RELATÓRIO PARCIAL (II) (PIBITI)

1. Identificação

Título do Projeto: Desenvolvimento de software para filtrar informações recebidas por UDP de câmeras num jogo da categoria Small Size League

Título do Subprojeto: Não se aplica.

Bolsista: Lucio Enzo Horie (lucio.horie@ime.eb.br)

Orientador: Gabriela Moutinho de Souza Dias, DSc (gabriela@ime.eb.br)

Data de início do bolsista no projeto: 1 de setembro de 2023

2. Introdução

O projeto busca enfrentar desafios comuns em competições de futebol de robôs, onde dados capturados por câmeras estão suscetíveis a ruídos e erros, como variações na qualidade da imagem, taxas de atualização da câmera e perdas de pacotes durante a transmissão de dados. Nesse contexto competitivo, a RoboIME busca aprimorar a precisão de seus dados para otimizar o desempenho de sua estratégia durante os jogos.

Além de filtrar dados para melhorar a precisão na localização de objetos, o projeto visa implementar estimadores para a velocidade dos robôs e da bola. Essa abordagem busca proporcionar uma compreensão mais completa do ambiente de jogo, contribuindo para decisões mais precisas e estratégias mais eficazes durante as competições.

3. Objetivos

O projeto tem como objetivos propostos listados no "Plano de Trabalho de PIBITI - Ciclo 2023/2024" e serão listados novamente a seguir.

• Desenvolver um software que estabeleça a comunicação via protocolo UDP (User Datagram Protocol) com o programa SSL-Vision [1], garantindo a recepção eficiente e confiável das informações transmitidas;

• Implementar algoritmos de filtragem, como filtro de Kalman, para reduzir ruídos e inconsistências nos dados recebidos, aprimorando a precisão e qualidade das estimativas de localização;

• Criar uma interface de usuário intuitiva e amigável que permita aos usuários configurar os filtros de acordo com suas preferências, adaptando o sistema de filtragem às necessidades específicas da Small Size League (SSL);

• Visualizar, de forma clara e compreensível, os resultados filtrados, apresentando informações como a trajetória dos objetos, velocidades estimadas e eventos relevantes, auxiliando as equipes participantes na análise e tomada de decisões durante os jogos.

• Realizar testes e avaliações rigorosas dos filtros desenvolvidos em cenários reais de jogo da SSL, comparando os resultados filtrados com as informações originais fornecidas pelo SSL-Vision [1], com o objetivo de verificar a melhoria na estimativa de localização e a contribuição para o desempenho dos robôs.

4. Cronograma

O cronograma atualizado, com as atividades planejadas visando a conclusão do projeto, é apresentado a seguir. O atraso na realização de testes com dados reais se deve ao fato de que, além da equipe de robótica do Instituto não ter participado de competições recentemente, o laboratório está em reforma, por isso esteve interditado por quase todo o período. Este documento condensa as fases 7 a 10, destacando as próximas etapas do projeto a serem abordadas daqui para frente.

• Fase 7: Realização de testes a partir de dados reais coletados em ensaios em laboratório e análise de desempenho da ferramenta;

• Fase 8: Revisão da segunda versão da ferramenta por meio de ajustes no código desenvolvido em C++;

• Fase 9: Validação final da ferramenta e análise de desempenho;

• Fase 10: Documentação final do projeto, incluindo a elaboração do Relatório Final e do vídeo para o EIC.

	2024			
Fase	Maio	Jun	Jul	Ago
7	x			
8	x	x		
9		x	x	x
10				x

5. Atividades realizadas no período

Nesta seção serão descritas as atividades realizadas no período, desde a entrega do primeiro relatório. A notação a seguir seŕa utilizada no presente trabalho.

$\mathbf{x}_k$	vetor de estados do sistema no tempo k
$\hat{\mathbf{x}}_{k\|i}$	estimativa de x no tempo k baseado no tempo i, $k \geq i$
$\tilde{\mathbf{x}}_{k\|k}$	erro da estimativa, $\hat{\mathbf{x}}_{k\|k} - \mathbf{x}_k$
$\mathbf{y}_k$	vetor das observações no tempo k
$\mathbf{P}_k$	matriz de covariância
$\mathbf{\Phi}_k$	matriz de transição de estado
$\mathbf{\Gamma}_k$	matriz de entradas do modelo
$\mathbf{H}_k$	matriz de transição de saída
$\mathbf{w}_k$	vetor ruído do processo
$\mathbf{v}_k$	vetor ruído da medição
$\mathbf{Q}_k$	matriz de covariância do ruído do processo
$\mathbf{R}_k$	matriz de covariância da medição do processo
$\mathbf{K}_k$	Ganho de Kalman
ν_k	resíduo da medição
$\mathbf{S}_k$	resíduo da covariância

5.1. Estudo de técnicas de inteligência artificial e avaliação de sua aplicabilidade no problema a ser resolvido

Foram examinadas diversas técnicas utilizadas por equipes participantes da categoria SSL para filtrar e estimar os dados recebidos pelas câmeras durante a competição. A pesquisa visou avaliar a viabilidade e eficácia de técnicas de inteligência artificial de forma mais abrangente, com o propósito de, futuramente, potencializar a precisão e a qualidade das estimativas de localização em ambientes dinâmicos e sujeitos a ruídos.

Equipe	Escola	Técnica utilizada
KIKS	*National Institute of Technology, Toyota College*	Média aritmética móvel e variância [2]
CMDragons	*Carnegie Mellon University*	Filtro de Kalman estendido [3]
ER-Force	*Friedrich-Alexander-Universität Erlangen-Nürnberg*	Filtro heurístico [4]
UBC Thunderbots	*The University of British Columbia*	Filtro de partículas [5]
RobôCIn	Universidade Federal de Pernambuco	Filtro de Kalman ou de Markov [6]
RoboTeam Twente	*University of Twente*	Filtro de Kalman [7]

Após as análises, é notável que o filtro de Kalman é utilizado por diversas equipes de robótica ou, pelo menos, muitas delas já a implementaram alguma vez em seus códigos. Logo, alguns problemas com relação ao Kalman foram percebidos, além de melhorias nos seus algoritmos foram feitas para melhor se adaptar à competição da *RoboCup*.

Entre todas as equipes analisadas, a que apresentou melhor descrições dos seus feitos foi a alemã "ER-Force". Por isso, será feito um breve resumo da técnica implementada pela equipe juntamente de algumas observações.

Primeiro, serão pontuados os principais objetivos, os quais se esperam ao acompanhar a trajetória da bola.
• Reduzir o ruído nos dados de posição.

• Estimar as velocidades para todos os objetos.

• Remover artefatos criados pelo SSL-Vision [1], como a posição da bola sendo projetada para uma posição diferente no chão enquanto está voando.

• Extrapolar o estado do mundo para o futuro.

Por segundo, será destacado os problemas do filtro de Kalman, os quais trazem a motivação de se aprimorar a técnica para o problema em específico.

• Um robô pode obstruir a visão da bola se ambos estiverem perto o suficiente. Em vários desses casos, o Kalman é incapaz de corrigir essas oclusões.

• Quando está se prevendo a posição da bola no futuro, o filtro não leva em conta a possibilidade da bola ser interceptada por um robô.

A exemplo disso, a *ER-Force* propõe o seguinte exemplo, considerando a Figura 1. A bola é lançada em um robô estacionário e depois é parada pelo mesmo. O círculo roxo indica a detecção da bola enviada pelo SSL-vision, enquanto o círculo laranja representa a bola futura prevista pelo nosso filtro de Kalman. Pode-se observar que a bola prevista está dentro do robô por algum tempo até que as detecções brutas da bola cheguem ao robô e a bola seja parada. Esse comportamento é indesejável, pois pode causar problemas no código de IA que utiliza o estado do mundo previsto.

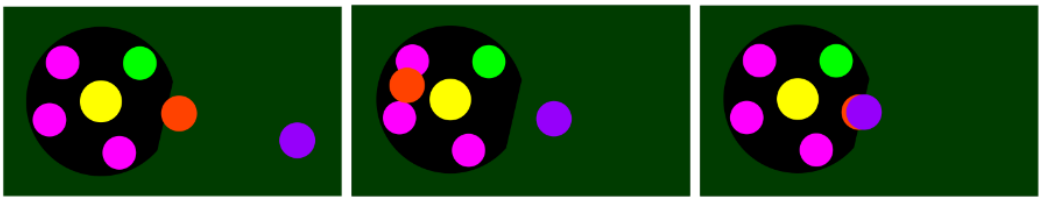

Figura 1 - Detecções originais da bola em roxo e bola rastreada em laranja. Ilustração retirada do trabalho da ER-Force [4].

Posteriormente, a equipe alemã afirma que, apesar desses problemas, ela não descartou o filtro de Kalman, uma vez que é um conceito já provado e largamente utilizado. Assim, para resolver o problema, os competidores resolveram por separar em vários casos, tentando cobrir os mais gerais, além de se preocuparem com alguns casos extremos.

Alguns dos modos criados para lidar com esses casos foram:

• Modo "driblando bola": Usado quando o robô está arrastando a bola com ele, e por isso, muitas vezes cobrindo a bola do ângulo de visão da câmera;

• Modo "colisão": Usado quando a bola prevista está intersectando um robô;

• Modo "bola voando": Um algoritmo a parte foi criado apenas para checar se a bola está fora do chão e, assim, projetar a posição da bola futura com maior precisão.

Portanto, pode-se observar que mesmo o filtro de Kalman apresentando alguns problemas, ele é confiável nos casos mais gerais. Logo, outros estudos analisando e solucionando os casos problemáticos podem ser desenvolvidos posteriormente, porém fogem do escopo deste projeto.

Em conclusão a esse estudo, não foram encontradas equipes, as quais mostraram publicamente o uso de aprendizagem de máquina, em inglês "machine learning", ou algo similar para filtrar ou estimar dados. Porém, o uso dessas tecnologias poderia melhorar a modelagem do problema. Já que, a modelagem real do movimento tanto dos robôs quanto da bola não são lineares, como previsto pelo filtro de Kalman.

5.2. Implementação da segunda versão da ferramenta por meio de ajustes nos algoritmos

Novos ajustes nos algoritmos foram feitos durante a implementação do código em C++, como adição de alguns passos para a facilitação de testes ou para deixar mais claro o código. Sendo assim, uma breve recapitulação da modelagem abordando as melhorias feitas no código serão abordadas neste tópico.

5.2.1. Modelo

Assim sendo, se baseando no livro do professor Franklin [8], o modelo do observador para o sistema será descrito como mostrado pelas equações a seguir e pode ser representado pelo diagrama de blocos da Figura 2.

$$\mathbf{x}_{k+1} = \mathbf{\Phi}_k \mathbf{x}_k + \mathbf{\Gamma}_k \mathbf{u}_k + \mathbf{w}_k$$

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$$

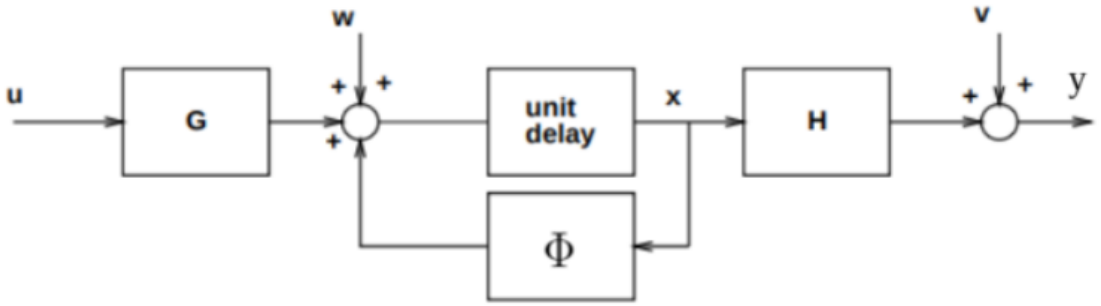

Figura 2 - Diagrama de blocos para o modelo do sistema.

5.2.2. Considerações

Na revisão do projeto, foram assumidas algumas considerações, as quais serão listadas abaixo.

• O ruído aleatório do processo e das medições $\mathbf{w}_k$ e $\mathbf{v}_k$ são não correlacionadas, são ruídos brancos de média zero e com matrizes de covariância conhecidas. Portanto, sendo E[x] a esperança de x:

$$E[\mathbf{w}_k \mathbf{w}_l^T] = \begin{cases} \mathbf{Q}_k & k = l, \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

$$E[\mathbf{v}_k\mathbf{v}_l^T] = \begin{cases} \mathbf{R}_k & k = l, \\ \mathbf{0} & \text{caso contrário} \end{cases}$$

$$E[\mathbf{w}_k\mathbf{v}_l^T] = \mathbf{0} \quad \forall k, l$$

Onde $\mathbf{Q}_k$ e $\mathbf{R}_k$ são matrizes simétricas semi-definidas positivas.

5.2.3. Algoritmo

Novos parâmetros para o filtro foram adicionados a fim de facilitar na sua representação. O novo passo a passo pode ser descrito pelo Algoritmo 1, na qual "k+1" é sempre a iteração atual e "k" a iteração anterior. Note que o código irá iterar enquanto ele estiver funcionando.

Algoritmo 1	Filtro de Kalman para estimativa de posição, velocidade e da aceleração de objetos
Requerimentos	y_k, tempo da coleta do dado, valores iniciais $\hat{\mathbf{x}}_{0\|0}$ e $\mathbf{P}_{0\|0}$, matrizes do modelo $\mathbf{\Phi}_k$, $\mathbf{H}_k$, $\mathbf{Q}_k$, $\mathbf{R}_k$ e podemos fazer $\mathbf{\Gamma}_k = \mathbf{0}$.

1: Inicialize o vetor de estados e a matriz de covariância com os valores iniciais

2: **enquanto** um novo pacote de dados da câmera não chegar **faça**

3: $\hat{\mathbf{x}}_{k+1\|k} = \Phi_k\hat{\mathbf{x}}_{k\|k} + \Gamma_k\mathbf{u}_k$ //Projete o estado seguinte

4: $\mathbf{P}_{k+1\|k} = \mathbf{\Phi}_k\hat{\mathbf{P}}_{k\|k}\Phi_k^T + \mathbf{Q}_k$ //Projete a covariância seguinte

5: **se** um novo pacote de dados chegar, **faça**:

6: $\nu_{k+1} = \mathbf{y}_{k+1} - \mathbf{H}_{k+1}\hat{\mathbf{x}}_{k+1\|k}$ //Compute o resíduo da medição

7: $\mathbf{S_{k+1}} = \mathbf{H}_k\mathbf{P}_{k+1\|k}\mathbf{H}_{k+1}^T + \mathbf{R}_{k+1}$ // Compute o resíduo da covariância

8: $\mathbf{K}_{k+1} = \mathbf{P}_{k+1\|k}\mathbf{H}_{k+1}^T\mathbf{S}^{-1}$ //Compute o ganho de Kalman

9: $\hat{\mathbf{x}}_{k+1\|k+1} = \hat{\mathbf{x}}_{k+1\|k} + \mathbf{K}_{k+1}\nu_{k+1}$ //Atualize o estado estimado

10: $\mathbf{P}_{k+1\|k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1})\mathbf{P}_{k+1\|k}$ //Atualize a matriz de covariância

11: **se** a comunicação foi interrompida, **termine**

12: caso contrário, **volte** para 2

Os passos no algoritmo consistem na seguinte ideia:

a) (Passos 3 e 4) Atualização com o tempo (“predição”): Prediz o estado e a variância no tempo k+1 dependendo da informação do tempo k.

b) (Passos 6 ao 10) Atualização das medidas (“correções”): Atualiza o estado e a variância usando a combinação da predição do estado e da observação $\mathbf{y}_{k+1}$.

Com essas atualizações no algoritmo, apesar de estarmos construindo um filtro de Kalman, é possível utilizar o mesmo código para conseguir estimar outros parâmetros que não são o vetor de estados. Segundo [9], o código é usado para estimar:

- $\hat{\mathbf{z}}_{k+1|k}$ é chamado de filtro de medição;
- $\hat{\mathbf{x}}_{k+1|k}$ é chamado de filtro de predição;
- $\hat{\mathbf{z}}_{k+1|k}$ é chamado de *"whitening filter"*;
- $\hat{\mathbf{z}}_{k+1|k}$ é chamado de filtro de Kalman.

5.2.4. Revisão do modelo discreto para um objeto se movendo com aceleração constante

Para modelar os objetos, a mesma ideia do relatório anterior foi utilizada. Foi considerado a bola como sendo uma partícula de aceleração constante, ou seja, um ponto que obedece às equações do movimento retilíneo uniformemente variado.

$$\mathbf{x}_{ball}(k) = \begin{bmatrix} x(k) & y(k) & z(k) & v_x(k) & v_y(k) & v_z(k) & a_x(k) & a_y(k) & a_z(k) \end{bmatrix}^T$$

$$\mathbf{y}_{ball}(k) = \begin{bmatrix} x(k) \\ y(k) \\ z(k) \end{bmatrix}$$

$$\mathbf{\Phi}_{k,ball} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{\Delta t^2}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \frac{\Delta t^2}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H}_{ball} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{Q}_{k,\text{ball}} = q \begin{bmatrix} \frac{\Delta t^5}{20} & 0 & 0 & \frac{\Delta t^4}{8} & 0 & 0 & \frac{\Delta t^3}{6} & 0 & 0 \\ 0 & \frac{\Delta t^5}{20} & 0 & 0 & \frac{\Delta t^4}{8} & 0 & 0 & \frac{\Delta t^3}{6} & 0 \\ 0 & 0 & \frac{\Delta t^5}{20} & 0 & 0 & \frac{\Delta t^4}{8} & 0 & 0 & \frac{\Delta t^3}{6} \\ \frac{\Delta t^4}{8} & 0 & 0 & \frac{\Delta t^3}{3} & 0 & 0 & \frac{\Delta t^2}{2} & 0 & 0 \\ 0 & \frac{\Delta t^4}{8} & 0 & 0 & \frac{\Delta t^3}{3} & 0 & 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & 0 & \frac{\Delta t^4}{8} & 0 & 0 & \frac{\Delta t^3}{3} & 0 & 0 & \frac{\Delta t^2}{2} \\ \frac{\Delta t^3}{6} & 0 & 0 & \frac{\Delta t^2}{2} & 0 & 0 & \Delta t & 0 & 0 \\ 0 & \frac{\Delta t^3}{6} & 0 & 0 & \frac{\Delta t^2}{2} & 0 & 0 & \Delta t & 0 \\ 0 & 0 & \frac{\Delta t^3}{6} & 0 & 0 & \frac{\Delta t^2}{2} & 0 & 0 & \Delta t \end{bmatrix}$$

Sendo q uma constante, esse resultado foi adaptado de [9].

5.2.5. Melhorias na implementação da comunicação com o SSL-Vision

Anteriormente, apenas havia sido implementado o recebimento dos pacotes do programa SSL-Vision [1]. Agora, também foi implementada uma nova funcionalidade para testes. É possível realizar testes usando o simulador "grSim" [10], uma vez que foi feita a comunicação TCP com tal simulador. Isso permite o envio de comandos únicos para a bola utilizando as teclas "W", "A", "S" e "D" para controlá-la.

Essa nova ferramenta pode ajudar em testes quando não houver disponibilidade de utilização do campo e da câmera para a coleta de dados, que é a situação atual. Pois, o laboratório encontra-se em reforma.

5.2.6. Escolha do padrão de projeto do código principal

O projeto do filtro de Kalman é o pontapé inicial para a migração do código da equipe de robótica do IME, RoboIME, para a linguagem C++. Sendo assim, novas ferramentas e funcionalidades serão adicionadas com o tempo, formando uma nova inteligência artificial (IA) para os robôs jogadores de futebol. Sendo assim, o design do Observador foi escolhido.

O padrão permite uma estrutura modular e escalável para a implementação de novos componentes. Além disso, é garantido que os componentes sejam atualizados de forma eficiente e reativa sempre que novos dados são recebidos, mantendo a flexibilidade e a clareza do código.

A estrutura pode ser observada na Figura 3, sendo que todo esse padrão é explicado de forma clara por [13]. E, para mostrar sua aplicabilidade, um exemplo presente no código pode ser visto na Figura 4.

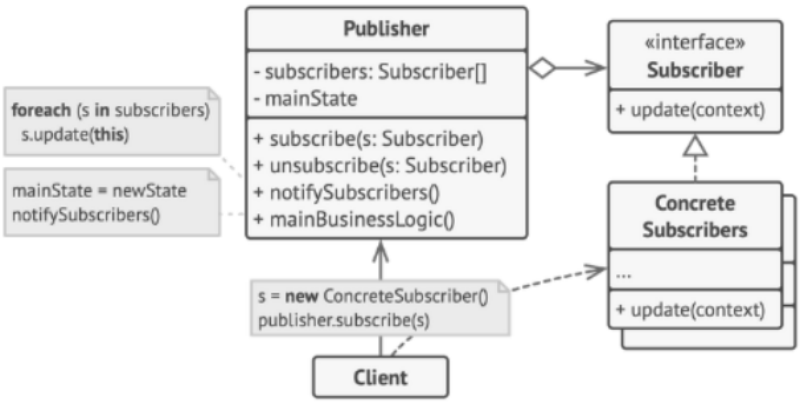

Figura 3 - Estrutura do padrão Observer. Imagem ilustrativa de [13].

```
// Subscriber
class Component{
public:
    virtual void update(Data);
};
// Publisher
class Vision{
private:
    Data data;
    vector<Component*> components;
public:
    virtual void receiveData() = 0;
    void filterData() {/*...*/};
    Data getData(){return data;}
    void subscribe(Component){/*...*/}
    void unsubscribe(Component){/*...*/}
    void noitifyComponents() {
        for(auto& component : components){
            component->update(data);
        }
    };
```

Figura 4 - Exemplo da implementação no código.

6. Resultados alcançados no período

Durante este segundo período de pesquisa e desenvolvimento, foi possível implementar o filtro de Kalman em C++, com melhorias na sua modelagem. Além disso, a estruturação do código dentro do projeto da equipe

foi realizada com sucesso. No entanto, devido a impedimentos relacionados ao uso do laboratório, não foi possível realizar testes com dados da câmera.

A implementação do filtro de Kalman dentro do projeto pode ser encontrada no repositório do GitHub [11]. O código foi desenvolvido com a ajuda da biblioteca Eigen [12]. Inicialmente, foi criada uma classe para um filtro de Kalman genérico utilizando templates, o que permitiu a definição flexível das matrizes do modelo e suas dimensões. Essa abordagem facilita o uso da mesma classe para filtrar diferentes tipos de dados.

Posteriormente, com base no modelo descrito anteriormente neste trabalho, foi criada uma classe específica para o "filtro de Kalman 3D". Esse nome é devido ao filtro estar modelado para os três eixos coordenados.

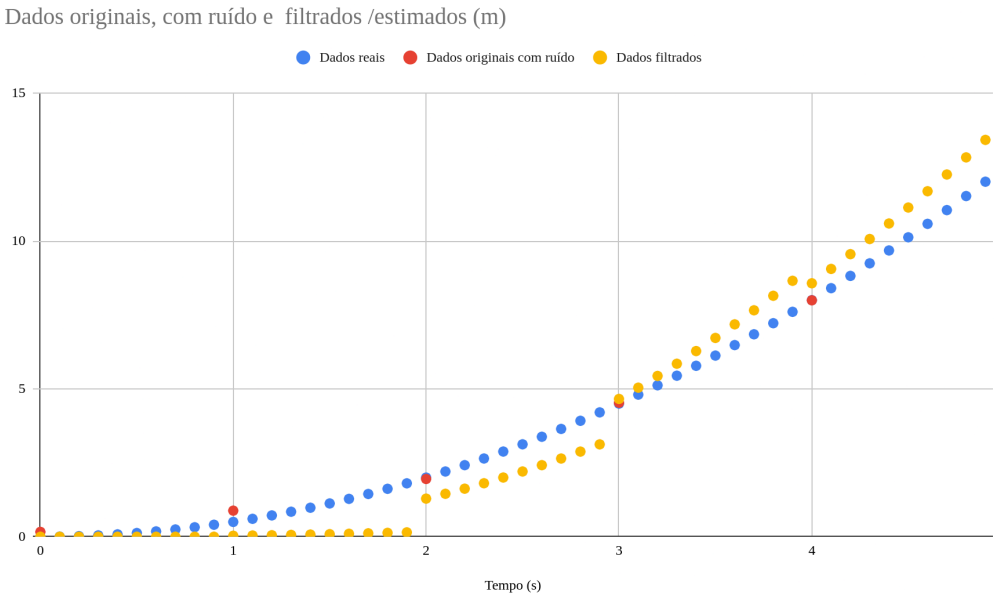

Figura 5 - Filtro estimando a posição de um objeto com aceleração constante com valores iniciais nulos.

Em uma última adição ao trabalho, foi realizada a validação da classe do filtro de Kalman desenvolvida, utilizando um teste simples implementado diretamente no código em C++. Foram gerados dados de um objeto em movimento com aceleração constante de 1 m/s² ao longo de 4 segundos. Durante esse período, foram enviadas informações com ruído, com desvio padrão de 0,1 metro, nos instantes de 0, 1, 2, 3 e 4 segundos. O filtro de Kalman foi então utilizado para estimar a posição do objeto nos intervalos de tempo entre as medições.

Os resultados obtidos pelo estimador são apresentados nas Figuras 5 e 6. A análise desses resultados demonstra que, mesmo com um deslocamento inicial na posição maior que o desvio padrão do ruído, o filtro de Kalman conseguiu prever adequadamente a posição do objeto já a partir do terceiro dado recebido.

É importante destacar que, conforme estudado anteriormente, em casos de mudanças abruptas no movimento do objeto – como uma parada súbita devido a uma colisão – o filtro de Kalman demoraria até o próximo recebimento de um pacote de dados para reagir a tal evento. Isso ocorre mesmo que o código seja capaz de realizar várias iterações nesse intervalo de tempo.

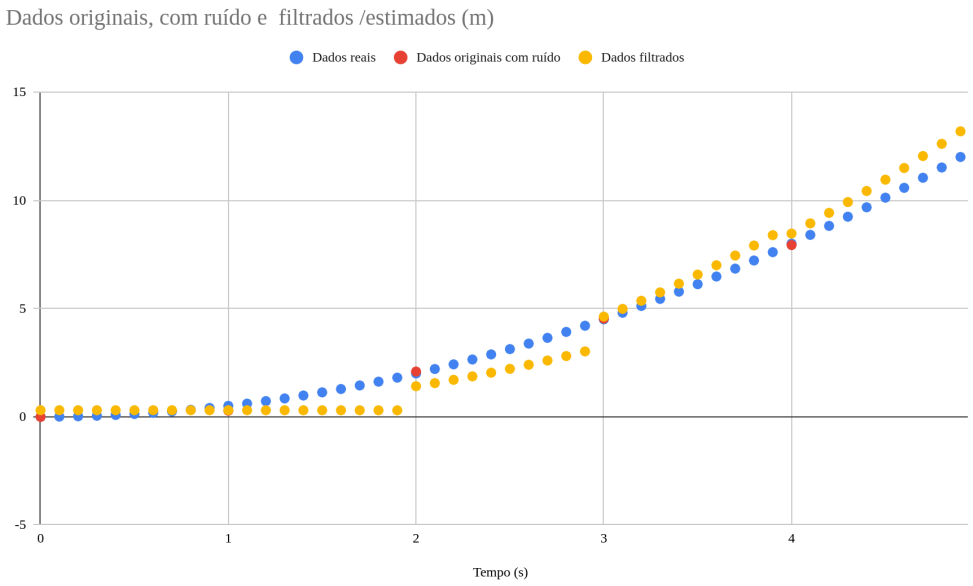

Figura 6 - Filtro estimando a posição de um objeto com aceleração constante com valores iniciais nulos exceto a posição inicial, colocada propositalmente igual a 0,3 m.

7. Próximos passos

No próximo período de trabalho, o foco será na validação final da ferramenta e na análise de seu desempenho. Para isso, a captura de dados reais utilizando uma câmera será prioritária no projeto. Seguiremos as fases 7 e 8, conforme descritas no cronograma, para validar a ferramenta desenvolvida neste projeto. Após essa etapa, o projeto será concluído com a redação do relatório final e a gravação do vídeo para o EIC.

8. Dificuldades encontradas

Durante este período, diversas dificuldades surgiram na implementação do código. Primeiramente, a integração da biblioteca externa não foi trivial, pois frequentemente ocorreram problemas durante a compilação do código. Além disso, surgiram vários problemas ao usar *templates* na classe do filtro de Kalman. Quando a classe foi separada em arquivos ".cpp" e ".h", o compilador não conseguia relacionar corretamente os arquivos, resultando em erros.

Para resolver essa questão, a classe inteira foi declarada e definida no arquivo ".h" do filtro. Outra solução possível seria definir todos os templates utilizados diretamente no ".h". Entretanto, essa abordagem pode ser desvantajosa, pois exigiria que todos os templates a serem usados fossem conhecidos antecipadamente.

9. Referências bibliográficas

[1] ROBOCUP-SSL. RoboCup Small Size League Shared Vision System. 2023. 20 nov. de 2023. Disponível em: <https://github.com/RoboCup-SSL/ssl-vision>.

[2] OHNO, K.; MIMURA, T.; YOKOTA, H.; OHMURA, T.; SANO, T.; WATANABE, M.; SUGIURA, T. KIKS 2017 Team Description — National Institute of Technology, Toyota College, Aichi, Japan, 2017.

[3] BISWAS, J.; COOKSEY, P.; KLEE, S.; MENDOZA, J. P.; WANG, R.; ZHU, D.; VELOSO, M. CMDragons 2015 Extended Team Description — Carnegie Mellon University, 2015.

[4] BERGMANN, P.; ENGELHARDT, T.; GAREIS, E.; HAHN, U.; HEINEKEN, T.; HOPF, V.; MöSER, R.; MUTTER, F.; SCHMID, M.; SCHMIDT, M.; STADLER, M.; WENDLER, A.; WIEDMANN, M. ER-Force 2023 Extended Team Description Paper — Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Alemanha, 2023.

[5] MACDOUGALL, M.; ELLIS, G.; HASHEMI, A.; JACKSON, E.; LIP, O. C.; IVANOV, N.; PETRIE, J.; TONKS-TURCOTTE, K.; SOUSA, C.; LAI, K.; XU, B.; LI, Q.; GOTO, E.; DEUTSCH, D.; BUONASSISI, A.; LEE, M. 2018 Team Description Paper: UBC Thunderbots — The University of British Columbia, Vancouver, BC, Canadá, 2018.

[6] SILVA, C.; MARTINS, F.; MACHADO, J. G.; CAVALCANTI, L.; SOUSA, R.; FERNANDES, R.; ARAÚJO, V.; SILVA, V.; BARROS, E.; BASSANI, H. F.; NETO, P. S. G. DE M.; REN, T. I. RobôCIn 2019 Team Description Paper — Universidade Federal de Pernambuco, Recife - Pernambuco, Brasil, 2019.

[7] BOS, L.; CITGEZ, Y.; DORENBOS, H.; EICHLER, A.; HULST, R. van der; NAGELVO-ORT, L. K.; KUIL, T. van der; LUONG, J.; POORT, L.; PRINSENBERG, F.; REGT, C. de; UITERKAMP, L. S.; SCHRIJVER, R.; STEERNEMAN, E.; VACARIU, P.; WERVEN, J. van; WERFF, S. van der; ZWERVER, S. RoboTeam Twente Extended Team Description Paper 2020 — University of Twente (UT), Enschede, Países Baixos, 2019.

[8] FRANKLIN, G. F.; POWELL, J. D.; WORKMAN, M. L. Digital Control of Dynamic Systems. 3. ed. Estados Unidos da América: Ellis-Kagle Press, 1998.

[9] REID, I. Estimation II — Hilary term, 2001.

[10] RAHIMI, M. M.; SEGRE, J.; MONAJJEMI, V.; KOOCHAKZADEH, A.; MOHAIMENI-ANPOUR, S.; OMMER, N.; KIMURA, A. K.; FELTRACCO, J.; SATO, K.; AHSANI,A. GRSIM. [S.l.]: GitHub, 2021. <https://github.com /RoboCup-SSL/grSim/>. Repositório GitHub. Acesso em: 20 de maio de 2024.

[11] MELO, E. H. V.; HORIE, L. E. SSL_AI_CPP. [S.l.]: GitHub, 2024. <https://github.com/roboime/SSL_AI_PP/tree/feature/estimator/Phoenixpp-cma ke/src/components/implementations/filters/kalman_filter>. Repositório GitHub. Acesso em: 20 de maio de 2024.

[12] GUENNEBAUD, G.; JACOB, B. et al. Eigen v3. Disponível em: <http://eigen. tuxfamily.org>. Acesso em: 20 de maio de 2024.

[13] SHVETS, A. Observer. Disponível em: <https://refactoring. guru/design-patterns/ observer>. Acesso em: 20 de maio de 2024.