmode_oneshot_wait);
115         if(errorCode==vrep.simx_return_ok)
116             Distance=vrep.simxUnpackFloats(Distance_Value);
117         end
118         for i=1:length(Distance)
119             Distance(i)=Distance(i)+0.05*randn; % Acrescenta um
                    ruido Gaussiano de media zero e desvio padrao 0,05
120         end
121         if (length(Distance)==181)
122             for i=1:length(ths)
123                 D(i)=Distance((abs(ths(1)-ths(2))*(i-1))+1);
124             end
125             boolean=false;
126         end
127     end
128
129     d=SensorLaserVirtual(r,ths,L);
130
```

```
131     Innovation=D'-d'; % Compara os valores de distancias dadas
            pelo sensor real e o sensor virtual
132
133     if Innovation~=0 % Se os dois valores forem diferentes, o
            algoritmo corrige a posicao
134         for i=1:length(D)
135             x1=x_minus{k}(1)+d(i)*cosd(rad2deg(x_minus{k}(3))+ths(i
                    ));
136             y1=x_minus{k}(2)+d(i)*sind(rad2deg(x_minus{k}(3))+ths(i
                    ));
137
138             H(i,1)=-(x1-x_minus{k}(1))/sqrt((x1-x_minus{k}(1))^2+(
                    y1-x_minus{k}(2))^2);
139             H(i,2)=-(y1-x_minus{k}(2))/sqrt((x1-x_minus{k}(1))^2+(
                    y1-x_minus{k}(2))^2);
140             H(i,3)=0; % Obtem-se a matriz H
141         end
142         K{k}=P_minus{k}*H'*(H*P_minus{k}*H'+W)^-1; % Ganho de
                Kalman
143         P_plus{k}=P_minus{k}-K{k}*H*P_minus{k}; % Covariancia a
                Posteriori
144         x_plus{k}=x_minus{k}+K{k}*Innovation; % Estado a
                Posteriori
145
146         clearvars H
147     else
148         P_plus{k}=P_minus{k};
149         x_plus{k}=x_minus{k};
150     end
151
152     % Obtem os valores reais de posicao dados pelo simulador
153     [errorCode,Position]=vrep.simxGetObjectPosition(robot,
            robot_handle,-1,vrep.simx_opmode_buffer);
154     [errorCode,Orientation]=vrep.simxGetObjectOrientation(robot,
            robot_handle,-1,vrep.simx_opmode_buffer);
155     % Posicao real do robo
156     x_fun{k}=[Position(1)
157               Position(2)
158               Orientation(3)]; % angulo em radianos
```

```
159 end
160
161 errorCode=vrep.simxSetJointTargetVelocity(robot,
        left_motor_handle,0,vrep.simx_opmode_streaming);
162 errorCode=vrep.simxSetJointTargetVelocity(robot,
        right_motor_handle,0,vrep.simx_opmode_streaming);
```