CENTRO UNIVERSITÁRIO FEI
GUILHERME LUIS PAULI

TRACKING DE OBJETOS EM CAMPO PARA SOLUCIONAR O PROBLEMA DE LATÊNCIA DAS INFORMAÇÕES NO FUTEBOL DE ROBÔS

São Bernardo do Campo
2018

GUILHERME LUIS PAULI

TRACKING DE OBJETOS EM CAMPO PARA SOLUCIONAR O PROBLEMA DE LATÊNCIA DAS INFORMAÇÕES NO FUTEBOL DE ROBÔS

Relatório Final de Iniciação Científica apresentado ao Centro Universitário FEI, como parte dos requisitos para obtenção de bolsa do Programa PIBIC-FEI. Orientado pelo Prof. Dr. Flavio Tonidandel

São Bernardo do Campo

2018

RESUMO

A categoria de futebol de robôs *Small Size League* é uma das mais dinâmicas presentes na RoboCup. No ambiente de jogo são utilizadas mais de uma câmera para a captação de imagens do campo, já que não é utilizado nenhum outro tipo de sensor, este seria o único meio de obter as informações que acontecem durante o jogo. Um dos problemas que ocorrem é que a velocidade de processamento dos dados é muito superior à captura das imagens feita pelas câmeras, isso resulta em um *gap* (i.e. um intervalo) em que a *estratégia* deixa de ser atualizada com novas informações provenientes do sistema de visão.

Além dessa latência para o recebimento de novas informações, existe um problema secundário gerado pelo uso de mais de uma câmera, o *overlapping* (i.e. a região de sobreposição de imagens), que acaba por gerar outros agravantes e instabilidades que prejudicam o desempenho da equipe RoboFEI nas jogadas de bola parada, em que ela está sobre a região de *overlap*, ou então, no recebimento de um passe. Esse problema está relacionado ao sistema de visão e é encontrado na categoria de futebol de robôs *Small Size League* por utilizar mais de uma câmera para captar o ambiente de jogo, diferente de outras categorias que utilizam apenas uma câmera como a Very Small Size League (IEEE Very Small Size Soccer, 2018) ou então que tenha o sistema de visão embarcado, como a Middle Size League (Middle Size League, 2018) e a Humanoid League (Humanoid League, 2018).

A proposta desta pesquisa é aplicar um filtro capaz de preencher a lacuna de informações a respeito da posição dos objetos em campo (i.e. da bola e dos robôs) e diminuir a variação quando estes passam pela região de *overlap* das câmeras. O modelo de filtro a ser estudado e implementado é o filtro de Kalman, que além de servir como um excelente filtro de ruídos também é considerado como um bom estimador de estados.

Por fim, serão comparadas as informações recebidas do filtro e do SSL-Vision, tendo foco principal no tempo em que uma nova informação é fornecida e a posição do objeto em campo recebida, para, então, determinar a eficiência da utilizaçãodo filtro para o problema apresentado.

Palavras-chave: Kalman. Small Size League. Tracking. Overlap.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Criada em 1997, a RoboCup tem como principal objetivo reunir um time de futebol de robôs que seja capaz de jogar e vencer um time de futebol de humanos (campeão da Copa do Mundo) até 2050. Assim, o objetivo é promover a evolução da parte mecânica, eletrônica e a inteligência artificial dos robôs para um patamar que hoje ainda é considerado distante, o que se tornaria um grande marco para a história da robótica (ROBOCUP, 1997).

A competição da RoboCup acontece anualmente, onde times de diversos países se encontram para disputar diferentes ligas como campeonatos de futebol, missões de resgate, gerenciamento de tarefas domésticas, entre outras. As categorias nas quais a FEI participa, através da equipe RoboFEI, são a de futebol de robôs Small Size League, Humanoid League e a categoria para serviços e assistência doméstica @Home (RoboCup @Home league, 2018).

A categoria SSL – Small Size League (RoboCup Small Size League, 2018) é uma das categorias mais dinâmicas presentes na competição devido à velocidade com que o jogo acontece, com chutes permitidos de até no máximo 6.5 m/s e robôs que se movem a mais de 2.0 m/s. Toda a dinâmica do jogo é capturada por câmeras e das imagens obtidas, após passarem pelo sistema de visão (Seção 2.2), extraem-se as informações sobre o que está acontecendo no ambiente do jogo. A captura das imagens tem uma velocidade limitada pela capacidade da câmera, e que acaba por ser inferior ao de processamento das informações que já chegaram para a *estratégia* (i.e., o software da equipe) existindo um intervalo de tempo em que a *estratégia* processa o mesmo dado e envia a mesma resposta para os robôs. Por ser um jogo dinâmico em que cada milésimo de segundo conta e que as mudanças no ambiente ocorrem nessa fração de tempo, essa falta de novas informações implica em alguns instantes em que nada aconteça ou que continue executando a mesma ação mesmo o ambiente já estar transitando de um estado para outro, o que gera um problema muito significativo de atraso para que a informação seja atualizada, processada, colocada no pacote de dados e enviada para o robô, para então ele vencer a inercia e se deslocar para um novo destino. Todo esse tempo que foi gasto pode influenciar negativamente no desempenho da equipe.

Dentro do campo existem algumas regiões onde ocorre a intersecção de imagens de duas ou mais câmeras distintas. Essa intersecção de imagens é chamada de *overlap*. O

problema que ela causa acontece na hora da captura e interpretação da imagem pelo software do sistema de visão. Quando um objeto é capturado na região de *overlap*, após o processamento, o dado que fica atrelado à sua posição acaba ficando incerto, pois ambas imagens retornam posições (x, y) diferentes mesmo que objeto não tenha se movido, e esse distúrbio é um tipo de ruído oriundo da captação das imagens por câmeras distintas

Para que esse ruído não se torne um problema na hora da tomada de decisão do software é necessária a utilização de um tipo de filtro que consiga lidar com isso. Nesta pesquisa será testado e verificado a implementação de um deles, conhecido como filtro de Kalman (WELCH e BISHOP, 2006), filtro utilizado pela maioria das equipes que competem na categoria da RoboCup, Small Size League.

1.2 OBJETIVO

Esta pesquisa tem como objetivo a resolução do período de *gap* em que a *estratégia* fica sem ser atualizada com novas informações e, além disso, corrigir a variação da posição dos objetos em campo quando estes transitam pelas regiões de *overlap,* visando uma maior acurácia nos cálculos feitos pelo software sem que isso seja um empecilho que prejudique o desempenho da equipe durante as partidas do futebol de robôs.

1.3 JUSTIFICATIVA

Em 2017 a equipe da RoboFEI (ROBOFEI, 2018) teve seu software da *estratégia* totalmente refeito, tendo em vista os problemas apresentados no software antigo já que este se tornou inadequado por conta dos problemas que vinha apresentando e sua complexidade pela falta de documentação para entender partes do código que continham *bugs*. Desde então a *estratégia* vem obtendo grandes avanços e a maioria das implementações básicas já foram realizadas. Uma dessas implementações que ainda estava em falta para o código era um filtro eficiente para amenizar ou zerar o ruído das informações que são retornadas pelo sistema de visão, principalmente quando algum objeto transita na região de *overlap*, e suprir a falta de informações para o software enquanto não estão disponíveis novas informações.

No laboratório da equipe existem duas câmeras que gravam o campo e, portanto, existe uma região de *overlap* em que os testes com os filtros serão realizados para

obtenção de dados reais (i.e. com a resposta do modelo real) e os métodos de predição serão testados tanto no simulador como no campo. Nos campeonatos são utilizadas mais de duas câmeras por campo.

Esse aprimoramento é importante, porque ao receber uma informação ruidosa, contaminada de erro e com um alto grau de incerteza, tudo aquilo que utiliza dessas informações só acaba por contribuir para que esse erro se propague. Não somente isso, mas também interfere em muitas das jogadas essenciais que acontecem durante um jogo do futebol de robôs, como a realização/interceptação de passes ou simplesmente o fato do robô chegar até a bola, como no *indirect* (jogada semelhante a cobrança de falta indireta) ou ainda o *kickoff* (jogada que precede o início de jogo). Para que estas coisas possam ocorrer é necessário que não exista variações das posições dos objetos e que elas consigam representar com precisão o que acontece no campo real. Outra informação acerca da posição da bola é a estimativa de onde ela se encontrará no futuro.

Atualmente existem, para o problema de *overlap* e para a predição da posição bola, métodos/funções já implementadas no software da RoboFEI, todavia o filtro atualmente utilizado não é o ideal e precisa ser melhorado ou substituído, já o método para a estimativa da posição da bola apresentou, nos testes feitos no campo da equipe e no simulador *grSim*, um baixo erro de precisão, em que o teste feito apenas comparou a predição feita com as posições futuras, e reais, em que a bola se encontrou.

Com a realização desta pesquisa espera-se que o filtro de Kalman seja capaz de atender aos requisitos do problema, sendo a primeira e mais importante como um bom estimador e a segunda como um bom filtro de ruídos. O filtro de Kalman é um ótimo filtro a ser usado quando se trata de sistemas dinâmicos lineares, ou seja, sistemas que apresentam comportamento continuo ao longo do tempo, que é o caso da bola.

2 REVISÃO BIBLIOGRÁFICA

2.1 TRABALHOS RELACIONADOS

A escolha de se implementar o filtro de Kalman (KALMAN, 1960) começou por ele ser um filtro muito conhecido e utilizado por aqueles que trabalham com visão computacional, e muito utilizado pelas equipes da categoria SSL, como o RoboIME (ROBOIME, 2017), Warthog Robotics (WARTHOG ROBOTICS, 2016) e UaiSoccer (UAISOCCER, 2013). Ele é um algoritimo computacional eficiente e quanto melhor o modelamento empregado mais robusto o filtro se torna, fornecendo uma ótima estimativa ao longo do tempo (ZERVOS, 2012). A equipe de robos humanoide NUbot utiliza o filtro de Kalman coletando as informações da visão de todos os objetos vistos, juntamente com os dados provenientes de sensores adicionais, como a odometria, e posições passadas para conseguir obter uma estimativa de onde o robô, a bola e os outros robô se encontram. E isso é aplicado para cada um deles, ou seja, cada robô fornece ao filtro medidas e dados coletados da visão para que ele retorne a melhor estimativa possivel dos objetos, tudo em tempo real (NUBOTS TEAM, 2003).

Na grande maioria dos casos que empregam o uso do filtro de Kalman, sua principal função é o *tracking* de objetos. Comparado a outro filtro, como o filtro de particulas, o custo computacional para gerar as estimações é muito inferior, dado que a quantidade de iterações e o número de amostras que precisam ser fornecidas para o cálculo no filtro de particulas é bem maior. Já o filtro de Kalman não necessita de todo um histórico, ele consegue convergir para uma ótima estimativa apenas com um pequeno conjunto de amostras (MERVEN, NICOLLS e DE JAGER, 2003).

2.2 SISTEMA DE VISÃO

Atualmente o sistema de visão consiste em uma câmera que fica cerca de 4m acima do campo. As imagens que são tiradas, também chamadas de *frames*, são enviadas para um computador em que está presente o software de calibração desenvolvido pela própria comunidade SSL, o SSL-Vision (SSL-VISION DEVELOPER TEAM, 2009). O sistema de visão disponibiliza os dados para ambas as equipes que participam de um jogo, por esse motivo também é chamado de *shared vision*.

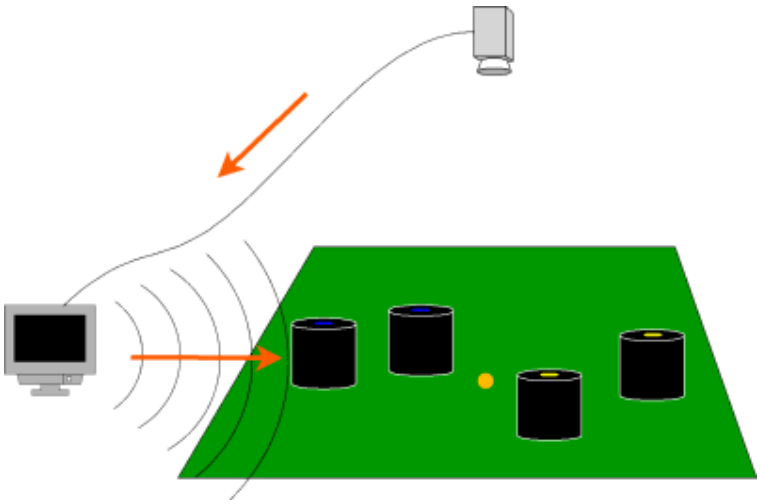
No software SSL-Vision, as imagens passam por algumas etapas de processamento para que seja possível extrair informações delas, como por exemplo, a calibração dos parâmetros do campo, detecção de padrões, entre outros. Ao identificar os *blobs*, que são as bolas coloridas localizadas acima do robô, é possível saber para qual time ele pertence (amarelo ou azul), qual a sua orientação e, também, a sua posição e a da bola naquele *frame*. Isso é realizado para cada câmera que está filmando o campo, quando o software processa as imagens das *n* câmeras tais informações são disponibilizadas para os computadores das equipes (S. ZICKLER, T. LAUE, O. BIRBACH, M. WONGPHATI, M. VELOSO, 2009).

Com estes dados em mãos, cada time consegue utilizá-los dentro da sua *estratégia*, ou seja, ambas equipes recebem do sistema de visão as mesmas informações, inclusive com os mesmos problemas referentes aos ruídos, por exemplo o *overlap*. Por isso cabe a cada equipe tratar problemas como esse, utilizando aquele método que lhe é cabível e que melhor atende a solucionar o problema.

Quando cada equipe termina o processamento dos dados, um pacote contendo as instruções que devem ser executadas pelo robô é enviada via rádio. Cada robô, por sua vez, interpreta aquele pacote, decodifica as informações e executa o que foi pedido, mudando o ambiente para os próximos *frames* capturados pelas câmeras, completando então um ciclo. A

Figura 1 ilustra o funcionamento destes ciclos nos jogos que acontecem nas competições de robótica da categoria SSL.

Figura 1 - Representação do Sistema de Visão



Fonte: Wiki RoboCup – Small Size League

Normalmente, nas competições oficiais da categoria SSL, os campos em que ocorrem os jogos ficam divididos em quatro regiões e cada uma delas é equipada com uma câmera responsável pela captura das imagens daquele quadrante, e tal visão aérea exibe uma projeção em duas dimensões do campo.

Cada objeto que está na visão das câmeras tem sua respectiva posição armazenadas no seu vetor de posição. Quando um objeto cruza a região de *overlap*, o que fica armazenado no seu vetor posição são dois valores diferentes, que são a posições provenientes de cada câmera, e enquanto tal objeto está naquela região, ou na visão, o vetor de pontos continua sendo atualizado com estas posições.

A *Figura 4* mostra como esse efeito é visto pelo software. A imagem foi retirada do *graphicalClient*, uma ferramenta auxiliar que existe junto com o software de calibração SSL-Vision. Pode-se notar uma repetição das imagens dos objetos, isso ocorre quando o mesmo objeto é visto pelas duas câmeras (*Figura 2 - Visão do campo pela câmera 0* e *Figura 3*), além disso, elas aparecem deslocadas indicando que cada câmera "enxerga" os objetos em posições diferentes. Essa é a incerteza da posição do objeto que transita por essa região, e esse é um problema que pode vir a atrapalhar cálculos como a trajetória da bola e sua velocidade, a interceptação da bola e até a previsão de onde ela vai parar.

2.2.1 TAXAS DE PROCESSAMENTO

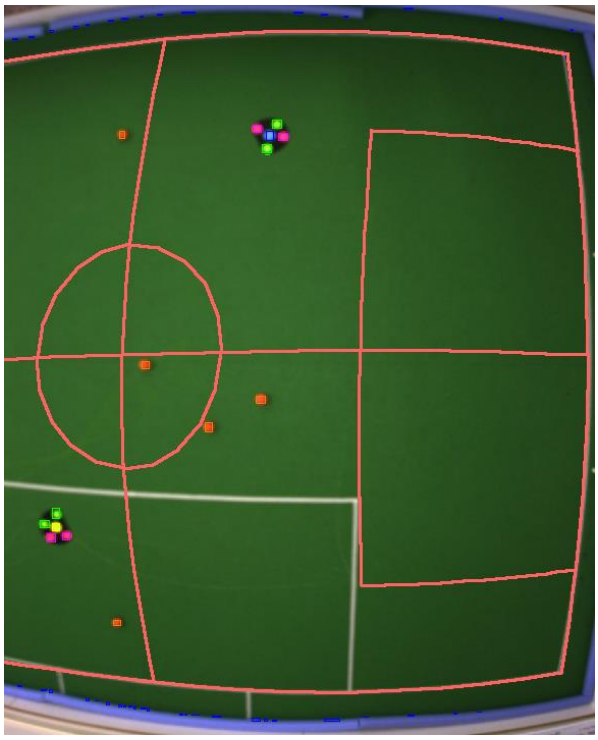
O software utilizado para a calibração e processamento das imagens, SSL-Vision, recebe novas imagens a uma certa taxa de *Frames Per Second* (FPS, i.e., quadros por segundo) que aquela câmera consegue fazer a captura. As câmeras utilizadas tanto no laboratório como nas competições tem a taxa próximo de 60 FPS, ou seja, as câmeras captura uma nova imagem, aproximadamente, a cada $16{,}67 * 10^{-3}\ segundos$.

Supondo que o software utilize as informações do *frame* que acaba de ser capturado e tratado pelo SSL-Vision, ele realiza as contas de uma função, ou método, em uma fração de tempo muito menor, restando um intervalo em que não existem novas informações para serem processadas/utilizadas. Isso faz com que a *estratégia* entenda que durante aquele período o que está acontecendo ainda faz parte do cenário do mesmo *frame*. Uma das funcionalidades do filtro de Kalman é que ele consegue estimar os estados mesmo que ainda não existam novas informações disponíveis, suprindo essa falta (ER-FORCE, 2012).

Figura 2 - Visão do campo pela câmera 0

Fonte: Autor

Figura 3 - Visão do campo pela câmera 1



Fonte: Autor

Figura 4 - Efeito do *overlap*

Fonte: Autor

2.3 PROBLEMA DE OVERLAP

Como já mencionado brevemente, boas equipes brasileiras utilizam o filtro de Kalman como solução para o *overlap*, assim como no cenário internacional a maioria dos times também utilizam, como por exemplo a equipe alemã ER-Force (ER-FORCE, 2008), a equipe iraniana Parsian (PARSIAN, 2010) e a equipe que detém o posto de número um de maiores ganhadores da RoboCup, a equipe norte americana CMDragons (CMDRAGONS, 2015). O sucesso das equipes não está somente atrelado ao Filtro de Kalman, mas ele garante que as informações que chegam até a *estratégia* desses times apresentem uma alta confiabilidade.

As equações matriciais utilizadas para o cálculo no filtro de Kalman são as mesmas, entretanto o que pode diferir para cada equipe são as matrizes de controle, de covariância e do modelo dinâmico dos sensores, posteriormente explicadas. Entretanto o que vale ressaltar é que o filtro de Kalman consegue convergir para um valor muito perto do valor real pois o dado é contaminado com ruído gaussiano branco, com média zero e desvio padrão baixo. A mudança nos parâmetros de algumas dessas matrizes vai atender ao requisito de cada equipe e do modelo levantado por ela. Dependendo da matriz que se altera o valor, é possível fazer com que o filtro de Kalman convirja para o valor real mais rápido ou mais devagar, ou então o grau de confiabilidade dos dados provenientes dos sensores, no caso, dos dados vindo do sistema de visão.

Por fim, como cada equipe desenvolve a sua estratégia em um ambiente de programação diferente e, para algumas delas, em outras linguagens de programação o que resulta em uma implementação que pode variar conforme a equipe. Por exemplo, muitas equipes utilizam as bibliotecas do OpenCV por ele atender às linguagens mais utilizadas (como é o caso desta pesquisa), e outras ainda utilizam as bibliotecas prontas do próprio ambiente de desenvolvimento, como no caso de quem utiliza o LabVIEW (NATIONAL INSTRUMENTS, 2018).

2.4 FILTRO SIMPLES

Atualmente, está implementado na *estratégia* da RoboFEI um filtro de média simples para o tratamento do ruído. Ele consiste em utilizar as posições provenientes da visão e calcular a média dessas variações. Antes de aplicar a média, as posições dos objetos ficam armazenadas em um novo *buffer* circular de pontos de um tamanho pré-

definido, este *buffer* tem as posições inseridas no seu início e fica como no representado abaixo:

$$vetorDasPosições(PosiçãoCamera1, PosiçãoCamera2, \dots)$$

Em seguida, as médias das posições em X e em Y são calculadas através das equações (1) e (2). E o resultado desta conta é retornado como ponto o estimado.

$$mediaPosiçãoEmX = \frac{1}{n}\sum_{i=1}^{n}(vetorDasPosições(i-1).x + vetorDasPosições(i).x) \quad (1)$$

$$mediaPosiçãoEmY = \frac{1}{n}\sum_{i=1}^{n}(vetorDasPosições(i-1).y + vetorDasPosições(i).y) \quad (2)$$

2.5 FILTRO DE KALMAN

O Filtro de Kalman (FK) foi elaborado por Rudolf Kalman na década de 1960. Ele é considerado como um eficiente filtro recursivo que estima o estado de um sistema dinâmico linear a partir de uma série de medições ruidosas. Isso significa que na maioria dos casos, mesmo que os dados recebidos contenham um alto valor de incerteza, ainda assim ele consegue convergir para uma ótima aproximação do valor real. Por esse motivo ele é amplamente utilizado em pesquisas e aplicações, particularmente na área de navegação autônoma ou assistida (WELCH e BISHOP, 2006).

O FK que será implementado apresenta duas fases distintas, a fase de Atualização e a fase de Predição. A fase de Predição utiliza a estimativa do estado no passo anterior para obter uma estimativa do estado no tempo atual, e é chamada de estimativa *a priori*. Na fase de Atualização, a observação atual é combinada com a *a priori* para refinar a estimativa do estado e esta é chamada de estimativa *a posteriori* (JUNIOR, 2015).

Antes de começar sua implementação é necessário entender sobre a dinâmica do ambiente em que o filtro será aplicado, pois é a partir desta análise que irá ser retirado o modelo comportamental do objeto estudado, no caso a bola, em forma de equações. Essas

equações e outros parâmetros serão a base para que o FK possa ser implementado, tendo seu comportamento avaliado e comparado com o do Filtro Simples (Seção 2.4).

2.6 FUNÇÕES PREDITIVAS

Aqui será apresentado como o FK pode utilizar dos seus cálculos para poder gerar uma estimativa futura da posição da bola e também como é o método desenvolvido e utilizado atualmente pela equipe RoboFEI.

2.6.1 ESTIMANDO A POSIÇÃO ATRAVÉS DO FK

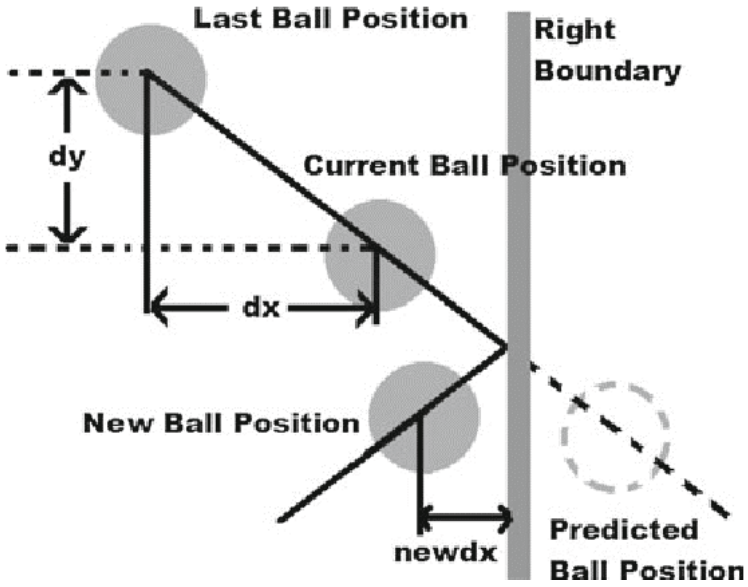
O FK também pode ser usado para estimar a posição da bola em campo. Como o modelo da bola é aproximadamente linear ao longo de sua trajetória, i.e. este modelo não varia, para conseguimos obter a posição estimada em n frames no futuro é necessário rodar a fase de predição na mesma proporção. Para estimar a posição da bola 30 frames no futuro deve-se processar a fase de predição 30 vezes (JUNIOR, 2015).

2.6.2 FUNÇÃO ESTIMATIVA ATUAL

O software atual conta com a presença de um método para a estimativa da posição da bola. Esse método é baseado em um cálculo estatístico sobre a variação da posição da bola ao longo do tempo, para a realização do cálculo é retirado uma amostra das posições da bola para então ser calculado a média dessas posições nos eixos cartesianos, tal qual a variância e o desvio padrão que são valores que dizem algo a respeito do comportamento da bola em instantes anteriores e que podem ser espelhados para os instantes seguintes (LUIZ GONZAGA MORETTIN, 2010).

As posições da bola são armazenadas em um *buffer* circular com um tamanho ajustável, e tais posições passam pelo filtro simples antes de serem adicionadas. Todas as contas consideram a variação das posições em x e em y - *Figura 5*.

Figura 5 – Estimativa da posição da bola



Fonte: *(NADARAJAH, S., & SUNDARAJ, K., 2011)*

Os cálculos dessa estimativa se baseiam nas fórmulas do espaço amostral (MORETTIN, 2010), para se estimar a posição da bola em um instante futuro, atualmente, são seguidos estes cinco passos:

1. Média das variações da posição da bola $(\bar{x}, \bar{y})$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} buffer(i-1).x - buffer(i).x \quad (3)$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} buffer(i-1).y - buffer(i).y \quad (4)$$

Seja n_b o tamanho do *buffer* circular, como a média das variações (dx, dy) trata das distâncias entre uma posição e a próxima, a quantidade de valores que são levados em conta caem pra $n_b - 1$, por exemplo, se o buffer contém um intervalo de $n_b = 10$

valores, i.e. $buffer(0), \dots, bufer(9)$, a variação que é calculada é dada por $buffer(0).x - buffer(1).x$ fazendo com que o número de valores para o cálculo da média seja de apenas 9 valores, ou $n = n_b - 1$.

Para o cálculo da média das variações em X e Y, são utilizadas as equações (3) e (4), respectivamente.

2. Calculo da Variância Amostral (${s_x}^2$, ${s_y}^2$)

$$ {s_x}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{5} $$

$$ {s_y}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{6} $$

As equações (5) e (6) representam a variância (S^2) de uma amostra da variável X $\{x_1, \dots, x_n\}$ ou da variável Y $\{y_1, \dots, y_n\}$ de n elementos. Ela é definida como a soma ao quadrado dos desvios dos elementos em relação à sua média $(\bar{x}, \bar{y})$ dividido pelo número de amostras menos uma $(n-1)$. A média é dada pelas equações (3) e (4), já vistas.

Para este caso, considerar:

$$ x_i = buffer(i-1).x - buffer(i).x $$
$$ y_i = buffer(i-1).y - buffer(i).y $$

Que são os valores nos eixos X e Y das posições contidas no *buffer*.

3. Cálculo do Desvio Padrão Amostral (S_x, S_y)

$$ s_x = \sqrt{{s_x}^2} = \sqrt{\sum_{i=1}^{n}\frac{(x_i - \bar{x})}{n-1}} \tag{7} $$

$$ s_y = \sqrt{{s_y}^2} = \sqrt{\sum_{i=1}^{n}\frac{(y_i - \bar{y})}{n-1}} \tag{8} $$

Para se obter o Desvio Padrão (S) das variáveis X (7) e Y (8), basta aplicar a raiz quadrada das Variâncias, equações (5) e (6). Para este caso X e Y são as posições de tal objeto em cada eixo. O desvio padrão representa a dispersão de uma distribuição de dados, ou seja, ele serve para mensurar a distância entre cada dado e a média (KHAN ACADEMY, 2018).

4. Determinação do número de pontos (n)

Esse é o último passo para a estimativa da posição da bola. É necessário determinar quantos pontos à frente deseja-se estimar sua posição. A partir da quantidade de pontos escolhida é possível inferir algumas colocações. A primeira é que quanto maior a quantidade de pontos maior é a distância entre o ponto estimado e a posição atual da bola. A segunda é que quanto maior é o número de pontos escolhido, ou, à frente do atual, maior é o erro em relação a posição que ela vai ter.

5. Estimando a posição da bola n posições à frente

$$X_{inf} = buffer(0).x + (\bar{x} - S_x) * n \tag{9}$$

$$Y_{inf} = buffer(0).y + (\bar{y} - S_y) * n \tag{10}$$

Para estimar a posição da bola, leva-se em conta a posição atual dela, como utilizamos o *buffer* circular a primeira posição que é posta nele é sempre a posição mais recente da bola $(buffer(0))$.

Pode-se inferir que, num ambiente ideal, ou seja, em um ambiente em que não existisse o atrito da bola com o carpete e sua velocidade fosse constante, a posição seguinte seria dada pela média da variação da posição da bola (1) somada com a posição atual dela ($buffer(0)$), já que ela sempre se deslocaria com a mesma magnitude, entretanto, para o modelo real, com atrito presente e desaceleração constante, existe um erro ao considerar tal afirmação. Por esse motivo é calculado o desvio padrão (3) da posição da bola, ele mensura a dispersão que existe em relação às médias anteriores,

fazendo com que o “erro” possa ser diminuído para a estimativa das próximas posições da bola.

Seja (X_{inf}, Y_{inf}) a coordenada estimada que o ponto n (4) posições a frente terá, as fórmulas que serão utilizadas para gerá-los são as equações (9) e (10). Sendo x e y as posições da bola, $\bar{x}$ e $\bar{y}$ a média da variação da sua posição, S_x e S_y o desvio padrão da bola e n o número de posições desejadas à frente da atual.

3 METODOLOGIA

Atualmente algumas equipes utilizam a biblioteca OpenCV (OPENCV, 2018), uma biblioteca *open source* em linguagens como C/C++, Java e Python visada para tarefas que utilizam visão computacional, e disponibilizada para o ambiente acadêmico e de pesquisas. A proposta também terá a utilização de uma destas bibliotecas que contém códigos/classes prontos para a implementação do Filtro de Kalman. (OPENCV - KALMAN FILTER CLASS, 2018).

Primeiramente será feito o estudo do sistema, i.e., um estudo para conhecer os aspectos necessários para a implementação do Filtro de Kalman, tais como: funcionamento das funções de Predição e Atualização, a retirada de parâmetros que são necessários para a modelagem das equações matriciais e um plano geral para o desenvolvimento. Alguns dos parâmetros serão retirados de testes experimentais feitos no laboratório da equipe RoboFEI, que contém um campo com ¼ do tamanho do campo oficial da liga B e duas câmeras que filmam cada metade do campo, sendo o ambiente ideal para o levantamento do modelo par ao projeto.

Após entender como as equações e matrizes funcionam e o que elas representam para o FK, em paralelo com a retirada dos parâmetros necessários para o seu funcionamento, as equações serão montadas e poderá ser dado o início à fase de testes junto com as possíveis correções em relação à modelagem.

Caso necessário será considerado o uso do software MATLAB para validação de alguns cálculos, tendo em vista a praticidade de resolução de problemas matriciais, além disso, a obtenção de alguns gráficos poderá ser feita através dele.

Para garantir o funcionamento das equações e do filtro de Kalman serão feitos mais testes dentro de campo e análise dos dados coletados utilizando o campo do laboratório e o software atual, feito em linguagem de programação C++ e usando a IDE QtCreator (QTCREATOR, 2018).

Por fim e talvez a parte mais significativa da pesquisa, após garantir que tanto o filtro e a função preditiva estão funcionando, serão feitos ensaios para fins de comparação entre o dado cru da visão e o Filtro de Kalman. Além disso será comparado, também, a diferença de tempos entre os dados da visão e a predição através do FK. Com os dados coletados em mãos será feita a análise e a conclusão desta pesquisa.

4 RESULTADO PARCIAL

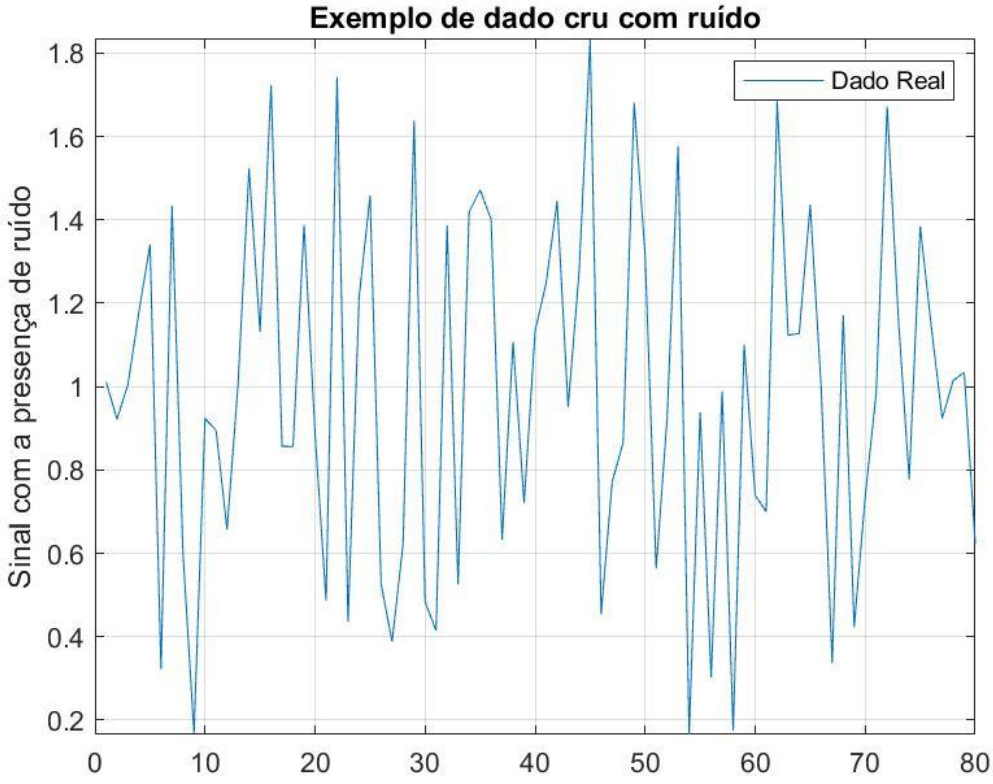
Ao longo dos primeiros meses de pesquisa e desenvolvimento foi adaptado para o nosso software um código do FK para que fosse realizada uma primeira avaliação. Este código foi feito com base em um exemplo simplificado de MATLAB (MATHWORKS).

4.1 EXEMPLO DO FUNCIONAMENTO DO FK NO MATLAB

As figuras abaixo servem como um exemplo para ilustrar o funcionamento do FK. Neste experimento é utilizado o MATLAB para a implementação de um código disponível em um blog online (Introdução ao Filtro de Kalman, 2017).

O código foi modificado para criar um sinal ruidoso com amplitude variada (*Figura 6*), tendo como valor real um sinal contínuo de magnitude uma unidade. Essa variação na amplitude se assemelha com o que acontece com a posição da bola quando ela está em uma região de *overlap*, existindo em posições diferentes, ao mesmo tempo, para cada uma das imagens.

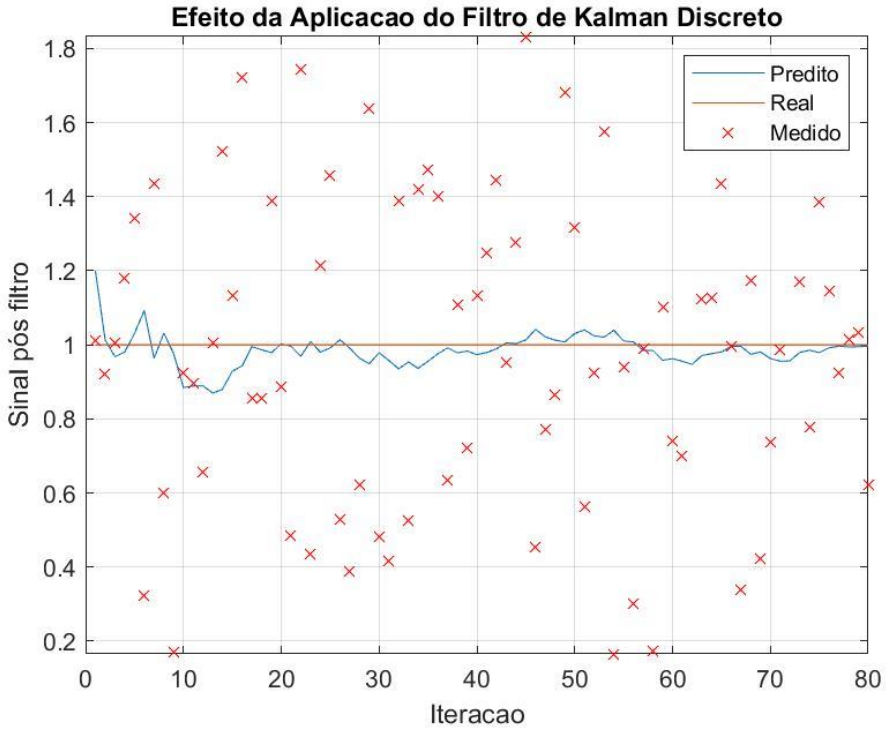
Figura 6- Exemplo dado contaminado com ruído



Fonte: Autor

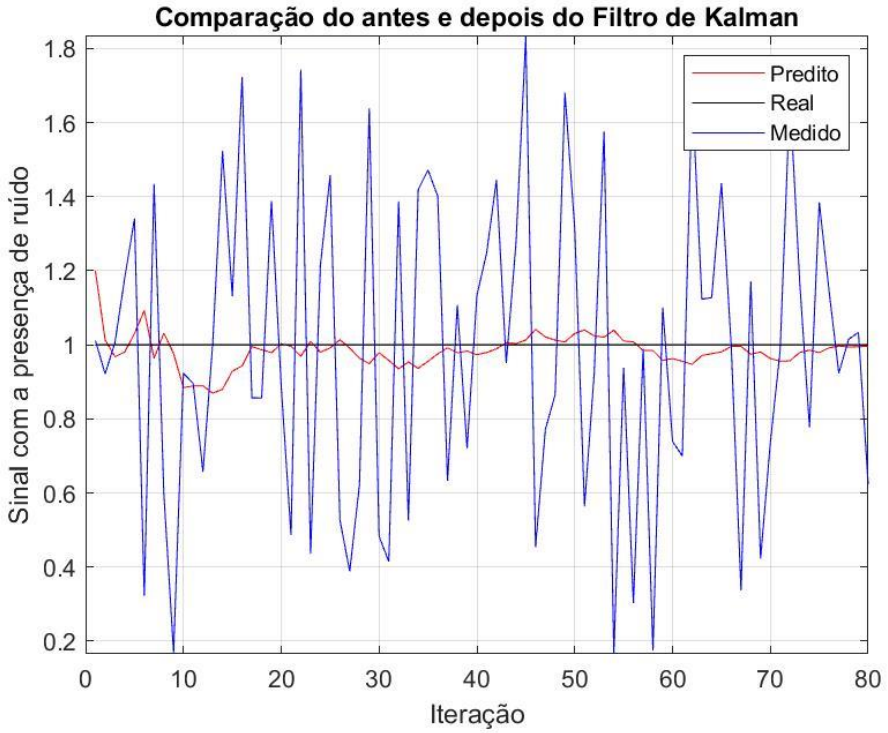
É necessário estimar uma primeira posição como o estado inicial, foi utilizado o valor de 1.2, que é próximo do valor real, para melhor visualização. As próximas figuras (*Figura 7* e *Figura 8*) mostram o efeito do filtro de Kalman ao longo das iterações variando junto com o ruído, todavia convergindo para um valor muito próximo do valor real.

Figura 7 - Sinal resultante do filtro de Kalman



Fonte: Autor

Figura 8 - Comparação dos sinais pré e pós filtro



Fonte: Autor

Ambos os gráficos (*Figura 7* e *Figura 8*) representam a mesma coisa, porém na *Figura 7*, para uma melhor visualização, os dados pontuais em vermelho não estão ligados uns aos outros e simulam as medições provenientes da câmera com o efeito de *overlap*. A linha laranja e continua em 1un é o sinal real que se espera que o filtro obtenha e a linha em azul representa o valor predito pelo filtro de Kalman. É possível notar que o filtro funciona e apresenta um resultado satisfatório para a eliminação de ruídos, mas deve-se levar em conta que para este exemplo muitas das matrizes de controle foram considerados como unitárias ou foram desprezadas por não influenciarem no cálculo.

4.2 DESEMPENHO NA LARC 2018

Com o código do exemplo acima adaptado para o uso da equipe, foi possível visualizar uma melhora significativa, ainda nos testes dentro do laboratório, em relação ao filtro de média simples e um avanço em diminuir o efeito dos ruídos das câmeras e das regiões de *overlap*. Os objetos em campo, a bola e os robôs, tiveram uma diminuição da variação de suas posições. Este FK simplificado implementado dentro do software foi utilizado no período da competição da LARC 2018 (LATIN AMERICAN AND

BRAZILIAN ROBOTICS COMPETITION, 2018), junto com o avanço do software em si, acarretou em uma melhora no desempenho da equipe RoboFEI e como consequência foi obtido o terceiro lugar na competição.

É importante levar em conta que apesar do filtro adaptado ter cumprido seu papel de forma positiva, aprimorando a precisão das posições dos objetos, ele ainda não é o adequado ao modelo previsto pelo projeto e pela teoria a respeito do FK.

4.3 LEVANTAMENTO DO MODELO MATEMÁTICO DO FK

Existem variações no modelo do filtro de Kalman para diferentes tipos de variáveis observadas e de processos. O modelo que será utilizado nesta pesquisa, é chamado de *Discrete Kalman filter*, i.e, filtro de Kalman Discreto ou DKF (WELCH e BISHOP, 2006). O DKF é utilizado nos modelos discretos no tempo em que o processo que se deseja controlar apresenta uma equação diferencial estocástica linear (WELCH e BISHOP, 2006), isso significa que apesar de um ou mais termos da equação serem estocásticas, ou seja, com origem aleatória e dependentes de acontecimentos anteriores (PUC-RIO), seu modelo físico não apresenta mudanças no tempo. O DKF vai ser utilizado pois o comportamento da bola ao longo do tempo, em campo, é linear e regido pelas equações de movimento, sendo importante apenas seus valores discretizados e não a origem do movimento. O modelo é conhecido, e assim, o processo pode ser representado por equações diferenciais.

Todo o DKF tem sua representação feita pelo método de espaço de estados, o que facilita a parte de cálculos que serão feitos em forma matricial. Para o caso desta pesquisa foram utilizadas as equações diferenciais que descrevem o movimento do objeto de estudo, são elas de posição e velocidade.

Dada por:

$$X_k = X_{k-1} + \dot{X}_{k-1}.\Delta t + \frac{1}{2}.a_x.\Delta t^2 \tag{11}$$

$$Y_k = Y_{k-1} + \dot{Y}_{k-1}.\Delta t + \frac{1}{2}.a_y.\Delta t^2 \tag{12}$$

$$\dot{X}_k = \dot{X}_{k-1} + a_x.\Delta t \tag{13}$$

$$\dot{Y}_k = \dot{Y}_{k-1} + a_y.\Delta t \tag{14}$$

Seja k o tempo amostrado da variável, X e Y representando as posições em cada eixo, $\dot{X}$ e $\dot{Y}$ suas derivadas de primeira ordem, ou seja, velocidade em cada eixo, a_x e a_y o coeficiente de desaceleração do carpete e Δt a variação de tempo entre cada amostra.

Após apresentadas as equações que descrevem o comportamento do processo, é necessário saber como relacioná-las ao modelo matemático do DKF. Esta parte irá descrever quais são as fases e as equações utilizadas no filtro, assim como o que significa cada termo da equação.

4.3.1 ETAPAS DO FILTRO DE KALMAN

As variações do FK apresentam sempre estas duas etapas no cálculo do filtro: a Predição e a Atualização.

PREDIÇÃO

1. *Predição do Estado (a priori)*

$$\hat{X}_{k|k-1} = F_k \hat{X}_{k-1|k-1} + B_k u_k \tag{15}$$

2. *Predição de covariância*

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{16}$$

ATUALIZAÇÃO

3. *Resíduo da medição*

$$Y_{k|k} = Z_k - H_k \hat{X}_{k|k-1} \tag{17}$$

4. *Resíduo da covariância*

$$S_{k|k} = H_k P_{k|k-1} H_k^T + R_k \tag{18}$$

5. *Ganho Kalman*

$$K_{k|k} = P_{k|k-1} H_k^T S_k^{-1} \tag{19}$$

6. *Estado atualizado (a posteriori)*

$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k Y_k \tag{20}$$

7. *Covariância estimada*

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{21}$$

Sendo k a referência para o estado atual e $k-1$ para o estado anterior, ou também a *priori* e a *posteriori*, respectivamente. A notação ^ é uma referência à uma estimativa pois, para a primeira etapa do filtro é necessário uma estimativa das variáveis de estados daquele objeto, ou seja, uma posição estimada inicial e uma velocidade estimada inicial. Com esse estado estimado, é gerado a predição do estado a *priori* que também é uma estimativa, afinal, o DKF é um estimador ótimo, então apesar de conseguir convergir as variáveis de estado para um valor muito próximo do real, elas continuam sendo estimadas com base no modelo do processo que foi utilizado no filtro. Por isso é necessário que o processo seja linear, para que as estimativas se tornem uma representação muito próxima do real, o modelo não pode variar.

As equações (11), (12), *(12)*(13) e (14) podem ser representadas, ou reescritas, na forma matricial, mostrada abaixo:

$$\begin{bmatrix} X \\ Y \\ \dot{\mathrm{X}} \\ \dot{\mathrm{Y}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta\mathrm{t} & 0 \\ 0 & 1 & 0 & \Delta\mathrm{t} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{X}_{\mathrm{k}-1} \\ \mathrm{Y}_{\mathrm{k}-1} \\ \dot{\mathrm{X}}_{\mathrm{k}-1} \\ \dot{\mathrm{Y}}_{\mathrm{k}-1} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{22}$$

Para o modelo do DKF, considere necessário uma ou um conjunto de variáveis que se deseja observar e também estimar seus estados, sejam eles passados ou futuros. Este conjunto de variáveis será chamado de “Variáveis de Estado”. Sendo k a referência para o estado atual e $k-1$ para o estado anterior, ou também a priori e a posteriori, respectivamente.

O conjunto de variáveis de estado ($\boldsymbol{X_k}$) é dado por:

$$X_k = \begin{bmatrix} X \\ Y \\ \dot{\mathrm{X}} \\ \dot{\mathrm{Y}} \end{bmatrix} \tag{23}$$

A matriz $\boldsymbol{F_k}$, chamada de modelo de transição de estados, de dimensão $n \times n$ representa a relação do estado em $\boldsymbol{k-1}$ com a do estado em $\boldsymbol{k}$. Dependendo do modelo adotado, ela pode variar a cada iteração. Seja $\boldsymbol{F_k^T}$ a matriz transposta de $\boldsymbol{F_k}$.

A matriz do modelo de transição de estados ($\boldsymbol{F_k}$) é dada por:

$$F_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

A matriz $\boldsymbol{B}$, de dimensão $n \times l$ está relacionada à alguma entrada de controle que esteja presente no modelo. Para o caso desta pesquisa o produto da matriz $\boldsymbol{B}$ com a matriz $\boldsymbol{u}$ resultará nos escalares de desaceleração das componentes da bola para x e y ao longo do tempo, i.e., representa o atrito entre o campo e a bola.

As matrizes ($\boldsymbol{B}$) e ($\boldsymbol{u}$) são dadas abaixo:

$$B = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \tag{25}$$

$$u = \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{26}$$

Como o DKF pode fornecer uma estimativa de um estado no futuro utilizando apenas um valor das variáveis de entrada em $\boldsymbol{k-1}$ e um valor atual realimentado em $\boldsymbol{k}$, pode-se inferir que o processo apresenta ruídos e/ou distúrbios quando as informações, tais como posição e velocidade, são inseridas no filtro. O papel das matrizes $\boldsymbol{Q}$ e $\boldsymbol{R}$ é justamente tentar quantizar a perturbação a fim de corrigi-la durante as etapas do filtro. A matriz $\boldsymbol{Q}$ é chamada de matriz de covariância do processo e $\boldsymbol{w_k}$ representa o ruído que existe proveniente do processo. A matriz $\boldsymbol{R}$ é chamada de matriz de covariância das medições (ou observações) e $\boldsymbol{v_k}$ representa o ruído presente nas novas informações que são inseridas no filtro na etapa de Atualização. Dado que ambas são independentes (entre si) e com ruído Gaussiano Branco, i.e., com uma distribuição normal com média 0.

$$p(w_k) \sim N(0, Q_k)$$
$$p(v_k) \sim N(0, R_k)$$

Dependendo o processo que está sendo estudado e o acesso às informações das variáveis de estado, obter a matriz $\boldsymbol{R}$ é quase sempre mais simples. A matriz $\boldsymbol{Q}$ contém a quantificação da variância e da covariância dos estados, o que é quase sempre difícil de ser determinado. A matriz $\boldsymbol{R}$ será composta pelos valores da variância das posições em x e y. Outra informação importante a respeito de como as matrizes de covariância alteram o comportamento do filtro é que quanto maior for $\boldsymbol{Q}$ mais a resposta do DKF tende a convergir para as medições que estão sendo feitas, ou seja, ele quase não cumpre o seu papel de filtro, entretanto ele converge para os valores com um número muito baixo de iterações. Já aumentando o valor de $\boldsymbol{R}$ o DKF converge para valores mais próximos do valor real/esperado, entretanto o custo para se chegar nisso cresce e são necessários um número maior de iterações para chegar àquela resposta.

$$Q = Q.Q^{\mathrm{T}} = \begin{bmatrix} \frac{1}{4}\Delta t^4 & 0 & \frac{1}{2}\Delta t^3 & 0 \\ 0 & \frac{1}{4}\Delta t^4 & 0 & \frac{1}{2}\Delta t^3 \\ \frac{1}{2}\Delta t^3 & 0 & \Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^3 & 0 & \Delta t^2 \end{bmatrix} \tag{27}$$

$$R = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \tag{28}$$

A matriz $\boldsymbol{P}$ é chamada de matriz de covariância do erro das variáveis de estado. Ela pode ser dita como uma representação da acurácia dos estados estimados, estima a quantidade de erro na predição dos estados. Ela é calculada a cada iteração, logo, seus valores também mudam.

O conjunto das novas medições ($\boldsymbol{Z_k}$) corresponde a novos valores no processo para as variáveis de estado, essas medições serão adicionadas ao filtro na parte de Atualização. $\boldsymbol{Z_k}$ tem a mesma dimensão de $\boldsymbol{X_k}$.

A matriz $\boldsymbol{H}$ é a matriz Jacobiana do modelo dinâmico do sensor/observação. Ela é responsável por linearizar o valor predito pelo DKF e compará-lo com a nova medida inserida ($\boldsymbol{Z_k}$). Seja $\boldsymbol{H^T}$ a matriz transposta de $\boldsymbol{H}$.

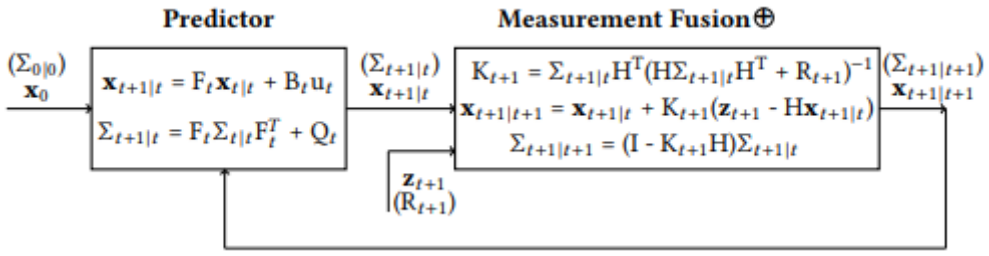
$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (29)$$

O filtro começa na fase de Predição. A primeira parte (1) é estimar as variáveis de estado ($\boldsymbol{X_k}$), com os valores iniciais da suas variáveis, i.e., os valores de $x_0, y_0, \dot{x}_0, \dot{y}_0$. Na segunda parte (2) é calculado o erro da predição dos estados feita em (1) medição, normalmente a matriz $\boldsymbol{P}$ é iniciada com valores igual a 1.

Após (1) e (2) entra a fase de Atualização. As partes (3) e (4) não existem nas contas originais do FK, essas passagens estão inseridas direto nas passagens seguintes (5),(6) e (7) , todavia, foi optado por separá-las a fim de dar uma explicação mais detalhada no que está ocorrendo dentro do filtro. Em (3) é calculado o resíduo da medição, esse resíduo mostra a discrepância entre as variáveis de estado estimadas $\boldsymbol{(H.X_k)}$ e a medida real delas $\boldsymbol{(Z_k)}$. Quando esse resíduo é zero significa que não existe erro aparente entre eles. Na passagem (4) se calcula o resíduo da covariância ($\boldsymbol{S}$), comparando o erro real com o predito, uma vez que $\boldsymbol{H}$ lineariza o estado com a medição é possível saber como o "sensor" que captura os novos valores de estado $\boldsymbol{(Z_k)}$ se comporta adicionando a covariância existente na medição ($\boldsymbol{R}$). Em (5) é calculado o ganho de Kalman ($\boldsymbol{K}$), ele é importante pois serve para minimizar o erro de covariância *a posteriori* (7). O FK é do tipo recursivo, por esse motivo ele pode conseguir fazer estimativas a respeito de estados, em tempos passados e futuros, utilizando apenas duas amostras, além disso, ele também é chamado de estimador ótimo, i.e., dado este número de amostras limitadas ele consegue entregar o conjunto das variáveis de estado com erro de variância mínimo (PUC-RIO). Nesta etapa se utiliza o inverso da matriz resultante do resíduo da covariância ($\boldsymbol{S^{-1}}$). As duas últimas passagens (6) e (7) são as partes que demonstram isso, elas tem o propósito de atualizar as variáveis recursivas do DKF. Essas variáveis são as variáveis de estados ($\boldsymbol{X_k}$) e a matriz de covariância do erro ($\boldsymbol{P}$), elas irão realimentar o filtro, porém na próxima iteração elas passam a ser a estimativa *a priori* e repetem todo o processo das equações para gerar uma nova estimativa, e assim por diante.

Todas as etapas comentadas acima estão presentes na figura (*Figura 9*) que mostra o funcionamento do DKF.

Figura 9 – Diagrama representativo das etapas e funcionamento do DKF



Fonte: *(PEI, S. FUSSELL*, et al.*, 2017)*

Para o DKF conseguir funcionar, são necessários alguns pré-requisitos, como conhecer estimativas iniciais das variáveis de estado e do erro da covariância, saber interpretar as propriedades do sinal ruidoso e do modelo de espaço de estados (FADALI).

5 RESULTADOS FINAIS

Após a extração do modelo dinâmico do objeto que se deseja observar, é possível dar início a fase de teste com o modelo real. As equações servirão para que o DKF estime o comportamento deste objeto com um certo índice de confiança. Quanto mais próximo o modelo levantado for da realidade, mais certeza pode-se ter na estimativa fornecida pelo filtro.

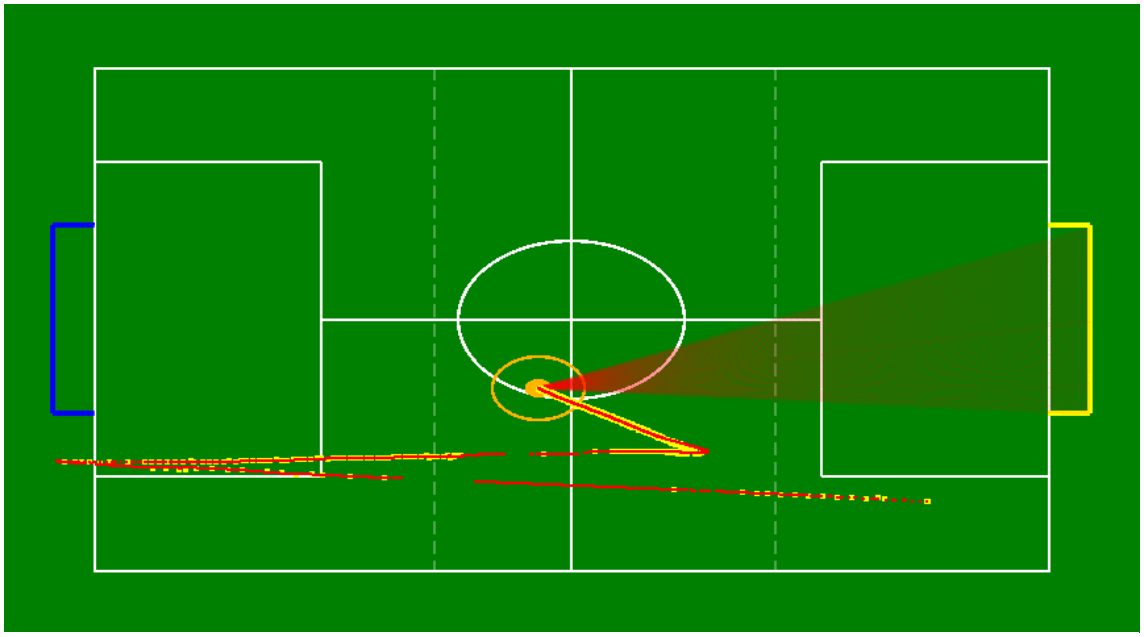
O filtro foi inicialmente utilizado para corrigir o problema de *overlapping* e da posição dos objetos em campo, entretanto, se viu a possibilidade de potencializar seu uso em suprir a falta de informações durante o período entre o recebimento dos pacotes de informações da visão. Todo o procedimento foi feito via software da equipe utilizando um objeto da biblioteca *Qtimer* do próprio QtCreator. Para que o software gerasse internamente novas informações durante este *gap*, foi atribuído ao objeto *timer* o tempo de 1 milissegundo (menor tempo suportado pela biblioteca). A cada estouro do *timer* um sinal é gerado e uma rotina é chamada. Nessa rotina o DKF realiza uma nova estimativa acerca das posições dos objetos, apenas a fase de Predição é utilizada para gerar essas informações, afinal, uma nova medida do sensor (câmera) ainda não está disponível para fazer a atualização do filtro. Com esta medida é possível preencher a lacuna de informações deixada durante o período entre cada pacote da visão, deixando as funções, que dependem de tais informações, atualizadas praticamente a todo instante de tempo.

5.1 RESULTADOS DOS TESTES EM CAMPO

Para os testes em campo foi empregado o modelo dinâmico da bola, como ela apresenta um comportamento linear pode-se dizer que as variáveis de estado geradas pelo DKF são de alta confiança e certeza, se aproximando da medida real. Nas figuras abaixo, estão os resultados de alguns testes dentro do campo do laboratório da equipe. Nestes testes a bola é lançada dentro do campo percorrendo um trajeto qualquer, e sua posição dada pelas câmeras e a estimada pelo DKF foram guardadas e representadas nas figuras. O trajeto em amarelo corresponde a posição da bola provida pela câmera, sendo possível observar que apenas em alguns trechos existem poucos ou um único ponto desenhado. Isto pode acontecer devido a alguns fatores, como a bola estar com uma velocidade muito elevada ou a alguma má calibração que faz com que a bola "desapareça" em certo trecho. O trajeto em vermelho corresponde a posição gerada pelo filtro, sendo possível observar

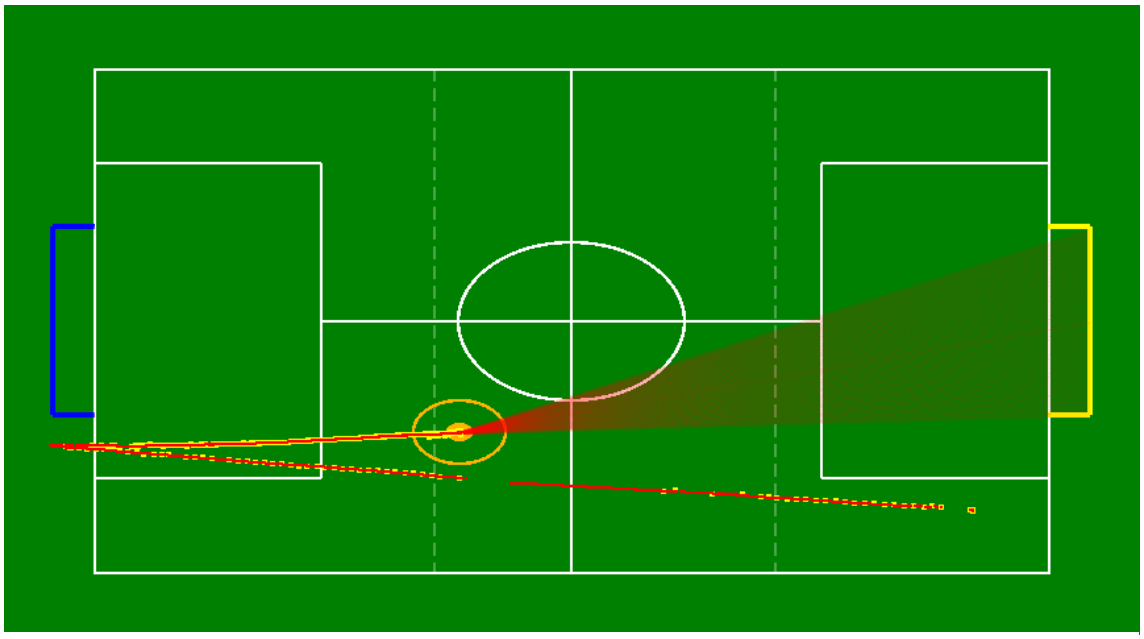
que, com o auxílio do objeto *timer*, a informação de posição é quase constante, tendo o trajeto preenchido quase em sua totalidade e formando um único e sólido caminho.

Figura 10 - Comparação dos trajetos formados pelas informaçãos do SSL-Vision e o DKF.



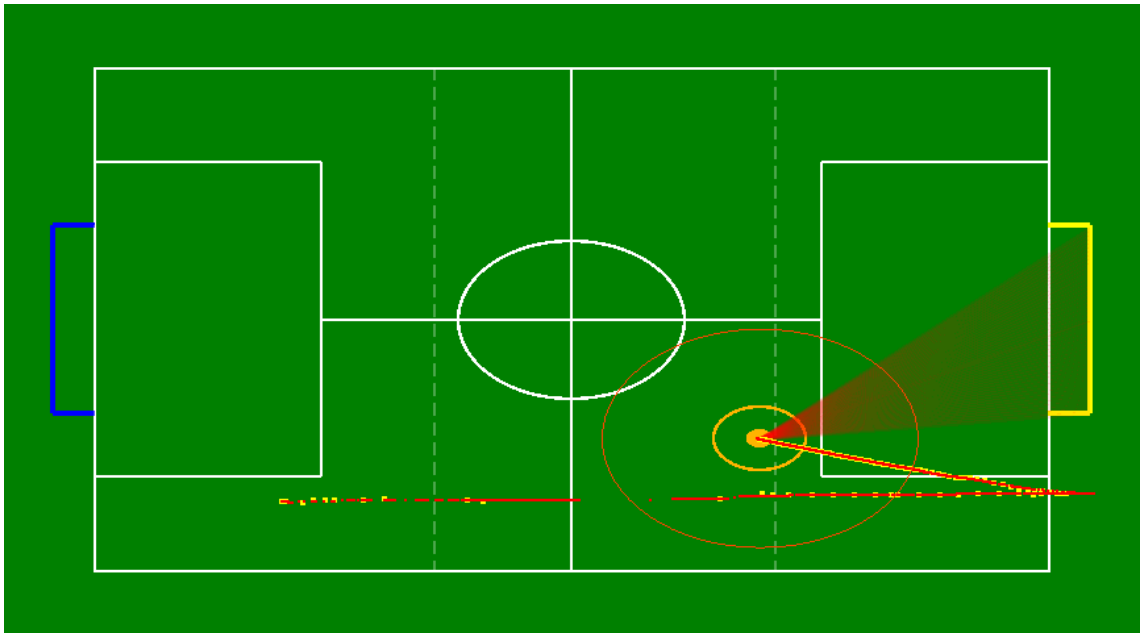
Fonte: Autor

Figura 11 - Comparação dos trajetos formados pelas informaçãos do SSL-Vision e o DKF



Fonte: Autor

Figura 12- Comparação dos trajetos formados pelas informaçãos do SSL-Vision e o DKF



Fonte: Autor

É possível observar, também, pequenos erros de trajeto quando a bola colide com as paredes ou algum robô. Isto acontece porque no momento da colisão o modelo da bola deixa de ser linear e o filtro leva algumas iterações para poder convergir de volta ao valor real. Uma solução para isso é resetar o filtro e alguns de seus parâmetros.

5.2 ESTUDO DE TEMPO

Utilizando o software da equipe, foram feitos estudos de tempo entre os pacotes provindos do SSL-Vision e do tempo de execução da rotina chamada pelo timer (que faz a atualização do estado dos objetos utilizando o DKF). Foram coletadas amostras de cada um desses tempos de um determinado trajeto. Essas amostras revelam números interessantes a respeito da implementação do *timer*, comparando-se ao período entre pacotes da visão. Os gráficos das figuras *Figura 13* e *Figura 14* e correspondem, respectivamente, aos períodos de cada amostra entre pacotes e do período entre cada atualização via DKF, junto com sua respectiva reta média sobre o total de amostras. Também foi possível construir uma tabela de tempos (*Tabela 1*) contendo os valores de máximos, mínimos e a média que as amostras de cada situação apresentaram ao longo do período de aquisição de dados, a quantidade de amostras que superaram o tempo médio

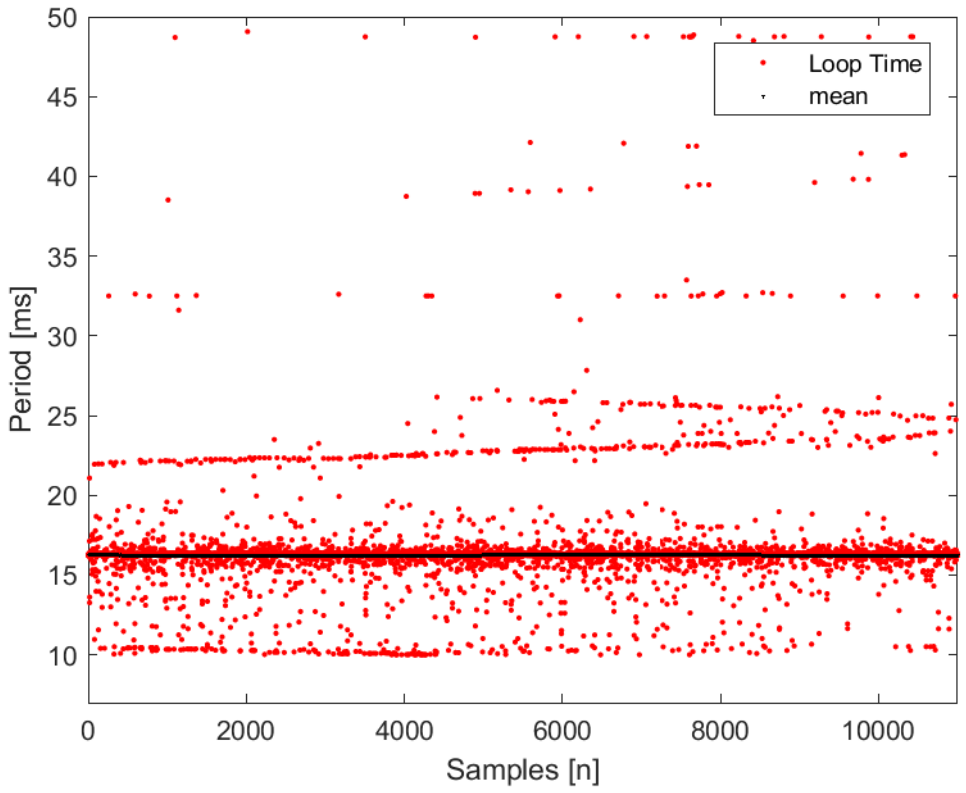
entre pacotes da visão (>17ms) e a quantidade de amostras da rotina de atualização que superaram valores de tempo de estouro do *timer* (>1ms).

Tabela 1- Comparação de tempos entre as amostras

	Pacotes do SSL-Vision	*Rotina com o DKF*
Número de amostras	11.000	11.000
Tempo mínimo ***[ms]***	10,0031	0,5141
Tempo máximo ***[ms]***	835,5530	8,4065
Tempo médio ***[ms]***	16,2442	0,5291
Número de amostras com período maior que 1ms	Não se aplica	95
Número de amostras com period maior que 17ms	1030	Não se aplica
Representatividade (do total) ***[%]***	9,3636	0,8636

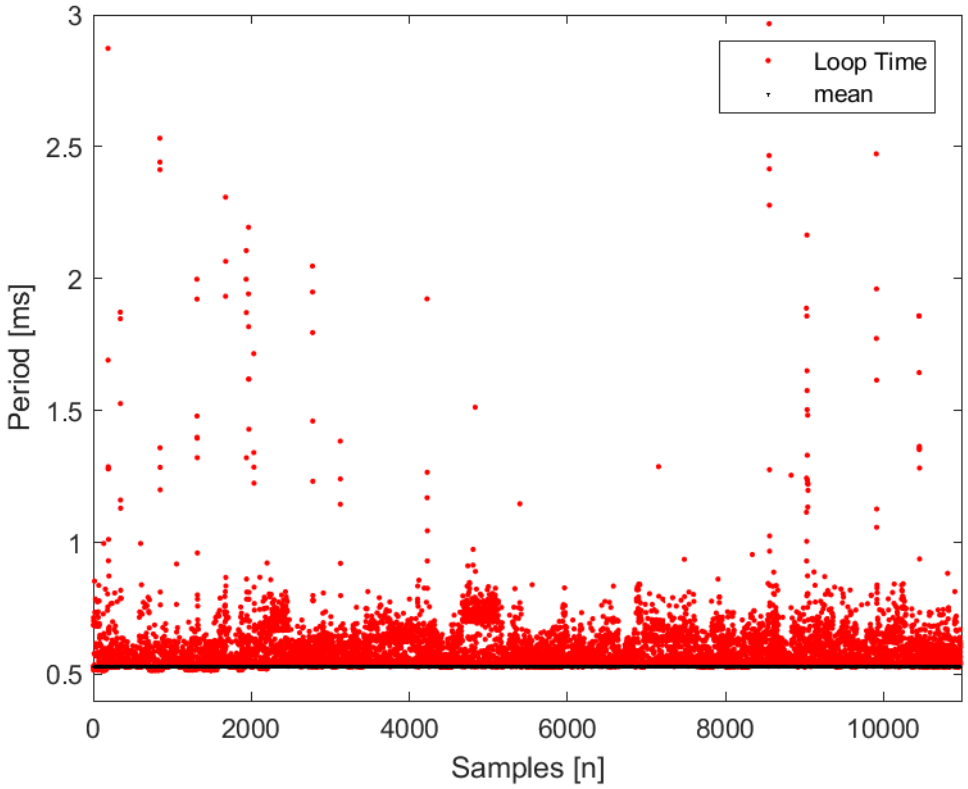
Fonte: Autor

Figura 13 - Amostras de tempo do SSL-Vision



Fonte: Autor

Figura 14 - Amostras de tempo da rotina de atualização com o DKF



Fonte: Autor

Com as informações apresentadas pela tabela 1 associadas às imagens dos testes com a bola em campo, é possível identificar que existe uma diferença entre utilizar ou não o modelo junto ao filtro para atualização das informações e que essa diferença passa a ser grande e influenciadora se a equipe quiser que o desempenho em função da latência seja melhorado. É possível perceber, também, que a porcentagem de valores que extrapola a média de quase 17ms entre pacotes da visão é significativamente alto, próximo de 10%, principalmente se comparado à porcentagem de amostras da atualização pelo DKF, que são maiores do que o próprio estouro do timer de 1ms, próximo de 0.87%. Estes dados revelam uma grande variância de tempo entre a chegada de um pacote e outro, tornando a atualização apenas via SSL-Vision muito instável apesar de confiável, enquanto é possível perceber que a rotina de atualização pelo DKF acontece, quase em sua totalidade, dentro de 1 ms.

Ao se utilizar das médias de tempos entre os dois métodos, a diferença se torna ainda mais evidente sendo possível, através do timer, executar em até 30 vezes a atualização das informações dos objetos em campo até a chegada de uma nova informação

via SSL-Vision. Um número muito significativo para uma categoria com uma grande dinâmica e interação entre os robôs, especialmente em termos de rodar a ‘estratégia’ a tempos muito rápidos.

6 CRONOGRAMA

Tabela 2- Cronograma de atividades da pesquisa

	Meses											
Atividade	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Estudo do sistema												
Modelagem das equações												
Simulações/ajustes quanto ao funcionamento das equações												
Relatório Parcial												
Testes finais												
Comparativos												
Relatório Final												

Fonte: Autor

A cor azul representa os meses que já decorreram do tempo total da pesquisa. A cor amarela representa o acréscimo no tempo de desenvolvimento na parte de simulações e ajustes. A cor cinza representa o tempo restante das partes da pesquisa até o tempo total (12 meses).

Estudo do sistema – Para se obter sucesso na resposta do filtro de Kalman, é necessário estudar o sistema que se deseja fazer o *tracking*. Nesta parte serão retirados os parâmetros para que seja possível a modelagem das equações.

Modelagem das equações – Com os parâmetros em mãos, serão realizadas as modelagens das equações que o filtro necessita, em paralelo a isso, serão realizadas simulações para sua validação.

Simulações e ajustes – Nesta parte será feita verificação das equações que foram empregadas, e possíveis ajustes serão feitos para que se obtenha a melhor resposta possível.

Testes finais – Quando o modelamento já estiver adequado e a resposta do filtro estiver satisfatória, serão aplicados os últimos testes.

Comparativos – Serão levantados gráficos e outros tipos de analises para comprovar o funcionamento do filtro, mostrando que a necessidade foi atendida.

7 CONCLUSÃO

A correlação entre este trabalho e o que outras equipes já fazem é a utilização dos modelos dinâmicos dos objetos em campo e, através do DKF ou de redes neurais, predizer o estado futuro do jogo para ser possível minimizar os problemas de latência ao mandar os comandos corrigidos para os robôs que já contam com a influência do *delay*. A principal diferença é que a proposta apresentada neste artigo é lidar com o fato dos *delays* existirem, mas fazer com que ele não seja influente na atualização dos objetos através dos modelos empregados utilizando uma rotina para realizar esta atualização sem que seja necessário predizer os comandos como forma de compensação, que é uma nova aproximação para solucionar este problema.

Ao se comparar ambos os trajetos nas figuras e a *Tabela 1*, é possível perceber a visível vantagem de se utilizar o timer com o auxílio do DKF para preencher a lacuna deixada pela dependência da capacidade da câmera em prover novas informações. Pelos testes, foi comprovado que este é um modo viável e com custo praticamente nulo para o processamento das informações dentro do software, trazendo apenas benefícios para o desempenho da equipe.

8 CONSIDERAÇÕES FINAIS

O período de desenvolvimento até a entrega do relatório final foi de muito aprendizado sobre o funcionamento do filtro de Kalman e os parâmetros necessários para que fosse possível implementá-lo. Atualmente o código do DKF foi feito e está diretamente no Software da equipe RoboFEI sem o auxílio de outras bibliotecas, contrário ao que foi proposto no relatório inicial.

O projeto de pesquisa de iniciação cientifica propôs um novo e grande desafio. Ao iniciar as pesquisas, do relatório inicial até o final, foi possível aprender coisas novas e diferentes que culminaram em um grande acumulado de conhecimentos que poderão ser usados em ambientes fora do de futebol de robôs. A implementação do filtro também foi importante para melhorar o aprendizado para controle digital e, também, serviu de aprendido antecipado às matérias de engenharia que estavam por vir.

A necessidade de estender um pouco o período de simulações e ajustes presente no cronograma (cor amarela) foi essencial para aprimorar o funcionamento do filtro. Apesar deste acontecimento, os testes no laboratório da equipe e o desempenho do filtro nos jogos da RoboCup 2019 – Sydney e LARC/CBR 2019 – Rio Grande foram relevantes para o desempenho. Atrelado a isso, o desenvolvimento da equipe nas diferentes áreas foi reflexo dos resultados obtidos nas competições: quinto lugar na categoria B da RoboCup e campeão da LARC/CBR.

Resumidamente, esta pesquisa foi de grande importância para o desenvolvimento pessoal e para contribuir com a necessidade de avanço que a equipe de futebol de robôs tinha.

REFERÊNCIAS

CMDRAGONS. **Extended Team Description**. [S.l.]. 2015.

DAVID S. MOORE. **A estatística básica e sua prática**. Rio de Janeiro: LTC - livros tecnicos e cientificos editora LTDA, 2014.

ER-FORCE. **Team Description Paper for RoboCup 2008**. [S.l.]. 2008.

ER-FORCE. **Team Description Paper for RoboCup 2012**. [S.l.]. 2012.

FADALI, M. S. **Discrete Kalman Filter**. University of Nevada. [S.l.].

HUMANOID League. **RoboCup Humanoid League**, 2018. Disponivel em: <https://www.robocuphumanoid.org/>. Acesso em: 20 Setembro 2018.

IEEE Very Small Size Soccer. **CBR - VSSS**, 2018. Disponivel em: <http://www.cbrobotica.org/?page_id=81>. Acesso em: 25 Setembro 2018.

INTRODUÇÃO ao Filtro de Kalman, 2017. Disponivel em: <https://blogdocontroleiro.wordpress.com/2017/08/01/introducao-ao-filtro-de-kalman-com-exemplo-no-matlab/.>. Acesso em: 23 Setembro 2018.

JUNIOR, F. R. **Sistema de interceptação de bolas no Futebol**. Centro Universitário da FEI. SBC. 2015.

KALMAN, R. E. **A new approach to linear filtering and prediction problems (adaptado)**. [S.l.]. 1960. Texto adaptado por: John Lukesh.

KHAN ACADEMY. Khan Academy, 2018. Disponivel em: <https://pt.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/variance-standard-deviation-sample/a/population-and-sample-standard-deviation-review>. Acesso em: 10 Setembro 2018.

LATIN AMERICAN AND BRAZILIAN ROBOTICS COMPETITION, 2018. Disponivel em: <http://www.cbrobotica.org/>. Acesso em: 29 Setembro 2018.

LUIZ GONZAGA MORETTIN. **Estatistica básica:** probabilidade e inferência. São Paulo: Pearson Prentice Hall, v. Único, 2010.

MATHWORKS. MATLAB. Disponivel em: <https://www.mathworks.com/products/matlab.html>. Acesso em: 05 Abril 2019.

MERVEN, B.; NICOLLS, F.; DE JAGER, G. **Multi-Camera person tracking using an extended Kalman Filter**. [S.l.]. 2003.

MIDDLE Size League. **RoboCup Middle Size League**, 2018. Disponivel em: <https://msl.robocup.org/>. Acesso em: 20 Setembro 2018.

MORETTIN, L. G. Análise exploratória dos dados de uma amostra. In: _____ **Estatítica básica:** probabilidade e inferência. [S.l.]: Pearson, 2010. Cap. 8, p. 189.

NADARAJAH, S., & SUNDARAJ, K. **A survey on team strategies in robot soccer: team strategies and role description.** [S.l.], p. 271-304. 2011.

NATIONAL INSTRUMENTS. NI - National Instruments , 2018. Disponivel em: <http://www.ni.com/pt-br/shop/labview.html>. Acesso em: 20 Setembro 2018.

NUBOTS TEAM. **The 2003 NUbots Team Report**. [S.l.]. 2003.

OPENCV - KALMAN FILTER CLASS. Kalman Filter Class Reference. **OpenCV**, 2018. Disponivel em: <https://docs.opencv.org/trunk/dd/d6a/classcv_1_1KalmanFilter.html>. Acesso em: 10 Setembro 2018.

OPENCV. **OpenCV**, 2018. Disponivel em: <https://opencv.org/>. Acesso em: 10 Setembro 2018.

PARSIAN. **Team Description for Robocup**. [S.l.]. 2010.

PEI, Y. et al. **An Elementary Introduction to Kalman Filtering**. University of Texas at Austin. [S.l.]. 2017.

PUC-RIO. **Filtro de Kalman**. PUC. Rio de Janeiro.

QTCREATOR, 2018. Disponivel em: <https://www.qt.io/>. Acesso em: 20 Setembro 2018.

ROBOCUP @Home league. **@Home league**, 2018. Disponivel em: <http://www.robocupathome.org/>. Acesso em: 20 Setembro 2018.

ROBOCUP. **Wiki RoboCup**, 1997. Disponivel em: <http://wiki.robocup.org/Small_Size_League>. Acesso em: 23 Agosto 2018.

ROBOCUP Small Size League, 2018. Disponivel em: <http://www.robocup.org/leagues/7>. Acesso em: 19 set. 2018.

ROBOFEI, 2018. Disponivel em: <https://fei.edu.br/robofei/>. Acesso em: 20 Setembro 2018.

ROBOIME. **RoboIME: on the road to RoboCup 2017**. Instituto Militar de Engenharia. Rio de Janeiro. 2017.

S. ZICKLER, T. LAUE, O. BIRBACH, M. WONGPHATI, M. VELOSO. **The Shared Vision System for the RoboCup SmallSize League**. [S.l.]. 2009.

SSL-VISION DEVELOPER TEAM. RoboCup Small Size League Shared Vision System, 2009. Disponivel em: <https://github.com/RoboCup-SSL/ssl-vision>. Acesso em: 19 Setembro 2018.

UAISOCCER. **TDP da Equipe UaiSoccer de Futebol de Robôs – Small Size**. UFSJ. MG. 2013.

WARTHOG ROBOTICS. **Description of the Warthog Robotics SSL 2016**. Universidade de São Paulo. São Carlos. 2016.

WELCH, G.; BISHOP, G. **An Introduction to the Kalman Filter**. University of North Carolina. [S.l.]. 2006.

ZERVOS, M. **Real time multi-object tracking**. [S.l.]. 2012.