VOL. 05 JULHO 2025

CHRONO CODE

DA ESTIMATIVA AO GOL: A LÓGICA COMPUTACIONAL POR TRÁS DA JOGADA PERFEITA.

CHRONO CODE

EDITORIAL

DA ESTIMATIVA AO GOL: A LÓGICA COMPUTACIONAL POR TRÁS DA JOGADA PERFEITA

No campo em constante evolução da Inteligência Artificial (IA), a compreensão dos agentes inteligentes é fundamental. Esta quarta edição da ChronoCode se aprofunda nesse componente essencial da IA, explorando seus fundamentos e aplicações avançadas.

Em nossos estudos acadêmicos sobre IA, a exploração dos agentes inteligentes, com foco particular naqueles baseados em conhecimento. Aprofundamos nas bases de conhecimento que sustentam suas operações, distinguindo entre as abordagens procedural e declarativa na forma como esses agentes processam e utilizam informações. A maestria da lógica proposicional e da inferência é crucial para a funcionalidade desses sistemas, capacitando-os a derivar conclusões e tomar decisões informadas.

Embora exemplos como o Wumpus World sejam frequentemente utilizados para ilustrar conceitos fundamentais de agentes baseados em conhecimento, esta edição da ChronoCode apresentará uma análise de aplicações distintas e inovadoras. Exploraremos como a inferência lógica permite a agentes inteligentes realizar tarefas complexas, como diagnósticos médicos automatizados ou a otimização de rotas logísticas.

Prepare-se para uma imersão nas arquiteturas e funcionalidades dos agentes inteligentes, desvendando os mecanismos que governam seu comportamento. Complementando a discussão teórica, ofereceremos um exemplo de implementação em código, demonstrando a transição da teoria para a prática no desenvolvimento de sistemas inteligentes.

1. FUNDAMENTOS DA DECISÃO E INFERÊNCIA SOB INCERTEZA

1.1. TEORIA DA UTILIDADE E A RACIONALIDADE DA DECISÃO

A tomada de decisão em ambientes incertos é o ponto de partida. A Teoria da Decisão, e em particular sua vertente mais influente, a Teoria da Utilidade Esperada (EU), formam o núcleo da teoria econômica e da inteligência artificial para agentes racionais (RUSSELL; NORVIG, 2022; CUSINATO; PORTO JÚNIOR, 2004). O problema fundamental é que a simples maximização do valor monetário esperado pode não capturar adequadamente o comportamento humano ou o objetivo de um agente. Por exemplo, a maioria das pessoas prefere um ganho certo de 1 milhão de reais a uma aposta de 50% de chance de ganhar 2,1 milhões, embora o valor esperado da aposta seja maior.

A Teoria da Utilidade resolve este paradoxo ao postular que agentes racionais buscam maximizar a utilidade esperada, não o valor esperado. A utilidade é uma medida da satisfação ou preferência subjetiva de um indivíduo por um resultado (CUSINATO; PORTO JÚNIOR, 2004). A função de utilidade de um agente, $U(x)$, mapeia resultados x para números reais, de forma que $U(x1)>U(x2)$ se, e somente se, o resultado $x1$ é preferível a $x2$.

Esta teoria permite classificar a atitude de um tomador de decisão em relação ao risco (CUSINATO; PORTO JÚNIOR, 2004):

- Averso ao risco: Possui uma função de utilidade côncava. A utilidade de um ganho certo é maior que a utilidade esperada de uma aposta com o mesmo valor esperado. Este perfil prefere a certeza e tende a se contentar com ganhos menores, mas seguros.

- **Propenso ao risco: Possui uma função de utilidade convexa. A utilidade esperada de uma aposta é maior que a utilidade de um ganho certo equivalente, incentivando a tomada de riscos maiores para obter retornos potencialmente maiores.**
- **Neutro ao risco: Possui uma função de utilidade linear. A utilidade é diretamente proporcional ao valor monetário, e a decisão é baseada puramente no valor esperado.**

O ponto crucial é que a racionalidade, sob esta ótica, não reside em ter uma função de utilidade específica, mas em agir de forma consistente com as próprias preferências. Um agente é racional se suas escolhas maximizam sua própria função de utilidade esperada. Essa aceitação da subjetividade individual como um componente legítimo de um modelo de racionalidade formal é um precursor filosófico direto da interpretação subjetivista da probabilidade, que é central para o paradigma Bayesiano.

1.2. O PARADIGMA BAYESIANO: ATUALIZANDO CRENÇAS COM EVIDÊNCIAS

Uma vez estabelecido um critério para a decisão, a próxima tarefa é quantificar e manipular a própria incerteza. O Teorema de Bayes, formulado por Thomas Bayes no século XVIII e desenvolvido por Pierre-Simon Laplace, fornece a pedra angular para este fim (RUSSELL; NORVIG, 2022). Ele descreve como atualizar a probabilidade de uma hipótese (H) à luz de novas evidências (E). Matematicamente, é expresso como:

$$P(H \mid E) = P(E)P(E \mid H)P(H)$$

Onde:

- **P(H | E) é a probabilidade a posteriori: a probabilidade atualizada da hipótese após a observação da evidência.**

- P(E | H) é a verossimilhança (likelihood): a probabilidade de observar a evidência se a hipótese for verdadeira. Este termo quantifica a força com que a evidência suporta a hipótese.
- P(H) é a probabilidade a priori: a probabilidade inicial da hipótese, baseada no conhecimento ou crença prévia, antes da observação da evidência.
- P(E) é a evidência marginal: a probabilidade total de observar a evidência, somando sobre todas as hipóteses possíveis. Funciona como uma constante de normalização.

A inferência Bayesiana adota uma visão de probabilidade que pode ser subjetiva, interpretando-a como um "grau de crença" (RUSSELL; NORVIG, 2022). Isso contrasta com a visão frequentista, que define a probabilidade como a frequência de longo prazo de um evento (RUSSELL; NORVIG, 2022).

A abordagem Bayesiana é particularmente poderosa porque permite a incorporação formal de conhecimento prévio, algo que os seres humanos fazem intuitivamente. Por exemplo, na prática clínica, um médico combina sua experiência sobre a prevalência de doenças (a priori) com os resultados de um novo exame (evidência) para chegar a um diagnóstico mais preciso (a posteriori) (ASSUNÇÃO et al., 2024).

Nesse sentido, o Teorema de Bayes transcende a estatística para se tornar a codificação matemática do próprio método científico e do raciocínio indutivo. Ele formaliza o processo de formular uma hipótese (a priori), coletar dados (evidência) e refinar a hipótese com base nos dados (a posteriori). É um motor de aprendizado que dita como um agente racional deve alterar suas crenças ao ser confrontado com novas informações, tornando-se um modelo normativo para a aprendizagem em qualquer domínio que lide com incerteza (RUSSELL; NORVIG, 2022).

1.3. REDES BAYESIANAS: MODELANDO DEPENDÊNCIAS COMPLEXAS

Enquanto o Teorema de Bayes é ideal para relacionar duas variáveis, problemas do mundo real frequentemente envolvem dezenas ou centenas de variáveis inter-relacionadas. Modelar a distribuição de probabilidade conjunta de todas essas variáveis é computacionalmente intratável, pois o número de parâmetros cresce exponencialmente. As Redes Bayesianas (RBs) resolvem este problema de forma elegante (LUNA, 2020).

Uma Rede Bayesiana é um modelo gráfico probabilístico que consiste em dois componentes:

- Uma estrutura de grafo: Um Grafo Acíclico Direcionado (DAG), onde os nós representam as variáveis aleatórias do domínio e as arestas direcionadas representam as relações de dependência condicional direta entre elas. A ausência de uma aresta entre dois nós implica uma suposição de independência condicional (LUNA, 2020).
- Um conjunto de parâmetros: Para cada nó, uma Tabela de Probabilidade Condicional (TPC) que quantifica a distribuição de probabilidade daquele nó, condicionada aos seus pais no grafo (LUNA, 2020).

A propriedade fundamental de uma RB é que ela permite fatorar a distribuição de probabilidade conjunta de todas as variáveis como o produto das probabilidades condicionais de cada variável dados os seus pais. Para um conjunto de variáveis X1,...,Xn:

- $P(X1,...,Xn)=i=1\Pi nP(Xi \mid Pais(Xi))$

Esta fatorização reduz drasticamente o número de parâmetros necessários e torna a representação do conhecimento mais compacta e intuitiva (LUNA, 2020).
As RBs formam uma ponte poderosa entre o conhecimento qualitativo, muitas vezes humano, e o conhecimento quantitativo, derivado de dados. A estrutura do grafo (as setas) pode ser desenhada por um especialista de domínio, que codifica suas crenças sobre as relações causais ou de influência no sistema. Por exemplo, um médico pode desenhar um grafo onde "Fumar" aponta para "Câncer de Pulmão", refletindo um conhecimento causal estabelecido (LUNA, 2020).
Em seguida, os parâmetros (as TPCs) podem ser estimados a partir de dados de pacientes, quantificando a força dessas relações.

Essa capacidade de fundir sinergicamente o conhecimento de especialistas com a evidência empírica torna as RBs ferramentas robustas e interpretáveis, o que é crucial para gerar confiança em sistemas de IA, especialmente em domínios críticos como a medicina ou o direito (LUNA, 2020).

1.4. O DESAFIO DA INFERÊNCIA: MÉTODOS EXATOS E APROXIMADOS

Ter um modelo compacto como uma RB é apenas metade da batalha. A tarefa principal é usá-lo para fazer inferências, ou seja, para calcular a distribuição de probabilidade posterior de um conjunto de variáveis de consulta, dadas as observações de outras variáveis (evidências).
Infelizmente, a inferência exata em Redes Bayesianas gerais é um problema NP-difícil (RUSSELL; NORVIG, 2022).

A complexidade computacional cresce exponencialmente com uma medida da "conectividade" do grafo chamada treewidth.
Isso leva a uma bifurcação fundamental nos métodos de inferência:

- Inferência Exata: Algoritmos como a Eliminação de Variáveis calculam a resposta exata. Eles são eficientes e práticos para redes com estruturas simples, como cadeias ou árvores (politrees), mas se tornam intratáveis para redes densamente conectadas (RUSSELL; NORVIG, 2022).
- Inferência Aproximada: Quando a inferência exata é inviável, recorre-se a métodos que fornecem uma resposta aproximada. Estes se dividem principalmente em duas categorias:
  - Métodos Estocásticos (Monte Carlo): Técnicas como a Amostragem de Gibbs e, mais geralmente, Markov Chain Monte Carlo (MCMC), geram um grande número de amostras da distribuição da rede e estimam as probabilidades a partir das frequências dessas amostras (RUSSELL; NORVIG, 2022).
  - Métodos Determinísticos: Técnicas como a Inferência Variacional ou a Propagação de Crenças em Laços (Loopy Belief Propagation) transformam o problema de inferência em um problema de otimização, buscando uma distribuição mais simples que se aproxime da distribuição posterior real (RUSSELL; NORVIG, 2022).).

2. MODELOS DE ESTADO-ESPAÇO PARA DADOS SEQUENCIAIS

Muitos problemas do mundo real envolvem dados que chegam em sequência ao longo do tempo. A análise de séries temporais requer modelos que possam capturar a dinâmica e a evolução do sistema. Esta seção introduz os modelos de estado-espaço como uma especialização do framework Bayesiano para dados sequenciais.

2.1. O PROBLEMA DA ANÁLISE TEMPORAL: FILTRAGEM, PREDIÇÃO E SUAVIZAÇÃO

A inferência em sistemas dinâmicos pode ser categorizada em três tarefas principais, que se distinguem pelo conjunto de informações disponíveis em relação ao estado que se deseja estimar (RUSSELL; NORVIG, 2022).

Seja Xt o estado (oculto) do sistema no tempo t e e1:n={e1 ,e2,...,en} a sequência de evidências (observações) até o tempo n. As tarefas são:

- Filtragem (Filtering): Estimar o estado atual, dado todo o histórico de evidências até o momento presente. A tarefa é calcular a distribuição de probabilidade P(Xt | e1:t). Isso é útil para aplicações em tempo real, como rastrear a posição de um objeto (RUSSELL; NORVIG, 2022).
- Predição (Prediction): Estimar um estado futuro, dado o histórico de evidências até o momento presente. A tarefa é calcular P(Xt+k | e1:t) para algum k>0. Esta é a base para previsão e planejamento (RUSSELL; NORVIG, 2022).

- **Suavização (Smoothing):** Refinar a estimativa de um estado passado, dado todo o conjunto de evidências, incluindo aquelas que ocorreram após o estado em questão. A tarefa é calcular P(Xk | e1:t) para algum k<t. A suavização produz as estimativas mais precisas, pois utiliza mais informações (RUSSELL; NORVIG, 2022).

A distinção entre essas tarefas revela uma característica fundamental da inferência temporal: o valor e a certeza de uma estimativa são diretamente proporcionais ao "horizonte de informação" disponível. A predição é inerentemente a mais incerta, pois projeta-se no desconhecido com base apenas no passado.

A filtragem é mais certa, pois ancora a estimativa presente com a evidência presente. A suavização é a mais certa de todas, pois permite que a "sabedoria do futuro" (evidências posteriores) corrija as imprecisões das estimativas passadas. Os algoritmos desenvolvidos para modelos de estado-espaço, como o algoritmo Forward-Backward para HMMs ou o suavizador de Rauch-Tung-Striebel para Filtros de Kalman, são, em essência, mecanismos para propagar crenças (informação) para frente e para trás ao longo do eixo do tempo, a fim de responder a essas três questões canônicas.

2.2. MODELOS OCULTOS DE MARKOV (HMMS) PARA ESTADOS DISCRETOS

Quando o estado oculto de um sistema dinâmico é discreto (e.g., "chuvoso" ou "ensolarado", "mercado em alta" ou "mercado em baixa"), o Modelo Oculto de Markov (HMM) é a ferramenta de modelagem padrão (SOUZA, 2009). Um HMM é, efetivamente, uma Rede Bayesiana dinâmica com uma estrutura específica: uma cadeia de variáveis de estado ocultas, X_t, que satisfazem a propriedade de Markov de primeira ordem (o estado futuro X_{t+1} depende apenas do estado presente X_t) e uma sequência de variáveis de observação, E_t, onde cada observação depende apenas do estado atual correspondente (SOUZA, 2009).

2.2.1. O MODELO DE TRANSIÇÃO E O MODELO DE SENSORES

A arquitetura de um HMM é definida por um conjunto de parâmetros que separam a dinâmica do sistema do processo de observação (SOUZA, 2009; MARQUES, 2007):

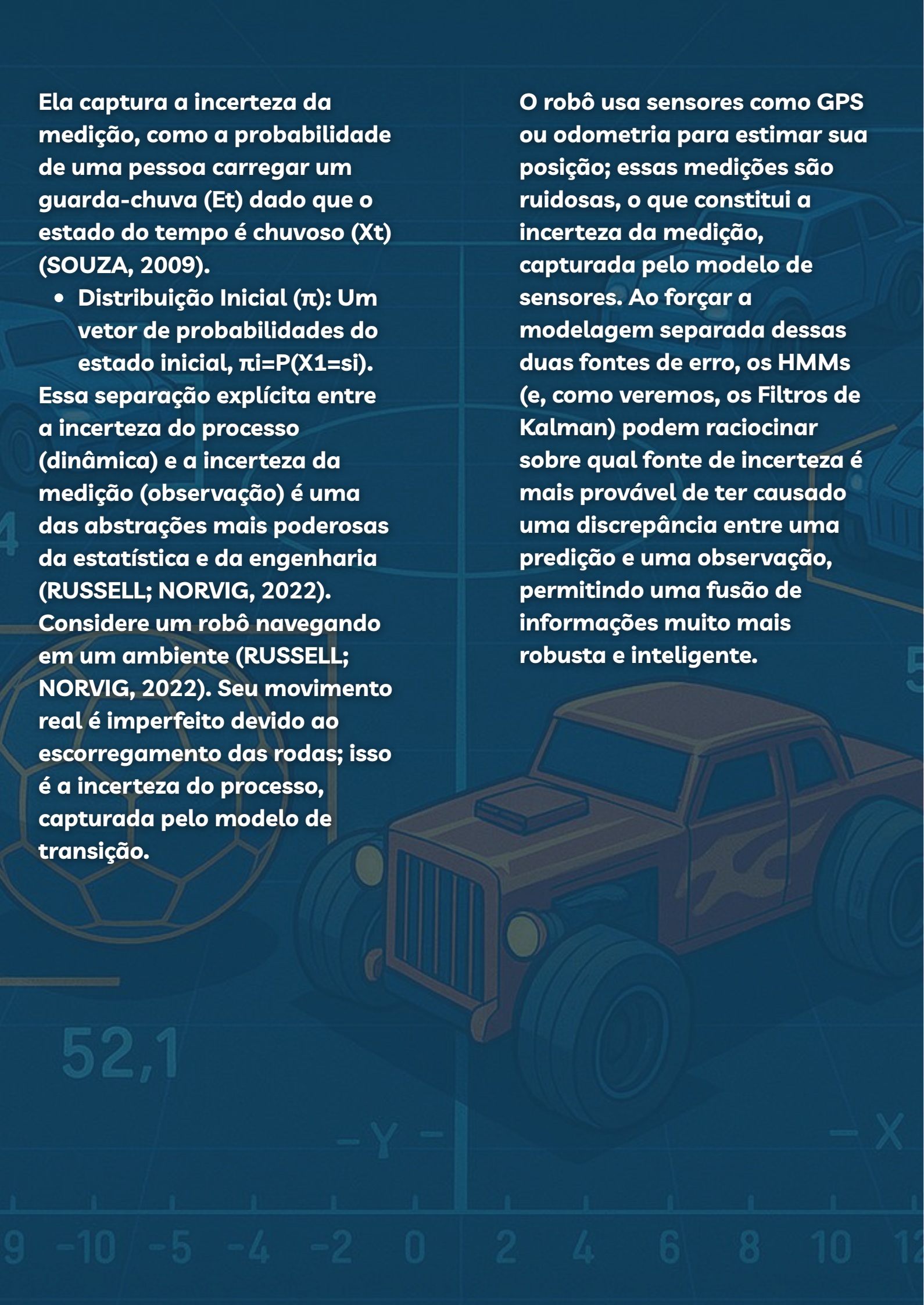
- Modelo de Transição (A): Uma matriz de probabilidades de transição de estado, $A_{ij}=P(X_t=s_j \mid X_{t-1}=s_i)$. Esta matriz descreve "como o mundo evolui", governando a dinâmica intrínseca do sistema. Ela captura a incerteza do processo, como a probabilidade de um dia ensolarado ser seguido por um dia chuvoso.
- Modelo de Sensores ou Emissão (B): Uma matriz de probabilidades de observação, $B_{jk}=P(E_t=o_k \mid X_t=s_j)$. Esta matriz descreve "como percebemos o mundo", relacionando os estados ocultos às observações ruidosas.

Ela captura a incerteza da medição, como a probabilidade de uma pessoa carregar um guarda-chuva (Et) dado que o estado do tempo é chuvoso (Xt) (SOUZA, 2009).

- Distribuição Inicial (π): Um vetor de probabilidades do estado inicial, $\pi i = P(X1 = si)$.

Essa separação explícita entre a incerteza do processo (dinâmica) e a incerteza da medição (observação) é uma das abstrações mais poderosas da estatística e da engenharia (RUSSELL; NORVIG, 2022). Considere um robô navegando em um ambiente (RUSSELL; NORVIG, 2022). Seu movimento real é imperfeito devido ao escorregamento das rodas; isso é a incerteza do processo, capturada pelo modelo de transição.

O robô usa sensores como GPS ou odometria para estimar sua posição; essas medições são ruidosas, o que constitui a incerteza da medição, capturada pelo modelo de sensores. Ao forçar a modelagem separada dessas duas fontes de erro, os HMMs (e, como veremos, os Filtros de Kalman) podem raciocinar sobre qual fonte de incerteza é mais provável de ter causado uma discrepância entre uma predição e uma observação, permitindo uma fusão de informações muito mais robusta e inteligente.



2.2.1. O MODELO DE TRANSIÇÃO E O MODELO DE SENSORES

- Algoritmo de Viterbi: Resolve o problema da decodificação, encontrando a sequência de estados ocultos única e mais provável que poderia ter gerado a sequência de observações observada. Em vez de somar as probabilidades de todos os caminhos que chegam a um estado (como o algoritmo Forward), ele mantém apenas o caminho de maior probabilidade (SOUZA, 2009).

Algoritmo de Baum-Welch (ou Forward-Backward): Resolve o problema do aprendizado, estimando os parâmetros do modelo (A, B, π) a partir de um conjunto de sequências de observações.

2.2.2. ALGORITMOS MATRICIAIS PARA INFERÊNCIA EM HMMS

Dada a estrutura do HMM, os três problemas canônicos de inferência (avaliação, decodificação e aprendizado) são resolvidos de forma eficiente usando algoritmos de programação dinâmica que exploram a estrutura da cadeia para evitar a enumeração exponencial de sequências de estados (SOUZA, 2009).

- Algoritmo Forward: Resolve o problema da avaliação, calculando a probabilidade de uma sequência de observações, P(e1:t). Ele o faz de forma recursiva, calculando a probabilidade conjunta da sequência de observações e do estado final, P(e1:t,Xt). Este algoritmo é a base para a tarefa de filtragem (SOUZA, 2009).

É uma instância do algoritmo Expectation-Maximization (EM), que itera entre um passo E (Expectation), onde usa os algoritmos Forward e Backward para calcular as probabilidades esperadas de transições e emissões, e um passo M (Maximization), onde atualiza os parâmetros do modelo para maximizar essas expectativas (SOUZA, 2009).

É crucial entender a distinção sutil, porém fundamental, entre o que os algoritmos de Viterbi e Forward-Backward calculam. O Viterbi fornece a *explicação mais provável* como um todo, encontrando a sequência de estados globalmente ótima (SOUZA, 2009). O Forward-Backward, por outro lado, é usado para calcular a crença marginal sobre cada estado em cada instante de tempo, somando sobre todos os caminhos possíveis que passam por aquele estado.

Uma consequência disso é que a sequência de estados mais provável (saída do Viterbi) não é necessariamente composta pelos estados que são individualmente mais prováveis em cada instante. A escolha do algoritmo depende da aplicação: para reconhecimento de fala, deseja-se a sequência de palavras mais provável (Viterbi); para estimar a probabilidade de recessão em um trimestre específico, deseja-se a crença marginal sobre o estado "recessão" naquele trimestre (filtragem/suavização via Forward-Backward).

3. O FILTRO DE KALMAN: INFERÊNCIA ÓTIMA EM SISTEMAS LINEARES-GAUSSIANOS

Quando o espaço de estados e as observações de um sistema dinâmico são contínuos, o Filtro de Kalman (FK) emerge como a solução canônica. Desenvolvido por Rudolf E. Kálmán na década de 1960, o filtro é um algoritmo recursivo que fornece uma estimativa ótima do estado de um sistema linear sujeito a ruído Gaussiano (KALMAN, 1960).

3.1. ARQUITETURA E FORMULAÇÃO DO FILTRO

O Filtro de Kalman pode ser visto como a contraparte de variável contínua do HMM. Ele assume que o sistema pode ser descrito por um modelo de estado-espaço linear (FETTER, 2002; KALMAN, 1960):

- Equação de Estado (Modelo de Transição): Descreve como o estado do sistema $x_k \in \mathbb{R}^n$ evolui de um instante de tempo $k-1$ para k.

$$x_k = F_k x_{k-1} + B_k u_k + w_k$$

- Onde F_k é a matriz de transição de estado, B_k é a matriz de controle, u_k é o vetor de controle (entrada), e w_k é o ruído do processo. Assume-se que w_k é um ruído branco Gaussiano com média zero e matriz de covariância Q_k, ou seja, $w_k \sim N(0, Q_k)$. Q_k representa a incerteza na dinâmica do sistema.
- Equação de Medição (Modelo de Sensor): Descreve como o estado oculto x_k se relaciona com a medição observável $z_k \in \mathbb{R}^m$.

- $z_k = H_k x_k + v_k$

- Onde H_k é a matriz de observação, e v_k é o ruído da medição. Assume-se que v_k também é um ruído branco Gaussiano com média zero e matriz de covariância R_k, ou seja, $v_k \sim N(0, R_k)$. R_k representa a incerteza nos sensores (SILVA, 2024).

O objetivo do filtro é estimar o estado x_k com base no modelo e na sequência de medições ruidosas $z_1, z_2, \ldots, z_k$.

3.2. O CICLO RECURSIVO DE PREDIÇÃO E CORREÇÃO

O coração do Filtro de Kalman é seu ciclo de duas etapas, que se repete a cada nova medição. Este ciclo é uma implementação elegante e eficiente da inferência Bayesiana para o modelo linear-gaussiano (SILVA, 2024).

Etapa 1: Predição (Priori)
Nesta fase, o filtro usa o modelo de estado para prever o estado e sua incerteza no próximo instante de tempo, antes de incorporar a nova medição.

- **Predição do Estado:** $\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k$
- **Predição da Covariância do Erro:** $P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$

O estado predito, $\hat{x}_{k|k-1}$, e sua covariância, $P_{k|k-1}$, constituem a crença a priori sobre o estado no tempo k. A covariância aumenta nesta etapa devido à adição do ruído do processo Q_k, refletindo que a incerteza cresce com o tempo na ausência de novas informações.
Etapa 2: Atualização ou Correção (Posteriori)
Nesta fase, o filtro usa a medição atual, z_k, para corrigir a predição.

- Cálculo do Ganho de Kalman (Kk):
- Kk=Pk | k−1HkT(HkPk | k−1 HkT+Rk)−1

- Atualização do Estado:
- x^k | k=x^k | k−1+Kk(zk−Hk x^k | k−1)

- Atualização da Covariância do Erro:
- Pk | k=(I−KkHk)Pk | k−1

O termo (zk−Hkx^k | k−1) é chamado de resíduo ou inovação da medição; ele representa a "surpresa", a diferença entre a medição real e a medição que o filtro esperava. O Ganho de Kalman, Kk, é a peça central do filtro. Ele é uma matriz de ponderação que determina o quanto a predição deve ser corrigida com base na inovação (KALMAN, 1960). O valor de Kk é calculado para minimizar a covariância do erro a posteriori, Pk | k.

A análise do Ganho de Kalman revela a natureza Bayesiana do filtro. Se a incerteza da medição, Rk, for muito alta em comparação com a incerteza da predição, Pk | k−1, o ganho Kk será pequeno, e o filtro confiará mais em sua própria predição. Inversamente, se a incerteza da medição for baixa, o ganho será alto, e o filtro dará mais peso à nova e confiável medição. O estado atualizado, x^k | k, e sua covariância, Pk | k, representam a crença a posteriori, que combina otimamente a crença a priori com a evidência da nova medição para produzir a estimativa com a menor incerteza possível. O Filtro de Kalman, portanto, não é meramente um "filtro de sinal", mas uma instância recursiva e computacionalmente brilhante da Regra de Bayes aplicada a sistemas dinâmicos.

3.3. VANTAGENS, LIMITAÇÕES E APLICAÇÕES PRÁTICAS

O Filtro de Kalman possui um conjunto de propriedades que o tornam extremamente atraente para aplicações práticas.

Vantagens:

- Otimalidade: Para sistemas lineares com ruído Gaussiano, o Filtro de Kalman é o estimador ótimo no sentido de que minimiza a variância do erro de estimação (erro quadrático médio) (HARVEY, 1989).
- Eficiência Computacional: Sendo um algoritmo recursivo, ele não precisa armazenar todo o histórico de medições, apenas a última estimativa de estado e sua covariância. Isso o torna ideal para sistemas embarcados e aplicações em tempo real (FETTER, 2002).
- Robustez e Predição: Sua capacidade de modelar explicitamente o ruído o torna robusto a medições imprecisas e sua etapa de predição é crucial para sistemas de controle que precisam antecipar o comportamento do sistema (FETTER, 2002).

Limitações:

- Suposições Restritivas: As suposições de linearidade do sistema e de normalidade do ruído são suas maiores fraquezas, pois muitos sistemas do mundo real são não-lineares (SILVA, 2024; FETTER, 2002).
- Sensibilidade ao Modelo: O desempenho do filtro é altamente dependente da precisão dos modelos (Fk ,Hk) e das matrizes de covariância do ruído (Qk,Rk). Erros de modelagem ou uma inicialização inadequada podem levar à divergência, onde a estimativa se afasta da realidade (FETTER, 2002).

Aplicações:

Apesar de suas limitações, o FK é uma das ferramentas algorítmicas mais amplamente utilizadas na engenharia e tecnologia. Suas aplicações incluem:

- **Navegação e Aeroespacial: É o coração dos sistemas de GPS, navegação inercial (INS) e rastreamento de satélites e mísseis (FETTER, 2002; KALMAN, 1960).**
- **Robótica: Usado para localização, mapeamento (SLAM - Simultaneous Localization and Mapping) e fusão de dados de múltiplos sensores (e.g., odometria, IMU, LiDAR) (FETTER, 2002; RUSSELL; NORVIG, 2022).**
- **Economia e Finanças: Aplicado na análise de séries temporais, estimação de parâmetros variantes no tempo como o Beta de um ativo, e modelagem da volatilidade (HARVEY, 1989).**
- **Visão Computacional: Utilizado para rastreamento de objetos em sequências de vídeo (KALMAN, 1960).**

4. EXTENSÕES DO FILTRO DE KALMAN PARA SISTEMAS NÃO-LINEARES

A principal limitação do Filtro de Kalman é sua suposição de linearidade. Para superar isso, diversas variantes foram desenvolvidas. As duas mais proeminentes são o Filtro de Kalman Estendido (EKF) e o Filtro de Kalman Unscented (UKF).

4.1. A ABORDAGEM POR LINEARIZAÇÃO: O FILTRO DE KALMAN ESTENDIDO (EKF)

O Filtro de Kalman Estendido (EKF) aborda a não-linearidade através de uma estratégia de linearização local. Em vez das equações lineares, o EKF lida com funções não-lineares para a transição de estado e a medição:

- $x_k = f(x_{k-1}, u_k) + w_k$
- $z_k = h(x_k) + v_k$

A ideia do EKF é, a cada passo de tempo, aproximar essas funções não-lineares por uma expansão de Série de Taylor de primeira ordem em torno da estimativa de estado atual. Isso resulta em matrizes Jacobianas (matrizes de derivadas parciais) que substituem as matrizes Fk e Hk do filtro linear (FETTER, 2002; PEREIRA, 2020).

O EKF segue o mesmo ciclo de predição-correção, mas com as matrizes de transição e observação sendo as Jacobianas de f e h, calculadas na estimativa mais recente. No entanto, essa abordagem tem desvantagens significativas:

- O cálculo das Jacobianas pode ser analiticamente complexo e propenso a erros de implementação.
- A aproximação linear pode ser muito pobre se o sistema for altamente não-linear, levando a um desempenho ruim ou até mesmo à divergência do filtro (FETTER, 2002).
- O EKF não oferece garantias de otimalidade ou estabilidade (FETTER, 2002).

A falha fundamental do EKF reside no fato de que ele aproxima a função, não a distribuição de probabilidade. Uma propriedade crucial de sistemas não-lineares é que, quando uma distribuição de probabilidade Gaussiana passa por uma função não-linear, a distribuição resultante geralmente não é mais Gaussiana. Ela pode se tornar assimétrica, multimodal ou distorcida de outras formas.

O EKF ignora essa realidade: ele propaga a média da Gaussiana através da função não-linear e, em seguida, usa a função linearizada para propagar a covariância, assumindo que o resultado ainda é perfeitamente Gaussiano (FETTER, 2002). O erro do EKF, portanto, não vem apenas da linearização em si, mas de sua incapacidade de capturar como a distribuição de probabilidade inteira é transformada pela não-linearidade. Essa limitação motiva diretamente a busca por uma abordagem superior.

4.2. A ABORDAGEM POR AMOSTRAGEM DETERMINÍSTICA: O FILTRO DE KALMAN UNSCENTED (UKF)

O Filtro de Kalman Unscented (UKF) adota uma filosofia radicalmente diferente. Baseia-se no princípio de que é mais fácil aproximar uma distribuição de probabilidade do que aproximar uma função não-linear (WAN; VAN DER MERWE, 2000). Em vez de linearizar a função, o UKF utiliza a Transformada Unscented (UT) para capturar a propagação da média e da covariância através das funções não-lineares (PEREIRA, 2020).

O processo funciona da seguinte maneira:

- Seleção de Pontos Sigma: Em vez de usar uma única estimativa, o UKF seleciona um conjunto mínimo de pontos de amostra determinísticos, chamados pontos sigma, em torno da média da estimativa de estado atual. Esses pontos são escolhidos de tal forma que sua média, covariância e, possivelmente, momentos de ordem superior, correspondam exatamente aos da distribuição Gaussiana do estado (WAN; VAN DER MERWE, 2000).

- Propagação: Cada um desses pontos sigma é propagado individualmente através da função não-linear exata (f ou h). Nenhum cálculo de Jacobiana é necessário.

- Reconstrução: A média e a covariância da distribuição resultante são então calculadas a partir da média ponderada e da covariância dos pontos sigma transformados. O resultado é uma nova aproximação Gaussiana que captura com muito mais precisão o efeito da não-linearidade (WAN; VAN DER MERWE, 2000).

O UKF é geralmente mais preciso e robusto que o EKF, especialmente para sistemas com não-linearidades severas, e seu custo computacional é comparável (PEREIRA, 2020; WAN; VAN DER MERWE, 2000). Ele representa uma solução de engenharia elegante que se situa entre a aproximação grosseira do EKF e o alto custo computacional de um filtro de Monte Carlo completo (como um Filtro de Partículas). Ao propagar a informação estatística (representada pela nuvem de pontos sigma) através da função exata, o UKF captura os benefícios da amostragem — uma estimativa mais precisa da estatística da distribuição transformada — sem o fardo computacional da amostragem aleatória, superando o EKF na grande maioria dos cenários práticos.

REFERÊNCIAS

- ASSUNÇÃO, L. M. de; et al. Método de Bayes como base na tomada de decisão médica - uma revisão bibliográfica. Revista Saúde e desenvolvimento, Curitiba, v. 13, n. 35, p. 1-13, 2024.
- CUSINATO, R. T.; PORTO JÚNIOR, S. S. A teoria da decisão sob incerteza e a hipótese da utilidade esperada. Textos para Discussão FEE, Porto Alegre, n. 11, p. 1-28, 2004.
- FETTER, C. H. O Filtro de Kalman. Porto Alegre: Universidade Federal do Rio Grande do Sul, 2002. Disponível em: http://www.ece.ufrgs.br/~fetter/cca99006/kalman.pdf. Acesso em: 10 ago. 2024.
- HARVEY, A. C. Forecasting, structural time series models and the Kalman filter. Cambridge: Cambridge University Press, 1989.
- KALMAN, R. E. A New Approach to Linear Filtering and Prediction Problems. Journal of Basic Engineering, v. 82, n. 1, p. 35-45, 1960.
- LUNA, Oi. Redes Bayesianas. Medium, 2020. Disponível em: https://medium.com/oiluna/redes-bayesianas-fcf35516dedb. Acesso em: 10 ago. 2024.
- MARQUES, V. S. Hidden Markov Models (HMMs). Rio de Janeiro: Universidade Federal do Rio de Janeiro, 2007. Disponível em: https://www.gta.ufrj.br/grad/07_2/viviane/HiddenMarkovModels-HMMs.html. Acesso em: 10 ago. 2024.
- PEREIRA, A. F. C. O Filtro de Kalman Unscented aplicado na estimação de estados de carga e de saúde de baterias de íon-lítio. João Pessoa: Universidade Federal da Paraíba, 2020.
- RUSSELL, S.; NORVIG, P. Inteligência Artificial: Uma Abordagem Moderna. 4. ed. Rio de Janeiro: LTC, 2022.

- SILVA, F. C. da. Filtros de Kalman. Évora: Universidade de Évora. Disponível em: https://home.uevora.pt/~fc/notas/FiltrosKalman.html. Acesso em: 10 ago. 2024.
- SOUZA, V. P. de. Modelos Ocultos de Markov. Rio de Janeiro: Universidade Federal do Rio de Janeiro, 2009. Disponível em: https://www.gta.ufrj.br/grad/09_1/versao-final/impvocal/hmms.html. Acesso em: 10 ago. 2024.
- WAN, E. A.; VAN DER MERWE, R. The unscented Kalman filter for nonlinear estimation. In: Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium. Lake Louise, Alberta, Canada, 2000. p. 153-158.

EXEMPLO PRÁTICO

O Teste da Nova Câmera da Equipe TITANS

A competição está acirrada na Liga de Futebol de Carros Autônomos (LFCA)! Sua equipe, a TITANS, está na vanguarda da tecnologia, mas enfrenta um desafio clássico: o sistema de visão global, composto por câmeras que monitoram todo o campo, fornece as posições dos carros-robôs e da bola de energia com um certo nível de ruído. Essas pequenas imprecisões, ou "tremores", nas coordenadas podem fazer com que os algoritmos de controle tomem decisões subótimas, resultando em chutes imprecisos e movimentos hesitantes.

seu desafio:

Como engenheiro(a) de software da equipe, sua missão é implementar e comparar dois métodos de filtragem para suavizar os dados de posição da bola recebidos do sistema de visão da Liga. O objetivo é obter uma estimativa mais estável e precisa da trajetória da bola, permitindo que os carros-robôs ajam com mais inteligência e rapidez.

Você irá:

1. **Receber os dados de posição (x, y) da bola diretamente do sistema de visão da LFCA.**
2. **Aplicar em tempo real um Filtro de Média Móvel e um Filtro de Kalman.**
3. **Salvar os dados brutos e os dados filtrados em um arquivo .csv para análise posterior, com as colunas: raw_x, raw_y, moving_avg_x, moving_avg_y, kalman_x e kalman_y.**

Vamos analisar o código que realiza essa tarefa.

SOLUÇÃO PASSO A PASSO

A solução é dividida em dois arquivos principais: filters.py, que define nossas lógicas de filtragem, e main.py, que captura os dados e aplica os filtros.

ARQUIVO 1: FILTERS.PY AS FERRAMENTAS DE FILTRAGEM

Este arquivo contém as classes e funções que fazem o trabalho pesado de suavização dos dados.

```
from collections import deque
import math
import numpy as np
from filterpy.kalman import KalmanFilter

def moving_average_position(history,
window_size=7):
    if len(history) == 0:
        return None
    avg_x = sum(pos[0] for pos in history) /
len(history)
    avg_y = sum(pos[1] for pos in history) /
len(history)
    return (avg_x, avg_y)
```

```
# Classe para o Filtro de Kalman
class KalmanFilter2D:
    def __init__(self):
        # Inicializa o filtro de Kalman
        self.kf = KalmanFilter(dim_x=4, dim_z=2)
        self.kf.x = np.array([0., 0., 0., 0.])  #
Posição inicial [x, y, vx, vy]
        self.kf.F = np.array([[1., 0., 1., 0.],
# Matriz de transição
                              [0., 1., 0., 1.],
                              [0., 0., 1., 0.],
                              [0., 0., 0., 1.]])
        self.kf.H = np.array([[1., 0., 0., 0.],
# Matriz de observação
                              [0., 1., 0., 0.]])
        self.kf.P *= 1000.  # Covariância inicial
        self.kf.R = np.array([[10., 0.], [0.,
10.]])  # Covariância do ruído de medição
        self.kf.Q = np.array([[0.5, 0., 0., 0.],
# Covariância do processo
                              [0., 0.5, 0., 0.],
                              [0., 0., 0.5, 0.],
                              [0., 0., 0., 0.5]])

    def update(self, position):
```

```
        # Atualiza o estado do filtro de Kalman
com a nova posição
        self.kf.predict()
        self.kf.update(np.array(position))  #
Converte a posição para array NumPy
        return self.kf.x[:2]  # Retorna a posição
suavizada (x, y)
```

Análise do filters.py:

- **moving_average_position: Esta é a nossa abordagem mais simples. A função recebe um histórico de posições e simplesmente calcula a média das coordenadas x e y. É rápido e eficaz para ruídos simples, mas pode introduzir um atraso na percepção da posição real.**
- **KalmanFilter2D: Aqui está a nossa solução avançada.**
- **__init__(self): O construtor inicializa o filtro. Ele usa a biblioteca filterpy.**
- **dim_x=4: Nosso "estado" tem 4 dimensões: posição x, posição y, velocidade vx e velocidade vy.**
- **dim_z=2: Nossa "medição" (o que a câmera nos dá) tem 2 dimensões: apenas x e y.**
- **self.kf.F: A matriz de transição de estado. Ela define nosso modelo de movimento: nova_posicao = posicao_antiga + velocidade.**

- self.kf.H: A matriz de observação. Ela extrai do estado apenas o que podemos medir (a posição).
- self.kf.R: A covariância do ruído de medição. Este valor diz ao filtro o quão "barulhenta" ou incerta é a medição da câmera.
- self.kf.Q: A covariância do ruído do processo. Isso modela a incerteza do nosso próprio modelo de movimento (por exemplo, a bola pode sofrer uma aceleração que não previmos).
- update(self, position): Este método é chamado a cada nova medição da câmera. Ele executa os dois passos clássicos do Filtro de Kalman: predict() (prever onde a bola deveria estar) e update() (corrigir a previsão com a medição real).

ARQUIVO 2: MAIN.PY - O ORQUESTRADOR

- Este script conecta todas as peças: captura os dados da SSL, chama os filtros e salva os resultados.

```
import sslclient
import time
import csv
from collections import deque
from filters import moving_average_position,
KalmanFilter2D

# Configurações do cliente SSL
c = sslclient.client(ip='224.5.23.2', port=10006)
c.connect()

# Configurações da média móvel
ball_position_history = deque(maxlen=5)

# Inicializa o filtro de Kalman
kalman_filter = KalmanFilter2D()

# Abrir arquivo CSV para registrar os dados
with open('ball_positions.csv', mode='w') as
file:
    writer = csv.writer(file)
    writer.writerow(['time', 'raw_x', 'raw_y',
'moving_avg_x', 'moving_avg_y', 'kalman_x',
'kalman_y'])
```

```
 while True:
 # Receber dados do cliente
 data = c.receive()
 current_time = time.time()

 if data.HasField('detection'):
 detection = data.detection

 # Acessa a posição da bola
 ball_position = (detection.balls[0].x,
detection.balls[0].y) if detection.balls else
None

 if ball_position:
 # Suavização pela média móvel
 ball_position_history.append(ball_position)
 moving_avg_position =
moving_average_position(ball_position_history)

 # Suavização pelo filtro de Kalman
 kalman_position =
kalman_filter.update(ball_position)
```

```
 # Registrar dados no CSV
 writer.writerow([
 current_time,
 ball_position[0], ball_position[1], # Posições
brutas
 moving_avg_position[0], moving_avg_position[1],


# Posição pela média móvel
 kalman_position[0], kalman_position[1] # Posição
pelo filtro de Kalman
 ])

 print(f"Raw: {ball_position}, Moving Avg:
{moving_avg_position}, Kalman:
{kalman_position}")
```

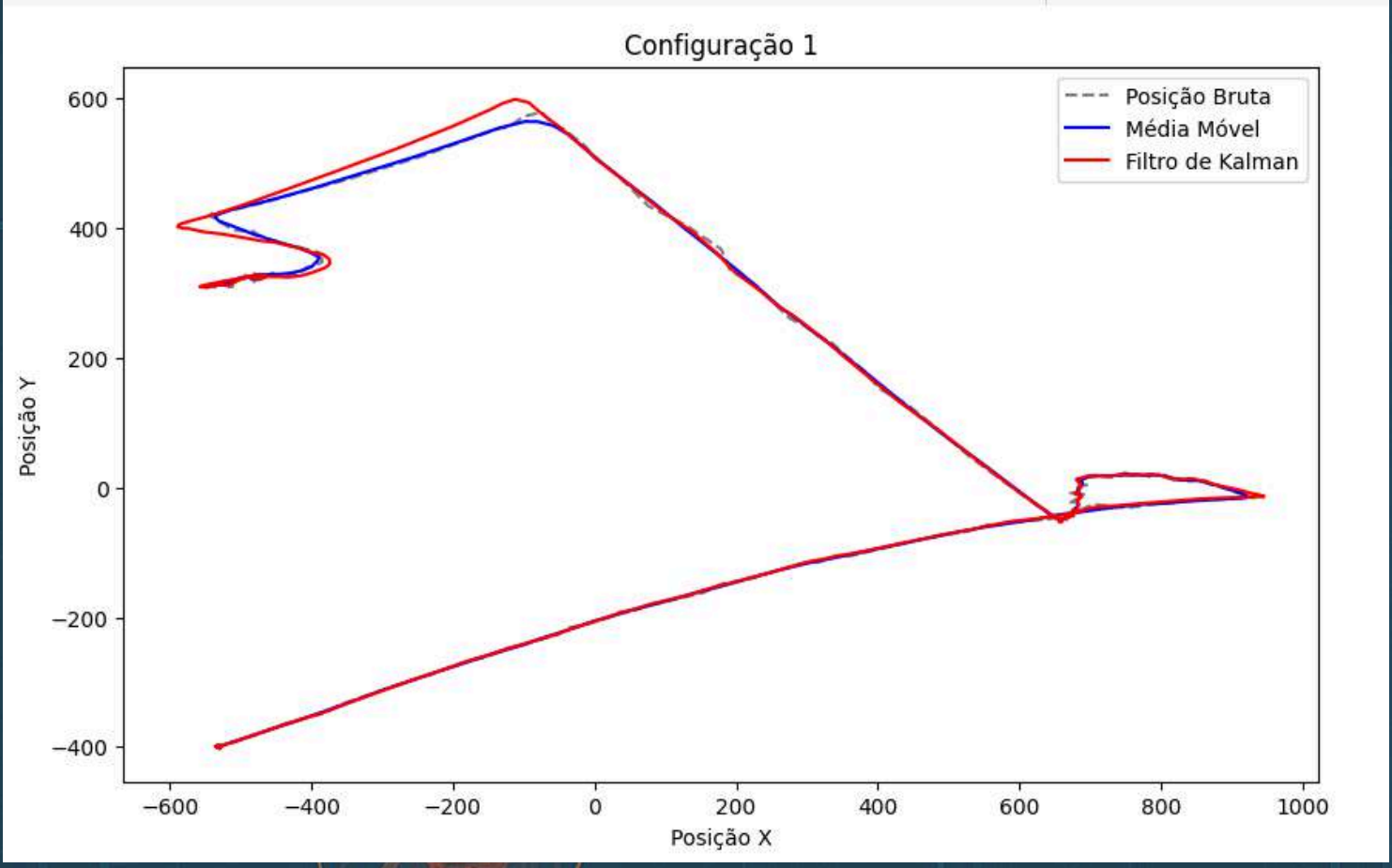
Análise do main.py:

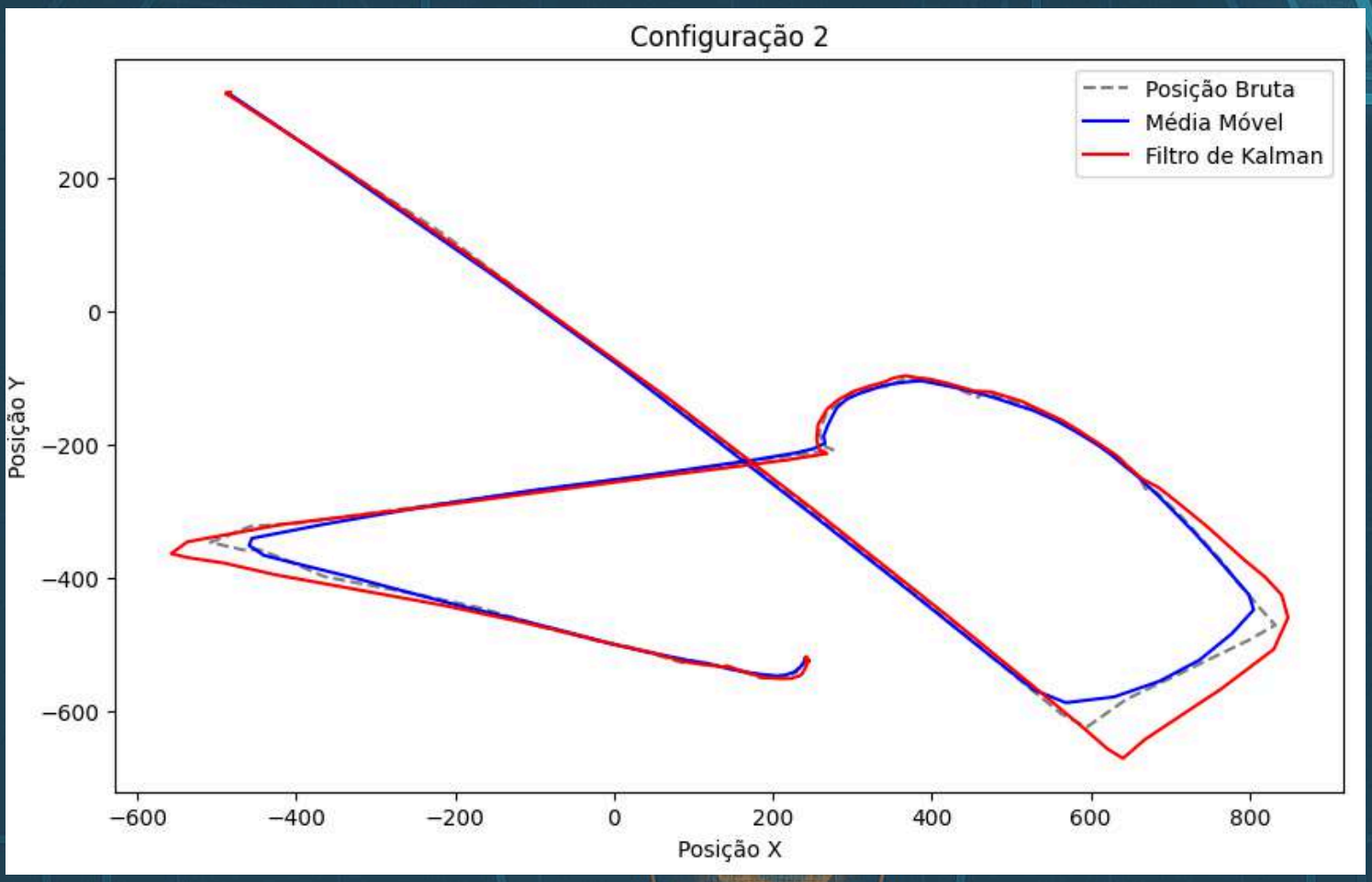
- **Conexão: O script começa se conectando ao endereço de multicast (224.5.23.2:10006) onde o sistema de visão da SSL (como o SSL-Vision) transmite os dados do campo.**
- **Inicialização:**
- **ball_position_history = deque(maxlen=5): Um deque é uma lista otimizada para adicionar e remover itens de ambas as extremidades. Usamos um com tamanho máximo de 5 para guardar apenas as últimas 5 posições da bola para o filtro de média móvel.**

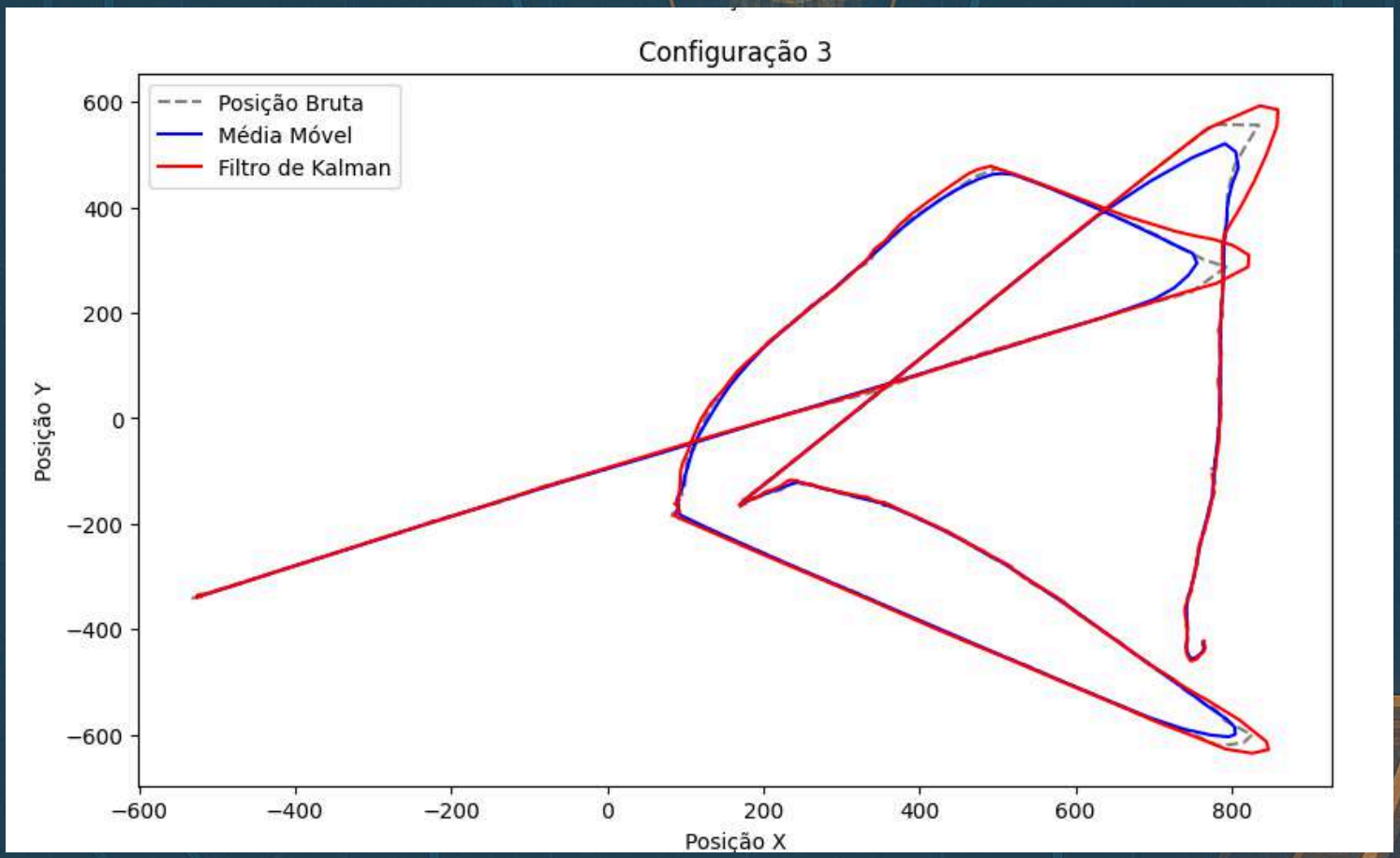
- kalman_filter = KalmanFilter2D(): Uma instância do nosso filtro de Kalman é criada.
- Loop Principal: O while True garante que o código rode continuamente.
- data = c.receive(): Aguarda e recebe um novo "pacote" de dados de visão.
- if data.HasField('detection'): Verifica se o pacote contém informações de detecção (robôs, bola).
- ball_position = ...: Extrai as coordenadas x e y da bola.
- Aplicação dos Filtros: Se a bola foi detectada, o script aplica os dois filtros aos dados brutos (ball_position).
- Salvando os Dados: A linha mais importante para nossa análise. writer.writerow([...]) salva o timestamp, as coordenadas brutas, as coordenadas da média móvel e as coordenadas do filtro de Kalman no arquivo ball_positions.csv.

Análise dos Resultados

Com os dados em formato .csv podemos analisar eles para chegar em certas comncluções como por exemplo, podemos criam um gráfico comparando as coordenadas para ter uma ideia da precição como as imagens abaixo:




Configuração 2
Posição Bruta
Média Móvel
Filtro de Kalman
Posição Y
Posição X
200
0
−200
−400
−600
−600
−400
−200
0
200
400
600
800


Configuração 3
Posição Bruta
Média Móvel
Filtro de Kalman
Posição Y
Posição X
600
400
200
0
−200
−400
−600
−600
−400
−200
0
200
400
600
800

Fica muito difícil saber qual método está mais correto olhando apenas para os gráficos outro método é analisar e comparar o Erro Quadrático Médio
e o Erro Médio Absoluto para ver a performance de cada filtro :

Configuração 1 -
Média Móvel: MAE: 16.9356, RMSE: 32.6964
Filtro de Kalman: MAE: 6.0883, RMSE: 12.6835
Análise: Nesta configuração, o Filtro de Kalman foi drasticamente superior. Seu erro médio (MAE) foi quase 3 vezes menor que o da Média Móvel, e o RMSE foi 2.5 vezes menor.

Configuração 2 -
Média Móvel: MAE: 55.6710, RMSE: 107.4572 Configuração 2
Filtro de Kalman: MAE: 15.9975, RMSE: 32.3575
Análise: A superioridade do Filtro de Kalman é ainda mais evidente aqui. O erro da Média Móvel disparou, provavelmente devido a movimentos mais bruscos e rápidos. O Kalman manteve um erro controlado, com um MAE aproximadamente 3.5 vezes menor.

Configuração 3
Média Móvel: MAE: 41.1839, RMSE: 69.8673 Configuração 3
Filtro de Kalman: MAE: 11.0840, RMSE: 22.8195

Configuração 3
Média Móvel: MAE: 41.1839, RMSE: 69.8673 Configuração 3
Filtro de Kalman: MAE: 11.0840, RMSE: 22.8195
Análise: Novamente, o Filtro de Kalman demonstra um desempenho muito melhor, com um MAE cerca de 3.7 vezes menor e um RMSE 3 vezes menor que a Média Móvel.

conclusão

O Filtro de Kalman apresentou um resultado consistentemente e significativamente superior ao da Média Móvel em todos os cenários testados.
Os dados numéricos (MAE e RMSE) provam que a estimativa de posição gerada pelo Filtro de Kalman é muito mais próxima da trajetória real do que a estimativa da Média Móvel.
Isso se deve à natureza de cada filtro:

- A Média Móvel é um filtro simples que apenas suaviza o sinal calculando a média de pontos passados. Isso a torna suscetível a um atraso (lag), especialmente em curvas e mudanças rápidas de direção, o que explica seus erros mais altos. Nos gráficos, isso é visível pela linha azul (Média Móvel) que parece "atrasada" em relação à trajetória real.

- O Filtro de Kalman, por outro lado, é um estimador mais inteligente. Ele não apenas olha para o passado, mas utiliza um modelo de movimento para prever a próxima posição do carro. Em seguida, ele corrige essa previsão com a nova medição da câmera. Esse ciclo de "previsão e correção" permite que ele siga a trajetória real com muito mais precisão e com um atraso mínimo, resultando em erros drasticamente menores.

Para a equipe TITANS, a escolha é clara. Implementar o Filtro de Kalman no sistema de controle dos carros-robôs fornecerá uma percepção de mundo muito mais estável e precisa, permitindo que os algoritmos de tomada de decisão (como mirar para um chute ou desviar de um oponente) operem com máxima eficiência.