UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MARCO ANTONIO DALCIN TOCCHETTO

ESTIMADOR DE ESTADOS PARA ROBÔ DIFERENCIAL

Porto Alegre
2017

MARCO ANTONIO DALCIN TOCCHETTO

ESTIMADOR DE ESTADOS PARA ROBÔ DIFERENCIAL

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.
Área de concentração: Controle e Automação

ORIENTADOR: Prof. Dr. Alexandre Sanfelice Bazanella

Porto Alegre
2017

MARCO ANTONIO DALCIN TOCCHETTO

ESTIMADOR DE ESTADOS PARA ROBÔ DIFERENCIAL

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. Alexandre Sanfelice Bazanella,
Doutor pela Universidade Federal de Santa Catarina

Banca Examinadora:

Prof. Dr. Marco Henrique Terra, USP
Doutor pela Universidade de São Paulo – São Paulo, Brasil

Prof. Dr. Valner João Brusamarello, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Prof. Dr. Diego Eckhard , UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Coordenador do PPGEE: _____
Prof. Dr. Valner João Brusamarello

Porto Alegre, 28 de Março de 2017.

AGRADECIMENTOS

Aos meus pais, Vânia e Homero, que entendem meu tempo, e sabem que tudo tem o seu tempo. Seu suporte e carinho me deram forças para seguir firme nesta caminhada.

À minha namorada, Bruna, pela sua simples presença, pela sua paciência, pela sua compreensão, pelo seu amor e pelo seu carinho.

Ao meu irmão, Rodrigo, pelo companheirismo e apoio.

Aos meus amigos, pela presença e pela ajuda de forma direta ou indireta.

Ao meu orientador, Prof. Dr. Bazanella, pela oportunidade dada, seus ensinamentos, paciência, orientação e excelente pessoa e profissional que sempre demonstrou ser.

Ao CNPq pelo seu apoio financeiro.

RESUMO

Nesta dissertação é apresentada a comparação do desempenho de três estimadores - o Filtro de Kalman Estendido, o Filtro de Kalman *Unscented* e o Filtro de Partículas - aplicados para estimar a postura de um robô diferencial. Uma câmera foi fixa no teto para cobrir todo o campo operacional do robô durante os experimentos, a fim de extrair o mapa e gerar o *ground truth*. Isso permitiu realizar uma análise do erro de forma precisa a cada instante de tempo.

O desempenho de cada um dos estimadores foi avaliado sistematicamente e numericamente para duas trajetórias. Os resultados desse primeiro experimento demonstram que os filtros proporcionam grandes melhorias em relação à odometria e que o modelo dos sensores é crítico para obter esse desempenho. O Filtro de Partículas mostrou um desempenho melhor em relação aos demais nos dois percursos. No entanto, seu elevado custo computacional dificulta sua implementação em uma aplicação de tempo real. O Filtro de Kalman *Unscented*, por sua vez, mostrou um desempenho semelhante ao Filtro de Kalman Estendido durante a primeira trajetória. Porém, na segunda trajetória, a qual possui uma quantidade maior de curvas, o Filtro de Kalman *Unscented* mostrou uma melhora significativa em relação ao Filtro de Kalman Estendido.

Foi realizado um segundo experimento, em que o robô planeja e executa duas trajetórias. Os resultados obtidos mostraram que o robô consegue chegar a um determinado local com uma precisão da mesma ordem de grandeza do que a obtida durante a estimação de estados do robô.

Palavras-chave: Estimação de estados, Robô diferencial, Filtro de Kalman Estendido, Filtro de Kalman *Unscented*, Filtro de Partículas.

ABSTRACT

In this dissertation, the performance of three nonlinear-model based estimators - the Extended Kalman Filter, the Unscented Kalman Filter and the Particle Filter - applied to pose estimation of a differential drive robot is compared. A camera was placed above the operating field of the robot to record the experiments in order to extract the map and generate the ground truth so the evaluation of the error can be done at each time step with high accuracy.

The performance of each estimator is assessed systematically and numerically for two robot trajectories. The first experimental results showed that all estimators provide large improvements with respect to odometry and that the sensor modeling is critical for their performance. The particle filter showed a better performance than the others on both experiments, however, its high computational cost makes it difficult to implement in a real-time application. The Unscented Kalman Filter showed a similar performance to the Extended Kalman Filter during the first trajectory. However, during the second one (a curvier path) the Unscented Kalman Filter showed a significant improvement over the Extended Kalman Filter.

A second experiment was carried out where the robot plans and executes a trajectory. The results showed the robot can reach a predefined location with an accuracy of the same order of magnitude as the obtained during the robot pose estimation.

Keywords: State estimation, Extended Kalman Filter, Unscented Kalman FIlter, Particle Filter, Differential drive robot.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

f.d.p	função densidade de probabilidade
f.m.p	função massa de probabilidade
FD	função de distribuição
FK	Filtro de Kalman
FKE	Filtro de Kalman Estendido
FKU	Filtro de Kalman Unscented
FP	Filtro de Partículas
fps	*frames per second*
GT	Ground-truth
IV	Infravermelho
SIR	*Sampling Importance Resampling*
SIS	*Sequence Importance Sampling*
v.a.	Variável Aleatória

SUMÁRIO

1 INTRODUÇÃO

1.1 Definição do Problema

Os robôs são vistos normalmente como máquinas com grandes capacidade de locomoção, visão, manipulação, interação e percepção devido a histórias de ficção científica presente em filmes, seriados e livros. Embora a robótica esteja em constante evolução, algumas habilidades humanas ainda são inigualáveis quando comparadas às dos robôs atuais.

Com o objetivo de realizar tarefas de forma precisa e sem a necessidade da manipulação ou observação humana, a capacidade de autonomia em robôs é uma das funções mais almejadas pela comunidade científica. Um robô é considerado autônomo quando é capaz de realizar uma tarefa de forma independente, ou seja, sem qualquer necessidade de interferência humana durante o processo (BEKEY, 2005).

Existem diversos exemplos, como quadrirrotores, que transportam objetos ou realizam alguma tarefa envolvendo o processamento de imagens; robôs exploradores, que exploram o terreno em outros planetas no espaço, que percorrem locais de acesso crítico como ambientes radioativos ou que simplesmente exploram o terreno da sua casa e aspiram o pó do chão. Em todos exemplos mencionados o robô requer autonomia e mobilidade. Para que o robô consiga realizar suas tarefas ele precisa mover-se em um ambiente e isto é conhecido como navegação.

A Navegação é a tarefa de um robô autônomo para mover-se com segurança de um determinado local para outro. O problema geral da navegação pode ser formulado em três perguntas (LEONARD; DURRANT-WHYTE, 1991): onde estou? O robô tem que saber onde está, a fim de tomar decisões; onde eu vou? Para cumprir alguma tarefa o robô tem que saber para onde está indo; como eu chego lá? Uma vez que o robô sabe onde está e onde tem que ir, tem que decidir como chegar lá.

Pode-se dizer que robôs têm os mesmos problemas que os seres humanos ao se locomoverem. Ao caminhar, por exemplo, existe uma realimentação constante da visão para corrigir a forma de andar e desviar de obstáculos. Os outros sentidos podem ser utilizados de forma complementar para captar informações que não estão ou não são percebidas no campo de visão.

Se os movimentos não forem supervisionados pela visão, a locomoção até um determinado ponto pode ficar comprometida devido à imperfeição no movimento. Isso pode ser facilmente observado ao tentar andar em uma linha reta de olhos fechados. De uma forma análoga os robôs precisam extrair as informações providas de sensores para corrigir os movimentos e ter uma ideia melhor de onde estão.

Enquanto um robô terrestre está em movimento, normalmente são utilizados *encoders*, que contam o número de revoluções que as rodas fazem durante um instante de tempo.

Essas leituras podem ser usadas para estimar o deslocamento feito pelo robô. Porém, essas medidas estão sujeitas a erros e devido à derrapagem, diferentes tipos de piso e outros fatores podem causar discrepâncias ainda maiores nas leituras do sensor. Isso faz com que o erro se acumule conforme o robô se locomove.

Utilizando somente essas informações não é possível saber o quão errado está o deslocamento (da mesma forma aconteceria ao caminhar de olhos cerrados), e nem a posição inicial do robô. Para manter o erro de posição dentro de certos limites, o robô pode usar sensores que visualmente sentem o ambiente e/ou forneçam mais informações sobre o seu movimento. Estas informações podem ser utilizadas para diminuir a incerteza de sua localização. Para isso diversos tipos de sensores podem ser utilizados: câmeras, magnetômetro, IMU (do inglês, *Inertial Measurement Unit*), GPS (do inglês, *Global Positioning System*), sensores de distância (infravermelho, sonar, laser *rangefinder*). Essas informações que são adquiridas podem ser utilizada para corrigir o erro que foi introduzido pelos *encoders*, por exemplo. Porém, assim como os *encoders*, os sensores usados para diminuir o erro na posição também estão sujeitos a erros e estes também devem ser levados em consideração para obter uma boa estimativa de sua localização.

A combinação de múltiplas informações providas por sensores é conhecida como fusão de sensores. Ela consiste em combinar dados de diferentes sensores para obter uma estimativa melhor do que utilizando-os de forma separada. A criação de novos sensores e/ou filtros para compreender informações providas pelo ambiente e inovação de algoritmos de estimação de movimento a partir da integração de vários sensores vêm sendo desenvolvida com o objetivo de atingir a autonomia dos robôs.

Neste trabalho é abordado o problema de localização e o planejamento de trajetórias para um robô diferencial autônomo a partir de dados provenientes dos *encoders*, magnetômetro e cinco sensores de infravermelho. Este trabalho envolve a estimação do vetor de estados do robô – posição no eixo x, y e sua orientação – em um ambiente fechado. A maioria dos trabalhos presente na literatura compara o desempenho dos estimadores utilizando apenas a posição final e apenas poucos trabalhos realizam uma análise do erro durante toda a trajetória.

1.2 Objetivos

O objetivo principal deste trabalho é realizar um estudo comparativo entre três filtros Bayesianos utilizados para localizar um robô diferencial em um mapa pré-definido. Os trabalhos presente na literatura não especificam como é feita a análise do erro de cada estimador e muitas vezes a comparação é feita somente em relação à posição final do robô. Neste trabalho o *ground truth* (GT) é extraído com o objetivo de realizar uma comparação de forma precisa entre os filtros. Para isso, fixou-se uma câmera no teto a fim de gravar os experimentos. O GT e o mapa foram extraídos das gravações utilizando técnicas de processamento de imagens, permitindo, dessa forma, mensurar o erro entre o a posição real do robô e a posição estimada por cada um dos filtros a cada instante de tempo. Os objetivos específicos desse trabalho são:

1. Apresentar o modelo do robô e desenvolver modelos dos sensor de infravermelho utilizados na localização do robô em um ambiente fechado.

2. Comparar as estimativas da localização obtida por meio de três filtros Bayesianos: o Filtro de Kalman Estendido, Filtro de Kalman *Unscented* e o Filtro de Partículas.

3. Planejar e executar uma trajetória até um determinado local pré-definido no mapa.

1.3 Apresentação da Dissertação

No Capítulo 2 é feita uma revisão bibliográfica sobre a localização de robôs móveis. O capítulo 3 contém uma introdução aos filtros Bayesianos e uma revisão de conceitos probabilísticos que serão necessários para que se tenha uma boa compreensão dos filtros que serão apresentados no capítulo 4. No capítulo 5 é apresentado o robô utilizado neste trabalho, o seu modelo e o modelo de medição necessários para fazer a implementação dos filtros. O capítulo 6 mostra como é feita a extração do mapa, a localização do robô, o planejamento de rota e um algoritmo para seguir uma trajetória. O Capítulo 7 contém os experimentos práticos realizados. Por fim, o Capítulo 8 apresenta as conclusões deste trabalho, bem como as propostas de trabalhos futuros a partir do que já foi elaborado.

2 REVISÃO BIBLIOGRÁFICA

O presente capítulo realiza uma revisão bibliográfica sobre a localização de robôs em ambientes fechados.

A localização de robôs é um problema desafiador, o qual consiste em encontrar a postura – posição x,y e a sua orientação (θ). Determinar a sua localização em relação ao ambiente é fundamental para que o robô possa realizar alguma tarefa de navegação de forma autônoma.

A estimação dos estados do robô (x, y, θ) é feita utilizando os dados disponíveis de medidas junto com o conhecimento da dinâmica do sistema e do instrumento de medida. No entanto, tanto os dados quanto os modelos contêm incertezas: a medida dos sensores contém ruído e os modelos limitações. Esse tipo de problema é abordado na literatura através de métodos probabilísticos, em que ao invés de manter uma hipótese de onde o robô se encontra, é utilizada uma distribuição de probabilidade sobre o espaço de todas as hipóteses. Este tipo de representação leva em consideração as incertezas presentes no movimento e nos sensores do robô.

Independente do sensor utilizado e por mais perfeito que o modelo seja sempre haverá uma incerteza. Entretanto, essa incerteza pode ser reduzida, utilizando a fusão de sensores, que é a combinação de dados obtidos por meio de sensores, de modo que a informação resultante é melhor do que se essas informações fossem utilizadas de forma individual.

Um dos algoritmos utilizados para fazer a estimação de estados é o filtro de Kalman (FK). O primeiro artigo sobre o FK foi publicado em 1960 por Rudolf Emil Kálmán (KALMAN, 1960). Esse artigo descreve um processo recursivo para solucionar problemas lineares relacionados à filtragem de dados discretos.

A maioria das aplicações, incluindo a robótica móvel terrestre e aérea, é não linear e deve ser abordada por técnicas não lineares ou aproximações para manter o desempenho e a estabilidade do sistema modelado. Existem extensões do FK na literatura para sistemas não lineares e os dois algoritmos mais usados para realizar a fusão dos sensores em aplicações robóticas são: o Filtro de Kalman Estendido (FKE) e o Filtro de Kalman *Unscented* (FKU) (CHEN, 2003). Outro filtro capaz de lidar com problemas não lineares é o Filtro de Partículas (FP), o qual fornece uma estimativa ótima para modelos não lineares e não-Gaussianos quando o número de partículas utilizado é suficientemente grande (GUSTAFSSON et al., 2002).

O método mais utilizado para encontrar a postura do robô é baseada na odometria, o qual as velocidades, linear e angular, do robô são integradas ao longo do tempo para determinar a sua posição. No entanto, isso gera um acúmulo de erros proporcional à distância navegada. Ou seja, utilizando somente a odometria não é suficiente para estimar a posição do robô de forma precisa. Logo, faz-se necessário utilizar sensor(es) para corrigir

este erro proveniente da odometria.

O FKE foi uma das primeiras técnicas utilizadas para problemas não lineares e tem sido aplicado à localização de robôs há bastante tempo. Um estudo feito por (ROUMELIOTIS, 1997) utiliza o FKE para integrar os dados dos sensores (*encoders* e giroscópio) juntamente com uma equação dinâmica que descreve o movimento do robô. Os resultados da simulação utilizando o FKE mostraram uma performance 40% melhor em relação à estimativa feita utilizando somente a odometria. Um estudo semelhante feito por (ALESSANDRI et al., 1997) mostrou que ao ao usar um modelo modificado com um giroscópio é possível eliminar os erros associados ao derrapamento das rodas por meio do FKE.

O trabalho de (ROUMELIOTIS; SUKHATME; BEKEY, 1999) dividiu o problema de navegação em um processo de estimação em dois estágios; o primeiro com a estimativa de orientação, o segundo pela estimativa de posição e, em seguida, utilizou o FKE para fazer a fusão dos resultados. O trabalho de (SASIADEK J.Z.; WANG, 1999) fez a fusão de um Sistema de Navegação Inercial (INS) e um GPS usando o FKE com regras de lógica Fuzzy.

Grande parte dos problemas de localização são resolvidos com um mapa dado a priori. Os trabalhos de (DRUMHELLER 1987 e CROWLEY 1989) introduzem uma abordagem de localização baseada em mapas usando sonares. O trabalho feito por (PINZ, 2001) e (IVANJKO; PETROVIć; BREZAK, 2009) fazem uma comparação de diferentes abordagens para construir um mapa de *grid* de ocupação de ambientes utilizando sonares.

Uma comparação em uma implementação prática do desempenho entre o FKE e o FKU na estimação da postura de um robô utilizando sonares é feita por (D'ALFONSO et al., 2015). Os resultados obtidos pelo autor não mostraram uma diferença significativa entre os dois filtros utilizados. No entanto, no trabalho de (HOUSHANGI; AZIZI, 2005), onde é utilizada a fusão da odometria com um giroscópio de fibra óptica existe uma diferença significativa entre os resultados dos filtros. Em um de seus experimentos o erro encontrado na posição x do robô foi semelhante, no entanto o erro na posição y foi quase três vezes maior utilizando o FKE do que utilizando o FKU.

O trabalho de (BRŠČIĆ; HASHIMOTO, 2008) utiliza um laser (*rangefinder*) e faz uma comparação entre o FK utilizando intersecção de covariância e o FP. Os resultados obtidos são semelhantes e devido ao custo computacional envolvido e à facilidade de implementação, o autor sugere a utilização FK apresentado. O trabalho de (MONTEMERLO; THRUN; WHITTAKER, 2002) faz uma abordagem mais complexa onde é feita a estimação da posição de um robô e de pessoas próximas em um ambiente previamente mapeado usando um FP condicional.

Existem diversos outros trabalhos presentes na literatura que fazem o uso de uma câmera para localizar o robô. Os trabalhos de odometria visual utilizam diferentes técnicas de processamento e correlação de imagens. O trabalho de (FANG; YANG; YANG, 2007) faz a fusão da odometria dos encoders com a odometria visual por meio do FKU. O seu trabalho consiste em localizar um robô autônomo utilizando uma câmera direcionada para baixo e um mapa de texturas de imagens. As informações dos encoders são corrigidas por meio de keypoints identificados com o algoritmo Harris Corner Detector e correlacionados através de técnicas de otimização

O trabalho de (BAK et al., 2012) utiliza o FKE para fazer a fusão da odometria visual juntamente com Sistemas de navegação inercial de baixo custo para estimar a posição do robô. Já o trabalho de (HOANG et al., 2013) utiliza o FKE para a localização em um ambiente aberto onde a predição é proveniente de um laser *rangefinder* e a correção é feita utilizando a odometria visual por meio do algoritmo SIFT.

Trabalhos mais recentes se concentraram no problema de navegação mais geral conhecido como Localização Simultânea e Mapeamento (do ingês, SLAM). Como o próprio nome já sugere, o SLAM consiste em fazer o mapeamento e a localização ao mesmo tempo em tempo real, dessa forma o robô pode explorar ambientes desconhecidos e realizar suas tarefas de forma autônoma. Tanto este problema como o problema de localização ainda são problemas em aberto e ainda existe espaço para pesquisa nessa área.

Uma das primeiras abordagens feitas para o SLAM foi feita por (SMITH; SELF; CHEESEMAN, 1988), em que o FKE é usado para realizar a estimação conjunta da postura do robô e dos parâmetros referente aos elementos do mapa. Devido à maior capacidade de processsamento e às limitações do FKE relacionada ao processo de linearização que ocasionalmente geravam a divergência do filtro, começaram a ser utilizadas outras abordagens que se preocupavam com a consistência na criação do mapa. O trabalho de (ANJUM et al., 2010) utiliza o FKU e (MONTEMERLO; THRUN; WHITTAKER, 2002) propõe um algoritmo utilizando o FP para melhorar essa estimativa.

Existem outros trabalhos relacionados como o de (BEEVERS; HUANG, 2006) que faz o SLAM usando o FP através de dados esparsos de ultrassom. Os dados esparsos dificultam a extração de informações do ambiente, então foi feita uma modificação no algoritmo para integrar as informações de uma sequência de medições. Os experimentos mostraram que é possível o robô fazer o SLAM, mas de uma forma limitada. O trabalho de (MOURIKIS; ROUMELIOTIS, 2006) faz uma análise do desempenho do uso do C-SLAM ou SLAM cooperativo, onde vários robôs se ajudam mutuamente para realizar o SLAM, utilizando o FKE como estimador de estados.

Pode-se perceber que o problema de localização de um robô vem sendo desenvolvido há um bom tempo e dependendo dos sensores utilizados e do *hardware* disponível ele pode ser considerado um problema resolvido. Porém, dificilmente se encontra na literatura uma análise do erro na estimação da postura do robô e muitas vezes o método pelo qual a posição real do robô foi obtida não é informada ou carece de detalhes. Portanto, não se sabe qual é a precisão da estimativa apresentada.

Neste trabalho é feita uma implementação dos filtros mais utilizados para localizar um robô em um ambiente fechado e ao contrário dos trabalhos presentes na literatura, propõe-se uma forma de extrair o *ground truth* de cada um dos experimentos. Isso permite saber a posição real do robô durante todo o experimento e torna possível quantificar o erro de cada método a cada instante de tempo.

3 INTRODUÇÃO AOS FILTROS BAYESIANOS

Neste capítulo é introduzido o filtro de Bayes, por meio de um exemplo intuitivo. Também são revistos conceitos probabilísticos como a variável aleatória, a média, a variância e as distribuições de probabilidade (com uma ênfase na distribuição Gaussiana) que serão necessários para o desenvolvimento do trabalho.

O mundo está repleto de dados e eventos que despertam interesse, em que grande parte dessas informações pode ser adquirida por meio de sensores. No entanto, as informações fornecidas pelos sensores contêm ruídos.

Como se deve proceder em casos em que seja de interesse utilizar a medida de sensores para rastrear os movimentos de um robô e/ou criar um piloto automático para o mesmo? Como a natureza dos ruídos presentes é aleatória é necessário utilizar uma abordagem probabilística para resolver esse tipo de problema. Esses problemas podem ser resolvidos por meio de filtros Bayesianos. De forma sucinta, pode-se dizer que a probabilidade Bayesiana determina o que é provável que seja verdadeiro com base em informações passadas.

Supondo-se que um robô esteja aspirando o pó do chão de uma sala e deseja-se encontrar qual é a sua orientação. Os valores inteiros possíveis da sua orientação estão em um intervalo entre $0°$ e $359°$. Ou seja, caso não haja nenhuma informação sobre o robô, a probabilidade de acertar a sua orientação é de 1/360. Entretanto, caso seja informado que o robô está andando em linha reta e há um segundo atrás a sua orientação era de $45°$, pode-se inferir que a sua orientação não deve ter variado muito. Logo, pode-se dizer que ela será provavelmente um valor próximo dos $45°$.

Nesse caso está sendo utilizada uma informação passada para inferir uma informação sobre o presente ou o futuro. Neste exemplo a informação a priori ajuda a ter uma estimativa melhor, mas também está sujeita a ruídos. Poderia ocorrer de algum sensor presente no robô reportar uma mudança brusca que de fato não aconteceu. Como também poderia acontecer do robô derrapar ou de colidir com um obstáculo e alterar significativamente a sua orientação. O conhecimento é incerto e a confiança nele é alterada conforme a evidência. Os filtros Bayesianos misturam o conhecimento limitado e com ruído de como o sistema e os sensores se comportam para produzir a melhor estimativa possível, onde não são descartadas informações. Quanto mais informações houver, melhor será a estimativa.

A melhor forma de começar a criar uma intuição de como funciona a abordagem Bayesiana é por meio de um exemplo. Um exemplo de (THRUN; BURGARD; FOX, 2005) sobre a localização de um robô é apresentado de uma maneira reformulada e mais detalhada. Supondo que um robô, que possui movimentos limitados, pode se mover somente uma unidade para frente, uma unidade para trás ou ficar parado. Assume-se também que ele porta um sensor que reporta esse movimento e outro sensor que permite determinar se ele está na frente de uma porta (1) ou em frente a uma parede (0), ou seja, detecta a

presença de uma porta. No primeiro momento os sensores são considerados ideais, ou seja, não possuem nenhum erro relativo à informação fornecida (movimento e detecção de porta/parede).

O objetivo é rastrear um robô em um corredor, como ilustrado na Figura 1. Este corredor contém 10 posições e em cada uma dessas posições pode haver uma porta ou uma parede. Assume-se, também, que este corredor é circular, ou seja, quando o robô se encontra na posição 10 e anda para frente ele vai estar na posição 1 e vice-versa.

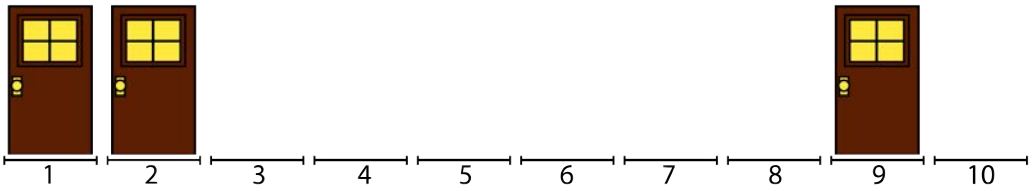


Figura 1: Ilustração do corredor utilizado no exemplo.

Inicialmente não há nenhuma evidência de onde o robô se encontra (sem obter a leitura do sensor), então a probabilidade do robô se encontrar em qualquer uma das posições é de $\frac{1}{10}$. Na estatística Bayesiana isso é conhecido como distribuição de probabilidade a priori, ou seja, é a probabilidade que se tem antes de incorporar medidas ou outras informações. A distribuição de probabilidade é uma coleção de todas as probabilidades possíveis para um evento e a sua soma é sempre igual a 1.

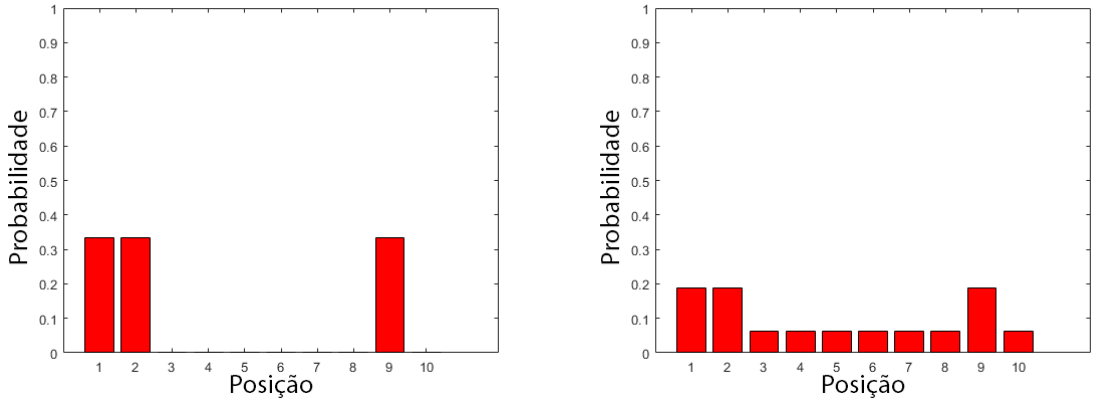
Coloca-se o robô para andar para frente por um determinado tempo tempo. Após alguns segundos o seu movimento é interrompido e, em seguida, é feita a leitura do sensor; este sensor retorna que o robô está na frente de uma porta. A função de verossimilhança calcula o quão provável é cada posição dada a medida.

Nesse momento, sabe-se que o robô pode estar na posição 1,2 ou 9, mas não se sabe em qual delas. Então a probabilidade dele estar na frente de qualquer uma dessas portas é $\frac{1}{3}$. Existe uma convicção (Bayesiana) de que há $33,3\%$ de chance do robô estar na posição 1, $33,3\%$ na posição 2 e $33,3\%$ de chance de estar na posição 9, ilustrado na Figura 2(a). Este resultado é chamado de distribuição a posteriori, que é a distribuição de probabilidade após a incorporação de informação de medição. A posteriori é dada pela equação (1), onde α é um fator de normalização utilizado para torná-la uma distribuição.

$$\text{posteriori} = \frac{\text{função de verossimilhança} \times \text{priori}}{\alpha} \tag{1}$$

Isso gera uma grande melhoria, em que foram descartadas 7 posições e a convicção nas posições restantes aumentou de 10% para 33%. Conforme o conhecimento melhora, as probabilidades se aproximarão de 100%. Neste exemplo é possível detectar onde o robô se localiza com 100% de certeza após dois movimentos: caso ele esteja na posição 2 ou 9, ou com um movimento caso ele esteja na primeira porta.

No entanto, este exemplo ocorre em um mundo perfeito onde os sensores não possuem ruído e na prática isso não existe. Logo, quando o sensor detectou uma porta, por exemplo, não se pode atribuir uma probabilidade de $0,33\%$ para as posições que contêm portas (posição 1, 2 e 9) e 0 para as posições que não contêm uma porta (posição 3, 4, 5, 6, 7, 8 e 10). Então, supõe-se, por exemplo, que o sensor tem 75% de chance de estar correto e 25% de estar errado. Agora a probabilidade do robô estar em frente a uma porta é de $18,75\%$ e a de estar em frente a uma parede é de $6,25\%$, como é mostrado na Figura 2b e calculado utilizando (1). Estes valores podem ser obtidos atribuindo as probabilidades



(a) (b)

Figura 2: Distribuição de probabilidade ao detectar uma porta: (a) sem ruído no sensor (b) com ruído no sensor.

de 0,75 e 0,25 às posições. Porém, ao somar essas probabilidades o valor é igual a quatro ($\frac{3*0,75}{\alpha} + \frac{7*0,25}{\alpha}$), o que não representa uma distribuição de probabilidade. Portanto, é necessário normalizar os resultados, ou seja, dividir cada um desses valores por quatro, que é o papel do α na equação (1).

Anteriormente, para conseguir encontrar a posição do robô foi incorporado um movimento, no entanto na realidade este movimento também está sujeito a ruídos. Por exemplo, o sensor pode ter reportado que o robô andou uma posição para frente, mas ele pode ter andado duas para frente ou nenhuma. Então é necessário adicionar um ruído ao movimento. Supõe-se que o sensor tem 70% de chance de estar correto, 15% de errar uma posição para a direita e 15% de errar uma posição para a esquerda.

Observe que se somente a informação referente ao movimento com a presença de ruído for utilizada, com o passar do tempo a incerteza que se tem sobre a posição do robô crescerá e se espalhará por todas as possíveis posições. A Figura 3 ilustra alguns passos do que ocorre com a distribuição de probabilidade do robô que se locomove uma posição para a direita 100 vezes. A Figura 3a mostra que a posição inicial do robô é conhecida, ou seja, sabe-se a posição do robô com 100% de certeza. O robô começa a se movimentar e, como existe uma incerteza na sua locomoção, começa a surgir uma incerteza na posição do robô. As Figura 3b,3c e 3d ilustram a probabilidade do robô estar em cada uma das posições após incorporar 10, 50 e 100 movimentos, respectivamente.

O problema de perder informações durante a estimativa a priori e consequentemente aumentar a incerteza pode ser resolvido utilizando a correção, onde incorpora-se a medida na estimativa (estimativa do estado a posteriori), fazendo com que a incerteza no estado diminua.

Um último exemplo é apresentado, onde o robô possui ruído no movimento e no sensor que detecta portas. Neste caso ele desconhece a sua posição inicial. Coloca-se o robô na primeira posição e faz com que ele se locomova uma posição para a direita a cada instante de tempo. As estimativas a priori e a posteriori são apresentadas durante algumas etapas na Figura 4. Inicialmente, como não se sabe em qual posição o robô se encontra, atribui-se uma probabilidade igual para todas as dez posições (Figura 4a). Em seguida, é feita feita uma leitura do sensor para saber se ele se encontra em frente a uma porta ou não. A Figura 4b mostra a estimativa a posteriori. O robô continua na primeira posição e agora realiza um movimento para a segunda posição. A Figura 4c mostra a

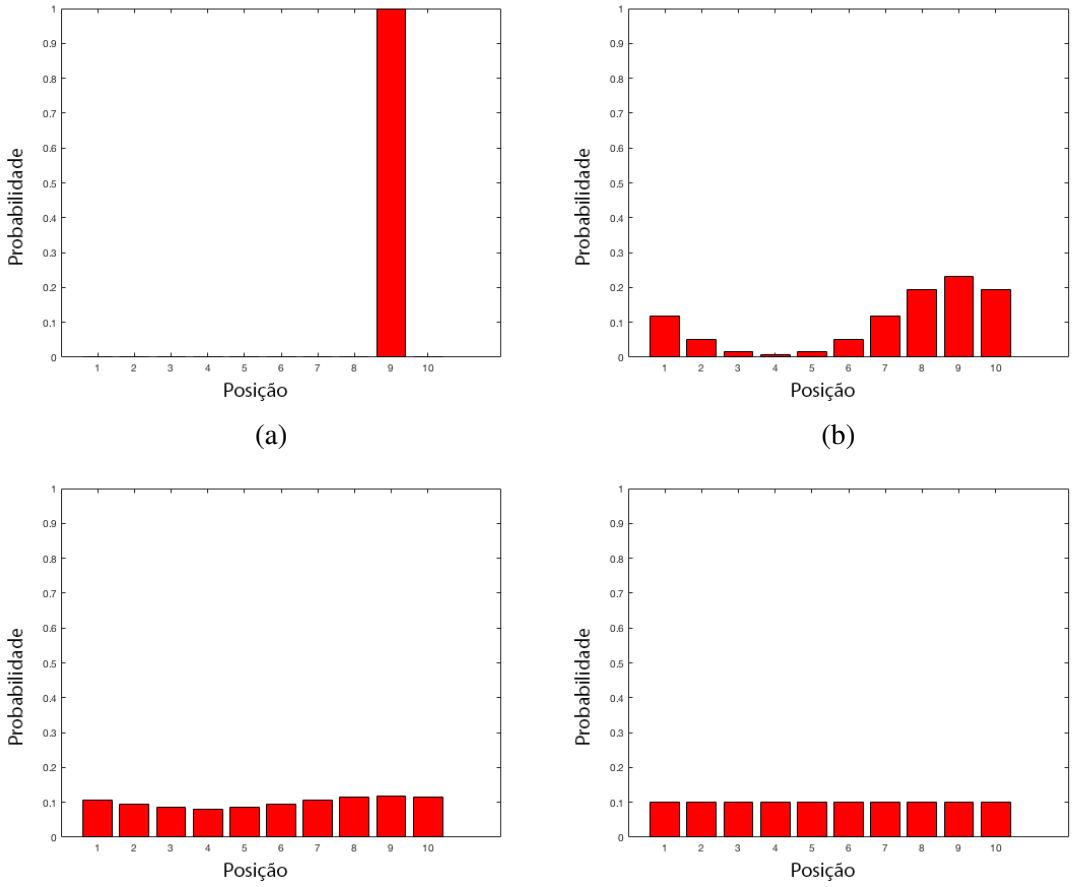


Figura 3: Distribuição de probabilidade do robô utilizando somente a estimativa a priori: (a) posição inicial conhecida (b),(c) e (d) após 10,50 e 100 movimentos.

estimativa a priori, que ocorre após incorporar o movimento. Finalmente, as Figuras 4d, 4e e 4f mostram as estimativas a posteriori (leitura do sensor na segunda posição), a priori (locomoção de uma posição para frente) e a posteriori (leitura do sensor na terceira posição), respectivamente.

Pode-se observar na Figura 4d que existe uma probabilidade maior do robô se encontrar na segunda posição. Isto corresponde ao caso (correto) do robô ter começado na primeira posição. Inicialmente o sensor detectou a presença de uma porta e após ele se descolar 1 unidade para a direita o sensor detectou outra porta. Nenhuma outra posição torna este conjunto de observações possível, pois existe apenas uma situação em que existe uma porta adjacente de outra.

Após a segunda leitura do sensor (Figura 4d) a probabilidade do robô estar em frente a porta é de $29,8\%$, aumentando cerca de 60% em relação a primeira leitura do sensor, que era de $18,75\%$. Em seguida, após o movimento (Figura 4d) observa-se que diminui essa probabilidade. Porém, novamente após a leitura do sensor pela terceira vez a probabilidade dele se encontrar na quarta posição é de 31%. Essa probabilidade pode não parecer muito grande, no entanto ela é 55% maior do que em relação a de estar na posição 5, que é a segunda maior probabilidade. Dessa forma mesmo utilizando sensores com ruídos é possível, através de uma abordagem probabilística, localizar o robô.

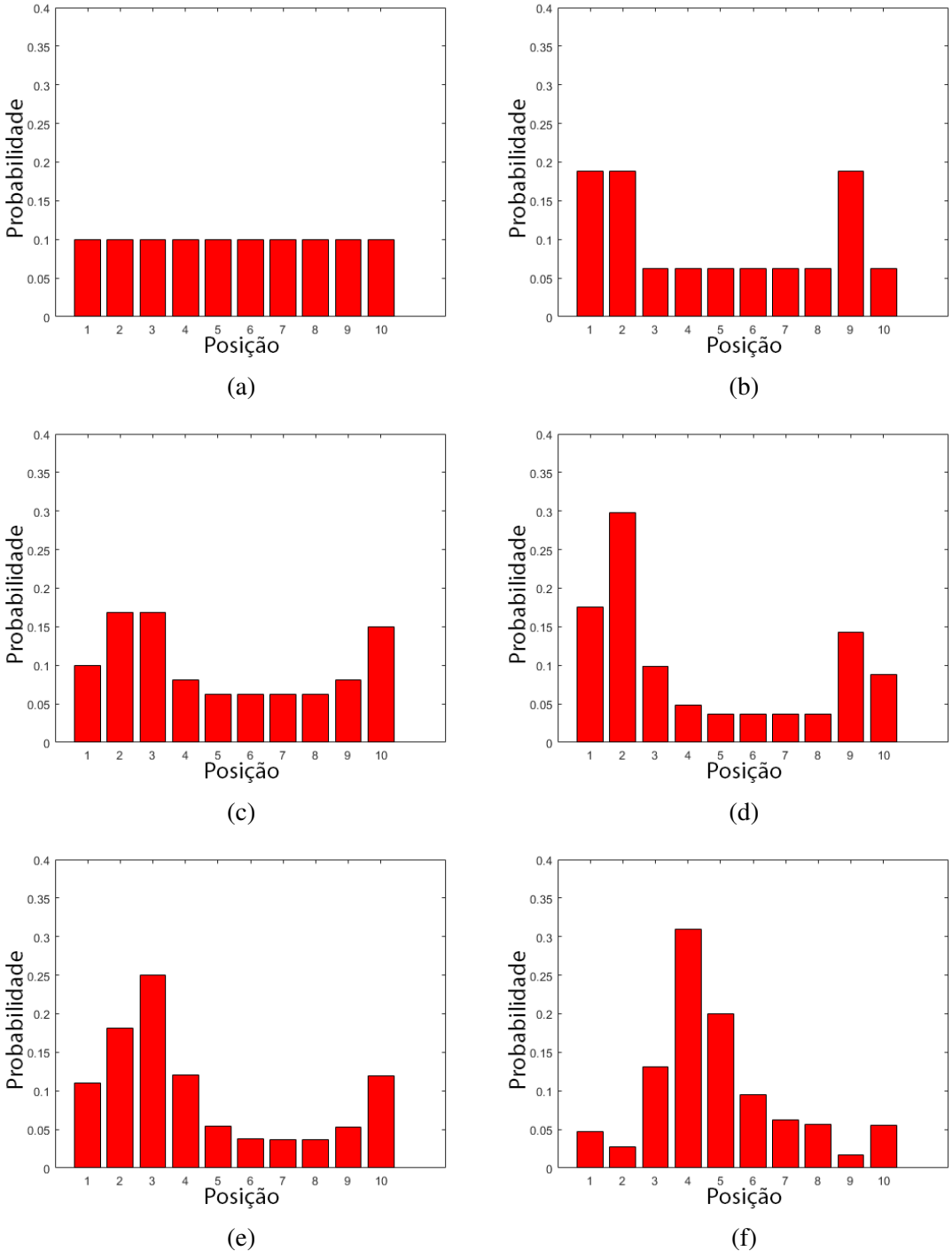


Figura 4: Distribuição de probabilidade: (a) inicial (sem prévio conhecimento da posição do robô) (b) distribuição a posteriori (sensor detectou a presença de uma porta) (c) distribuição a priori (robô executou um movimento para a direita) (d) distribuição a posteriori (sensor detectou a presença de outra porta) (e) distribuição a priori após realizar um movimento para a direita (f) distribuição a posterior (sensor não detectou a presença de uma porta)

3.1 Teorema de Bayes e Teorema da Probabilidade Total

O teorema de Bayes consiste em calcular a probabilidade de um evento dada uma informação. Isso é exatamente o que foi feito no exemplo anterior, mais especificamente o

que foi feito utilizando a equação (1). O teorema de Bayes é dado em termos de probabilidades, conforme a equação (2) (DURRANT-WHYTE; HENDERSON, 2008, Section C).

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{2}$$

Com o objetivo de formalizar de uma forma matemática a seguinte seção apresenta os teoremas utilizados no exemplo da seção anterior. O termo $P(A)$ é a probabilidade do evento A ocorrer e o termo $P(A \mid B)$ é a probabilidade condicional, isto é, a probabilidade de A dado que o evento B ocorreu. A probabilidade condicional pode ser ilustrada pelo seguinte exemplo: deseja-se retirar duas cartas do mesmo naipe em um baralho de cartas. Das 52 cartas, existem 13 cartas em cada naipe. Supondo que a primeira carta do baralho foi retirada e ela é um dois de copas, agora somente existem 12 copas restantes no baralho e o como uma carta foi removida só restam 51 cartas. Assim, a probabilidade condicional P (segunda carta de copas| primeira carta de copas) $= \frac{12}{51}$.

No teorema de Bayes B é chamado de evidência, $P(A)$ é a probabilidade a priori, $P(B \mid A)$ é a função de verossimilhança e $P(A \mid B)$ é a probabilidade a posteriori. No exemplo do robô o objetivo era saber a posição x_i dada uma medida do sensor Z, que indicava a presença de uma porta ou de uma parede. Ou seja, estava sendo calculado $P(x_i \mid Z)$. Utilizando a equação (2): $P(Z \mid x_i)$ é a função de verossimilhança ou a probabilidade da medida para cada posição x_i; $P(x_i)$ é a priori, que é o conhecimento antes de ter a medida; $P(Z)$ é a probabilidade da medida Z independente da localização do robô e serve como um fator de normalização para tornar a multiplicação de $P(Z \mid x_i)P(x_i)$ uma distribuição de probabilidade. O termo de normalização pode ser encontrado na literatura com outra forma, que será abordada quando o Filtro de Partículas for apresentado (seção 4.5).

A probabilidade do robô estar em qualquer posição i é expressa como $P(x_i)$ e isso pode ser calculado como a soma da priori $P(x_i)$ multiplicado pela probabilidade de se mover da posição x_i para x_j, dado pela equação (3), onde N é o número total de posições. Esta equação é conhecida como teorema de probabilidade total:

$$P(x_i) = \sum_{j=1}^{N} P(x_j)P(x_i|x_j) \tag{3}$$

Todos os filtros presentes nessa dissertação são Bayesianos e eles vão ter a mesma estrutura apresentada pela Figura 5. O filtro Bayesiano depois de inicializado é baseado em duas etapas: predição, em que é utilizado o estado anterior do sistema para propagar no tempo uma estimativa a priori do estado futuro; correção, que é a correção do estado utilizando a medida obtida por meio do sensor (z_k), gerando uma estimativa a posteriori dos estados do sistema ($\hat{x}_k$).

3.2 Variável Aleatória

Normalmente uma variável aleatória (v.a.) é representada utilizando a letra maiúscula X. Definindo X como uma v.a. que representa o lançamento de um dado (de 6 lados), por exemplo, o espaço amostral, Ω, é um subconjunto dos números naturais composto por $[1, 2, 3, 4, 5, 6]$. Ao jogar um dado a probabilidade de obter um três é igual a $\frac{1}{6}$. Caso seja obtido o valor três, este resultado é chamado de uma realização da v.a. X. O resultado

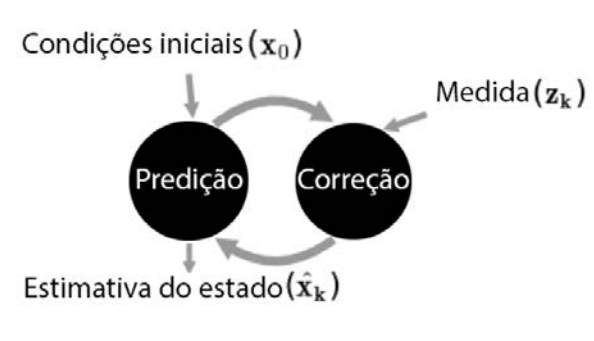


Figura 5: Funcionamento do filtro Bayesiano.

de um experimento não é uma v.a., ou seja, a v.a. existe independentemente de suas realizações.

O comportamento de uma v.a. é perfeitamente determinado pela sua função de distribuição (FD), que é definida como:

$$F_X(x) = P(X \leq x) \quad \forall x \in \mathbb{R} \tag{4}$$

A partir da FD as v.a.s podem ser classificadas em discretas e contínuas. Quando o conjunto de valores possíveis de uma v.a. X for enumerável ela é uma v.a. discreta. Já se X for um conjunto não enumerável ela é uma v.a. contínua.

A função massa de probabilidade (f.m.p) associa a cada possível ocorrência de uma v.a. discreta uma probabilidade, ou seja:

$$p_X(x) = P(X = x) \tag{5}$$

A f.m.p de uma v.a. X é uma função que satisfaz as seguintes condições ((SIMON, 2006)):

1. $p_X(x) \geq 0 \quad \forall x \in \mathbb{R}$
2. $\sum\limits_x p_X(x) = 1$

A primeira condição indica que qualquer probabilidade deve ser um valor positivo, visto que nenhuma probabiliade pode ser menor do que zero. A segunda condição mostra que a soma de todas as probabilidades deve ser igual a 1, neste caso o somatório é feito para o conjunto de todos os valores possíveis de X ($\forall x \in \mathrm{Im}(X)$).

Para casos onde a v.a. X é contínua, utiliza-se a função densidade de probabilidade (f.d.p), $f_X(x)$. Esta é definida como a derivada da FD.

$$f_X(x) = \frac{dF_X(x)}{dx} \tag{6}$$

De forma semelhante à f.m.p, a soma de todas as probabilidades deve ser igual a um, mas ao invés de utilizar um somatório utiliza-se uma integral. A f.d.p possui as seguintes propriedades (SIMON, 2006):

1. $f_X(x) \geq 0 \quad \forall x \in \mathbb{R}$
2. $\int_{-\infty}^{\infty} f_X(x)dx = 1$

Na seção 3.1 foi abordado a probabilidade condicional. A distribuição condicional e a densidade de uma v.a. X dado que um evento A ocorreu são definidas como (SIMON, 2006):

$$F_X = (x \mid A) = P(X \leq x \mid A) \tag{7}$$

$$f_X(x \mid A) = \frac{dF_X(x \mid A)}{dx} \tag{8}$$

O teorema de Bayes pode ser generalizado utilizando densidades condicionais. A f.d.p condicional de uma v.a. X_1 dado o fato que a v.a. X_2 é igual a realização x_2 é definida como:

$$\begin{aligned} f_{X_1|X_2}(x_1 \mid x_2) &= P[(X_1 \leq x_1) \mid (X_2 = x_2)] \\ &= \frac{f_{X_1,X_2}(x_1, x_2)}{f_{X_2}(x_2)} \end{aligned} \tag{9}$$

A esperança ou valor esperado de uma v.a. X é a média ponderada dos valores que X pode assumir, onde cada valor possível é ponderado por sua respectiva probabilidade: seja X uma v.a. discreta que assume os valores $\{x_1, x_2, \ldots\}$ e seja $p_X(x)$ a f.m.p de X. A esperança de X, é denotada por $\mathbb{E}(X)$ e dada por (10).

$$\mathbb{E}[X] = \sum_{x \in \mathrm{Im}(X)} x p_X \tag{10}$$

A esperança, $\mathbb{E}(X)$, só está definida se (10) for absolutamente somável. A esperança de X é a média dos seus valores, ponderada pelas respectivas probabilidades. Caso estes valores tiverem a mesma probabilidade de ocorrência ela será o mesmo que a média aritmética. De forma semelhante, a esperança de uma v.a. contínua com uma f.d.p $f_X(x)$ é dada pela equação (11) e esta só é definida se for absolutamente integrável.

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x \, f_X(x) \, dx \tag{11}$$

As v.a.s e suas distribuições de probabilidade são frequentemente caracterizadas por um pequeno número de parâmetros, os quais também têm uma interpretação prática, como é o caso da esperança. No entanto, as vezes a esperança pode não fornecer maiores detalhes sobre uma determinada amostra. Considere um caso onde se deseja medir a distância entre o robô e um obstáculo, onde serão observados cinco medidas obtidas por três sensores diferentes (S_1, S_2 e S_3). As cinco distâncias (em centímetros) obtidas, respectivamente, por cada um deles foram: $\{7; 7; 7; 7, 7 \text{ e } 6, 3\}$ $\{5; 9; 8; 6 \text{ e } 7\}$ $\{2; 4; 6; 8 \text{ e } 9, 5\}$.

A média fornece uma informação sobre os dados, mas não fornece maiores detalhes sobre o grupo amostrado. No exemplo acima a média dos sensores é a mesma (7 cm) e somente conhecendo este valor não é possível saber se as distâncias obtidas estão próximas da média ou não. No entanto, olhando os valores obtidos por cada sensor, pode-se perceber que existe uma variação maior no terceiro sensor do que no segundo, e que o primeiro possui a menor variação.

A estatística formalizou o conceito de medir a variação utilizando a notação de desvio padrão (σ) e variância (σ^2). A equação para calcular a variância é dada por (12).

$$\sigma^2 = VAR(X) = \mathbb{E}[(X - \mu)^2] \tag{12}$$

Percebe-se que a variância é o valor esperado para o quanto a v.a. X varia da média (ao quadrado). Assumindo que cada nota é equiprovável pode-se substituir o valor esperado pela média (μ), logo a variância de uma v.a. discreta pode ser escrita como 13.

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{13}$$

A unidade da variância é dada pelo quadrado da unidade, neste caso em cm^2. Esse valor acaba sendo de difícil interpretação, então, normalmente utiliza-se o desvio padrão, que é a raiz quadrada da variância. Dessa forma, o valor está expresso na mesma unidade que o medido. No caso do exemplo da distância medida pelos sensores o desvio padrão do primeiro sensor é de $0,49$ cm, o do segundo é $1,58$ cm e o do terceiro é 3 cm. O desvio padrão diz o quanto as distâncias variam entre si. O "quanto"não é um termo matemático, mas este será melhor definido quando for introduzido o conceito o conceito da distribuição Normal a seguir.

3.3 Distribuição Normal

A distribuição Normal ou Gaussiana é unimodal e é contínua. A notação utilizada para essa distribuição para uma v.a. X é: $X \sim \mathcal{N}(\mu, \sigma^2)$ (PAPOULIS; PILLAI, 2002). Ela é uma função densidade de probabilidade completamente descrita por μ e σ e é definida conforme a equação (14):

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(x-\mu)^2/\sigma^2} \tag{14}$$

As distribuições de probabilidade dos sensores (seção 3.2) são plotadas como se fossem uma distribuição Normal. A Figura 6 mostra as funções densidades de probabilidade com a mesma média ($\mu = 7$), porém com desvios padrão diferentes: $\sigma_{S1} = 1,58$ cm (em vermelho), $\sigma_{S2} = 0,49$ cm (em preto) e $\sigma_{S3} = 3$ cm (em azul). Olhando para essa figura pode-se dizer, por exemplo, que é mais provável que o sensor 1 leia um valor entre 6 ou 8 cm do que um valor menor do que 6 ou maior do que 8 cm.

Como a distribuição Gaussiana é contínua a probabilidade da medida ser exatamente um número é zero, pois existem infinitos valores possíveis. Para calcular a probabilidade da medida estar entre 5 cm e 7 cm, por exemplo, pode-se calcular a área sob a curva da função de probabilidade de interesse entre esse intervalo, simplesmente integrando a equação (14) entre os limites desejados, conforme a equação (15).

$$\int_a^b \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(x-\mu)^2/\sigma^2} dx \tag{15}$$

Caso esses limites sejam de $-\infty$ a ∞ o resultado dessa integral é 1, pois ela é uma função densidade de probabilidade. Olhando para (14), pode-se perceber que a variável que altera o formato da Gaussiana é o σ. Como a área sob a curva é sempre igual a 1, uma variância pequena resultará em uma curva estreita, enquanto que uma variância grande deixará a curva mais larga, o que pode ser observado na Figura 6 para diferentes valores de σ.

A distribuição Normal é muito conhecida por descrever uma série de fenômenos presentes na natureza, financeiros e muitos outros, inclusive até mesmo o de sensores. Na verdade, quase tudo pode ser aproximado por uma Gaussiana utilizando o Teorema Central do Limite. Este afirma que quando o tamanho da amostra aumenta, a distribuição

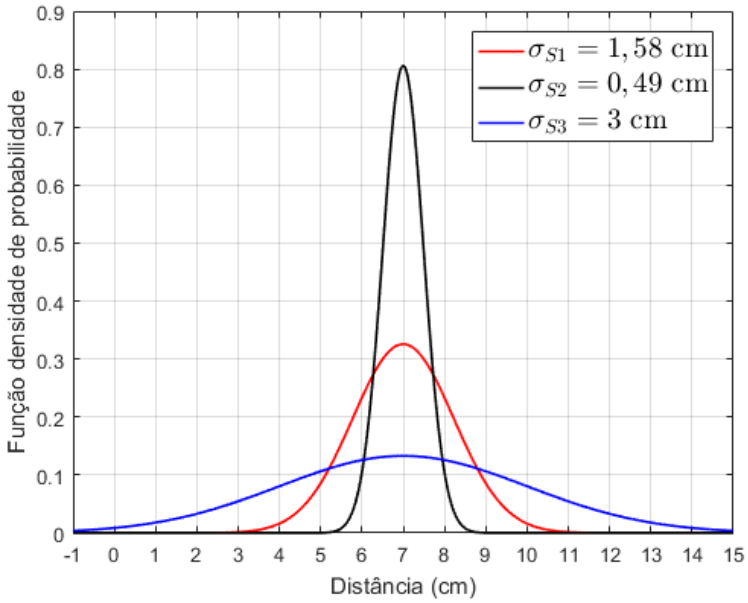


Figura 6: Aproximação do sensor 1 por uma distribuição normal com $\mu = 7$ e $\sigma = 1,58$.

amostral da sua média (v.a.s independentes com mesma média e variância) aproxima-se cada vez mais de uma distribuição normal (PAPOULIS; PILLAI, 2002).

Pode-se perceber que na Figura 6 existe uma probabilidade do valor lido pelo sensor ser menor do que zero, que de fato é impossível, ou então do valor da distância ser maior do que 10, o quê não ocorre em nenhum dos casos. Isso é uma limitação do modelo matemático, e neste caso ele é imperfeito. A distribuição Gaussiana é usada em diversos ramos da matemática não pelo fato de ser a mais precisa, mas pelo fato de ser completamente descrita por apenas dois parâmetros (μ e σ) e fazer uma boa aproximação.

Anteriormente foi mencionado o significado de estar a um desvio padrão abaixo ou a dois acima não tinha um significado claro. Sabe-se que o desvio padrão é a medida de quanta variação existe da média; para distribuições Gaussianas, 68.26% das observações estão a um desvio padrão da média, 95.44% a dois e 99.73% a três.

No caso onde a distribuição de probabilidade do sensor 1 foi aproximada por uma Normal com $\mu = 7$ cm e $\sigma = 1,58$ cm, espera-se que 68.26% das distâncias estejam entre $5,42$ cm e $8,58$ cm. No exemplo só existem cinco medidas e três delas estão dentro do primeiro desvio (60%) e as outras duas estão dentro de dois desvios (100%). Isso não retrata exatamente a distribuição Gaussiana, mas é o mais próximo que se pode chegar através da aproximação feita utilizando apenas cinco amostras.

3.3.1 Propriedades da distribuição Gaussiana

Além da distribuição Normal permitir capturar um número infinito de valores utilizando somente a média e a variância, ela possui outra propriedade muito importante referente à soma e ao produto entre Gaussianas.

Seja X e Y duas v.a.s independentes com funções de densidade $f_X(x)$ e $f_Y(y)$ definidas para todo x. Então a soma $Z = X + Y$ é uma v.a. com função de densidade $f_Z(z)$, onde $f_Z(z)$ é a convolução de suas densidades $f_Z(z) = f_X(x) * f_Y(y)$. Utilizando duas v.a.s $X \sim \mathcal{N}(\mu_1, \sigma_1^2)$ e $Y \sim \mathcal{N}(\mu_2, \sigma_2^2)$ a sua soma é dada por (16):

$$f_Z(z) = \int_{-\infty}^{\infty} f_y(z-x) f_x(x)\, dx$$

$$f_Z(z) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_Y} \exp\left(-\frac{(z-x-\mu_Y)^2}{2\sigma_Y^2}\right) \frac{1}{\sqrt{2\pi}\sigma_X} \exp\left(-\frac{(x-\mu_X)^2}{2\sigma_X^2}\right) dx$$

$$f_Z(z) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sqrt{\sigma_X^2+\sigma_Y^2}} \exp\left(-\frac{(z-(\mu_X+\mu_Y))^2}{2(\sigma_X^2+\sigma_Y^2)}\right) \frac{1}{\sqrt{2\pi}\frac{\sigma_X\sigma_Y}{\sqrt{\sigma_X^2+\sigma_Y^2}}} \exp\left(-\frac{\left(x-\frac{\sigma_X^2(z-\mu_Y)+\sigma_Y^2\mu_X}{\sigma_X^2+\sigma_Y^2}\right)^2}{2\left(\frac{\sigma_X\sigma_Y}{\sqrt{\sigma_X^2+\sigma_Y^2}}\right)^2}\right) dx$$

$$f_Z(z) = \frac{1}{\sqrt{2\pi(\sigma_X^2+\sigma_Y^2)}} \exp\left(-\frac{(z-(\mu_X+\mu_Y))^2}{2(\sigma_X^2+\sigma_Y^2)}\right) \underbrace{\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\frac{\sigma_X\sigma_Y}{\sqrt{\sigma_X^2+\sigma_Y^2}}} \exp\left(-\frac{\left(x-\frac{\sigma_X^2(z-\mu_Y)+\sigma_Y^2\mu_X}{\sigma_X^2+\sigma_Y^2}\right)^2}{2\left(\frac{\sigma_X\sigma_Y}{\sqrt{\sigma_X^2+\sigma_Y^2}}\right)^2}\right) dx}_{f_x(x,\mu,\sigma^2)=1}$$

$$f_Z(z) = \frac{1}{\sqrt{2\pi}\sqrt{\sigma_X^2+\sigma_Y^2}} \exp\left(-\frac{(z-(\mu_X+\mu_Y)))^2}{2(\sigma_X^2+\sigma_Y^2)}\right) \tag{16}$$

Portanto, a densidade (16) é uma Gaussiana com média (17a) e variância (17b).

$$\mu_Z = \mu_X + \mu_Y \tag{17a}$$
$$\sigma_Z^2 = \sigma_X^2 + \sigma_Y^2 \tag{17b}$$

A função Gaussiana é uma função não linear, e normalmente ao multiplicar uma função não linear por ela mesma resulta em uma função completamente diferente. No entanto, o produto entre duas distribuições Gaussianas resulta em outra distribuição Gaussiana. O produto entre duas Gaussianas independentes ($X \sim \mathcal{N}(\mu_x, \sigma_x^2)$ e $Y \sim \mathcal{N}(\mu_y, \sigma_y^2)$) é derivado por meio do teorema de Bayes (2) e o produto entre duas Normais é dado por (18):

$$P(X \mid Y) = \frac{P(Y \mid X)P(X)}{P(Y)}$$
$$P(X \mid Y) \propto P(Y|X)P(X)$$
$$P(X \mid Y) \propto \exp\Big[-\frac{(y-x)^2}{2\sigma_y^2}\Big] \exp\Big[-\frac{(x-\mu_x)^2}{2\sigma_x^2}\Big]$$
$$P(X \mid Y) \propto \exp\Big[-\frac{(y-x)^2}{2\sigma_y^2} - \frac{(x-\mu_x)^2}{2\sigma_x^2}\Big]$$
$$P(X \mid Y) \propto \exp\Big[-\frac{1}{2\sigma_y^2\sigma_x^2}[\sigma_x^2(y-x)^2 - \sigma_y^2(x-\mu_x)^2]\Big]$$
$$P(X \mid Y) \propto \exp\Big[-\frac{1}{2\sigma_y^2\sigma_x^2}[\sigma_x^2(y^2-2xy+x^2) + \sigma_y^2(x^2-2x\mu_x+\mu_x^2)]\Big]$$

$$P(X \mid Y) \propto \exp\Big[-\frac{1}{2\sigma_y^2\sigma_x^2}[x^2(\sigma_x^2+\sigma_y^2) - 2x(\sigma_y^2\mu_x + \sigma_x^2 y) + (\sigma_x^2 y^2 + \sigma_y^2\mu_x^2)]\Big]$$

$$P(X \mid Y) \propto \exp\Big[-\frac{1}{2}\frac{x^2(\sigma_x^2+\sigma_y^2) - 2x(\sigma_y^2\mu_x + \sigma_x^2 y)}{\sigma_y^2\sigma_x^2}\Big]$$

$$P(X \mid Y) \propto \exp\Big[-\frac{1}{2}\frac{x^2 - 2x(\frac{\sigma_y^2\mu_x+\sigma_x^2 y}{\sigma_x^2+\sigma_y^2})}{\frac{\sigma_y^2\sigma_x^2}{\sigma_x^2+\sigma_y^2}}\Big]$$

$$P(X \mid Y) \propto \exp\Big[-\frac{1}{2}\frac{(x - \frac{\sigma_y^2\mu_x+\sigma_x^2 y}{\sigma_x^2+\sigma_y^2})^2}{\frac{\sigma_y^2\sigma_x^2}{\sigma_x^2+\sigma_y^2}}\Big]$$

O termo $P(Y)$ é somente um fator de normalização, portanto pode-se dizer que a posteriori $P(X \mid Y)$ é proporcional à $P(Y|X)P(X)$. Em seguida é feita a multiplicação entre as duas distribuições Gaussianas e expande-se os termos ao quadrado e agrupa-se os termos que contém a posteriori x. Pode-se utilizar a proporcionalidade para remover as constantes (termos que não possuem o x). Isso resulta no fato de uma multiplicação de duas Gaussianas ser proporcional a uma outra Gaussiana.

$$\mu_Z = \frac{\sigma_Y^2\mu_X + \sigma_X^2\mu_Y}{\sigma_X^2 + \sigma_Y^2} \tag{18a}$$

$$\sigma_Z^2 = \frac{\sigma_Y^2\sigma_X^2}{\sigma_X^2 + \sigma_Y^2} \tag{18b}$$

3.3.2 Distribuição Normal Multivariada

A distribuição Normal multivariada é uma generalização da Normal univariada. Para fins ilustrativos será considerado um caso bidimensional. Casos com dimensão maior são de difícil visualização. Porém, toda matemática envolvida será a mesma. A única diferença é que o vetor contendo as médias ($\boldsymbol{\mu}$) e a matriz contendo a covariância ($\boldsymbol{\Sigma}$) possuirão dimensões maiores. Para uma dimensão n, serão necessárias n médias:

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}$$

A covariância indica o quanto duas variáveis variam juntas, onde covariância é uma abreviação de variâncias correlacionadas. Quando a esperança do produto entre duas v.a.s for igual ao produto de suas esperanças elas são ditas descorrelacionadas e caso seja diferente de zero elas são correlacionadas. A equação (19) mostra a covariância entre da v.a X.

$$COV(X,Y) = \Sigma = \mathbb{E}\big[(X-\mu_x)(X-\mu_x)\big] \tag{19}$$

Utiliza-se uma matriz de covariância para representar as covariâncias de uma v.a. Para uma dimensão n, $\boldsymbol{\Sigma}$ é dado por:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} & \cdots & \sigma_{1,n} \\ \sigma_{2,1} & \sigma_2^2 & \cdots & \sigma_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n,1} & \sigma_{n,2} & \cdots & \sigma_n^2 \end{bmatrix}$$

A diagonal de $\boldsymbol{\Sigma}$ contém a variância para cada variável escalar e todos os elementos fora da diagonal contém a covariância entre a i-ésima e a j-ésima do vetor x. Dessa forma, σ_7^2 é a variância da sétima variável, e $\sigma_{1,5}$ é a covariância entre a primeira e a quinta variável.

A distribuição normal multivariada é dada pela equação (20). A versão multivariada simplesmente troca os escalares da versão univariada, média e variância, por uma por um vetor e uma matriz, respectivamente.

$$f(\mathbf{x}, \mu, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu)^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\mu)\right] \tag{20}$$

A Figura 7 mostra uma Gaussiana multivariada com $\boldsymbol{\mu} = \left[\begin{smallmatrix} 2 \\ 1 \end{smallmatrix}\right]$ e com covariância de $\boldsymbol{\Sigma} = \left[\begin{smallmatrix} 0{,}5 & 0 \\ 0 & 5 \end{smallmatrix}\right]$. O gráfico tridimensional mostra a densidade de probabilidade conjunta para qualquer valor (X, Y) no eixo z e a projeção de cada uma das Gaussianas no fundo do gráfico. A densidade de probabilidade conjunta, denotada por $P(X, Y)$ é a probabilidade de tanto X como Y ocorrer simultaneamente e para ser válida ela tem que cumprir os mesmos requisitos que o de uma distribuição de probabilidade: $P(x_i, y_j) \geq 0$ e a soma de todas a probabilidades deve ser igual a um.

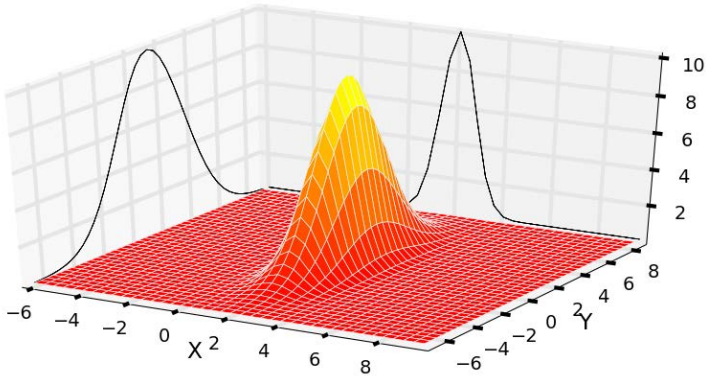


Figura 7: Ilustração de uma distribuição normal multivariada.

A soma e a multiplicação de Gaussianas multivariadas $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x)$ e $\mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$ são semelhantes à univariada. A soma possui a mesma forma que (16), no entanto a média é dada pela soma dos vetores de média a e covariância pela soma das matrizes de covariância. A multiplicação de duas gaussianas (21) também possui a mesma forma que (18), mas neste caso o denominador é representado pela matriz inversa.

$$\boldsymbol{\mu} = \boldsymbol{\Sigma_y}(\boldsymbol{\Sigma_x} + \boldsymbol{\Sigma_y})^{-1}\boldsymbol{\mu_x} + \boldsymbol{\Sigma_x}(\boldsymbol{\Sigma_x} + \boldsymbol{\Sigma_y})^{-1}\boldsymbol{\mu_y} \tag{21a}$$

$$\boldsymbol{\Sigma} = \boldsymbol{\Sigma_x}(\boldsymbol{\Sigma_x} + \boldsymbol{\Sigma_y})^{-1}\boldsymbol{\Sigma_y} \tag{21b}$$

4 FILTROS BAYESIANOS

4.1 Introdução

Neste capítulo são apresentados os filtros Bayesianos utilizados neste trabalho. Inicialmente é introduzido o filtro de Kalman, que é um filtro Gaussiano. Filtros Gaussianos são métodos que modelam todas as v.a.s do sistema do sistema por meio de uma função de densidade de probabilidade Gaussiana. Em seguida são mostrados os efeitos de funções não lineares em Gaussianas e duas extensões do Filtro de Kalman para lidar com modelos não lineares: o Filtro de Kalman Estendido e o Filtro de Kalman Unscented. Por fim, um filtro não Gaussiano, Filtro de Partículas, é apresentado.

Para realizar a estimação de estados, o primeiro passo é construir um modelo matemático do sistema em questão. No espaço de estados, um sistema dinâmico pode ser representado por dois conjuntos de equações: as equações de processo (22) e as equações de medida (23). Seja um sistema linear estocástico descrito pelo modelo em espaço de estados:

$$\boldsymbol{x}_k = \boldsymbol{F}_k \boldsymbol{x}_{k-1} + \boldsymbol{B}_k \boldsymbol{u}_k + \boldsymbol{w}_k \tag{22}$$

$$\boldsymbol{z}_k = \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{v}_k \tag{23}$$

Nestas equações, $\boldsymbol{x}_k$ representa o estado que se deseja estimar a cada instante k, $\boldsymbol{F}_k$ é a matriz de transição de estados, que descreve a evolução temporal das variáveis de estados entre os instantes $k-1$ e k; $\boldsymbol{B}_k$ é o modelo das entradas de controle, aplicado no vetor das entradas de controle $\boldsymbol{u}_k$; $\boldsymbol{H}_k$ é a matriz de medição que converte o estado na medida e $\boldsymbol{z}_k$ é o vetor com os sinais medidos no instante k; $\boldsymbol{w}_k$ e $\boldsymbol{v}_k$ são os vetores contendo os ruídos de estado e os ruídos de medidas, respectivamente.

4.2 Filtro de Kalman

O filtro de Kalman é um filtro Bayesiano que é baseado na descrição em espaço de estados de um sistema dinâmico linear e que fornece uma solução recursiva de mínimo erro quadrático médio para o problema de estimação de estados. Ao invés de utilizar histogramas, como o do exemplo na seção 3, ele utiliza Gaussianas. Como foi visto, tanto a soma quanto a multiplicação de Gaussianas podem ser calculadas de forma fácil e analítica e esta é uma propriedade fundamental, e uma das principais razões pela qual o FK possui um custo computacional reduzido.

No FK assume-se que os ruídos $\boldsymbol{w}_k$ e $\boldsymbol{v}_k$ em 22 e 23 são brancos, Gaussianos, descorrelacionados, com média zero e matrizes de covariância dadas por: $\boldsymbol{Q_k} = \mathbb{E}[\boldsymbol{w_k}\boldsymbol{w_k}^T]$ e $\boldsymbol{R_k} = \mathbb{E}[\boldsymbol{v_k}\boldsymbol{v_k}^T]$. Assume-se também que o estado inicial $\boldsymbol{x}_0$ é uma v.a. Gaussiana com

média $\hat{x}_0 = \mathbb{E}[x_0]$ e covariância $\boldsymbol{P}_0 = \mathbb{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]$. Caso sejam atendidos esses requisitos o FK produz uma estimativa não polarizada e é ótimo para a estimação do estado de um processo linear.

As equações que regem o FK são dadas por (24) e as suas derivações estão disponíveis amplamente na literatura, podendo ser encontradas com maiores detalhes em (ANDERSON; MOORE, 2012) e (THRUN; BURGARD; FOX, 2005). O FK é dividido em duas etapas: na etapa de predição a equação de processo é usada para realizar uma predição do valor do estado e de sua incerteza num instante k, a partir da estimativa do estado calculada no instante $k-1$; na etapa de correção, os valores preditos para as variáveis de estados são corrigidos a partir da equação de medida e dos sinais observados no instante k e atualiza-se a sua covariância.

$$
\begin{aligned}
\hat{\boldsymbol{x}}_{k|k-1} &= \boldsymbol{F_k}\hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{B}_k\boldsymbol{u}_k \\
\boldsymbol{P}_{k|k-1} &= \boldsymbol{F}_k\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}_k^{\mathsf{T}} + \boldsymbol{Q}_k \\
\tilde{\boldsymbol{y}}_k &= \boldsymbol{z}_k - \boldsymbol{H}_k\hat{\boldsymbol{x}}_{k|k-1} \\
\boldsymbol{K}_k &= \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^{\mathsf{T}}(\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^{\mathsf{T}} + \boldsymbol{R}_k)^{-1} \\
\hat{\boldsymbol{x}}_{k|k} &= \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\tilde{\boldsymbol{y}}_k \\
\boldsymbol{P}_{k|k} &= (I - \boldsymbol{K}_kH_k)\boldsymbol{P}_{k|k-1}
\end{aligned}
\tag{24}
$$

As duas primeiras equações de 24 são responsáveis pela parte da predição. Como pode ser percebida as estimativas do estado e da matriz de covariância do erro de estimação ($\boldsymbol{P}_{k|k-1}$) são atualizadas com base, apenas, no conhecimento da dinâmica do sistema e na estimativa anterior.

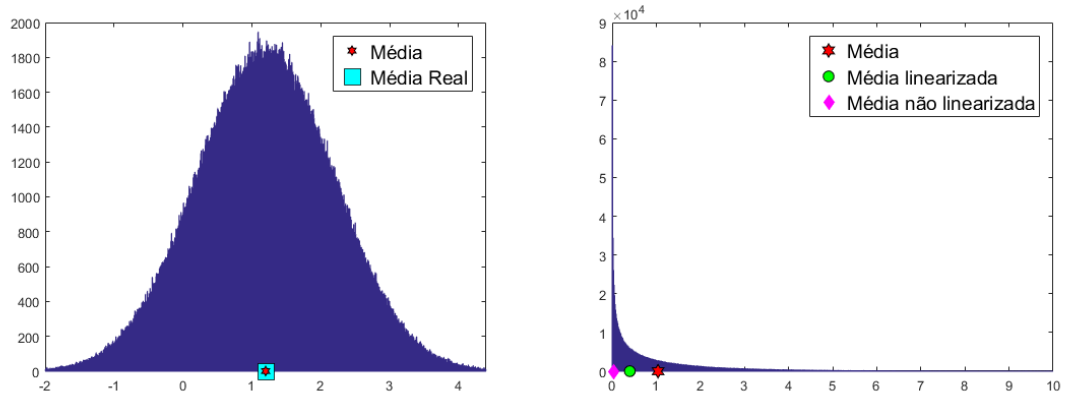
A matriz $\boldsymbol{H}_k$ projeta projeta o estado e a matriz de covariância no espaço de medição. A matriz $\boldsymbol{K}_k$, que é chamada de ganho de Kalman, é calculada como a quarta equação de (24),onde este ganho gera as estimativas de mínimo erro quadrático. A estimativa a posteriori $\hat{\boldsymbol{x}}_{k|k}$ é uma combinação da melhor estimativa linear de $\boldsymbol{x}_k$ baseada na predição e um termo de correção, $\boldsymbol{K}_k\tilde{\boldsymbol{y}}_k$, que representa o erro na predição de $\boldsymbol{x}_k$ a partir de $\hat{\boldsymbol{x}}_{k|k-1}$. Um valor baixo de $\tilde{\boldsymbol{y}}_k$, por exemplo, indica que a estimativa $\boldsymbol{H}_k\hat{\boldsymbol{x}}_{k|k-1}$ está próxima do valor medido.

4.2.1 Gaussianas e funções não lineares

O FK apresentado utiliza equações lineares e ele é ótimo, no entanto o seu desempenho deixa de ser ótimo com a presença de não linearidades. Para lidar com problemas não lineares, normalmente encontra-se uma maneira de linearizar o problema, transformando-o em um conjunto de equações lineares e/ou utiliza-se um *software* para encontrar uma solução aproximada. Porém, a linearização de um problema não linear pode levar a respostas inexatas e em um algoritmo recursivo como o FK, esses erros, mesmo que pequenos, se acumulam em cada passo e podem fazer com que o algoritmo divirja.

São apresentados dois exemplos utilizando duas f_x diferentes, uma linear, $f_1x = x+1$ e outra não linear, $f_2x = x^2$. Durante a etapa de predição do FK utiliza-se uma Gaussiana representando o estado e, em seguida, ela passa pela função do processo ou modelo do processo (f_x), que é o que ocorre na predição do FK.

A Gaussiana é gerada utilizando 1 milhão de pontos com uma distribuição Normal ($\mu = 0,2$ e $\sigma^2 = 2$), onde cada um desses pontos serve de entrada para a f_1x e a f_2x. A Figura 8 mostra o histograma da saída de cada uma das funções.



(a) (b)

Figura 8: Distribuição de probabilidade: utilizando $f_1x = x + 1$ (a); utilizando $f_2x = x^2$ (b).

A Figura 8a é uma Gaussiana com a média deslocada de $0,2$ para $1,2$ e neste caso ela mantém a mesma variância. Caso função utilizada estivesse na forma de $\alpha x + 1$ e $\alpha \geq 1$ a variância aumentaria e se $0 \leq \alpha \leq 1$ a variância diminuiria. A média desta função pode ser calculada de forma analítica e a média dessa Gaussiana é 1,2. A média obtida utilizando todos os pontos foi 1,198, que se aproxima da média real.

A função f_2x não é linear e não é possível calcular a sua distribuição de uma forma analítica, portanto utiliza-se a técnica de injetar todos esses 1 milhão de pontos gerados para calcular a sua distribuição e a sua média. A distribuição obtida não é uma Normal, conforme pode ser observado na Figura 8b.

Caso fosse ignorado que essa função não é linear, ou seja, utilizando o Filtro de Kalman a sua distribuição seria uma Gaussiana com média 0,04. Se fosse feita uma aproximação por meio de uma linearização a sua distribuição também é aproximada por uma Gaussiana, porém com uma média de 0,4. No entanto, ao calcular a média utilizando todos os pontos, que é uma aproximação do valor real da média, o seu valor é igual a 1,04 e, como pode ser observado, a sua distribuição não é Gaussiana.

Todas as equações envolvidas no FK assumem que uma Gaussiana passada através da f_x resulta em outra Gaussiana. Se isso não for verdade, então todas as suposições e garantias do FK não são mantidas. Logo, as aproximações feitas pela versão linearizada e não linearizada introduzirão um erro devido a sua média, variância e distribuição incorretas. Conforme o tempo passar, dependendo da não linearidade esse erro crescerá fazendo com que o valor do estado seja completamente diferente do seu valor real.

A forma utilizada para linearizar a função f_2x é a mesma do que o FKE e essa forma pode introduzir grandes erros conforme a não linearidade presente no sistema. O FKE lineariza as equações diferenciais em um ponto, o que requer encontrar a Jacobiana. Isso em diversos casos pode ser fácil de encontrar, mas em alguns casos pode ser extremamente difícil ou até impossível de resolver analiticamente. Existem técnicas numéricas para encontrar a Jacobiana, no entanto isso pode ter um custo computacional elevado e pode introduzir erros no sistema.

Há uma outra variação do FK para problemas não lineares, que é o Filtro de Kalman *Unscented*. Este tem como vantagem tratar o problema de estimação sem a necessidade de linearizar o modelo do sistema e o modelo de medição. No entanto, enquanto que o FKE utiliza uma matriz com dimensão n para estimar os estados o FKU utiliza uma

matriz com dimensão $2n + 1$.

Existe uma outra técnica que pode resolver qualquer tipo de problema não linear, esta técnica é conhecida como Método Sequencial de Monte Carlo ou Filtro de Partículas. Este método gera milhares de pontos aleatórios e projeta esses pontos conforme o modelo do sistema (da mesma forma que foi feito nos exemplos acima). Porém, esta técnica envolve a utilização de um número grande pontos fazendo com que a sua computação seja muito mais lenta quando comparada com os Filtros de Kalman, por exemplo.

A seguir serão apresentados os três filtros Bayesianos e os algoritmos utilizados neste trabalho. Os algoritmos que serão apresentados a seguir assumem que os modelos de medição e do processo são lineares com respeito ao ruído (ruído aditivo).

4.3 Filtro de Kalman Estendido

O filtro de Kalman utiliza equações lineares, portanto não funciona com problemas não-lineares, como foi visto na seção 4.2.1. As equações podem ser não lineares no modelo do processo e/ou no modelo de medição. Agora representa-se a expressão do modelo não linear $\boldsymbol{F}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$ pela função não linear $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ e a expressão referente ao modelo de medição $\boldsymbol{H}\boldsymbol{x}$ pela função $\boldsymbol{h}(\boldsymbol{x})$.

$$\boldsymbol{F}_{k-1} = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k-1|k-1}} \tag{25a}$$

$$\boldsymbol{H}_{k} = \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k|k-1}} \tag{25b}$$

O Filtro de Kalman Estendido (FKE) não altera as equações lineares do FK, ao invés disso, ele lineariza as equações não lineares no ponto do estado estimado atual e utiliza essa aproximação como uma Gaussiana no FK padrão. Então o FKE é simplesmente um FK onde as matrizes do processo e de medição são dadas por 25 e sua saída é uma sequência de estimativas de estado $\hat{\boldsymbol{x}}_{k|k}$ e com uma matriz de covariância $P_{k|k}$. Os passos do FKE são dados pelo algoritmo 1 (ANDERSON; MOORE, 2012), onde é feita uma iteração de N passos.

Algoritmo 1 Filtro de Kalman Estendido

Inicialização:

$\hat{\boldsymbol{x}}_0 = \mathbb{E}[\boldsymbol{x}_0]$; $\boldsymbol{P}_0 = \mathbb{E}[(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0])(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0])^T]$

FOR $k = 1, 2, ...N$

Predição:

$\hat{\boldsymbol{x}}_{k|k-1} = f(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_{k-1})$

$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_{k-1}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}_{k-1}^{\top} + \boldsymbol{Q}_{k-1}$

Correção:

$\tilde{\boldsymbol{y}}_k = \boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1})$

$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^{\top}(\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^{\top} + \boldsymbol{R}_k)^{-1}$

$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\tilde{\boldsymbol{y}}_k$

$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_{k|k-1}$

ENDFOR

4.4 Filtro de Kalman Unscented

O Filtro de Kalman *Unscented* (FKU) não realiza a linearização de primeira ordem conforme o FKE, este é baseado na transformada *Unscented* (WAN; Van Der Merwe, 2000), (JULIER; UHLMANN; DURRANT-WHYTE, 2000). A ideia central do FKU é semelhante à abordagem utilizada na seção 4.2.1, que utiliza o método de Monte Carlo: encontrar a média de uma distribuição normal ao passá-la por uma função não linear. No entanto, a solução foi obtida utilizando 1 milhão de pontos e embora a média calculada seja precisa, esta grande quantidade de pontos é recalculada a cada nova etapa do filtro, fazendo com que sua performance (não paralelizada) seja lenta.

O FKU utiliza uma abordagem de amostragem determinística, onde a distribuição do estado também é aproximada por uma v.a. Gaussiana, mas agora é representada usando um conjunto mínimo de pontos, $2n+1$ (onde n é a dimensão da variável de estado) numa proporção da raiz quadrada da covariância, chamados de pontos sigma. Ao transformar uma v.a. através de qualquer função não linear analítica, as estimativas da média e da matriz de covariância apresentam erros de terceira ordem no máximo para qualquer v.a., enquanto que para o FKE os erros são de segunda ordem (WAN; Van Der Merwe, 2000).

Uma forma geral de calcular a média ao passar os pontos em uma função não linear é dada por 28, onde é calculada a média ponderada, por um peso W_i, dos pontos sigma ($\mathcal{X}$). Os pontos sigma por definição, satisfazem:

$$1 = \sum_{i=0}^{2n} W_i^m \tag{26}$$

$$1 = \sum_{i=0}^{2n} W_i^c \tag{27}$$

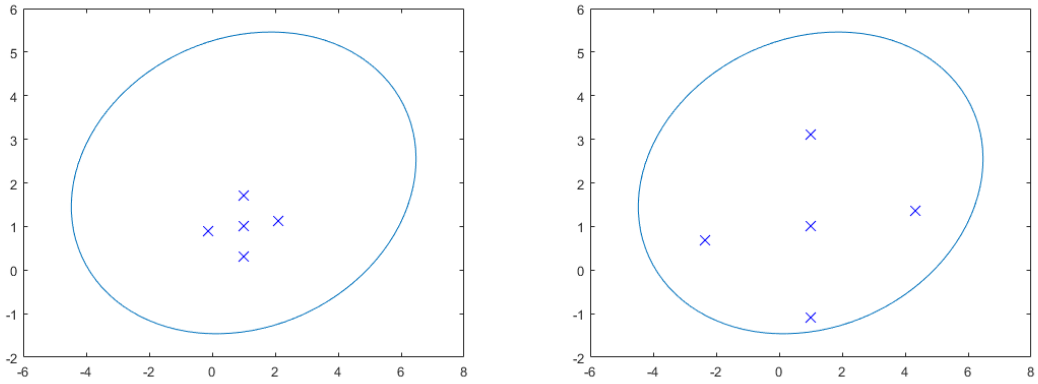
$$\mu = \sum_{i=0}^{2n} W_i^m f(\mathcal{X}_i) \tag{28}$$

$$\Sigma = \sum_{i=0}^{2n} W_i^c (f(\mathcal{X})_i - \mu)(f(\mathcal{X})_i - \mu)^\mathsf{T} \tag{29}$$

As duas primeiras equações (26 e 27) são as restrições que os pesos relacionados a média (W^m) e a covariância (W^c) devem somar um; a última equação (29) é a covariância. Essas restrições não formam uma solução única. Por exemplo, se W_0^m for pequeno, pode-se compensar fazendo $W_1^m, \ldots, W_{2n}^m$ maiores.

A disposição e a ponderação dos pontos sigma afetam a forma como a distribuição é amostrada. A Figura 9 mostra os pontos sigma, onde eles podem ser espalhados pela elipse (covariância) de qualquer forma: um ponto pode possuir um peso muito maior do que os outros; todos os pesos podem ser iguais; os pontos podem possuir um espaçamento igual ou diferente e etc.

O algoritmo mais utilizado para calcular os pontos sigma está descrito em (JULIER; UHLMANN; DURRANT-WHYTE, 2000). Essa formulação utiliza três parâmetros (α, β, κ) para controlar como os pontos sigma são distribuídos e ponderados. Onde α é um parâmetro que espalha os pontos sigma da média; κ é um parâmetro de escala secundário; β é usado para incorporar conhecimento prévio da distribuição. A Figura 9 mostra o efeito de alterar o parâmetro alfa, $\alpha = 0.5$ (Figura 9a) e $\alpha = 1.5$ (Figura 9b), na geração dos pontos sigma com $\boldsymbol{\mu} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $\boldsymbol{\Sigma} = \begin{bmatrix} 5 & 0.5 \\ 0.5 & 2 \end{bmatrix}$, $\kappa = -1$,$\beta = 2$.



(a) (b)

Figura 9: Pontos sigma utilizando diferentes valores do parâmetro α (a) 0.5 (b) 1.5

O cálculo dos pontos sigma é feito da seguinte forma: o ponto central é a média da entrada $\mathcal{X}_0 = \mu$ e os restantes são calculados conforme a equação (30), onde o subscrito i é o i-ésimo vetor coluna da matriz de covariância $\boldsymbol{\Sigma}$ e $\lambda := \alpha^2(n + \kappa) - n$.

$$\boldsymbol{\mathcal{X}}_i = \begin{cases} \mu + \left[\sqrt{(n+\lambda)\boldsymbol{\Sigma}}\right]_i & \text{para i=1 .. n} \\ \mu - \left[\sqrt{(n+\lambda)\boldsymbol{\Sigma}}\right]_{i-n} & \text{para i=(n+1) . . . 2n} \end{cases} \tag{30}$$

O cálculo dos pesos é feito da seguinte forma: o peso para a média do primeiro ponto $\mathcal{X}_0$ e o peso para a covariância de $\mathcal{X}_0$ são dados por 31 e 32, respectivamente. Os pesos para os demais pontos sigma são calculados da mesma forma (Equação 33).

$$W_0^m = \frac{\lambda}{n+\lambda} \tag{31}$$

$$W_0^c = \frac{\lambda}{n+\lambda} + 1 - \alpha^2 + \beta \tag{32}$$

$$W_i^m = W_i^c = \frac{1}{2(n+\lambda)} \tag{33}$$

O algoritmo 2 mostra como o FKU funciona. A etapa de predição do FKU calcula a priori usando o modelo do processo. Em seguida são gerados os pontos sigma $\mathcal{X}$ e seus pesos correspondentes W^m, W^c. Passando cada ponto sigma através da $\boldsymbol{f}$, projeta-se os pontos sigma no tempo, originando os novos pontos sigma $\boldsymbol{\mathcal{Y}}$. Após isso, calcula-se a média e a covariância da posteriori utilizando a Transformada *Unscented* nos pontos sigma transformados. Na etapa de correção, os pontos sigma da priori são convertidos em medições, usando a função de medição, então a média e a covariância desses pontos são calculadas usando a Transformada *Unscented* e o ganho K é dado pela covariância cruzada do estado e das medições. Os parâmetros que normalmente são utilizados para o FKU são $\alpha = 0,001, \beta = 2, \kappa = 3 - n$, onde n é a dimensão do estado (WAN; Van Der Merwe, 2000).

4.5 Filtro de Partículas

O Filtro de Partículas (FP), ou Método Sequencial de Monte Carlo, funciona com qualquer distribuição de probabilidade arbitrária e não-analítica, desde que o número de

Algoritmo 2 Filtro de Kalman *Unscented*

Inicialização:

$\hat{\boldsymbol{x}}_0 = E[\boldsymbol{x}_0]$; $\boldsymbol{P}_0 = E[(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0])(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0])^T]$

FOR $k = 1, 2, ...N$

Cálculo dos pontos σ e dos pesos:

$\boldsymbol{\mathcal{X}}_{k-1} = \left[\hat{\boldsymbol{x}}_{k-1}, \hat{\boldsymbol{x}}_{k-1} + \sqrt{(\lambda+n)\boldsymbol{P}}_{k-1}, \hat{\boldsymbol{x}}_{k-1} - \sqrt{(\lambda+n)\boldsymbol{P}}_{k-1}\right]$

$W_0^m = \frac{\lambda}{n+\lambda}$; $W_0^c = \frac{\lambda}{n+\lambda} + 1 - \alpha^2 + \beta$;

$W_i^m = W_i^c = \frac{1}{2(n+\lambda)} \quad i = 1..2n$

Predição:

$\boldsymbol{\mathcal{X}}_{k|k-1} = \boldsymbol{f}(\boldsymbol{\mathcal{X}}_{k-1}, \boldsymbol{u}_{k-1})$

$\hat{\boldsymbol{x}}_k^- = \sum_{i=0}^{2n} W_i^m (\boldsymbol{\mathcal{X}}_{i,k|k-1})$

$\boldsymbol{P}_k^- = \sum_{i=0}^{2n} W_i^c [(\boldsymbol{\mathcal{X}}_{i,k|k-1}) - \hat{\boldsymbol{x}}_k^-][(\boldsymbol{\mathcal{X}}_{i,k|k-1}) - \hat{\boldsymbol{x}}_k^-]^T + \boldsymbol{Q}$

Correção:

$\boldsymbol{\mathcal{Y}}_{k|k-1} = \boldsymbol{h}(\boldsymbol{\mathcal{X}}_{k|k-1})$

$\hat{\boldsymbol{y}}_k^- = \sum_{i=0}^{2n} W_i^m (\boldsymbol{\mathcal{Y}}_{i,k|k-1})$

$\boldsymbol{P}_{yy} = \sum_{i=0}^{2n} W_i^c [(\boldsymbol{\mathcal{Y}}_{i,k|k-1}) - \hat{\boldsymbol{y}}_k^-][(\boldsymbol{\mathcal{Y}}_{i,k|k-1}) - \hat{\boldsymbol{y}}_k^-]^T + \boldsymbol{R}$

$\boldsymbol{P}_{xy} = \sum_{i=0}^{2n} W_i^c [(\boldsymbol{\mathcal{X}}_{i,k|k-1}) - \hat{\boldsymbol{x}}_k^-][(\boldsymbol{\mathcal{Y}}_{i,k|k-1}) - \hat{\boldsymbol{y}}_k^-]^T$

$\boldsymbol{K}_k = \boldsymbol{P}_{xy}\boldsymbol{P}_{yy}^{-1}$

$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k)$

ENDFOR

partículas seja grande o suficiente, ele forma uma aproximação precisa da distribuição. Ele possui um bom desempenho mesmo na presença de não-linearidades acentuadas. O FP, de certo modo, é semelhante ao FKU, uma vez que transforma um conjunto de pontos através de equações não-lineares conhecidas. No entanto, há duas grande diferença entre eles: o FKU utiliza um pequeno número de pontos (escolhidos de forma determinística) e aproxima por uma Gauassiana enquanto que e o FP utiliza um grande número de pontos (escolhidos de forma aleatória). O FP dessa forma consegue estimar toda a f.d.p. O erro de estimação no FKU não converge para zero, mas o erro de estimação tende a zero no FP à medida que o número de partículas se tende ao infinito.

Para qualquer sistema existe uma função de distribuição de probabilidade que descreve o seu comportamento. Ao saber a sua distribuição é possível calcular a integral dela utilizando o método de Monte Carlo, como foi feito na seção 4.2.1. O método de Monte Carlo encontra uma solução para o cálculo da integral $\Phi = \int f(x)dx$, onde esta representa a função densidade de probabilidade, utilizando uma quantidade grande de amostras do integrando.

No entanto, existem problemas em que não se sabe como é essa distribuição. Há uma técnica que permite encontrar a verdadeira função densidade de probabilidade do sistema a partir de uma função densidade de probabilidade arbitrária. Este método é conhecido como amostragem por importância (do inglês, *Importance Sampling*). Supondo que existe um distribuição de probabilidade $\pi(x)$ que se quer amostrar, no entanto, não sabe como ela é; em vez disso, só se conhece uma distribuição de probabilidade alternativa $q(x)$. A esperança de uma função $f(x)$ com distribuição de probabilidade $\pi(x)$ é dada por 34.

$$\mathbb{E}\big[f(x)\big] = \int f(x)\pi(x)\,dx \tag{34}$$

Não é possível calcular essa integral, pois não se sabe $\pi(x)$. No entanto, se conhece uma

distribuição alternativa $q(x)$:

$$\mathbb{E}\big[f(x)\big] = \int f(x)\pi(x)\frac{q(x)}{q(x)}\,dx = \int f(x)q(x)\ \cdot\ \frac{\pi(x)}{q(x)}\,dx$$

Como sabe-se $q(x)$, então é possível calcular $\int f(x)q(x)$ pelo método de Monte Carlo e a razão $\pi(x)/q(x)$ é definida como uma peso, logo a esperança da $f(x)$ é dada por 35:

$$\mathbb{E}\big[f(x)\big] = \sum_{i=1}^{N_P} f(x^i)W(x^i) \tag{35}$$

Agora a função densidade de probabilidade é representada por uma distribuição discreta de probabilidade definida através de um conjunto de amostras (partículas) com seus respectivos pesos. Essas partículas são denotadas por $\chi_k = x_k^1, x_k^2, \ldots, x_k^{N_P}$, onde cada partícula x_k^n (com $n \leq N_P \in \mathbb{N}$) possui um peso W_k^n que deve satisfazer $\sum_{i=1}^{N_P} W_k^i = 1$ para ser uma densidade de probabilidade válida.

O FP é um filtro Bayesiano logo ele funciona utilizando a recursão em duas etapas, onde a predição é dada por (36):

$$P(x_k|z_{k-1}) = \int P(x_k|x_{k-1})P(x_{k-1}|z_{k-1})dx_{k-1} \tag{36}$$

em que z_{k-1} é a medição, a $P(x_{k-1}|z_{k-1})$ é conhecida através da recursão do algoritmo e $P(x_k|x_{k-1})$ é dado pelo modelo do sistema. Em seguida, a priori é atualizada com a nova medição utilizando o teorema de Bayes (37):

$$P(x_k|z_k) = \frac{P(z_k \mid x_k)P(x_k \mid z_{k-1})}{\int P(z_k|x_k)P(x_k \mid z_{k-1})\mathtt{d}x_k}) \tag{37}$$

Inicialmente, as partículas do FP são espalhadas uniformemente sobre toda a região com pesos iguais. Ao obter uma medição e realizar a etapa de correção, as partículas que não valores próximo aos obtidos pelas medições recebem um um peso baixo, enquanto que as partículas que estão próximas às medidas possuem um peso maior.

Partículas que possuem um peso muito pequeno não tem um impacto significativo na aproximação de $P(x_k|z_k)$. Quando o número de partículas que não contribuem é grande, costuma-se dizer que o filtro degenerou. O problema da degeneração pode ser resolvido utilizando um método de reamostragem de partículas. Este método descarta as partículas com probabilidades muito baixas e as substitui por novas partículas com probabilidades maiores. Isso é feito duplicando as partículas com probabilidades relativamente altas. Na próxima etapa de predição as partículas duplicadas serão dispersas pelo ruído presente no modelo do processo. Isso faz com que agora a grande maioria das partículas representem a distribuição de probabilidade.

A reamostragem não precisa e nem deve ser feita a cada passo. Pode-se determinar quando a reamostragem deve ser realizada utilizando o tamanho amostral efetivo $\hat{N}_{\text{eff}}$ (38), que mede aproximadamente o número de partículas que significativamente contribuem para a distribuição de probabilidade.

$$\hat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N_P}(W_k^i)^2} \tag{38}$$

A forma como as partículas são reamostradas impacta no desempenho do filtro. O que se busca é uma forma de selecionar uma população representativa das partículas de

maior probabilidade, mas este também deve incluir algumas partículas com probabilidade menores para que o filtro tenha chance de detectar comportamentos extremamente não-lineares. Os algoritmos mais utilizados na literatura são a reamostragem multinomial, residual, estratificada e sistemática (BOLIC; DJURIC; HONG 2005 e DOUC; CAPPE 2005).

É feita uma apresentação de cada um desses métodos, mas maiores detalhes podem ser encontradas em (FERNÁNDEZ-MADRIGAL, 2012). O método multinomial separa cada partícula em uma seção, onde o tamanho desta é proporcional ao peso da partícula e seleciona-se aleatoriamente N_P partículas.

No método residual os pesos são multiplicados por N_P, gerando novos pesos. Então o valor inteiro de cada peso é usado para definir quantas amostras dessa partícula serão tomadas. Isso garante que as partículas com maior peso sejam escolhidas pelo menos uma vez. No entanto, isso não gera as N_P partículas necessárias. As demais são obtidas utilizando o método multinomial para escolher as partículas baseado na parte decimal do novo peso.

O método sistemático divide em N_P seções e escolhe um *offset* aleatório para usar em todas as divisões, garantindo que cada partícula esteja $1/N_P$ de distância da outra. O método estratificado também divide em N_P seções, porém ele escolhe aleatoriamente uma partícula de cada seção e isso garante que as partículas estão com uma distância entre 0 e $2/N_P$.

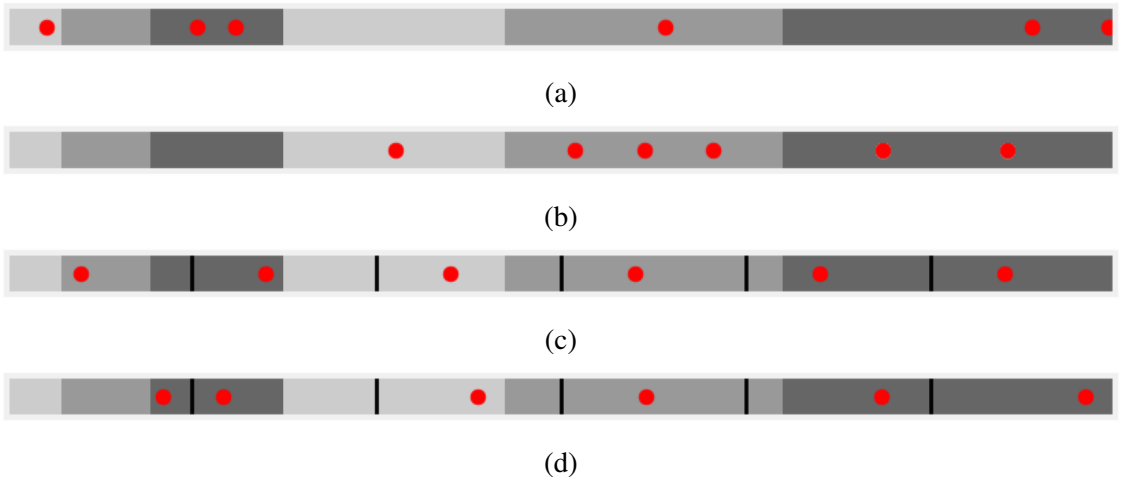


Figura 10: Exemplo dos métodos de reamostragem: (a) Multinomial (b) Residual (c) Sistemática (d) Estratificada

A figura 10 mostra um exemplo de cada um desses métodos para reamostrar 6 partículas, onde o peso de cada uma delas é $W_k = \{0,05; 0,08; 0,12; 0,2; 0,25; 0,3\}$ e na imagem para fins ilustrativos cada peso está associado a um comprimento. Por exemplo a partícula um possui um comprimento de $0,05$ enquanto que a partícula quatro é quatro vezes maior (comprimento de $0,2$). Foi ilustrado um dos infinitos casos possíveis. No caso da amostragem multinomial (Figura 10a), qualquer uma das 6 partículas pode ser escolhida para a reamostragem, mas a probabilidade de escolher cada uma está em ordem crescente; No caso de reamostragem residual (Figura 10b), a mutiplicação do N_P pelos pesos garante que as partículas 4,5 e 6 serão reamostradas uma vez e as demais serão aleatoriamente escolhidas, onde a probabilidade de cada uma é dada por $\{\frac{1}{10}, \frac{4}{25}, \frac{6}{25}, \frac{1}{15}, \frac{5}{30}, \frac{4}{15}\}$. Na reamostragem sistemática (Figura 10c), pode-se perceber a que existem 6 regiões (r_n),

em que $r_1 \supset x_k^1, x_k^2, x_k^3$; $r_2 \supset x_k^3, x_k^4$; $r_3 \supset x_k^4, x_k^5$; $r_4 \supset x_k^4$; $r_5 \supset x_k^4, x_k^5$,$r_6 \supset x_k^5, x_k^6$. É escolhido um *offset* para a escolha da primeira partícula e as demais possuem uma distância fixa de $1/6$ entre elas. Finalmente, na reamostragem estratificada é feita a mesma divisão de regiões, porém escolhe-se aleatoriamente uma partícula dentro de cada região.

Dentre os métodos apresentados a reamostragem sistemática e a estratificada funcionam melhor. A reamostragem estratificada não é tão uniforme quanto a reamostragem sistemática, mas ela faz com que as partículas que possuem uma probabilidade mais alta dentro de cada região tenham uma maior tendência de serem reamostradas.

O algoritmo descrito aqui é o algoritmo SIS (do inglês, *Sequence Importance Sampling*) seguido do algoritmo SIR (do inglês, *Sampling Importance Resampling*). Todos os passos para realizar o FP são descritos no algoritmo 3.

Algoritmo 3 Filtro de Partículas

1: $x_0^i \sim p_{x_0}$
2: $W_0^i = 1/N_P$
3: FOR k=1,2,...N
4: FOR i=1,2,...N_P
5: $x_k^i \sim p(x_k|x_{k-1}^i, u_k)$ ▷ Faz a predição
6: $W_k^i = p(z_k|x_k^i)$ ▷ Calcula os pesos conforme a medição
7: ENDFOR
8: $W_k^i = W_k^i/sum(W_k^i)$ ▷ Normaliza
9: $\hat{x_k} = sum(x_k^i)/N_P$ ▷ Calcula o estado estimado
10: $\hat{N}_{eff} = 1/sum(W_k^i)^2$
11: IF $\hat{N}_{eff} < 2N_P/3$
12: $x_k^i = REAMOSTRAGEM(W_k^i, x_k^i)$
13: $W_k^i = 1/N_P$
14: ENDIF
15: ENDFOR

5 MODELO DO ROBÔ E MODELO DE MEDIÇÃO

Este capítulo apresenta o Qbot, que é o robô utilizado neste trabalho, e descreve os dois componentes restantes para implementar os algoritmos dos filtro descritos até o momento: o modelo da cinemática do robô e o modelo de medição.

5.1 Robô Qbot

O Qbot da Quanser é composto pelo robô da iRobot Create. A Quanser integra uma placa em cima do iRobot create (Figura 11) que contém cinco sensores de infravermelho (Sharp GP2Y0A02YK), 3 sonares (Maxbotix MaxSonar-EZ0) e uma câmera (Logitech Quickcam Pro 9000) por meio do sistema Gumstix embarcado, que possui uma placa Verdex XL6P e um processador PXA270 XScale de 600 MHz. Nele roda um Linux embarcado modificado contendo o *Quanser's Rapid Control Prototyping* (QuaRC), o qual consiste em uma biblioteca em blocos para controle de robôs.

Figura 11: O robô Qbot

O conjunto de blocos utilizados contém as APIs básicas fornecidas pela iRobot Create para realizar o movimento do robô (Figura 12a) e pelo QuaRC para ler as informações dos sensores (Figura 12b). A biblioteca QuaRC presente no MATLAB permite que o usuário construa um sistema baseado em um modelo no Simulink e a partir desse modelo pode-se gerar um código, compilá-lo e transmiti-lo para o Qbot. Tanto os valores da entrada de controle do robô (velocidade da roda direita e esquerda) quanto os valores lidos dos sensores podem ser manipulados diretamente pelo Gumstix. O Gumstix também pode se comunicar com o MATLAB em um computador via Ethernet ou WiFi. Desta forma, é possível fazer o processamento de um filtro, por exemplo, no computador.

(a) (b)

Figura 12: Blocos utilizados no Simulink (a) para movimentação (b) para aquisição de dados dos sensores.

O modelo criado com o Simulink utilizado neste trabalho consiste em fornecer valores para a entrada (velocidade da roda direita e esquerda) e ler as informações proveniente dos cinco sensores de infravermelho, do magnetômetro, do *bumper* e dos *encoders*. As funções utilizadas são: *get_ir_dists* que converte a tensão obtida por cada sensor em uma distância; *read_mag* que converte os campos magnéticos no ângulo de orientação do robô; *for_kin* que retorna os valores de distância e mudança de ângulo dos *encoders*. O código 7 utilizado no bloco *Roomba Query List* retorna se houve alguma colisão (obtida pelo *bumper*) ou se houve alguma alteração na inclinação no rodízio ou em alguma das rodas; O código 19 e o 20 retornam a distância percorrida e o deslocamento angular entre o instante de tempo atual e a última vez que ele foi requisitado, respectivamente.

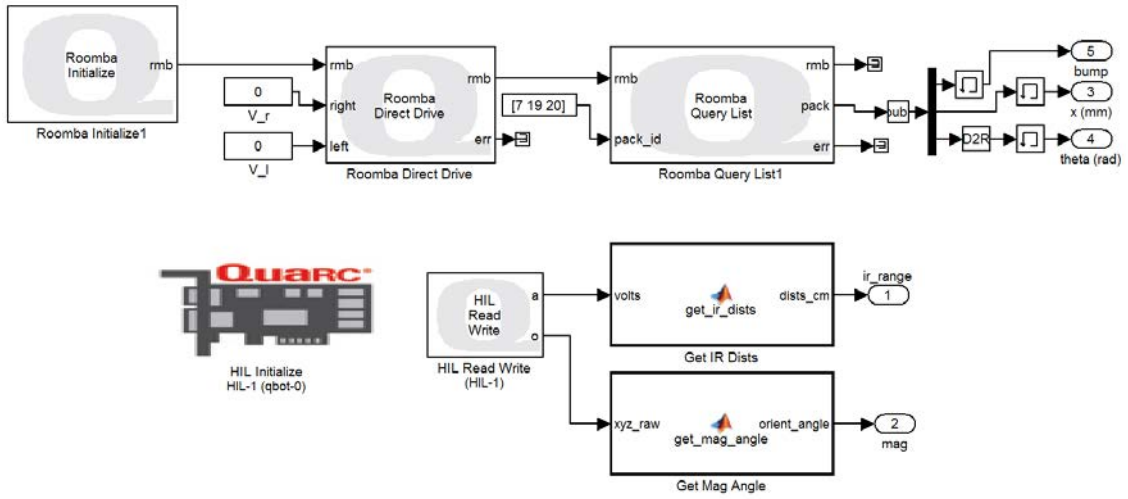


Figura 13: Modelo do Simulink utilizado para locomoção e aquisição de dados dos sensores do Qbot.

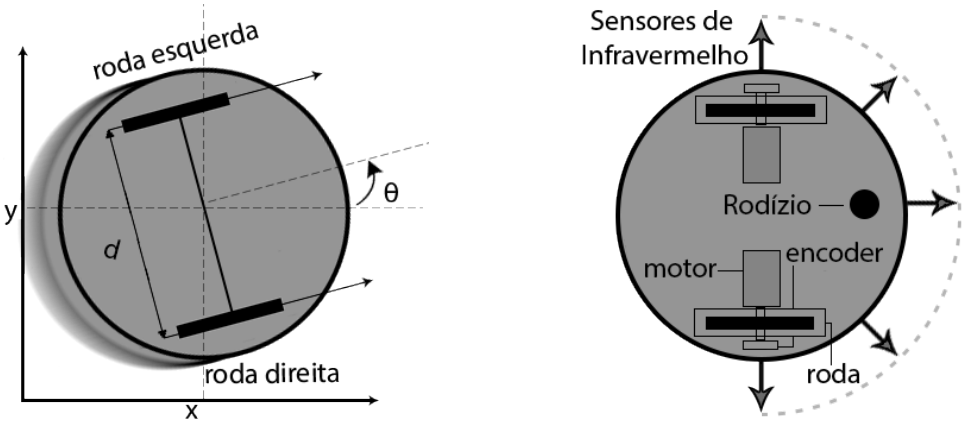
A Figura 13 mostra o modelo criado. Este modelo foi compilado e transmitido para rodar no Gumstix do Qbot. Tanto as manipulações nos valores de entrada quanto os valores lidos dos sensores foram feitos por feitos por meio de um *script*, onde os valores de entrada das rodas foram alterados utilizando a função *set_param()* e as variáveis de saída foram lidas através da função *get_param()*.

5.2 Modelo do Robô

A cinemática robótica descreve o efeito das ações de controle no movimento do robô, determinando a sua posição em cada instante. A posição é calculada a partir das mudanças que ocorrem nos sistemas de locomoção do robô (motores). O modelo cinemático não

leva em conta a inércia do robô, deformações em sua estrutura, forças oriundas do deslocamento (atrito, escorregamento, etc.), e demais fatores internos e externos que possam afetar a locomoção.

O Qbot é um robô diferencial que possui duas rodas tracionadas de forma independente com *encoders* e um rodízio giratório para a sua estabilidade. O esboço do Qbot e da sua cinemática são mostrados na Figura 14.



(a) (b)

Figura 14: Robô diferencial: (a) cinemática (b) esboço do robô utilizado.

Os detalhes do modelo que descrevem a cinemática do robô podem ser encontrados em (DUDEK; JENKIN, 2000) e (THRUN, 2001). Um modelo discreto aproximado, que ignora a dinâmica e atritos dos motores é dado pelas seguintes relações (Figura 14a):

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \begin{bmatrix} x_k + D_k \cos(\theta_k + \phi_k) \\ y_k + D_k \sin(\theta_k + \phi_k) \\ \theta_k + \phi_k \end{bmatrix} \tag{39}$$

$$D_k = \frac{V_{d,k} + V_{e,k}}{2} T \tag{40}$$

$$\phi_k = \frac{V_{d,k} - V_{e,k}}{d} T \tag{41}$$

As variáveis presentes na Equação (39), que descrevem a cinemática do robô diferencial são: x e y são as coordenadas centrais do robô e θ é a sua orientação; V_d e V_e são a entrada de controle e representam as velocidades das rodas (direita e esquerda), $\boldsymbol{u}_k = \begin{bmatrix} V_{d,k} & V_{e,k} \end{bmatrix}^\mathsf{T}$. As constantes presente em 40 e 41 são d, que é a distância entre os centros das duas rodas, e T, que é o período de amostragem.

O modelo do robô diferencial representa a forma pela qual o estado é obtido do estado anterior, onde k é o passo. O vetor de estado é expresso pela postura do robô, $\boldsymbol{x} = \begin{bmatrix} x, y, \theta \end{bmatrix}^\mathsf{T}$ e a sua distribuição é assumida como Gaussiana com uma matriz de covariância $\boldsymbol{P_k}$.

A função de transição de estado (42) move o estado atual para o próximo estado de acordo com o modelo do robô diferencial (39) acrescido de um ruído de processo imprevisível $\boldsymbol{w}_k$, que é considerado Gaussiano com média zero e covariância $\boldsymbol{Q}_k$.

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k \tag{42}$$

A matriz de covariância $\boldsymbol{Q}_k$ foi modelada assumindo duas fontes independentes de erro, translacional (D_k) e angular (ϕ_k) com as suas respectivas incertezas (σ_D^2 e σ_ϕ^2):

$$\boldsymbol{Q}_k = \begin{bmatrix} \sigma_D^2 & 0 \\ 0 & \phi_k^2 \sigma_\phi^2 \end{bmatrix} \tag{43}$$

A odometria consiste em incorporar as informações de deslocamento linear (D_k) e angular (ϕ_k) dos *enconders* no modelo. No entanto, este modelo serve apenas para calcular a posição do robô após um certo tempo, pois essas informações adquiridas dos *encoders* estão apenas disponíveis depois que o robô executou o movimento. Este modelo também é conhecido como a cinemática direta e só pode ser utilizado para localizar o robô e não pode ser utilizado para conduzir um robô até determinado ponto.

Existem algumas restrições ao tipo de deslocamento que o robô diferencial pode realizar. O robô diferencial é considerado um robô não-holonômico, que é um sistema com dimensão finita onde algum tipo de restrição é imposta a um ou mais estados do sistema. Estas limitações no caso do Qbot são impostas pela impossibilidade de deslocar sobre o seu eixo transversal.

A forma mais intuitiva de percorrer uma trajetória arbitrária (de um ponto até outro) em um determinado tempo é utilizando as velocidades linear (V_k) e angular (ω_k). No entanto, a entrada utilizada para realizar o deslocamento de um robô diferencial é composta pelas velocidades da roda direita ($V_{d,k}$) e da roda esquerda ($V_{e,k}$). Dessa forma, é necessário realizar a conversão de V_k e ω_k para $V_{d,k}$ e $V_{e,k}$. Isso pode ser feito utilizando 40 e 41, onde $D_k/T = V_k$ e $\phi_k/T = \omega_k$:

$$V_{d,k} = \frac{2V_k + \omega_k d}{2} \tag{44}$$

$$V_{e,k} = \frac{2V_k - \omega_k d}{2} \tag{45}$$

5.3 Modelo de medição

Os modelos de medição descrevem o processo de formação pelo qual as medições dos sensores são geradas no mundo físico. O objetivo principal deste trabalho é encontrar a posição do robô, logo são necessárias informações referente à distância que o robô se encontra de obstáculos. Os sensores de infravermelho, sonar e o laser *rangefinder* são alguns exemplos de sensores que podem ser utilizados para obter essa distância. Cada um desses sensores possui um princípio de funcionamento diferente do outro e este deve ser incorporado ao modelo de medição.

Quanto mais preciso o modelo do sensor melhor os resultados, no entanto é muito difícil criar um modelo perfeito de um sensor. Isso ocorre muitas vezes devido à falta de conhecimento do que ocorre com a medida obtida pelo sensor. Por exemplo, no caso da medida de distância do sensor de IV são necessárias variáveis de estado são desconhecidas, como o material da superfície em que o sinal reflete, alguma interferência externa e etc. Ao utilizar uma abordagem probabilística é possível acomodar as incertezas presente nas medidas dos sensores de uma forma não determinística (THRUN, 2001).

A função que mede a distância da posição do sensor, em uma linha reta, até o objeto mais próximo, ignorando o erro de medida, é dado pela equação de medição (46). A orientação e a posição do robô são obtidas pelo estado atual do robô $\boldsymbol{x}_k$. A variável $\boldsymbol{p} = (x_p, y_p)$ fornece a localização do objeto mais próximo detectado pelo sensor através

de uma consulta do mapa do ambiente e o estado atual do robô (sabendo a posição no mapa e a orientação é possível traçar uma reta até encontrar um obstáculo presente no mapa). A seção 6 contém maiores detalhes sobre como o mapa é construído.

$$h(\boldsymbol{x_k}, \boldsymbol{p}) = \sqrt{(x_p - x_k)^2 + (y_p - y_k)^2} \tag{46}$$

O sensor utilizado para medir distâncias neste trabalho é um sensor infravermelho (IV). A distância obtida pelo sensor de IV é calculada por um processo de triangulação, onde o transmissor, que contém um led infravermelho, e tem uma distância e e orientação fixa em relação ao receptor, que é um foto-receptor sensível à luz infravermelha. As distâncias entre e os emissores e os receptores são calculadas pelas diferenças de fase.

O robô utilizado neste trabalho contém cinco sensores de IV e estes estão localizados na parte superior formando um semi-círculo de cinco sensores, igualmente espaçados com uma diferença de $45°$), cobrindo aproximadamente um arco na frente do robô. A Figura 14b mostra a disposição de cada um desses sensores, onde o central está apontando para frente ($0°$) e os outros dois estão à direita $-45°$, $-90°$, e os outros dois estão à esquerda $45°$ e $90°$.

Cada um dos sensor retorna uma tensão referente a uma distância, então para obter a distância, em cm, foi feita uma calibração de cada um dos sensores. Essa calibração foi feita para cada sensor obtendo a média de 500 leituras do sensor para cada distância medida, onde foram medidas 11 distâncias – de 23 cm até 123 cm – com um espaçamento de 10 cm entre cada uma. A Figura 15 mostra a aproximação polinomial de quarta ordem para fazer a interpolação dos valores para um dos sensores.

A função de medição (46) calcula a distância do sensor, em linha reta, até o obstáculo mais próximo. No entanto, como são utilizados cinco sensores essa função de medição tem que ser adaptada:

$$\boldsymbol{h}_i(\boldsymbol{x}_k, \boldsymbol{p}_i) = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2} \tag{47}$$

O parâmetro de entrada de função $p_i = (x_n, y_n)$ fornece a localização do obstáculo mais próximo detectado pelo i-ésimo sensor através de uma consulta ao mapa do ambiente e uma direção com base na orientação atual do sensor. O sensor consegue medir uma distância de 20 a 123 cm. A função de medição é executada somente para os sensores que retornam um valor menor do que 123 cm. A equação (48) descreve a medição com um ruído Gaussiano de média zero para cada sensor modelado, $\boldsymbol{v}_k$.

$$\boldsymbol{z}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{p}) + \boldsymbol{v}_k \tag{48}$$

$\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{p})$ é um vetor constituído pelas cinco medidas $h_i(\boldsymbol{x}_k, \boldsymbol{p}_i)$ em (47). A matriz de covariância de medição – $\boldsymbol{R}_k$ – é uma matriz diagonal contendo a variância de cada sensor de IV. A dimensão da matriz $\boldsymbol{R}_k$ (n) pode variar a cada iteração, isto é, ela é dada pelo número de sensores de IV que estão a uma distância inferior a 123cm. Caso as medidas dos cinco sensores forem maior do que 123 cm a dimensão de $\boldsymbol{R}_k$ será zero, ou seja, não será realizada a etapa de correção nesta iteração.

Para modelar a variância da medição de cada IV, $\boldsymbol{R}_k$, foram utilizadas as mesmas 500 leituras utilizadas para a calibração. Os resultados são mostrados na Tabela 1, onde as variâncias da amostra de cada sensor, a cada uma dessas 11 distâncias, são apresentadas. Pode-se observar que as variâncias variam significativamente conforme a distância e que o sensor S1 e S2 possuem uma variância menor do que os outros três.

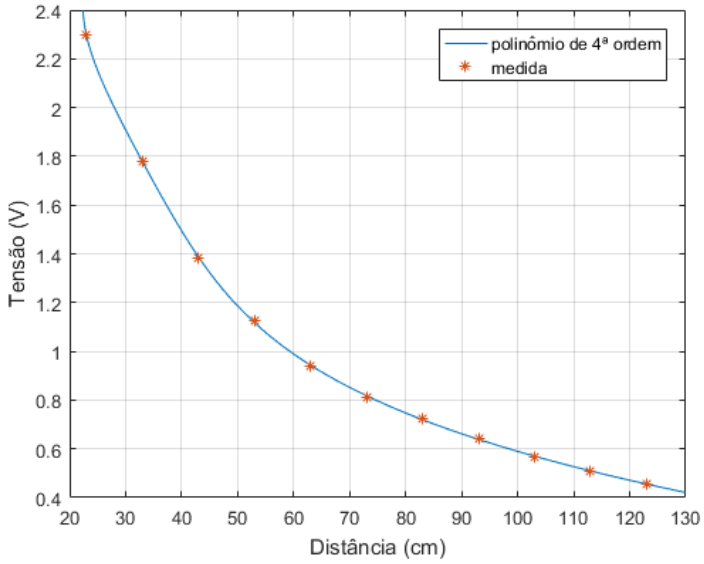


Figura 15: Calibração do sensor de IV por meio de uma aproximação polinomial de quarta ordem.

Tabela 1: Variância de cada sensor para cada uma das distâncias medida.

Distância	Variância (cm^2)				
	$\sigma^2{}_{S1}$	$\sigma^2{}_{S2}$	$\sigma^2{}_{S3}$	$\sigma^2{}_{S4}$	$\sigma^2{}_{S5}$
23 cm	0,47	0,53	0,98	0,73	1,03
33 cm	0,52	0,61	1,09	1,72	1,43
43 cm	0,96	0,67	2,59	3,17	2,82
53 cm	2,65	1,77	5,78	8,31	4,66
63 cm	4,37	2,84	12,17	11,14	9,23
73 cm	7,87	5,30	17,19	27,24	16,94
83 cm	13,60	6,54	28,52	42,26	26,36
93 cm	19,57	10,76	44,55	60,19	37,17
103 cm	21,35	12,35	65,84	94,66	63,20
113 cm	34,64	22,95	82,66	124,92	77,21
123 cm	45,54	32,18	129,01	130,72	83,36

Para ilustrar a diferença na variância em diferentes distâncias, a Figura 16 mostra um histograma das amostras a uma distância de 23 cm (em preto) e 103 cm (em azul) obtidas pelo S5. Pode-se perceber que as medições a uma distância de 23 cm são muito mais concentradas perto desse valor, no entanto ao aumentar para uma distância de 103 cm as amostras se encontram muito mais dispersas.

Essa diferença na variância conforme a distância ocorre devido ao fato da intensidade de luz refletida ser menor e deve ser incorporada no modelo de cada sensor. Para isso foi criada uma função definida por partes (em 10 segmentos). Essa função para o sensor S5 é ilustrada na na Figura 17.

Outro fator que distorce a projeção do raio IV é o ângulo entre o obstáculo e o sensor. Essa outra dependência, no entanto, requer outros meios mais sofisticados para poder ser medida com precisão. Foi observado que a variância aumenta por um fator de 0,08 a cada 1° até 70° e para medições com ângulos maiores o valor obtido pelo IV é descartado.

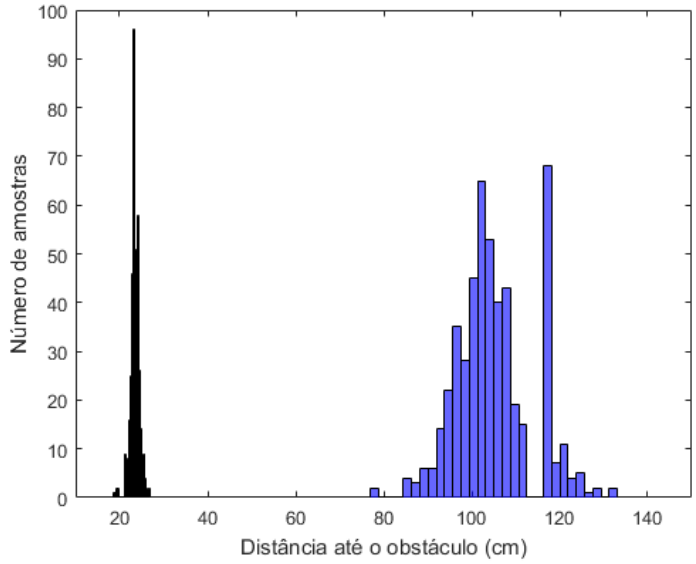


Figura 16: Histograma das medições para duas distâncias referente ao sensor S3: 23 cm em azul claro e 123 cm em azul.

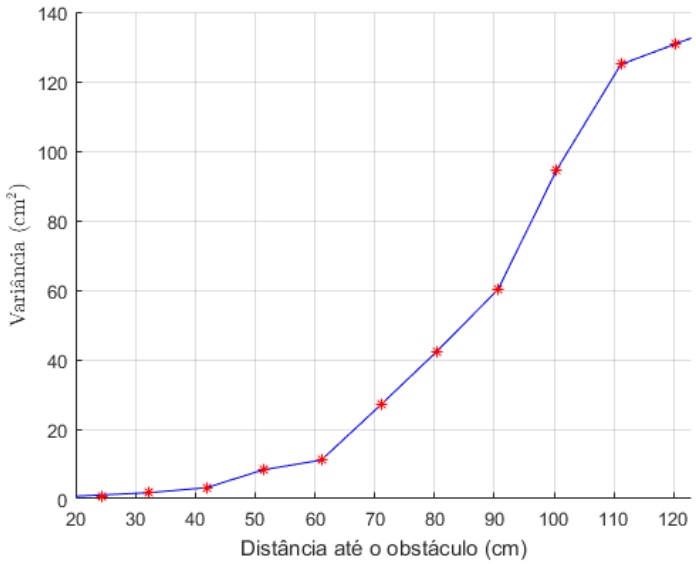


Figura 17: Função definida por partes para determinar a variância do S3.

Esse efeito também foi incorporado no modelo do sensor por meio de um fator de ajuste $\iota_{Si} = 0.08\alpha_i + 1$, se $\alpha_i \leq 70°$. Tanto as variâncias de cada sensor (quando disponíveis) quanto os ângulos são calculados a cada iteração e utilizam a estimativa da postura atual do robô.

Finalmente, a matriz de covariância $\boldsymbol{R}_k$ é modelada utilizando as variâncias (σ^2_{Si}) obtidas pelas funções definidas por partes de cada um dos sensores baseadas na tabela (1)

e multiplicadas por um fator de ajuste ι_{Si}.

$$\boldsymbol{R}_k = \begin{bmatrix} \sigma_{S1}^2 \iota_{S1} & 0 & 0 & 0 & 0 \\ 0 & \sigma_{S2}^2 \iota_{S2} & 0 & 0 & 0 \\ 0 & 0 & \sigma_{S3} \iota_{S3} & 0 & 0 \\ 0 & 0 & 0 & \sigma_{S4} \iota_{S4} & 0 \\ 0 & 0 & 0 & 0 & \sigma_{S5} \iota_{S5} \end{bmatrix} \tag{49}$$

6 MAPAS

Neste capítulo é abordado problema da localização de um robô, o qual consiste em determinar a postura do robô em relação a um mapa do ambiente. Este é um problema básico de quase todos robôs autônomos, pois eles precisam se localizar primeiro para depois poder executar as suas tarefas. Os mapas contêm as informações relativas ao ambiente e são independentes da postura do robô. Também é abordado um método para que o robô realize o planejamento de uma trajetória até um determinado ponto no mapa. Por fim, um algoritmo de seguimento de trajetória é apresentado para gerar as entradas de controle necessárias para o robô se locomover até o ponto definido.

6.1 Localização

A localização pode ser vista como um problema de transformação de coordenadas. Os mapas são descritos em um sistema de coordenadas, que é independente da postura de um robô. A localização é o processo de estabelecer uma correspondência entre esses dois sistemas de coordenadas. Utilizando as informações da postura do robô e as medidas do robô em relação ao mapa é possível encontrar a localização do robô dentro do mapa.

Existem diversos tipos de problema de localização e neste trabalho será abordado a localização de um robô com postura inicial desconhecida e com um mapa dado a priori. A forma utilizada para representar o mapa é por meio de um *grid*, ou seja divide-se o mapa em n células, onde cada uma dessas pode estar ocupada (1) ou não (0) por um obstáculo.

A Figura 18 mostra um mapa *grid*, onde as regiões em cinza marcam as delimitações (paredes) de uma sala e em branco são as regiões desocupadas. A célula em azul representa a posição do robô e as setas representam as direções que os dois sensores estão apontando. O sensor horizontal está a uma distância de cinco unidades, isso significa que o robô pode estar em na terceira coluna e pode estar na segunda, terceira, quarta ou quinta linha. Utilizando a medida de duas unidades do sensor vertical é possível determinar que a posição do robô é $(4, 3)$.

Este é um exemplo do que se faz para localizar um robô, no entanto ao localizar o robô não se sabe a posição inicial e nem a sua orientação, ou seja, a distância retornada pode corresponder a mais do que uma possível localização do robô no mapa. Um erro na posição atual e/ou na sua orientação podem causar uma diferença substancial.

6.2 Planejamento de rota

O planejamento consiste em encontrar uma sequência de ações que leva o robô de um ponto inicial até a meta. Um caminho é ótimo se a soma de seus custos de transição

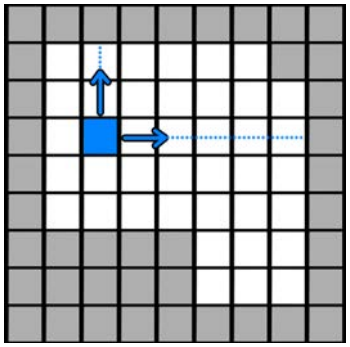


Figura 18: Exemplo de mapa *grid*, onde células em cinza mostram uma região ocupada e celúlas em branco mostram uma região desocupada.

(custos de borda) for mínima em todos os caminhos possíveis que vão desde a posição inicial até a posição do objetivo. Um algoritmo de planejamento é completo se ele sempre encontra um caminho em tempo finito quando existe um e informa em tempo finito se nenhum existe. Da mesma forma, um algoritmo de planejamento é ótimo se ele sempre encontra um caminho ideal.

Existem várias técnicas para calcular o caminho dada uma representação do ambiente. As técnicas mais populares são por meio de algoritmos determinísticos (DIJKSTRA, 1959), determinísticos baseados em heurísticas (HART; NILSSON; RAPHAEL (1968), NILSSON (1980)) e algoritmos aleatorizados (KAVRAKI et al. (1996), LAVALLE; JR. (2001)). No caso de encontrar o caminho mais curto para o problema apresentado, os algoritmos determinísticos são o suficiente.

Planejar a rota do robô de forma eficiente consiste em determinar o caminho mais curto da posição que o robô se encontra até uma posição final definida da forma mais rápida possível. A Figura 19 mostra um exemplo de situação onde se quer saber a trajetória de um ponto inicial (em azul) até um ponto objetivo (em roxo).

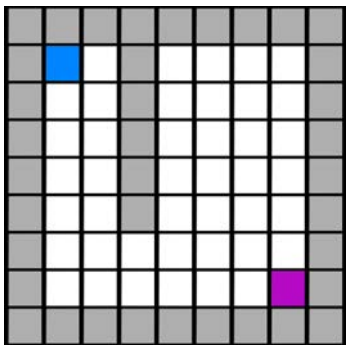


Figura 19: Exemplo de um mapa com a posição inicial (azul) e a meta (roxo) do robô.

Uma técnica comum utilizada para planejar rotas na robótica consiste em representar o ambiente como um grafo $G = (S, E)$, onde S é o conjunto de possíveis locais (nós) e E é um conjunto de arestas que representam transições entre esses locais. Existem vários algoritmos que solucionam o problema do caminho mais curto num grafo; os dois mais populares e que retornam um caminho ótimo são o algoritmo de Dijkstra (DIJKSTRA, 1959) e o A* (HART; NILSSON; RAPHAEL, 1968).

O algoritmo A* funciona da seguinte forma: O planejamento do caminho parte de um nó inicial $s_0 \in S$ para um objetivo $s_x \in S$, onde S é o conjunto finito de nós. Para isso,

ele armazena $g(s)$, que é o custo do caminho do nó inicial para cada nó s. Inicialmente, $g(s) = \infty$ para todos os nós $s \in S$. O algoritmo começa a atualizar o custo do nó inicial zero e, em seguida, coloca esse nó em uma fila de prioridade conhecida como uma lista aberta. Cada elemento dessa fila é ordenado de acordo com a soma de seu custo de caminho atual desde o início, $g(s)$, e uma estimativa heurística de seu custo do caminho até a meta, $h(s, s_x)$. O nó com a menor soma fica na frente da fila de prioridade.

O algoritmo então coloca o nó s na frente da fila e atualiza o custo de todos os nós acessíveis a partir deste nó. Se o custo do nó $s, g(s)$, mais o custo da aresta entre s e um nó vizinho $s', c(s, s')$, é menor que o custo atual s', então o custo de s' é atribuído como o de valor mais baixo. Se o vizinho s' for alterado, ele será colocado na lista aberta. O algoritmo continua até chegar no objetivo. Neste estágio, se a heurística é admissível, isto é, garantida para não superestimar o custo de trajetória de qualquer nó para a meta, então o custo de trajetória de s_x é garantido como ótimo. O algoritmo de Dijkstra possui o mesmo funcionamento que o A*. No entanto, o Dijkstra é um caso especial onde a função heurística é $h(x, y) = 0$.

A Figura 20a ilustra quantas vezes o nó foi expandido utilizando o algoritmo com as possíveis movimentações sendo para cima, baixo, esquerda ou direita sem a utilização de uma função de heurística. Estes números não informam o custo e sim o número de expansões totais para chegar ao objetivo.

Utilizando uma função heurística é possível diminuir o número de expansões. Esta se puder ser definida tem que ser menor ou igual que a distância do ponto até o objetivo. Neste caso ela pode ser definida e ela representa a distância do objetivo até a posição inicial do robô sem considerar os obstáculos. Dessa forma ao calcular o custo utilizando $h(x, y) = x+y-16$, onde x e y representam a linha e a coluna da célula, respectivamente, que para fins de visualização resulta na Tabela 2.

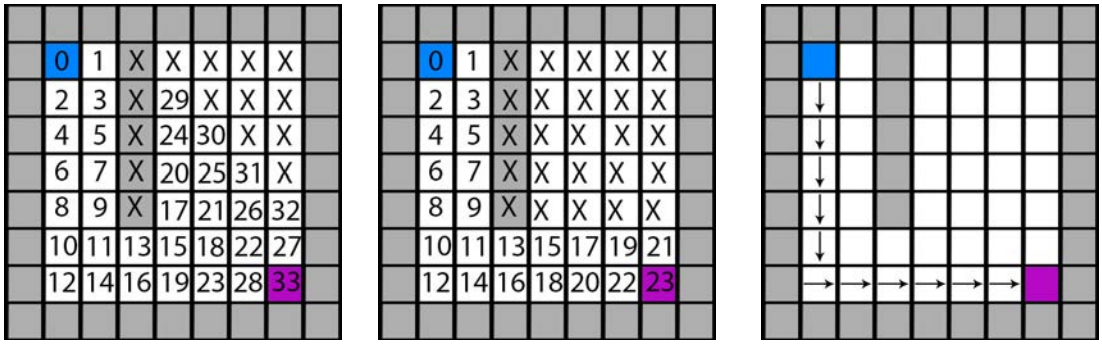
Com essa função definida é possível a partir da décima quinta expansão verificar qual é o melhor trajeto a seguir. Por exemplo, na célula (7,5) foram realizados 7 movimentos. A sua expansão é para a célula superior, inferior e à direita e ao verificar os valores da heurística (5,3,3), percebe-se que o movimento para cima aumenta o custo em relação ao objetivo (sendo que o número de movimentos é o mesmo, 8), portanto a de maior peso é descartada.

A Figura 20b mostra o número de expansões do nó utilizando o algoritmo A* com a heurística apresentada. O número de movimentos totais executados para fazer ambas as expansões foi de 12 e a trajetória feita pode ser obtida fazendo o caminho reverso e verificando qual movimentação foi feita para chegar naquela célula (ilustrado na Figura 20c.

12	11	10	9	8	7	6
11	10	9	8	7	6	5
10	9	8	7	6	5	4
9	8	7	6	5	4	3
8	7	6	5	4	3	2
7	6	5	4	3	2	1
6	5	4	3	2	1	0

Tabela 2: Valores da função heurística $h(x, y)$

Os quatro únicos possíveis movimentos utilizados no algoritmo fazem com que o robô só execute uma rotação de 90° em sentido horário ou anti-horário e um movimento em



(a) (b) (c)

Figura 20: Número de expansões do nó utilizando: (a) algoritmo de Dijkstra (b) algoritmo A*; reconstrução dos comandos envolvidos para ir do ponto inicial até o final (c).

linha reta. Essa não é a melhor forma do robô executar um movimento, pois ela faz com que o robô pare e execute um giro toda vez que é necessário mudar de direção. Existem diversas formas de suavizar a trajetória (a mais comum é pelo método do gradiente), fazendo com que robô percorra a trajetória sem a necessidade de realizar paradas. No entanto, o algoritmo *pure-pursuit*, que é utilizado neste trabalho, não requer que a trajetória seja previamente suavizada.

Existem vários métodos de seguimento de trajetória na teoria de controle não-linear, controle preditivo, linearização do modelo cinemático, métodos geométricos, etc. Os controladores de seguimento de trajetória têm como objetivo obter as leis de controle que permitam que o robô siga a rota estabelecida da forma mais aproximada possível. Dentro dos métodos geométricos, o mais utilizado devido a sua simplicidade e sua eficácia é o *pure-pursuit* (ANDERSEN et al., 2016).

6.3 Seguimento de rota utilizando o *Pure-Pursuit*

O *pure-pursuit* é um método utilizado para realizar o seguimento de uma trajetória. O ponto chave do algoritmo é escolher um ponto situado no trajeto que está a uma distância L_{ah} (do inglês, *lookahead distance*) da posição atual do veículo. Calculando, dessa maneira, a curvatura necessária para mover o veículo de sua posição atual até o próximo ponto e, eventualmente, chegar ao ponto de meta. Este método possui esse nome devido a uma analogia que é feita quando se dirige um carro: para chegar a um certo local, no campo de visão, costuma-se olhar para um ponto no caminho a uma certa distância a frente, dessa forma é possível realizar curvas e desviar de eventuais obstáculos.

O algoritmo *pure-pursuit* (COULTER, 1992) tem como objetivo fazer que o robô siga uma trajetória especificada que garanta um erro pequeno na posição e na orientação. Para isso, o robô utiliza a sua localização atual e o um conjunto de pontos que compõem a trajetória que deve ser realizada pelo robô. A cada instante de tempo é encontrado o ponto mais próximo do robô (no sentido da trajetória) a uma distância L_{ah}. Em seguida, é calculado o raio do arco que deve ser seguido para ir da posição atual até esse ponto encontrado. Após o robô percorrer essa distância o algoritmo é repetido até encontrar o ponto de meta. Ao encontrar o ponto final é definido um limiar que deve ser satisfeito entre a distância desse ponto e o ponto o qual o robô se encontra.

A Figura 21 ilustra todos os elementos envolvidos no cálculo da curvatura ,$\gamma = \frac{1}{R}$, no método do *pure-pursuit*, onde o ponto de meta é (g_x, g_y), a localização atual do robô é

dada por (r_x, r_y), o ângulo entre a posição do robô e o ponto de meta α. O raio, R, pode ser obtido utilizando a lei do seno, conforme (50):

$$\begin{aligned} \frac{R}{\pi/2 - \sin(\alpha)} &= \frac{L_{Ah}}{\sin(2\alpha)} \\ \frac{R}{\cos(\alpha)} &= \frac{L_{Ah}}{2\sin(\alpha)\cos(\alpha)} \\ R &= \frac{L_{Ah}}{2\sin(\alpha)} \end{aligned} \tag{50}$$

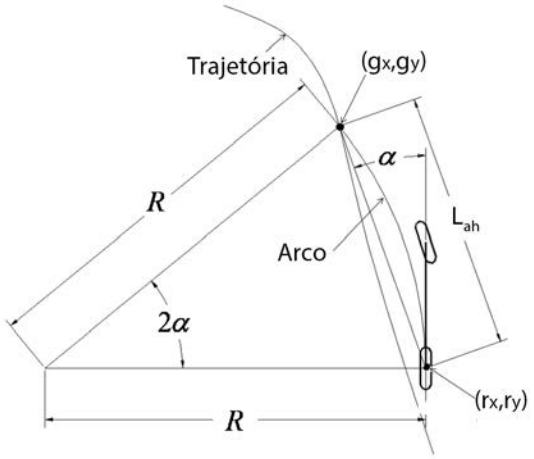


Figura 21: Geometria do método *pure-pursuit*. Fonte: Adaptado de SNIDER (2009).

Defini-se e_{Ah} como a distância lateral (o erro) entre o robô e o ponto de meta pode-se reescrever a lei de controle conforme 51. Dessa forma, o *pure-pursuit* pode ser visto como um controlador proporcional, onde e_{Ah} é o erro e $2/L_{Ah}$ representa o ganho. Como pode ser observado, este método conta somente com um parâmetro de controle, a distância L_{Ah}. Por esse motivo o algoritmo pode ser facilmente implementado e configurado.

$$\begin{aligned} \sin(\alpha) &= \frac{e_{Ah}}{L_{Ah}} \\ \gamma &= e_{Ah}\frac{2}{L_{Ah}^2} \end{aligned} \tag{51}$$

Os efeitos de mudar a distância L_{Ah} devem ser considerados dentro do contexto de dois problemas: No caso do robô precisar recuperar-se em um determinado ponto no caminho e no caso do veículo estar no caminho e querer permanecer no caminho. No primeiro caso é almejado que o veículo mesmo estando a uma distância considerável do caminho, ele tenha condições de retornar e se manter nele durante o percurso. Neste caso uma L_{Ah} grande tende a convergir para o caminho gradualmente e com menos oscilações. Por outro lado, uma L_{Ah} pequena pode convergir para o caminho de forma mais rápida, porém pode gerar oscilações.

No outro problema, quanto maior for L_{Ah}, menor será a curvatura do caminho que será seguido. Se o caminho entre o veículo e o ponto de meta é suficientemente curvo, então não há nenhum único arco que una os dois pontos, ou seja, qualquer arco gerado induzirá um erro.

A figura 22 ilustra o impacto da escolha de 7 valores de L_{Ah} diferentes para uma pequena trajetória. Tanto neste exemplo como no trabalho será assumida uma velocidade linear constante de 20 cm/s. Dessa forma, a única entrada de controle que deve ser calculada é a velocidade angular necessária para mover o robô até determinado ponto.

Pode-se perceber que para valores de L_{Ah} maiores do que 1 o veículo chega na trajetória de forma mais rápida, no entanto ele possui um erro lateral maior do que nos outros casos. A escolha de L_{Ah} menores fazem com que o robô se aproxime mais da trajetória, no entanto um valor baixo como o caso de $L_{Ah} = 0,01$ fez com que ocorressem oscilações durante a trajetória e no final fosse feita uma pequena volta desnecessária.

Portanto, o parâmetro L_{Ah} deve ser testado e escolhido por meio de testes (simulações e/ou experimentos), pois dependendo da velocidade e o mapa envolvidos pode ser que a trajetória executada cause colisões com obstáculos ou faça um caminho maior e desnecessário.

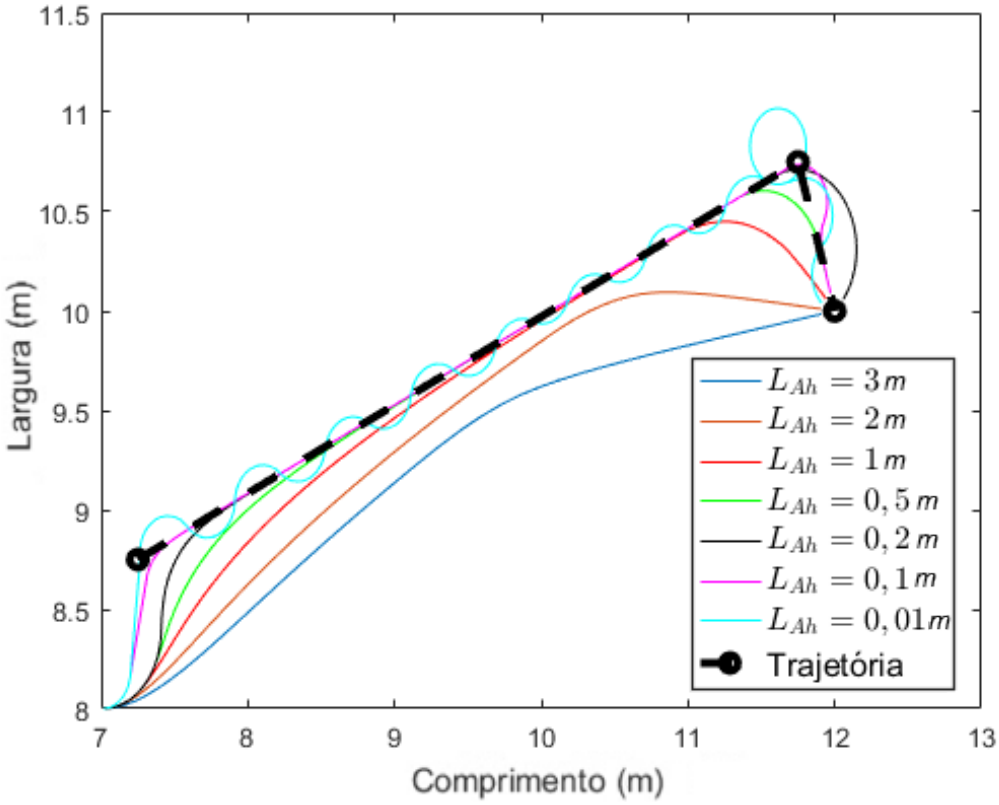


Figura 22: Algoritmo *pure-pursuit* para diferentes valores de L_{Ah}

7 EXPERIMENTOS E RESULTADOS

Neste capítulo são apresentados os dois experimentos realizados com o Qbot. No primeiro experimento foram fornecidas as entradas de controle para que o robô executasse duas trajetórias diferentes. Estas duas trajetórias percorridas pelo robô foram gravadas por uma câmera e todos os dados obtidos pelos sensores no decorrer das duas trajetórias foram armazenados. A gravação feita foi utilizada para gerar o *ground truth*, o qual fornece uma aproximação da posição real do robô durante todos os experimentos. Os dados dos *encoders* e dos sensores de IV das duas trajetórias foram injetados nos três filtros para obter a estimativa da posição do robô. Por fim, o desempenho de cada um dos filtros foi comparado com o GT.

Um segundo experimento foi realizado onde o robô faz o planejamento e a execução de duas trajetórias. Neste experimento o robô foi posicionado em dois locais diferente no mapa. O robô teve que planejar o caminho utilizando o mapa do ambiente, o ponto objetivo fornecido e a estimativa da sua posição inicial. Após encontrar o caminho que deve ser percorrido, o robô calculou as entradas de controle necessárias utilizando o *pure-pursuit* a cada iteração para chegar até o ponto objetivo e a estimativa da sua posição durante todo o trajeto foi realizada utilizando o FKU.

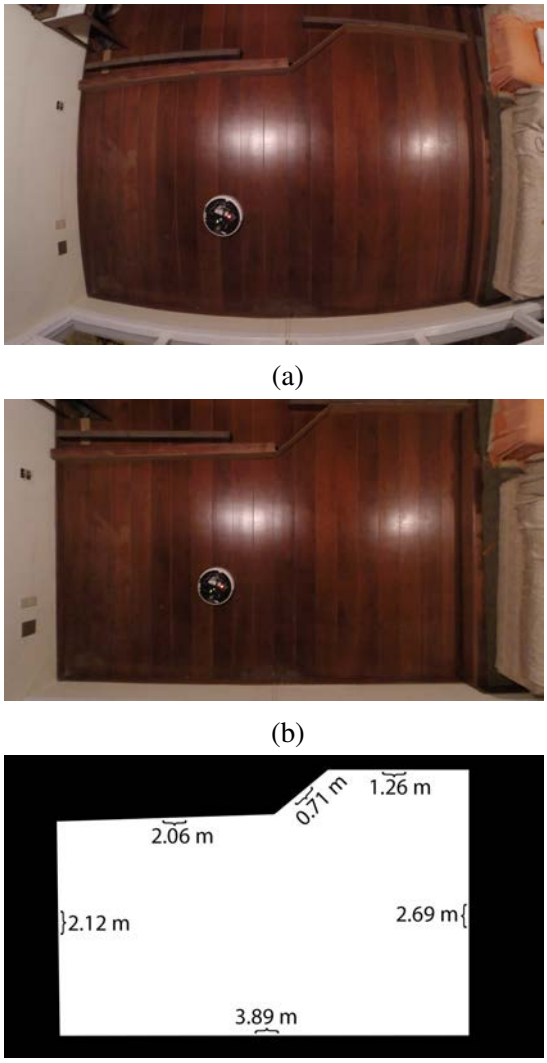
7.1 Obtenção do *ground truth* e detalhes do experimento

Para poder realizar a comparação entre o desempenho dos filtros é necessário saber o caminho exato que o robô percorreu. Para isso a trajetória do Qbot foi gravada utilizando uma câmera. O mapa do ambiente e a postura do robô foram extraídas utilizando técnicas de processamento de imagem. Dessa forma, foi possível calcular o erro de estimação dos filtros em cada instante de tempo.

O primeiro experimento foi realizado em um sala com piso de madeira, cujas dimensões são: 2,60 m de largura e 3,89 m de comprimento. A parte superior da sala foi adaptada utilizando uma madeira com papelão para ter um formato trapezoidal. A largura foi limitada em 2,6 m para encaixar no campo de visão de câmera. A Figura 23a mostra uma foto da sala onde foram realizados os experimentos.

Como pode ser observado a imagem da Figura 23a está distorcida. A distorção presente é chamada de distorção radial, que ocorre devido à lente ”olho de peixe” presente na câmera utilizada. Optou-se por uma câmera com este tipo de lente, pois ela fornece um campo de visão mais amplo.

Percebe-se que a distorção aumenta à medida que se afasta do centro da imagem, fazendo com que os objetos mais afastados do centro modifiquem a sua forma em comparação com o que ocorre na realidade. Existe outro tipo de distorção presente chamada de distorção de translação que deriva do fato da lente não estar perfeitamente alinhada com



(a)

(b)

(c)

Figura 23: Sala utilizada nos experimentos: (a) imagem obtida da câmera (b) imagem com a remoção da distorção (c) extração do mapa

o sensor de imagem. Maiores detalhes sobre os tipos de distorção podem ser encontrados em (HARTLEY; ZISSERMAN, 2003).

Para obter medidas quantitativas da imagem é necessário remover as distorções presentes. Isso pode ser feito por meio de uma calibração. A calibração consiste em obter os parâmetros intrínsicos e extrínsecos da câmera. Ela pode ser feita por meio do *software* da própria câmera, pelo MATLAB utilizando a *toolbox* de visão computacional ou utilizando o OpenCV. O resultado obtido por qualquer uma das opções é o mesmo se a calibração for feita de forma correta. O resultado obtido pode ser visto na Figura 23b.

O *software* da câmera realiza um um corte automático na imagem para que não haja *pixels* ausentes, graças a isso pode-se perceber uma perda de informações ao redor das bordas. Já ao utilizar o OpenCV ou o MATLAB é possível ajustar um parâmetro de corte: resultando em áreas pretas onde não há informações da imagem original. Isso pode vir a ser útil em casos onde há informações relevantes na periferia da imagem.

No entanto, após testar a remoção da distorção feita pelo próprio *software* da câmera foi observado que a imagem continha todas as informações necessárias para extrair o

mapa da sala. Essa alternativa foi utilizada pelo fato de ser a forma mais fácil de ser reproduzida.

Após remover a distorção na imagem é possível extrair o mapa. Para detectar as bordas da imagem foi utilizado o algoritmo de detecção de borda de Canny e dilatações. Após isso as linhas foram reconstruídas manualmente resultando na Figura 23c, que é o mapa utilizado durante os experimentos com as dimensões especificadas. Esse mapa pode ser utilizado de duas formas: utilizando os *pixels* da própria imagem ou por meio de um polígono definido por seis retas.

As duas abordagens são válidas, no entanto na hora de calcular a distância da posição do robô até uma borda do mapa é necessário traçar uma reta partindo da posição atual do robô com orientação no sensor de interesse. Após isso é necessário verificar onde houve a intersecção do ponto. Utilizando o método baseado no mapa de *grid* é necessário percorrer *pixel* por *pixel* e utilizando o outro método pode-se calcular a intersecção das retas de uma maneira mais rápida.

O Qbot está presente nas figuras contendo o mapa do ambiente. Ele possui três LEDs verdes que podem ser utilizados para determinar a sua postura no ambiente. Para fazer isso pode-se utilizar uma técnica de segmentação de cor. Utiliza-se neste trabalho o modelo de cor HSV, que baseia-se na matiz (H, do inglês *Hue*), saturação (S, do inglês *Saturation*) e o valor ou brilho (V, do inglês *Value*). Assim, pode-se definir um intervalo de valores para cada componente H, S e V de forma que englobe somente a cor verde de interesse presente na imagem.

O robô presente nas figuras anteriores é mostrado em detalhes (por meio de um *zoom*) na Figura 24a. Aplicando a segmentação da cor verde na Figura 24a resulta na Figura 24b. O Qbot possui três LEDs verdes: dois estão alinhados horizontalmente, que são considerados como um LED maior; o outro LED está a dois centímetros de distância do centro do robô (d_c). A distância entre o centroide do LED maior e do centroide menor é de 4.05cm (d_p) e o ângulo entre eles (θ) é de $36°$. Usando essas informações é possível encontrar o centro do robô e sua orientação em cada *frame* do vídeo. A postura em cada *frame* do vídeo é armazenada para ser usada como o GT.

(a) (b) (c)

Figura 24: Segmentação de cor: (a) imagem do robô (b) segmentação da cor verde (c) método para calcular a postura do robô.

As trajetórias feitas nos experimentos são mostradas na Figura 25, onde a posição inicial do robô é marcada com um quadrado verde claro, o caminho com uma linha contínua vermelha e a posição final com um círculo verde escuro. As posições do robô nas trajetórias 25a e 25b foram extraídas utilizando a técnica de segmentação de cor.

As configurações de aquisição da câmera são: cadência de 60 fps com uma resolução de 1080 x 1920 *pixels*. A velocidade máxima utilizada nos experimentos foi de 20 cm/s,

portanto a posição calculada do robô utilizando a imagem pode ter um erro de no máximo 0,33 cm (ou 1,23 *pixels*). Durante as trajetórias os sensores do robô (*encoders*, IV, magnetômetro) amostram os dados a cada 50 ms. Antes do robô começar a se movimentar foram recolhidas 500 amostras para que a sua média sirva como uma estimativa inicial. Os dados do magnetômetro somente foram utilizados nesta etapa para indicar a orientação inicial aproximada do robô e não foram utilizados na etapa de correção.

(a) (b)

Figura 25: Primeiro experimento realizado pelo Qbot: (a) primeira trajetória experimento (b) segunda trajetória

Ao iniciar as duas trajetória as distâncias dos 3 sensores localizados no meio e na parte lateral esquerda do robô retornam uma distância maior do que 123 cm, portanto somente os dois sensores localizados a direita do robô são utilizados neste momento. Como pode ser observado, as distâncias desses dois sensores são relativas à parede inferior e utilizando essas informações e a informação do mapa do ambiente é possível encontrar a sua localização no eixo y. No entanto, não é possível obter nenhuma informação em relação ao eixo x.

A Figura 26 e a Figura 28 ilustram a quantidade de sensores disponíveis (que retornam uma medida com menos de 123 cm) e a quais coordenadas se tem informação durante as trajetórias. A cor verde indica que o sensor ou os sensores disponíveis só possuem informações referente à coordenada y, ou seja, neste momento os únicos sensores, que retornam uma distância válida, estão apontados para a parede inferior ou superior. De forma similar, a cor azul indica que as medidas dos sensores disponíveis são somente sobre a coordenada x (laterais do ambiente). A cor vermelha indica que os sensores, que estão dentro do seu raio de operação, estão apontados para duas paredes ortogonais e nesse caso é possível extrair informações sobre as duas coordenadas. A cor preta mostra que todas as medidas são maiores do que 123 cm, ou seja, nenhuma das medidas é válida. Finalmente, o gráfico de setores localizado no lado direito das figuras indica quanto tempo a informação sobre as coordenadas ficaram disponível durante a trajetória.

De forma complementar, a Figura 27 mostra o número de sensores que estavam dentro do raio de operação durante a primeira trajetória. Onde cada gráfico de setor representa um segmento da trajetória apresentado na Figura 26. Por exemplo, durante o primeiro percurso existem 6 segmentos (iniciando da parte inferior a esquerda): verde, vermelho, azul, preto, verde e vermelho. O primeiro segmento (verde) da Figura 26 fornece somente informações sobre a coordenada y, denotado pela cor verde. O gráfico de setor da Figura 27 mostra que durante esse primeiro segmento de trajetória (em verde) somente dois dos cinco sensores estavam disponíveis. O terceiro segmento (em azul) fornece informações

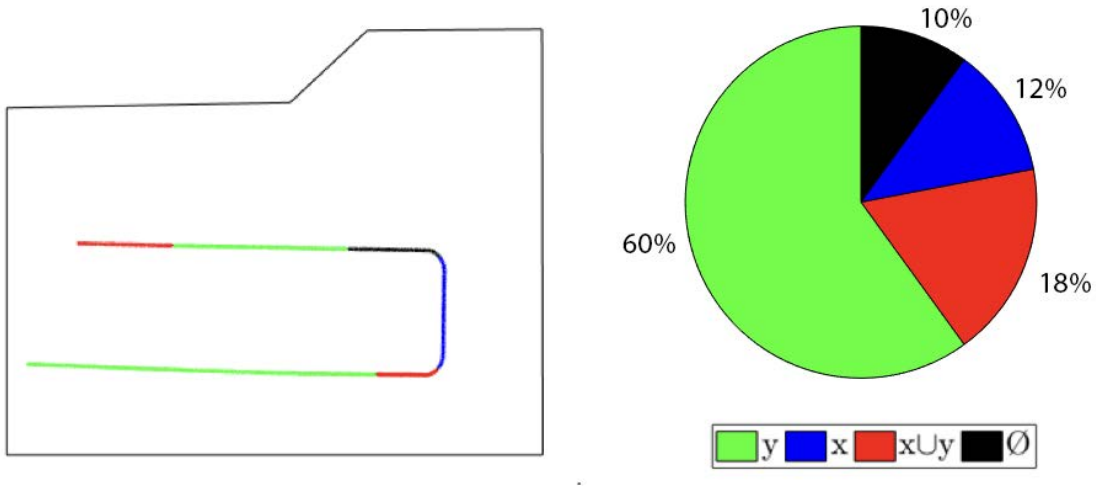


Figura 26: Sensores no campo de operação durante o experimento 1. As cores indicam a qual coordenada os sensores fornecem a informação.

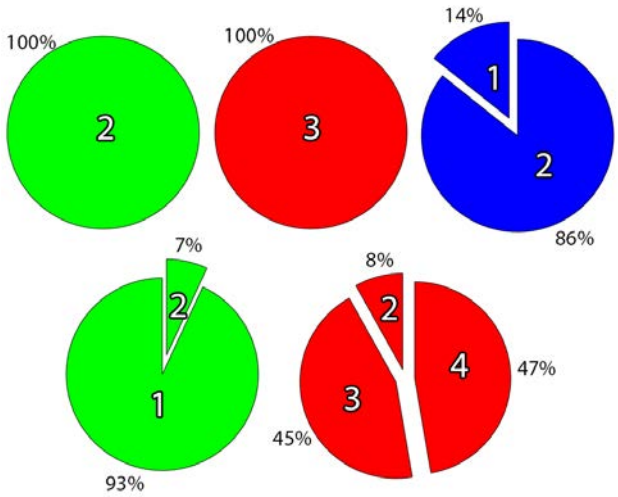


Figura 27: O gráfico de setores mostra o número de sensores que estavam dentro de sua distância operacional em cada um dos segmentos da primeira trajetória.

da componente x e o terceiro gráfico de setor da Figura 27 mostra que 86% das amostras de distâncias obtidas são provenientes de 2 sensores de IV enquanto que o restante das amostras (14%) são provenientes de apenas um sensor de IV.

Durante esta primeira trajetória mais de 92% das amostras de distância são provenientes de apenas 1 ou 2 sensores. Ou seja, a etapa de correção de cada um dos filtros na maior parte do tempo foi feita por no máximo dois sensores.

As informações referentes aos sensores do segundo experimento são mostradas na Figura 28 e na Figura 29. Nesta trajetória o robô executa uma trajetória em curva que dificulta o processo de localização, visto que a não linearidade do movimento está presente durante toda a trajetória. Neste segundo trajeto, pode-se perceber que existe uma informação limitada até o início da terceira curva (primeiro segmento vermelho). Nesta trajetória pode-se observar que existe um contato maior com duas paredes ortogonais ao mesmo tempo e que durante este percurso mais de 25 % das amostras de distâncias foram obtidas por pelo menos três sensores de IV.

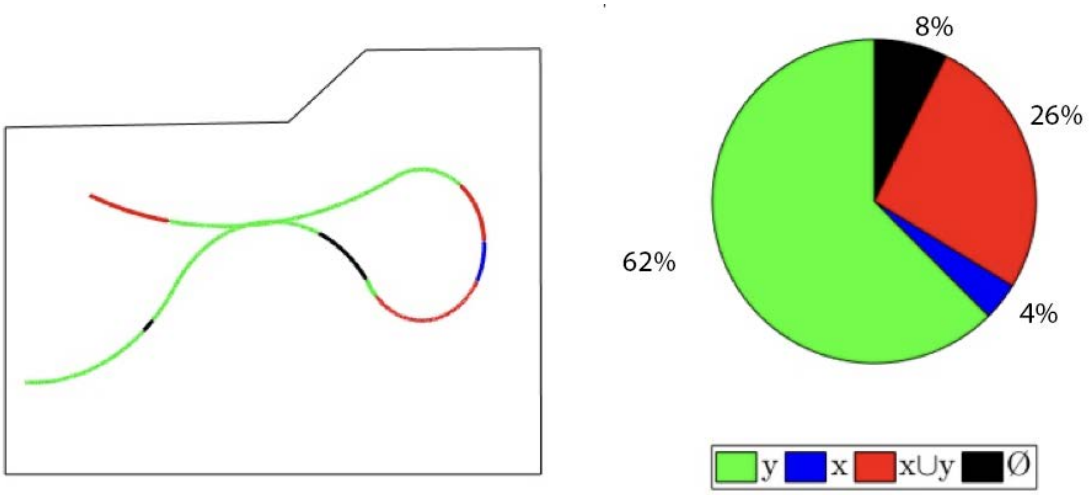


Figura 28: Sensores no campo de operação durante o experimento 2. As cores indicam a qual coordenada o sensor fornece a informação.

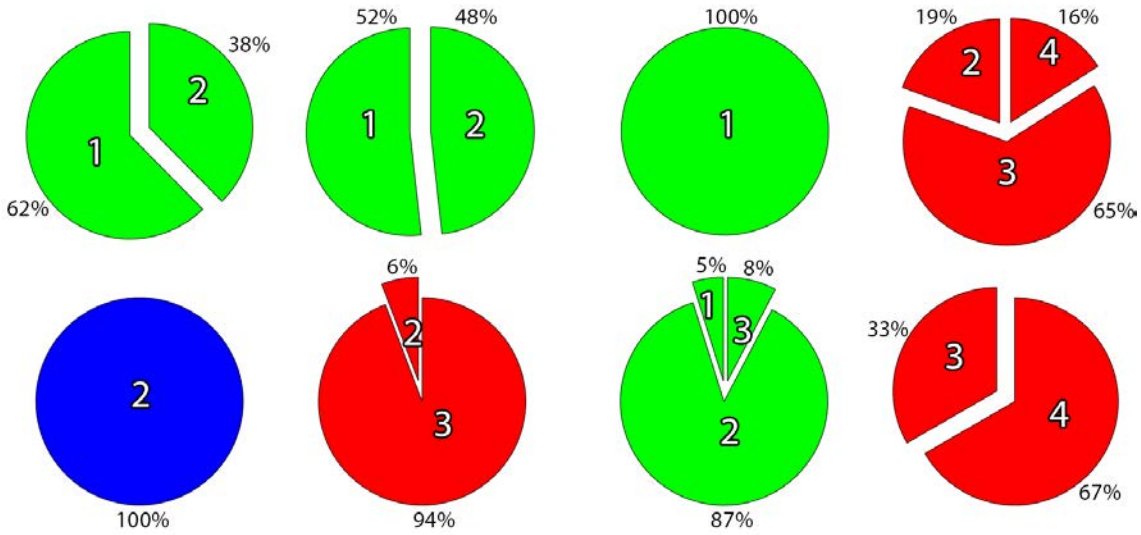


Figura 29: O gráfico de setores mostra o número de sensores que estavam dentro de sua distância operacional em cada um dos segmentos da segunda trajetória.

7.2 Comparação de desempenho dos filtros para localização do robô

O modelo do robô e dos sensores de IV utilizados foram apresentados no capítulo 5. Ambos os modelos assumem um ruído gaussiano aditivo branco com média zero e covariância $\boldsymbol{Q}_k$ e $\boldsymbol{R}_k$. As matrizes de covariância $\boldsymbol{Q}_k$ e $\boldsymbol{R}_k$ são apresentadas em (43) e (49), respectivamente. As variâncias (σ_D^2 e σ_ϕ^2) foram obtidas por meio de experimentos do movimento do robô em linha reta e executando giros. Com os modelos definidos basta utilizar o algoritmo de cada filtro.

O primeiro filtro, o FKE, consiste em realizar a linearização do modelo do movimento e do modelo de medição. Durante o movimento do robô e durante a medição dos sensores existe uma fonte de incerteza ao mover do passo k para $k+1$ relacionado à postura do robô. Logo é necessário calcular as Jacobianas da função de transição (39) e da função de medição (47) em torno de $\boldsymbol{x}_k$. Utilizando 25a e 25b resulta em:

$$\boldsymbol{F} = \nabla f_x = \begin{bmatrix} 1 & 0 & -D(k)\sin(\theta(k)+\phi(k)) \\ 0 & 1 & D(k)\cos(\theta(k)+\phi(k)) \\ 0 & 0 & 1 \end{bmatrix} \tag{52}$$

$$\boldsymbol{H} = \nabla h_i(\boldsymbol{x}_{(k+1)}, \boldsymbol{p}_i) = \begin{bmatrix} \frac{x(k+1)-x_i}{\sqrt{(x_i-x(k+1))^2+(y_i-y(k+1))^2}} \\ \frac{y(k+1)-y_i}{\sqrt{(x_i-x(k+1))^2+(y_i-y(k+1))^2}} \\ 0 \end{bmatrix} \tag{53}$$

Obtidas as Jacobianas $\boldsymbol{F}$ e $\boldsymbol{H}$ é possível utilizar o Algoritmo 1 para realizar a estimação de estados do robô. Para realizar a estimativa dos estados utilizando o FKU, basta utilizar o Algoritmo 2. Os parâmetros utilizados para calcular os pontos sigma foram: $\alpha = 0.001$, $\beta = 2$ e $\kappa = 0$.

O FP, por sua vez, utiliza n partículas, onde cada partícula representa uma possível posição para o robô. Ao executar a predição ($x_k^i \sim p(x_k|x_{k-1}^i, u_k)$), cada partícula faz um deslocamento de acordo com a entrada de controle mais o ruído presente no movimento. A correção ($W_k^i = p(z_k|x_k^i)$) simplesmente calcula os pesos que indicam o quão próxima cada partícula está da medida. Para este experimento foram utilizadas 1000 partículas e o método de reamostragem utilizado foi o estratificado.

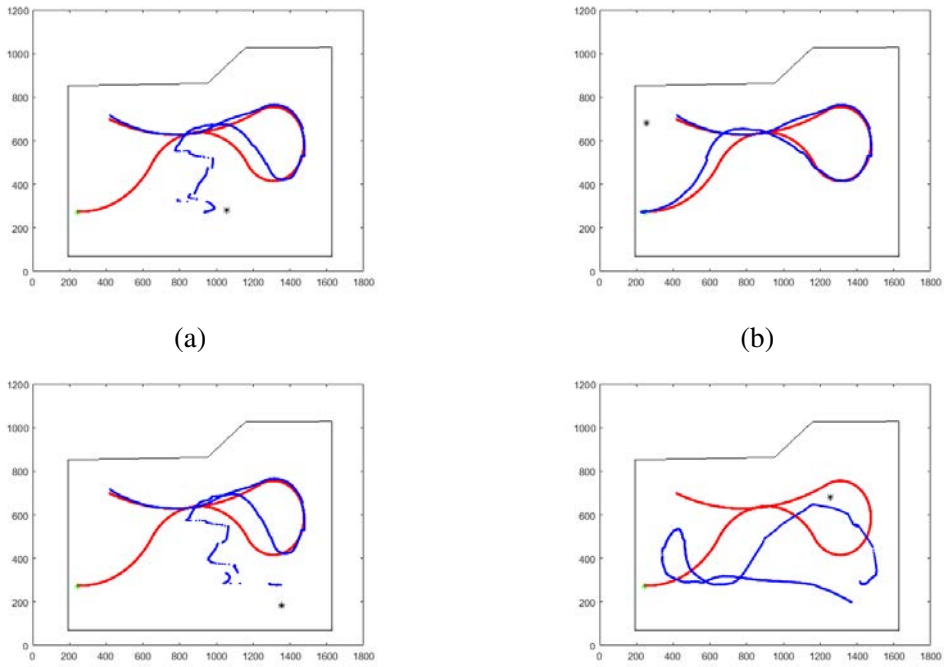
Os dados coletados dos *encoders* e dos sensores de IV no decorrer das duas trajetórias são injetados nos três filtros. As estimativa da postura realizada por cada um dos filtros é gerada e comparada a cada instante de tempo com o GT.

Tanto o FKE quanto o FKU são unimodais e possuem uma medida da qualidade da estimativa do estado, $\boldsymbol{P}$. Devido ao fato de inicialmente existir pouca informação referente à posição do robô pode ser que alguma condição inicial faça com que os filtros divirjam ou que demorem um pouco mais de tempo para localizar o robô. Durante a primeira trajetória a posição inicial do robô não tem uma influência muito grande, pois sabe-se que o robô se encontra a uma determinada distância da parede inferior e como ele se locomove em linha reta em algum momento algum sensor reportará uma distância com relação à parede lateral.

No entanto, para a segunda trajetória a condição inicial é extremamente importante para a precisão do filtro. Uma vez que o robô realiza uma trajetória em curva que faz com que aumente a incerteza na medida e no deslocamento. Então para o segundo trajeto foi informada uma localização próxima da posição real do robô. A Figura 30 mostra o resultado do FKU para 4 condições iniciais diferentes, onde esta é denotada por um asterisco preto em cada um dos quatro casos apresentados. Pode-se perceber que a trajetória começou a ser traçada corretamente na Figura 30a, 30b e 30c após a distância em relação à duas paredes ser fornecida pelos sensores (início da terceira curva). No entanto, para a posição inicial apresentada na Figura 30d foi traçada uma trajetória que não condiz com a que o robô executou. As quatro trajetórias geradas utilizando o FKE são similares às do FKU.

Apesar do FP não possuir uma limitação da condição inicial, as partículas foram distribuídas dentro de um raio de 50 cm da posição inicial real do robô ao invés de distribuir partículas por todo o mapa (por critérios de justiça em relação aos outros filtros).

As estimativas da primeira trajetória obtidas através dos três filtros são apresentados na Figura 31 juntamente com o GT. Também é apresentada a odometria, que é a estimativa utilizando utilizando somente as informações dos *encoders* dada a posição inicial real do robô. A trajetória só foi traçada a partir do momento em que foram obtidas as informações de distâncias em relação a duas paredes (por motivos de clareza).



(a) (b)

(c) (d)

Figura 30: Ilustração das trajetórias obtidas pelo FKU para diferentes condições iniciais (em azul). A trajetória real do robô é mostrada em vermelho.

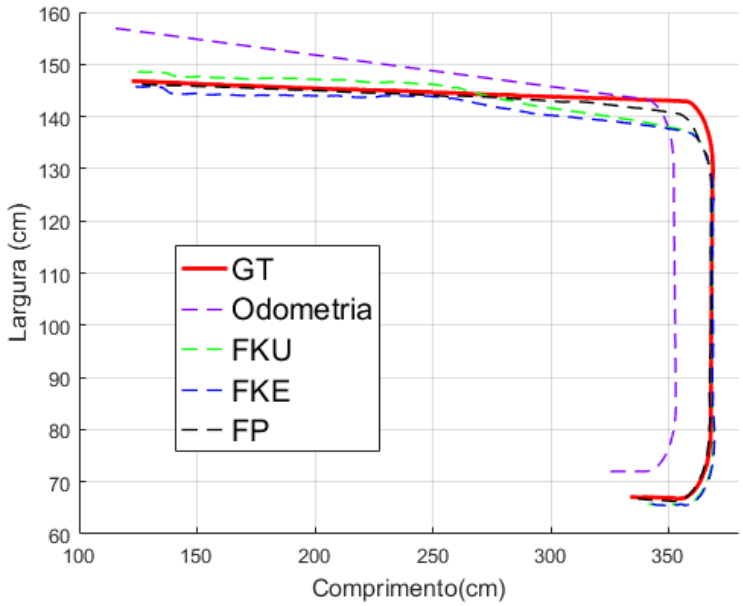


Figura 31: Estimativa da posição do robô durante a primeira trajetória utilizando o FKE, FKU, FP e odometria.

Percebe-se que o filtro que mais se aproximou do GT foi o FP. Todos os filtros conseguiram estimar a postura do robô até um pouco antes da segunda curva. Um pouco antes da segunda curva começar só havia informação de um sensor e em seguida nenhuma medida estava disponível, portanto os filtros não realizaram a correção. Os dois FK executaram a curva um pouco antes e isso fez com que a sua estimativa durante esse período

degradasse um pouco o seu desempenho.

O FKE e o FKU se recuperaram após as informações de distâncias referentes à parede superior estarem disponíveis e depois disso se mantiveram próximos da trajetória. No entanto, o FKU conseguiu se restabelecer na trajetória. Somente olhando para a Figura 31 não é possível saber o erro em relação a posição de uma forma quantitativa. Então, foi gerado um gráfico (Figura 32) que mostra o erro entre o GT e cada um dos filtros a cada instante de tempo.

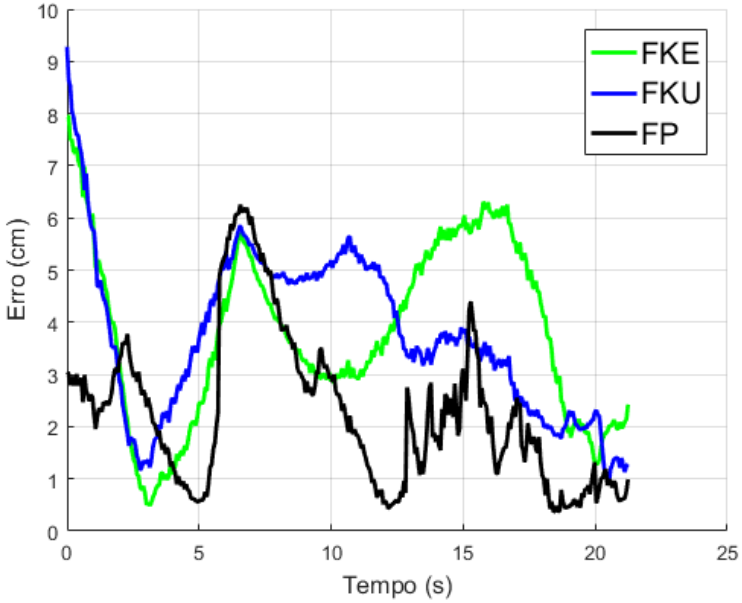


Figura 32: Distância entre o GT e as estimativas obtidas por cada filtro (erro) na primeira trajetória.

O erro foi calculado utilizando a distância entre o centro do robô (GT) e o centro do robô estimado para cada instante de tempo. Nota-se claramente que o FP obteve a melhor estimativa, mas de uma forma geral eles foram semelhantes e não é possível dizer qual obteve o segundo melhor desempenho. Então, calculou-se o erro quadrático médio das distâncias (ϵ_{dist}) por meio de (54).

$$\epsilon_{dist} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2} \tag{54}$$

A Tabela 3 mostra o valor desses erros para a primeira e para segunda trajetória. O FP, como esperado, é o que possui o menor erro quadrático médio. O seu desempenho é cerca de 65% superior à estimativa feita utilizando o FKU e o FKE.

A última coluna da Tabela 3 mostra o tempo médio que leva para executar cada passo do algoritmo. As computações das estimativas de cada filtro foram feitas por meio de um computador de mesa com um processador Core i5-4250U. Os algoritmos implementados não possuem nenhuma otimização específica e nenhum tipo de paralelização.

O resultado da estimativa da segunda trajetória para cada um dos filtros é apresentado na Figura 33. Mesmo fornecendo a posição inicial o FKU, inicialmente, desvia um pouco do trajeto enquanto que o FKE desvia um pouco menos e o FP parece permanecer no trajeto durante a maior parte do tempo. As trajetória começam a ser mostradas apenas

	Trajetória 1	Trajetória 2	Tempo
FKE	3,9 cm	7,8 cm	0,008 s
FKU	3,7 cm	6,0 cm	0,0120 s
FP	2,2 cm	4,2 cm	0,7479 s
Odometria	14,9 cm	36,5 cm	–

Tabela 3: Erro quadrático médio (ϵ_{dist}) para cada filtro em cada uma das trajetórias.

próximo de onde não há informação de nenhum sensor em em seguida surgem as informações de distâncias de duas paredes distintas. Nessa etapa que os filtros se recompõem, pode-se perceber que o FKU e o FKE demoram um pouco para fazer a correção da posição em relação ao FP. É nessa parte que os filtros se diferenciam mais, após essa parte eles possuem pequenas diferenças.

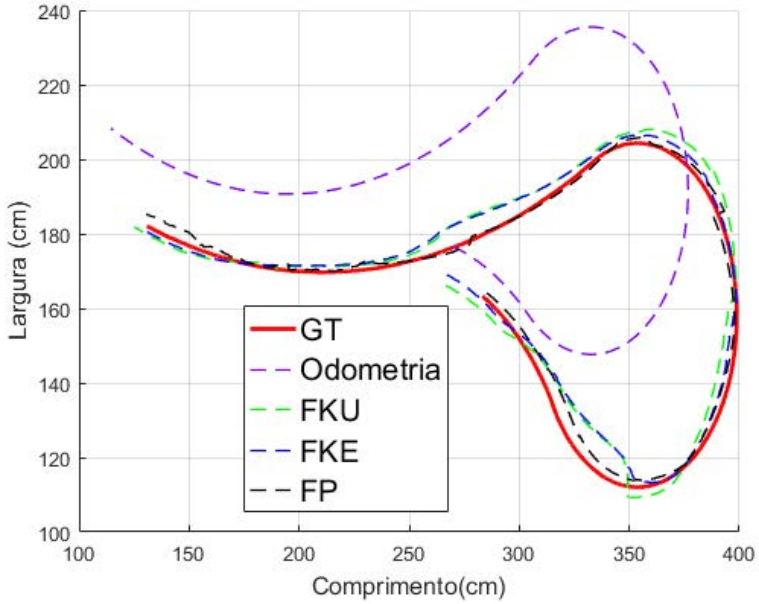


Figura 33: Estimativa da posição do robô durante a segunda trajetória utilizando o FKE, FKU, FP e odometria.

Nesta segunda trajetória fica mais claro qual é a ordem de melhor desempenho somente olhando a trajetória traçada. O erro quadrático é apresentado na Figura 34 e o erro quadrático médio é apresentado na Tabela 3. Durante este percurso o FP possui um desempenho 44% melhor do que o FKU e o FKU é 29% melhor do que o FKE.

Os resultados mostram que apesar de possuir uma diferença significativa em termos percentuais nos filtros apresentados dependendo da aplicação é muito mais indicado utilizar um dos FK pelo custo computacional apresentado. Uma diferença em média de 2 cm pode não ser crucial para a maioria das aplicações. Esta estimativa pode ser melhorada se forem incluídos outros sensores mais precisos e/ou sensores que possuam um alcance maior. Existem diversas abordagens na literatura com uso de sonares, câmeras ou até Laser *rangefinder*.

Neste primeiro experimento a estimativa foi feita utilizando um computador e foi feita de forma *offline*. No segundo experimento será feito o planejamento e a execução de duas trajetórias baseados na estimativa da posição inicial do robô e de um ponto objetivo para

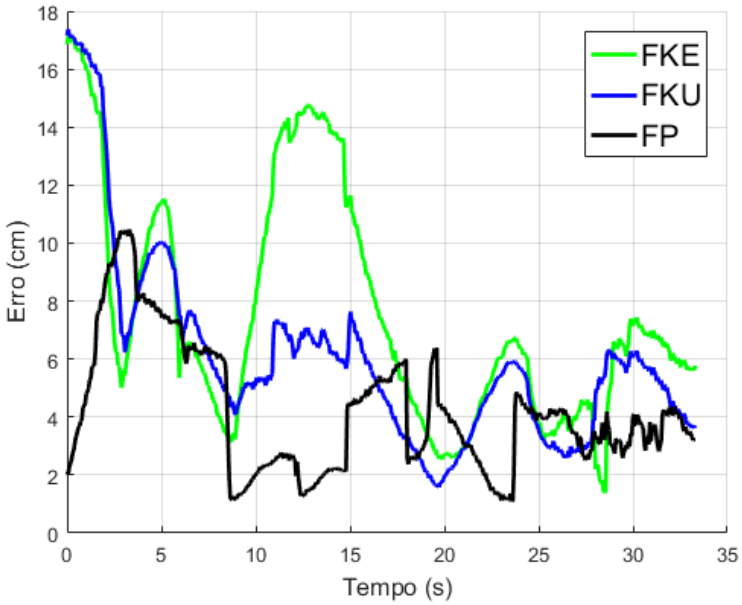


Figura 34: Distância entre o GT e as estimativas obtidas por cada filtro (erro) na segunda trajetória.

cada trajetória.

Como o FP demora cerca de 0,75 s em cada iteração ele não pode ser utilizado para realizar a estimativa da posição do robô durante todo o percurso. O FP será utilizado somente no início com o robô parado para encontrar a sua posição inicial no mapa com uma maior probabilidade de acerto. Após encontrar a postura inicial do robô é calculada a trajetória até o ponto objetivo utilizando o algoritmo A*.

Após obter a trajetória, o algoritmo *pure-pursuit* é utilizado para calcular a entrada necessária para que o robô se movimente até determinado ponto da trajetória e após cada movimento a sua postura é estimada utilizando o FKU. O FKU mostrou uma performance satisfatória com um custo computacional não muito elevado. Esta escolha permite utilizar a mesma taxa de amostragem para realizar o experimento.

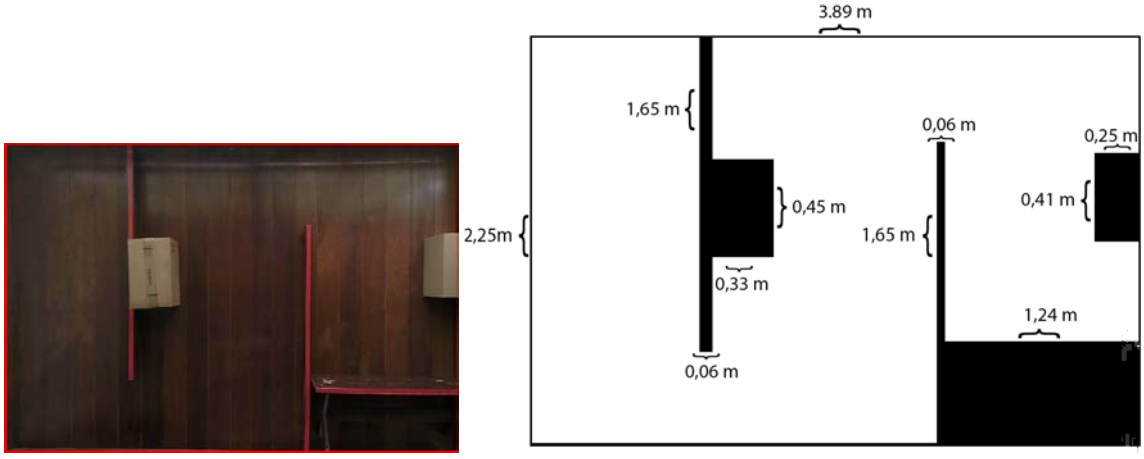
7.3 Planejamento e seguimento de rota

Neste experimento será apresentado um novo cenário e o objetivo do robô é encontrar a sua posição inicial e calcular as entradas de controle necessárias para ir até um ponto pré-definido mapa.

O ambiente contém alguns obstáculos para que o robô calcule uma rota com desvios e que ao mesmo tempo estes obstáculos auxiliem o robô a se localizar no mapa. O cenário escolhido é apresentado na Figura 35a. A Figura 35b mostra a extração do mapa com as dimensões da sala e dos objetos.

O primeiro passo que o robô realiza é a sua localização no ambiente. Para isso, o robô utiliza o FP para obter a sua localização com maior precisão. Após se localizar o robô calcula a rota que deve ser seguida e a estimação de estados durante a trajetória é feita utilizando o FKU.

O algoritmo A* utilizado para planejar a trajetória encontra o menor caminho entre dois pontos, se possível. No entanto, utilizando o mapa da Figura 35b a trajetória feita



(a) (b)

Figura 35: Mapa utilizando no experimento: (a) imagem da câmera com a distorção removida (b) extração do mapa

encontra-se muito próxima dos obstáculos fazendo com que o robô colida nos mesmos. Existem métodos clássicos que podem ser utilizados para evitar obstáculos no processo de gerar a trajetória, como o uso campos potenciais ou utilizando o diagrama de Voronoi.

Neste trabalho, no entanto, foi feita uma abordagem diferente. Os obstáculos foram dilatados de forma que o trajeto feito faça com que o robô mantenha uma distância fixa dos obstáculos, conforme ilustrado na Figura 36. Após gerar a trajetória o algoritmo de *pure-tracking* é utilizado para gerar as velocidades de cada uma das rodas do robô para chegar até a trajetória desejada. Porém, antes de implementar o algoritmo no robô é necessário definir a L_{Ah}. Para escolher o seu valor foram feitas algumas simulações da trajetória feita no mapa pelo algoritmo utilizando diferentes valores de L_{Ah}.

(a) (b)

Figura 36: Geração da trajetória: (a) Expansão dos objetos presentes no mapa (b) trajetória resultante planejada.

As simulações foram realizadas utilizando uma velocidade linear de $0,2$ m/s e variando o valor de L_{Ah} ($0,5$ m, $0,4$ m, $0,3$ m, $0,25$ m e $0,2$ m), conforme ilustrado na Figura 37. O valor mais alto de L_{Ah} ($0,5$ m) gera cortes acentuados na trajetória. Percebe-se que o corte executado na segunda curva é feito de uma forma exagerada e faz com que ocorra uma colisão com o obstáculo.

Apesar da trajetória resultante de $L_{Ah} = 0,4$ m não apresentar uma colisão ela possui uma distância muito pequena em relação aos obstáculos. Este valor implica em uma

margem de erro muito pequena por parte do robô durante o experimento prático, podendo fazer com que o mesmo colida com algum objeto durante o percurso. Portanto, foi optado por utilizar o valor de $L_{Ah} = 0.25$ m pelo fato da trajetória executada manter uma distância relativamente segura em relação aos obstáculos.

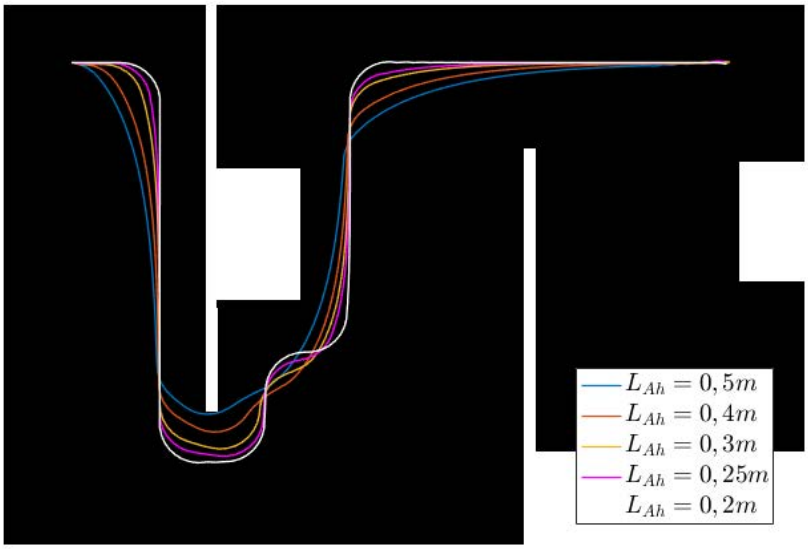


Figura 37: Teste de diferentes valores de L_{Ah} para a trajetória definida.

No primeiro caso o robô situa-se no canto superior esquerdo da sala e o resultado do experimento é mostrada na Figura 38, onde o trajeto em verde é o conjunto de pontos que compõem o caminho planejado (algoritmo A*), o tracejado em amarelo representa o trajeto simulado (utilizando o algoritmo *pure-pursuit*) e o tracejado em vermelho mostra o caminho feito pelo robô durante o experimento prático.

Pode-se perceber que inicialmente o robô começou a dar pequenos giros depois da primeira curva (devido a erros na sua estimativa), mas ele conseguiu manter-se próximo à trajetória definida e chegar no ponto desejado com uma distância de $5,09$ cm em relação a ele. Este experimento foi repetido 7 vezes e a distância média final entra a posição final do robô e o ponto de meta foi de $4,42$ cm.

No segundo experimento foi definida um ponto mais próximo do robô, o que fez a trajetória a ser executada um pouco mais curta. Dessa vez, o robô iniciou a sua trajetória no canto inferior direito da sala. A Figura 39 mostra a trajetória traçada (utilizando o algoritmo A*), a simulação das entradas necessárias para levar o robô até o ponto definido (utilizando o algoritmo *pure-pursuit*) e, em vermelho, o trajeto feito pelo robô na prática.

Neste experimento, no entanto, houve uma diferença maior em relação à trajetória simulada. Observa-se que a primeira curva na simulação foi mais fechada, mas no experimento o robô não conseguiu fazer essa mesma curva. As trajetórias realizadas diferem, pois as entradas são calculadas a cada iteração e estas são baseadas na postura atual do robô e um ponto situado na trajetória a uma distância de pelo menos 25 cm. Não obstante, apesar das trajetórias serem diferentes o robô consegue chegar até o ponto pré-definido sem colidir com nenhum obstáculo.

A distância entre o ponto planejado e o ponto que o robô efetivamente chegou foi de $3,13$ cm. Da mesma forma que ocorreu anteriormente, o experimento foi repetido sete vezes e a distância média entre o ponto final da trajetória e o robô foi de $4,06$ cm.

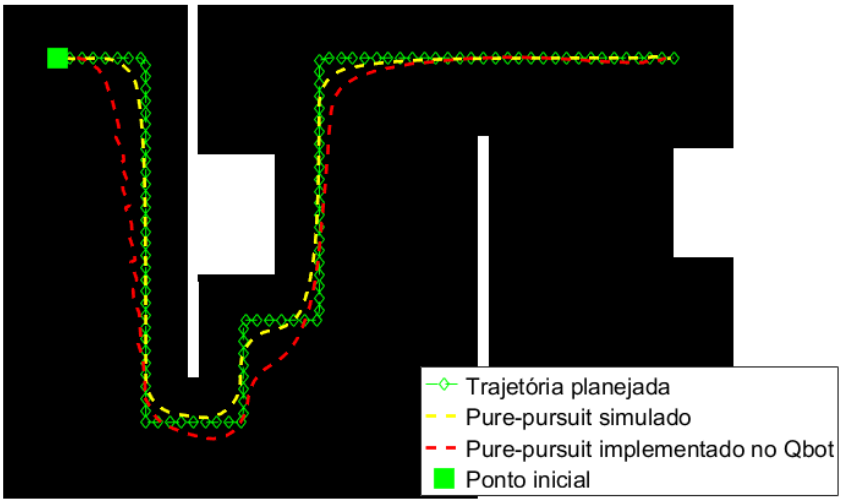


Figura 38: Teste prático do Qbot envolvendo o planejamento e a execução da primeira trajetória para chegar até o ponto pré-definido.

O erro médio da posição final foi semelhante nos dois experimentos. Como o robô ajusta as entradas de controle para se aproximar da trajetória, os erros associados à própria entrada de controle ou à estimativa da postura fazem com que o robô modifique o seu caminho, fazendo com que a trajetória executada mude um pouco toda a vez que o experimento é realizado.

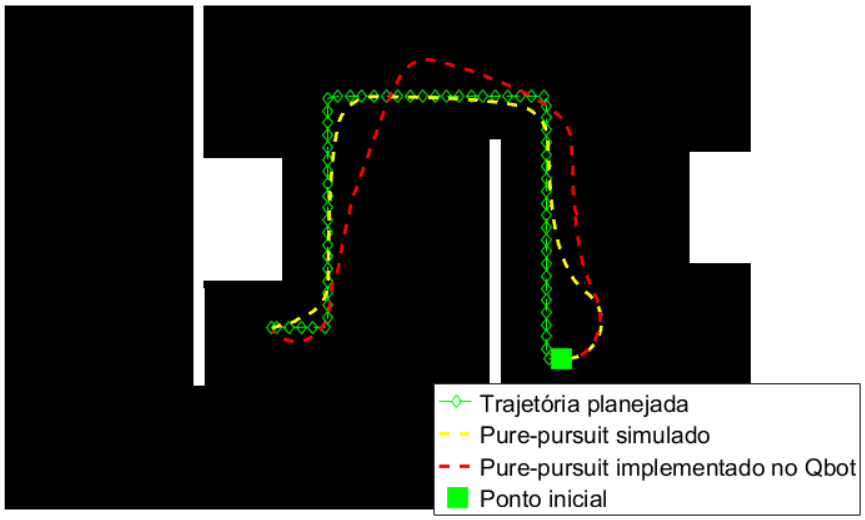


Figura 39: Teste prático do Qbot envolvendo o planejamento e a execução da segunda trajetória para chegar até o ponto pré-definido.

8 CONCLUSÕES

Foram apresentados, implementados e comparados experimentalmente três filtros para estimar a postura de um robô diferencial: o FKE, o FKU e o FP. Foi mostrado que utilizando um modelo simples e sensores imprecisos foi possível melhorar significativamente a estimativa da postura do robô. O modelo do sensor mostrou-se crítico durante os experimentos, visto que quando a variância não foi considerada como função da distância e do ângulo em relação ao obstáculo não foi possível obter uma estimativa significativamente melhor do que utilizando somente a odometria.

A maioria dos trabalhos presentes na literatura não descrevem a forma pela qual o GT foi obtido e tampouco fazem uma análise do erro durante todo o experimento. Para mensurar de forma qualitativa e quantitativa a eficácia de cada um dos filtros apresentados, foi proposto um método simples, mas eficaz para gerar o GT de cada um dos experimentos. Dessa forma, o desempenho de cada filtro pôde ser comparado a cada instante de tempo de forma precisa.

Dentro dos resultados obtidos o FP, como esperado, possui a melhor estimativa. Ele obteve uma boa precisão mesmo quando havia pouca ou nenhuma informação provida pelos sensores de IV. No entanto, este desempenho apresentou um custo computacional muito mais elevado, onde este valor é proporcional ao número de partículas escolhido.

O custo computacional do FKU mostrou-se cerca de quase duas ordens de grandeza mais rápido do que o custo computacional do FP implementado. Tanto o FKU quanto o FKE obtiveram uma melhora considerável em relação à estimativa feita utilizando somente a odometria. Devido ao baixo custo computacional, tanto o FKE quanto o FKU podem ser implementados em microprocessadores de baixo custo, podendo ser embarcado em um robô para realizar a estimação de estados e inclusive executar outras tarefas.

O filtro de partículas, por sua vez, requer um *hardware* mais sofisticado para ser implementado, implicando em um custo maior envolvido no projeto. Neste caso deve ser analisado se a utilização de um número maior de sensores, sensores com um alcance maior e/ou sensores com uma variância menor já não oferecem a precisão necessária para a aplicação através da fusão de suas informações por meio de um dos filtros Gaussianos.

Neste trabalho também foram apresentados e implementados um método simples de extração do mapa e um método utilizado para conduzir o robô até um determinado ponto no mapa, isto é, gerar as entradas de controle necessárias para conduzir o robô até determinado ponto. Para isso foram utilizados um algoritmo para encontrar o menor caminho entre dois pontos (planejamento de rota) e um algoritmo de perseguição de trajetória. As entradas de controle do algoritmo *pure-pursuit* são calculadas a cada instante de tempo e para isso é necessário saber a posição do robô. Após os resultados obtidos com o primeiro experimento, optou-se pelo FKU para realizar a estimativa da postura do robô devido ao seu desempenho e ao seu custo computacional.

Foi mostrado por meio de dois experimentos que utilizando estes dois algoritmos juntamente com a estimação da postura do robô obtida através da fusão dos sensores por meio do FKU é possível conduzir o robô até um determinado ponto definido no mapa. A precisão obtida na posição final do robô neste segundo experimento condiz com a precisão obtida pelo FKU durante o primeiro experimento.

Existem diversas extensões que podem ser feitas para o trabalho realizado. Como o foco deste trabalho foi a estimação do vetor de estados do robô, o planejamento e a execução da trajetória foram feitos utilizando um método simples e amplamente abordado na literatura para seguir uma trajetória. Ao invés disso poderia ser utilizado um método reativo, onde o robô desviaria de obstáculos baseados nas informações obtidas pelos sensor de distância sem a necessidade de previamente mapear os obstáculos.

Outra extensão possível seria fazer o mapeamento da sala por meio dos sensores. No entanto, devido às características dos sensores utilizado neste trabalho deve-se considerar utilizar as medições com uma distância curta (inferior à 40 cm) para não comprometer a geração do mapa ou considerar a utilização de um outro sensor para este fim.

REFERÊNCIAS

ALESSANDRI, A. et al. An application of the extended Kalman filter for integrated navigation in mobile robotics. In: AMERICAN CONTROL CONFERENCE, 1997, Albuquerque, New Mexico, USA. **Anais...** [S.l.: s.n.], 1997. v.1, p.527–531.

ANDERSEN, H. et al. Geometric path tracking algorithm for autonomous driving in pedestrian environment. In: IEEE INTERNATIONAL CONFERENCE ON ADVANCED INTELLIGENT MECHATRONICS (AIM), 2016, Banff, Canada. **Anais...** Piscataway: IEEE, 2016. p.1669–1674.

ANDERSON, B.; MOORE, J. **Optimal Filtering**. [S.l.]: Dover Publications, 2012. (Dover Books on Electrical Engineering).

ANJUM, M. L. et al. Sensor data fusion using Unscented Kalman Filter for accurate localization of mobile robots. In: INTERNATIONAL CONFERENCE ON CONTROL, AUTOMATION AND SYSTEMS, 2010, Gyeonggi-do, South Korea. **Anais...** [S.l.: s.n.], 2010. p.947–952.

BAK, A. et al. Multi-sensor localization - Visual Odometry as a low cost proprioceptive sensor. In: INTERNATIONAL IEEE CONFERENCE ON INTELLIGENT TRANSPORTATION SYSTEMS, 15., 2012, Anchorage, Alaska, USA. **Anais...** IEEE, 2012. p.1365–1370.

BEEVERS, K. R.; HUANG, W. H. SLAM with sparse sensing. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2006, Taipei, Taiwan. **Anais...** Piscataway: IEEE, 2006. p.2285–2290.

BEKEY, G. **Autonomous Robots**: from biological inspiration to implementation and control. [S.l.]: MIT Press, 2005. (Intelligent robotics and autonomous agents).

BOLIC, M.; DJURIC, P. M.; HONG, S. Resampling algorithms and architectures for distributed particle filters. **IEEE Transactions on Signal Processing**, [S.l.], v.53, n.7, p.2442–2450, July 2005.

BRŠČIĆ, D.; HASHIMOTO, H. Comparison of robot localization methods using distributed and onboard laser range finders. In: IEEE INTERNATIONAL CONFERENCE ON ADVANCED INTELLIGENT MECHATRONICS, 2008, Xian, China. **Anais...** Piscataway: IEEE, 2008. p.746–751.

CHEN, Z. Bayesian filtering: from kalman filters to particle filters, and beyond. **Statistics**, [S.l.], v.182, n.1, p.1–69, 2003.

COULTER, R. C. **Implementation of the Pure Pursuit Path Tracking Algorithm**. Pittsburgh, PA: Robotics Institute, 1992.

CROWLEY, J. L. World modeling and position estimation for a mobile robot using ultrasonic ranging. In: INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1989, Scottsdale, Arizona, USA. **Anais...** [S.l.: s.n.], 1989. p.674–680 v.2.

D'ALFONSO, L. et al. Mobile robot localization via EKF and UKF: a comparison based on real data. **Robotics and Autonomous Systems**, [S.l.], v.74, p.122–127, 2015.

DIJKSTRA, E. W. A Note on Two Problems in Connexion with Graphs. **Numerische Mathematik**, [S.l.], v.1, n.1, p.269–271, 1959.

DOUC, R.; CAPPE, O. Comparison of resampling schemes for particle filtering. In: INTERNATIONAL SYMPOSIUM ON IMAGE AND SIGNAL PROCESSING AND ANALYSIS, 4., 2005, Zagreb, Croatia. **Proceedings...** Piscataway: IEEE, 2005. p.64–69.

DRUMHELLER, M. Mobile Robot Localization Using Sonar. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.PAMI-9, n.2, p.325–332, Mar. 1987.

DUDEK, G.; JENKIN, M. **Computational Principles of Mobile Robotics**. New York, NY, USA: Cambridge University Press, 2000.

DURRANT-WHYTE, H.; HENDERSON, T. C. Multisensor Data Fusion. In: SICILIANO, B.; KHATIB, O. (Ed.). **Springer Handbook of Robotics**. Berlin: Springer-Verlag Berlin Heidelberg, 2008, Section C. p.585–610.

FANG, H.; YANG, M.; YANG, R. Ground Texture Matching based Global Localization for Intelligent Vehicles in Urban Environment. In: IEEE INTELLIGENT VEHICLES SYMPOSIUM, 2007, Istanbul, Turkey. **Anais...** Piscataway: IEEE, 2007. p.105–110.

FERNÁNDEZ-MADRIGAL, J. **Simultaneous Localization and Mapping for Mobile Robots**: introduction and methods: introduction and methods. [S.l.]: Information Science Reference, 2012. (Advances in Computational Intelligence and Robotics:).

GUSTAFSSON, F. et al. Particle filters for positioning, navigation, and tracking. **IEEE Transactions on Signal Processing**, [S.l.], v.50, n.2, p.425–437, 2002.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. **IEEE Transactions on Systems, Science, and Cybernetics**, [S.l.], v.4, n.2, p.100–107, 1968.

HARTLEY, R.; ZISSERMAN, A. **Multiple View Geometry in Computer Vision**. 2nd.ed. [S.l.]: Cambridge University Press, 2003.

HOANG, V. D. et al. Localization estimation based on Extended Kalman filter using multiple sensors. In: ANNUAL CONFERENCE OF THE IEEE INDUSTRIAL ELECTRONICS SOCIETY, 39., 2013, Vienna, Austria. **Anais...** Piscataway: IEEE, 2013. p.5498–5503.

HOUSHANGI, N.; AZIZI, F. Accurate mobile robot position determination using unscented Kalman filter. In: CANADIAN CONFERENCE ON ELECTRICAL AND COMPUTER ENGINEERING, 2005, Saskatoon, SK, Canada. **Anais...** Piscataway: IEEE, 2005. p.846–851.

IVANJKO, E.; PETROVIć, I.; BREZAK, M. Experimental Comparison of Sonar Based Occupancy Grid Mapping Methods. **Automatika**, [S.l.], v.50, p.65–79, 2009.

JULIER, S.; UHLMANN, J. K.; DURRANT-WHYTE, H. F. A new method for the nonlinear transformation of means and covariances in filters and estimators. **IEEE Transactions on Automatic Control**, [S.l.], v.45, n.3, p.477–482, 2000.

KALMAN, R. E. A New Approach to Linear Filtering and Prediction Problems. **Transactions of the American Society of Mechanical Engineers – Journal of Basic Engineering**, [S.l.], v.82, p.35–45, 1960.

KAVRAKI, L. et al. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1996, Minneapolis, USA. **Anais...** IEEE, 1996. p.566–580.

LAVALLE, S. M.; JR., J. J. K. Randomized Kinodynamic Planning. **The International Journal of Robotics Research**, [S.l.], v.20, n.5, p.378–400, 2001.

LEONARD, J. J.; DURRANT-WHYTE, H. F. Mobile robot localization by tracking geometric beacons. **IEEE Transactions on Robotics and Automation**, [S.l.], v.7, p.376–382, Jun 1991.

MONTEMERLO, D.; THRUN, S.; WHITTAKER, W. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In: INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2002, Washington, DC, USA. **Proceedings...** Piscataway: IEEE, 2002. v.1, p.695–701.

MOURIKIS, A. I.; ROUMELIOTIS, S. I. Performance analysis of multirobot Cooperative localization. **IEEE Transactions on Robotics**, [S.l.], v.22, n.4, p.666–681, Aug 2006.

NILSSON, N. J. **Principles of Artificial Intelligence**. San Francisco, USA: Morgan Kaufmann Publishers Inc., 1980.

PAPOULIS, A.; PILLAI, S. U. **Probability, Random Variables and Stochastic Processes**. 4th.ed. [S.l.]: McGraw-Hill Europe, 2002.

PINZ, M. R. A. A comparison of three uncertainty calculi for building sonar-based occupancy grids. **Robotics and Autonomous Systems**, [S.l.], v.35, p.201–209, 2001.

ROUMELIOTIS, S. I. Extended Kalman filter for frequent local and infrequent global sensor data fusion. **Proceedings of SPIE**, [S.l.], n.213, p.11–22, 1997.

ROUMELIOTIS, S. I.; SUKHATME, G. S.; BEKEY, G. A. Circumventing dynamic modeling: evaluation of the error-state kalman filter applied to mobile robot localization. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1999, Detroid, MI, USA. **Proceedings...** IEEE, 1999. v.2, p.1656–1663.

SASIADEK J.Z.; WANG, Q. Sensor fusion based on fuzzy Kalman filtering for autonomous robot vehicle. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1999, Detroit, MI, USA. **Proceedings...** Piscataway: IEEE, 1999. v.4, p.2970–2975.

SIMON, D. **Optimal State Estimation**: kalman, h infinity, and nonlinear approaches. Hoboken, New Jersey, United States: Wiley-Interscience, 2006.

SMITH, R.; SELF, M.; CHEESEMAN, P. A Stochastic Map for Uncertain Spatial Relationships. In: INTERNATIONAL SYMPOSIUM ON ROBOTICS RESEARCH, 4., 1988, Santa Clara, California, USA. **Proceedings...** Cambridge: MIT Press, 1988. p.467–474.

SNIDER, J. M. **Automatic Steering Methods for Autonomous Automobile Path Tracking**. Pittsburgh, PA: Robotics Institute, 2009.

THRUN, S. Learning occupancy grids with forward models. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS. EXPANDING THE SOCIETAL ROLE OF ROBOTICS IN THE THE NEXT MILLENNIUM, 2001, Maui, Hawaii, USA. **Proceedings...** Piscataway: IEEE, 2001. v.3, p.1676–1681.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)**. [S.l.]: The MIT Press, 2005.

WAN, E.; Van Der Merwe, R. The unscented Kalman filter for nonlinear estimation. In: IEEE ADAPTIVE SYSTEMS FOR SIGNAL PROCESSING, COMMUNICATIONS, AND CONTROL SYMPOSIUM, 2000, Lake Louise, Alberta, Canada. **Proceedings...** Piscataway: IEEE, 2000. p.153–158.