UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ESTUDO DE UMA TÉCNICA PARA O TRATAMENTO DE *DEAD-TIMES* EM OPERAÇÕES DE RASTREAMENTO DE OBJETOS POR SERVOVISÃO

DIEGO SAQUI

ORIENTADOR: PROF. DR. EDILSON REIS RODRIGUES KATO

São Carlos - SP
Abril/2014

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ESTUDO DE UMA TÉCNICA PARA O TRATAMENTO DE *DEAD-TIMES* EM OPERAÇÕES DE RASTREAMENTO DE OBJETOS POR SERVOVISÃO

DIEGO SAQUI

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Inteligência Artificial.
Orientador: Dr. Edilson Reis Rodrigues Kato.

São Carlos - SP
Abril/2014

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

S242et

Saqui, Diego.
Estudo de uma técnica para o tratamento de *dead-times* em operações de rastreamento de objetos por servovisão / Diego Saqui. -- São Carlos : UFSCar, 2014.
127 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2014.

1. Inteligência artificial. 2. Servovisão. 3. Kalman, Filtragem de. 4. *Dead-times*. 5. Rastreamento de objetos. 6. Controlador de velocidades. I. Título.

CDD: 006.3 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Estudo de uma técnica para o tratamento de Dead-times em operações de rastreamento de objetos por Servovisão"

Diego Saqui

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Prof. Dr. Edilson Reis Rodrigues Kato
(Orientador - DC/UFSCar)

Prof. Dr. Roberto Hideaki Tsunaki
(EESC/USP)

Prof. Dr. Rafael Vieira de Souza
(FZEA/USP/Pirassununga)

Prof. Dr. Glauco Augusto de Paula Caurin
(EESC/USP)

São Carlos
Maio/2014

AGRADECIMENTOS

Agradeço a Deus por permitir a conclusão de mais uma etapa em minha vida.

Agradeço a minha família e minha noiva pelo apoio e suporte necessários para que eu consegui-se concluir esse trabalho.

Agradeço ao professor Edilson R. R. Kato, que foi meu orientador pela confiança, dando apoio e suporte e compartilhando seu conhecimento para que eu pudesse desenvolver esse trabalho.

Agradeço aos professores Roberto H. Tsunaki e Emerson C. Pedrino, meu amigo Fernando C. Sato e todos os envolvidos que auxiliaram no desenvolvimento de trabalhos realizados durante essa pesquisa e que consequentemente também contribuíram com essa pesquisa.

Agradeço ao meu amigo Vinícius Caridá que me auxiliou como um tutor durante o mestrado.

Agradeço ao meu amigo e funcionário do departamento de computação Augusto C. H. Pinha que realizou diversos “quebra galhos” durante minha permanência no mestrado.

Agradeço aos meus amigos Murilo P. de Oliveira, Guilherme de M. Barsoti, Rafael V. B. Garcia, Rodolfo Jacinto, Celso A. Takahashi, Hugo M. Bolognesi, César A. da Silva, Geraldo A. S. Martins, Everaldo Gonçalves, Rodrigo Scattone, Bruno Trefilio, Lucas F. de Castro, Lucas B. Branisso, André Bevilaqua, Fabrício A. Rodrigues e todos os amigos que me apoiaram e auxiliaram de alguma forma no desenvolvimento dessa pesquisa.

Agradeço a CAPES pela concessão da bolsa de estudos.

Por fim agradeço a todos os funcionários, professores e amigos do departamento de computação e demais envolvidos que contribuíram de alguma forma para o desenvolvimento desse trabalho.

RESUMO

Servovisão é uma técnica que utiliza visão computacional para obter informações visuais (através de câmera) e um sistema de controle com circuito em malha fechada para controlar robôs. Uma das aplicações típicas de servovisão é no rastreamento de objetos sobre esteiras transportadoras em ambientes industriais. Servovisão possui a vantagem em relação a outros tipos de sensores de permitir a obtenção de um grande número de informações a partir do ambiente e maior flexibilidade nas operações. Uma desvantagem são os atrasos conhecidos como *dead-times* ou *time-delays* que podem ocorrer durante o tratamento de informações visuais nas tarefas de visão computacional ou em outras tarefas do sistema de controle que necessitam de grande capacidade de processamento. Os *dead-times* em servovisão aplicada em operações industriais como no rastreamento de objetos em esteiras transportadoras são críticos e podem afetar negativamente na capacidade de produção em ambientes de manufatura. Algumas metodologias podem ser encontradas na literatura para esse tipo de problema sendo muitas vezes baseadas no filtro de Kalman. Nesse trabalho foi selecionada uma metodologia baseada na formulação do filtro de Kalman que já possuí um estudo na previsão futura de pose de objetos com movimentação linear. Essa metodologia foi estudada detalhadamente, testada através de simulações e analisada sobre outros tipos de movimentos e algumas aplicações. No total foram gerados três tipos de experimentos: um para diferentes tipos de movimentação e outros dois aplicados em diferentes tipos de sinais no controlador de velocidades. Os resultados a partir da movimentação do objeto demonstraram que o método é capaz de estimar a pose futura de objetos com movimento linear e com curvas suaves, porém é ineficiente para alterações drásticas no movimento. Com relação ao sinal a ser filtrado no controlador de velocidades a metodologia se demonstrou aplicável (com as condições de movimento) somente na estimativa da pose do objeto após a ocorrência de *dead-times* causados por visão computacional e posteriormente essa informação é utilizada para calcular o erro futuro do objeto em relação ao manipulador robótico utilizado no cálculo da velocidade do robô. A tentativa de aplicação da técnica diretamente no erro utilizado no cálculo da velocidade a ser aplicada ao robô não apresentou bons resultados. Com os resultados obtidos a metodologia se demonstrou eficiente para o rastreamento de objetos de forma linear e curvas suaves como no caso de objetos transportados por esteiras em ambientes industriais.

Palavras-chave: Servovisão, filtro de Kalman, *dead-times*, *time-delays,* rastreamento de objetos, estimativa de pose, controlador de velocidades.

ABSTRACT

Visual servoing is a technique that uses computer vision to acquire visual information (by camera) and a control system with closed loop circuit to control robots. One typical application of visual servoing is tracking objects on conveyors in industrial environments. Visual servoing has the advantage of obtaining a large amount of information from the environment and greater flexibility in operations than other types of sensors. A disadvantage are the delays, known as dead-times or time-delays that can occur during the treatment of visual information in computer vision tasks or other tasks of the control system that need large processing capacity. The dead-times in visual servoing applied in industrial operations such as in the tracking of objects on conveyors are critical and can negatively affect production capacity in manufacturing environments. Some methodologies can be found in the literature for this problem and some of these methodologies are often based on the Kalman filter. In this work a technique was selected based on the formulation of the Kalman filter that already had a study on the prediction of future pose of objects with linear motion. This methodology has been studied in detail, tested and analyzed through simulations for other motions and some applications. Three types of experiments were generated: one for different types of motions and two others applied in different types of signals in the velocity control systems. The results from the motion of the object shown that the technique is able to estimate the future pose of objects with linear motion and smooth curves, but it is inefficient for drastic changes in motion. With respect to the signal to be filtered in the velocity control, the methodogy has been shown applicable (with motions conditions) only in the estimation of pose of the object after the occurrence of dead-times caused by computer vision and this information is subsequently used to calculate the future error of the object related to the robotic manipulator used to calculate the velocity of the robot. The trying to apply the methodogy directly on the error used to calculate the velocity to be applied to the robot did not produce good results. With the results the methodogy can be applied for object tracking with linear motion and smooth curves as in the case of objects transported by conveyors in industrial environments.

Keywords: Visual servoing, Kalman filter, dead-times, time-delays, object tracking, pose estimation, velocity control.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

PBVS – Servo-controle baseado em posição (do inglês "*Position-Based Visual Servo"*)

IBVS – Servo-controle baseado em imagem (do inglês "*Image-Based Visual Servo"*)

2D – Duas dimensões

3D – Três dimensões

AGVs – veículos guiados automaticamente (do inglês *"automated guided vehicles"*)

DSP – (do inglês *Digital Signal Processor*)

FPGAs – (do inglês *Field-programmable gate array*)

TDD – Transferência de dados

PDD – Processamento de dados

EKF – filtro de Kalman estendido (do inglês *extended Kalman Filter*)

AKF – filtro de Kalman adaptativo (do inglês *adaptive Kalman Filter*)

SUMÁRIO

Capítulo 1

INTRODUÇÃO

Robôs são utilizados em diversos ambientes onde normalmente executam tarefas repetitivas que necessitam de precisão ou que apresentam riscos para os seres humanos. Em ambientes industriais automatizados existem robôs que executam tarefas como deslocamento, montagem e classificação de objetos tais como produtos, componentes de produtos e matérias primas. O uso de robôs industriais vem aumentando no decorrer dos anos, isso vem acontecendo devido à redução de custos em sua produção e aumento de sua eficiência, ambos os fatores influenciam diretamente na economia de muitas empresas (CRAIG, 2013).

Um manipulador robótico possibilita a execução de diversas tarefas como deslocamento de materiais, montagem, soldagem, pintura, entre outros. Para a realização dessas tarefas o robô precisa localizar o objeto que pode estar em movimento ou não, e para isso são necessárias técnicas de rastreamento. De acordo com Corke (2011) robôs também são muito utilizados em operações médicas, navegação, entretenimento, entre outras. Porém nesse trabalho a pesquisa possui ênfase em robôs industriais e mais especificamente em manipuladores robóticos industriais.

Para robôs convencionais, a precisão é determinada a partir da retroalimentação (do inglês "*feedback")* para o controlador de articulações/juntas. Alguns problemas como o desgaste das juntas do robô ou mudanças no ambiente em que o robô interage podem afetar na precisão de suas operações (LI, 2007).

Uma das preocupações na utilização de robôs em ambientes industriais é proporcionar a capacidade de flexibilidade a esses robôs, não sendo necessário um ambiente totalmente estruturado (condicionado e com limitações) para que esses robôs possam realizar suas tarefas (SICILIANO, et al., 2009). Outra dificuldade

encontrada é com a configuração e programação de um robô onde incertezas relacionadas com a movimentação que esse robô deve efetuar durante a execução de uma determinada tarefa necessitam ser previstas (MASON, 2012).

Uma tecnologia que vem sendo explorada para contornar essa necessidade é a utilização de câmeras (operando como sensores de visão) que é capaz de obter uma quantidade maior de informações do ambiente em relação á outros sensores. Assim como descrito por Hutchinson, Hager e Corke (1996) as câmeras além de serem capazes de identificar grandes quantidades de informações, permitem a medição sem o contato do ambiente, e após a aquisição de informações transferem essas informações para serem processadas por técnicas de visão computacional.

A utilização de câmeras em conjunto a um circuito de malha fechada proporcionam á capacidade de visão e atualização através da retroalimentação constante das informações aos controladores de robôs. Essa técnica é denominada servovisão e permite aos robôs maior flexibilidade e precisão para as na realização de suas operações (CORKE, 1996). A flexibilidade é dada pela capacidade de visão ao robô, ou seja, pela possibilidade de adquirir e processar grande quantidade de informações. A precisão é influenciada pela atualização constante das informações visuais (através de retroalimentação visual).

Uma aplicação comum de servovisão é o rastreamento de objetos, como por exemplo, quando um órgão terminal/ferramenta robótica (manipulador robótico) deve ser posicionado corretamente em relação a um objeto e as informações visuais são utilizadas no retroalimentação do sistema. As informações visuais seguidas de um devido processamento de imagem permitem identificar corretamente o objeto de interesse. Caso o objeto esteja em movimento, através da atualização do sistema e algoritmos específicos, o robô consegue se posicionar corretamente (KHO e KWON, 2012).

Com a evolução de tecnologias e técnicas para processamento de imagem e visão computacional á servovisão se torna mais atrativa e utilizada em ambientes industriais. Uma vantagem é a questão de flexibilidade, já que robôs são cada vez mais utilizados em ambientes industriais e necessitam interagir com outros robôs e objetos (que são obstáculos) no ambiente.

Além das vantagens citadas existem também diversos problemas podem ser encontrados em servovisão. Um problema bastante comum está relacionado a atrasos de processamento de dados (ou simplesmente atrasos, encontrados em

inglês *"time-delays"* ou "*dead-times*") na execução do ciclo do sistema de controle conforme demonstrado na Figura 1.1.

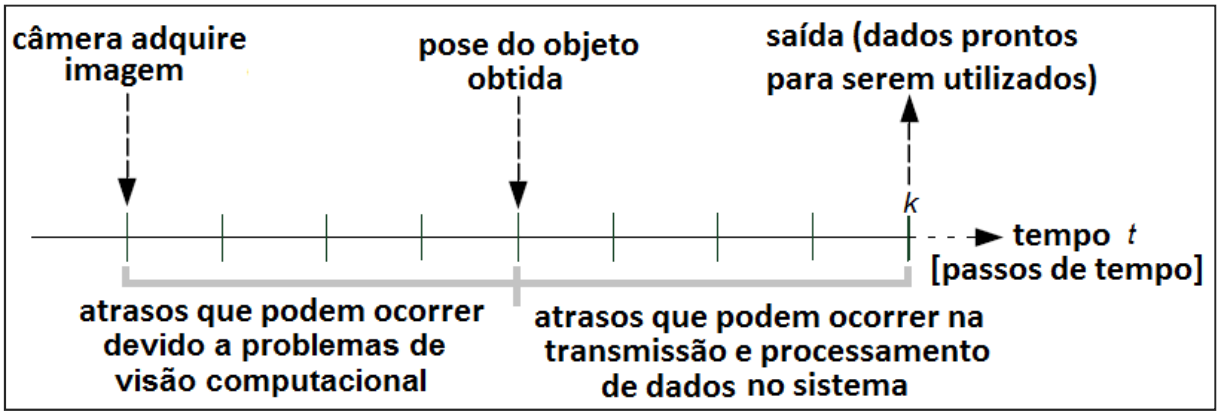


Figura 1.1 – Demonstração *dead-times* em operações com servovisão.

Os principais motivos que geram *dead-times* são:

- Processamento de imagens durante a etapa de extração de características que representam informações da pose (pose é igual a posição e a orientação) do objeto a ser rastreado;
- Na taxa de amostragem da câmera, ou seja, do tempo em que a imagem é adquirida até estar disponível para ser utilizada;
- Transmissão de dados entre os módulos do sistema, como a partir da pose obtida transmitir os dados para o sistema de controle;
- Processamento de dados (além do processamento de imagem) do sistema, que consistem desde o processamento da lei de controle do sistema até ao processamento de algoritmos auxiliares como preditores de posição futura de objetos em movimento a ser rastreados.

Os dois primeiros problemas são problemas de visão computacional e normalmente acontecem antes da pose do objeto ser obtida enquanto que os outros dois problemas são após a pose do objeto estar obtida.

Um sistema de controle com servovisão, quando não ocorrem *dead-times* possui um tempo pouco ou não variável em cada amostra (tempo que o sistema leva desde a aquisição da imagem até alimentar a informação da lei de controle no robô). Isso faz com que a taxa de amostragem seja um valor aproximadamente fixo e

permita um melhor desenvolvimento e controle no sistema de controle, contribuindo posteriormente para o desempenho desse sistema. Oscilações no tempo da amostra podem ser causadas por *dead-times* e isso faz com que a cada tempo em uma amostra do sistema de controle varie significativamente em relação a taxa de amostragem implicando em um sistema de controle com desempenho ruim (CORKE, 1996).

Em operações industriais objetos são comuns de serem movimentados por esteiras transportadoras e a ocorrência de *dead-times* nas operações de rastreamento podem causar graves danos a produção (DENKER, SABANOVIC e KAYNAK, 1994).

Os métodos para solução de *dead-times* de uma forma generalizada podem ser classificados em métodos que tentam evitar a ocorrência de *dead-times* e métodos que contornam os *dead-times* quando esses ocorrem. Os métodos que tentam evitar a ocorrência de *dead-times* normalmente são executados por dispositivos (*hardwares*) com alta capacidade de processamento, computação distribuída, processamento *pipeline* (busca instruções além da próxima instrução a ser executada), entre outras. Os métodos que tentam contornar a ocorrência de *dead-times* são normalmente utilizados na forma de compensadores, como por exemplo, no caso do rastreamento de um objeto em movimento, na ocorrência de *dead-times,* o método pode compensar a estimativa da pose do objeto algumas etapas a frente da situação atual.

Considerando o problema de *dead-times* e a importância de operações de rastreamento de objetos em movimento executadas por manipuladores robóticos com servovisão esse trabalho apresenta um estudo e avaliação de um método de compensação para *dead-times* em servovisão. Como previamente descrito o trabalho tem ênfase em aplicações de rastreamento em ambientes industriais, principalmente em esteiras transportadoras, onde esse problema é crítico para produção industrial.

1.1 Justificativa e Motivação

A automatização em ambientes industriais é uma característica crescente nos últimos anos e manipuladores robóticos são comuns e essenciais nesses ambientes. A utilização de robôs, ou seja, de manipuladores robóticos proporcionam menores custos e maior rendimento no processo produção influenciando positivamente na economia das empresas.

Servovisão proporciona diversos benefícios a operações executadas por manipuladores robóticos principalmente relacionados ao rastreamento de objetos em movimento. Além dos benefícios traz consigo alguns problemas como o problema de *dead-times*, um fator crítico em diversos ambientes.

Durante o desenvolvimento dessa pesquisa foram encontrados diversos trabalhos em diferentes épocas tratando desse problema. Como o problema com *dead-times* existe desde o surgimento de servovisão e crítico principalmente para ambientes industriais, pois *dead-times* podem comprometer operações robóticas e gerar problemas na linha de produção, diferentes abordagens foram propostas, porém ainda existe uma busca por soluções adequadas.

Essa necessidade de resolver o problema de *dead-times* e a importância de tarefas de rastreamento com servovisão em ambientes industriais motivaram o desenvolvimento dessa pesquisa e o estudo do método apresentado nesse trabalho. Nessa pesquisa a abordagem utilizada para problemas de *dead-times* se enquadra em métodos compensadores. A escolha foi realizada devido ao fato de que métodos para evitar a ocorrência de *dead-times* podem não ser totalmente eficientes já que estão sujeitos a ocorrência de *dead-times* em algum momento.

1.2 Objetivos

O objetivo geral desse trabalho é estabelecer um método para tratamento de *dead-times* e realizar um estudo, análise e aplicação desse método em diferentes condições. O método deve principalmente poder ser aplicado sobre o rastreamento de objetos em movimento sobre esteiras transportadoras utilizando servovisão.

Os objetivos específicos são:

- implementação de um método para tratar *dead-times* no rastreamento de objetos por servovisão que possa ser posteriormente adaptado para esteiras transportadoras;
- análise do método selecionado sobre a possibilidade da aplicação tratar *dead-times* no rastreamento de objetos por servovisão no âmbito geral;
- desenvolvimento de simuladores para avaliar o método proposto;
- aplicação do método proposto em um modelo baseado em sistemas de controle por servovisão.

1.3 Estrutura do Trabalho

- O capítulo 1 apresentou uma introdução e visão geral do problema abordado no trabalho.
- O capítulo 2 apresenta uma fundamentação teórica do trabalho com o propósito de contribuir com uma revisão dos principais tópicos abordados e com leitores que não estão muito familiarizados com o tema desse trabalho.
- O capítulo 3 apresenta as referencias bibliográficas relacionadas com o problema e as soluções aplicadas.
- O capítulo 4 apresenta a proposta e um estudo do método utilizado.
- O capítulo 5 apresenta questões do, desenvolvimento do simulador e simulador utilizado e a aplicação do método proposto nesse simulador.
- O capítulo 6 apresenta os experimentos realizados e os resultados.
- O capítulo 7 apresenta a conclusão e os trabalhos futuros.
- As nomenclaturas e notações utilizadas nesse trabalho são baseadas nos trabalhos clássicos que podem ser encontrados na literatura como Corke (2011), Hutchinson, Hager e Corke (1996), Chaumette e Hutchinson (2006) e Chaumette e Hutchinson (2007) e Craig (2013).

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta uma fundamentação teórica para auxiliar no entendimento do tema tratado nesse trabalho. São abordados os conceitos de robôs e manipuladores robóticos destacando alguns dos principais componentes envolvidos com a área de servovisão. É abordado um tópico descrevendo os principais conceitos de servovisão, visão computacional em servovisão, classificações de servovisão, simuladores para servovisão, operações de rastreamento através de servovisão. Por fim um tópico de filtro de Kalman e aplicações com filtro de Kalman em servovisão é descrito.

2.1 Robôs e Manipuladores robóticos

Existem diversas formas de definições de robôs, porém baseado em alguns trabalhos de autores clássicos como Craig (2013), Corke (2011) e Siciliano, et al. (2009) para a presente pesquisa, robôs são máquinas desenvolvidas para executar tarefas repetitivamente com velocidade e precisão ou que apresentam algum risco aos seres humanos. Uma definição de robôs para área industrial, que é de interesse nesse trabalho, é a de Crawford (2011) onde são definidos como um manipulador multiuso, controlado automaticamente, programável e reprogramável.

A utilização de robôs industriais tem aumentado significativamente nas ultimas décadas, um breve estudo sobre esse aumento pode ser observado em Craig (2013). Ainda em Craig (2013) é descrito que esse aumento se deve á redução

de custo na produção de robôs e melhoria de sua eficiência, ambos os fatores influenciam diretamente na economia de muitas empresas.

Aplicações típicas dos robôs incluem soldagem, pintura, montagem, manipulação de objetos, transporte, embalagem e paletização, inspeção de produtos e testes, navegação e entretenimento, além de atualmente também existirem pesquisas com robôs para tarefas como entrega de encomendas e cuidar de idosos e deficientes físicos.

Siciliano, et al. (2009) categorizam robôs de acordo com atributos de mobilidade como fixos ou móveis, sendo que os móveis podem ser movimentados por rodas ou pernas robóticas e também que podem ser robôs de rastreamento movimentados por juntas. Outra classificação demonstrada por Corke (2011) descreve as categorias de robôs móveis como robôs capazes de locomover pelo ambiente e manipuladores robóticos.

Utilizando o conceito de Corke (2011), robôs móveis podem ser utilizados em tarefas de transporte de peças em ambientes industriais ou na exploração de ambientes desconhecidos. Pesquisas na área de robôs móveis envolvem problemas de navegação que é o problema de guiar o robô para um objetivo e de autolocalização que envolve o estudo de fazer com que o robô saiba sua própria localização em relação a pontos referênciais.

Os manipuladores robóticos são frequentemente utilizados em indústrias em tarefas repetitivas e que necessitam de precisão. Executam tarefas como pintura, soldagem, deslocamento de materiais, entre outras (CRAIG, 2013).

Muitas vezes os manipuladores robóticos industriais ficam fixos e operam com objetos em esteiras transportadoras. De acordo com Mason (2012) as esteiras transportadoras são consideradas um exemplo de ambiente estruturado e facilitam diversas tarefas executadas por um robô. Esteiras transportadoras são dispositivos mecânicos normalmente movimentados por motores e utilizados para mover itens ou materiais em grande quantidades, sendo normalmente são encontradas em ambientes internos como indústrias (GROOVER, 2013).

Mason (2012) também descreve que ambientes não estruturados são aqueles que não foram preparados para o uso de robôs. No caso de ambientes industriais que necessitam de flexibilidade (normalmente os não estruturados) o uso de veículos guiados automaticamente (AGVS) (do inglês *“automated guided vehicles*

system") é comum. OS AGVS utilizados em ambientes industriais podem possuir manipuladores robóticos acoplados nos veículos (CRAWFORD, 2011).

Um exemplo de manipulador robótico que será utilizado nesse trabalho é o AFMA-6 representado na Figura 2.1. O AFMA-6 é um manipulador robótico industrial com estrutura cartesiana cuja primeira versão foi construída em 1992 pela empresa francesa *AFMA-Robots Company.* O AFMA-6 possui 6 graus de liberdade e pode ser adaptado para operar em conjunto com sistema de servovisão (LAGADIC, 2013).

Figura 2.1 – Robô manipulador AFMA-6 (LAGADIC, 2013).

Para executar suas tarefas, os manipuladores robóticos necessitam interagir com objetos (matérias primas, produtos, etc). Para interação com um objeto existe a necessidade de ter conhecimento da pose (posição e orientação) atual do manipulador, identificar a localização/pose do objeto, encontrar uma relação entre o manipulador robótico e o objeto (denotada por ${}^{C}\xi_{T}$, essa relação se dá através do sistema de coordenadas do manipulador e do objeto e é demonstrada na Figura 2.2), ter conhecimento de como a pose atual do manipulador está formada/configurada através da posição de cada junta do manipulador robótico e determinar a melhor posição dessas juntas para que o manipulador robótico consiga alcançar o objeto a ser manipulado (CRAIG, 2013). Um estudo completo sobre os

sistemas de coordenadas e relações entre poses de objetos e robôs podem ser encontrados em Craig (2013).

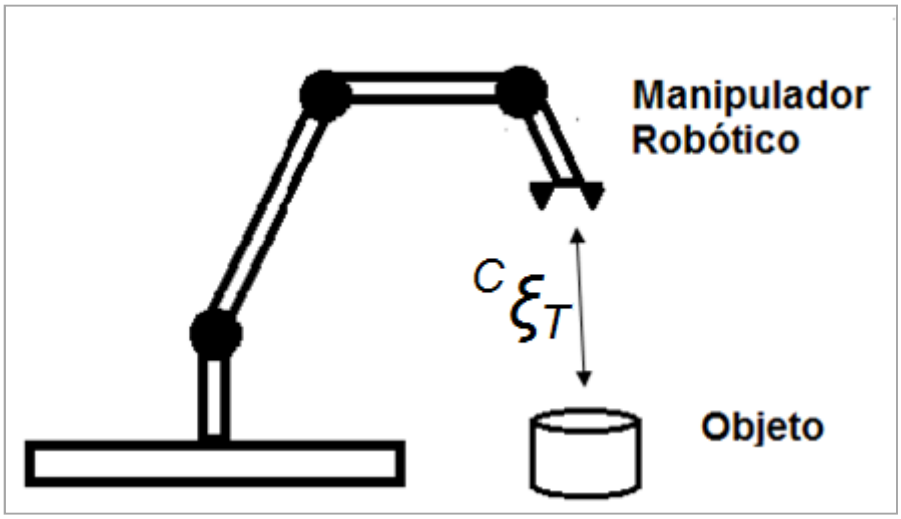


Figura 2.2 – Relação de um manipulador robótico e um objeto.

Para que o manipulador robótico consiga identificar os objetos são necessários vários tipos de componentes e a seguir são apresentados os principais componentes para o contexto abordado neste trabalho.

2.1.1 Componentes de manipuladores robóticos.

Manipuladores robóticos são compostos por elos, juntas e órgão terminal/efetuador conforme demonstrado na Figura 2.3. Usualmente uma extremidade do manipulador robótico é fixa a uma base e a outra extremidade é livre para se movimentar no espaço de trabalho (CRAIG, 2013).

Os graus de liberdade determinam a movimentação dos manipuladores robóticos e são estabelecidos de acordo com as juntas dos robôs. Os órgãos terminais podem ser ganchos, dedos, ventosas de sucção, imãs, ferramentas de solda, dispositivos para furação ou pintura, entre outras. O movimento das juntas dos robôs é realizados através de acionadores, que podem ser elétricos, pneumáticos ou hidráulicos (GROOVER, 1988 apud CARRARA, 2013).

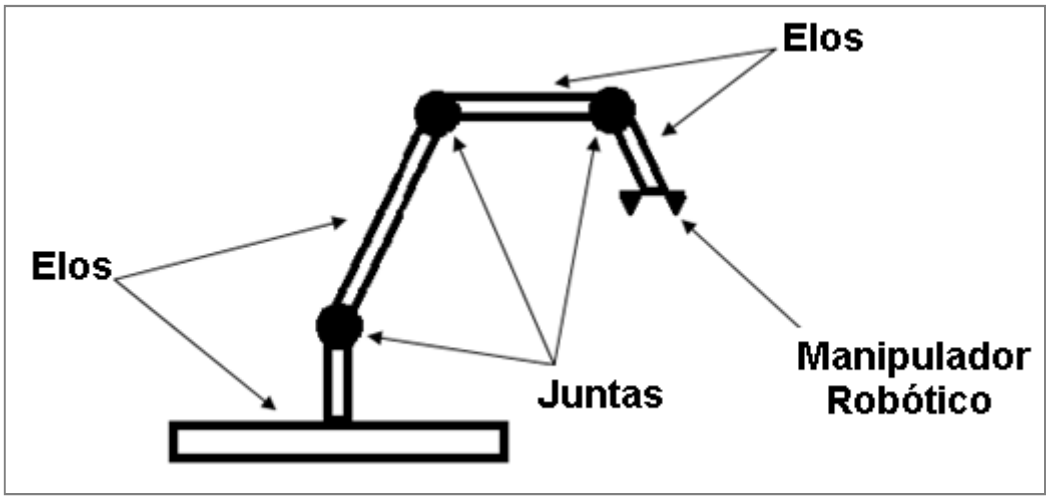


Figura 2.3 – Composição de manipuladores robóticos.

Para interagir com o ambiente os manipuladores robóticos fazem o uso de sensores. Sensores são responsáveis por proporcionar a interatividade entre robôs e o meio ambiente (WANG, HUANG e SHENG, 2007). Existem diversos tipos de sensores, como sensores de tato, sensores de proximidade, sensores de distância e em especial os sensores de visão (GROOVER, 1988 apud CARRARA, 2013).

Os sensores de visão são utilizados na forma de câmeras e possuem a vantagem de medição a distância e são capazes de obter um grande número de informações do ambiente. Com a informação obtida através dos sensores, o robô deve executar alguma ação através de uma regra e para isso são utilizados os sistemas de controle (CORKE, 2011).

Os sistemas de controle tem a função de processar os sinais obtidos pelos sensores e utiliza-los nos robôs. Sistemas de controle podem ser constituídos de *hardware* e/ou *software*. A regra utilizada nos sistemas de controle é normalmente denominada lei de controle.

Um sistema de controle ainda pode ter características de circuito de malha aberta ou circuito de malha fechada, mas normalmente em robôs são utilizados sistemas de circuito de malha fechada.

Os sistemas de controle com circuito em malha fechada são diferentes dos sistemas de circuito de malha aberta por possuírem um sistema de retroalimentação responsável por atualizar as informações no sistema de controle (OGATA, 2003).

O conceito de taxa de amostragem do sinal do controle para robô também é importante para esse trabalho. Uma amostra do ciclo do sistema de controle consiste

do momento em que a informação é obtida pelo sensor, passando pelo cálculo do erro, até a informação poder ser utilizada no robô, supondo que esse ciclo seja executado 1 vez a cada 1 segundo a taxa de amostragem é de 1Hz (Hertz).

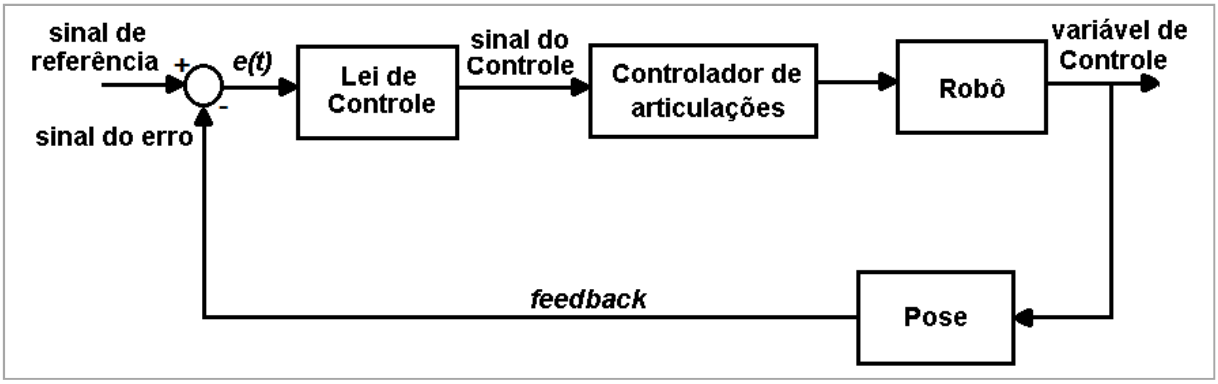


Figura 2.4 – Diagrama de blocos - sistema de controle de malha fechada.

A Figura 2.4 representa um sistema de controle em malha fechada e também é considerado o uso de um sensor qualquer capaz de identificar a pose do objeto. Na Figura 2.4 é possível observar a existência de um retroalimentação (*feedback*) levando informações para que seja estabelecido o erro no sistema de controle.

Por exemplo, considerando a Figura 2.4 e uma operação de rastreamento de objetos por manipuladores robóticos a pose atual do manipulador robótico em relação ao objeto é descrita na Figura 2.4 como *sinal do erro* e a pose desejada do manipulador robótico em relação ao mesmo objeto é descrita na Figura 2.4 como *sinal de referência*, também denotada em inglês como *set-point*. O erro calculado (na Figura 2.4 descrito como *e(t)*) é utilizado em uma lei de controle, que pode ter como objetivo determinar a configuração necessária das juntas/articulações do robô fornecida pelo *sinal de referência* para que o robô consiga se movimentar até a pose desejada em relação ao objeto. O sinal do controle estabelecido pela lei de controle é utilizado no controlador de articulações que por sua vez ajusta as articulações do robô para as configurações desejadas sendo estabelecido o novo valor da variável de controle (pose do robô em relação ao objeto). O sensor verifica a nova pose do robô em relação ao objeto e passa essa informação através da retroalimentação para o sistema de controle que formula novo erro e reinicia o ciclo do sistema.

2.2 Servovisão

Nos primeiros métodos de controle de robô por imagens, os sistemas eram na forma de malha aberta e possuíam a característica de olhar e então movimentar-se (do inglês "*look-then-moving").* Nos sistemas *look-then-moving* a precisão depende diretamente do único conjunto de informações obtidos pelo sensor visual e da qualidade da estrutura do manipulador robótico se tornando um sistema sensível a incertezas. Por exemplo, ao executar uma tarefa se a precisão de posicionamento do manipulador é ruim ou se o objeto é movido durante o movimento do manipulador robótico a tarefa pode não ser executada corretamente (SICILIANO, et al., 2009).

A utilização de um sistema de malha fechada para essa tarefa e uma retroalimentação visual, permitem atualizações constantes da cena (a cada unidade do ciclo do sistema). Dessa forma as medidas visuais são introduzidas de volta ao sistema de controle, para que seja calculado o erro apropriado para cada determinado instante. O erro calculado é utilizado no controlador do sistema e normalmente o objetivo do sistema é reduzir ao máximo possível esse erro. O erro normalmente é composto pela diferença entre a pose atual (pose é igual a posição e a orientação) do objeto em relação ao manipulador robótico e a pose desejada do mesmo objeto em relação a esse mesmo manipulador. O cálculo do erro é representado pela equação 1.

$$e = p - p^* \tag{1}$$

e: erro, p e p^*: são respectivamente a pose atual e desejada do objeto em relação ao manipulador robótico.

A Figura 2.5 demonstra exemplos sobre a simulação do manipulador robótico AFMA-6 que é utilizado nesse trabalho e detalhado posteriormente. Na Figura 2.5 ainda são considerados que o manipulador apenas se movimenta apenas com relação a posição nos eixos x e y, sendo z e a orientação constantes.

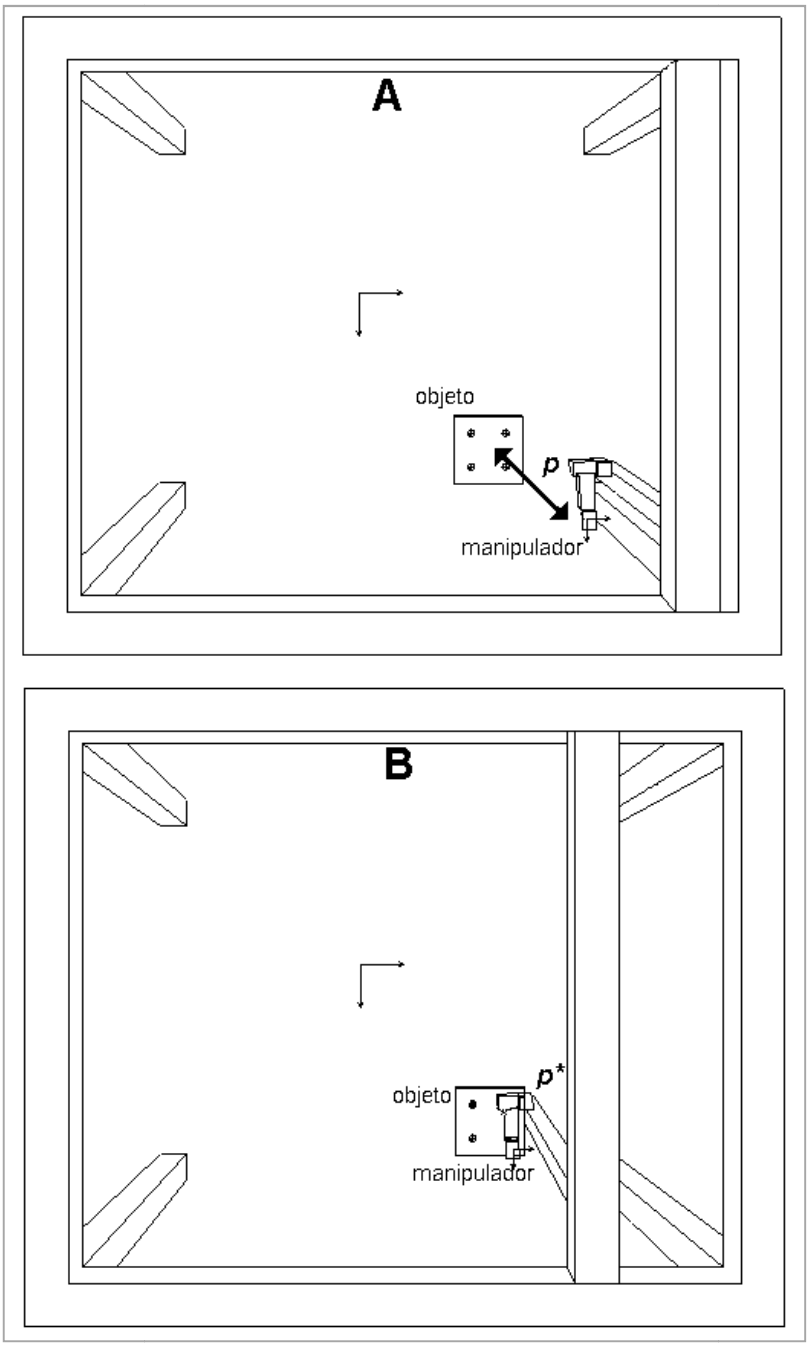


Figura 2.5 – Pose atual (parte A) e desejada (parte B) do manipulador robótico (AFMA-6) em relação ao objeto.

Na parte A da Figura 2.5 é demonstrado a variável *p*, ou seja, a pose atual do objeto em relação ao manipulador robótico. Na parte B da Figura 2.5 é demonstrado a variável *p**, ou seja, a pose desejada do objeto em relação ao manipulador robótico. Como demonstrado na equação 1 essas duas variáveis são utilizadas no cálculo do erro *e* que é utilizado no sistema de controle. Como a informação é obtida através de câmeras, a variável *p* é extraída ou utilizada a partir de informações

visuais das imagens obtidas pela câmera. Essa característica de retroalimentação a partir de informações visuais no sistema é conhecida como servovisão (ou controle servo visual) (HUTCHINSON, HAGER e CORKE, 1996).

Em trabalhos encontrados na literatura, um exemplo de definição de servovisão é a de Corke (2011) que a descreve como a tarefa de o posicionamento do robô em relação a um alvo a partir de características visuais extraídas de uma imagem. Outra definição é de Siciliano, et al. (2009) que definem servovisão como o controle baseado na retroalimentação de medidas visuais.

No sistema de controle a retroalimentação utiliza informações obtidas a partir de uma imagem, sendo esse sistema de controle responsável por uma ação de um robô. Servovisão envolve o estudo de diferentes áreas como cinemática, dinâmica, teoria de controle, e de interesse desse trabalho ás áreas de processamento de imagem, visão computacional e computação em tempo real (CORKE, 2011).

A servovisão pode ser encontrada em operações de localização e rastreamento de objetos como demonstrado nos trabalhos de Denker, Sabanovic e Kaynak (1994), Hong, et al. (2001), Kho e Kwon (2012).

Denker, Sabanovic e Kaynak (1994) possuem suas aplicações e testes em esteiras transportadoras, enquanto Hong, et al. (2001) utilizam um sistema de rastreamento em um veículo autônomo capaz de perseguir um objeto circular através de servovisão. Liu, Sun e Fujii (2010) utilizam servovisão na tarefa de um manipulador robótico selecionar entre peças cilíndricas em uma bandeja. Mondrágon (2010) apresenta um trabalho desenvolvido para veículos autônomos aéreos não tripulados onde utiliza visão planar e servovisão para operações nos veículos. Outras aplicações incluem operações de montagem, em equipamentos bélicos, em equipamentos subaquáticos, além de outras inúmeras aplicações (CORKE, 2011).

2.2.1 Visão computacional e utilização de câmeras como sensores

Um importante campo dentro da servovisão é a visão computacional a qual é responsável por obter informações do ambiente e interpretar essas informações de acordo com a aplicação que será utilizada (SICILIANO, et al., 2009).

Algumas subáreas da visão computacional também são importantes como o estudo de projeções (como a projeção perspectiva) para a interpretação/conversão do mundo real (3 dimensões - 3D) para o plano da imagem (2 dimensões - 2D) e

vice-versa. Algoritmos e técnicas para localização do objeto na imagem como morfologia matemática também são de grande importância para sistemas de servovisão (SAQUI, et al., 2013).

Nos sistemas de servovisão uma ou mais câmeras também podem ser utilizadas. Essas câmeras podem estar localizadas na mão do manipulador robótico (*"eye-in-hand"*) ou localizadas em algum local fixo no ambiente (*"eye-to-hand")* conforme demonstrado na Figura 2.6.

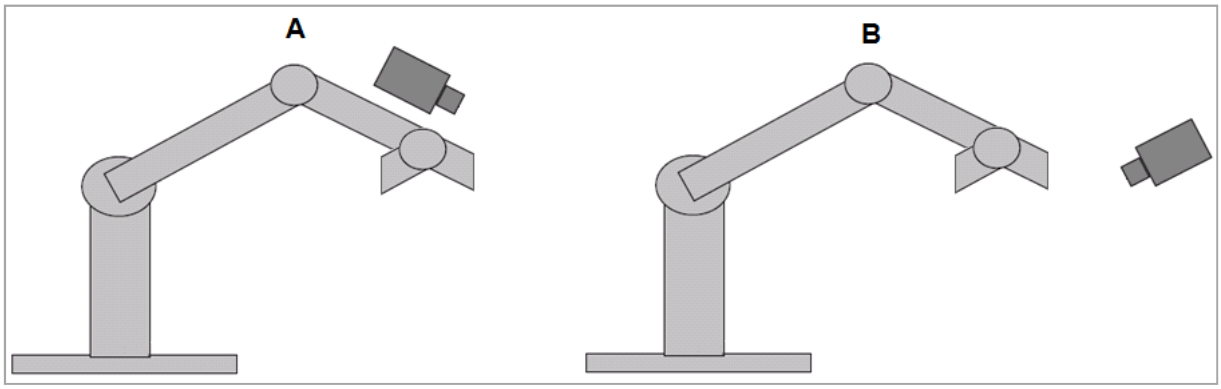


Figura 2.6 – Localização da câmera em servovisão *eye-in-hand* (parte A) e *eye-to-hand* (parte B) (adaptado de Corke (2011)).

Outra configuração da câmera está relacionada com parâmetros intrínsecos (configurações internas) como distância focal, tamanho do pixel, etc e parâmetros extrínsecos (configurações externas) como a localização das câmeras. Esses parâmetros normalmente são obtidos através de um processo denominado calibração da câmera (CORKE, 2011).

2.2.2 Classificações de Servovisão

Sanderson e Weiss (1980) apud Hutchinson, Hager e Corke (1996) introduziram uma taxonomia onde todos os sistemas de servovisão podem ser categorizados.

Uma das classificações trata da arquitetura do controle, tendo duas categorias de classificação, a servovisão dinâmica e a servovisão direta:

(a) A servovisão dinâmica possui uma estrutura hierárquica de controle, onde o servocontrole de pose e a servovisão atuam em conjunto, ou seja, o sistema de servovisão recebe a informação (através da retroalimentação) referente a pose

atual do manipulador em relação ao objeto, calcula o erro e atua no controlador de servovisão que por sua vez atua no sistema de servocontrole fornecendo os sinais de referência *(set-points)* necessários para a atuação nos servomotores.

(b) A servovisão direta é quando o controlador do robô é substituído inteiramente por um servo-controlador visual, sendo que este computa diretamente as entradas das articulações e utiliza somente a visão para estabilizar o mecanismo.

Por vários motivos, a maioria dos sistemas implementados adotam a abordagem de servovisão dinâmica embora a servovisão direta também é utilizada frequentemente em pesquisas com desempenho da taxa de amostragem do sistema (HUTCHINSON, HAGER e CORKE, 1996).

Outra classificação trata de como o sinal de erro está definido com base no espaço de tarefas real do robô (tridimensional) ou diretamente no plano da imagem (bidimensional). Essa classificação gerou dois tipos de sistemas de servovisão:

(a) A servovisão baseada em pose (PBVS) (do inglês "*Pose-Based Visual Servo"*) representada na Figura 2.7 onde é utilizado no espaço de tarefas real do robô (tridimensional). No modelo PBVS as coordenadas são extraídas da imagem então são convertidas para unidades de medidas relacionadas com esse espaço de tarefas. Outra questão referente ao PBVS é a necessidade da utilização de um modelo que represente o objeto a ser rastreado ou manipulado. Esse modelo do objeto deve representar a forma geométrica do objeto ou imagens demonstrando como a câmera deve visualizar o objeto para executar suas tarefas. São também necessárias algumas informações da câmera como, por exemplo, os parâmetros intrínsecos (como distância focal e tamanho dos pixels) e os parâmetros extrínsecos (como orientação e posição da câmera em relação a um sistema de referência no mundo). Os parâmetros intrínsecos e extrínsecos da câmera são obtidos com um processo de calibração da câmera sendo essa etapa muito importante para o sistema de servovisão PBVS (MALIS, 2002).

Na Figura 2.7 considerando uma operação de rastreamento de um objeto com servovisão PBVS, após a câmera adquirir a imagem é realizada uma operação de extração de características. Na etapa de extração de características são extraídas as informações f (em pixels) que representam a pose do objeto. Essas informações são convertidas para unidades de medidas relacionadas com o espaço de tarefas do robô, e utilizadas para representar a pose atual do robô em relação ao alvo como

${}^{C}\xi_{T}$. Com a pose desejada ${}^{C}\xi_{T}^{*}$ e atual ${}^{C}\xi_{T}$ é calculado o erro (diferença entre os valores) para alimentar o sistema.

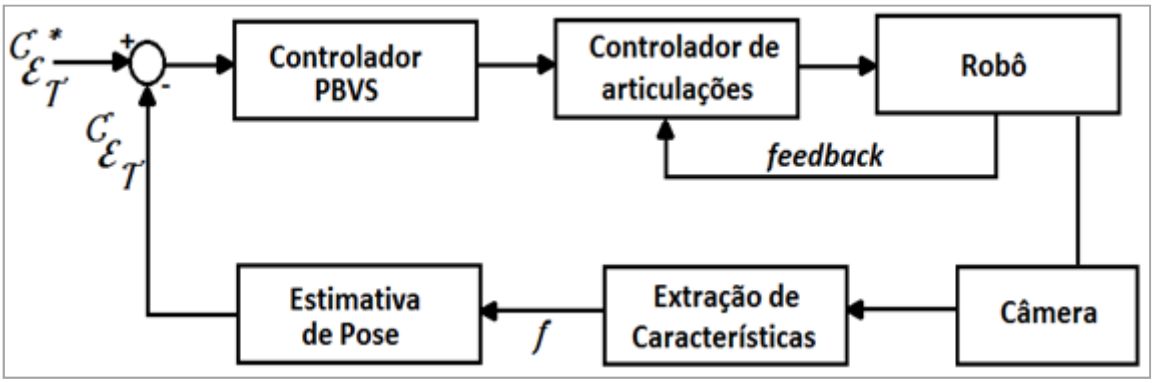


Figura 2.7 – Diagrama do modelo PBVS (adaptado de Corke (2011)).

(b) Na servovisão baseado em imagem (IBVS) (do inglês *Image-Based Visual Servo*) representada na Figura 2.8 o sinal de erro é computado diretamente com base na diferença entre os valores das características da imagem atuais e os valores desejados. A determinação de uma função de erro é feita diretamente por equações. Se a tarefa é definida em com base em um objeto em movimento, o erro não será apenas uma função da diferença entre a pose desejada e atual do manipulador em relação ao objeto, mas também da variação desse erro no tempo. Em relação ao modelo PBVS, IBVS é sujeito a menos erros devido a utilização de uma calibração de câmera menos complexa (CORKE, 2011).

Na Figura 2.8 considerando uma operação de rastreamento de um objeto, após a aquisição da imagem do ambiente é realizada uma operação de extração de características. Na etapa de extração de características a partir da imagem são extraídas as informações f (em pixels) que representam a pose do objeto. Essas informações são utilizadas diretamente e comparadas com a pose em pixels f^{*} sendo calculado o erro (diferença entre os dois valores) utilizado no sistema.

(c) Ainda existem métodos híbridos que combinam as características da IBVS e da PBVS. Esses métodos permitem a obtenção de diferentes resultados em servovisão e vem sendo bastante explorados recentemente (MALIS, 2002).

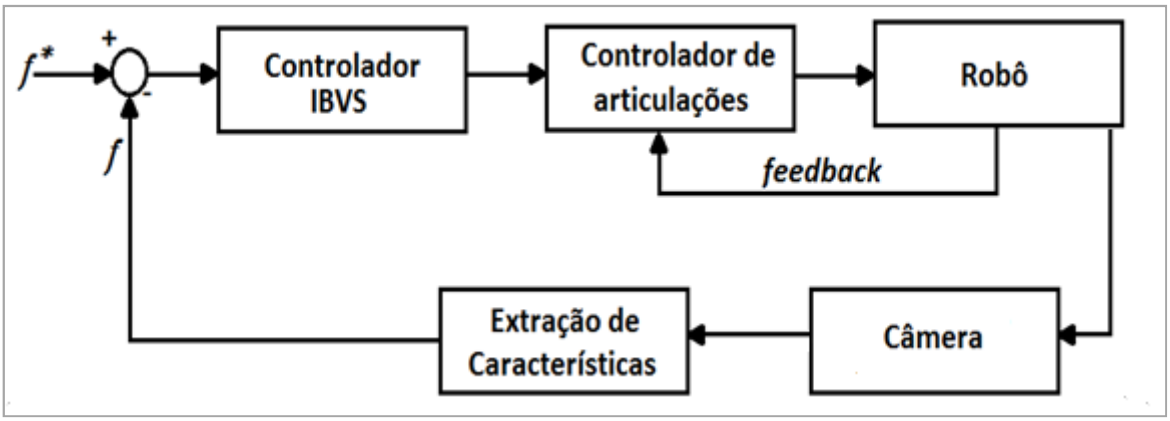


Figura 2.8 – Diagrama do modelo IBVS (adaptado de Corke (2011)).

2.2.3 Rastreamento de objetos com servovisão

Sistemas de servovisão para tarefas de rastreamento de objetos costumam fazer parte de veículos autônomos ou robôs manipuladores. Objetos em movimento rastreados por manipuladores robóticos podem ser deslocados por esteiras transportadoras ou até por veículos autônomos (transporte). Veículos autônomos também podem ser utilizados diretamente para rastrear objetos (CORKE, 2011).

Na indústria o rastreamento de objetos é de grande interesse pelo fato de que em muitas aplicações os objetos necessitam ser deslocados entre estações de trabalho (GORTCHEVA, et al., 2001).

A preocupação desse trabalho é principalmente no rastreamento de objetos por servovisão em esteiras transportadoras. Esteiras transportadoras movem objetos por caminhos fixos e específicos possuindo movimentos contínuos com velocidade constante ou assíncronos com alternância entre movimentações e pausas. Os movimentos de esteiras transportadoras podem ser de direção única (como movimentos lineares) ou movimentos contínuos e circulares (GROOVER, 2013). Sem a utilização de servovisão é bastante comum a utilização de marcadores para indicar a posição da esteira e auxiliar no rastreamento de objetos (BORANGIU, ANTON e DOGAR, 2010). Essa utilização de marcadores na esteira é uma operação com pouca flexibilidade e dessa forma existem várias pesquisas para o uso de sensores de visão nas operações de rastreamento como demonstrado em Shin, et al. (2006). Shin, et al. (2006) descrevem que para o rastreamento de objetos

em esteiras transportadoras ser realizado é necessário informações referente a pose e velocidade do objeto e sensores de visão podem permitir isso.

Algumas das primeiras técnicas para o rastreamento de objetos em movimento através de servovisão podem ser encontradas em Papanikolopoulos, Khosla e Kanade (1993), Hashimoto e Kimura (1995) e Corke e Good (1993) sendo esse ultimo bastante explorado em outros trabalhos.

Corke e Good (1993) descrevem uma arquitetura para o controle de robôs por servovisão e operações de rastreamento de alto desempenho. O objetivo do sistema de servovisão nesse trabalho é manter o objeto centralizado no campo de visão da câmera. Nesse trabalho um controle por antecipação (em inglês "*feedfoward")* foi utilizado, onde o objetivo era estimar a velocidade de um estado seguinte ao estado atual de um objeto em um sistema de controle. Para o processo de estimativa foi utilizado um filtro de Kalman, que além de realizar a estimativa também tinha a função de realizar o processo de filtragem de ruídos causados por problemas de visão computacional no reconhecimento da pose do objeto. O conjunto da arquitetura do sistema de controle e o filtro de Kalman se demonstraram mais eficientes em relação á outros métodos comparados na época e são frequentemente encontrados em diversos trabalhos na literatura.

A técnica de servovisão para o rastreamento de objetos pode ser IBVS ou PBVS, porém a maioria dos trabalhos encontrados nessa pesquisa utilizam a técnica IBVS e com isso o seguimento desse trabalho é baseado técnica.

Objetos em movimento podem possuir variações suaves (pouca variação no tipo do movimento) ou com variações aleatórias. Normalmente os trabalhos que estudam servovisão em ambientes industriais descrevem as variações dos objetos como suaves com movimentação linear e/ou circular (um exemplo é o caso de esteiras transportadoras) como em Papanikolopoulos, Khosla e Kanade (1993), Denker, Sabanovic e Kaynak (1994) e Kho e Kwon (2012). Movimentações com variações aleatórias são normalmente encontrados em ambientes não estruturados como em Gupta, Messom e Demidenko (2005), Chaudhuri e Konar (2007) e Gortcheva, et al. (2001).

Em Chaumette e Hutchinson (2007) é descrito que uma forma de executar o rastreamento de objetos é através do controle de velocidade do robô. O controlador de velocidade é baseado em uma configuração *eye-in-hand* em um manipulador robótico. O objetivo do controlador é determinar a velocidade de movimentação

adequada do manipulador robótico para reduzir o erro entre a pose desejada desse manipulador robótico em relação ao objeto e a pose atual do manipulador robótico em relação ao mesmo objeto, sendo que esse objeto está em movimento.

Baseado no trabalho de Chaumette e Hutchinson (2007) é considerado uma câmera localizada na ponta do manipulador robótico, ou seja, uma configuração *eye-in-hand* e o objetivo do sistema é posicionar essa câmera corretamente em relação ao objeto. Os estudos e notações utilizados a seguir também são baseados nos trabalhos de Chaumette e Hutchinson (2006), Siciliano, et al. (2009) e Corke (2011).

O erro *e* utilizado nesse trabalho para IBVS representa a diferença entre a pose desejada e atual do objeto em relação ao manipulador robótico na imagem. A equação 2 representa o cálculo de *e* nos termos de características da imagem.

$$e = s - s^* \tag{2}$$

e: erro entre a pose desejada e atual do objeto em relação ao manipulador robótico (diferente da equação 1, *e* está em termos de pixels da imagem). *s* e *s**: são respectivamente a pose atual e desejada do objeto em relação ao manipulador robótico na imagem.

Considerando um objeto sem movimento, um método para redução do erro *e* pode ser proposto através de um controlador de velocidade para o manipulador robótico. Dessa forma a velocidade que deve ser inserida no manipulador robótico é baseada no erro *e*. Para isso é necessário uma relação da variação de *s* no tempo e a velocidade com que o manipulador robótico se movimenta conforme a equação 3.

$$\dot{s} = L_s * V_c \tag{3}$$

$\dot{s}$ é a variação de *s* no tempo. V_c é a velocidade com que o manipulador robótico se movimenta. $L_s \in R^{kx6}$ é a matriz de interação relacionada com s^*. *k* é a quantidade de características da imagem utilizada para representar o objeto.

Nesse trabalho, por exemplo, o padrão adotado são 4 pontos na imagem com duas coordenadas cada (*x* e *y*) sendo assim *k=8*. L_s é responsável por converter informações de V_c (que está no espaço de trabalho) para o plano da imagem. Nesse trabalho L_s é denotado como

$Ls=$

$-1/Z_1$	0	x_1/Z_1	x_1*y_1	$-(1+x_1^2)$	y_1
0	$-1/Z_1$	y_1/Z_1	$1+y_1^2$	$-x_1*y_1$	$-x_1$
$-1/Z_2$	0	x_2/Z_2	x_2*y_2	$-(1+x_2^2)$	y_2
0	$-1/Z_2$	y_2/Z_2	$1+y_2^2$	$-x_2*y_2$	$-x_2$
$-1/Z_3$	0	x_3/Z_3	x_3*y_3	$-(1+x_3^2)$	y_3
0	$-1/Z_3$	y_3/Z_3	$1+y^2$	$-x_3*y_3$	$-x_3$
$-1/Z_4$	0	x_4/Z_4	x_4*y_4	$-(1+x_4^2)$	y_4
0	$-1/Z_4$	y_4/Z_4	$1+y_4^2$	$-x_4*y_4$	$-x_4$

onde x_n e y_n são coordenadas 2D do ponto *n* na imagem e Z_n é a profundidade do ponto *n* no mundo real (3D) em relação ao sistema de referência da câmera.

O sistema de controle utilizado nesse trabalho tenta garantir que o erro sofra uma diminuição proporcionalmente ao seu valor atual assim como na equação 4

$$\dot{e} = \dot{s} = -\lambda * (s - s^*) = -\lambda * e \tag{4}$$

λ: ganho proporcional. $\dot{e}$: variação de *e*. Considerando que o objetivo do sistema seja que o erro chegue a 0, $\dot{e}$ está relacionado com $\dot{s}$.

A partir da equação 4 e da equação 3, a velocidade da câmera pode ser obtida através de um controle proporcional como definido na equação 5

$$V_c = -\lambda * L_s^+ * e \tag{5}$$

L_s^+: estimativa da matriz pseudo-inversa ($L_s^+ = (L_s^T * L_s)^{-1} * L_s^T$), gerada a partir de s^* e a maioria de seus parâmetros são constantes (*Z* deve ser definida).

O processo de como gerar L_s e da aproximação L_s^+ não são abordados nesse trabalho, pois envolvem questões com projeção perspectiva do mundo real (3D) para uma imagem (2D) e estimativas de parâmetros. Informações sobre essas abordagens podem ser encontradas em Chaumette e Hutchinson (2006).

Com a variação do erro no tempo para um objeto fixo é possível definir uma variação do erro no tempo para um objeto em movimento acrescentando uma

variação de *e* no tempo em relação ao movimento do objeto. Dessa forma adicionando $\frac{\delta e}{\delta t}$ na equação 3 é obtida a equação 6.

$$\dot{e} = \dot{s} = L_s * V_c + \left(\frac{\delta e}{\delta t}\right) \quad (6)$$

$\frac{\delta e}{\delta t}$: variação de *e* no tempo em relação ao movimento do objeto.

A nova lei de controle (equação 7) para velocidade do robô com o objeto em movimento é definida isolando V_c na equação 6 e substituindo ė pela equação 4 já considerando L_s como a matriz pseudo-inversa e uma estimativa, ou seja, L_s^+.

$$V_c = -\lambda * L_s^+ * e - L_s^+ \left(\frac{\hat{\delta} e}{\hat{\delta} t}\right) \quad (7)$$

L_s^+: matriz de interação. $\frac{\hat{\delta} e}{\hat{\delta} t}$: estimativa de $\frac{\delta e}{\delta t}$ utilizado para compensar o erro após o movimento do objeto.

Como $\frac{\hat{\delta} e}{\hat{\delta} t}$ esta relacionado com a imagem, L_s^+ foi adicionado à equação para realizar a conversão para essa variável em relação a velocidade da câmera. A estimativa de $\frac{\hat{\delta} e}{\hat{\delta} t}$ está relacionada com a eficiência do rastreamento de objetos móveis através de servovisão. Essa lei de controle considera objetos em movimento com velocidade constante que pode ser utilizada no caso de rastreamento de objetos através de esteira transportadora.

Tendo em vista que o erro seguinte do objeto precisa a ser estimado antes da movimentação do objeto acontecer, um método baseado em controle por antecipação (*feedfoward*) foi utilizado. Esse método necessita que as medições atuais do objeto na imagem e velocidade da câmera estejam disponíveis, e dessa forma a expressão matemática para estimativa de $\frac{\hat{\delta} e}{\hat{\delta} t}$ é dada pela equação 8

$$\frac{\hat{\delta} e}{\hat{\delta} t} = \dot{e}^* - L_s * V_c \quad (8)$$

$\dot{e}^*$ descrito em função do tempo *t* é representado pela equação 9

$$\dot{e}^*(t) = \frac{e(t)-e(t-\Delta t)}{\Delta t} \quad (9)$$

Desta forma um método que pode ser utilizado para auxiliar na estimativa da variável $\frac{\widehat{\delta e}}{\delta t}$ é o filtro de Kalman (CHAUMETTE e HUTCHINSON, 2007).

2.2.4 Ferramentas para o desenvolvimento de servovisão

Algumas ferramentas para auxiliar no desenvolvimento e simulação de sistemas de servovisão são descritas a seguir. As três principais ferramentas encontradas foram as *toolboxes Vision* e *Robotics* para Matlab de Peter Corke (CORKE 2011) com um tutorial da ferramenta, a ferramenta VISP (*Visual Servoing Platform*) do grupo LAGADIC (LAGADIC, 2013) e a ferramenta JaViSS (*Java-based Visual Servo Simulator*) desenvolvida por Cervera (2010).

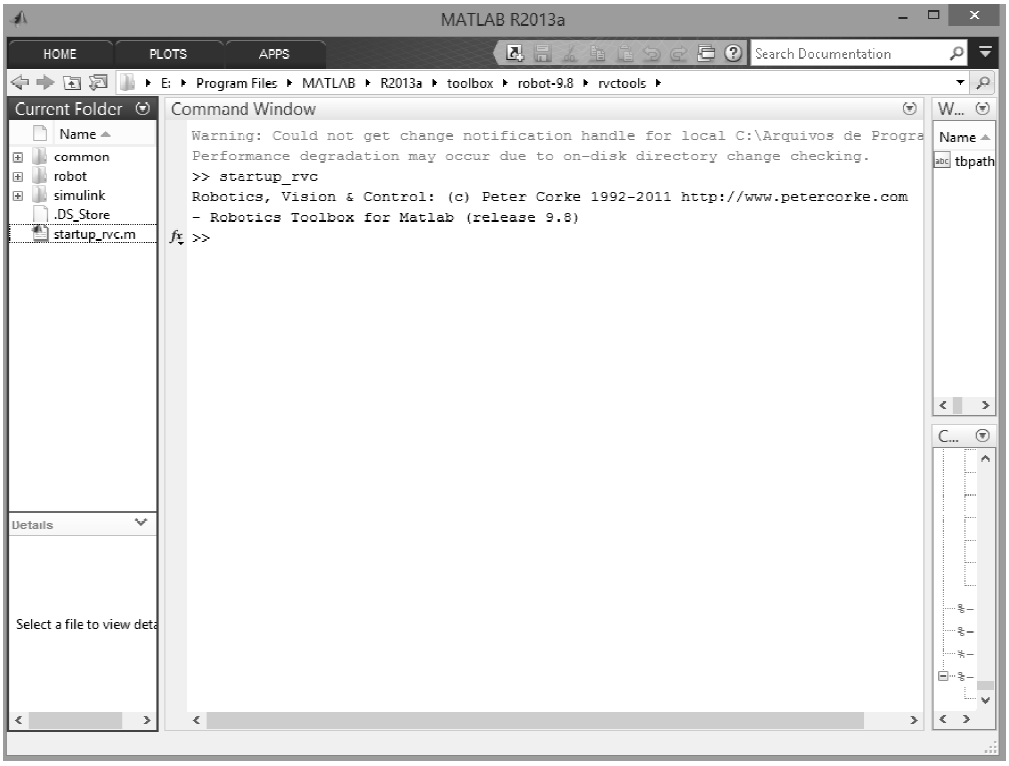


Figura 2.9 – Ambiente MATLAB 2013 com a toolbox Robotics de Peter Corke .

As caixas de ferramentas Robotics (ilustrada na Figura 2.9 em conjunto com o MATLAB 2013) e Vision para o Matlab foram desenvolvidas por Peter Corke. Essas ferramentas oferecem muitas funções úteis em robótica como dinâmica, cinemática e geração de trajetória, recursos para realizar operações de visão computacional. As caixas de ferramentas são úteis para simulação com braços robóticos, e análise de resultados de experimentos de robôs reais. As caixas de ferramentas possuem funcionalidades para o ambiente Simulink, métodos de comparações para outras implementações diferentes, rotinas simples, entre outras (CORKE, 2011).

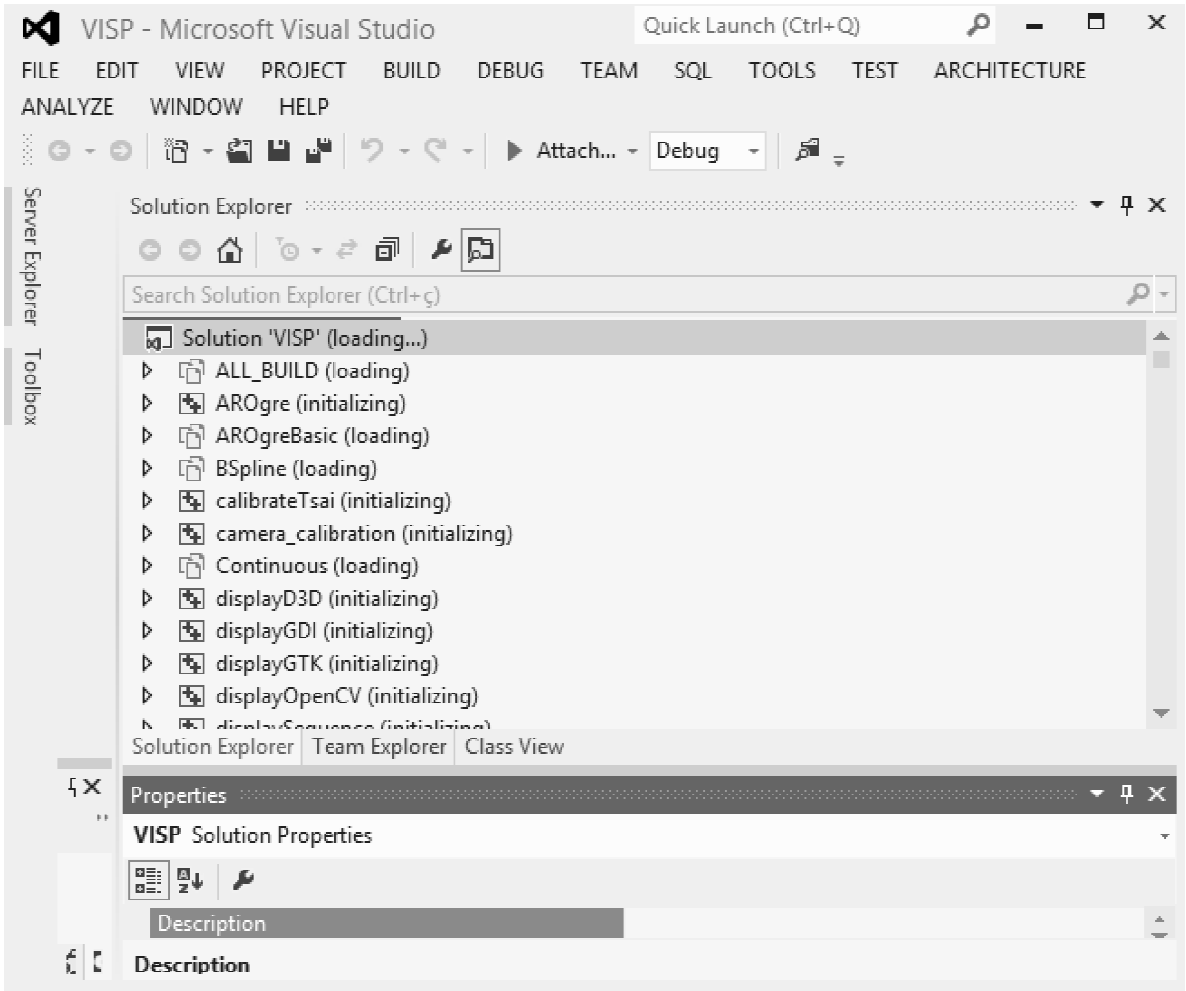


Figura 2.10 – Exemplo de funções da ferramenta VISP e Visual Studio 2012 .

A ferramenta VISP é uma plataforma de servovisão desenvolvida e mantida pelo grupo Lagadic da empresa IRISA (*Institute for Research in IT and Random Systems*) em Rennes na França. VISP é uma biblioteca completa que permite aplicações e desenvolvimento de rastreamento visual e servovisão. Desenvolvido

em C++, atualmente na versão 2.6.2, open-source com licença GNU GPLv2, possui módulos disponíveis para Linux, OSX e Windows. A ferramenta VISP também permite a interoperabilidade com bibliotecas de terceiros (LAGADIC, 2013). Na Figura 2.10 pode ser visualizado a algumas funções da ferramenta VISP sendo utilizada em conjunto com a ferramenta VISUAL STUDIO 2012.

A ferramenta JaViSS (representada na Figura 2.11) é um ambiente de simulação escrito em Java, que possui recursos gráficos para simulação de um manipulador robótico. Para o funcionamento é necessário o JAVA JDK. JaViSS é executado por meio de Java Web Start funcionando no próprio navegador. A cinemática do manipulador robótico é baseado na caixa de ferramentas Robotics de Peter Corke. Essa ferramenta permite simular os diferentes tipos de servovisão IBVS e PBVS, controlar a pose alvo do objeto, características da câmera, visão do observador, entre outros detalhes. Um problema é que com essa ferramenta não é possível fazer alterações no código o que a torna pouco flexível.

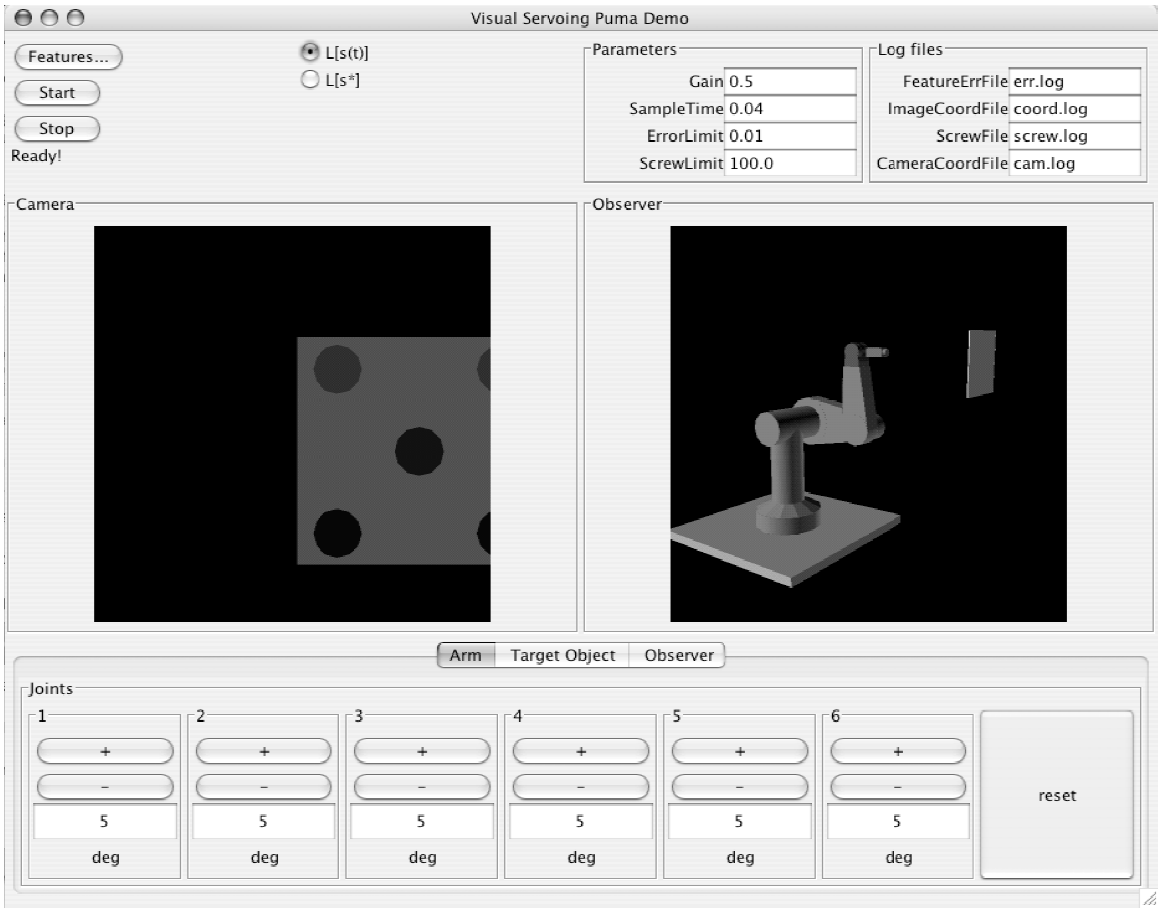


Figura 2.11 – Ferramenta JaViSS (CERVERA, 2010).

2.3 Filtro de Kalman

Rudolf Kalman descreveu que alguns dos problemas das áreas de controle envolvem a predição de sinais, separação de sinais a partir de ruídos aleatórios e detecção de sinais na presença de ruídos. Tais problemas são tratados através de interpolação, filtragem e predições que em conjunto são referenciados como estimativa. Através de seus estudos sobre sinais, probabilidade, sistemas dinâmicos, estatísticos, entre outros estudos, Kalman chegou ao desenvolvimento de um método matemático conhecido como filtro de Kalman (KALMAN, 1960).

Pela definição de Welch e Bishop (1995) o filtro de Kalman é um conjunto de equações que proporciona uma computação recursiva eficiente para estimar o estado de um processo de forma a minimizar o erro quadrático. O filtro de Kalman é um estimador para resolver o problema linear quadrático, um problema de estimação do estado instantâneo de um sistema linear perturbado por ruído. O propósito do filtro de Kalman é estimar valores que se aproximem dos valores reais através de valores medidos ao longo do tempo (corrompidos por ruídos ou incertezas). Esse propósito também pode ser observado na devinição de Russell e Norvig (2009) onde o filtro de Kalman é descrito como uma ferramenta de raciocínio probabilístico ao longo do tempo, ou seja, uma ferramenta que permite realizar uma estimativa atual de um determinado evento a partir de medidas que ocorreram anteriormente. Russell e Norvig (2009) ainda descrevem que o filtro de Kalman é capaz de considerar aspectos dinâmicos do problema que são essenciais para o processo de estimativa por sofrerem alterações rapidamente.

Uma das áreas de aplicações do filtro de Kalman é a servovisão, onde é utilizado em tarefas de localização de robôs, navegação, rastreamento, controle de movimento, estimativa e previsão, reconstrução de imagens, entre outros. Uma revisão sobre aplicações com filtro de Kalman pode ser encontrada em Chen (2012).

Em visão computacional e servovisão ruídos são causados por problemas na extração de características da imagem utilizadas no processo de localização da pose do objeto. Alguns exemplos da literatura em trabalhos com servovisão envolvem ruídos gaussianos como apresentado por Malis e Rives (2003) e Corke e Good (1993) onde o filtro de Kalman é utilizado para o tratamento desses ruídos.

O filtro de Kalman no tempo discreto (modelo original) é definido por duas equações onde essas equações são descritas no tempo *k*. A equação 10 é responsável por realizar uma estimativa do estado.

$$X_k = F*X_{k-1} + B*U_{k-1} + W_{k-1} \quad (10)$$

X_k: estado estimado do processo estocástico $X_0, X_1, X_2, ..., X_k$ gerado a partir das observações $Z_0, Z_1, Z_2, ..., Z_{k-1}$, nas entradas $U_0, U_1, U_2, ..., U_{k-1}$ e nas estimativas anteriores $X_0, X_1, X_2, ..., X_{k-1}$. F: modelo de transição de estados. X_{k-1}: estado predito. B: modelo de entradas. U_{k-1}: vetor de entradas. W_{k-1}: ruído do processo.

$$Z_k = H*X_k + V_k \quad (11)$$

A equação 11 demonstra o valor observado. Z_k: combinação da estimativa e ruído observado. H: modelo de observação. V_k: ruído da observação.

A utilização e relação dessas equações são estudadas no modelo do processo recursivo do filtro de Kalman

2.3.1 Processo Recursivo do Filtro de Kalman

Por usualidade e facilidade as notações utilizadas a seguir consideram estimações ou predições no tempo *k* dados estados anteriores até o tempo *k,* denotadas por $k|k$ (são encontradas no trabalho $_{K|K}$, $_{K|K-1}$, $_{K-1|K-1}$, entre outros). Os cálculos apresentados são baseados nos estudos de Kalman (1960), Welch e Bishop (1995) e Corke (2011) e fazem parte dos modelos implementados na ferramenta VISP utilizada nesse trabalho. O processo recursivo do filtro de Kalman é apresentado na Figura 2.12.

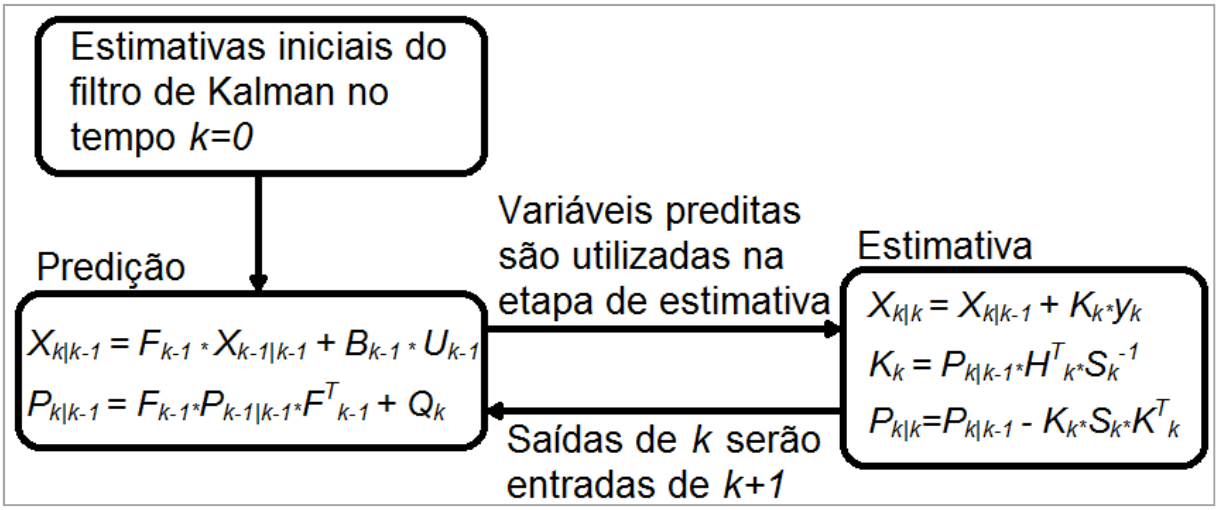


Figura 2.12– Representação do processo recursivo do filtro de Kalman.

$X_{k|k-1}$: estado predito. F_{k-1}: modelo de transição de estados. $X_{k-1|k-1}$: estimativa do estado anterior ao atual. B_{k-1}: modelo de entradas de controle. U_{k-1}: vetor de entradas do controle. $P_{k|k-1}$: previsão da covariância do estado estimado. $P_{k-1|k-1}$: estimativa da covariância no processo anterior. F^T_{k-1}: matriz transposta do modelo de transição de estados, Q_k: covariância do ruído do processo. $X_{k|k}$: estado estimado. K_k: ganho do filtro. y_k: resíduo da observação. K^T_k: matriz transposta do ganho do filtro. H^T_k: matriz transposta da evolução da observação. S_k^{-1}: matriz da inversa do resíduo da covariância. $P_{k|k}$: estimativa da covariância. S_k: resíduo da covariância.

No processo de predição (*a priori*) são utilizadas informações do estado $X_{k|k-1}$ e da matriz de covariância do estado $P_{k|k-1}$ para obter os novos valores preditos que posteriormente são utilizados na etapa de correção. A etapa de estimativa (*a posteriori*) realiza os cálculos de valores auxiliares (K_k, S_k e y_k) e corrige a estimativa do estado e da covariância, após essa etapa $X_{k|k}$ é utilizado no sistema de controle.

Conforme demonstrado na Figura 2.12 o processo de recursão do filtro de Kalman é dado por uma sequencia de equações que podem ser classificadas em dois grupos: As equações 12 e 13 que são de atualização do tempo / predição / informação a priori e as equações 14, 15 3 16 que são de atualização da observação / filtragem ou estimativa / informação a posteriori.

$$X_{k|k-1} = F_{k-1} * X_{k-1|k-1} + B_{k-1} * U_{k-1} \quad (12)$$

$X_{k|k-1}$ é o estado predito e está representado na equação 12, de forma que este estado X_k é predito no tempo *k-1* de acordo com a estimativa anterior $X_{k-1|k-1}$. A estimativa anterior $X_{k-1|k-1}$ quando utilizada para gerar o estado predito $X_{k|k-1}$, é corrigida de acordo com a observação Z_{k-1}.

O modelo de transição de estados F_{k-1} é responsável por relacionar o estado anterior á predição do estado atual sendo aplicado á estimativa de estado $X_{k-1|k-1}$.

$$P_{k|k-1} = F_{k-1} * P_{k-1|k-1} * F^T_{k-1} + Q_k \quad (13)$$

A previsão da covariância do estado estimado $P_{k|k-1}$ é utilizada para calcular o ganho do filtro K_k e a estimativa da covariância no próximo estado $P_{k|k}$ (pela equação 16). A previsão da covariância $P_{k|k-1}$ é calculada com base no modelo de transição de estados F_{k-1}, na estimativa da covariância no processo anterior $P_{k-1|k-1}$ (no estado atual é denotada por $P_{k|k}$) e com a covariância *Q* do ruído do processo *W*. A covariância do estado $P_{k|k}$ é uma estimativa da incerteza e da correlação entre as incertezas dos componentes do estado. A covariância Q_k descreve as instabilidades dos componentes do estado.

$$X_{k|k} = X_{k|k-1} + K_k * y_k \quad (14)$$

O estado estimado $X_{k|k}$ pelo filtro de Kalman é calculado na etapa de correção/atualização onde o novo estado ($X_{k|k}$) é uma combinação dos dados passados com a última observação.

$$K_k = P_{k|k-1} * H^T_k * S_k^{-1} \quad (15)$$

O ganho do filtro K_k reflete na precisão do estado predito em relação à nova observação e também é utilizado no processo de estimativa de $P_{k|k}$ onde tem a função de minimizar o erro dessa estimativa.

H_k descreve a evolução da observação, ou seja, como o sistema é medido. Essa variável é utilizada na observação do sistema, no cálculo de K_k e no cálculo do resíduo da covariância.

$$P_{k|k}=P_{k|k-1} - K_k * S_k * K^T_k \quad (16)$$

O resíduo da covariância S_k é descrito na equação 17. R_k é a covariância do ruído da observação *V.*

$$S_k= H_k * P_{k|k-1} * H^T_k + R_k \quad (17)$$

O resíduo da observação $y_{k|k}$ reflete a discrepância entre o estado (observação) predito e a observação atual e quanto mais próximo de zero é esse valor mais próximo do estado predito e a observação atual estarem de acordo. A equação 18 define esse resíduo

$$y_{k|k}=Z_k-(H_k * X_{k|k-1}) \quad (18)$$

Para execução do filtro de Kalman são necessários estimativas iniciais para o estado do sistema e e a covariância do estado. Para que essas estimativas possam ser estabelecidas são necessários também a inicialização de outros valores.

2.3.2 Aplicações com filtro da Kalman em servovisão

Na ferramenta VISP (LAGADIC, 2013) existe um componente com um filtro de Kalman baseado nos trabalhos clássicos da área de servovisão encontrados na literatura como Chaumette e Hutchinson (2006), Chaumette e Hutchinson (2007), Corke e Good (1993) e Bar-Shalom, Li e Kirubarajan (2001).

A ferramenta VISP permite a utilização de filtros de Kalman específicos para diferentes abordagens. As equações para o filtro de Kalman utilizadas na ferramenta VISP são as mesmas que foram apresentadas previamente (equações de 12 a 18).

O filtro de Kalman pode ser utilizado para melhorar o resultado da estimativa de pose futura de objetos em movimento ou auxiliar melhorando algumas estimativas de variáveis úteis em controladores de velocidade e aceleração, sendo que todas essas abordagens estão sujeitas a incertezas e ruídos gerados no processo de aquisição de dados e etapas de processamento. Das aplicações descritas consideramos duas delas: (a) melhoria na estimativa de posição futura de

objetos em movimento e (b) melhoria na estimativa de velocidade (que pode ser utilizada em controladores de velocidade de braços robóticos).

Considerando o modelo IBVS e as implementações da ferramenta VISP, a seguir são apresentadas algumas particularidades da utilização do filtro de Kalman para cada uma dessas aplicações.

2.3.2.1 Filtro de Kalman na estimativa de pose futura de objetos em movimento

Com o modelo IBVS um único ponto/pixel é considerado (mais pontos poderiam ter sido utilizados), sendo definido com as coordenadas *x* e *y* (denotadas por *px* e *py*) do objeto na imagem para representar a pose de um objeto. Esse ponto/pixel é utilizado como a observação (Z_k) no filtro de Kalman.

Para estimativa da pose futura do objeto em movimento é utilizado a equação 19 que utiliza as coordenas *px* e *py* (sinais) do objeto na imagem (plano da imagem)

$$Z_k = \begin{array}{|l} px_{k+1} = px_k + \frac{px_k - px_{k-1}}{\Delta k} \\ \hline py_{k+1} = py_k + \frac{py_k - py_{k-1}}{\Delta k} \end{array} \qquad (19)$$

px_{k+1} e py_{k+1}: pose futura do objeto. px_k e py_k: pose atual do objeto. px_{k-1} e py_{k-1}: pose anterior do objeto. Δk: variação de tempo discreto entre $px_k|py_k$ e $px_{k-1}|py_{k-1}$.

Todos os cálculos são baseados em operações aritméticas com matrizes e com isso algumas matrizes são definidas previamente ao funcionamento do sistema. A matrizes estão na forma de linhas por colunas. As variáveis ω e δ representam respecivamente o número de estados e sinais. Nesse trabalho o número de estados é 1 e o número de sinais é 2 – cordenadas x e y.

O modelo de transição de estado *F* é uma matriz quadrada de $2^{\delta}*^{\omega}$ X $2^{\delta}*^{\omega}$, sendo para o caso de dois sinais definido como

$$F = \begin{bmatrix} 1 & \Delta k & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta k \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

O modelo de estado *H* é uma matriz com δ*ω *X 2* $^{\delta*\omega}$ onde no exemplo é utilizada como

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

O ruído da observação *R* é uma matriz com δ*ω X δ*ω onde no exemplo é utilizada como

$$R = \begin{bmatrix} sR[0] & 0 \\ 0 & sR[1] \end{bmatrix}$$

sR é uma matriz 1 X δ*ω onde no exemplo é utilizada como

$$sR = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Na estimativa da pose futura do objeto *sQ* é uma matriz 1 X 2 $^{\delta*\omega}$

$$sQ = \begin{bmatrix} 0.000001 \\ 0 \\ 0.000001 \\ 0 \end{bmatrix}$$

e matriz de covariância do estado é definida como

$$Q = \begin{bmatrix} sQ*\frac{\Delta k^3}{3} & sQ*\frac{\Delta k^2}{2} & 0 & 0 \\ sQ*\frac{\Delta k^2}{2} & sQ*\Delta k & 0 & 0 \\ 0 & 0 & sQ*\frac{\Delta k^3}{3} & sQ*\frac{\Delta k^2}{2} \\ 0 & 0 & sQ*\frac{\Delta k^2}{2} & sQ*\Delta k \end{bmatrix}$$

A matriz de covariância do processo no caso do rastreamento de objeto em movimento é

$$P_{k|k} = \begin{bmatrix} sR & \frac{sR}{(2 * \Delta k\ )} & 0 & 0 \\ \frac{sR}{(2 * \Delta k\ )} & \frac{sQ*2*\Delta k}{3} + \frac{sR}{(2*\Delta k^2)} & 0 & 0 \\ 0 & 0 & sR & \frac{sR}{(2 * \Delta k\ )} \\ 0 & 0 & \frac{sR}{(2 * \Delta k\ )} & \frac{sQ*2*\Delta k}{3} + \frac{sR}{(2*\Delta k^2)} \end{bmatrix}$$

2.3.2.2 Filtro de Kalman para auxiliar na estimativa de velocidade

Na lei de controle utilizada em um controlador de velocidades e apresentada na equação 7 é observado a necessidade de estimar o valor da variável $\left(\frac{\delta e}{\delta t}\right)$. Para isso é necessário uma estimativa inicial dessa variável e o valor obtido é melhorado pelo filtro de Kalman. A equação 20 representa uma extensão para cada variável da equação 9 referente a estimativa inicial da variável $\left(\frac{\hat{\delta} e}{\hat{\delta} t}\right)$.

$$Z_k = de/dt = \begin{array}{|l} deX/dt = \frac{errox_k - errox_{k-1}}{\Delta k} - jx*vmx \\ \hline deY/dt = \frac{erroy_k - erroy_{k-1}}{\Delta k} - jy*vmy \end{array} \quad (20)$$

Onde *deX/dt* e *deY/dt* compõem a variável $\left(\frac{\hat{\delta} e}{\hat{\delta} t}\right)$ e representam a velocidade (no plano da imagem) em que o erro entre a pose atual objeto em relação ao manipulador robótico e a pose desejada do objeto em relação ao manipulador robótico é alterada. As variáveis $errox_k$ e $erroy_k$ são consideradas no instante atual e referentes aos erros atuais entre o objeto e o manipulador robótico e os erros desejados entre o objeto e o manipulador robótico.

As variáveis $errox_{k-1}$ e $erroy_{k-1}$ são consideradas no instante anterior e referentes aos erros atuais entre o objeto e o manipulador robótico e os erros

desejados entre o objeto e o manipulador robótico. As variáveis *jx* e *jy* são o jacobiano da imagem para cada posição, utilizado para converter as velocidades *vmx* e *vmy*. As variáveis *vmx* e *vmy* representam as velocidades em cada posição do manipulador robótico. Δk é a variação de tempo entre a ocorrência de $errox_k|erroy_k$ e $errox_{k-1}|erroy_{k-1}$. É notável que para calcular o erro entre o objeto e o manipulador robótico também são necessários as posições dos pixels *px* e *py*.

O modelo de transição de estado *F* possui a mesma estrutura do modelo utilizado na estimativa de posição futura do objeto sendo uma matriz quadrada de $2^{\delta*\omega}$ X $2^{\delta*\omega}$ (nesse trabalho é considerado o número de estados igual á 1), sendo para o caso de dois sinais definido como

$$F = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & rho & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & rho \end{bmatrix}$$

rho: grau de correlação entre sucessivas acelerações que possuí o valor entre 0 e 0.9 e pode ser definido previamente.

H, R e *sR* são os mesmos utilizados no caso da estimativa de posição futura de objeto, ou seja, são comuns para os dois modelos.

No controlador de velocidade sQ é uma matriz 1 X $2^{\delta*\omega}$

$$sQ = \begin{bmatrix} 0 \\ 0.000001 \\ 0 \\ 0.000001 \end{bmatrix}$$

e matriz de covariância do estado

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & sQ & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & sQ \end{bmatrix}$$

A matriz de covariância do processo no caso do controlador de velocidade

$$P_{k|k} = \begin{bmatrix} sR & 0 & 0 & 0 \\ 0 & \dfrac{sQ}{(1-rho^2)} & 0 & 0 \\ 0 & 0 & sR & 0 \\ 0 & 0 & 0 & \dfrac{sQ}{(1-rho^2)} \end{bmatrix}$$

2.3.2.3 Metodologia de um sistema de controle com filtro de Kalman

As etapas dos sistemas de controle para estimativa de pose futura do objeto ou estimativa da variável auxiliar para o controlador de velocidade considerando a utilização do filtro de Kalman são descritas como:

1- Inicia as variáveis do sistema de controle (variáveis da estimativa de pose futura do objeto ou controlador de velocidades)

2- Define variáveis iniciais e inicializa o filtro de Kalman.

3- Inicia a execução do sistema de controle. Aqui é iniciado o ciclo de execução do sistema de controle, onde ocorrem desde a etapa de aquisição da imagem, passando pela etapa de verificação do erro entre o manipulador robótico e o objeto (para o caso do controlador de velocidades) até á etapa de reatroalimentar (*feedback*) o sistema.

4- Adquire imagem.

5- Realiza tratamento na imagem, extração de características e identificação das poses do objeto e manipulador.

6- Verifica se o erro desejado e atual do manipulador robótico e objeto está suficientemente mínimo para executar alguma operação de correção (para o caso do controlador de velocidades).

7- Calcula as variáveis (de acordo com o sistema de controle) que irão compor o sinal a ser filtrado pelo filtro de Kalman.

8- O filtro de Kalman recebe as informações referentes ao sinal que irão compor a observação Z_k no instante/iteração *K*.

9- Realiza a filtragem do sinal através do Filtro de Kalman de acordo com o fluxograma da Figura 2.12.

10- Utiliza a estimativa do filtro de Kalman para a lei de controle.

11- O robô recebe os valores estimados pelo filtro de Kalman e se movimenta de acordo com essa informação.

12- Se ainda for necessário executar o rastreamento retorna a etapa 4, se não finaliza o processo.

Na Figura 2.12 é considerada uma iteração de filtragem do filtro de Kalman sendo possível observar a utilização das equações (12 a 18) do filtro de Kalman. As equações das variáveis auxiliares estão subentendidas. Na primeira iteração do processo é apenas carregado o estado estimado $X_{k|k}$ e realizado uma etapa de predição para posteriormente continuar a execução do filtro. As variáveis de predição ficam armazenadas no sistema para serem utilizadas na etapa seguinte do sistema de controle e filtro de Kalman.

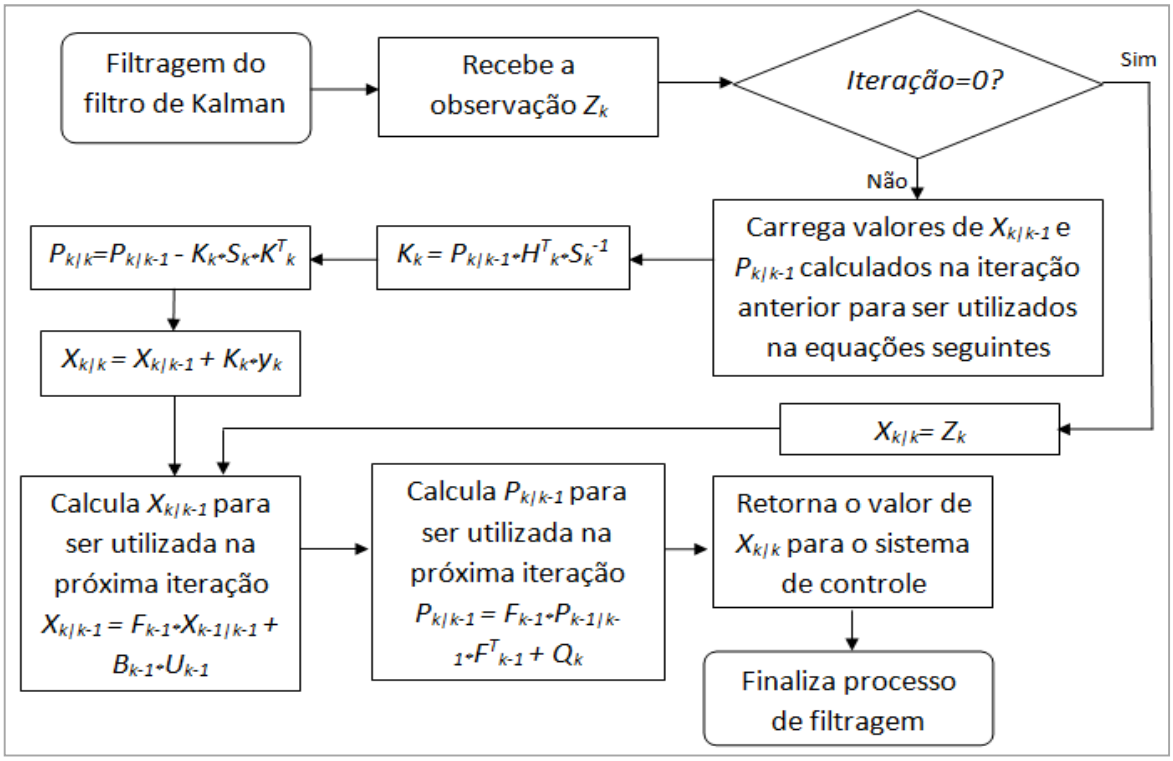


Figura 2.13 – Fluxograma de execução do filtro de Kalman (etapa 9).

Esse processo demonstra genericamente o ciclo geral de um sistema de controle e será utilizado posteriormente nos testes desse trabalho.

Capítulo 3

REVISÃO BIBLIOGRÁFICA

Esse capítulo apresenta uma revisão bibliográfica dos problemas e soluções relacionados com servovisão. Um tópico em destaque em relação aos problemas de servovisão são os *dead-times,* e nesse capítulo é realizado e apresentado um estudo de como os trabalhos estão organizados para tratar o problema.

3.1 Problemas gerais em servovisão

Servovisão é uma linha de pesquisa multidisciplinar que envolve o estudo de áreas como robótica, sistemas de controle, visão computacional, computação em tempo real, entre outros (CORKE, 2011). Justamente por ser uma área multidisciplinar problemas nessas diferentes áreas são somados a problemas específicos das áreas de servovisão e dessa forma permitindo diversas pesquisas.

Siciliano, et al. (2009) relatam que os primeiros trabalhos com servovisão foram motivados para tratar de problemas relacionados com a execução precisa de operações robóticas. Os autores descrevem que problemas causados por desgastes nas juntas de manipuladores robóticos e movimentação de objetos durante a execução de uma tarefa podem ser contornados por servovisão.

Em Malis (2002) e Malis e Rives (2003) são apontados questões relacionadas à visão computacional para servovisão. Nesses trabalhos podem ser encontrados os problemas de modelos de representação de objeto para PBVS, problemas de encontrar as características semelhantes entre a imagem e outra imagem ou modelo, problemas relacionados com oclusão de objetos, reconstrução 3D por visão

estéreo, entre outros. Ainda na área de visão computacional Corke (2011) descreve as tarefas de calibração da câmera, estimativa da matriz jacobiana da imagem além dos problemas tradicionais como o processamento de imagens, distorção da imagem e problemas com iluminação.

Com relação ao controle, além da questão da modelagem do sistema de forma que opere através de servovisão, existe uma preocupação para tratar problemas de *dead-times* que são introduzidos durante a execução do sistema e são críticos principalmente em operações de rastreamento.

De acordo com os trabalhos encontrados nessa pesquisa os *dead-times* acontecem por atrasos em operações de visão computacional como no processamento de imagem e extração de características (BAEZA, MEDINA e VÁZQUEZ, 2002), atrasos na própria operação de aquisição de imagem pela câmera (CORKE e HUTCHINSON, 2000) ou ainda por latência na transmissão de dados como transmissão de informações entre o sensor/câmera para o processador (KAWAMURA, et al, 2012). Esses *dead-times* geram atrasos na taxa de amostragem do sistema de controle, fazendo com que essa taxa de amostragem varie em relação a uma taxa de amostragem em operações sem *dead-times* (LIU, HUANG e WANG, 2012b).

3.2 *Dead-times* em servovisão

Os problemas com *dead-times* em servovisão são estudados desde o início do desenvolvimento dessa técnica. Durante essa pesquisa (em diferentes bases de trabalhos científicos como IEEE, ACM e SCOPUS) foram encontrados diversos trabalhos conforme apresentado no gráfico da Figura 3.1 onde é apresentado o número de trabalhos encontrados que mencionam o problema de *dead-times* em servovisão por épocas.

Na Figura 3.1, pode ser visualizado que recentemente existem pesquisas tratando ou mencionando o problema de *dead-times* o que também contribuiu para motivação dessa pesquisa.

A ocorrência de um simples atraso em operações de processamento de imagem (ou seja, um *dead-time*), muitas vezes, pode comprometer a execução de operações simples de servovisão (HUTCHINSON, HAGER e CORKE, 1996).

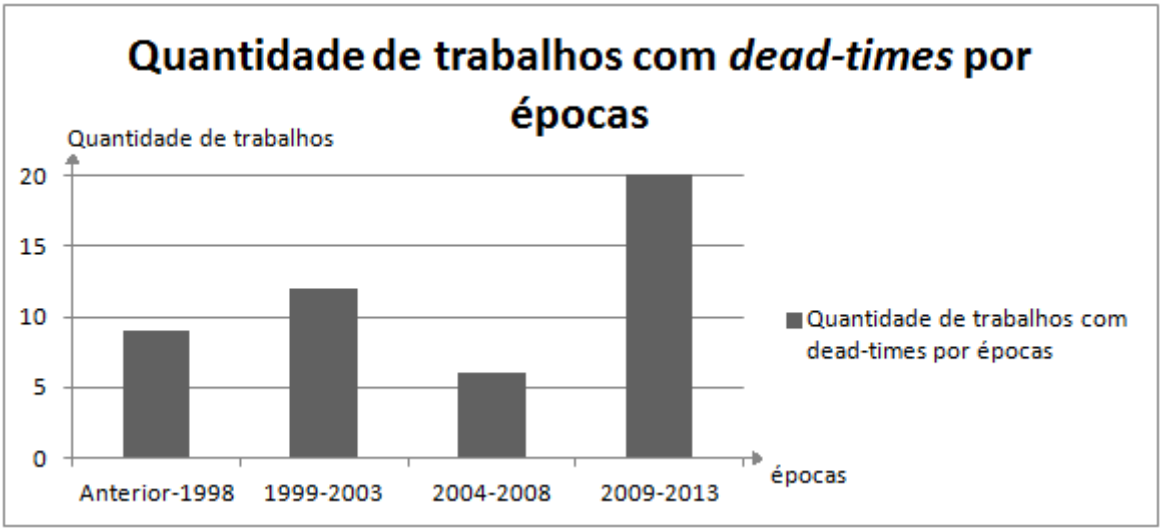


Figura 3.1 – Gráfico representando a quantidade de trabalhos encontrados nessa pesquisa que mencionam ou tratam *dead-times*.

Em tarefas de rastreamento de objetos através de manipuladores robóticos, Nagahama, et al. (2000) descrevem que *dead-times* causados por processamento de imagens (pode-se considerar outros tipos de *dead-times*) implicam em precisão ruim quando o manipulador robótico se aproxima do objeto. Outros problemas causados por *dead-times* também são encontrados em processos de identificação e reconhecimento de objetos como demonstrado por Yoon, Kosaka e Kak (2008).

De acordo com Corke e Hutchinson (2000) sistemas de servovisão necessitam que as informações sejam atualizadas em tempo real e um fator crítico é o problema de *dead-times*. Qualquer tipo de *dead-times*, quando esses acontecem, pode ser um agravante na execução de tarefas como no processo de produção em ambientes de manufatura (KHO e KWON, 2012) ou também em operações médicas que utilizam robôs como demonstrado por Hussnain, Sounkalo e Nicolas (2012).

No rastreamento de objetos em movimento o problema de *dead-times* é crítico e acontece pelo fato de ao término de um processamento com atrasos (no sistema de controle) e determinação da pose do objeto alvo pelo controlador, esse mesmo objeto pode ter se deslocado em relação à medida obtida (DENKER, SABANOVIC e KAYNAK, 1994).

A forma de tratamento dos *dead-times* é bastante ampla, existindo várias abordagens para esse tipo de problema. De acordo com os trabalhos encontrados na literatura e de uma forma um pouco mais geral é possível classificar os métodos de tratamento de *dead-times* em duas categorias: (a) métodos que procuram minimizar ou evitar a ocorrência de *dead-times* através *hardwares* com capacidade de processamento em tempo real, algoritmos ou estrutura do sistema de controle; e (b) métodos para compensar *dead-times* onde normalmente são utilizados algoritmos como filtro de Kalman, técnicas com Lógica *Fuzzy*, entre outros. Alguns exemplos de trabalhos em cada categoria são apresentados a seguir.

3.2.1 Métodos que procuram minimizar/evitar a ocorrência de *dead-times*

Masar (2006) utiliza processadores de sinais digitais (DSP - do inglês *Digital Signal Processor*) para proporcionar alto desempenho no processamento e transmissão de dados em um sistema de servovisão em operações de rastreamento de objetos onde o sistema desenvolvido se demonstrou bastante eficiente durante sua execução. Tu e Ho (2010) além de utilizarem um DSP no sistema também utilizaram um FPGA (do inglês "*Field-programmable gate array"*). O DSP utilizado nesse trabalho é responsável pela lei de controle do sistema enquanto o FPGA é responsável por executar os diversos algoritmos utilizados em *pipeline.* Outros trabalhos com servovisão que utilizam FPGAs e possuem o propósito de explorar características de *pipeline* e programação reconfigurável podem ser encontrados em Wang, Huang e Sheng (2007) e Moreno-Armendáriz, Rubio e Pérez-Olvera (2010).

Kho e Kwon (2012) também fazem o uso de *hardware* especializado em servovisão onde utilizam *hardware* especifico do robô Yamaha iVY. Os problemas de servovisão nesse trabalho são considerados em tarefas de rastreamento de objetos em esteiras transportadoras e a principal preocupação é focar essa técnica aplicada a ambientes industriais. Os autores avaliaram o sistema no quesito de precisão e embora o teste principal não esteja relacionado com tratamento de *dead-times* eles tinham a necessidade de que o *hardware* conseguisse evitar a ocorrência desses atrasos para que o sistema pudesse ser avaliado, por esse motivo foi utilizado o *hardware* especializado do robô.

Além de recursos de *hardware* outro método utilizado para evitar a ocorrência de *dead-times* está relacionado com a estrutura de controle do sistema de

servovisão, considerando questões de servovisão direta ou dinâmica. Em Pieters (2013) a preocupação da utilização de servovisão também está relacionada com ambientes industriais e evitar a ocorrência de *dead-times* no sistema de servovisão. Pieters (2013) estuda arquiteturas diretas de servovisão para melhorar o desempenho do sistema e realizam testes com o modelo de servovisão IBVS. Nos testes realizados o sistema de Pieters (2013) apresentou melhorias no desempenho em relação ao método com servovisão dinâmica. A exploração de sistemas de serovisão direta também pode ser encontrada nos trabalhos de Biro, Mcmurray e Lipkin (1997) e Corke (1995).

De Best, Molengraft e Steinbuch (2009) apresentam o uso de servovisão direta, ou seja, sem uma estrutura hierárquica no sistema de controle (servovisão dinâmica) em máquinas utilizadas na produção de produtos onde são executadas operações de forma repetitiva. Nesse trabalho é reforçado através de demonstrações a divisão entre servovisão direta e dinâmica, uma vez que as informações visuais são utilizadas diretamente como entradas no controlador de articulações do robô, sem a necessidade de outros conjuntos (blocos) no sistema de controle. O objetivo final da abordagem é conseguir um bom desempenho na taxa de amostragem do sistema. Também foi utilizado um processamento de imagens adequado e um preditor baseado filtro de Kalman para conseguir bons resultados na taxa de amostragem do sistema.

Assim como demonstrado em De Best, Molengraft e Steinbuch (2009), em Barreto, et al. (2002) é utilizado além do sistema de controle direto de servovisão abordagens de processamento de imagens e algoritmos preditores para compor um sistema que consiga um bom desempenho e que esse sistema sofra pouco com *dead-times* que podem ocorrer durante a execução. Esse bom desempenho é considerado a partir de uma taxa de amostragem em tempo real do sistema de servovisão e pouco variável durante cada etapa do ciclo do sistema.

Corke (1996) descreve em seu trabalho que entre os dois modelos de servovisão, ou seja, o modelo PBVS e o modelo IBVS esse segundo modelo apresenta menor ocorrência de *dead-times* pelo fato de que não necessita de etapas de conversão do plano 2D da imagem para o plano real 3D e também em muitas aplicações não necessita do processo de calibração da câmera.

Martinet, Berry e Gallice (1996) tentam melhorar o desempenho do sistema de servovisão através do trabalho com as primeiras derivadas das variáveis que

representam informações geométricas e de pose do objeto que são utilizadas para compor a lei de controle do sistema. Nos testes realizados Martinet, Berry e Gallice (1996) consideraram a ocorrência de *dead-times* e demonstraram uma forma geral que a ocorrência de *dead-times* pode ser reduzida e o desempenho do sistema de servovisão melhorado.

Durante essa pesquisa foi encontrado também uma técnica em que através de um algoritmo específico para processamento de imagens em conjunto com um preditor de Smith modificado tenta evitar a ocorrência de *dead-times* (XIAO e LI, 2012). Xiao e Li (2012) desenvolveram métodos na área de servovisão para operações de posicionamento em escala microscópica. Tais operações necessitam de alta capacidade de processamento e o sistema tem que evitar qualquer tipo de *dead-time.* O projeto como um todo desenvolvido foi considerado eficiente para o posicionamento microscópico.

Em Wu, et al. (2010) é proposto a utilização de diversas câmeras para o sistema de servovisão que adquirem a imagem, e as etapas de extração de características, estimativa de pose do objeto e lei de controle são realizados através de processamento distribuído. Foram utilizados um protocolo de transmissão de alta capacidade para informações visuais, terminais para processamento e esses terminais operam em paralelo. O objetivo do sistema era conseguir alta capacidade de processamento através dessa estrutura e o sistema conseguiu bons resultados. Outra abordagem com técnica de processamento distribuído pode ser encontrada em Cervera (2005).

3.2.2 Métodos que executam tratamento de *dead-times*

Além das técnicas que se preocupam em evitar a ocorrência de *dead-times* também existem as técnicas que se preocupam com abordagens para compensar os *dead-times* quando esses ocorrem através de técnicas de previsão. Gortcheva, et al., (2001) descrevem que uma forma para tratar a ocorrência de *dead-times* é através de algoritmos de estimativa de pose de objetos em movimento.

Em Kawamura et al. (2012) é demonstrado um novo método para tratar do problema de *dead-times*. Kawamura et al. (2012) ressaltam que os problemas de *dead-times* são causados por baixa taxa de amostragem da câmera, alto custo computacional no processamento de imagens e latência na transmissão de dados

entre a câmera e o processador. O método apresentado nesse trabalho utiliza simulações virtuais do sistema de referência do objeto em conjunto com equações para compensar os *dead-times* causados por processamento de imagens quando esses ocorrem.

Outros métodos para tratar *dead-times* envolvem o uso de técnicas de inteligência artificial como demonstrado em Suh e Kim (2000), Ramakoti, Vinay e Jatoth (2009) e Liu, Huang e Wang (2012b).

Em Suh e Kim (2000) são utilizadas técnicas de Lógica Fuzzy e redes neurais para aprender o comportamento do robô e alguns parâmetros do sistema de servovisão. No caso da ocorrência de *dead-times* o robô pode se comportar de acordo com o comportamento aprendido pelo sistema e dessa forma o sistema sofre menos com incertezas causadas durante sua execução.

Outro trabalho que pode ser encontrado a utilização de Lógica Fuzzy, porém para o tratamento direto de *dead-times* é em Liu, Huang e Wang (2012b) em operações de rastreamento de objetos em escala microscópica. O sistema Fuzzy utilizado nesse trabalho é aplicado em conjunto com um filtro de Kalman. A técnica demonstrou ser capaz de rastrear o objeto tanto em condições de movimentação suave do objeto como também em condições de alterações drásticas de velocidade. Um diferencial apresentado nos resultados desse trabalho é que a técnica com filtro de Kalman em conjunto com Lógica Fuzzy consegue um estabilidade mais rápida em relação ao modelo tradicional do filtro de Kalman.

A pesquisa de métodos compensativos de *dead-times* também envolvem técnicas aplicadas diretamente na modelagem do sistema de controle como *feedback* ou *feedforward* adicionais. Essas técnicas normalmente tentam prever a próxima posição do objeto rastreado com base no modelo do sistema de controle e quando ocorrem *dead-times* esses são ajustados sucessivamente nas etapas seguintes do ciclo do sistema alguns exemplos trabalhos com esse tipo de técnicas são os trabalhos de Hashimoto e Kimura (1995), Fujimoto (2003), Fujimoto e Hori (2001) e Chroust, et al. (2001).

Uma técnica bastante utilizada para compensar *dead-times* é o preditor de Smith desenvolvido por Otto J. M. Smith em 1957 (SLAVOV e ROEVA, 2011). Durante essa pesquisa a utilização do preditor de Smith foi encontrado em diferentes trabalhos como em Bowthorpe, et al. (2013), Baeza, Medina e Vázquez (2002), Vigorelli, Bonfe e Fantuzzzi (2001), Iwazaki, Murakami e Ohniski (1997) e Sim, Hong

e Lim (2002). O preditor de Smith é normalmente utilizado para compensar *dead-times*, possui a vantagem de permitir modelar o sistema sem se preocupar com questões de *dead-times* no sistema, mas esse fator também se torna uma desvantagem uma vez que o desempenho do preditor de Smith depende da precisão com que o sistema de controle foi projetado sendo muito sensível a erros de modelagem (SLAVOV e ROEVA, 2011).

Outra variedade de técnicas que funcionam como compensadores de *dead-times* podem ser encontradas em Bai, Chen e Zeng (2009), Li e Xie (2010), Kinbara, Komada e Hirai (2006), Suh, Ro e Kang (2006) e Vidal, et al. (2009).

O filtro de Kalman que já foi mencionado previamente nesse trabalho além de ser utilizado em conjunto com outras técnicas é frequentemente utilizado também sozinho ou na forma de suas extensões para o tratamento de *dead-times.* Em Gortcheva, et al., (2001) o filtro de Kalman é utilizado para auxiliar na previsão de movimento do objeto, onde a posição futura do objeto é estimada. Wunsch e Hirzinger (1997) o filtro de Kalman é capaz de compensar a ocorrência de dead-times em operações de servovisão.

Em operações de servovisão o filtro de Kalman é frequentemente utilizado devido sua capacidade de tratamento de ruídos, estimação e compensação de *dead-times* (HASHIMOTO e KIMURA, 1995). O controle *feedforward* é explorado frequentemente em conjunto ao filtro de Kalman como demonstrado por Corke e Good (1993), Hashimoto e Kimura (1995) e Chaumette e Hutchinson (2007). Chroust, et al. (2001) descrevem que técnicas de controle adicional utilizando *feedforward* são vantajosas em relação a outros tipos de técnicas quando as alterações no movimento do alvo é considerado suave.

Outros métodos que aprimoram o filtro de Kalman clássico também podem ser encontrados na literatura, como o modelo adaptativo (AKF – do inglês *adaptive Kalman Filter*) de Wira e Urban (2000). Wira e Urban (2000) inicialmente ressaltam que o modelo extendido do filtro de Kalman (EKF – do inglês *extended Kalman Filter*) é utilizado para realizar um processo de linearização. Tanto no filtro de Kalman tradicional assim como no EKF, os ruídos do processo e da observação são normalmente tratados como sendo constantes. Essa característica em muitos sistemas não acontece e os ruídos do processo e da observação sofrem alterações constantes. Sendo assim Wira e Urban (2000) desenvolveram um método AKF onde as matrizes responsáveis por ruídos de processo e observação são ajustadas

durante a execução do sistema. Nesse trabalho também é ressaltado que na etapa de predição do filtro de Kalman ocorrem o tratamento de *dead-times* e como o método adaptativo considera alterações em variáveis que estão relacionadas com essa etapa irá influenciar no resultado do tratamento de *dead-times*. De uma forma geral e comparado com as abordagens tradicionais do filtro de Kalman e o EKF, o modelo AKF apresentou bons resultados, sendo indicado pelos autores principalmente para tratar de problemas não lineares que não podem ser aproximados de modelos lineares ou quando o modelo do sistema é desconhecido.

O *Switching Kalman Filter* de Chroust e Vincze (2003) foi desenvolvido exclusivamente para tratar de problemas de *dead-times* em operações de rastreamento de objetos, onde esses objetos estão sujeitos a alterações de movimento. A técnica proposta utiliza um conjunto de outras técnicas que são o filtro de Kalman adaptativo, se for observado uma descontinuidade o método alterna para o filtro de Kalman *steady-state* que é considerado melhor para tratar problemas de descontinuidade e um controlador auxiliar para garantir que o movimento do objeto rastreado é contínuo. O *Switching Kalman Filter* se demonstrou melhor quando comparado as outras técnicas que compõe esse método individualmente.

Durante essa pesquisa um método para tratamento de *dead-times* que pareceu bastante interessante foi apresentado em Lutteke e Franke (2013). O método de Lutteke e Franke (2013) possui formulação matemática baseada diretamente no filtro de Kalman tradicional, é bastante simples e pode operar em conjunto ao filtro de Kalman. Pelas características apresentadas e também por ser um método desenvolvido recentemente, esse trabalho optou por realizar uma analise e aplicação dessa abordagem e detalhes são apresentados nos próximos capítulos.

Capítulo 4

PROPOSTA

Esse capítulo apresenta os aspectos relacionados com a proposta e condições analisadas, tais como questões do problema de *dead-times*, simulação em que a metodologia foi aplicada, materiais utilizados e a delimitação do escopo.

4.1 Análise do problema

Conforme demonstrado na Figura 4.1 a metodologia proposta deve tratar a ocorrência de *dead-times* que são causados por processamento de imagens e outras operações de visão computacional (antes de ser estabelecido o sinal a ser utilizado pelo filtro de Kalman) e também de *dead-times* causados durante a transmissão e processamento de dados no sistema (após ser estabelecido o sinal a ser utilizado pelo filtro de Kalman).

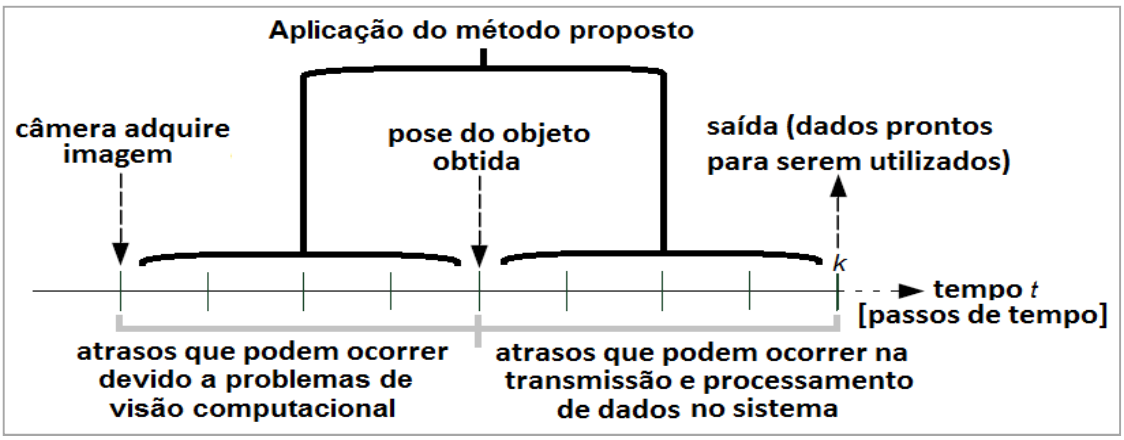


Figura 4.1 – Representação da área de aplicação da metodologia proposta.

Na Figura 4.2 (sem *dead-times*) e 4.3 (com *dead-times*) são considerados dois modelos de tempo para o sistema de controle desse trabalho. Ambas as Figuras contém duas linhas de tempo *k* e k_1. A linha mais clara (descrita como tempo *k*) representa o tempo de cada aquisição de imagem e preparo da informação (processamento de imagem, extração de características, filtragem, etc) até que essa possa ser utilizada no controlador de articulações. A linha mais escura (descrita como tempo k_1) representa a estimativa que está sendo utilizada em cada instante/amostra do sistema robótico.

A observação Z_{k-s} (etapa k antes do tempo necessário para a informação ficar disponível para o controlador de articulações denotado por *s*) é adquirida no sistema de exatamente no tempo *k*, cada uma unidade de *k* equivale a 3 unidades de k_1. Essa unidade de *k* (3 unidades de k_1) é a taxa de amostragem da "câmera" denotada pela letra *s* e também representa o tempo para informação filtrada ficar disponível para controlador robótico (extração e transferência de informações para o sistema de controle alimentar o robô). Durante o tempo *s* o controlador de articulações (responsável pelo movimento do robô) utiliza a estimativa calculada com base na informação anterior ajustada pelo filtro de Kalman.

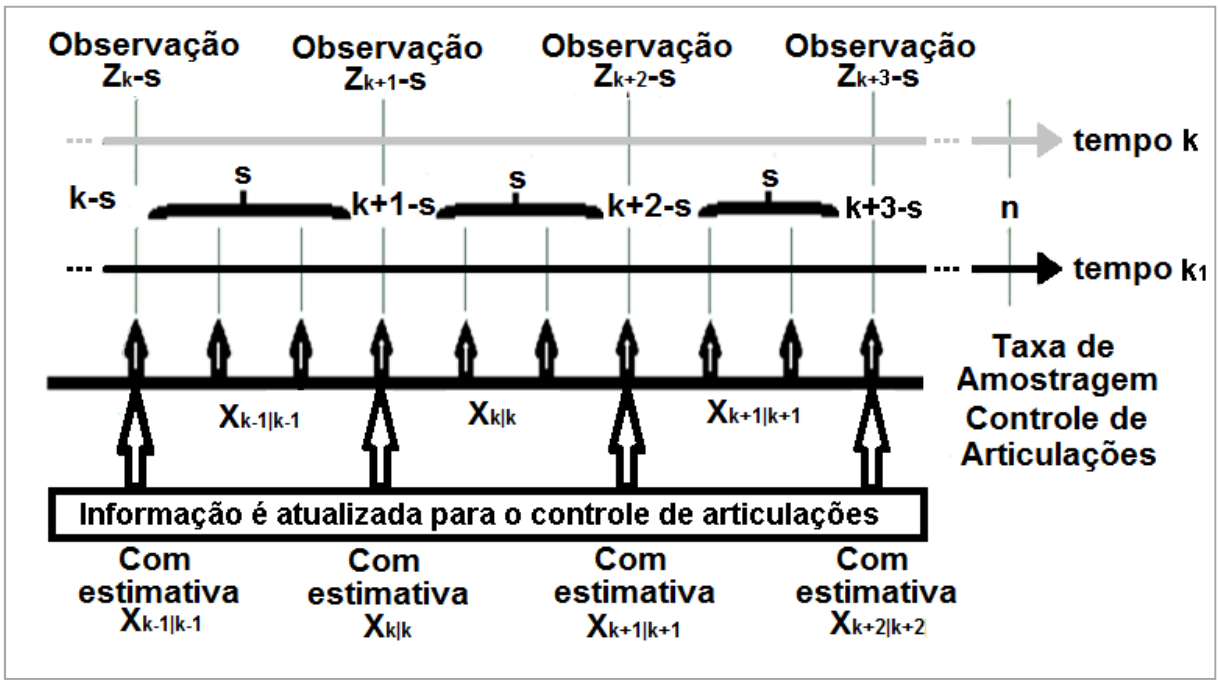
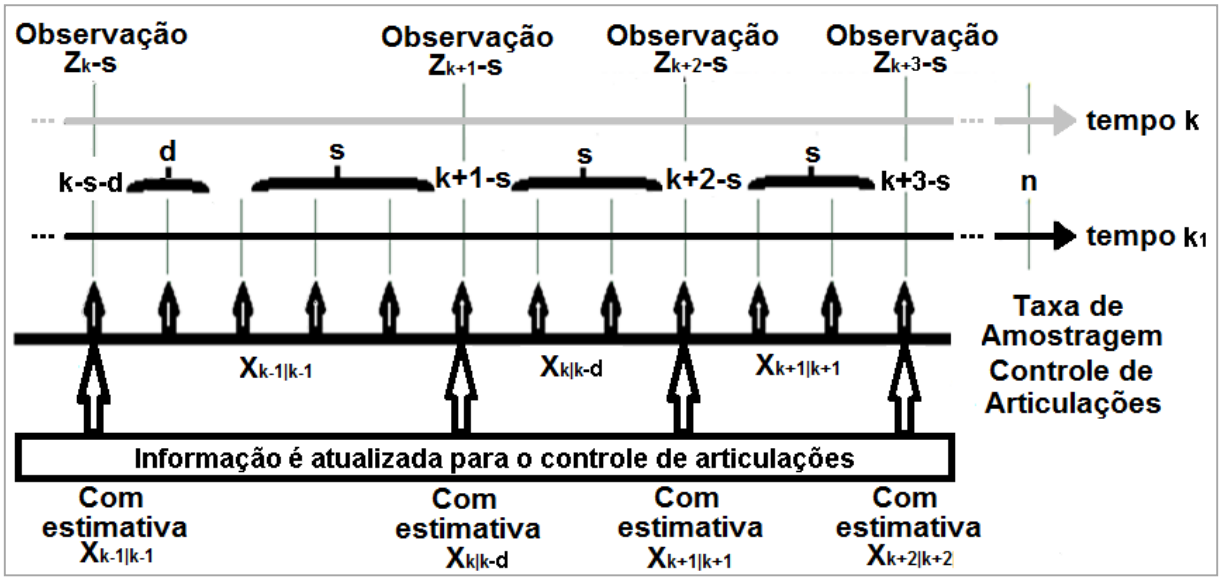


Figura 4.2 – Demonstração do funcionamento da alimentação do sistema de controle com estimativas realizadas por filtro de Kalman sem *dead-times*.

Na Figura 4.3 é considerado a ocorrência de 2 *dead-times* (causados nas etapas de processamento em visão computacional). Embora o filtro de Kalman seja capaz de ajustar as informações ao longo o tempo, no momento que ocorrem *dead-times,* a estimativa disponível no controle de articulações é utilizada mais tempo do que seria na "não ocorrência de *dead-times"*. Essa característica pode gerar um problema de instabilidade em uma operação de rastreamento de objetos onde o controlador de articulações é ajustado com base na velocidade necessária para que o robô se movimente (caso considerado nesse trabalho). O problema acontece devido ao filtro de Kalman ser uma ferramenta de raciocínio ao longo do tempo, ou seja, depende das informações anteriores que apresentavam um comportamento padrão e de repente sofreram uma alteração. Conforme o fluxo do filtro de Kalman apresentado nos capítulos anteriores, essa alteração (tempo adicional causado pelos *dead-times*) também não é considerada imediatamente na etapa posterior à etapa que ocorreram *dead-times* o que pode gerar mais instabilidade.



Fiura 4.3 – Demonstração do funcionamento da alimentação do sistema de controle com estimativas realizadas por filtro de Kalman com *dead-times.*

Considerando que as estimativas $X_{k|k-d}$ e $P_{k|k-d}$ estão disponíveis no sistema e a diferença entre *k* e *k-d*. Em um processo normal essa diferença é corrigida durante as etapas seguintes na execução do filtro Kalman. Os valores $X_{k|k-d}$ e $P_{k|k-d}$ podem ser utilizados para estimar e compensar os *dead-times* de $X_{k|k}$ e $P_{k|k,}$ sendo esses novos valores denotados por $X_{k|k}^*$ e $P_{k|k}^*$.

4.2 Materiais e métodos

4.2.1 Método de Lutteke e Franke (2013)

A partir das formulas do filtro de Kalman (equações de 12 a 16) estudadas no capítulo 2 e considerando o modelo de transição de estados *F*, o modelo de entradas de controle *B* e o vetor de entradas de controle *U* constantes, uma forma de calcular $X_{k|k}^*$ e $P_{k|k}^*$ é estudada em Lutteke e Franke (2013). Em Lutteke e Franke (2013) e nesse trabalho é considerado um filtro de Kalman aplicado a sistemas lineares, invariantes no tempo e com qualquer quantidade de *dead-times*. Também são considerados *F, B, Q, K, H, Z, R* e *U* constantes.

O estado e covariância preditos na ocorrência de *dead-times* são definidos respectivamente como $X_{k|k-1}^*$ e $P_{k|k-1}^*$. Essas variáveis são expressas em *d* estimativas passadas como descrito na equação 21 ($X_{k|k-1}^*$) e equação 22 ($P_{k|k-1}^*$).

$$X_{k|k-1}^* = F^d * (X_{k|k-1-d}) + F^{d-1} * B_k * U_k + F^{d-2} * B_{k-2} * U_{k-2} + \ldots + B_{k-1} * U_{k-1} \quad (21)$$

$$P_{k|k-1}^* = F^d * (P_{k|k-1k|k-1-d}) * F^{t\,d} + F^{d-1} * Q_k * F^{t\,d-1} + F^{d-2} * Q_k * F^{t\,d-2} + \ldots + Q \quad (22)$$

Considerando a equação 21 na primeira parte ($F^d * (X_{k|k-1-d})$) é demonstrado a matriz de transição de estado *F* elevada a *d* multiplicando o valor da previsão do estado antes da ocorrência dos *d dead-times.* Na segunda parte ($F^{d-1} * B_k * U_k + F^{d-2} * B_{k-2} * U_{k-2} + \ldots + B_{k-1} * U_{k-1}$) é possível observar um comportamento padronizado onde em cada *d* que está elevado sobre *F* e multiplicado por B_k e U_k é reduzido uma unidade até que o valor que está elevado seja igual á 0.

Agora com a equação 22 na primeira parte ($F^d * (P_{k|k-1} - d) * F^{t\,d}$) é demonstrado a matriz de transição de estado *F* e sua transposta F^t ambas elevadas ao valor de *d* e multiplicando o valor da previsão da covariância realizada antes da ocorrência dos *d dead-times.* Na segunda parte ($F^{d-1} * Q_k * F^{t\,d-1} + F^{d-2} * Q_k * F^{t\,d-2} + \ldots + Q$) da equação é possível observar um comportamento padronizado onde em cada *d* que está elevado sobre *F* e F^t multiplicando Q_k é reduzido uma unidade até que o valor que está elevado seja igual á 0.

O comportamento padronizado permite definir a segunda parte das equações 21 e 22 representadas como ΔU (equação 23) e ΔO (equação 24) respectivamente.

$$\Delta U = F^{d-1} * B_k * U_k + F^{d-2} * B_{k-2} * U_{k-2} + \ldots + B_{k-1} * U_{k-1} \quad (23)$$

$$\Delta O = F^{d-1} * Q_k * F^{td-1} + F^{d-2} * Q_k * F^{td-2} + \ldots + Q \quad (24)$$

Substituindo a equação 23 na equação 21 e a equação 24 na equação 22 são obtidos respectivamente as equação 25 e 26. As variáveis ΔU (equação 25) e ΔO (equação 26) são importantes no desenvolvimento do método estudado.

$$X_{k|k-1}^* = F^d * (X_{k|k-1-d}) + \Delta U \quad (25)$$

$$P_{k|k-1}^* = F^d * (P_{k|k-1-d}) * F^{t\,d} + \Delta O \quad (26)$$

Com as equação 25 e 26 é possível isolar ΔU e ΔO chegando a equação 27 e 28.

$$\Delta U = X_{k|k-1}^* - F^d * (X_{k|k-1-d}) \quad (27)$$

$$\Delta O = P_{k|k-1}^* - F^d * (P_{k|k-1-d}) * F^{t\,d} \quad (28)$$

Alterando as predições do estado da equação 25 e da covariância da equação 26 (baseadas em informações a-priori) para trabalhar com o valor das estimativas (informações a-posteriori) obtém-se duas novas variáveis $X_{k|k}^*$ e $P_{k|k}^*$ que são representadas pelas equações 29 e 30.

$$X_{k|k}^* = F^d * (X_{k|k-d}) + \Delta U \quad (29)$$

$$P_{k|k}^* = F^d * (P_{k|k-d}) * F^{t\,d} + \Delta O \quad (30)$$

Nas equações 29 e 30 é observado que ΔU e ΔO estão presentes e dessa forma podem ser substituídas pelas equações 27 e 28 obtendo-se as equações 31 e 32.

$$X_{k|k}* = F^{d} * (X_{k|k-d}) + X_{k|k-1}* - F^{d} * (X_{k|k-1-d}) \quad (31)$$

$$P_{k|k}* = F^{d} * (P_{k|k-d}) * F^{t\,d} + P_{k|k-1}* - F^{d} * (P_{k|k-1-d}) * F^{t\,d} \quad (32)$$

Para finalizar essas equações podem ter algumas variáveis isoladas de forma a chegar as equações 33 e 34.

$$X_{k|k}* = X_{k|k-1}* + F^{d} * (X_{k|k-d} - X_{k|k-1-d}) \quad (33)$$

$$P_{k|k}* = P_{k|k-1}* + F^{d} * (P_{k|k-d} - P_{k|k-1-d}) * F^{t\,d} \quad (34)$$

Uma observação importante é que as operações matriciais de multiplicações relacionadas com F e F^t elevados a d tem um alto custo computacional, porém essas informações podem ser previamente armazenadas para serem utilizadas durante a execução do sistema.

Em Lutteke e Franke (2013) baseado no filtro de Kalman e na ocorrência de *d dead-times* os valores de previsão $X_{k|k-1-d}$ e $P_{k|k-1-d}$ e estimativas $X_{k|k-d}$ e $P_{k|k-d}$ (*-d* indica antes da ocorrência de *d dead-times*) podem ser utilizados para alimentar o sistema. Sobre as mesmas condições do trabalho de Lutteke e Franke (2013), são utilizados apenas os valores de estimativa $X_{k|k-d}$ e $P_{k|k-d}$ para alimentar o sistema. Por exemplo, no caso de estimativa de pose futura do objeto $X_{k-2|k-2}$ é alimentado para o sistema no passo de tempo *k-1*, $X_{k-1|k-1}$ é alimentado para o sistema no passo de tempo *k*, e assim por diante...

4.2.1.1 Método alternativo ao método de Lutteke e Franke (2013)

Baseado no contexto apresentado foi utilizada uma forma diferente ao estudo de Lutteke e Franke (2013) para desenvolver uma fórmula matemática que represente a estimativa de $X_{k|k}*$ e $P_{k|k}*$ tratando a ocorrência de *dead-times*. A fórmula encontrada foi semelhante a formula de Lutteke e Franke (2013), porém com

um detalhe diferente, algumas variáveis adicionais nas equações finais foram obtidas nesse trabalho (influenciam pouco no resultado final dependendo da aplicação) e no trabalho de Lutteke e Franke (2013) não foram consideradas.

Através do filtro de Kalman estudado no capítulo 2 quando é estimado um valor o processo executado é representado pela equação 14 para estimativa do estado $X_{k|k}$ e pela equação 16 para estimativa da covariância do processo $P_{k|k}$. Substituindo y_k na equação 14 é possível obter a equação 35.

$$X_{k|k} = X_{k|k-1} + K * (Z - (H * X_{k|k-1})) \tag{35}$$

e na equação 16 substituindo S_k é obtido a equação 36

$$P_{k|k} = P_{k|k-1} - K * (H * P_{k|k-1} * H^T + R) * K^T \tag{36}$$

O recurso de substituir algumas variáveis por equações que essas variáveis representam é amplamente utilizado na formulação das equações desse trabalho.

As equações 35 e 36 indicam que a estimativa de estado $X_{k|k}$ e da covariância do processo $P_{k|k}$ são obtidas respectivamente com base nas predições $X_{k|k-1}$ e $P_{k|k-1}$ e estão á um passo a frente (por exemplo, de *k* para *k+1*) da estimativa de estado $X_{k-1|k-1}$ e estimativa da covariância do processo $P_{k-1|k-1}$.

Utilizando o raciocínio de estimar a próxima situação do estado, juntos as equações de 12 a 18 do filtro de Kalman e um processo recursivo é possível estimar o número de passos a frente que forem necessários, como por exemplo, 2 passos a frente, ou seja, $X_{k+2|k+2}$ e $P_{k+2|k+2}$. Para estimar $X_{k+2|k+2}$ e $P_{k+2|k+2}$ são necessários 2 execuções de predições e 2 execuções de estimativas.

Primeiramente são demonstradas as equações do filtro de Kalman para estimativas de estado $X_{k+1|k+1}$ (equação 37) e da covariância $P_{k+1|k+1}$ (equação 38).

$$X_{k+1|k+1} = (F * X_{k|k} + B * U) + K * (Z - (H * (F * X_{k|k} + B * U))) \tag{37}$$

$$P_{k+1|k+1} = (F * P_{k|k} * F^T + Q) - K * (H * (F * P_{k|k} * F^T + Q) * H^T + R) * K^T \tag{38}$$

Com as equações para as primeiras estimativas é possível obter as equações do filtro de Kalman para estimativas de $X_{k+2|k+2}$ (equação 39) e $P_{k+2|k+2}$ (equação 40).

$$X_{k+2|k+2} = (F * ((F * X_{k|k} + B * U) + K * (Z - (H * (F * X_{k|k} + B * U)))) + B * U) + K * (Z - (H * (F * ((F * X_{k|k} + B * U) + K * (Z - (H * (F * X_{k|k} + B * U)))) + B * U))) \quad (39)$$

$$P_{k+2|k+2} = (F * ((F * P_{k|k} * F^T + Q) - K * (H * (F * P_{k|k} * F^T + Q) * H^T + R) * K^T) * F^T + Q) - K * (H * (F * ((F * P_{k|k} * F^T + Q) - K * (H * (F * P_{k|k} * F^T + Q) * H^T + R) * K^T) * F^T + Q) * H^T + R) * K^T \quad (40)$$

Isolando alguns valores nas equações 39 e 40 são aplicadas as distribuitivas, definindo duas variáveis representadas pelas equações 41 e 42.

$$\Delta U = F * B * U + B * U \quad (41)$$

$$\Delta O = F * Q * F^T + Q \quad (42)$$

Essas variáveis são aplicadas ás equações 39 e 40, gerando as equações 43 e 44.

$$X_{k+2|k+2} = (F * F * X_{k|k} + \Delta U) + (F * K * Z - F * H * F * X_{k|k} - F * K * H * B * U + K * Z - K * H * F * F * X_{k|k} - K * H * F * B * U - K * H * F * K * Z + K * H * H * F * X_{k|k} + K * H * F * K * H * B * U - K * H * B * U) \quad (43)$$

$$P_{k+2|k+2} = (F * F * P_{k|k} * F^T * F^T + \Delta O) + (- F * K * H * F * P_{k|k} * F^T * H^T * K^T * F^T - F * K * H * Q * H^T * K^T * F^T - F * K * R * K^T * F^T - K * H * F * F * P_{k|k} * F^T * F^T * H^T * K^T - K * H * F * Q * F^T * K^T * H^T + K * H * F * K * H * F * P_{k|k} * F^T * H^T * K^T * F^T * H^T * K^T + K * H * F * K * H * Q * H^T * K^T * F^T * H^T * K^T + K * H * F * K * R * K^T * F^T * H^T * K^T - K * H * Q * H^T * K^T - K * R * K^T) \quad (44)$$

Com essas novas equações já é possível notar certa semelhança com as equações de Lutteke e Franke (2013). Durante o desenvolvimento das estimativas $X_{k+2|k+2}$ e $P_{k+2|k+2,}$ em algum momento, foi necessário passar por equações que determinassem as predições $X_{k+2|k+1}$ e $P_{k+2|k+1}$ (o desenvolvimento dessas equações

pode ser encontrado no apêndice A). Utilizando as equações que determinam as predições $X_{k+2|k+1}$ e $P_{k+2|k+1}$ e as variáveis ΔU e ΔO são obtidas as 45 e 46.

$$X_{k+2|k+1} = (F * F * X_{k|k-1} + \Delta U) + (F * F * K * Z - F * K * F * H * X_{k|k-1} + F * K * Z - F * K * H * F * X_{k|k-1} - F * K * H * F * K * Z + F * K * H * F * K * H * X_{k|k-1} - F * K * H * B * U) \quad (45)$$

$$P_{k+2|k+1} = (F * F * P_{k|k-1} F^T * F^T + \Delta O) + (- F * F * K * H * P_{k|k-1} * H^T * K^T F^T * F^T - F * F * K * R * K^T F^T * F^T - F * K * H * F * P_{k|k-1} F^T * H^T * K^T * F^T + F * K * H * F * K * H * P_{k|k-1} * H^T * K^T F^T * H^T * K^T * F^T + F * K * H * F * K * R * K^T F^T * H^T * K^T * F^T - F * K * H * Q * H^T * K^T * F^T - F * K * R * K^T * F^T) \quad (46)$$

Isolando as variáveis ΔU e ΔO nas equações 45 e 46 e substituindo as equações obtidas em 43 e 44 são geradas as equações 47 e 48

$$X_{k+2|k+2} = X_{k+2|k+1} + F^2 * (X_{k|k} - X_{k|k-1}) + L \quad (47)$$

$$P_{k+2|k+2} = P_{k+2|k+1} + F^2 * (P_{k|k} - P_{k|k-1}) * F^{T2} + M \quad (48)$$

A variável *L* e a variável *M* representadas respectivamente pelas equações 49 e 50 são as diferenças em relação a formulação de Lutteke e Franke (2013).

$$L = - (F * F * K * Z) + (F * F * K * H * X_{k|k-1}) + (F * F * K * H * X_{k|k-1}) + (F * F * K * K * H * Z) - (F * F * K * K * H * H * X_{k|k-1}) - (F * H * H_x * X_{k|k}) + (K * Z) - (F * F * K * H * X_{k|k}) - (F * K * H * B * U) - (F * K * K * H * Z) + (F * K * H * H * X_{k|k}) + (F * H * K * K * H * B * U) - (K * H * B * U) \quad (49)$$

$$M = (F * F * K * H * P_{k|k-1} * H^T * K^T * F^T * F^T) + (F * F * K * R * K^T * F^T * F^T) + (F * F * K * H * P_{k|k-1} * H^T * K^T * F^T * F^T) - (F * F * K * K * H * H * P_{k|k-1} * H^T * H^T * K^T * K^T * F^T * F^T) - (F * F * K * K * H * R * H^T * K^T * K^T * F^T * F^T) - (F * F * K * H * P_{k|k} * K^T * H^T * F^T * F^T) - (F * F * K * H * P_{k|k} * H^T * K^T * F^T * F^T) - (F * K * H * Q * K^T * H^T * F^T) + (F * F * K * K * H * H * P_{k|k} * H^T * H^T * K^T * K^T * F^T * F^T) + (F * K * K * H * H * Q * H^T * H^T * K^T * K^T * F^T) + (F * K * K * H * R * H^T * K^T * K^T * F^T) - (K * H * Q * H^T * K^T) - (K * R * K^T) \quad (50)$$

No caso foram consideradas duas etapas a frente de $X_{k|k}$ e $P_{k|k}$, porém essas equações podem ser descritas em termos de n etapas. Como demonstrado nas equações 51 e 52.

$$X_{k+n|k+n} = X_{k+n|k+n-1} + F^n * (X_{k|k} - X_{k|k-1}) + L \quad (51)$$

$$P_{k+n|k+n} = P_{k+n|k+n-1} + F^n * (P_{k|k} - P_{k|k-1}) * F^{t\ n} + M \quad (52)$$

Considerando as amostras de tamanhos fixos (taxa de amostragem *s* sem variações) e também os *d dead-times*, uma ocorrência de *dead-time* é equivalente a uma amostra do sistema. Dessa forma a partir das equações 51 e 52 podem ser convertidas para considerar *d dead-times*, e finalmente duas novas equações (equação 53 e 54) são estabelecidas para estimar de forma mais correta os valores de $X_{k|k}$* e $P_{k|k}$*.

$$X_{k|k}^* = X_{k|k-1}^* + F^d * (X_{k|k-d} - X_{k|k-1-d}) + L \quad (53)$$

$$P_{k|k}^* = P_{k|k-1}^* + F^d * (P_{k|k-d} - P_{k|k-1-d}) * F^{t\ d} + M \quad (54)$$

$X_{k|k-1}$* e $P_{k|k-1}$* precisam ser estimados e uma forma de fazer isso é através da equação 55 para $X_{k|k-1}$*

$$X_{k|k-1}^* = F^d * (X_{k+1|k-d}) + \Delta U \quad (55)$$

e a equação 56 para $P_{k|k-1}$*

$$P_{k|k-1}^* = F^d * (P_{k+1|k-d}) * F^{T\ d} + \Delta O \quad (56)$$

Durante os testes realizados nesse trabalho foi notado que a variável *L* (e consequentemente *M*) costuma assumir valores extremamente pequenos (menores que a escala nanométrica) e talvez seja esse o motivo de que no trabalho de Lutteke e Franke (2013) não tenham sido consideradas. Porém se de alguma forma existir a

necessidade de utilização dessa medida esse valor necessita ser estimado. Dessa forma nesse trabalho foram utilizadas as mesmas fórmulas de Lutteke e Franke (2013), que foram apresentadas nesse trabalho como as equações 33 e 34.

4.2.2 Ferramentas utilizadas

No *website* da ferramenta VISP (LAGADIC, 2013) existem vários exemplos de aplicações desenvolvidas com possibilidade de alteração nos códigos fontes. A ferramenta VISP possui licença gratuita, código fonte aberto, é desenvolvida com a linguagem C++ e possui recursos para trabalhar com ferramentas de desenvolvimento como no caso desse trabalho o Visual Studio 2012. Outra questão é que no *website* da ferramenta pode ser encontrados tutoriais para instalação da ferramenta além de *links* para ferramentas de terceiros que podem ser instaladas e utilizadas. Uma importante ferramenta de terceiro utilizada nesse trabalho é o Opencv que possui uma licença *open source.*

Como previamente descrito a ferramenta VISP permite a utilização de funções e módulos já pré-desenvolvidos como o robô, a câmera, o objeto, a localização da câmera no robô, partes que compõe o sistema de controle, o tipo de controle de servovisão utilizado, entre outros. Uma vantagem dos módulos desenvolvidos na ferramenta VISP (como toda parte de servovisão, robótica e filtro de Kalman) é que foram baseados em diversos trabalhos que podem ser facilmente encontrados na literatura como Hutchinson, Hager e Corke (1996), Chaumette e Hutchinson (2006), Chaumette e Hutchinson (2007), Corke e Good (1993), Bar-Shalom, Li e Kirubarajan (2001), entre outros. Com todas as características apresentadas os simuladores utilizados nesse trabalho foram construídos com o auxílio da ferramenta VISP.

4.3 Proposta

Esse trabalho apresenta a utilização de uma metodologia para tratamento de *dead-times* aplicado em operações de rastreamento de objetos, onde os testes foram realizados em simuladores. Os estudos, análises e testes devem possibilitar a

conclusão se a metodologia pode ser aplicado em uma situação de rastreamento de objetos em movimento sobre esteiras transportadoras utilizando servovisão.

A metodologia utilizada nesse trabalho é constituida da abordagem de Lutteke e Franke (2013) em conjunto com o filtro de Kalman para sistemas de controle por servovisão da ferramenta VISP (apresentada no capítulo 2) que foi baseada nos trabalhos de Corke e Good (1993) e Bar-Shalom, Li e Kirubarajan (2001).

Nesse trabalho é adotado a utilização do filtro de Kalman tradicional (apresentado no capitulo 2) que é capaz de realizar uma filtragem em sinais utilizados no rastreamento de objetos por servovisão. A utilização do filtro de Kalman não-linear também pode ser encontrada na literatura, porém nesse trabalho adotamos o modelo linear uma vez que modelos não-lineares com o filtro de Kalman são aproximados (através de técnicas que podem ser encontradas na literatura) a modelos lineares para serem utilizados. Outra questão é que diversos trabalhos utilizados como base utilizam modelos lineares e de acordo com Ogata (2003) esse modelo é importante para tratar do problema com aproximações de situações reais.

A metodologia é utilizada em servovisão IBVS e aplicada em 3 experimentos:

1- Estimativa da pose futura de um objeto em movimento.
2- Em um controlador de velocidades em operações de rastreamento de objetos em movimento com a velocidade do manipulador utilizada como sinal a ser filtrado pelo filtro de Kalman.
3- Em um controlador de velocidades em operações de rastreamento de objetos em movimento com a pose do objeto utilizada como sinal a ser filtrado pelo filtro de Kalman.

O rastreamento sempre é considerado de forma planar (eixos x e y) e o movimento do objeto nas operações de rastreamento ocorre de forma linear ou circular que é comum em esteiras transportadoras e também zig-zag para analisar a alteração do sentido do movimento. Em todos os casos o objeto se movimenta com velocidade constante, mas nos casos dos controladores de velocidades, a velocidade do manipulador sofre alterações para se aproximar do objeto.

No capítulo de experimentos, testes e resultados são demonstrados os resultados da abordagem permitindo realizar conclusões em relação ao método.

4.4 Delimitação do escopo

O presente trabalho apresenta um estudo e avaliação de um método para tratar problemas de *dead-times* em operações de servovisão aplicado em objetos em movimento sobre esteiras transportadoras. Para isso um simulador é desenvolvido para um sistema de servovisão considerando o problema de *dead-times*.

Embora na simulação todo o sistema de servovisão foi considerado nessa pesquisa são levados em consideração apenas os *dead-times* desconsiderando outros problemas comuns em servovisão e outras abordagens adicionais como:

- aquisição e extração de características da imagem;
- calibração da câmera;
- configurações da câmera;
- reconstrução 3D;
- detalhes sobre algoritmos adicionais utilizados ou que podem ser encontrados em outros trabalhos;
- outros problemas que fazem parte do estudo de servovisão;

O único evento inesperado que é considerado nesse trabalho é a ocorrência de *dead-times*, demais eventos inesperados são desconsiderados e uma revisão sobre os demais tópicos pode ser encontrada em Corke (2011), Hutchinson, Hager e Corke (1996), Chaumette e Hutchinson (2006) e Chaumette e Hutchinson (2007).

Capítulo 5

APLICAÇÃO DA METODOLOGIA E SIMULADORES

Conforme descrito anteriormente o objetivo desse trabalho é propor uma metodologia para o tratamento de *dead-times* em servovisão quando aplicada sobre esteiras transportadoras (ambientes industriais). A abordagem estudada foi baseada no trabalho de Lutteke e Franke (2013). Nesse capítulo são apresentados questões da aplicação da metodologia e do desenvolvimento dos simuladores utilizados.

5.1 Aplicação da metodologia proposta

Após a execução do cálculo/abordagem proposta as estimativas $X_{k|k}$ e $P_{k|k}$ e as predições $X_{k|k-1}$ e $P_{k|k-1,}$ respectivamente passam a ser iguais á $X_{k|k}$*, $P_{k|k}$*, $X_{k|k-1}$* e $P_{k|k-1}$*. Também é necessário a execução de um processo de predição no sistema para o próximo estado em relação aos valores apresentados. Esse processo é necessário para deixar o filtro de Kalman sincronizado com os novos valores.

Nas simulações desse trabalho é considerado como taxa de amostragem da camera (*s*) como o tempo em que a imagem é adquirida e a informação (observação) fica disponível para o controlador de articulações. Nesse trabalho é esperado que cada amostra seja equivalente a taxa de amostragem da imagem *s*, porém na ocorrência de *dead-times* isso não acontece.

A taxa de amostragem da câmera *s* foi estabelecida de acordo com uma média utilizando pesos como demonstrado na equação 57. Esse recurso faz com

que seja considerado o menor tempo necessário (em média) para que o sistema opere e os valores excessivos são considerados como *dead-times*.

s é calculada antes da etapa de aquisição da imagem pela câmera conforme representado na Figura 5.1. A variável *n_ocorrencias* é o número de ocorrências que *s* (estabelecido no processo) foi utilizada, *s_anterior* é igual á *s* antes do acréscimo da nova variação de tempo e *dif_tempo_atual_anterior* é a nova variação de tempo. *dif_tempo_atual_anterior* é a variação de tempo referente a última etapa antes da aquisição da imagem pela câmera e etapa atual.

$$s = \frac{n\_ocorrencias * s\_anterior + dif\_tempo\_atual\_anterior}{1 + n\_ocorrencias} \quad (57)$$

Os *d dead-times* são calculados quando ocorre uma variação de tempo em relação a taxa de amostragem da imagem *s*. A ocorrência de *d dead-times* e o cálculo de suas variáveis auxiliares são verificados antes do valor estabelecido pela lei de controle (IBVS) ser alimentado no sistema (conforme a Figura 5.1).

Para calcular *d* é utilizada a equação 58, que indica *dif_tempo_dead_times_atual_anterior* menos a taxa de amostragem da imagem *s*. *dif_tempo_dead_times_atual_anterior* é variação de tempo referente a última etapa antes do sistema ser alimentado pela lei de controle e etapa atual.

$$d = dif\_tempo\_dead\_times\_atual\_anterior - s \quad (58)$$

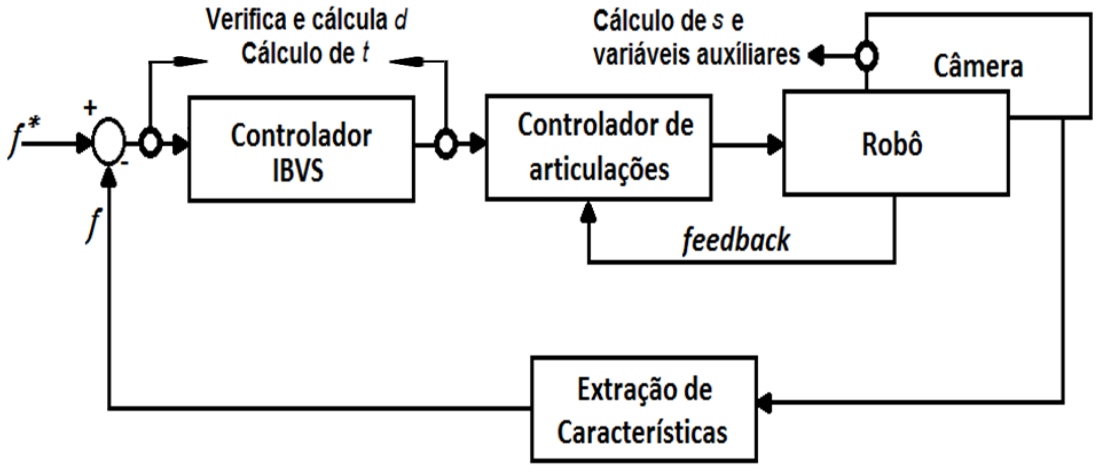


Figura 5.1 – Diagrama de blocos do sistema IBVS demonstrando os locais de cálculo de *s* e *d* e suas variáveis auxiliares.

Quando não ocorrem *d dead-times*, *dif_tempo_dead_times_atual_anterior* e *s* são valores muito próximos (em sistemas invariantes no tempo são valores iguais). Na ocorrência de *dead-times dif_tempo_dead_times_atual_anterio*r é maior que *s*.

Especificamente nesse trabalho com a regra do cálculo para taxa de amostragem da imagem *s*, durante os testes do sistema, *s* variou em aproximadamente 50 milissegundos. Eventualmente quando atrasos (*dead-times*) eram introduzidos, se o tempo atingi-se um valor múltiplo de 50 milissegundos maior que 1, ou seja, múltiplo 2 com 100 milissegundos, múltiplo 3 com 150 segundos, a regra com cálculo de *d* era disparada com *d=1* para 100 milissegundos, *d=2* para 150 milissegundos, ... Quando não ocorriam *dead-times* o sistema era executado normalmente utilizando o filtro de Kalman padrão.

A Tabela 5.1 representa um exemplo das etapas com os resultados da execução do sistema em algumas iterações no cálculo de *s*.

- Na iteração *n_ocorrencias = 1* a taxa de amostragem da câmera *s* e os *d dead-times* não são calculados, pois dependem do valor de *dif_tempo_atual_anterior* que é baseada no cálculo da iteração no tempo anterior (que na primeira iteração não existe) e atual.
- Na iteração *n_ocorrencias = 2* foi considerado *s = dif_tempo_atual_anterior*, pois ainda não existe média com pesos e foi utilizado o valor 1 como amostragem padrão.
- Na iteração *n_ocorrencias = 3* as equações 57 e 58 foram utilizadas, porém como não ocorreu atrasos não foi necessário a execução do método para tratar *dead-times*.
- Na iteração *n_ocorrencias = 4* equações 57 e 58 foram utilizadas e *s* é calculado normalmente, porém nessa equação ocorrem *d dead-times* o que faz com que o método de tratamento de *dead-times* (equações 33 e 34 Lutteke e Franke (2013)) seja executado. Nessa etapa também é executado um processo que compensa *dif_tempo_atual_anterior* acrescentando o tempo utilizado que gerou *d* ao tempo da última etapa antes da aquisição da imagem pela câmera.
- Na iteração *n_ocorrencias = 5* as equações 57 e 58 foram utilizadas, porém como não ocorreu atrasos não foi necessário a execução do

método para compensar *dead-times.* Na iteração 5 o tempo de *dif_tempo_atual_anterior* ocorre corretamente devido ao processo de compensação desse tempo em relação a *d* ter sido executado na iteração 4.

Tabela 5.1 – Exemplo da execução de processos de iterações do sistema com taxa de amostragem *s* e de *d dead-times* com descrições dos processos.

n_ocorrencias	Processo de *s*	Processo de *d*
1	*s* não é utilizado	*d* não é utilizado
2	*s = dif_tempo_atual_anterior = 1* onde *n_ocorrencias = 2,* *dif_tempo_atual_anterior = 1*	*d = 0,* não é utilizado onde *t=1*
3	*s = (3 * 1 + 1) / (1 + 3) → s = 1* onde *n_ocorrencias = 3,* *s_anterior = 1,* e *dif_tempo_atual_anterior = 1*	*t-s=0 → d = 0* onde *t=1,* não é utilizado
4	*s = (4 * 1 + 1) / (1 + 4) → s = 1* onde *n_ocorrencias = 4,* *s_anterior = 1,* e *dif_tempo_atual_anterior = 1*	*t-s=2 → d = 2* onde *t=3,* é utilizado, executa processo de compensar *s_anterior*
5	*s = (5 * 1 + 1) / (1 + 5) → s = 1* onde *n_ocorrencias = 5,* *s_anterior = 1, e* *dif_tempo_atual_anterior = 1*	*...* continua a execução *...*

Uma observação importante é que quando os *dead-times* são tratados o tempo anterior é atualizado para não causar problemas nas ocorrências seguintes.

5.2 Simuladores

Um dos exemplos de arquivos da ferramenta VISP possuí o título de “testKalmanVelocity.cpp” e foi utilizado para o desenvolvimento da aplicação para testar a operação de previsão futura da posição de um objeto em movimento. Nessa

aplicação as variáveis utilizadas no filtro de Kalman foram especificas para estimativa de posição futura de objetos, recurso permitido pela ferramenta VISP no arquivo “vpLinearKalmanFilterInstantiation.cpp” sendo a função com o nome de “initStateConstVel_MeasurePos” responsável por esse recurso.

Os simuladores para o rastreamento de objetos em movimento através de um controlador de velocidades foram desenvolvidos utilizando diversos arquivos com código fonte, o arquivo “servoSimuFourPoints2DPolarCamVelocityDisplay.cpp” foi o principal nesses simuladores e foi modificado para funcionar com filtro de Kalman, controlador de velocidades e o robô AFMA-6. Nessa aplicação as variáveis utilizadas no filtro de Kalman foram específicas para o controlador de velocidade do robô na operação de rastreamento de objetos por servovisão, recurso permitido pela ferramenta VISP (arquivo “vpLinearKalmanFilterInstantiation.cpp”) sendo a função “initStateConstVelWithColoredNoise_MeasureVel” responsável por esse recurso.

As equações 33 e 34 utilizadas como o método para tratar *dead-times* foram implementadas na forma de uma função como título de “*delay_correction*” no arquivo “vpLinearKalmanFilterInstantiation.cpp” com algumas variáveis também definidas nos arquivos “vpLinearKalmanFilterInstantiation.h” e “vpKalmanFilter.h”. Essa função recebe a quantidade de *d dead-times* e ajusta de acordo com as equações 33 e 34 do método proposto para tratar *dead-times.* O fluxograma que representa essa função é apresentado na Figura 5.2.

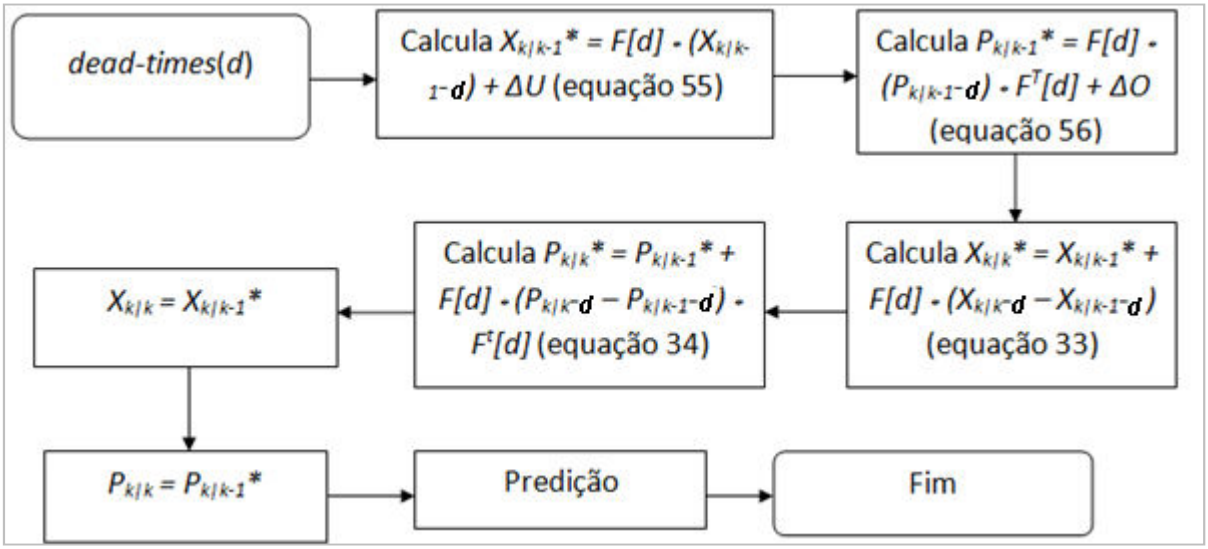


Figura 5.2 – Fluxograma de representação do processo para tratamento de *dead-times* utilizado nesse trabalho.

Nesse fluxograma é possível notar que F e F^t foram utilizados na forma de vetores por estarem previamente armazenados no sistema. Esse recurso foi utilizado nesse trabalho para reduzir a necessidade de equações matriciais no sistema, o que proporciona na melhoria do desempenho do sistema. O processo para armazenar os valores pré-definidos de cada quantidade de *d dead-times* que podem ocorrer durante a execução do sistema foi colocado junto a função de inicialização do filtro de Kalman no sistema no arquivo "vpKalmanFilter.cpp". No final do fluxograma apresentado também tem a chamada da função da predição que é utilizada para corrigir a etapa (avançar até a nova estimativa) do filtro de Kalman.

5.2.1 Simulador para prever a pose futura de um objeto em movimento

Esse simulador foi desenvolvido com propósito de prever a pose futura de um objeto em movimento através de um único ponto (pixel). Em seu desenvolvimento foram utilizados o filtro de Kalman da ferramenta VISP e o método para compensar *dead-times* apresentado nesse trabalho.

A simulação (conforme a Figura 5.3) acontece com o objeto se movimentando de acordo com um dos três tipos de movimentos (não simultâneos): diagonal/linear, circular ou zig-zag. O objetivo do sistema é prever a próxima posição do objeto utilizando o filtro de Kalman e na ocorrência de *dead-times* tratar essa situação.

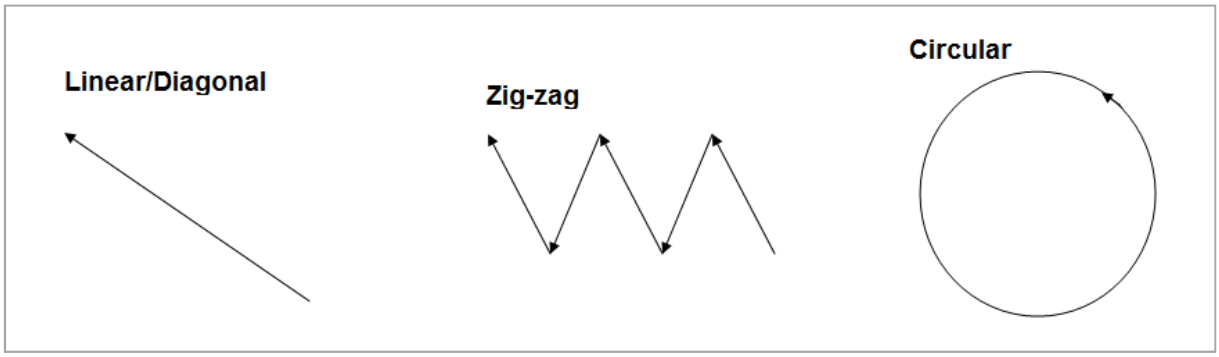


Figura 5.3 – Representação das movimentações do objeto na simulação

A movimentação linear e a circular são comuns em esteiras transportadoras em ambientes industriais. A movimentação circular também permite testar se o método é capaz de rastrear objetos que realizam curvas. A movimentação em zig-zag permite considerar alterações drásticas no movimento do objeto.

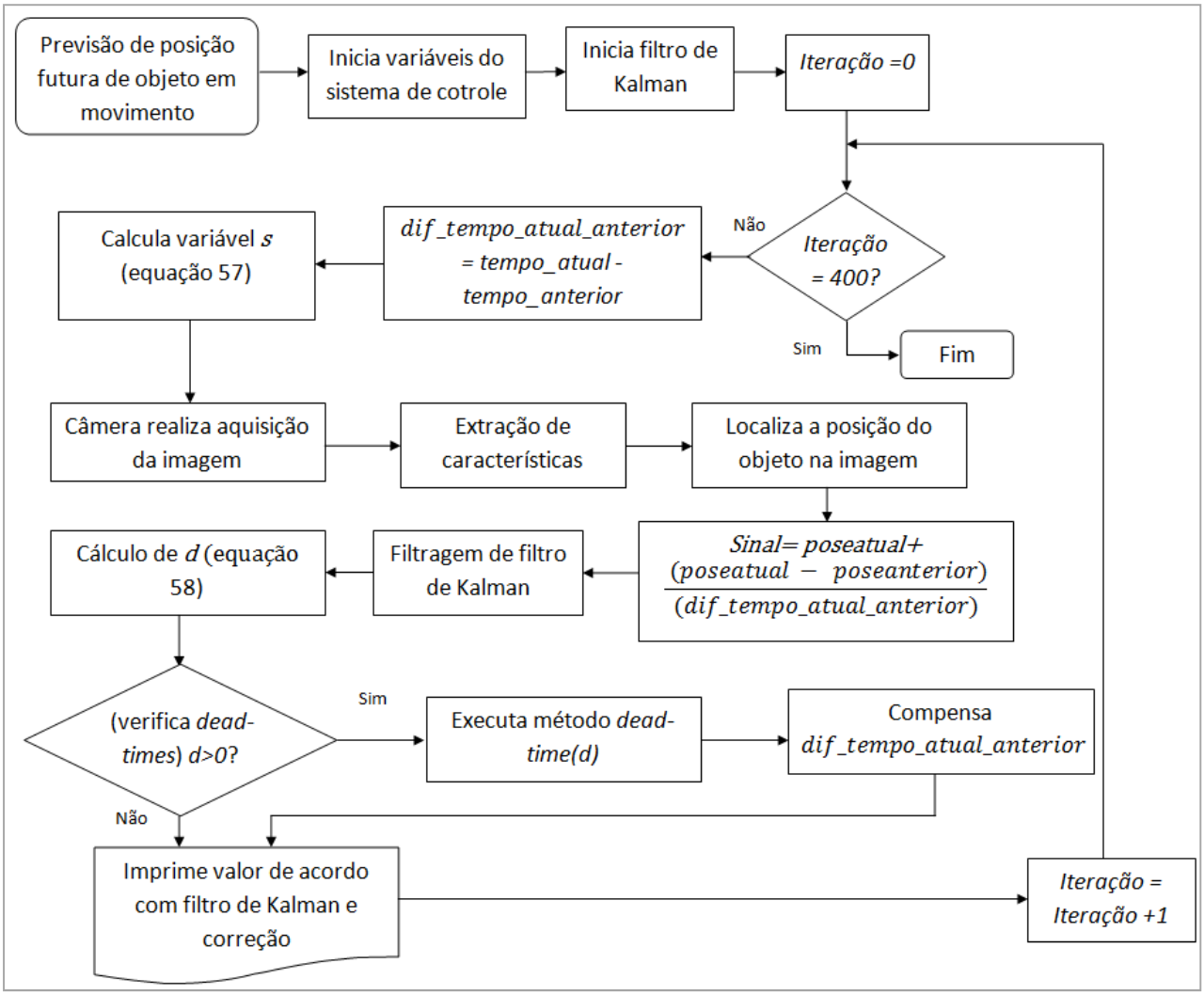


Figura 5.4 – Fluxograma do processo de IBVS com filtro de Kalman e método para *dead-times* na previsão da pose futura de objeto em movimento.

A Figura 5.4 representa o fluxograma do processo do sistema de controle com o filtro de Kalman e o método para tratar *dead-times.* O fluxograma representa o caso da previsão da pose futura de um objeto em movimento para um sistema IBVS.

A parte de maior interesse desse trabalho no fluxograma é a partir do momento onde o sistema entra em recursividade, pois antes disso apenas são definidas algumas variáveis do sistema de controle e do filtro de Kalman. Nessa recursão são definidas 400 iterações até que o programa pare de ser executado.

No início do processo de recursividade é observado á equação *dif_tempo_atual_anterior* que representa a variação de tempo do processo. Em seguida o sistema passa pelo cálculo de tempo de *s* (equação 57). Nessa

representação foi omitida (por questão de espaço no trabalho) a iteração inicial onde não é possível obter *dif_tempo_atual_anterior* devido a falta do tempo anterior e consequentemente nas etapas que fazem uso de *dif_tempo_atual_anterior*, essa iteração no fluxograma é considerada como já tratada.

Um processo de estimativa de pose futura do objeto é considerado de acordo com o modelo IBVS. Com a pose do objeto é possível estabelecer o sinal que será utilizado no filtro de Kalman.

No cálculo do sinal pode ser observado a pose atual junto a adição da derivada de pose atual em relação a pose anterior do objeto, que sem adições de erros e incertezas pode ser considerada diretamente a pose futura do objeto em movimento. Em um sistema que o ambiente não provoca incertezas a adição dessa derivada seria suficiente, porém no caso de sistemas de reais, principalmente no caso de servovisão essa ação não é exata e sendo assim existe a necessidade do uso do filtro de Kalman. Os resultados da execução do filtro de Kalman quando esse valor não é exato, ou seja, podendo sofrer com incertezas do processo e variações causadas por ruídos, melhoram o desempenho do sistema. Outra questão é que a utilização do filtro de Kalman permite a utilização do método estudado, onde esse método é executado de acordo com suas equações.

Ainda no fluxograma da Figura 5.5, após a execução do filtro de Kalman, é executado a verificação de ocorrência de *dead-times* (equação 58), seguido se necessário pela execução do método proposto para compensar *dead-times*. Na execução do método proposto é também executado o método de compensação de *dif_tempo_atual_anterior* para permitir que o filtro de Kalman e o sistema operem normalmente, ou seja, como se não tivesse ocorrido alterações no tempo após a ocorrência de *dead-times.*

Finalmente após a execução de uma etapa do processo, o sistema exibe o próximo valor da pose do objeto e retorna ao início do ciclo. Ao mesmo tempo do processo de exibição da pose do objeto em movimento, poderia ter sido utilizado essa informação (pose futura do objeto) para um sistema de controle.

Essa simulação tem o objeto de demonstrar de uma forma detalhada o funcionamento do método proposto onde os testes realizados e os resultados obtidos são discutidos no capítulo posterior.

5.2.2 Simuladores para o rastreamento de objetos em movimento através de um controlador de velocidade

Nessa sessão são explicados os simuladores com controlador de velocidades utilizados nesse trabalho. Um desses dois simuladores considera o caso de um sinal filtrado no filtro de Kalman como a derivada do erro em função do tempo e o método corretor de *dead-times* aplicado a essa variável. O outro simulador tem o sinal sendo a previsão futura do objeto em movimento e também é utilizada uma equação para estabelecer a pose futura do manipulador robótico e em seguida é aplicado os processos do controlador de velocidades incluindo um segundo filtro de Kalman, o método corretor de *dead-times* aqui é aplicado na previsão futura do objeto.

Cada uma das simulações com controlador de velocidades são constituídas de um manipulador robótico AFMA - 6 com controle IBVS, com a câmera localizada em seu órgão terminal (mão do robô) e 6 graus de liberdade. O objetivo do robô é rastrear um objeto em movimento de forma a reduzir a diferença (erro) entre a pose de atual do manipulador robótico em relação ao objeto e a pose desejada do manipulador robótico em relação ao objeto. O IBVS é utilizado para controlar a velocidade do manipulador de acordo com a movimentação necessária para esse manipulador se aproximar do objeto.

A Figura 5.5-B representa o objeto simulado com 4 pontos a ser rastreados pelo manipulador robótico com o sistema IBVS. A Figura 5.5-A representa a visão da câmera instalada no braço robótico (cor cinza na Figura 5.5-A, isto é, linhas mais claras) em relação a pose atual do objeto que está sendo rastreado (cor preta na Figura 5.5-A, isto é, linhas mais escuras).

Para o reconhecimento da pose do objeto são utilizados 4 pontos que representam as características extraídas do objeto em uma imagem e que indicam a pose *p* no plano da imagem (2D). *p* é também importante para estabelecer a pose no mundo real em sistemas PBVS, onde são realizadas conversões para esse plano.

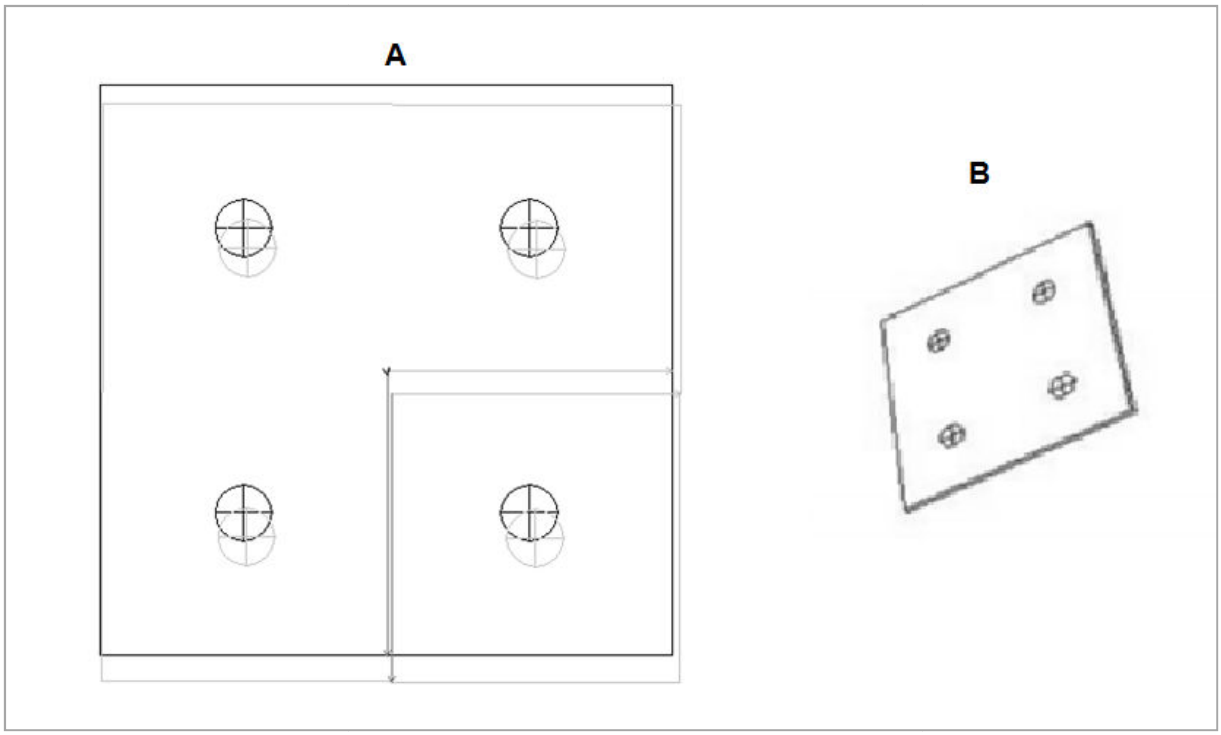


Figura 5.5 – Visualização da câmera instalada no manipulador robótico em relação a pose do objeto em movimento das simulações com AFMA-6.

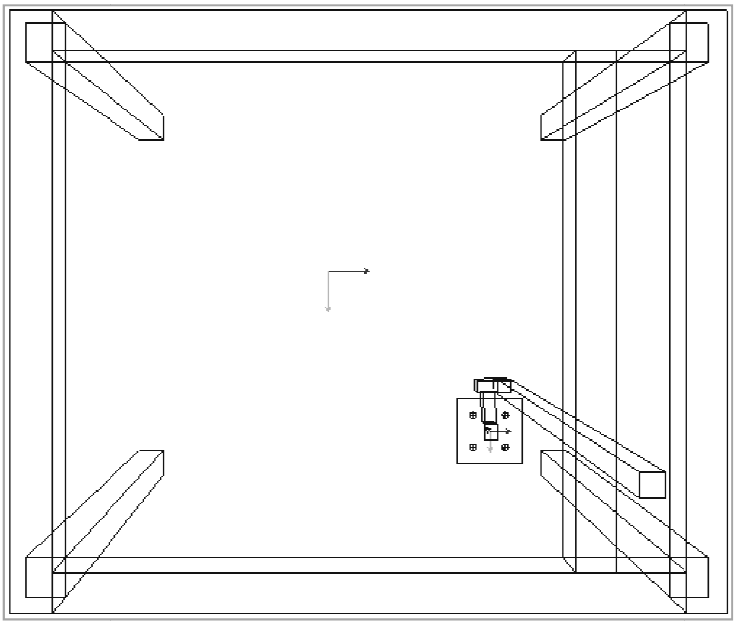


Figura 5.6 – Modelo da simulação do AFMA – 6 (visão de planta).

A Figura 5.6 demonstra uma visualização da superfície da versão simulada do robô AFMA - 6 que consiste no manipulador robótico e seu espaço de trabalho. Também é representado na Figura 5.6 o objeto rastreado onde a pose inicial do

objeto em relação ao manipulador robótico. O manipulador robótico possui a capacidade de se deslocar por praticamente todo o espaço de trabalho, definido dentro das quatro barras também visualizadas na Figura 5.6.

Assim como no caso da estimativa de pose futura do objeto demonstrado anteriormente o movimento do objeto é realizado de diferentes formas como linear, *zig-zag* e circular (Figura 5.3).

Em tarefas comuns com manipuladores robóticos é possível e comum o órgão terminal do robô possuir alguma ferramenta para executar suas tarefas (como dedos, pinças, etc.) (CRAIG, 2013), mas como o propósito desse trabalho é o rastreamento, apenas a câmera é considerada. A câmera utiliza o modelo de projeção perspectiva, sem levar em consideração distorções da cena que não é o foco principal do trabalho.

Nas simulações a variável controlada (erro) é a diferença entre a pose desejada e corrente do objeto em relação ao manipulador robótico. A variável manipulada é a velocidade da câmera no órgão terminal e os próprios simuladores fazem a conversão dessa velocidade para as articulações do robô, sendo que as alterações dessa variável no decorrer da execução do processo do sistema de controle afetam na variável controlada.

Após o início da movimentação do objeto começa um ciclo de rastreamento do robô e de realimentação (constante no decorrer desse ciclo) do sistema de controle. A taxa de amostragem no sistema de controle é de aproximadamente 50 milissegundos.

No fluxograma da Figura 5.7 é representado o método corretor de *dead-times* aplicado a derivada do erro em função do tempo do manipulador robótico em relação ao objeto. No fluxograma da Figura 5.8 é representado o método corretor de *dead-times* aplicado a pose futura do objeto em movimentação e com a equação de pose futura do manipulador robótico em movimentação.

Ambos os fluxogramas representam o caso de um sistema de controle por IBVS para controlar a velocidade do AFMA – 6 na operação de rastreamento de objetos em movimento. Assim como na simulação anterior a parte de maior interesse desse trabalho nos fluxogramas é a partir do momento onde o sistema entra em recursividade (cada unidade dessa recursividade é uma amostra para o sistema de controle), pois antes disso apenas são definidas algumas variáveis do sistema de controle e do filtro de Kalman.

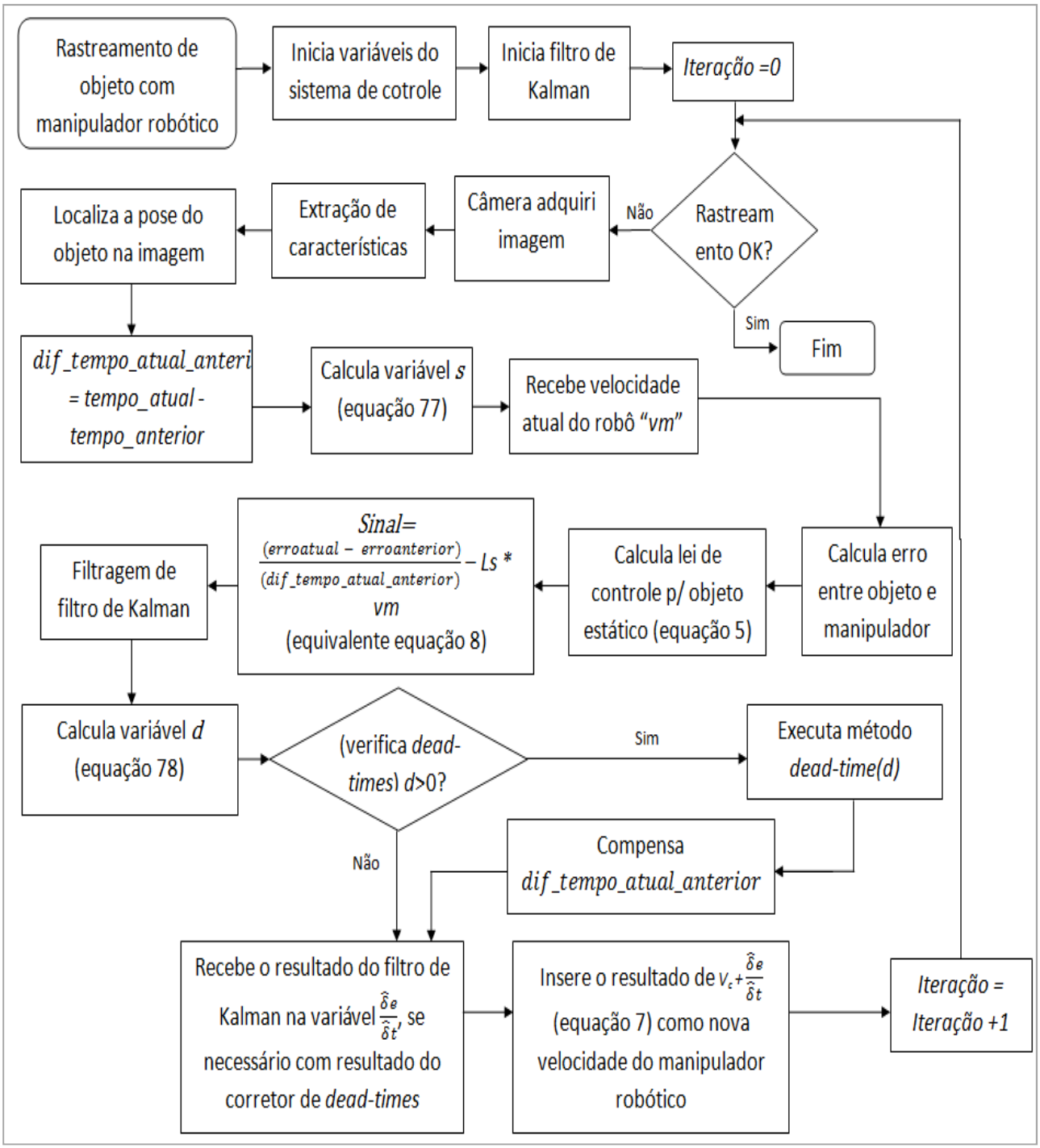


Figura 5.7 – Fluxograma do processo do sistema de controle de velocidade (IBVS) para o robô AFMA - 6 com utilização do filtro de Kalman e método para tratar *dead-times* aplicado na derivada do erro entre o manipulador robótico e o objeto.

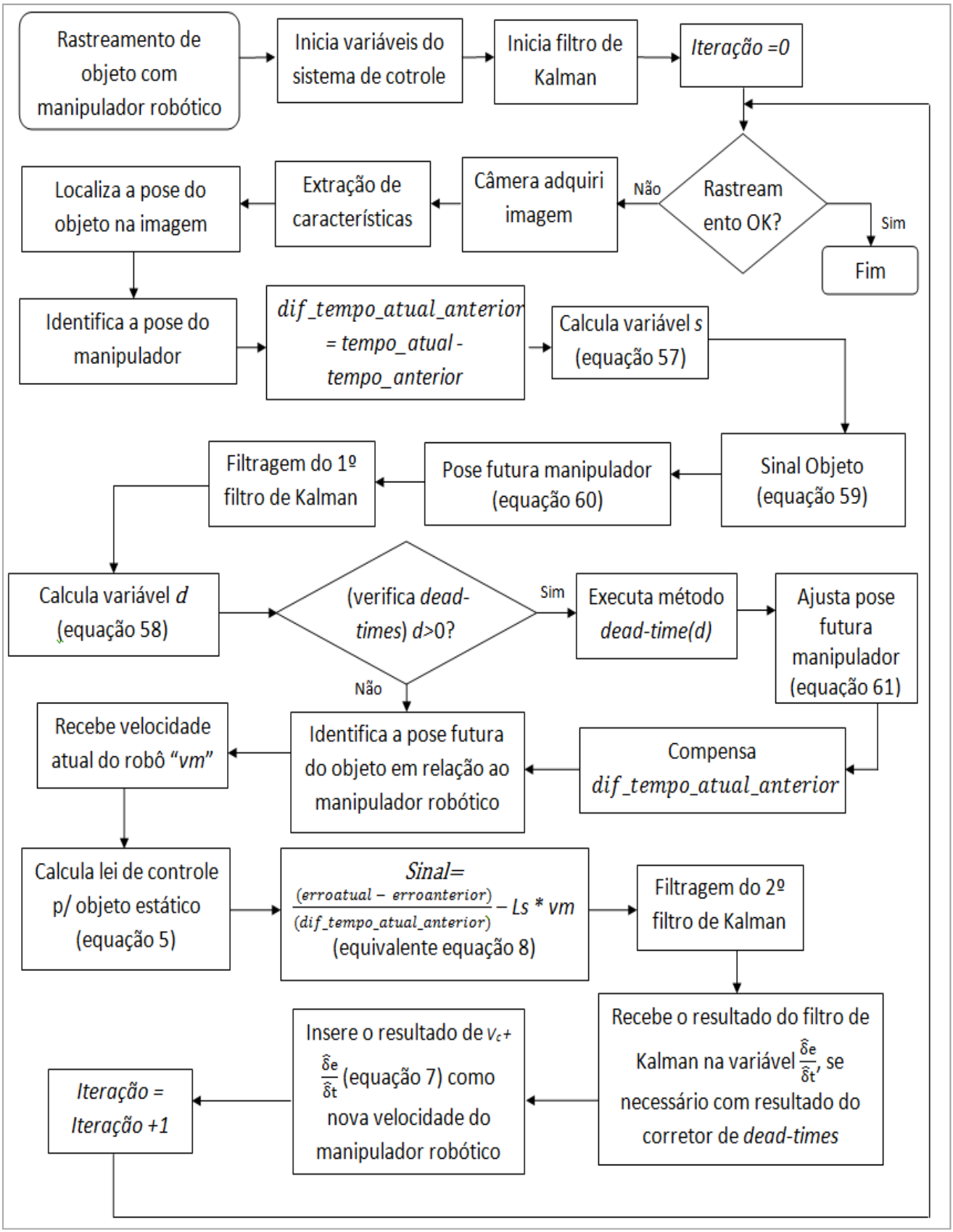


Figura 5.8 – Fluxograma do processo do sistema de controle de velocidade (IBVS) para o robô AFMA - 6 com utilização do filtro de Kalman e método para tratar *dead-times* aplicado sobre a pose futura do manipulador robótico e do objeto.

No início do processo de recursividade ambos os sistemas são semelhantes entre si e ao fluxograma apresentado anteriormente, onde são inicializadas as variáveis utilizadas no sistema de controle e do filtro de Kalman. Após essas variáveis inicializadas o sistema entra no processo recursivo de calcular o erro do objeto em relação ao manipulador robótico, calcular a lei de controle e alimentar o sistema robótico com a nova velocidade calculada. O processo recursivo utilizado nesse trabalho foi executado de acordo com um número determinado na etapa de testes, porém em um sistema real normalmente seria executado até o manipulador robótico se posicionar a uma pose aceitável em relação ao objeto.

No processo recursivo inicialmente acontece a estimativa de pose do objeto de acordo com o modelo IBVS, nessa etapa são consideradas a aquisição da imagem pela câmera, extração de características (identificar os pixels que serão utilizados como comparação com a imagem de referência) e com as características de interesse identificadas é possível determinar a pose do objeto na imagem.

São calculados também á *dif_tempo_atual_anterior* e a taxa de amostragem *s* (equação 57). Nos fluxogramas das Figuras 5.7 e 5.8 foram omitidas (por questão de espaço) a iteração inicial sendo considerada como tratada.

A partir daqui são explicados individualmente cada um dos fluxogramas (Figuras 5.7 e 5.8), pois cada um possui características especificar em relação ao outro nas próximas etapas.

(a) Figura 5.7 – Utiliza o método para tratar *dead-times* aplicado a derivada do erro em função do tempo do manipulador robótico em relação ao objeto:

Com a pose do objeto é possível estabelecer o sinal que será utilizado no filtro de Kalman, mas para isso também é necessário a velocidade atual do manipulador robótico, sendo assim essa velocidade é obtida do manipulador.

Utilizando a pose atual do manipulador robótico em relação ao objeto é calculado o erro e também com a velocidade do manipulador é calculado a lei de controle (utilizando a equação 5 já apresentada nesse trabalho) para o caso do objeto estático.

Nesse fluxograma é subentendido que o erro entre o manipulador robótico e o objeto no instante anterior ao atual está armazenado. Dessa forma é calculada a derivada do erro do manipulador robótico em relação ao objeto (equivalente á equação 8) em função do tempo já considerando também algumas variáveis de

conversão para o plano real (necessário para trabalhar com velocidade da câmera). O novo valor obtido é utilizado como sinal do filtro de Kalman.

Ao final da execução da filtragem do filtro de Kalman pode ou não ser executado o método para tratar *dead-times.* Para que o método de execução de *dead-times* seja executado ou não, é utilizada a equação 58 e o resultado obtido como forma de verificação e se for necessário o método corretor de *dead-times* é executado corrigindo a estimativa do filtro de Kalman.

O novo valor é utilizado para compor a nova lei de controle do sistema de controle (equação 7), que calcula a nova velocidade do manipulador robótico e ao final do processo esse novo valor é alimentado ao sistema robótico.

(b) Figura 5.7 – Utiliza o método para tratar *dead-times* aplicado a pose futura do objeto em movimento e uma equação para o manipulador em movimento:

Na Figura 5.8 são utilizados dois filtros de Kalman, um para estimar a pose futura do objeto e outro para tratar a velocidade estimada do manipulador.

Além da pose futura do objeto é necessária também uma equação para determinar a pose futura do manipulador. A pose futura do objeto em movimento é determinada a partir da equação 59 e a pose futura do manipulador robótico em movimento é determinada a partir da equação 60.

$$\textit{Sinal objeto} = \textit{pose\_atual} + \frac{pose\_atual - pose\_anterior}{dif\_tempo\_atual\_anterior} \quad (59)$$

$$\textit{Previsão Manipulador} = \textit{pose\_atual\_manipulador} + \textit{deslocamento\_eq\_ciclo\_controle\_velocidade} \quad (60)$$

Na equação 59 a variável *pose_atual representa* a pose atual do objeto e a variável *pose_anterior* representa a pose anterior do objeto. É subentendido que a *pose_anterior* do objeto está armazenada.

Na equação 60 a variável *pose_atual_manipulador* representa a pose atual do manipulador robótico. A variável *deslocamento_eq_ciclo_controle_velocidade* representa o deslocamento do manipulador em uma unidade do ciclo de controle (amostra do ciclo de controle) com a velocidade atual do manipulador robótico.

O sinal do objeto é filtrado no filtro de Kalman. O primeiro filtro de Kalman é responsável por melhorar o sinal da pose futura do objeto e do manipulador.

Depois de estimado o sinal pelo primeiro filtro de Kalman pode se necessário ser executado o método para tratamento de *dead-times.* Para isso novamente é executado a equação 58 responsável pelo cálculo da quantidade de atrasos *d*, se ocorreram *dead-times* é executado o método corretor de dead-times.

Quando executado o método para tratar *dead-times* é executado a equação 61 para prever a pose futura do manipulador robótico conforme sua velocidade.

$$Previsão\ Manipulador = pose\_atual\_manipulador + d * (deslocamento\_eq\_ciclo\_controle\_velocidade) \quad (61)$$

O sinal estimado pelo filtro de Kalman tratado ou não pelo método para tratar *dead-times* e a previsão futura da pose do manipulador são utilizados para estabelecer a relação futura entre o objeto e o manipulador robótico. Baseados na relação futura entre o objeto e o manipulador robótico são executados os cálculos agora referente a estimativa de velocidade do robô.

Primeiramente é recebida a velocidade atual do manipulador robótico, essa variável é utilizada em conjunto a relação futura entre o objeto e o manipulador robótico para compor a lei de controle para o objeto estático (equação 5).

Um novo sinal também é calculado para ser utilizado em outro filtro de Kalman. O sinal calculado de forma equivalente á equação 8 onde a variável referente ao sinal atual está corrigida de acordo com o primeiro filtro de Kalman e corretor de *dead-times* . O “segundo” filtro de Kalman utilizado nesse trabalho é responsável por melhorar a estimativa da derivada do erro entre o objeto e o manipulador robótico.

O novo valor é utilizado para compor a nova lei de controle do sistema de controle (equação 7), que calcula a nova velocidade do manipulador robótico e ao final do processo esse novo valor é alimentado ao sistema robótico.

Na simulação onde foram utilizadas as estimativas das poses futuras do objeto em movimento e manipulador pode ser observado que o método corretor de *dead-times* pode ser aplicado somente para tratar de problemas causados nos processos de visão computacional. Na simulação que foram utilizadas as estimativas

diretas do erro entre o manipulador robótico e objeto é observado que o método corretor de *dead-times* é aplicado tanto para tratar problemas de *dead-times* em visão computacional como *dead-times* causados em processamento e transferência de dados entre os componentes do sistema.

Essas simulações têm o objetivo de demonstrar de uma forma um pouco mais prática o funcionamento do método proposto aplicado ao controle de um manipulador robótico. Dessa forma é possível analisar o método proposto para o tratamento de diferentes tipos de sinais. No capítulo seguinte são discutidos e apresentados os testes e resultados desse trabalho avaliando o método utilizado.

Capítulo 6

EXPERIMENTOS, TESTES E RESULTADOS

Com o propósito de demonstrar as características do método para tratar *dead-times* utilizado nesse trabalho, esse capítulo descreve os experimentos e os testes realizados nessa pesquisa. As formas de demonstração de resultados são através de gráficos, tabelas e quando necessário algumas situações são explicadas com maiores detalhes. O objetivo final do trabalho é testar e comparar a métodologia proposta que utiliza o método para tratar *dead-times* em relação a abordagem padrão sem o método para tratar *dead-times* enfatizando a simulação do método proposto em esteiras transportadoras.

6.1 Experimentos e testes

Nesse trabalho foram realizados três tipos de experimentos para validar a metodologia proposta:

(a) experimentos que utilizam uma estimativa da pose futura do objeto em movimento como o sinal do filtro de Kalman e do método para tratar *dead-times*;

(b) experimentos que utilizam uma estimativa da pose futura do objeto em movimento como o sinal para o método corretor de *dead-times*, e depois disso, utilizam essa variável para compor a derivada do erro entre a pose atual e desejada do manipulador robótico em relação ao objeto otimizando esse sinal no filtro de Kalman;

(c) experimentos que utilizam a derivada do erro entre a pose atual e desejada do manipulador robótico em relação objeto como sinal no filtro de Kalman e no método para tratar *dead-times*.

O objetivo do primeiro tipo de experimento é avaliar a capacidade de previsão do objeto também em diferentes tipos de movimento. O objetivo dos dois últimos tipos de experimentos é proporcionar dados que permitam uma análise próxima das condições reais de sistema de rastreamento de objetos por servovisão e controlador de velocidade. Em todos os 3 tipos de experimentos foram avaliados o método com 3 tipos de movimentação gerando um total de 9 experimentos. Todos os experimentos consideram o modelo IBVS onde podem ocorrer *dead-times*. Os experimentos foram simulados e os detalhes de cada simulador utilizado foram explicados no capítulo anterior.

Nos três tipos de experimentos o objeto é rastreado de forma planar (eixos *x* e *y*). Algumas formas de movimentação foram elaboradas para os testes e são descritas a seguir:

- forma diagonal/linear: o objeto se desloca de forma diagonal em único sentido (linearmente) com 0.0017 unidades no eixo *x* e 0.001 unidades no eixo *y* a cada iteração. Essas unidades são da simulação, não tem um valor usual definido e visualmente depende dos pixels da tela.
- forma circular: o movimento é definido pelas expressões 62 e 63

$$x = (0 - sin * (unidade\ tempo) * 1.2) \quad (62)$$

$$y = (0.1 - cos * (unidade\ tempo) * 1.2) \quad (63)$$

que representam uma relação entre os catetos (adjacente e oposto) e o ângulo que representa o descolamento de objetos obtido pela regra de movimento circular uniforme. A variável *unidade_tempo* é iniciada com o valor 10 e acrescida a cada iteração 0.005 unidades no caso do rastreamento de objeto em movimento e 0.002 unidades no caso de auxílio no controlador de velocidades.

- forma zig-zag: o objeto se desloca linearmente de forma diagonal para cima ou para baixo com 0.0017 unidades no eixo *x* e 0.005 ou -0.005

unidades no eixo *y* a cada iteração. O valor de *y* é alterado sempre que o objeto atinge as posições -0.35 (0.005) ou 1.5 (-0.005) no eixo 5.

Esses três tipos de movimentos permitem analisar o método estudado nesse trabalho em condições próximas a movimentação real, sendo o rastreamento linear comum, o rastreamento circular permite analisar situações de curvas e o movimento em zig-zag permite verificar alterações drásticas na movimentação do objeto.

Detalhes sobre o método para tratar *dead-times* estudado nesse trabalho e utilizado junto a esses simuladores são discutidos nas sessões seguintes.

6.1.1 Experimentos com estimativa de pose futura de objeto

Através da abordagem que realiza a previsão futura da posição de um objeto em movimento por servovisão foram realizados três experimentos com o objeto em diferentes tipos de movimentação. Nessa simulação e experimentos são considerados que cada etapa normal de amostra (sem a ocorrência de *dead-times*) do sistema de controle tem uma unidade de tempo e também nesse tempo o objeto se desloca com um valor fixo de acordo com as regras estabelecidas anteriormente (as unidades de tempo de cada etapa são invariantes).

A cada unidade de tempo são executados todos os passos de uma amostra do sistema de servovisão, considerando a aquisição da imagem, formulação da lei de controle e alimentação do sistema de juntas robóticas. Nessa unidade de tempo o objeto se desloca com um valor fixo (por exemplo, 0.0017 unidades em *x* e 0.001 unidades em *y* para o caso diagonal/linear).

Na ocorrência de *d dead-times,* a cada unidade de *d* equivalente a uma unidade de tempo é considerado um deslocamento a mais na movimentação do objeto e o acréscimo de uma unidade de tempo no sistema.

Um conjuntos de regras foi estabelecido nesse trabalho tentando abranger o máximo de situações o possível onde ocorrem *dead-times*. Para isso alguns *dead-times* utilizados nessa simulação foram “forçados” em determinadas iterações para proporcionar um melhor controle da operação e análise dos resultados. As seguintes regras determinam a execução de *dead-times*:

- Acontecem somente *dead-times* de visão computacional nas iterações que são múltiplas de 4 e não múltiplas de 7.

- Acontecem somente *dead-times* de processamento de dados/transferência de dados nas iterações que são múltiplas de 7 e não múltiplas de 4.
- Acontecem ao mesmo tempo *dead-times* de visão computacional e de processamento de dados/transferência de dados nas iterações que são ao mesmo tempo múltiplas de 4 e 7.
- Todas iterações como 20/21 (sendo 20 múltiplo de 4 e 21 múltiplo de 7) e 35/36 colocam a ocorrência de dois atrasos diferentes consecutivos.
- As iterações 49/50/51, 221/222/223 e 302/303/304 estão submetidas a um processo de atrasos de processamento de dados/transferência de dados consecutivos.
- As iterações especificas 38/39/40, 296/297/298, 352/353/354 estão submetidas a um processo de *dead-times* de visão computacional.

Também foram considerados a quantidade de *dead-times* que podem ocorrer na simulação de forma aleatória. A regra que determina a quantidade de *d dead-times* é a seguinte:

- quando a ocorrência de *dead-times* é forçada por visão computacional alternadamente na primeira ocorrência é considerado d = 7, na segunda d = 10, na terceira d = 9, na quarta d = 8, na quinta d = 12 e na sexta d = 11. Ao utilizar a sexta ocorrência o sistema volta a repetir o processo a partir da primeira regra.
- quando a ocorrência de *dead-times* é forçada por processamento e transmissão de dados alternadamente na primeira ocorrência é considerado d = 5, na segunda d = 2, na terceira d = 4, na quarta d = 1, na quinta d = 6 e na sexta d = 3. Ao utilizar a sexta ocorrência o sistema volta a repetir o processo a partir da primeira regra.

É esperado que com todas regras seja possível avaliar a maioria das situações que pode ocorrer em um sistema real e os resultados são apresentados no subcapítulo seguinte. Esse experimento por possuir diversos dados controlados tem o objetivo de proporcionar uma melhor analise do método proposto.

6.1.2 Experimentos com controlador de velocidades (Robô AFMA-6)

Nos experimentos realizados com o controlador de velocidades foram considerados uma simulação de uma tarefa de rastreamento de um objeto em movimento através de um controle de velocidade com servovisão para o manipulador robótico AFMA-6.

a) Nos experimentos que consideram a estimativa da variável $\left(\frac{\hat{\delta}e}{\hat{\delta}t}\right)$ que representa a variação do erro (entre a pose atual e desejada do manipulador robótico em relação ao objeto) no tempo. A variável $\left(\frac{\hat{\delta}e}{\hat{\delta}t}\right)$ é utilizada para compor uma lei de controle através do modelo IBVS sendo utilizada em um controlador de velocidades do manipulador robótico. Essa estimativa é inicialmente realizada por uma derivada e posteriormente ajustada pelo filtro de Kalman. Após o ajuste do filtro de Kalman caso ocorram *dead-times* é utilizado o método corretor de *dead-times.* A derivada é utilizada pela lei de controle no cálculo da velocidade do robô que por sua vez influência na aproximação do manipulador robótico em relação ao objeto.

b) Nos experimentos onde o método é aplicado sobre a estimativa de pose futura do objeto, inicialmente é utilizado um primeiro filtro de Kalman, seguido pela aplicação da abordagem proposta e após essa aplicação e com a pose atual do manipulador, o erro (entre a pose atual e desejada do manipulador robótico em relação ao objeto) e a variável $\left(\frac{\hat{\delta}e}{\hat{\delta}t}\right)$ são corrigidos. Com os valores corrigidos são aplicados um segundo filtro de Kalman para melhorar a variável $\left(\frac{\hat{\delta}e}{\hat{\delta}t}\right)$ e posteriormente calculado a lei de controle (modelo de servovisão IBVS) no controlador de velocidades. Nessa abordagem é utilizado o método para tratar *dead-times* apenas nas operações de visão computacional em servovisão.

A cada unidade de tempo são executados todos os passos de uma amostra do sistema de controle (inclusive etapas adicionais como aquisição da velocidade atual do manipulador para ser utilizado na lei de controle). São considerados 4 pontos no objeto a ser rastreado e em uma unidade de tempo o objeto (ou cada ponto do objeto) se desloca com um valor fixo (por exemplo, 0.0017 unidades no eixo *x* e 0.001 unidades no eixo *y* para o caso diagonal/linear) e também nesse tempo o manipulador robótico se desloca de acordo com sua velocidade.

Diferentemente da simulação anterior e embora cada amostra do sistema possua cerca de 50 ms essa amostra varia um pouco. Isso acontece porque o tempo de alimentação da velocidade obtida na lei de controle no robô simulado pode variar um pouco devido ao tempo de processamento do computador utilizado. Essas variações no simulador fazem com que o método para tratar *dead-times* as considerem tentando realizar o tratamento. Além disso, *dead-times* são forçados para analise do desempenho do sistema.

Quando forçado em ambos os sistemas uma ocorrência de *d dead-times* a cada unidade de *d* é considerado um deslocamento a mais na movimentação do objeto e o acréscimo do tempo de processamento é realizado com base no tempo de processamento dessas *d* movimentações. Cada unidade *d* tem o tempo de 50 ms.

Os *dead-times* utilizados nessas simulações foram "forçados" em determinadas iterações e as seguintes regras foram utilizadas:

(a) Para o experimento que utiliza o método para tratar *dead-times* na derivada do erro do manipulador robótico em relação ao objeto:

- Acontecem somente *dead-times* de visão computacional nas iterações que são múltiplas de 20.
- Acontecem somente *dead-times* de processamento / transferência de dados nas iterações que são múltiplas de 10 e não múltiplas de 20.

(b) Para o experimento que utiliza o método para tratar *dead-times* com sinais de previsão futura do objeto e manipulador robótico:

- Acontecem somente *dead-times* de visão computacional nas iterações que são múltiplas de 10.
- Esse método não foi aplicado para o tratamento de *dead-times* causados por processamento/transferência de dados.

Também foram considerados a quantidade de *dead-times* que podem ocorrer nas simulações de forma aleatória. A regra em que determina a quantidade de *d dead-times* é a seguinte:

- Para ambos os experimentos quando a ocorrência de *dead-times* é forçada por visão computacional alternadamente na primeira ocorrência é considerado $d = 7$, na segunda $d = 10$, na terceira $d = 9$, na quarta $d = 8$, na quinta $d = 12$ e na sexta $d = 11$. Ao utilizar a sexta

ocorrência o sistema volta a repetir o processo a partir da primeira regra.

- Nos experimentos que o método é aplicado sobre a derivada do erro, quando a ocorrência de *dead-times* é forçada por processamento e transmissão de dados alternadamente na primeira ocorrência é considerado $d = 5$, na segunda $d = 2$, na terceira $d = 4$, na quarta $d = 1$, na quinta $d = 6$ e na sexta $d = 3$. Ao utilizar a sexta ocorrência o sistema volta a repetir o processo a partir da primeira regra.

Devido a variável a ser controlada ser a velocidade o comportamento do sistema fica difícil de ser analisado com um número grande de *dead-times*, sendo assim foram considerados a introdução de *dead-times* de maneira mais espaçada.

Os testes baseados nesses experimentos foram realizados sobre os simuladores comentados anteriormente e os resultados são apresentados no subcapítulo seguinte.

6.2 Resultados

Nesse tópico são analisados os resultados para os três tipos de experimentos com as condições de aplicação ou não aplicação do método corretor de *dead-times.*

Os subcapítulos estão organizados de acordo com o tipo de movimento do objeto e cada um dos três tipos de experimentos é analisado para cada movimento.

As tabelas/gráficos utilizados demonstram os resultados de uma forma geral (amostra ou todos os dados em alguns casos). Os dados foram obtidos nos testes considerando o erro linear quadrático em relação aos valores das coordenadas x e y. Quando necessário são apresentados dados e análises extras para demonstrar explicações sobre determinados casos.

No caso da estimativa de pose futura do objeto são consideradas apenas um ponto para representar o objeto, ou seja, um par x e y. O objetivo é deixar a pose futura estimada o mais próximo da pose futura desejada com o objeto em movimento, ou seja, quanto mais próximo de zero os valores melhor o resultado.

No caso de ambas as simulações com controlador de velocidades são considerados quatro pontos no objeto. Com os pontos do objeto, são calculados

para cada um dos pontos os erros em relação ao manipulador robótico e esses erros compõe o erro linear quadrático em relação às coordenadas *x* e *y* nos gráficos. O erro linear quadrático apresenta melhores resultados dependendo do desempenho da estimativa de cada método em cada simulação. Com o propósito de diminuir a influência de valores discrepantes no gráfico, os dados apresentados nos gráficos foram calculados com base na média de 100 repetições de cada uma das 400 iterações. Como no caso anterior o objetivo é deixar os valores mais próximos de zero que representa a proximidade do manipulador robótico em relação ao objeto.

6.2.1 Resultados das estimativas de pose futura do objeto

Nos gráficos das Figuras 6.1, 6.2 e 6.3 são apresentados duas relações. A primeira entre a pose futura real e a pose futura estimada pelo filtro de Kalman com o método corretor de *dead-times*. A segunda entre a pose futura real e a pose futura estimada com o filtro de Kalman, mas sem o método corretor.

O gráfico da Figura 6.1 considera o objeto com movimentação linear/diagonal, o gráfico da Figura 6.2 considera o objeto com movimentação circular e o Gráfico da Figura 6.3 considera o objeto com movimentação em zig-zag (alterações drásticas de movimentação).

Nos dois primeiros casos é possível observar que quando utilizado o filtro de Kalman sem corretor de *dead-times* (linhas contínuas) ocorreram diversos erros na estimativa da pose futura do objeto em movimento. Quando utilizado o método corretor de *dead-times* (linhas tracejadas) na ocorrência de *dead-times* os resultados demonstram maior proximidade do erro á 0 do que a não utilização do método corretor de *dead-times*, e assim essa analise é suficiente para determinar que nesse caso o método com corretor de *dead-times* foi superior.

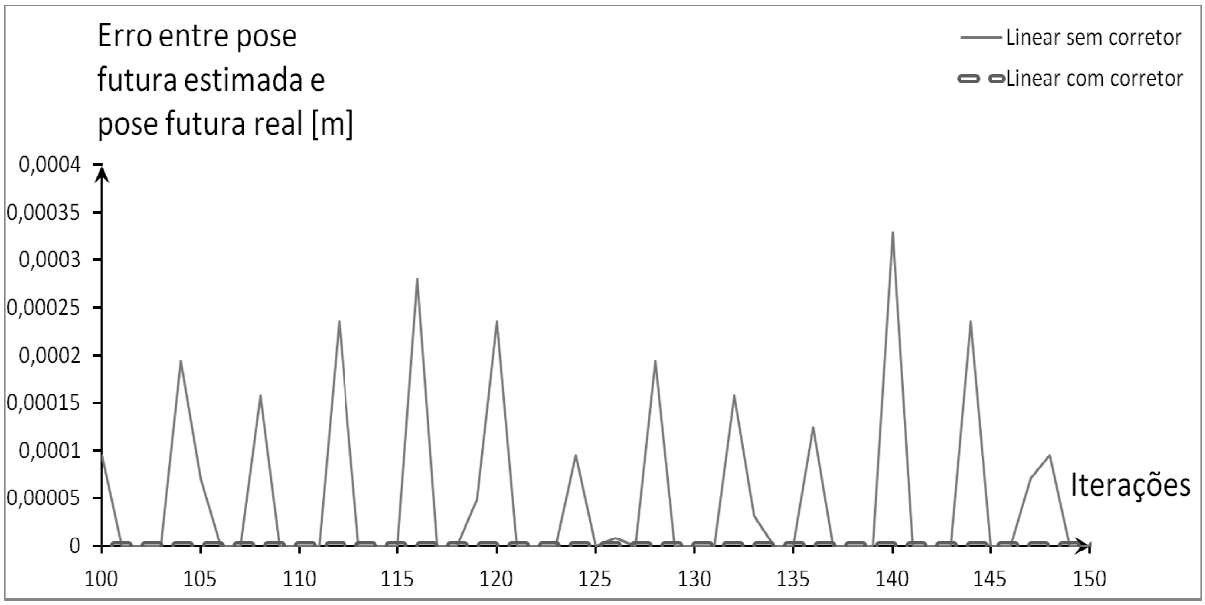


Figura 6.1 – Gráfico do erro de previsão futura de um objeto em movimento linear. Foi utilizado o filtro de Kalman e, em um dos casos, o corretor de *dead-times.*

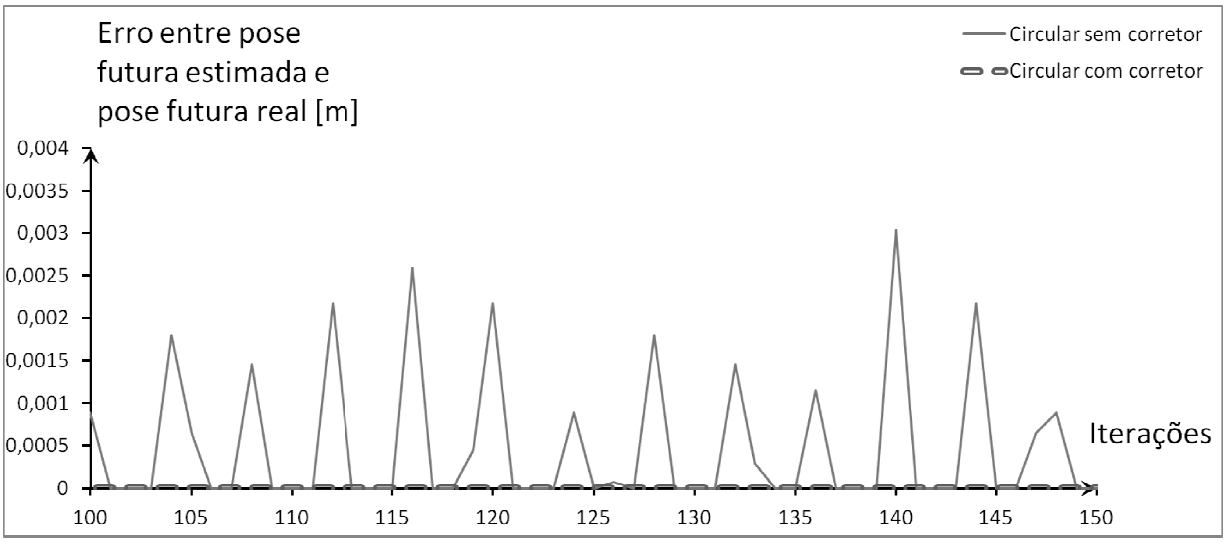


Figura 6.2 – Gráfico do erro de previsão futura de um objeto em movimento circular. Foi utilizado o filtro de Kalman e, em um dos casos, o corretor de *dead-times.*

No caso de alterações drásticas no movimento, ou seja, com movimentação em zig-zag do objeto, durante a execução do método com a abordagem para tratar *dead-times* as estimativas obtidas tiverem resultados inferiores em relação a abordagem com filtro de Kalman sem o método corretor de *dead-times*. A Tabela 6.1 representa alguns dados da movimentação em zig-zag para o caso de estimativa de pose futura do objeto em movimento.

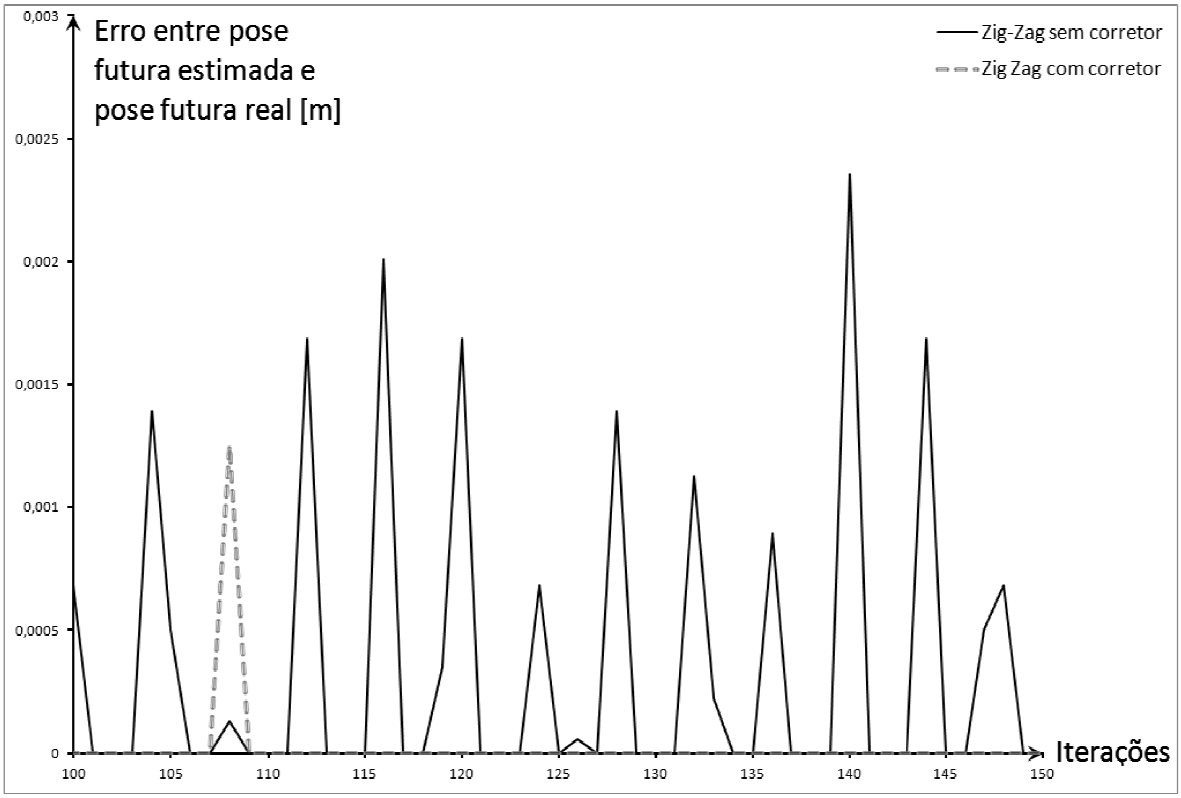


Figura 6.3 – Gráfico do erro de previsão futura de um objeto em movimento zig-zag. Foi utilizado o filtro de Kalman e, em um dos casos, o corretor de *dead-times.*

Tabela 6.1 – Amostra dos dados testados referente a previsão futura de objetos em movimento em zig-zag onde foi utilizado um filtro de Kalman e em um dos casos foi utilizado o método para tratar *dead-times.*

Iter	Próxima posição do objeto [m]		Filtro de Kalman [m]		Filtro de Kalman c/ corretor [m]		d *dead-times*	
	x	Y	x	Y	X	y	Visão	TDD/PDD
3	-0,0068	-0,02	-0,0068	-0,02	-0,0068	-0,02	0	0
4	-0,0289	-0,085	-0,0085	-0,025	-0,0286	-0,0843	12	0
5	-0,0306	-0,09	-0,0306	-0,09	-0,0306	-0,09	0	0
18	-0,1122	-0,33	-0,1122	-0,33	-0,1122	-0,33	0	0
19	-0,1139	-0,335	-0,1139	-0,335	-0,1139	-0,335	0	0
20	-0,1309	-0,315	-0,1156	-0,34	-0,1309	-0,385	9	0
21	-0,1394	-0,29	-0,1326	-0,31	-0,1394	-0,2832	0	4

O movimento zig zag é composto por pequenas movimentações de forma diagonal/linear como no caso da iteração 4 (na Tabela 6.1). Sempre que o movimento é diagonal/linear (que compõem parte da movimentação zig-zag) os resultados do método para tratar *dead-times* são semelhantes aos resultados apresentados no caso de rastreamento linear/diagonal.

Na iteração 19 o valor da coordenada *y* no movimento do objeto estava diminuído e quando chegou à iteração 20 passou a aumentar alterando drasticamente o sentido do movimento. Na iteração 21 o objeto continuou executando normalmente seu trajeto em relação ao sentido alterado na iteração 20. No momento da alteração de sentido na iteração 20 ocorreram 9 *dead-times*, porém essa ocorrência é baseada na movimentação anterior das iterações 18 e 19. O erro do método para tratar *dead-times* ocorreu justamente pela alteração de movimento do objeto no momento que o método utilizava as informações das iterações 18 e 19 que estavam em sentido diferente ao sentido que ocorreu na iteração 20 do objeto.

O fato da abordagem com filtro de Kalman ter apresentado melhores resultados, não acontece devido ao filtro de Kalman tratar a ocorrência de *dead-times* de forma eficiente, mas devido na ocorrência de *dead-times* durante a alteração de movimento do objeto onde o método para tratar *dead-times* estimou um número de poses à frente da pose real do objeto. A estimativa do filtro de Kalman ficou parada com a pose do objeto antes de ocorrer os *dead-times* e quando o objeto passou a realizar o novo sentido se aproximou dessa pose estimada até então.

6.2.2 Resultados das estimativas de velocidade com AFMA-6 (corretor de *dead-times* aplicado diretamente na pose do objeto)

Assim como no caso anterior, nos gráficos das Figuras 6.4, 6.5 e 6.6 são apresentados duas relações. A primeira relação representa o erro entre a pose desejada e atual do manipulador robótico em relação ao objeto utilizando o filtro de Kalman com o método corretor de *dead-times*. A segunda representa o erro entre a pose desejada e atual do manipulador robótico em relação ao objeto com o filtro de Kalman, mas sem o método corretor.

Os gráficos das Figuras 6.4, 6.5 e 6.6 consideram respectivamente o objeto com movimento linear/diagonal, movimento circular e movimento em zig-zag (alterações drásticas de movimentação).

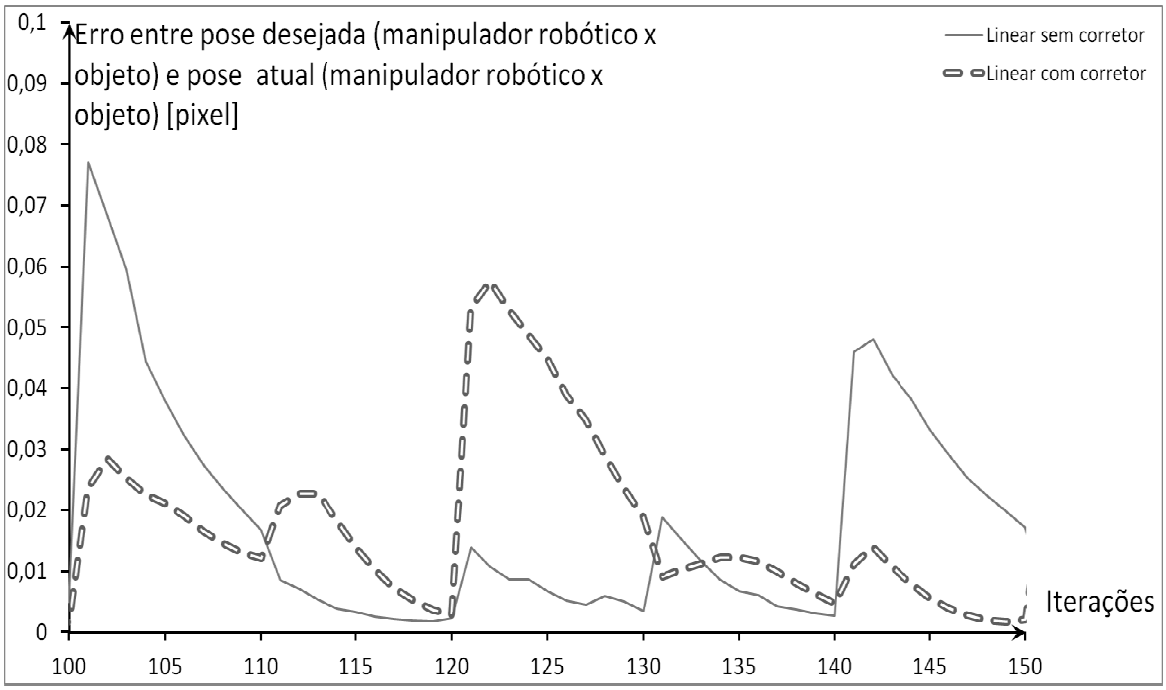


Figura 6.4 – Gráfico do erro entre as poses atuais e desejadas do manipulador robótico em relação ao objeto em movimento linear. Foi utilizado o filtro de Kalman e, em um dos casos, o corretor de *dead-times* (aplicado na pose do objeto).

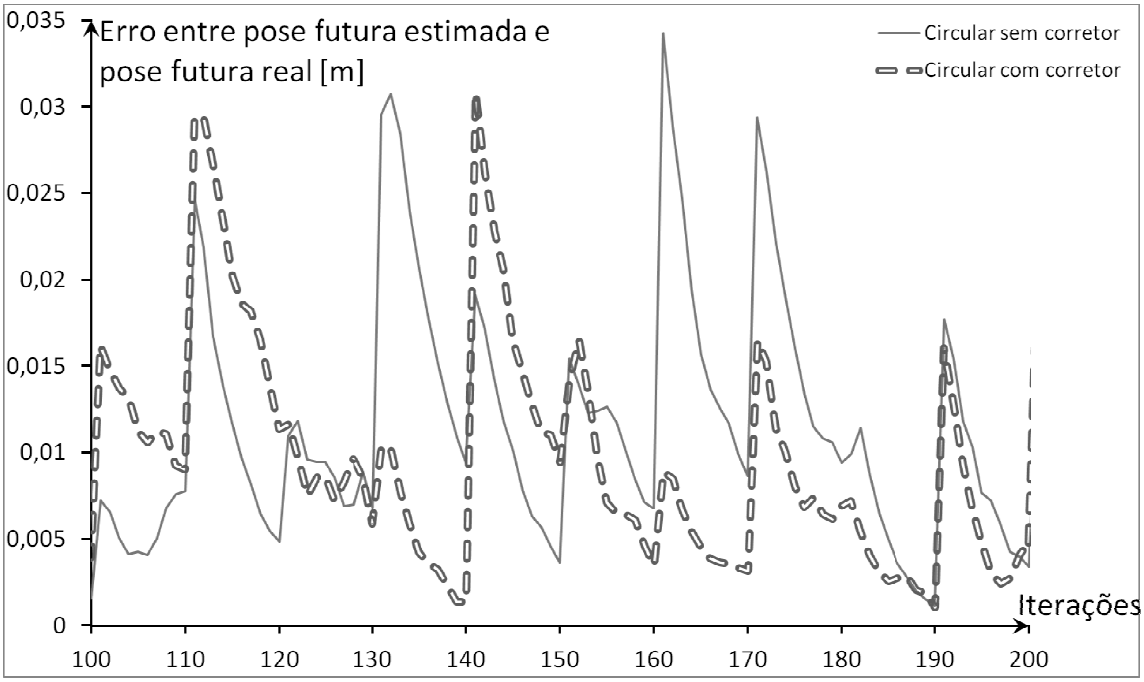


Figura 6.5 – Gráfico do erro entre as poses atuais e desejadas do manipulador robótico em relação ao objeto em movimento circular. Foi utilizado o filtro de Kalman e, em um dos casos, o corretor de *dead-times* (aplicado na pose do objeto).

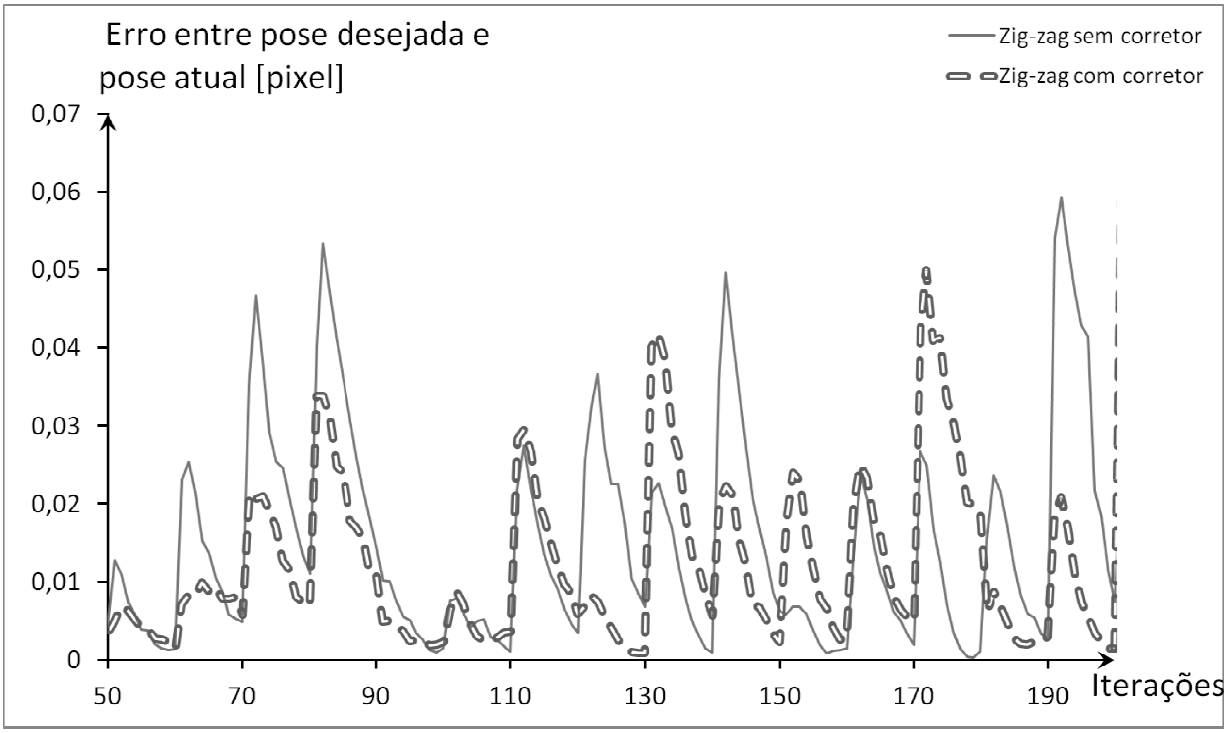


Figura 6.6 – Gráfico do erro entre as poses atuais e desejadas do manipulador robótico em relação ao objeto em movimento zig-zag. Foi utilizado o filtro de Kalman e, em um dos casos, o corretor de *dead-times* (aplicado na pose do objeto).

Nos três casos iniciais ficou difícil visualizar uma melhora significativa da utilização do método corretor de *dead-times* e a não utilização desse método. Com essa dificuldade foi necessário uma segunda análise, que consiste na verificação do resultado da estimativa da pose do objeto (com e sem o corretor de *dead-times*) antes desse sinal compor a lei de controle, sendo assim considerando uma pose mais próxima do real do objeto. O caso considerado foi a de movimentação linear/diagonal do objeto, que como demonstrado nos testes de estimativa de pose futura do objeto condiz com a maioria dos resultados dos outros tipos de movimentação. Esses dados podem ser observados na Figura 6.7.

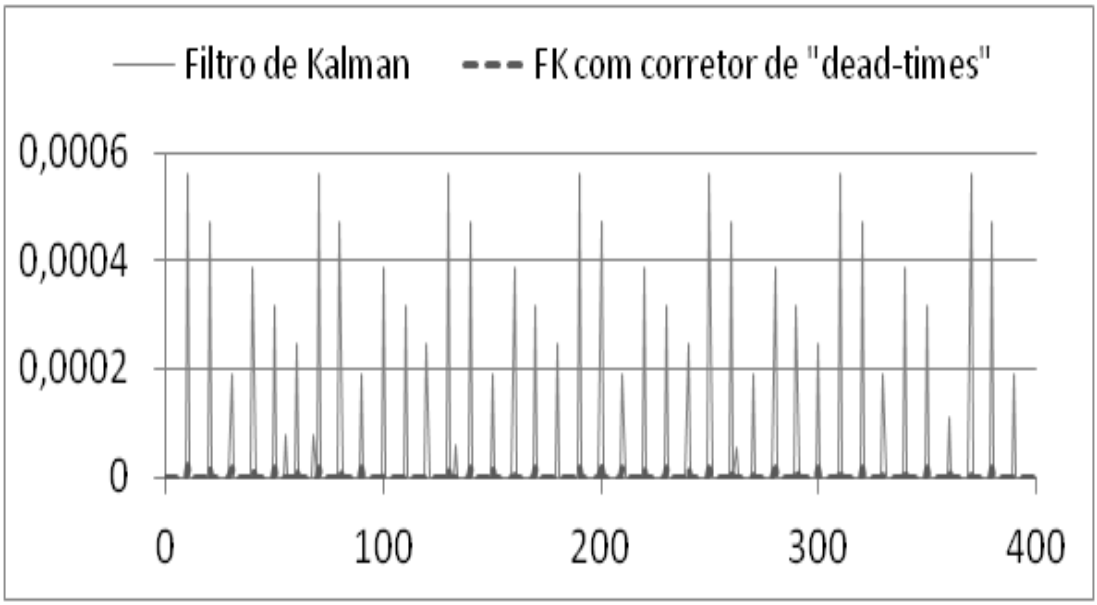


Figura 6.7 – Gráfico com dados da diferença entre o erro ideal do objeto e o erro estimado com o corretor de *dead-times* e sem o corretor de *dead-times*.

Essa análise permite verificar que o erro corrigido pelo método corretor de *dead-times* permite uma melhor aproximação do erro ideal (sem erro) a ser utilizado na lei de controle do controlador de velocidades e consequentemente irá gerar resultados mais confiáveis ao sistema.

6.2.3 Resultados das estimativas de velocidade com AFMA-6 (corretor de *dead-times* aplicado na variação do erro no tempo)

Nos gráficos das Figuras 6.8, 6.9 e 6.10 a primeira relação representa o erro entre a pose desejada e atual do manipulador robótico em relação ao objeto utilizando o filtro de Kalman com o método corretor de *dead-times*. A segunda representa o erro entre a pose desejada e atual do manipulador robótico em relação ao objeto com o filtro de Kalman, mas sem o método corretor.

Os gráficos das Figuras 6.8, 6.9 e 6.10 consideram respectivamente o objeto com movimento linear/diagonal, movimento circular e movimento em zig-zag (alterações drásticas de movimentação). A diferença entre esses testes e os anteriores com AFMA-6, é que nesses testes o método corretor de *dead-times* é aplicado na variação do erro no tempo e no outro método é aplicado na pode que compõe essa variação de erro no tempo. Em todos os casos ficou difícil analisar

alguma melhora nos métodos considerados, e assim foram necessários outros tipos de testes.

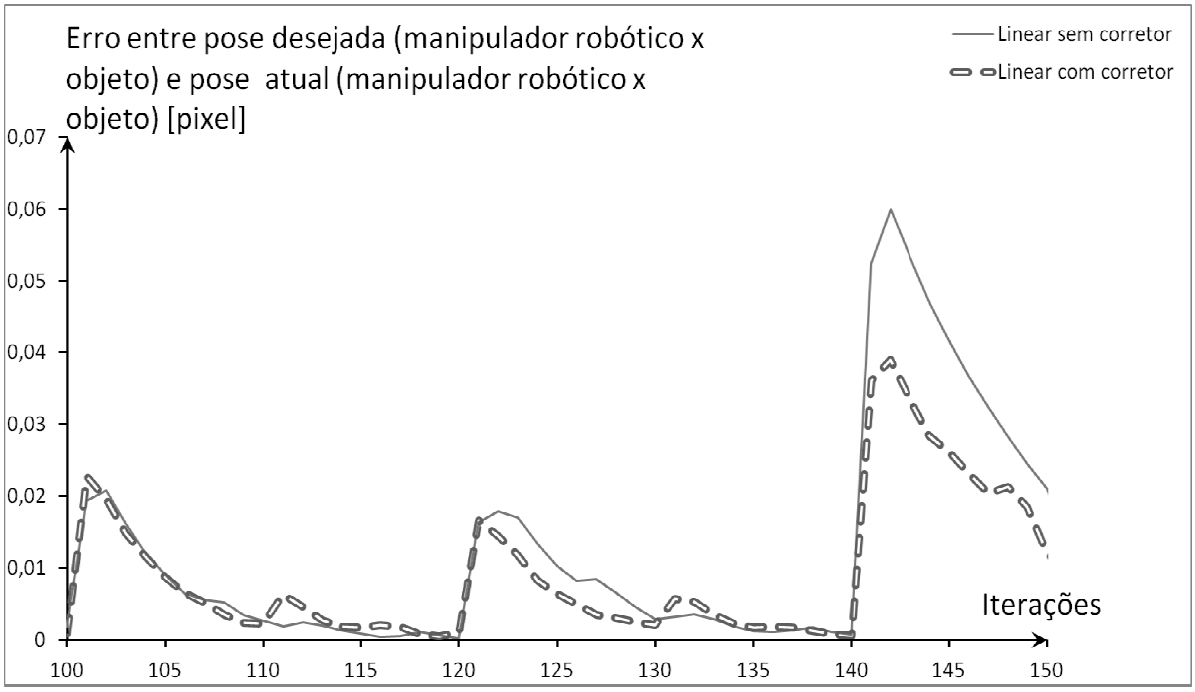


Figura 6.8 – Gráfico do erro entre as poses atuais e desejadas do manipulador robótico em relação ao objeto em movimento linear. Foi utilizado o filtro de Kalman e, em um dos casos, o corretor de *dead-times* (aplicado na variação do erro no tempo).

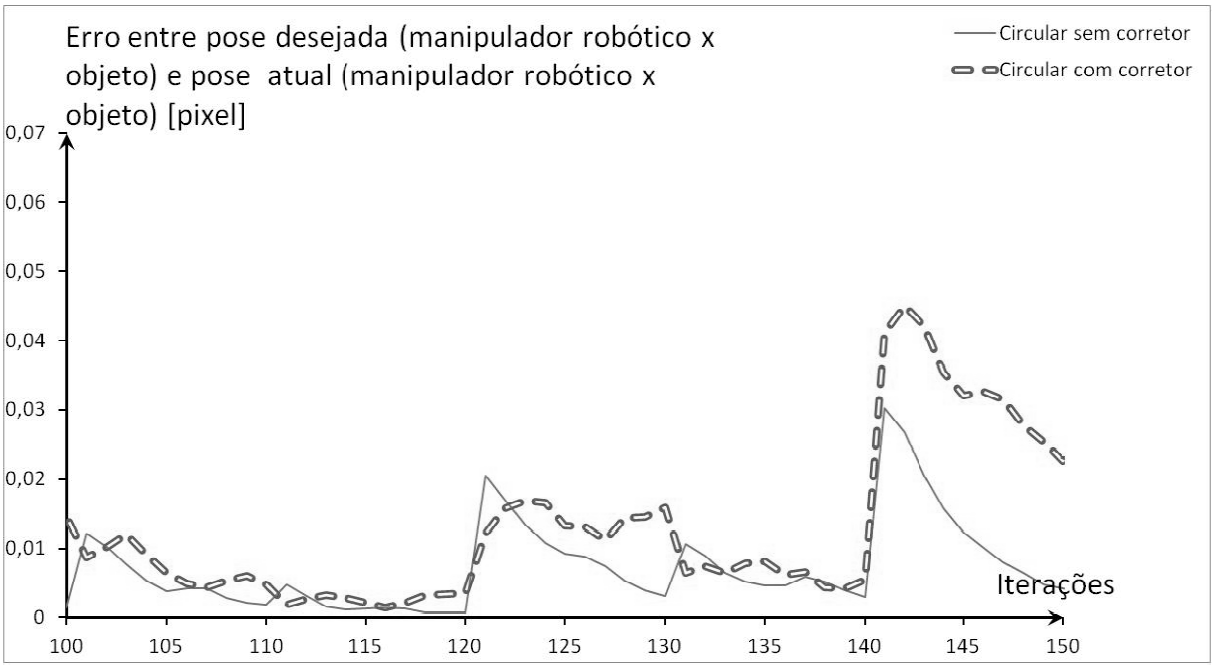


Figura 6.9 – Gráfico do erro entre as poses atuais e desejadas do manipulador robótico em relação ao objeto em movimento circular. Foi utilizado o filtro de Kalman e, em um dos casos, o corretor de *dead-times* (aplicado na variação do erro no tempo).

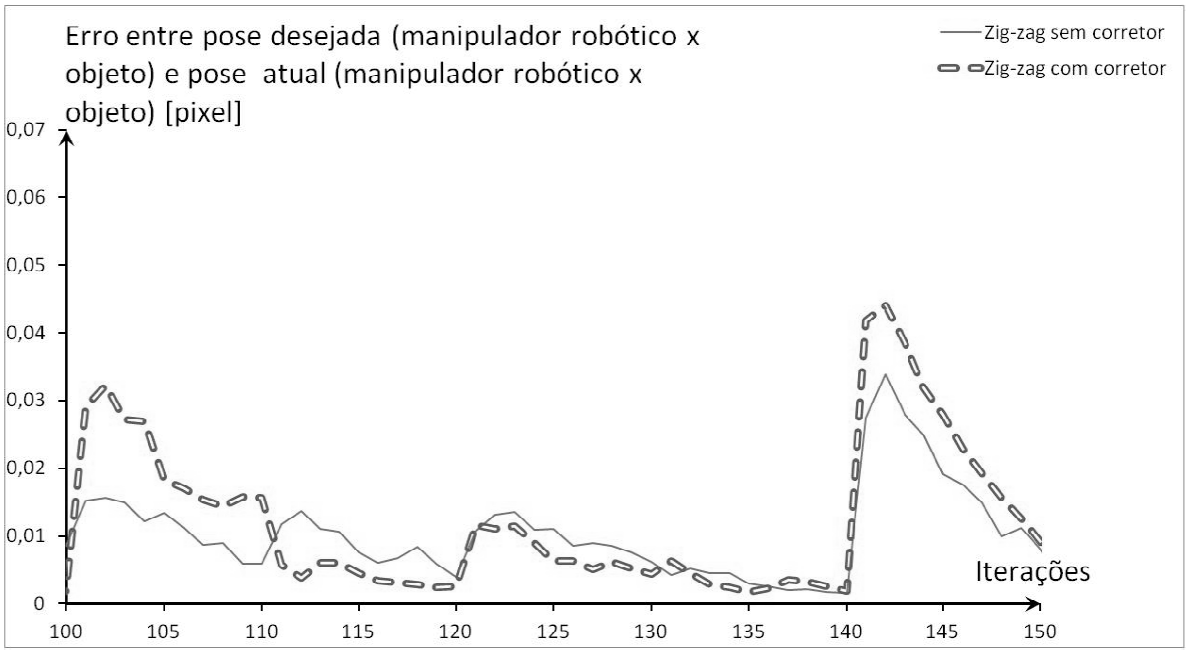


Figura 6.10 – Gráfico do erro entre as poses atuais e desejadas do manipulador robótico em relação ao objeto em movimento zig-zag. Foi utilizado o filtro de Kalman e, em um dos casos, o corretor de *dead-times* (aplicado na variação do erro no tempo).

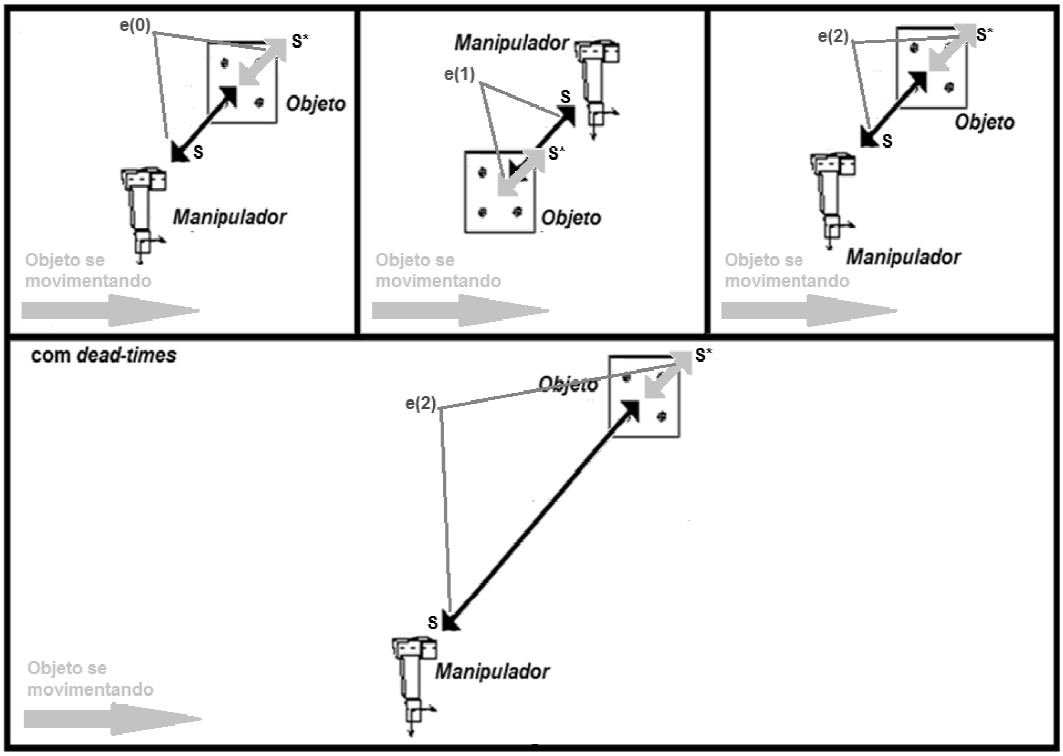


Figura 6.11 - Representação de variações do erro entre o manipulador robótico e objeto em instantes diferentes.

Na Figura 6.11 é possível observar diferentes variações (similar a um movimento de zig-zag) na pose atual com base na pose desejada do manipulador robótico em relação ao objeto. Essas variações são causadas pelas alterações de velocidades do manipulador robótico (causadas pelo controlador de velocidades) durante a estabilização desse manipulador em relação ao objeto.

Na Tabela 6.2 são demonstrados alguns valores de exemplo para o erro entre o manipulador robótico e objeto baseado na Figura 6.11. São considerados aplicações do filtro de Kalman e do método corretor de *dead-times*.

No instante *s(0)* supondo que o erro era $e(0)=(-1,-1)^T$, que o filtro de Kalman realizava operações com variação de erro $\dot{e}(0)=(2,2)^T$. Nesse instante a estimativa do erro realizada por filtro de Kalman vinha se mantendo e se mantém em um padrão de $X_{k|k}=(2,0,2,0)^T$ realizando uma estimativa de $X_{k|k-1}=(2,0,2,0)^T$, que representa a variação do erro *e* no tempo, ou seja, $\dot{e}$.

Tabela 6.2 - Exemplo do processo de estabilização do erro do manipulador robótico em relação ao objeto com aplicação do filtro de Kalman e método corretor de *dead-times*.

Instante *s*	erro *e*	Variação do erro $\dot{e}$	Estimativa por filtro de Kalman da etapa seguinte	Predição por filtro de Kalman da etapa seguinte
0	$(-1,-1)^T$	$(2,2)^T$	...	...
1	$(1,1)^T$	$(2, 2)^T$	...	$X_{k\|k-1}=(2,0,2,0)^T$
2	$(-1,-1)^T$	$(-2,-2)^T$	$X_{k\|k}=(-2,-1,-2,-1)^T$	$X_{k+1\|k}=(-2,-1,-2,-1)^T$
3	$(-3,-3)^T$	$(-2,-2)^T$	...	...
3 com 2 dead-times e aplicação do método corretor	$(-7,-7)^T$	$(-6,-6)^T$	...	...

No instante *s(1)* o erro alterou para $e(1)=(1,1)^T$, gerando a variação de erro $\dot{e}(1)= e(1)-e(0)= (1,1)^T-(-1,-1)^T=(2,2)^T$, e baseado nessa diferença o filtro de Kalman estimou como próxima variação de erro o valor de $\dot{e}(2)=X_{k|k-1}=(2,0,2,0)$.

No instante *s(2)* o erro alterou para $e(2)=(-1,-1)^T$, gerando a variação de erro $\dot{e}(2)= e(2)-e(2)= (-1,-1)^T-(1,1)^T=(-2,-2)^T$. Nesse instante é possível notar que houve diferença no valor e principalmente no sinal dos valores em relação ao a variação de erro anterior $\dot{e}(1)=(2,2)^T$. A predição do filtro de Kalman realizada nesse instante referente ao próximo instante como pode ser corrigida para $X_{k+1|k}=(-2,-1,-2,-1)^T$.

No instante s(3) o valor da variação do erro possivelmente seria $\dot{e}(3)= (-2,-2)^T$, porém é considerado a ocorrência de 2 *dead-times*. Um atraso de 2 *dead-times* deve fazer com que o método corretor de *dead-times* estime uma variação de erro de $\dot{e}(3)=(-6,-6)^T$.

No processo do método corretor de *dead-times* primeiro é aplicado a equação 55 gerando os valores de

$$X_{k|k-1}^* = F^2 * (-2,-1,-2,-1)^T = (-3.9,-0.98,-3.9,-0.98)^T.$$

Aqui é considerando que $F^2=F^d$

$$F^2 = \begin{bmatrix} 1 & 1,99 & 0 & 0 \\ 0 & 0,98 & 0 & 0 \\ 0 & 0 & 1 & 1,99 \\ 0 & 0 & 0 & 0,98 \end{bmatrix}$$

Posteriormente é aplicado a equação 33 obtendo os valores de

$$X_{k|k}^* = (-3.9,-0.98,-3.9,-0.98)^T + F^2 * ((-2,-1,-2,-1)^T – (2,0,2,0)^T) = (-3.9,-0.98,$$
$$-3.9,-0.98)^T + (-5.990-0.980-5.990-0.980)^T =$$
$$(-9.890, -1.960, -9.890, -1.960)^T$$

Esse erro é bem maior ao que deveria ter sido estimado, ou seja, $-9.890, -1.960, -9.890, -1.960)^T$ ao invés de (-6,-6).

6.3 Analise da variável adicional L

Durante os testes realizados também foi avaliado a variável L para situação de objetos em movimento de forma diagonal/linear. Essa variável compõe a equação 52 e alguns valores para essa variável podem ser observados na Tabela 6.3.

Tabela 6.3 – Amostra de dados para análise da variável L.

Iter	Próxima posição do objeto		Filtro de Kalman c/ corretor		d *"dead-times"*		L	
	x	y	X	y	PDI	TDD/PDD	x	Y
6	0,032	0,019	0,032	0,019	0	0	0	0
7	0,042	0,025	0,042	0,025	0	5	0	0
32	0,212	0,125	0,212	0,125	11	0	4,77E-18	-1,30E-18
112	0,783	0,461	0,783	0,461	8	3	4,77E-18	-1,30E-18
300	2,072	1,219	2,072	1,219	12	0	8,67E-19	-8,67E-19

Na Tabela 6.3 pode ser observado que os valores de L são extremamente pequenos e para muitas situações de rastreamento de objetos podem ser desconsiderados. Essa questão motivou a utilização das equações 33 e 34 de Lutteke e Franke (2013) ao invés das equações 53 e 54 obtidas nesse trabalho.

6.4 Comentários finais sobre resultados obtidos

Nesse capitulo foram demonstrados os experimentos, testes e análises realizadas nas condições de rastreamento de objetos prevendo a pose futura, prevendo a pose após a ocorrência de *dead-times* e analise de utilização de outro tipo de sinal como o erro do manipulador robótico em relação ao objeto.

Os resultados demonstraram que para a previsão futura de objeto o método corretor de *dead-times* conseguiu um bom desempenho para o caso de objeto com movimentação linear/diagonal e circular. No caso de alterações drásticas de

movimento do objeto como no caso de zig-zag o sistema não apresentou bons resultados.

Foram realizados alguns testes também para o caso de outro tipo de sinal a ser utilizado, como o erro da pose atual e desejada do manipulador robótico em relação ao objeto utilizado no controlador de velocidades de robôs manipuladores. Nesses testes foi identificado comportamento ruim do método para tratar *dead-times* o que demonstrou que o método não é eficiente para esse caso.

Em outro experimento foi considerado uma forma de utilização do método corretor de *dead-times* no controlador de velocidades para *dead-times* causados por visão computacional. Nesse caso foi utilizada a estimativa de pose do objeto após a ocorrência de *dead-times* (semelhante a estimativa de pose futura do objeto) em conjunto com a pose atual do manipulador robótico e velocidade para calcular o erro do objeto e posteriormente a velocidade do manipulador robótico no instante seguinte. Em resultados gerais o método com corretor de *dead-times* apresentou melhores resultados na média na maioria dos casos testados, porém esses resultados foram pouco expressivos em relação ao método sem corretor de *dead-times* (demonstrado nos gráficos e nas tabelas). Por final foi demonstrados que o método corretor de *dead-times* permitiu melhor aproximação do valor real da pose do objeto antes de ser calculado o erro entre o manipulador robótico e objeto.

Capítulo 7

CONCLUSÃO

Como apresentado nos capítulos anteriores desse trabalho, a servovisão faz o uso de câmeras como sensores de imagem e um circuito em malha fechada para realizar o controle de movimentação de robôs. As vantagens de servovisão incluem melhoria na precisão e maior flexibilidade nas operações robóticas.

Em tarefas desempenhadas por robôs que utilizam servovisão, são comuns problemas de *dead-times* que são atrasos no tempo causados por lentidão de processamento de imagens na transferência de dados e no processamento de outros dados do sistema como, por exemplo, em algoritmos auxiliares envolvidos na operação como o filtro de Kalman. O problema de *dead-times* pode afetar na execução de tarefas desempenhadas por robôs que fazem o uso de servovisão.

Uma aplicação de servovisão é no rastreamento de objetos por manipuladores robóticos em esteiras transportadoras que são comuns em ambientes industriais. A ocorrência de *dead-times* nesse tipo de operação pode fazer com que o manipulador robótico se posicione de maneira errada em relação ao objeto a ser manipulado causando atrasos e falhas na operação e podendo afetar negativamente no processo de produção.

Existem várias maneiras de se tratar *dead-times* em servovisão onde algumas abordagens consistem basicamente em evitar os *dead-times* através de *hardware* com processamento de alta capacidade e outras abordagens utilizam de técnicas (algoritmos) para compensar esses *dead-times* quando ocorrem. No presente trabalho o método explorado se encaixa na categoria de compensar *dead-times*.

Nesse trabalho foi desenvolvido/analisado um método para o tratamento de *dead-times,* sendo esse também o objeto principal desse trabalho. O método deve poder ser utilizado principalmente nas operações de rastreamento de objetos por

manipuladores robóticos em esteiras transportadoras utilizadas em ambientes industriais. A abordagem desenvolvida foi baseada nas equações do filtro de Kalman e também opera em conjunto com essa abordagem.

Foram analisadas as situações de estimativa de pose futura de um objeto em movimento e também duas formas distintas para a estimativa de velocidade (por um controlador de velocidades) utilizando servovisão. Em todos os métodos foi utilizado um filtro de Kalman para melhorar o sinal filtrado e o método para tratar *dead-times*.

Para realização dos testes foram desenvolvidos simuladores que colocam o objeto nas movimentações estudadas, simulando o movimento de esteiras transportadoras. Os simuladores foram desenvolvidos com o auxílio da ferramenta VISP para testar o método proposto na estimativa de posição futura do objeto e em ambos os casos da estimativa de velocidade. Nesses simuladores foram consideradas as possibilidades de diferentes situações e ocorrências de *dead-times* no sistema.

A abordagem apresentada nesse trabalho é de fácil implementação, porém os resultados foram significativos apenas para a situação de estimativa de pose futura do objeto e também quando estimado a pose atual do objeto após a ocorrência de *dead-times* no controlador de velocidades.

A estimativa de pose futura foi testada com diferentes valores considerando situações com ocorrências sucessivas para *dead-times* e a abordagem com filtro de Kalman e corretor de *dead-times* obteve um desempenho superior em relação ao método sem esse corretor quando o objeto movimentado de forma linear ou circular (com curvas suaves). Em alterações drásticas de movimento, ocorrem um ou mais *dead-times* devido a característica do filtro de Kalman e consequentemente devido o método desenvolvido ser baseado nas ultimas ocorrências o sistema não se demonstrou eficiente. Já no caso de aplicação do método proposto direto na variação do erro (da pose atual e desejada do manipulador robótico em relação ao objeto) no tempo, utilizada para estimar a velocidade, o método não obteve melhora significativa em relação a sua não utilização. Em compensação uma forma de utilizar o método proposto no controlador de velocidades é o aplicando após a ocorrência de *dead-times* por visão computacional para determinar a pose atual do objeto em movimento e dessa forma poder compor a variação de erro no tempo mais eficientemente.

Se utilizado em movimentos suaves, como movimentos lineares e circulares com curvas suaves o método pode ser aplicado em operações de rastreamento por servovisão para prever a posição do objeto, porém não é recomendável aplicar esse método diretamente a outros tipos de sinais como no caso de aplicação direta na variação de erro no tempo que é utilizada na lei de controle do controlador de velocidades do manipulador robótico.

O método não demonstrou a possibilidade de ser aplicado em qualquer tipo de movimentação (âmbito geral), porém, baseado nos resultados obtidos por simulação, é observado que o método pode ser aplicado (são necessários testes também em ambientes reais) no caso de esteiras transportadoras onde o objeto se movimenta de forma linear ou com curvas suaves.

7.1 Trabalhos Futuros

Algumas pesquisas e propostas de trabalhos futuros podem ser realizadas a partir do método proposto e algumas dessas são:

- Implementação do método proposto em um ambiente real (no trabalho foi implementado somente em simulações).
- Implementar o método proposto em conjunto com um *hardware* de alta capacidade para evitar e tratar *dead-times*.
- Desenvolvimento do método em conjunto á outras técnicas de tratamentos de *dead-times* e avaliar o desempenho.
- Avaliar o método proposto em outros tipos de sinais além da pose estimada futura do objeto e variação do erro utilizado no controlador de velocidades.
- Com a técnica estudada em conjunto com outras técnicas desenvolver um método para evitar dead-times no caso de movimentos aleatórios de objetos e para outros tipos de sinais.

REFERÊNCIAS

BAEZA, J. P.; MEDINA, F. T.; VÁZQUEZ, P. G. 2D Visual servoing with integration of multiple predictions of movement based on Kalman Filter. **World Congress**, Barcelona, Espanha, 15, 2002. 908-908.

BAI, L.; CHEN, F.; ZENG, X. Fuzzy Adaptive Proportional Integral and DIfferential with Modified Smith Predictor for Micro assembly Visual Servoing. **Information Technology Journal 8. Asian Network for Scientific Information**, 2009. 195-201.

BARRETO, J. P.; PEIXOTO, P.; BATISTA, J.; ARAUJO, H. Integrating Vision and Control to Achieve High Performance Active Tracking. **Tech. rep., TR-BAR-0202m ISR/DEEC**, 2002.

BAR-SHALOM, Y.; LI, X. R.; KIRUBARAJAN, T. **Estimation with Applications To Tracking and Navigation:** Theory Algorithms and Software. [S.l.]: John Wiley & Sons, 2001. 584 p.

BIRO, R. F.; MCMURRAY, G.; LIPKIN, H. Modular Visual Servo Tracking for Industrial Robots. **Modular Visual Servo Tracking for Industrial Robots**, 1997. Disponivel em: <http://helix.gatech.edu/papers/1997/Vservo.PDF>. Acesso em: 03 Novembro 2013.

BORANGIU, T.; ANTON, F. D.; DOGAR, A. Visual robot guidance in conveyor tracking with belt variables. **Automation Quality and Testing Robotics (AQTR), 2010 IEEE International Conference on**, 1, 28-30 Maio 2010. 1-6.

BOWTHORPE, M. et al. Smith Predictor Based Robot Control for Ultrasound-guided Teleoperated Beating-heart Surgery. **IEEE International Conference on Robotics and Automation (ICRA 2013)**, Alemanha, 2013. 5825-5830.

CERVERA, E. Distributed Visual Servoing: A Cross-Platform Agent-based Implementation reconstrução por visão estéreo. **2005 IEEE/RSJ International Conference on Intelligent Robots and Systems**, 2005.

CERVERA, E. JaViSS - Java-based Visual Servo Simulator. **JaViSS - Java-based Visual Servo Simulator**, 2010. Disponivel em: <http://www.robot.uji.es/research/projects/javiss>. Acesso em: 10 Janeiro 2013.

CHAUDHURI, S. S.; KONAR, A. Vision based target-tracking realized with mobile robots using extended Kalman filter. **Engineering Letters**, 14, n. 1, 2007.

CHAUMETTE, F.; BABEL, M.; KRUPA, A.; MARCHAND, E.; RIVES, P.; GIORDANO, P. R. Lagadic. **Ferramenta VISP**, 2013. Disponivel em: <http://www.irisa.fr/lagadic/visp/visp.html>. Acesso em: 01 Setembro 2013.

CHAUMETTE, F.; HUTCHINSON, S. Visual servo control, Part I: Basic approaches. **IEEE Robotics and Automation Magazine**, Dezembro 2006. 82-90.

CHAUMETTE, F.; HUTCHINSON, S. Visual servo control. Part II: Advanced approaches [Tutorial]. **Robotics & Automation Magazine, IEEE**, 14, n. 1, Março 2007. 109-118.

CHEN, S. Y. Kalman Filter for Robot Vision: A Survey. **IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS**, 59, Novembro 2012.

CHROUST, S.; BARRETO, J. P.; ARAÚJO, H.; VINCZE, M. Comparison of Control Structures for Visual Servoing. **ICAR2001 10th Internationl Conference on Advanced Robotics**, Agosto 2001.

CHROUST, S.; VINCZE, M. Improvement of the Prediction Quality for Visual Servoing with a Switching Kalman Filter. **The International Journal of Robotics Research**, 2003.

CORKE, P. Dynamic issues in robot visual-servo systems. **In International Symposium on Robotics Research**, 1995. 488–498.

CORKE, P. I. **Visual Control Of Robots:** High-Performance Visual Servoing. [S.l.]: CSIRO Division of ManufacturingTechnology,Australia, 1996.

CORKE, P. I. **Robotics, Vision and Control:** Fundamental Algorithms in MATLAB. [S.l.]: Springer, v. 1, 2011. 572 p.

CORKE, P. I.; GOOD, M. C. Controller design for high-performance visual servoing. **Proc. 12th World Congr. IFAC'93**, Sydney, Australia, Julho 1993. 395-398.

CORKE, P. I.; HUTCHINSON, S. A. Real-Time Vision, Tracking and Control. **Proceedings of the 2000 IEEE International Conference on Robotics & Automation.**, São Francisco, Abril 2000.

CRAIG, J. J. **Robótica**. 3. ed. [S.l.]: Pearson, 2013.

CRAWFORD, A. **Industrial Robots**. [S.l.]: The English Press, v.1, 2011. 88 p.

DE BEST, J. J. T. H.; MOLENGRAFT, M. J. G.; STEINBUCH, M. Direct dynamic visual servoing at 1 kHz by using the product as 1.5D encoder. **Control and Automation, 2009. ICCA 2009. IEEE International Conference on**, 9-11 Dezembro 2009. 361-366.

DENKER, A.; SABANOVIC, A.; KAYNAK, O. Vision-controlled robotic tracking and acquisition. **Intelligent Robots and Systems 94. Advanced Robotic Systems and the Real World, IROS 4. Proceedings of the IEEE/RSJ/GI International Conference on**, 3, 12-16 Setembro 1994. 2000-2006.

FUJIMOTO, H. Visual Servoing of 6 DOF Manipulator by Multirate Control with Depth Identification. **IEEE Conference on Decision and Control** , Maui, Hawai, USA, 2003.

FUJIMOTO, H.; HORI, Y. Visual Servoing Based on Intersample Disturbance Rejection by Multirate Sampling Control - Time Delay Compensation and Experimental Veri. **IEEE Conference on Decision and Control**, Orlando, Florida, USA, Dezembro 2001.

GORTCHEVA, E; GARRIDO, R; GONZÁLEZ, E; CARVALHO. Predicting a moving object position for visual servoing: theory and experiments. **INTERNATIONAL JOURNAL OF ADAPTIVE CONTROL AND SIGNAL PROCESSING**, 2001.

GROOVER, M. P. **Automation, Production Systems, and Computer-Integrated Manufacturing:** Pearson New International Edition. 3ª Edição. ed. [S.l.]: Pearson, 2013.

GROOVER, M. P. apud CARRARA, V. Apostila de Robótica Universidade Braz Cubas. **Apostila de Robótica**, 2013. Disponivel em: <http://www2.dem.inpe.br/val/homepage/cursos/rb_apostila.pdf>. Acesso em: 12 Dezembro 2013.

GUPTA, G. S.; MESSOM, C. H.; DEMIDENKO, S. Real-time identification and predictive control of fast mobile robots using global vision sensing. **IEEE Transactions on Instrumentation and Measurement**, 54, n. 1, 2005. 200-214.

HASHIMOTO, K.; KIMURA, H. Visual servoing with nonlinear observer. **Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on**, 1, 21-27 Maio 1995. 484-489.

HONG, P.; SAHLI, H.; COLON, E.; BAUDOIN, Y. Visual Servoing for Robot Navigation. **Third International Conference on Climbing and Walking Robots**, Alemanha, 2001. 255 – 264.

HUSSNAIN, Z.; SOUNKALO, D.; NICOLAS, A. Visual servoing in medical robotics. **IEEE International Workshop on Biomedical Robotics and Biomechatronics, BioRob'12**, Roma, Itália, 14 Junho 2012.

HUTCHINSON, S.; HAGER, G. D.; CORKE, P. I. A tutorial on visual servo control. **Robotics and Automation, IEEE Transactions on**, 12, n. 5, Outubro 1996. 651-670.

IWAZAKI, Y.; MURAKAMI, T.; OHNISKI, K. An approach of visual servoing control considering compensation of time delay. **Industrial Electronics, 1997. ISIE '97., Proceedings of the IEEE International Symposium on**, 7-11 Julho 1997. 723-728.

KALMAN, R. E. A New Approach to Linear Filtering and Prediction. **Transaction of the ASME—Journal of Basic Engineering**, Março 1960. 33-45.

KAWAMURA, A.; TAHARA, K.; KURAZUME, R.; HASEGAWA, T. Robust visual servoing for object manipulation with large time-delays of visual information. **Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on**, 7-12 Outubro 2012. 4797-4803.

KHO, Y.; KWON, Y. A study on the comparison of positioning tolerances depending on the state of vision sensors. **Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on**, 16-18 Maio 2012. 141-144.

KINBARA, I.; KOMADA, S.; HIRAI, J. Visual Servo of Active Cameras and Manipulators by Time Delay Compensation of Image Features with Simple On-line Calibration. **SICE-ICASE, 2006. International Joint Conference**, 18-21 Outubro 2006. 5317-5322.

LAGADIC, I. VISP. **VISP**, 2013. Disponivel em: <http://www.irisa.fr/lagadic/visp/visp.html>. Acesso em: Dezembro 2013.

LI, F.; XIE, H. L. Sliding Mode Variable Structure Control for Visual Servoing System. **International Journal of Automation and Computing**, 7 Agosto 2010. 317-323.

LI, Z. Visual servoing in robotic manufacturing systems for accurate positioning. **Thesis (M.A. Sc.)--Dept. of Mechanical and Industrial Engineering, Concordia University**, Canadá, 2007.

LIU, C.; HUANG, X.; WANG, M. Target tracking for visual servoing systems based on an adaptive Kalman filter. **Int. Journal of Advanced Robotic Systems**, 2012b.

LIU, K.; SUN, Z.; FUJII, M. Ellipse Detection Based Bin-Picking Visual Servoing System. **Pattern Recognition (CCPR), 2010 Chinese Conference on** , 21-23 Outubro 2010. 1-5.

LÜTTEKE, F.; FRANKE, J. Linear Kalman Filter for Dead Time Affected Measurement Signals Implemented in a Small Scale Automated Guided Vehicle. **Intelligent Autonomous Systems 12 - Advances in Intelligent Systems and Computing**, Berlin, 2013. 193–200.

MALIS, E. Survey of vision-based robot control. **ENSIETA European Naval Ship Design Short Course**, França, 2002.

MALIS, E.; RIVES, P. Robustness of image-based visual servoing with respect to depth distribution errors. **Proc. IEEE Int. Conf. Robotics and Automation**, 2003. 1056 -1061.

MARTINET, P.; BERRY, F.; GALLICE, J. Use of first derivative of geometric features in Visual Servoing. **Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on**, 22-28 Abril 1996. 3413-3419.

MASAR, I. An image processing system for visual servoing of mobile robots and object tracking. **The European DSP Education and Research Symposium, EDERS-2006**, Munique, Alemanha, Abril 2006.

MASON, M. T. Creation myths: The beginnings of robotics research. **Robotics & Automation Magazine, IEEE**, 19, 2012. 72–77.

MONDRÁGON, I. F. 3D pose estimation based on planar object tracking for UAVs control. **Robotics and Automation (ICRA), 2010 IEEE International Conference on**, 3-7 Maio 2010. 35-41.

MORENO-ARMENDÁRIZ, M. A.; RUBIO, E.; PÉREZ-OLVERA, C. A. Design and Implementation of a Visual Fuzzy Control in FPGA for the Ball and Plate System. **Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on** , 13-15 Dezembro 2010. 85-90.

NAGAHAMA, K.; HASHIMOTO, K.; NORITSUGU, T.; TAKAIAWA, M. Visual servoing based on object motion estimation. **Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on**, 1, 2000. 245-250.

OGATA, K. Engenharia de Controle Moderno , 2003. 800.

PAPANIKOLOPOULOS, N. P.; KHOSLA, P. K.; KANADE, T. Visual tracking of a moving target by a camera mounted on a robot: a combination of control and vision. **Robotics and Automation, IEEE Transactions on** , Fevereiro 1993. 14-35.

PIETERS, R. S. **Direct methods for vision-based robot control:** application and implementation. [S.l.]: Eindhoven: Technische Universiteit Eindhoven, 2013.

RAMAKOTI, N.; VINAY, A.; JATOTH, R. K. Particle Swarm Optimization Aided Kalman Filter for Object Tracking. **International Conference on Advances in Computing, Controll, and Telecommunication Technologies**, 2009.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence:** A Modern Approach. 3º Edição. ed. [S.l.]: Prentice Hall, 2009. 1152 p.

SAQUI, D.; SATO, F. C.; KATO, EDILSON R. R.; PEDRINO, E. C.; TSUNAKI, R. H. Mathematical Morphology Applied in Object Tracking on Position-Based Visual Servoing. **IEEE International Conference on Systems, Man, and Cybernetics**, 2013. 4030-4036.

SHIN, I. S.; NAM, S. H.; Yu, H. G.; ROBERTS, G. R.; MOON, S. B. Conveyor Visual Tracking using Robot Vision. **Florida Conference on Recent Advances in Robotics**, Maiami, Florida, 25-26 Maio 2006.

SICILIANO, B.; SCIAVICCO, L.; VILLANI, L.; ORIOLO, G. **Robotics:** Modelling, Planning and Control, Advanced Textbooks in Control and Signal Processing. 1. ed. [S.l.]: Springer, 2009. 632 p.

SIM, T. P.; HONG, G. S.; LIM, K. B. Modified Smith Predictor with DeMenthon-Horaud pose estimation algorithm for 3D dynamic visual servoing. **Robótica 2002**, 20, 21 Maio 2002. 615-624.

SLAVOV, T.; ROEVA, O. Genetic Algorithm Tuning of PID Controller in Smith Predictor for Glucose Concentration Control. **Int. J. Bioautomation, 2011**, 29 Julho 2011. 01-114.

SUH, H. I.; KIM, T. W. A visual servoing algorithm using fuzzy logics and fuzzy-neural networks. **Mechatronics**, 10, 2000. 1-18.

SUH, Y. S.; RO, Y. S.; KANG, H. J. H2 Filter for Time Delay Systems. **International Journal of Control, Automation, and Systems**, 4, Outubro 2006. 539-544.

TU, Y. W.; HO, M. T. Design and implementation of DSP and FPGA-based robust visual servoing control of an inverted pendulum. **Control Applications (CCA), 2010 IEEE International Conference on**, 8-10 Setembro 2010. 83-88.

VIDAL, C. P.-V. et al. Visual Control of Robots with Delayed Images. **Advanced Robotics**, 2009.

VIGORELLI, D.; BONFE, M.; FANTUZZI, C. Robotic manipulation experiments through visual feedback. **Mediterranean Conference on Control and Automation**, 2001.

WANG, F.; HUANG, D.; SHENG, G. A Pipelined Reconfigurable Architecture for Real-time Image Processing of Robot Vision Servoing. **Mechatronics and Automation, 2007. ICMA 2007. International Conference on** , 5-8 Agosto 2007. 2264-2269.

WELCH, G.; BISHOP, G. An Introduction to the Kalman Filter. **University of North Carolina at Chapel Hill Chapel Hill**, 1995.

WIRA, P.; URBAN, J. P. A New Adaptive Kalman Filter Applied To Visual Servoing Tasks. **Fourth Int. Conf. on Knowledge-Based Intelligent Engineering Systems & Allied Technologies, Proceedings of the KES'2000**, Reino Unido, 1 Setembro 2000.

WU, H.; LOU, L.; CHEN, C.-C.; KUHNLENZ, K.; HIRCHE, S. A Framework of Networked Visual Servo Control System with Distributed Computation. **Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on**, 60, Dezembro 2010. 1466–1471.

WUNSCH, P.; HIRZINGER, G. Real-Time Visual Tracking of 3-D Object with Dynamic Handling of Occlusion. **Proceedings of the 1997 IEE International Conference on Robotics and Automotation**, Alburquerque, Novo Mexico, Abril 1997.

XIAO, S.; LI, Y. Visual Servo Feedback Control of a Novel Large Working Range Micro Manipulation System for Microassembly. **Microelectromechanical Systems, Journal of**, PP, 2012. 1.

YOON, Y.; KOSAKA, A.; KAK, A. C. A New Kalman-Filter-Based Framework for Fast and Accurate Visual Tracking of Rigid Objects. **Robotics, IEEE Transactions on**, 24, Outubro 2008. 1238-1251.

Apêndice A

DESENVOLVIMENTO DETALHADO DAS EQUAÇÕES UTILIZADAS NESSE TRABALHO

Nesse apêndice são demonstrados todos os passos do desenvolvimento das equações de estimativa (53) e covariância (54) estudadas nesse trabalho.

$$X_{k|k}^{*} = X_{k|k-1}^{*} + F^{d} * (X_{k|k}\text{-}d - X_{k|k-1}\text{-}d) + L \tag{53}$$

$$P_{k|k}^{*} = P_{k|k-1}^{*} + F^{d} * (P_{k|k}\text{-}d - P_{k|k-1}\text{-}d) * F^{t\,d} + M \tag{54}$$

As equações numeradas foram apresentadas no conteúdo principal do trabalho enquanto as equações não numeradas foram apenas apresentadas nesse apêndice.

Foram desenvolvidas as equações de estimativa 35 e covariância 36.

$$X_{k|k} = X_{k|k-1} + K * (Z - (H * X_{k|k-1})) \tag{35}$$

$$P_{k|k} = P_{k|k-1} - K * (H * P_{k|k-1} * H^{T} + R) * K^{T} \tag{36}$$

O recurso de substituir algumas variáveis por equações que essas variáveis representam será amplamente utilizado na formulação das equações desse trabalho.

As equações 35 e 36 indicam que a estimativa de estado $X_{k|k}$ e da covariância do processo $P_{k|k}$ são obtidas respectivamente com base nas predições $X_{k|k-1}$ e $P_{k|k-1}$ e

estão á um passo a frente (por exemplo, de *k* para *k+1*) da estimativa de estado $X_{k-1|k-1}$ e estimativa da covariância do processo $P_{k-1|k-1}$.

Utilizando o raciocínio de estimar a próxima situação do estado, juntos as equações de 12 á 18 do filtro de Kalman e um processo recursivo é possível estimar o número de passos a frente que forem necessários, como por exemplo, 2 passos a frente, ou seja, $X_{k+2|k+2}$ e $P_{k+2|k+2}$. Para estimar $X_{k+2|k+2}$ e $P_{k+2|k+2}$ são necessários 2 execuções de predições e 2 execuções de estimativas.

Primeiramente são demonstrados os passos da equação do filtro de Kalman para estimativas de $X_{k+1|k+1}$ e $P_{k+1|k+1}$. Sendo assim é necessário o processo de predição (primeira execução de predição) do próximo valor em relação á $X_{k|k}$ e $P_{k|k}$, ou seja, $X_{k+1|k}$ e $P_{k+1|k}$

$$X_{k+1|k} = F * X_{k|k} + B * U$$

$$P_{k+1|k} = F * P_{k|k} * F^T + Q$$

Com essas predições é possível estabelecer a próxima etapa em relação á $X_{k|k}$ e $P_{k|k}$, as estimativas (primeira execução de estimativa) $X_{k+1|k+1}$ e $P_{k+1|k+1}$ conforme descritas nas equações a seguir

$$X_{k+1|k+1} = X_{k+1|k} + K * (Z - (H * (X_{k+1|k})))$$

$$P_{k+1|k+1} = P_{k+1|k} - K * (H * P_{k+1|k} * H^T + R) * K^T$$

Na próxima etapa foi aplicado um método de substituição devido ao desenvolvimento da formulação matemática do método proposto que será demonstrado em algumas próximas etapas. Substituindo os valores $X_{k+1|k}$ e $P_{k+1|k}$ nas equações anteriores respectivamente são obtidas duas novas equações. As equações 37 e 38 apresentadas no trabalho

$$X_{k+1|k+1} = (F * X_{k|k} + B * U) + K * (Z - (H * (F * X_{k|k} + B * U))) \quad (37)$$

$$P_{k+1|k+1} = (F * P_{k|k} * F^T + Q) - K * (H * (F * P_{k|k} * F^T + Q) * H^T + R) * K^T \quad (38)$$

Aplicando novamente um processo de predição (segunda execução de predição) para $X_{k+2|k+1}$ e $P_{k+2|k+1}$ são obtidas as equações

$$X_{k+2|k+1} = F * X_{k+1|k+1} + B * U$$

$$P_{k+2|k+1} = F * P_{k+1|k+1} * F^T + Q$$

Substituindo os valores de $X_{k+1|k+1}$ e $P_{k+1|k+1}$ são obtidas as equações

$$X_{k+2|k+1} = F * ((F * X_{k|k} + B * U) + K * (Z - (H * (F * X_{k|k} + B * U)))) + B * U$$

$$P_{k+2|k+1} = F * ((F * P_{k|k} * F^T + Q) - K * (H * (F * P_{k|k} * F^T + Q) * H^T + R) * K^T) * F^T + Q$$

Finalmente usando as predições $X_{k+2|k+1}$ e $P_{k+2|k+1}$ é possível estabelecer as estimativas $X_{k+2|k+2}$ e $P_{k+2|k+2}$ (segunda execução da estimativa).

$$X_{k+2|k+2} = X_{k+2|k+1} + K * (Z - (H * X_{k+2|k+1}))$$

$$P_{k+2|k+2} = P_{k+2|k+1} - K * (H * P_{k+2|k+1} * H^T + R) * K^T$$

Substituindo os valores de $X_{k+2|k+1}$ e $P_{k+2|k+1}$ nas equações anteriores são obtidas as equações 39 e 40 para estimativas de $X_{k+2|k+2}$ e $P_{k+2|k+2}$.

$$X_{k+2|k+2} = (F * ((F * X_{k|k} + B * U) + K * (Z - (H * (F * X_{k|k} + B * U)))) + B * U) + K * (Z - (H * (F * ((F * X_{k|k} + B * U) + K * (Z - (H * (F * X_{k|k} + B * U)))) + B * U))) \quad (39)$$

$$P_{k+2|k+2} = (F * ((F * P_{k|k} * F^T + Q) - K * (H * (F * P_{k|k} * F^T + Q) * H^T + R) * K^T) * F^T + Q) - K * (H * (F * ((F * P_{k|k} * F^T + Q) - K * (H * (F * P_{k|k} * F^T + Q) * H^T + R) * K^T) * F^T + Q) * H^T + R) * K^T \quad (40)$$

O processo de estimar $X_{k+2|k+2}$ e $P_{k+2|k+2}$ a partir de $X_{k|k}$ e $P_{k|k}$ descrito nas equações anteriores indica que foram avançadas duas etapas do filtro de Kalman durante o processo.

Com as equações 39 e 40 são aplicadas as distributivas e as equações ficam organizadas como a seguir

$$X_{k+2|k+2} = (F * F * X_{k|k} + F * B * U + B * U) + (F * K * Z - F * H * F * X_{k|k} - F * K * H * B * U + K * Z - K * H * F * F * X_{k|k} - K * H * F * B * U - K * H * F * K * Z + K * H * H * F * X_{k|k} + K * H * F * K * H * B * U - K * H * B * U)$$

$$P_{k+2|k+2} = (F * F * P_{k|k} * F^T * F^T + F * Q * F^T + Q) + (- F * K * H * F * P_{k|k} * F^T * H^T * K^T * F^T - F * K * H * Q * H^T * K^T * F^T - F * K * R * K^T * F^T - K * H * F * F * P_{k|k} * F^T * F^T * H^T * K^T - K * H * F * Q * F^T * K^T * H^T + K * H * F * K * H * F * P_{k|k} * F^T * H^T * K^T * F^T * H^T * K^T + K * H * F * K * H * Q * H^T * K^T * F^T * H^T * K^T + K * H * F * K * R * K^T * F^T * H^T * K^T - K * H * Q * H^T * K^T - K * R * K^T)$$

Nessas novas equações é possível observar a ocorrência de padrões como $F * B * U + B * U$ na primeira equação e $F * Q * F^T + Q$ na segunda equação. Isolando esses valores são definidas duas variáveis representadas pelas equações 41 e 42.

$$\Delta U = F * B * U + B * U \tag{41}$$

$$\Delta O = F * Q * F^T + Q \tag{42}$$

Essas variáveis são aplicadas ás equações 39 e 40 (já organizadas com as distributivas), gerando as equações 43 e 44.

$$X_{k+2|k+2} = (F * F * X_{k|k} + \Delta U) + (F * K * Z - F * H * F * X_{k|k} - F * K * H * B * U + K * Z - K * H * F * F * X_{k|k} - K * H * F * B * U - K * H * F * K * Z + K * H * H * F * X_{k|k} + K * H * F * K * H * B * U - K * H * B * U) \tag{43}$$

$$P_{k+2|k+2} = (F * F * P_{k|k} * F^T * F^T + \Delta O) + (- F * K * H * F * P_{k|k} * F^T * H^T * K^T * F^T - F * K * H * Q * H^T * K^T * F^T - F * K * R * K^T * F^T - K * H * F * F * P_{k|k} * F^T * F^T * H^T * K^T - K * H * F * Q * F^T * K^T * H^T + K * H * F * K * H * F * P_{k|k} * F^T * H^T * K^T * F^T * H^T * K^T + K * H * F * K * H * Q * H^T * K^T * F^T * H^T * K^T + K * H * F * K * R * K^T * F^T * H^T * K^T - K * H * Q * H^T * K^T - K * R * K^T) \tag{44}$$

Partindo das equações de predição $X_{k+2|k+1}$ e $P_{k+2|k+1}$, após a substituição dos valores de $X_{k+1|k+1}$ e $P_{k+1|k+1}$, propositalmente são substituídos os valores chegado as equações

$$X_{k+2|k+1} = F * ((F * (X_{k|k-1} + K * (Z - (H * X_{k|k-1}))) + B * U) + K * (Z - (H * (F * (X_{k|k-1} + K * (Z - (H * X_{k|k-1}))) + B * U)))) + B * U$$

$$P_{k+2|k+1} = F * ((F * (P_{k|k-1} - K * (H * P_{k|k-1} * H^T + R) * K^T) * F^T + Q) - K * (H * (F * (P_{k|k-1} - K * (H * P_{k|k-1} * H^T + R) * K^T) * F^T + Q) * H^T + R) * K^T) * F^T + Q$$

Agora também são aplicadas as distributivas sendo desenvolvidas as equações

$$X_{k+2|k+1} = (F * F * X_{k|k-1} + F * B * U + B * U) + (F * F * K * Z - F * K * F * H * X_{k|k-1} + F * K * Z - F * K * H * F * X_{k|k-1} - F * K * H * F * K * Z + F * K * H * F * K * H * X_{k|k-1} - F * K * H * B * U)$$

$$P_{k+2|k+1} = (F * F * P_{k|k-1} F^T * F^T + F * Q * F^T + Q) + (- F * F * K * H * P_{k|k-1} * H^T * K^T F^T * F^T - F * F * K * R * K^T F^T * F^T - F * K * H * F * P_{k|k-1} F^T * H^T * K^T * F^T + F * K * H * F * K * H * P_{k|k-1} * H^T * K^T F^T * H^T * K^T * F^T + F * K * H * F * K * R * K^T F^T * H^T * K^T * F^T - F * K * H * Q * H^T * K^T * F^T - F * K * R * K^T * F^T)$$

Substituindo os valores de ΔU e ΔO nas equações anteriores é possível chegar as equações 45 e 46

$$X_{k+2|k+1} = (F * F * X_{k|k-1} + \Delta U) + (F * F * K * Z - F * K * F * H * X_{k|k-1} + F * K * Z - F * K * H * F * X_{k|k-1} - F * K * H * F * K * Z + F * K * H * F * K * H * X_{k|k-1} - F * K * H * B * U) \quad (45)$$

$$P_{k+2|k+1} = (F * F * P_{k|k-1} F^T * F^T + \Delta O) + (- F * F * K * H * P_{k|k-1} * H^T * K^T F^T * F^T - F * F * K * R * K^T F^T * F^T - F * K * H * F * P_{k|k-1} F^T * H^T * K^T * F^T + F * K * H * F * K * H * P_{k|k-1} * H^T * K^T F^T * H^T * K^T * F^T + F * K * H * F * K * R * K^T F^T * H^T * K^T * F^T - F * K * H * Q * H^T * K^T * F^T - F * K * R * K^T * F^T) \quad (46)$$

Agora são isoladas as variáveis ΔU e ΔO e são obtidas

$$\Delta U = X_{k+2|k+1} - (F * F * X_{k|k-1}) - (F * F * K * Z) + (F * K * F * H * X_{k|k-1}) - (F * K * Z) + (F * K * H * F * X_{k|k-1}) + (F * K * H * F * K * Z) - (F * K * H * F * K * H * X_{k|k-1}) + (F * K * H * B * U)$$

$$\Delta O = P_{k+2|k+1} - (F * F * P_{k|k-1} F^T * F^T) + (F * F * K * H * P_{k|k-1} * H^T * K^T F^T * F^T) + (F * F * K * R * K^T * F^T * F^T) + (F * K * H * F * P_{k|k-1} F^T * H^T * K^T * F^T) - (F * K * H * F * K * H * P_{k|k-1} * H^T * K^T F^T * H^T * K^T * F^T) - (F * K * H * F * K * R * K^T F^T * H^T * K^T * F^T) + (F * K * H * Q * H^T * K^T * F^T) + (F * K * R * K^T * F^T)$$

Os valores de ΔU *e* ΔO das equações anteriores são substituídos nas equações 43 e 44 e são geradas as equações

$$X_{k+2|k+2} = (F * F * X_{k|k}) + (X_{k+2|k+1}) - (F * F * X_{k|k-1}) - (F * F * K * Z) + (F * F * K * H * X_{k|k-1}) + (F * F * K * H * X_{k|k-1}) + (F * F * K * K * H * Z) - (F * F * K * K * H * H * X_{k|k-1}) - (F * H * H * X_{k|k}) + (K * Z) - (F * F * K * H * X_{k|k}) - (F * K * H * B * U) - (F * K * K * H * Z) + (F * K * H * H * X_{k|k}) + (F * H * K * K * H * B * U) - (K * H * B * U)$$

$$P_{k+2|k+2} = (F * F * P_{k|k} * F^T * F^T) + (P_{k+2|k+1}) - (F * F * P_{k|k-1} F^T * F^T) + (F * F * K * H * P_{k|k-1} * H^T * K^T * F^T * F^T) + (F * F * K * R * K^T * F^T * F^T) + (F * F * K * H * P_{k|k-1} * H^T * K^T * F^T * F^T) - (F * F * K * K * H * H * P_{k|k-1} * H^T * H^T * K^T * K^T * F^T * F^T) - (F * F * K * K * H * R * H^T * K^T * K^T * F^T * F^T) - (F * F * K * H * P_{k|k} * K^T * H^T * F^T * F^T) - (F * F * K * H * P_{k|k} * H^T * K^T * F^T * F^T) - (F * K * H * Q * K^T * H^T * F^T) + (F * F * K * K * H * H * P_{k|k} * H^T * H^T * K^T * K^T * F^T * F^T) + (F * K * K * H * H * Q * H^T * H^T * K^T * K^T * F^T) + (F * K * K * H * R * H^T * K^T * K^T * F^T) - (K * H * Q * H^T * K^T) - (K * R * K^T)$$

Agora são definidas duas novas variáveis. A variável *L* representada pela equação 49 e a variável *M* representada pela equação 50 são as diferenças entre essas equações e a formulação de Lutteke e Franke (2013).

$$L = - (F * F * K * Z) + (F * F * K * H * X_{k|k-1}) + (F * F * K * H * X_{k|k-1}) + (F * F * K * K * H * Z) - (F * F * K * K * H * H * X_{k|k-1}) - (F * H * H_x * X_{k|k}) + (K * Z) - (F * F * K * H * X_{k|k}) - (F * K * H * B * U) - (F * K * K * H * Z) + (F * K * H * H * X_{k|k}) + (F * H * K * K * H * B * U) - (K * H * B * U) \quad (49)$$

$$M = (F * F * K * H * P_{k|k-1} * H^T * K^T * F^T * F^T) + (F * F * K * R * K^T * F^T * F^T) + (F * F * K * H * P_{k|k-1} * H^T * K^T * F^T * F^T) - (F * F * K * K * H * H * P_{k|k-1} * H^T * H^T * K^T * K^T * F^T * F^T) - (F * F * K * K * H * R * H^T * K^T * K^T * F^T * F^T) - (F * F * K * H * P_{k|k} * K^T * H^T * F^T * F^T) - (F * F * K * H * P_{k|k} * H^T * K^T * F^T * F^T) - (F * K * H * Q * K^T * H^T * F^T) + (F * F * K * K * H * H * P_{k|k} * H^T * H^T * K^T * K^T * F^T * F^T) + (F * K * K * H * H * Q * H^T * H^T * K^T * K^T * F^T) + (F * K * K * H * R * H^T * K^T * K^T * F^T) - (K * H * Q * H^T * K^T) - (K * R * K^T) \quad (50)$$

Utilizando *L* e *M* nas equações anteriores a essas no apêndice são obtidas duas novas equações.

$$X_{k+2|k+2} = X_{k+2|k+1} + F^2 * (X_{k|k} - X_{k|k-1}) + L \quad (47)$$

$$P_{k+2|k+2} = P_{k+2|k+1} + F^2 * (P_{k|k} - P_{k|k-1}) * F^{T2} + M \quad (48)$$

No caso foram consideradas duas etapas a frente de $X_{k|k}$ e $P_{k|k}$, porém essas equações podem ser descritas em termos de n etapas. Como demonstrado nas equações 51 e 52.

$$X_{k+n|k+n} = X_{k+n|k+n-1} + F^n * (X_{k|k} - X_{k|k-1}) + L \quad (51)$$

$$P_{k+n|k+n} = P_{k+n|k+n-1} + F^n * (P_{k|k} - P_{k|k-1}) * F^{t\,n} + M \quad (52)$$

A partir das equações 51 e 52 podem ser convertidas para considerar *d dead-times*, e finalmente duas novas equações (equação 53 e 54) são estabelecidas para estimar de forma mais correta os valores de $X_{k|k}^*$ e $P_{k|k}^*$.

$$X_{k|k}^* = X_{k|k-1}^* + F^d * (X_{k|k}\text{-}d - X_{k|k-1}\text{-}d) + L \quad (53)$$

$$P_{k|k}^* = P_{k|k-1}^* + F^d * (P_{k|k}\text{-}d - P_{k|k-1}\text{-}d) * F^{t\,d} + M \quad (54)$$