Sistema de navegação para robôs AMR de baixo custo

Rhuan Ferrer da Silva[1], **Adilson Luiz Bonifácio**[1]

[1]Departamento de Computação – Universidade Estadual de Londrina (UEL)
Caixa Postal 10.011 – CEP 86057-970 – Londrina – PR – Brasil

rhuan.ferrer.silva@uel.br, bonifacio@uel.br

Abstract. *The field of autonomous vehicle research and robotics in general has made great progress in recent years. This is present in all sectors, from large companies to people who consider robotics just a hobby. However, the literature does not provide clear indications of a method for SLAM systems with inexpensive equipment. The aim of this research is to carry out a systematic study of the best-known algorithms in this field and then create a useful implementation of each of them, as well as an interface for data collection. The final goal is to demonstrate the advantages and disadvantages of each algorithm when working with extremely inaccurate sensors.*

Resumo. *O campo de pesquisa de veículos autônomos e robótica em geral tem feito um grande progresso nos últimos anos. Isso está presente em todos os setores, desde as grandes empresas até as pessoas que consideram a robótica apenas um passatempo. Porém, a literatura não fornece indicações claras de um método para sistemas SLAM com equipamento barato. O objetivo desta pesquisa é realizar um estudo sistemático dos algoritmos mais conhecidos neste campo e, em seguida, criar uma implementação de cada um deles, bem como uma interface para coleta de dados. O objetivo final é demonstrar as vantagens e desvantagens de cada algoritmo ao trabalhar com sensores extremamente imprecisos.*

1. Introdução

Para que um robô seja capaz de operar em um ambiente desconhecido de forma autônoma, é fundamental que ele tenha um sistema de mapeamento e localização [Mataric 2007]. Mapeamento, pois ele deve ser capaz de realizar tarefas específicas que demandam dados do local em que ele está inserido, como calcular e seguir uma rota específica, encontrar objetos, desviar de obstáculos, etc. E localização, já que para realizar tais tarefas não bastam as informações do mapa, ele deve ser capaz de se localizar nele. Seja para, retomando os exemplos anteriores, determinar a sua localização atual como ponto de partida da sua rota, avaliar com precisão os movimentos necessários para interagir com um objeto, e por fim, ao encontrar um obstáculo, ser capaz de estimar o quanto ele deve ajustar a sua rota para não haver colisão. A soma destes dois problemas é conhecida na literatura como o problema de mapeamento e localização simultânea (Simultaneous Localization and Mapping - SLAM) [Becker 2022].

Nas últimas décadas, muitos trabalhos vêm sendo desenvolvidos para avaliar métodos que lidem com esse tipo de problema, uma vez que os sensores utilizados pela maior parte dos robôs, seja para pesquisas acadêmicas ou passatempo, são de forma geral

carregados de uma grande imprecisão em suas medições por serem de baixo custo. O problema principal em lidar com esse tipo de sensor é que, além da confiabilidade dos dados ser reduzida, ela tende a diminuir ainda mais com o passar do tempo, chegando a um ponto em que é impossível utilizá-los em um algoritmo de SLAM sem antes passar por algum tipo de tratamento de dados [Mataric 2007].

O motivo para a escolha do tema deste projeto veio a partir de alguns experimentos para desenvolver um sistema de navegação que utilize sensores de baixo custo aliada à dificuldade em encontrar na literatura uma boa indicação de algoritmo que lide com esse tipo de sensor, mas que também seja capaz de ser executado em um hardware de poder computacional muito pequeno.

2. Fundamentação

O problema da localização de robôs tem sido debatido extensivamente nos últimos anos tanto pela comunidade acadêmica quanto pelas grandes empresas, que vêm investindo cada vez mais em seus setores de pesquisa e desenvolvimento próprios.

Os sistemas de localização podem ser segmentados em dois tipos, considerando características específicas inerentes a cada situação. O primeiro tipo são os sistemas de localização absoluta, que são aqueles onde a principal fonte de informação sobre a posição do robô é externa a ele próprio, como, por exemplo, sistemas de GPS, redes UWB e os próprios mapas gerados pela componente de mapeamento dos sistemas SLAM.

O segundo tipo são os sistemas de localização relativa, onde a fonte de informação é interna ao robô e se move com ele. Um dos principais métodos desse tipo são os sistemas por odometria, que utilizam sensores *encoders* acoplados a cada uma das rodas do robô para medir a taxa de rotação de cada uma [Mataric 2007]. Essa taxa pode ser relacionada com as dimensões da roda e o tempo necessário para uma volta completa, resultando assim no cálculo da distância total percorrida por cada roda a partir da equação:

$$\text{Distância} = 2 \cdot \pi \cdot \text{RaioDaRoda} \cdot \frac{\text{TempoParaUmaVolta}}{\text{TaxaDeRotação}}$$

E por fim, a distância total percorrida pelo robô é obtida pela média aritmética das distâncias percorridas pelas rodas [Becker 2022].

Outra forma popular para localização relativa, e que será a adotada neste projeto, é por meio de um sistema de navegação inercial, este método utiliza giroscópios, acelerômetros e magnetômetros unificados em um único sistema para estimar a posição atual. Giroscópios são sensores responsáveis por medir a aceleração na orientação do robô. Acelerômetros são sensores que medem a aceleração nos eixos cartesianos do robô, que em estado estacionário na superfície terrestre possui valores $(x, y, z) = (0, 0, 1)$, sendo a aceleração no eixo Z a aceleração gravitacional de $10m/s^2$. E por fim, os magnetômetros medem o campo magnético local. Ao combinar as informações desses três tipos de sensores, o sistema de navegação inercial é capaz de estimar a posição e orientação do robô em relação ao seu ponto de partida.

A combinação dos dados desses sensores, ou *sensor fusion*, conforme conhecida na literatura, é possível a partir de vários métodos diferentes que unificam as informações de forma que seja possível estimar com certa precisão as coordenadas (x, y, z) do robô

levando em conta as imprecisões de cada sensor. Os métodos utilizados nesse estudo serão: Kalman Filter (KF), Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) entre outros métodos amplamente utilizados e validados. A seguir, os métodos baseados no filtro de Kalman, assim como sua versão original, serão apresentados individualmente com um breve resumo seguido pelo seu funcionamento básico.

2.1. Kalman Filter

O filtro de Kalman [Kalman 1960] é um algoritmo estatístico de predição de estado para modelos dinâmicos lineares. O que o torna ideal para ser usado neste trabalho é sua característica de considerar nas suas estimativas a incerteza associada aos dados obtidos pelos sensores, assim como as imprecisões do modelo, oferecendo a solução ótima para casos onde há uma modelagem matemática consistente com o mundo real e o grau de imprecisão dos sensores é bem conhecido. Outra característica fundamental deste filtro é sua capacidade de combinar os dados dos diferentes sensores necessários para a localização do robô, uma vez que estes dados podem ser relacionados através do modelo matemático [Higgins 1975].

O algoritmo original é dividido em 5 equações que serão descritas individualmente, mas antes disso é de extrema importância que se tenham alguns conhecimentos básicos para um bom entendimento sobre o seu processo e o de suas variações na totalidade. Tais conceitos são apresentados a seguir:

- n: Iteração atual.
- $n-1$: Iteração anterior.
- $n+1$: Iteração seguinte.
- $\hat{\mathbf{x}}$: Estado atual do sistema. É uma variável que representa as coordenadas do robô entre outros dados relevantes.
- $\hat{\mathbf{x}}_{\mathbf{n,n}}$: Estado do sistema na iteração n estimado na mesma iteração n.
- $\hat{\mathbf{x}}_{\mathbf{n+1,n}}$: Estado do sistema na iteração $n+1$ estimado na iteração n.
- $\hat{\mathbf{x}}_{\mathbf{n,n-1}}$: Estado do sistema na iteração n estimado na iteração $n-1$.
- $\mathbf{K_n}$: Ganho de Kalman. Variável que representa a influência da leitura dos sensores no cálculo da estimativa do estado.
- $\mathbf{z_n}$: Informações obtidas pelos sensores.
- $\mathbf{H}$: Matriz de observação. Converte as leituras "brutas" dos sensores para os dados necessários para a estimativa de estado. Esta conversão é feita a partir da análise do caso específico, e pode utilizar funções matemáticas lineares e não lineares.
- $\mathbf{P}$: Matriz de covariância de estado. Representa a incerteza atual do sistema.
- $\mathbf{P_{n,n}}$: Matriz de covariância para o estado na iteração n calculada na mesma iteração n
- $\mathbf{P_{n,n-1}}$: Matriz de covariância para o estado na iteração n calculada na iteração $n-1$
- $\mathbf{P_{n+1,n}}$: Matriz de covariância para o estado na iteração $n+1$ calculada na iteração n
- $\mathbf{P_{xx}}$: Variância de X
- $\mathbf{P_{yy}}$: Variância de Y
- $\mathbf{P_{xy}}$: Variância cruzada de X e Y
- $\mathbf{P_{yx}}$: Variância cruzada de Y e X

- $\mathbf{F}$: Matriz de transição de estado. Representa o modelo matemático do sistema a partir de fórmulas que descrevem o seu comportamento físico. Estas fórmulas podem ser tanto lineares como não lineares.
- $\mathbf{Q}$: Matriz de ruído de processo. Representa o ruído gerado pela imprecisão do modelo matemático do sistema.
- $\mathbf{A^T}$: Matriz transposta.
- $\mathbf{A^{-1}}$: Matriz inversa.
- $\mathbf{G}$: Matriz de controle. Representa alguma informação adicional a respeito do estado real do sistema que não é medida por nenhum sensor.
- $\mathbf{R_n}$: Representa a covariância entre as leituras dos sensores.
- $\mathbf{u_n}$: Representa o ruído na leitura dos sensores.

A seguir segue a apresentação e explicação das equações que compõem o algoritmo clássico:

- **Atualização do estado**
  O papel desta, que é a primeira equação do filtro, é o de combinar as estimativas de estado calculadas a priori com as medidas mais recentes do estado real do sistema.

$$\hat{\mathbf{x}}_{\mathbf{n,n}} = \hat{\mathbf{x}}_{\mathbf{n,n-1}} + \mathbf{K_n}(\mathbf{z_n} - \mathbf{H}\hat{\mathbf{x}}_{\mathbf{n,n-1}})$$

- **Atualização da covariância**
  A segunda equação do algoritmo atualiza a covariância estimada do sistema para a iteração atual.

$$\mathbf{P_{n,n}} = (\mathbf{I} - \mathbf{K_n H})\mathbf{P_{n,n-1}}(\mathbf{I} - \mathbf{K_n H})^T + \mathbf{K_n R_n K_n^T}$$

- **Atualização do Ganho de Kalman**
  Com base na atualização realizada no passo anterior, esta próxima equação é responsável por atualizar o Ganho de Kalman para esta iteração.

$$\mathbf{K_n} = \mathbf{P_{n,n-1}H^T}(\mathbf{HP_{n,n-1}H^T} + \mathbf{R_n})^{-1}$$

- **Extrapolação do estado**
  Com todas as estimativas para o estado atual do sistema atualizadas, esta etapa utiliza o modelo matemático para realizar a predição do estado mais provável do sistema se encontrar na iteração seguinte.

$$\hat{\mathbf{x}}_{\mathbf{n+1,n}} = \mathbf{F}\hat{\mathbf{x}}_{\mathbf{n,n}} + \mathbf{Gu_n}$$

- **Extrapolação da covariância**
  E por fim, seguindo o mesmo padrão da etapa passada, a última equação do algoritmo estima a covariância do sistema para a próxima iteração.

$$\mathbf{P_{n+1,n}} = \mathbf{FP_{n,n}F^T} + \mathbf{Q}$$

2.2. O problema da não-linearidade

Como dito anteriormente, o filtro de Kalman é um algoritmo estatístico recursivo para predição de estado de sistemas dinâmicos lineares, ou seja, ele pode estimar com precisão o estado em que um sistema se encontra com base nos estados anteriores deste sistema, contanto que o mesmo possa ser modelado matematicamente utilizando fórmulas físicas lineares tanto para o modelo (Matriz de transição de estado $\mathbf{F}$) como para a conversão dos dados (Matriz de observação $\mathbf{H}$) [Javaid et al. 2021]. Isso se deve ao fato de que a incerteza de modelos dinâmicos lineares pode ser descrita por uma função densidade de probabilidade gaussiana, o que não é verdade para sistemas não-lineares. Nesse tipo de sistema, são necessárias algumas alterações na estrutura do algoritmo para adaptá-lo para essas situações [Becker 2022].

Mas para apresentar essas variações, é necessário antes introduzir o conceito de problema não linear. Para isso, podemos dividi-lo em dois subtipos, que serão analisados individualmente a seguir:

- **Relação não linear entre estado e medição**
  Nesses casos, mesmo que a dinâmica do sistema seja bem conhecida e modelada de forma linear, a conversão da leitura dos sensores (realizada pela matriz de observação $\mathbf{H}$) para o estado do sistema não o é. Por exemplo, como apresentado anteriormente, um giroscópio é um sensor que detecta a aceleração na orientação do robô, ou de qualquer outro objeto que ele esteja fixado. A partir disso, é possível calcular o seu ângulo de rotação (roll) da seguinte forma:

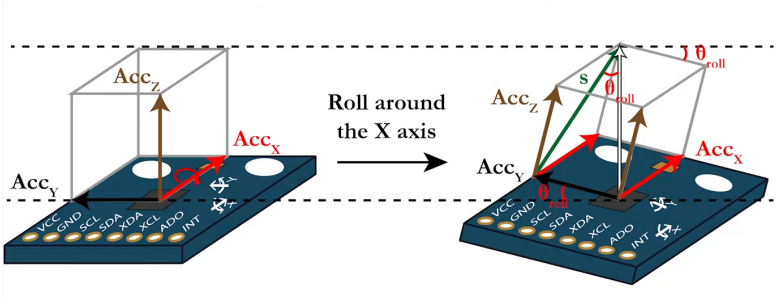

Figure 1. Exemplo giroscópio

$$\theta_{roll} = atan\left(\frac{-Acc_Y}{\sqrt{Acc_X^2 + Acc_Z^2}}\right)$$

A equação que converte a medida do sensor (a aceleração) para o estado atual do robô, neste caso seu ângulo de rotação θ_{roll}, é uma função não-linear, logo, o estado deste sistema não pode ser estimado pelo filtro de Kalman.

- **Dinâmica de sistema não linear**
  Neste caso do problema, a própria dinâmica do sistema (calculada pela matriz de transição de estado $\mathbf{F}$) configura um modelo não linear, tornando também inviável qualquer estimativa via filtro de Kalman. Vamos tomar como exemplo o sistema de pêndulo apresentado a seguir e desenvolver nossa análise:

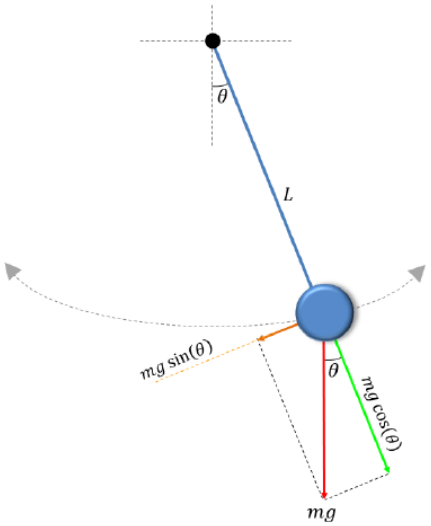

Figure 2. Exemplo pêndulo

Queremos estimar o ângulo θ atual do pêndulo. O tamanho do arco correspondente ao ângulo θ por onde o pêndulo se desloca vale $s = L\theta$, logo, a velocidade do pêndulo pode ser descrita pela equação:

$$v = \frac{\partial s}{\partial t} = L\frac{\partial \theta}{\partial t}$$

E a sua aceleração é descrita pela equação:

$$a = \frac{\partial^2 s}{\partial t^2} = L\frac{\partial^2 \theta}{\partial t^2} = -g\sin(\theta)$$

Que, como podemos observar, é uma função trigonométrica, portanto, não-linear, logo, o filtro de Kalman também não é capaz de estimar o estado desse sistema. De forma análoga, ele também não poderá estimar com precisão a posição do robô em qualquer trajeto onde o seu movimento se assimile ao descrito neste exemplo.

Contextualizado o problema, podemos seguir com as variações do filtro de Kalman que são capazes de lidar com sistemas não lineares.

2.3. Extended Kalman Filter

A partir da necessidade de um algoritmo que fosse capaz de estimar o estado dos sistemas mais complexos do mundo real, surge então, a partir de uma pesquisa da Administração Nacional da Aeronáutica e Espaço dos Estados Unidos (NASA), o algoritmo que veio a ser conhecido como *Extended Kalman Filter* [Schmidt].

Esta variação do filtro de Kalman projeta a covariância do sistema não-linear para uma curva Gaussiana utilizando uma técnica de linearização das funções de transição de estado $\mathbf{f}(\mathbf{x})$ e de observação $\mathbf{h}(\mathbf{x})$, que são os componentes do sistema que podem conter não-linearidade [Perea et al. 2007]. Isso é feito a partir do cálculo da Matriz Jacobiana, que é a matriz composta apenas pelas derivadas parciais de cada uma. Por exemplo, para

a função $\mathbf{f}(\mathbf{x})$, isso se dá da seguinte forma:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Realizado este processo para todos os casos onde há não-linearidade, o funcionamento do código ficará da seguinte forma:

- **Atualização do estado**
  Assim como no filtro clássico, esta equação irá ponderar o estado na iteração n estimado na iteração anterior com as medidas mais recentes realizadas pelos sensores.
  $$\hat{\mathbf{x}}_{\mathbf{n,n}} = \hat{\mathbf{x}}_{\mathbf{n,n-1}} + \mathbf{K_n}(\mathbf{z_n} - \mathbf{h}(\hat{\mathbf{x}}_{\mathbf{n,n-1}}))$$
- **Atualização da covariância**
  Cálculo da nova estimativa da covariância, dessa vez sendo realizado com a jacobiana da função $h(x)$ em caso de não-linearidade.
  $$\mathbf{P_{n,n}} = (\mathbf{I} - \mathbf{K_n}\frac{\partial \mathbf{h}}{\partial \mathbf{x}})\mathbf{P_{n,n-1}}(\mathbf{I} - \mathbf{K_n}\frac{\partial \mathbf{h}}{\partial \mathbf{x}})^T + \mathbf{K_n}\mathbf{R_n}\mathbf{K_n^T}$$
- **Atualização do Ganho de Kalman**
  O mesmo acontece para o cálculo do ganho de Kalman, que é atualizado aqui utilizando também a função transposta da jacobiana.
  $$\mathbf{K_n} = \mathbf{P_{n,n-1}}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^{\mathbf{T}}(\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\mathbf{P_{n,n-1}}\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^{\mathbf{T}} + \mathbf{R_n})^{-1}$$
- **Extrapolação do estado**
  A equação de extrapolação de estado, que estima o estado do sistema para a próxima iteração, segue sem modificação.
  $$\hat{\mathbf{x}}_{\mathbf{n+1,n}} = \mathbf{f}(\hat{\mathbf{x}}_{\mathbf{n,n}}) + \mathbf{G}\mathbf{u_n}$$
- **Extrapolação da covariância**
  E por fim, o cálculo da covariância para a próxima iteração utiliza tanto a jacobiana de $f(x)$ quanto sua transposta.
  $$\mathbf{P_{n+1,n}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\mathbf{P_{n,n}}\frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{\mathbf{T}} + \mathbf{Q}$$

2.4. Unscented Kalman Filter

Uma característica do filtro de Kalman estendido evidenciada pela literatura é a sua volatilidade em cenários onde a não-linearidade do sistema é alta. A partir dessa necessidade, Jeffrey Uhlmann propôs em sua tese de doutorado [Julier and Uhlmann 1997] uma variante para o filtro de Kalman que lida com modelos altamente não-lineares, a qual ele batizou de *Unscented Kalman Filter*.

Assim como o filtro estendido, essa variante utiliza linearização para adaptar o problema por um método estatístico chamado *Unscented Transform* (UT). Este processo

de linearização seleciona a média da distribuição do modelo e uma série de pontos localizados a uma certa distância dela, em seguida propaga esses pontos através da função não linear do modelo, gerando um novo conjunto de pontos. Após isso, são calculados pesos para cada ponto gerado, e por fim é calculada uma distribuição Gaussiana aproximada do modelo [Perea et al. 2007].

O processo será detalhado parte por parte a seguir:

- **Seleção dos Sigma Points**
  Estas são as equações propostas por Uhlmann para o cálculo inicial dos *sigma points* para um estado N-dimensional.

$$\chi^{(\mathbf{0})}_{\mathbf{n,n}} = \hat{\mathbf{x}}_{\mathbf{n,n}}$$

$$\chi^{(\mathbf{i})}_{\mathbf{n,n}} = \hat{\mathbf{x}}_{\mathbf{n,n}} + (\sqrt{(\mathbf{N}+\kappa)\mathbf{P}_{\mathbf{n,n}}})_i, i \in [1, N]$$

$$\chi^{(\mathbf{i})}_{\mathbf{n,n}} = \hat{\mathbf{x}}_{\mathbf{n,n}} - (\sqrt{(\mathbf{N}+\kappa)\mathbf{P}_{\mathbf{n,n}}})_i, i \in [N+1, 2N]$$

- **Propagação dos Sigma Points**
  Calculados os pontos, é realizada a sua propagação pela função de transição de estado.

$$\chi_{\mathbf{n+1,n}} = \mathbf{f}(\chi_{\mathbf{n+1,n}})$$

- **Calculo dos pesos dos Sigma Points**
  Em seguida, um peso é atribuído a cada um deles pelas equações abaixo.

$$w_0 = \frac{\chi}{N+\chi}$$

$$w_i = \frac{1}{2}(N+\chi)$$

- **Extrapolação do estado**
  Calculados os *sigma points* e seus respectivos pesos, o próximo estado do sistema pode ser estimado pelo somatório do produto de cada ponto por seu peso.

$$\hat{\mathbf{x}}_{\mathbf{n+1,n}} = \sum_{i=0}^{2N} w_i \chi^{(i)}_{n+1,n}$$

- **Extrapolação da covariância:** De forma similar, é feito o cálculo da covariância do estado.

$$\mathbf{P}_{\mathbf{n+1,n}} = \sum_{i=0}^{2N} w_i (\chi^{(i)}_{n+1,n} - \hat{\mathbf{x}}_{\mathbf{n+1,n}})(\chi^{(i)}_{n+1,n}+)^T$$

- **Atualização do estado**
  Esta equação se mantém parecida com as outras versões do filtro.

$$\hat{\mathbf{x}}_{\mathbf{n,n}} = \hat{\mathbf{x}}_{\mathbf{n,n-1}} + \mathbf{K}_{\mathbf{n}}(z_n - \bar{z}_n)$$

- **Atualização do Ganho de Kalman**
  Cálculo do novo valor do ganho de Kalman a partir da variância cruzada entre o estado estimado e o medido.

$$\mathbf{K_n} = P_{xz_n}(P_{z_n})^{-1}$$

- **Cálculo dos pesos da matriz de covariância**
  Atualização da variância de medição e variância cruzada entre estado-medição.

$$\mathbf{P_{z_n}} = \sum_{\mathbf{i=0}}^{\mathbf{2N}} \mathbf{w_i}(\mathbf{Z_n^{(i)}} - \mathbf{\bar{z}_n})(\mathbf{Z_n^{(i)}} - \mathbf{\bar{z}_n})^{\mathbf{T}} + \mathbf{R_n}$$

$$\mathbf{P_{xz_n}} = \sum_{\mathbf{i=0}}^{\mathbf{2N}} \mathbf{w_i}(\chi_{\mathbf{n,n-1}}^{\mathbf{(i)}} - \mathbf{\hat{x}_{n,n-1}})(\mathbf{Z_n^{(i)}} - \mathbf{\bar{z}_n})^{\mathbf{T}} + \mathbf{R_n}$$

- **Atualização da covariância**
  E por fim, é feito o cálculo da nova covariância do estado.

$$\mathbf{P_{n,n}} = \mathbf{P_{n,n-1}} - \mathbf{K_n P_z K_n^T}$$

3. Objetivos

Considerando a falta de análises que comparem os algoritmos apresentados em contextos onde a capacidade do hardware é limitada, o objetivo desse projeto é propor um estudo e uma comparação de métodos distintos para fusão de sensores de baixo custo para navegação autônoma utilizando a plataforma Arduino UNO. Para isso, será realizada uma revisão teórica dos métodos mais convencionais presentes na literatura (neste caso, os filtros de Kalman não-lineares e linear, juntamente de outros métodos populares encontrados) [Wulandari et al. 2018] aliada a uma implementação completa de um sistema SLAM e uma série de testes práticos para validação.

A ideia é que, ao fim do trabalho, os dados coletados a partir da implementação e teste de cada um dos métodos sejam analisados e comparados entre si de modo que seja possível identificar diferenças significativas de precisão e desempenho a fim de aferir o método mais adequado para operar utilizando sensores de baixa precisão.

Os objetivos específicos do projeto são:

1. Estudar sobre fusão de sensores no contexto de SLAM.
2. Pesquisar sobre trabalhos teóricos e práticos acerca do tema, com enfoque naqueles que abordam otimizações de desempenho.
3. Estudar e avaliar técnicas para lidar com dados altamente imprecisos.
4. Desenvolver um robô funcional utilizando um Arduino UNO como componente principal de computação.
5. Propor uma arquitetura de teste que avalie cada um dos métodos de forma a evidenciar suas respectivas vantagens e desvantagens.
6. Implementar os diferentes algoritmos estudados no sistema físico, assim como uma interface para coleta de dados e monitoramento em tempo real de desempenho.
7. Testar as implementações com diferentes configurações de sensores em um mesmo ambiente controlado e comparar os resultados obtidos.

4. Procedimentos metodológicos

O desenvolvimento deste trabalho seguirá um conjunto de atividades estabelecidas com o intuito de alcançar os objetivos propostos. As atividades estão previstas para serem realizadas durante o período planejado no cronograma da Tabela 1 e descritas abaixo:

1. Revisão bibliográfica sobre fusão de sensores de baixo custo, filtro de Kalman clássico e suas variações para modelos não-lineares, SLAM e otimizações específicas de hardware para os todos os métodos citados;
2. Estudo e levantamento bibliográfico acerca dos tópicos mencionados no Item 1 a partir de trabalhos validados sobre os temas;
3. Análise e avaliação dos sensores mais populares e acessíveis do mercado atualmente, assim como dos demais componentes necessários para estruturar um robô funcional;
4. Implementação do robô;
5. Elaboração de mecanismos para testes metódicos e repetíveis que produzam resultados significativos;
6. Implementação dos algoritmos e de uma interface para acompanhamento e coleta de dados em tempo real;
7. Teste e comparação dos resultados a partir das diferentes versões;
8. Divulgação dos resultados através de publicações.

Atividade	2024					2025	
	Ago	Set	Out	Nov	Dez	Jan	Fev
1	•						
2	•	•					
3		•	•				
4			•	•			
5				•	•		
6					•	•	
7						•	•
8							•

Table 1. Cronograma de execução

5. Contribuições e/ou Resultados esperados

Entre os resultados esperados desse projeto estão: os estudos aprofundados sobre o uso de técnicas de SLAM (Simultaneous Localization and Mapping) em sistemas de navegação autônoma utilizando equipamento de baixo custo; a implementação do sistema de navegação autônoma baseado em SLAM utilizando sensores econômicos; avaliação de desempenho e eficácia dos algoritmos de SLAM em ambientes reais; bem como a implementação dessa proposta.

Espera-se que os métodos e ferramentas desenvolvidos neste projeto resultem em sistemas de navegação autônoma mais acessíveis e eficazes, capazes de operar em plataformas de hardware com recursos limitados, oferecendo soluções práticas e de baixo custo para aplicações de robótica.

Também se espera que o projeto forneça uma base sólida para experimentação e melhorias futuras, fornecendo uma boa indicação dos melhores métodos a serem utilizados em contextos similares.

References

Becker, A. (2022). *Kalman Filter from the Ground Up*. Independently published, 1 edition.

Higgins, W. T. (1975). A comparison of complementary and kalman filtering. *IEEE Transactions on Aerospace and Electronic Systems*, AES-11(3):321–325.

Javaid, M., Haleem, A., Singh, R. P., Rab, S., and Suman, R. (2021). Significance of sensors for industry 4.0: Roles, capabilities, and applications. *Sensors International*, 2:100110.

Julier, S. J. and Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In Kadar, I., editor, *Signal Processing, Sensor Fusion, and Target Recognition VI*, volume 3068, pages 182 – 193. International Society for Optics and Photonics, SPIE.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45.

Mataric, M. J. (2007). *The Robotics Primer*. MIT Press, Cambridge, MA.

Perea, L., How, J., Breger, L., and Elosegui, P. (2007). Nonlinearity in sensor fusion: Divergence issues in ekf, modified truncated gsf, and ukf.

Schmidt, S. F. Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle.

Wulandari, H., Sutrisno, E., and Ando, Y. (2018). Sensor fusion algorithm by complementary filter for attitude estimation of quadrotor with low-cost imu. *TELKOMNIKA (Telecommunication, Computing, Electronics and Control)*, 16(2):868–875.