Ayrton Galindo Bernardino Florêncio

**Sistema de estabilização de vídeo baseado em
acelerômetro, filtragem robusta e algoritmo de busca
de três etapas ¹**

Dissertação de Mestrado apresentada à Escola de Engenharia
de São Carlos da Universidade de São Paulo, como parte dos
requisitos para obtenção do título de Mestre em Ciências,
Programa de Engenharia Elétrica.

Área de Concentração: Sistemas Dinâmicos
Orientador: Prof. Dr. Marco Henrique Terra

São Carlos,
2015

¹Trata-se da versão corrigida da dissertação. A versão original se encontra disponível na EESC/USP que aloja o programa de Pós-Graduação de Engenharia Elétrica.

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

F632s Florêncio, Ayrton Galindo Bernardino
Sistema de estabilização de vídeo baseado em
acelerômetro, filtragem robusta e algoritmo de busca de
três etapas / Ayrton Galindo Bernardino Florêncio;
orientador Marco Henrique Terra. São Carlos, 2015.

Dissertação (Mestrado) - Programa de Pós-Graduação
em Engenharia Elétrica e Área de Concentração em
Sistemas Dinâmicos -- Escola de Engenharia de São
Carlos da Universidade de São Paulo, 2015.

1. Visão computacional. 2. Unidade de medidas
inerciais. 3. Estimativa de movimento. 4. Vibração. 5.
Filtro robusto. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **AYRTON GALINDO BERNANRDIO FLORÊNCIO**.

Título da dissertação: "Sistema de estabilização de vídeo baseado em acelerômetro, filtragem robusta e algoritmo de busca de três etapas".

Data da defesa: 17/09/2015

Comissão Julgadora:

Prof. Associado **Marco Henrique Terra (Orientador)**
(Escola de Engenharia de São Carlos/EESC)

Prof. Dr. **Fernando Cesar Lizarralde**
(Universidade Federal do Rio de Janeiro/UFRJ)

Prof. Dr. **Valdir Grassi Junior**
(Escola de Engenharia de São Carlos/EESC)

Resultado:

Aprovado

APROVADO

aprovado

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica:
Prof. Associado **Luis Fernando Costa Alberto**

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Paulo César Lima Segantine**

Agradecimentos

Ao Prof. Dr. Marco Henrique Terra pela oportunidade, confiança, orientação e paciência dispensadas ao longo deste trabalho.

Ao Prof. Dr. Roberto Santos Inoue pela atenção, apoio e importantes contribuições para realização deste trabalho.

À instituição Universidade de São Paulo em São Carlos, aos professores das disciplinas pela dedicação demonstrada e aos funcionários do Departamento de Engenharia Elétrica, Daniel, Marisa, Leonardo e Rafael, que de alguma forma, direta ou indiretamente, contribuíram com o trabalho.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo auxílio financeiro.

À Embraer pelo apoio e espaço concedido para a conclusão deste trabalho.

À Ísis, minha amada esposa, pelo companheirismo, amor e incentivo constantes empregados durante este período.

Aos pais da Ísis, Ciro e Maria, que me ajudaram durante minha permanência em São Carlos.

À minha família, especialmente aos meus pais, Almir e Elzeny, e aos meus grandes amigos que, mesmo distantes, demonstraram todo o amor e apoio para que eu seguisse em frente e por acreditarem no meu sucesso.

E, finalmente, a Deus por permitir que eu esteja aqui hoje e por me dar toda a força necessária para que este trabalho fosse realizado.

Resumo

FLORENCIO, A. G. B. (2015). Sistema de estabilização de vídeo baseado em acelerômetro, filtragem robusta e algoritmo de busca de três etapas. Tese (Mestrado), Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2015.

Câmeras embarcadas em sistemas robóticos móveis com sensoriamento visual geralmente são afetadas pelo movimento de seu suporte quando é necessária a aquisição de imagens com alta confiabilidade. Algumas câmeras disponíveis no mercado já estão equipadas com sistemas de estabilização de imagens, implementados nas lentes da câmera ou no sensor de imagem. Esses sistemas são relativamente caros para serem instalados em sistemas robóticos móveis aéreos ou terrestres de pequeno porte. Outro conceito que tem sido utilizado para alcançar a estabilização consiste em adotar sensores inerciais, algoritmos de visão computacional e aplicação de filtros digitais para estimativa e suavização de movimento. Esta dissertação tem por finalidade apresentar um sistema de estabilização digital de vídeo em conjunto com um acelerômetro para detectar movimentos da câmera. Considera-se que a câmera está instalada sobre plataforma instável ou em veículo em movimento, assim, utiliza-se filtragem robusta para minimizar o efeito da vibração sobre a câmera. Vários experimentos são realizados adicionando vibração em vídeo por meio de simulações em computador. Considera-se também experimentos em ambiente real com a câmera montada em um suporte dentro de um veículo em movimento. A principal contribuição deste trabalho é a realização de um estudo comparativo para analisar a vantagem de se usar uma abordagem robusta entre os métodos de filtragem comumente utilizados em sistemas de estabilização de vídeo.

Palavras-chave: *Visão computacional, sensor inercial, estimativa de movimento, vibração, filtro robusto.*

Abstract

FLORENCIO, A. G. B. (2015). Video stabilization system based on accelerometer, robust filtering and three-step search algorithm. Dissertation (Master of Science), School of Engineering of São Carlos, University of São Paulo, São Carlos, 2015.

Embedded cameras in robotic systems usually are susceptible to movements of your basis. The measurements can be affected mainly when they should obtain high performance in actual applications. Although some cameras available on the market are already equipped with an optical image stabilization system, implemented either in the camera lenses or in the image sensor. They are usually expensive to be installed into small aerial or land robotic systems. This dissertation presents a video stabilization system coupled with an accelerometer for motion detection at an unstable platform. It is embedded in a vehicle. Robust filtering is used to minimize vibration effects on the camera. Several experiments were performed, adding either vibrations via computer simulations and taking images from the camera with the vehicle moving. The main contribution of this work is the accomplishment of a comparative study to analyse the advantages of applying a robust approach among others filtering methods that frequently are used to video stabilizing systems.

Keywords: *Computer vision, inertial sensor, motion estimation, vibration, robust filtering.*

Publicações

1. Florencio, A. G. B.; Terra, M. H.; Inoue, R. S.; Junior, V. G. (2013). Sistema de Estabilização Digital de Vídeo Baseado em Algoritmo TSS e Filtro Robusto BDU para Sistemas Robóticos. *XI Simpósio Brasileiro de Automação Inteligente, SBAI*, 13 a 17 de Outubro de 2013, Fortaleza-CE.

Lista de Figuras

1.1	Diagrama de blocos do sistema proposto.	5
2.1	Componentes essenciais de um sistema de aquisição de imagem digital.	8
2.2	Representação de um quadro, onde i e j são eixos para representar o número de linha e coluna, respectivamente. Como a figura indica, a origem $(i, j) = (1, 1)$ está no canto superior esquerdo do quadro.	8
2.3	Representação da letra 'a' em uma imagem digital com resolução 18×18	9
2.4	Estimativa de movimento usando <i>block matching</i>	12
2.5	Exemplo de regiões ilegíveis: circulado em azul está uma região de alta suavidade; circulado em vermelho está uma região de alta rugosidade.	14
2.6	Exemplo de aplicação do algoritmo TSS. A figura mostra um gráfico dos possíveis deslocamentos de $p = (n_i, n_j)$ do bloco I_p dentro da área de busca. O gráfico mostra que foi encontrado um bloco de posição $p''' = (n_i + \Delta l_p, n_j + \Delta c_p)$ semelhante ao bloco de posição p do instante anterior, onde $(\Delta l_p, \Delta c_p) = (4 + 0 + 1, -4 - 2 - 1) = (5, -7)$	15
3.1	Os gráficos mostram os níveis de vibrações nas proximidades de um aeroporto devido à aproximação de uma aeronave para pouso. Veja referência [48].	20
3.2	Sistema massa-mola-amortecedor básico.	21
3.3	Resposta em frequência. Experimentos: (a) rua pavimentada à 60 km/h, (b) rua pavimentada à 30 km/h, (c) rodovia à 90 km/h, (d) rua asfaltada à 30 km/h.	22
3.4	Exemplo diferença dos dados coletados pelo acelerômetro (círculos) e algoritmo TSS (losangos).	22
3.5	Arranjo câmera-acelerômetro no sistema de coordenadas da câmera.	23

3.6	Exemplo de aplicação do algoritmo TSS com a primeira etapa fornecida pelos cálculos dos dados do acelerômetro. A figura indica o deslocamento $(\Delta l_p, \Delta c_p) = (\Delta y_{im} + 3 - 1, \Delta x_{im} - 3 - 1) = (\Delta y_{im} + 2, \Delta x_{im} - 4)$	26
4.1	Exemplo de um filtro média móvel. Em (a) há um pulso retangular imerso em ruído. Em (b) e (c) é mostrado o sinal quando um filtro média móvel de ordem 11 e 51 foram respectivamente aplicados. A figura foi extraída da referência [34]. . . .	30
4.2	Gráfico para representação das variáveis do algoritmo genético.	35
5.1	Resultados da estimativa de movimento quando aplica-se vibração artificial.	38
5.2	Resultados da estimativa de movimento em um caso real.	39
5.3	Resposta da posição da câmera afetada por vibração. A curva em azul mostra o resultado para o sistema proposto em malha aberta e a curva em preto mostra o resultado para o sistema proposto em malha fechada.	40
5.4	Atualização do diagrama de blocos do sistema proposto, apresentado na Figura 1.1.	41
5.5	Componente horizontal do deslocamento dos quadros em <i>pixels</i> para o experimento em que o veículo faz uma curva de 90 graus.	43
5.6	Zoom da Figura 5.5 entre os instantes 3,5 e 4,5 segundos.	43
5.7	Componente vertical do deslocamento dos quadros em <i>pixels</i> para o experimento em que o veículo faz uma curva de 90 graus.	44
5.8	Componente horizontal do deslocamento dos quadros em <i>pixels</i> para o experimento em que o veículo atravessa uma ondulação transversal.	45
5.9	Componente vertical do deslocamento dos quadros em <i>pixels</i> para o experimento em que o veículo atravessa uma ondulação transversal.	46
5.10	Componente horizontal do deslocamento dos quadros em <i>pixels</i> para o experimento em que o veículo executa movimentos em zigue-zague.	47
5.11	Componente vertical do deslocamento dos quadros em <i>pixels</i> para o experimento em que o veículo executa movimentos em zigue-zague.	48
5.12	Componente horizontal do deslocamento dos quadros em <i>pixels</i> para o experimento em que o veículo executa curva leve e constante.	49
5.13	Componente vertical do deslocamento dos quadros em <i>pixels</i> para o experimento em que o veículo executa curva leve e constante.	50
5.14	Variância dos filtros. Análise com $T = 50$ experimentos.	50
A.1	Circuito filtro Butterworth de segunda ordem e frequência de corte 200 Hz.	61

A.1	Acelerômetro MMA7361. Figura retirada do manual do fabricante.	81
A.2	Microcontrolador Arduino Uno. Figura retirada do manual do fabricante.	83
A.3	Raspberry Pi. Foto: http://www.pcmag.com.br/us/article2/0,2817,2407058,00.asp	83
A.4	<i>Mini PC-camera</i> . Foto retirada do fabricante.	84

Lista de Tabelas

2.1	Principais características dos algoritmos. Tabela extraída da referência [23]	17
2.2	Cálculo do erro dos algoritmos. Os ambientes (a), (b), (c) e (d) contém vibração onde a variância σ_X^2 é 10, 20, 40 e 80, respectivamente.	17
A.1	Descrição dos pinos do MMA7361.	82

Lista de abreviaturas e siglas

DVS – *Digital Video Stabilization*

SLAM – *Simultaneous Localization And Mapping*

TSS – *Three-Step Search*

SCV – Sinal Composto de Video

GMV – *Global Motion Vector*

LMV – *Local Motion Vector*

OpenCV – *Open Source Computer Vision Library*

NTSS – *New Three-Step Search*

FSS – *Four-Step Search*

DS – *Diamond Search*

IMU – *Inertial Measurement Unit*

DFT – *Discrete Fourier Transform*

LASER – *Light Amplification by Stimulated Emission of Radiation*

IIR – *Infinite Impulse Response*

FMM – Filtro Média Móvel

BDU – *Bounded Data Uncertainties*

FKRR – Filtro de Kalman Robusto Recursivo

Sumário

Publicações	xi
Lista de Figuras	xv
Lista de Tabelas	xvii
Lista de Abreviaturas e Siglas	xix
1 Introdução	1
1.1 Objetivo	2
1.2 Revisão bibliográfica	3
1.2.1 Movimentação da câmera	3
1.2.2 Filtragem	4
1.3 Metas	5
1.4 Disposição dos capítulos	5
2 Estimativa de movimento em visão computacional	7
2.1 Aquisição de imagem digital	7
2.2 Vídeo digital	10
2.3 Detecção de movimento	10
2.4 Estimativa de movimento	11
2.5 Estimativa de movimento por <i>block matching</i>	12
2.5.1 Regiões ilegíveis	13
2.5.2 Algoritmo TSS	13
2.5.3 Geração do GMV	15
2.5.4 Outros algoritmos	16
3 Conjunto algoritmo TSS e acelerômetro	19

3.1	Vibração	19
3.2	Princípio de operação do acelerômetro	20
3.2.1	Análise em frequência	21
3.3	Disposição do conjunto câmera e acelerômetro	23
4	Filtragem	29
4.1	Filtro média móvel	29
4.2	Filtro IIR adaptativo	29
4.3	Filtro de Kalman	30
4.4	Filtro BDU	31
4.5	Filtro de Kalman robusto recursivo	33
4.6	Algoritmo genético	34
5	Experimentos e resultados	37
5.1	Configuração, testes e resultados do algoritmo TSS	37
5.2	Configuração, testes e resultados do conjunto acelerômetro e câmera	39
5.3	Filtragem	41
6	Conclusões	51
	Referências Bibliográficas	53
A	Modelagem do acelerômetro	59
A.1	Detecção de movimento	60
A.2	Pré-filtro	61
B	Códigos	63
	Anexo	81
A	Equipamentos	81
A.1	MMA7361	81
A.2	Arduino Uno	82
A.3	Raspberry Pi	82
A.4	Câmera	84

Capítulo 1

Introdução

O avanço da tecnologia tem proporcionado a evolução de sistemas robóticos mais compactos, leves e baratos, incentivando cada vez mais a busca e apoio à pesquisa e desenvolvimento. Os benefícios desses sistemas trazem auxílio para execução de tarefas e missões para diversos setores, como saúde, indústria e militar [1, 2]. Algumas das aplicações mais importantes de sistemas robóticos nessas áreas consiste em segurança, automação e inspeção de processos, auxílio à cirurgia e reabilitação, autonomia dos sistemas de transportes.

Sistemas robóticos são dotados de sensores, computadores e atuadores que permitem sua interação com o mundo pela percepção do ambiente e ações por movimentação. Essencialmente, esses sistemas são munidos de sensores com capacidade de percepção visual e deslocamento para viabilizar a execução de tarefas de navegação [3, 4] evitando colisões e condições inseguras durante trajetória. Dentre eles, a câmera [5, 6], *SONAR* [7] (Navegação e Determinação da Distância pelo Som, do inglês *Sound Navigation and Ranging*), *LASER* [8] (Amplificação da Luz por Emissão Estimulada de Radiação, do inglês *Light Amplification by Stimulated Emission of Radiation*), *RFID* [9] (Identificação por radiofrequência, do inglês *Radio-Frequency IDentification*) e sensores inerciais, como acelerômetros e giroscópios.

A câmera é bastante utilizada em navegação capturando imagens do ambiente que são processadas para identificar parâmetros que delimitam a tomada de decisão do robô. A câmera está sendo cada vez mais requisitada devido ao baixo custo, fácil implementação, evolução e tendência de semicondutores de tecnologia CMOS (semicondutor metal-óxido complementar, do inglês *Complementary Metal-oxide Semiconductor*) e CCD (dispositivo de carga acoplada, do inglês *Charge-Coupled Device*) [10, 11, 12] e alta flexibilidade de aplicações. Por exemplo: *dead-reckoning* [13] baseado em fluxo óptico [14]; SLAM (localização e mapeamento simultâneos, do inglês *Simulta-*

neous Localization and Mapping) com multi-câmeras [15]; veículos autônomos guiados por visão [16]; busca de características e padrões de terrenos [17].

Em geral, a câmera está sujeita a vibrações causadas pela interação da movimentação do robô com o ambiente prejudicando a qualidade de sensoriamento e o desempenho de algoritmos de processamento de imagens. Plataformas como o *gimbal* e conjunto de lentes móveis são largamente aplicados para minimizar vibrações ou movimentos indesejáveis da câmera. Tais soluções podem se tornar tecnicamente inviáveis devido a limitações do projeto, e.g. volume da estrutura e alto preço, veja a referência [18]. Recentemente, sistemas de estabilização baseados em visão computacional e filtragem digital, mais conhecidos como sistemas DVS (estabilização digital de vídeo, do inglês *Digital Video Stabilization*), têm sido empregados para detectar e amortecer os movimentos causados por distúrbios externos.

Os sistemas DVS são bastante requisitados por causa da boa resposta em tempo real e à tendência da produção de processadores de alto desempenho. No entanto, esses sistemas são suscetíveis quando a câmera sofre maior impacto da vibração ou mudança de trajetória¹. A análise do impacto da vibração e a percepção da mudança de trajetória podem ser feitas com o uso de acelerômetros, os quais têm a capacidade de indicar a frequência de ressonância e amplitude da vibração. Desta forma, é necessário que o filtro utilizado seja robusto o suficiente para suavizar vibrações e absorver os efeitos de mudança de trajetória do veículo.

1.1 Objetivo

O objetivo deste trabalho é apresentar um sistema de estabilização de vídeo para aplicação em veículos autônomos terrestres. A técnica utilizada é composta de acelerômetros e algoritmos de visão computacional e filtragem robusta para estimativa e suavização de movimento. O sistema proposto é concebido para ter baixo custo de processamento. Deve ser capaz de executar processamentos em tempo real, útil para ser usado em veículos autônomos de baixo custo.

¹Quando o veículo, no qual a câmera está embarcada, passa por um instante de transição de um tipo de movimento e executa outro. Por exemplo, uma aeronave está em uma trajetória de movimento retilíneo e inicia uma rolagem, rampa, etc. Isso faz com que o campo visual da câmera mude completamente.

1.2 Revisão bibliográfica

As próximas seções resumem o conteúdo técnico que será abordado nesta dissertação e a organização dos próximos capítulos.

1.2.1 Movimentação da câmera

A estimativa de movimento da câmera deve ser precisa e executada na ordem de milissegundos reproduzindo o movimento da câmera com o menor erro possível para que a filtragem seja aplicada corretamente. Existem vários algoritmos de visão computacional na literatura que apresentam um procedimento para fazer a estimativa de movimento da câmera. Dentre eles estão os algoritmos baseados em *block matching* [19, 20, 21, 22, 23], o fluxo óptico [24, 25] e a integração entre sistemas inerciais e visuais [26, 27, 28, 29, 30].

Em [23] é realizada uma pesquisa comparativa entre os algoritmos baseados em *block matching*. Neste, conclui-se que são algoritmos eficientes para fazer estimativa de movimentos em que a vibração presente não é muito intensa. Porém, a precisão do cálculo diminui quando a vibração se torna mais acentuada. Já em [25] é feito um estudo comparativo entre algoritmos baseados em fluxo óptico. Todos dependem fielmente das condições de iluminação do ambiente, pois a variação da intensidade da luz prejudica bastante o desempenho desses algoritmos.

Em [26] é apresentado um sistema de estabilização para sistemas robóticos humanóides baseando-se no reflexo vestibulo-ocular onde foi utilizada uma IMU (unidade inercial de medida, do inglês *Inertial Measurement Unit*) junto ao algoritmo de fluxo óptico. Em [27] é adotado um sistema de 4 camadas, sendo que a primeira camada detecta a vibração usando uma IMU e realiza um contra-movimento comandado por um *gimbal* motorizado. A segunda camada amortece a vibração usando dispositivos mecânicos, a terceira camada usa um sistema de estabilização óptico e a quarta camada usa software para filtragem da vibração remanescente. Outras aplicações são obtidas pelo uso de sensores inerciais com algoritmos de visão computacional, como apresentado em [28]. Nesta referência é proposta uma fusão de dados de uma IMU com algoritmo de visão para executar SLAM sem a necessidade de visão estereo².

²Técnica para calcular distância de objetos a partir de imagens do campo visual de duas câmeras em posições diferentes entre si.

1.2.2 Filtragem

Alguns métodos de filtragem mais utilizados em sistemas DVS são: filtro IIR (resposta ao impulso infinito, do inglês *Infinite Impulse Response*) adaptativo [31, 32]; DFT (transformada de Fourier discreta, do inglês *Discrete Fourier Transform*) [33], filtro média móvel [34, 35] e o filtro de Kalman [36, 37, 38, 39].

Em [31] foi proposto um filtro IIR adaptativo de primeira ordem, mostrando baixa complexidade de implementação, contudo, há dificuldade na escolha do fator de amortecimento. A aplicação da DFT em [33] garante robustez na eliminação de movimentos indesejáveis da câmera, porém, não é aplicável em sistemas projetados para atuarem em tempo real. Em [35] foram comparados alguns filtros digitais usualmente implementados em DSP's (processador de sinal digital, do inglês *Digital Signal Processor*). Entre eles o filtro passa baixas *Butterworth*, filtro passa baixas *Chebyshev* e filtro média móvel, o qual apresenta melhor resposta.

O filtro de Kalman fornece ótimo desempenho para tratar de sistemas sujeitos a ruídos e tem um algoritmo simples sem a necessidade de armazenamento de grandes bancos de dados. A abordagem desse filtro em [36, 37, 38, 39] apresentam resultados eficientes para cancelamento de vibrações e preservação de movimentos intencionais da câmera. Contudo, o filtro de Kalman pode ser prejudicado quando o modelo de espaço de estado apresenta incertezas.

Filtros similares à estrutura do filtro de Kalman foram desenvolvidos tratando o modelo de espaço de estado que contém incerteza limitada. O filtro robusto BDU (incerteza de dados limitado, do inglês *Bounded Data Uncertainties*), proposto em [40] e o FKRR (Filtro de Kalman Robusto Recursivo), proposto em [41], oferecem essas características. Algumas aplicações focando a abordagem robusta com esses filtros são apresentadas em [42, 43, 44].

Em [42] é apresentado um sistema para coordenação de robôs móveis para navegação em ambientes externos e dinâmicos³. Em [43] é feita a estimativa da atitude e posição de um robô helicóptero para coordenação com robôs móveis com rodas. Em [44] o acompanhamento de trajetória de um alvo móvel durante uma gravação de vídeo.

³Ambiente aberto, *outdoor*, i.e. há maior complexidade para sensoriamento devido à variação de iluminação, movimentação de objetos ou mesmo surgimento de obstáculos.

1.3 Metas

Neste trabalho é implementado um sistema DVS com algoritmo TSS (busca de três etapas, do inglês *Three-step Search*) junto a um acelerômetro para obter a estimativa de movimento da câmera com baixo esforço computacional. Para amortecimento da vibração, um estudo comparativo entre métodos de filtragem é realizado para analisar o comportamento de cada um dentre os experimentos realizados. A Figura 1.1 mostra a estrutura do sistema proposto neste trabalho, os detalhes serão explicados nos próximos capítulos.

Na Figura 1.1, o bloco Compensador significa a reprodução das imagens contendo o contramovimento resultado da estimativa da vibração. Essa etapa não será abordada neste trabalho.

Figura 1.1: Diagrama de blocos do sistema proposto.

1.4 Disposição dos capítulos

Capítulo 2: são introduzidos alguns conceitos sobre imagem e vídeos digitais, o algoritmo para estimativa de movimentos da câmera e resultados.

Capítulo 3: é apresentada a integração do acelerômetro com o algoritmo TSS e discutida a vantagem desse sistema.

Capítulo 4: são apresentados os métodos de filtragem aplicados no sistema DVS e a análise comparativa.

Capítulo 5: são discutidos os resultados, assim como as vantagens de se usar a integração de sensores para estimativa de movimento e a abordagem de filtragem robusta para eliminar

vibrações.

Capítulo 6: conclusões e propostas para extensão do estudo.

Capítulo 2

Estimativa de movimento em visão computacional

Este capítulo apresenta uma breve explicação dos conceitos de imagem e vídeo digitais e suas representações para aplicação neste trabalho. Em seguida é discutido o procedimento para detecção de movimento da câmera, o algoritmo TSS (busca por três etapas, do inglês *Three-step Search*) e uma análise comparativa com outros algoritmos baseados em *block matching*.

2.1 Aquisição de imagem digital

O processo de aquisição de imagem digital é efetuado em três etapas [45]: percepção da luz; conversão analógica-digital e armazenamento em computador. Veja a ilustração desse processo na Figura 2.1. A percepção da luz ocorre quando a luz refletida por objetos no ambiente atravessa a lente da câmera e atinge uma grade de fotosensores organizados matricialmente. A luz que incide em cada fotosensor produz uma diferença de potencial proporcional à intensidade da luz inserida no mesmo. O sinal nos terminais de cada fotosensor é transmitido em série⁶ a uma placa de captura onde é feita a conversão analógica-digital. Cada elemento digital provindo de cada fotosensor é chamado de *pixel* (elemento de imagem, do inglês *picture element*) e, tipicamente, é representado por 1 *byte*: há 256 níveis de cinza, onde '0' e '255' correspondem ao preto e branco. Por fim, é feito o armazenamento em computador de todos os *pixels* em um arquivo representando a imagem digital.

⁶A transmissão em série resulta em um sinal analógico chamado de sinal composto de vídeo (SCV), veja a referência [45].

Os *pixels* de uma imagem digital, denominado tecnicamente por quadro, são organizados no espaço bidimensional representados por linhas e colunas, que serão referenciadas pelas letras '*i*' e '*j*'. A resolução de um quadro é referenciada por $L \times C$, onde L e C representam o número de linhas e colunas. A Figura 2.2 mostra a representação de um quadro.

Figura 2.1: Componentes essenciais de um sistema de aquisição de imagem digital.

Figura 2.2: Representação de um quadro, onde i e j são eixos para representar o número de linha e coluna, respectivamente. Como a figura indica, a origem $(i, j) = (1, 1)$ está no canto superior esquerdo do quadro.

Neste trabalho, o valor de cada *pixel* de um quadro I é referenciado por $I(i, j)$, onde o índice (i, j) indica o número de linha e coluna de determinado *pixel* de um quadro I . A matriz abaixo mostra a representação numérica do conjunto de *pixels* que preenche um quadro.

$$I = \begin{bmatrix} I(1, 1) & I(1, 2) & \cdots & I(1, C) \\ I(2, 1) & I(2, 2) & \cdots & I(2, C) \\ \vdots & \vdots & \ddots & \vdots \\ I(L, 1) & I(L, 2) & \cdots & I(L, C) \end{bmatrix}$$

A Figura 2.3 exemplifica a representação de uma imagem digital de resolução 18×18 mostrando o valor do nível de cinza de cada *pixel*.

255	255	255	255	255	252	140	131	131	140	154	255	255	255	255	255	255	255
255	255	255	255	112	8	56	97	78	0	2	32	154	255	255	255	255	255
255	255	227	51	0	121	255	255	255	155	2	2	4	167	255	255	255	255
255	255	121	0	0	121	255	255	255	255	109	0	0	116	255	255	255	255
255	255	108	0	0	108	255	255	255	255	151	0	0	82	255	255	255	255
255	255	146	0	10	146	255	255	255	255	178	0	0	82	255	255	255	255
255	255	255	243	250	255	255	255	255	255	157	0	0	82	255	255	255	255
255	255	255	255	255	255	255	255	151	130	0	0	0	83	255	255	255	255
255	255	255	255	255	151	135	26	30	135	124	0	0	85	255	255	255	255
255	255	255	143	108	0	56	124	255	255	173	0	0	87	255	255	255	255
255	255	135	0	0	86	255	255	255	255	173	0	0	88	255	255	255	255
255	201	0	0	7	250	255	255	255	255	173	0	0	89	255	255	255	255
255	147	0	0	131	255	255	255	255	255	173	0	0	92	255	255	255	255
255	132	0	0	131	255	255	255	255	255	170	0	0	93	255	255	255	255
255	147	0	0	43	255	255	255	255	249	60	0	0	63	255	255	255	255
255	223	18	7	7	77	144	152	127	15	169	12	0	0	135	148	177	255
255	255	144	18	11	5	5	2	54	174	255	166	0	0	0	0	134	255
255	255	255	255	135	124	135	150	255	255	255	255	177	140	140	175	255	255

Figura 2.3: Representação da letra 'a' em uma imagem digital com resolução 18×18 .

2.2 Vídeo digital

Um vídeo digital é definido por uma sequência de quadros que são adquiridos em instantes de tempo

$$t_k = t_0 + k/f, \quad (2.1)$$

sendo t_0 o instante de tempo da aquisição do primeiro quadro, $k = 0, 1, \dots$. f é a frequência de aquisição de quadros. Tipicamente, os vídeos capturam quadros a uma frequência de 30 Hz , correspondente ao padrão NTSC (Comitê Nacional do(s) Sistema(s) de Televisão, do inglês *National Television System(s) Committee*) [46], i.e. o intervalo de tempo de aquisição de quadros é $\Delta t = (1/30) \text{ s}$.

Neste trabalho, cada *pixel* é referenciado em um vídeo digital por $I(i, j)_k$. O índice (i, j) indica o número de linha e coluna de determinado *pixel* e k indica o instante de tempo em que o quadro I foi adquirido.

2.3 Detecção de movimento

Assumindo que as condições de iluminação não variam no tempo k , as mudanças de imagens são causadas por um movimento relativo entre a câmera e a cena: a câmera poderia se mover em frente a uma cena estática; partes da cena poderiam se mover em frente de uma câmera imóvel; ou, em geral, a câmera e os objetos da cena estão se movendo com velocidades diferentes. Em [14] foram levantadas cinco heurísticas para identificar pontos correspondentes em um pequeno intervalo de tempo:

1. *Maximum velocity*. Do mesmo modo que as velocidades dos objetos no mundo são limitadas, a velocidade dos objetos projetados na placa de fotosensor também é limitada. Portanto, há um deslocamento máximo w em um instante de tempo Δt para todos os *pixels* de um quadro.
2. *Small velocity change*. A velocidade e sua direção calculadas entre dois instantes k e $k + 1$ não devem mudar bruscamente entre os próximos instantes $k + 1$ e $k + 2$.
3. *Common motion*. Há uma coerência no movimento espacial no instante de tempo Δt para um conjunto de pontos na imagem.
4. *Consistent match*. Dois pontos de uma imagem não correspondem a um único ponto de uma

outra imagem, caso não ocorra oclusão⁷.

5. *Known motion*. Se um modelo do mundo pode fornecer informações sobre movimento de objetos, a percepção de movimentos pode ser prevista e reconhecida.

Portanto, seguindo essas proposições, cada *pixel* $I(i', j')_k$ corresponderia a um *pixel* $I(i, j)_{k-1}$, onde $(i', j') = (i + \Delta l, j + \Delta c)$. Δl e Δc é o deslocamento do quadro I entre os instantes de tempo k e $k + 1$ na vertical e horizontal, respectivamente. Assim, conclui-se a seguinte relação:

$$I(i', j')_{k+1} = I(i' - \Delta l, j' - \Delta c)_k. \quad (2.2)$$

Contudo, no mundo real, alguns requisitos para análise de movimento devem ser considerados:

- As condições de iluminação podem mudar entre dois quadros adjacentes mesmo Δt sendo pequeno;
- Devido à imagem ser uma projeção 2-D do mundo, os movimentos 3-D relativos entre a câmera e os objetos causam diferenças na cena observada em dois quadros adjacentes;
- Objetos que fazem parte da cena se movem com relação ao restante da cena, modificando-as entre dois instantes de tempo;
- Rotação da câmera ou de objetos presentes na cena distorcem todo ou parte do campo visual;
- Se há uma região com *pixels* de mesmo valor em um quadro adquirido no instante k , quais são os pixels correspondentes no quadro adquirido no instante $k + 1$?

2.4 Estimativa de movimento

A estimativa de movimento da câmera consiste em encontrar o deslocamento da mesma no plano de sua placa de fotossensores, ou seja, detecta-se o deslocamento vertical e horizontal do campo visual entre quadros consecutivos adquiridos à taxa de 30 *Hz*.

Quando se trabalha em ambientes dinâmicos é possível observar três tipos de movimentos capturados pela câmera: o movimento de objetos presentes no campo visual, o movimento intencional e não-intencional da câmera. O movimento intencional é causado pela translação ou rotação da câmera devido ao caminho percorrido de seu transporte. Já o movimento não-intencional consiste

⁷Surgimento de algum objeto à frente do que está sendo observado.

na vibração do campo visão da câmera, afetada pela interação de sua plataforma com o ambiente. A composição desses três movimentos em um único vetor é chamado de GMV (vetor de movimento global, do inglês *Global Motion Vector*), o qual é estimado pelo algoritmo de visão. Já o movimento intencional da câmera é adquirido separadamente após a filtragem do GMV, veja **Capítulo 4**.

2.5 Estimativa de movimento por *block matching*

A técnica de estimativa de movimento por *block matching* é caracterizada por dividir o quadro em várias partições, essas também chamadas de blocos. Desta forma, a estimativa de movimento é feita individualmente sobre cada bloco do quadro para obter um valor médio do deslocamento compondo o GMV. Os blocos estão estruturados como uma matriz retangular de tamanho 16×16 , o qual foi definido através da eficiência apresentado nos resultados dos experimentos deste trabalho e pelas abordagens nas referências [31, 32].

Os blocos são referenciados por I_p . $p = (n_i, n_j)$ é a indicação do *pixel* que representa a origem do bloco, similar à representação de um quadro visto na Figura 2.2. A estimativa de deslocamento de um bloco I_{p_k} é feita procurando uma região nas proximidades de $I_{p_{k+1}}$ que tenha mesmas dimensões e maior semelhança com o bloco I_{p_k} . Esta procura é feita em uma área de busca com dimensões $(2w + 16) \times (2w + 16)$ *pixels* concêntrica a $I_{p_{k+1}}$, veja Figura 2.4. w corresponde ao máximo deslocamento possível de qualquer bloco.

Figura 2.4: Estimativa de movimento usando *block matching*.

Se for feita uma busca completa para encontrar o deslocamento de um bloco I_{p_k} no quadro seguinte, será necessário $(2w + 1)^2$ comparações dentro da área de busca. O algoritmo TSS faz uma busca em 25 pontos, i.e. a economia de processamento é de $((2w + 1)^2 - 25)/w^2$. Por exemplo, para $w = 10$ o algoritmo TSS proporciona uma redução de 94,3% de consumo de processamento comparado à busca exaustiva.

2.5.1 Regiões ilegíveis

Antes de apresentar o algoritmo utilizado, é preciso verificar se cada bloco de um quadro está em uma região ilegível, veja próximo parágrafo. Esta etapa é importante para evitar falsas estimativas de movimento dos blocos, podendo adicionar maiores erros no cálculo do movimento global da câmera.

Uma região é considerada ilegível quando há homogeneidade ou mudanças muito bruscas entre *pixels* adjacentes por qualquer região de um quadro, veja Figura 2.5. Essas regiões são chamadas de alta suavidade e alta rugosidade, respectivamente, e são verificadas pelas inequações que seguem:

$$\sqrt{\sum_{i=n_i+1}^m |I_p(i, j) - I_p(i-1, j)| + \sum_{j=n_j+1}^n |I_p(i, j) - I_p(i, j-1)|} < th_1 \text{ e} \quad (2.3)$$

$$\sqrt{\sum_{i=n_i+1}^m |I_p(i, j) - I_p(i-1, j)| + \sum_{j=n_j+1}^n |I_p(i, j) - I_p(i, j-1)|} > th_2, \quad (2.4)$$

sendo $m = n_i + 16$, $n = n_j + 16$, th_1 e th_2 *thresholds* predefinidos pelo projetista que delimitam as condições de alta suavidade e alta rugosidade. Através de (2.3) e (2.4), cada bloco será tratado no algoritmo TSS quando não estiverem em uma região de alta suavidade ou alta rugosidade. Caso contrário, o cálculo do deslocamento desses blocos não será adicionado para compor o GMV do quadro.

2.5.2 Algoritmo TSS

O próximo passo é fazer a estimativa de deslocamento de cada bloco, esse deslocamento é chamado de LMV (vetor de movimento local, do inglês *Local Motion Vector*). A estimativa do deslocamento de um bloco I_{p_k} no instante $k + 1$ através do algoritmo TSS é feita nas três etapas seguintes:

Figura 2.5: Exemplo de regiões ilegíveis: circulado em azul está uma região de alta suavidade; circulado em vermelho está uma região de alta rugosidade.

1. Compara-se o bloco I_{p_k} com blocos $I_{p'_{k+1}}$, sendo a origem do bloco na posição $p' = (n_i + \Delta l_1, n_j + \Delta c_1)$. Δl_1 e Δc_1 são testados com os valores $\{0, \lceil w/2 \rceil, \lceil -w/2 \rceil\}$ ⁸, totalizando nove possibilidades de comparações com I_{p_k} . A diferença de posição do bloco $I_{p'_{k+1}}$ que possui maior semelhança a I_{p_k} é atribuída ao vetor $(\Delta l_{p_k}, \Delta c_{p_k})$. Nesta etapa o resultado é $(\Delta l_{p_k}, \Delta c_{p_k}) = (\Delta l_1, \Delta c_1)$;
2. Compara-se o bloco I_{p_k} com blocos $I_{p''_{k+1}}$, sendo a origem do bloco na posição $p'' = (n_i + \Delta l_1 + \Delta l_2, n_j + \Delta c_1 + \Delta c_2)$. Δl_2 e Δc_2 são testados com os valores $\{0, \lceil w/4 \rceil, \lceil -w/4 \rceil\}$ menos o caso $(\Delta l_2, \Delta c_2) = (0, 0)$ já visto na etapa 1, totalizando oito possibilidades de comparações com I_{p_k} . A diferença de posição do bloco $I_{p''_{k+1}}$ que possui maior semelhança a I_{p_k} é somada ao vetor definido na etapa anterior, i.e. $(\Delta l_{p_k}, \Delta c_{p_k}) = (\Delta l_1 + \Delta l_2, \Delta c_1 + \Delta c_2)$;
3. Compara-se o bloco I_{p_k} com blocos $I_{p'''_{k+1}}$, sendo a origem do bloco na posição $p''' = (n_i + \Delta l_1 + \Delta l_2 + \Delta l_3, n_j + \Delta c_1 + \Delta c_2 + \Delta c_3)$. Δl_3 e Δc_3 são testados com os valores $\{0, \lceil w/8 \rceil, \lceil -w/8 \rceil\}$ menos o caso $(\Delta l_3, \Delta c_3) = (0, 0)$ já visto na etapa 2, totalizando oito possibilidades de comparações com I_{p_k} . A diferença de posição do bloco $I_{p'''_{k+1}}$ que possui maior semelhança a I_{p_k} é somada ao vetor $(\Delta l_{p_k}, \Delta c_{p_k})$. O resultado final do deslocamento do bloco I_{p_k} é $(\Delta l_{p_k}, \Delta c_{p_k}) = (\Delta l_1 + \Delta l_2 + \Delta l_3, \Delta c_1 + \Delta c_2 + \Delta c_3)$. Finaliza busca.

A Figura 2.6 ilustra graficamente um exemplo do resultado da busca usando o algoritmo TSS.

⁸O teto $\lceil \cdot \rceil$ de um número real x retorna a parte inteira somada de 1. Caso x seja um número inteiro, então $\lceil x \rceil = x$.

A comparação entre o bloco I_{p_k} e os blocos procurados na área de busca é medida pelo grau de associação entre eles através da expressão

$$\rho_{p_{k+1},k} = \frac{\sum_{i=n_i}^m \sum_{j=n_j}^n (I(i,j)_{q_{k+1}} - \bar{I}_{q_{k+1}}) (I(i,j)_{p_k} - \bar{I}_{p_k})}{\sqrt{\sum_{i=n_i}^m \sum_{j=n_j}^n (I(i,j)_{q_{k+1}} - \bar{I}_{q_{k+1}})^2} \sqrt{\sum_{i=n_i}^m \sum_{j=n_j}^n (I(i,j)_{p_k} - \bar{I}_{p_k})^2}} \quad (2.5)$$

sendo $\rho_{p_{k+1},k}$ o coeficiente de correlação, $\bar{I}_{p_k}$ e $\bar{I}_{q_{k+1}}$ é o valor médio dos *pixels* dos blocos I_{p_k} e $I_{q_{k+1}}$. q representa p' , p'' ou p''' caso esteja sendo executada a primeira, segunda ou terceira etapa do algoritmo TSS.

Figura 2.6: Exemplo de aplicação do algoritmo TSS. A figura mostra um gráfico dos possíveis deslocamentos de $p = (n_i, n_j)$ do bloco I_p dentro da área de busca. O gráfico mostra que foi encontrado um bloco de posição $p''' = (n_i + \Delta l_p, n_j + \Delta c_p)$ semelhante ao bloco de posição p do instante anterior, onde $(\Delta l_p, \Delta c_p) = (4 + 0 + 1, -4 - 2 - 1) = (5, -7)$.

2.5.3 Geração do GMV

Usualmente, os algoritmos baseados em *block matching* geram o GMV através do valor médio dos LMV's dos blocos. Para melhorar a estimativa do GMV, os coeficientes de correlação $\rho_{p_{k+1},k}$ são armazenados para substanciar o movimento de cada LMV encontrado. Assim, obtem-se

$$\Delta l_{g_k} = \frac{\sum \Delta l_{p_k} \rho_{p_{k+1},k}}{\sum \rho_{p_{k+1},k}}, \quad \Delta c_{g_k} = \frac{\sum \Delta c_{p_k} \rho_{p_{k+1},k}}{\sum \rho_{p_{k+1},k}}, \quad (2.6)$$

onde Δl_{g_k} e Δc_{g_k} é o deslocamento de linhas e colunas do quadro I_k visto no instante $k + 1$. A posição atualizada do quadro devido ao deslocamento é

$$x_{TSS_k} = \Delta c_{g_k} + \Delta c_{g_{k+1}}, \quad y_{TSS_k} = \Delta l_{g_k} + \Delta l_{g_{k+1}}. \quad (2.7)$$

2.5.4 Outros algoritmos

Outros algoritmos bastante utilizados para estimativa de movimento da câmera são o NTSS (nova busca por três etapas, do inglês *New Three-step Search*), FSS (busca por quatro etapas, do inglês *Four-step Search*), e DS (busca de diamante, do inglês *Diamond Search*). Em [21], foi elaborada uma pesquisa para analisar as vantagens e desvantagens desses algoritmos e o TSS. Os resultados podem ser visualizados na Tabela 2.1.

Neste trabalho foi feita a comparação técnica entre os algoritmos calculando-se a variância do erro de estimativa utilizando a aproximação

$$\varepsilon \|x_k - \hat{x}_k\| \approx \frac{1}{T} \sum_{j=1}^T \|x_k^{(j)} - \hat{x}_k^{(j)}\|, \quad (2.8)$$

sendo $\|\cdot\|$ a norma euclidiana. Cada ponto no instante k em cada curva da variância é uma média calculada sobre $T = 50$ experimentos. A Tabela 2.2 mostra a média do cálculo do erro de (2.8) realizado sobre os resultados dos algoritmos TSS, NTSS, FSS e DS, estes aplicados para detecção de movimento sob vibração artificial. Veja o conceito na Seção 5.1.

Tabela 2.1: Principais características dos algoritmos. Tabela extraída da referência [23]

Algoritmo	Vantagens	Desvantagens
TSS	Simples e robusto, e próximo do desempenho ótimo; Melhores vetores de movimento para um comportamento padrão;	Pouca eficiência para estimar movimentos curtos;
NTSS	Melhor estimativa de movimento comparado aos algoritmos TSS e DS devido à boa estimativa de movimentos curtos;	Tempo de processamento maior do que os algoritmos TSS e DS;
FSS	Mais robusto, comparado aos algoritmos TSS e NTSS;	Maior tempo de processamento, comparado aos outros algoritmos;
DS	Menor tempo de processamento.	Não é eficiente em implementação em <i>hardware</i> .

Tabela 2.2: Cálculo do erro dos algoritmos. Os ambientes (a), (b), (c) e (d) contém vibração onde a variância σ_X^2 é 10, 20, 40 e 80, respectivamente.

Algoritmo	(a)	(b)	(c)	(d)	Média
TSS	1.03	1.01	0.88	0.92	0.96
NTSS	0.85	0.91	0.84	0.84	0.86
FSS	0.72	0.76	0.7	0.7	0.72
DS	0.84	0.88	0.84	0.88	0.86

Capítulo 3

Conjunto algoritmo TSS e acelerômetro

Esse capítulo apresenta os conceitos básicos de vibração necessários para este trabalho; O acelerômetro como unidade de medida da vibração, a adaptação do algoritmo TSS e as técnicas para vincular este aos dados do acelerômetro.

3.1 Vibração

Um objeto é considerado imerso em vibração quando ele está sujeito a um movimento oscilatório em torno de um ponto de referência [47]. O movimento oscilatório geralmente contém sinais distribuídos em uma larga faixa de frequências. Por exemplo, o movimento de turbinas de uma aeronave devido à reação da combustão realizada pela mistura do fluxo de ar comprimido e combustível, veja referência na Figura 3.1. Como se pode ver nessa figura, a visualização do sinal de interesse torna-se difícil quando é feita a análise no tempo.

Os instrumentos usados para medição de vibração em laboratório são LASER vibrômetro ou acelerômetros. Um LASER vibrômetro é capaz de fazer medição da vibração distante do objeto alvo observando-se o deslocamento da superfície do objeto sob a radiação do LASER. Este é um instrumento relativamente caro e não funciona apropriadamente quando não está sobre um ambiente estável. Para sistemas de robôs móveis, prefere-se a aplicação de acelerômetros devido à necessidade de ter o dispositivo embarcado e com baixo custo. Veja seu princípio de operação na Seção 3.2.

Figura 3.1: Os gráficos mostram os níveis de vibrações nas proximidades de um aeroporto devido à aproximação de uma aeronave para pouso. Veja referência [48].

3.2 Princípio de operação do acelerômetro

A aceleração é intrinsecamente uma dinâmica variável, decorrente da aplicação de uma força resultante sobre um corpo. O elemento básico de qualquer acelerômetro é a massa m suspensa por uma estrutura elástica composta por mola e amortecedor. Essa estrutura transmite uma força à massa m fazendo com que ela tenha o mesmo movimento do objeto em observação. Além desses componentes, há um elemento sensível que gera uma diferença de potencial em seus terminais proporcional a força sobre a massa m .

A estrutura básica de um acelerômetro pode ser modelada como um sistema massa-mola-amortecedor, como na Figura 3.2. O deslocamento da massa m em resposta à aceleração sofrida é tratado como um sistema de segunda ordem, como mostrado na equação (3.1),

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2x = \ddot{u} \quad (3.1)$$

sendo x a posição relativa, $\zeta = \frac{b}{2\sqrt{k_s m}}$ é o coeficiente de amortecimento, e $\omega_n = \sqrt{\frac{k_s}{m}}$ é a frequência natural. u , paralelo à x , é a força aplicada sobre o corpo o qual tem comportamento não linear.

Para obter valores de posição, primeiramente o sinal de aceleração captado pelo acelerômetro é pré-filtrado, veja modelagem do acelerômetro no **Apêndice A**. Após aplicar o pré-filtro, o sinal é integrado para obter o deslocamento no tempo. A integração é realizada como segue:

$$v_{x,t+1} = v_{x,t} + a_{x,t}T, \quad (3.2)$$

$$v_{y,t+1} = v_{y,t} + a_{y,t}T, \quad (3.3)$$

$$x_{t+1} = x_t + v_{x,t}T + a_{x,t}\frac{T^2}{2}, \quad (3.4)$$

$$y_{t+1} = y_t + v_{y,t}T + a_{y,t}\frac{T^2}{2}, \quad (3.5)$$

sendo a_x e v_x a aceleração e velocidade da mudança de posição de x após filtragem. Analogamente, a_y e v_y a aceleração e velocidade da mudança de posição de y . T é o período de amostragem e t é o momento em que cada amostra é coletada.

Figura 3.2: Sistema massa-mola-amortecedor básico.

Neste trabalho, foi proposto fazer a análise em frequência devido à dificuldade de resolver a equação (3.1). A análise em frequência tem o objetivo de verificar a banda de frequência significativa que envolve a movimentação do acelerômetro em meio à vibração.

3.2.1 Análise em frequência

A princípio, foram realizados vários experimentos com o acelerômetro sobre o veículo em movimento. A análise em frequência foi realizada aplicando a DFT (transformada discreta de Fourier, do inglês *Discrete Fourier Transform*) sobre o sinal obtido do acelerômetro à uma taxa de amostragem de 10 kHz . Os gráficos da Figura 3.3 mostram o valor médio dos sinais dos experimentos após aplicar a DFT.

Os ambientes onde foram realizados os experimentos não causaram vibração significativa acima de 100 Hz , veja os gráficos da Figura 3.3. Pelos gráficos, a banda abaixo de 100 Hz contém sinais de maior intensidade referente aos distúrbios externos causados pela movimentação do veículo. Portanto, os dados do acelerômetro tornam-se desejáveis para esse estudo para cooperar com o algoritmo de estimativa de movimento dos quadros. O ganho dessa cooperação busca minimizar o erro da detecção de vibração e diminuir o custo computacional do algoritmo TSS.

O acelerômetro deverá trabalhar à taxa de amostragem equivalente a um harmônico de 30 Hz para sincronizar com o período de amostragem da aquisição de quadros da câmera. O harmônico

de maior frequência dentro da banda de 100 Hz corresponde ao período de amostragem $\Delta t = \frac{1}{90Hz}$. Pelo teorema da amostragem de Nyquist-Shannon [52], o período de amostragem deve ser duas vezes menor do que a frequência da banda limitada, sendo assim $\Delta t = \frac{1}{2*90Hz}$.

A Figura 3.4 mostra a discrepância que surge devido a diferença entre a taxa de amostragem dos dados dos dispositivos. Essa discrepância que o acelerômetro proporciona é utilizada para ajudar o algoritmo TSS. Veja o método na próxima seção.

Figura 3.3: Resposta em frequência. Experimentos: (a) rua pavimentada à 60 km/h, (b) rua pavimentada à 30 km/h, (c) rodovia à 90 km/h, (d) rua asfaltada à 30 km/h.

Figura 3.4: Exemplo diferença dos dados coletados pelo acelerômetro (círculos) e algoritmo TSS (losangos).

3.3 Disposição do conjunto câmera e acelerômetro

O acelerômetro é acoplado atrás da câmera colocando o seu plano XY em paralelo com o plano $j \times i$ da câmera, como mostra a Figura 3.5.

Figura 3.5: Arranjo câmera-acelerômetro no sistema de coordenadas da câmera.

Para usar os dois sensores em conjunto é preciso transformar os dados obtidos do acelerômetro, que está no sistema de coordenadas do mundo, para o sistema de coordenadas da câmera. A mudança de coordenadas possibilita o cálculo do deslocamento dos *pixels* na imagem digital. A equação (3.6) mostra a relação entre o sistema de referência da câmera e do mundo.

$${}^wP = {}^wR_c {}^cP + {}^wT_c, \quad (3.6)$$

sendo wP a posição do objeto no sistema de coordenadas do mundo. wR_c é a diferença da orientação da câmera em relação à orientação do sistema de coordenadas do mundo. cP é a localização do objeto observado no sistema de coordenadas da câmera e wT_c é a distância entre a câmera e a referência da origem do sistema de coordenadas do mundo. A equação (3.6) pode ser reescrita para obter cP em função de wP , como segue abaixo:

$$\begin{aligned} {}^wP - {}^wT_c &= {}^wR_c {}^cP + {}^wT_c - {}^wT_c \Rightarrow \\ \Rightarrow ({}^wR_c)^T ({}^wP - {}^wT_c) &= ({}^wR_c)^T {}^wR_c {}^cP \Rightarrow \\ \Rightarrow {}^cP &= {}^cR_w ({}^wP - {}^wT_c), \end{aligned} \quad (3.7)$$

sendo cR_w a diferença da orientação da mundo em relação à orientação do sistema de coordenadas da câmera. A equação (3.7) também pode ser reescrita como

$$\begin{bmatrix} {}^cX \\ {}^cY \\ {}^cZ \end{bmatrix} = \begin{bmatrix} R_1^T \\ R_2^T \\ R_3^T \end{bmatrix} ({}^wP - {}^wT_c). \quad (3.8)$$

Assim produz:

$${}^cX = R_1^T ({}^wP - {}^wT_c), \quad (3.9)$$

$${}^cY = R_2^T ({}^wP - {}^wT_c) \quad (3.10)$$

e

$${}^cZ = R_3^T ({}^wP - {}^wT_c). \quad (3.11)$$

Em seguida deve-se encontrar os valores de deslocamento x e y de (3.4) e (3.5) na imagem digital. As equações (3.12) e (3.13) determinam a relação entre a posição do objeto observado pelo mundo e pelo campo visual da câmera, estes representados por x_c e y_c . (3.14) e (3.15) mostram a conversão de x_c e y_c para a posição em *pixels* no quadro:

$$x_c = f \frac{X}{Z} \quad (3.12)$$

$$y_c = f \frac{Y}{Z}, \quad (3.13)$$

$$x_c = -(x_{im} - o_x)s \quad (3.14)$$

$$y_c = -(y_{im} - o_y)s \quad (3.15)$$

sendo x_{im} e y_{im} a posição de x e y na imagem digital. o_x e o_y é o centro da imagem digital para as coordenadas de x e y , respectivamente. s é o tamanho do *pixel*, em $mm/pixel$, e f é a distância focal da câmera. Aplicando (3.9), (3.10), (3.11), (3.12) e (3.13) em (3.14) e (3.15) obtém-se

$$\begin{aligned} -(x_{im} - o_x)s &= f \frac{R_1^T ({}^wP - {}^wT_c)}{R_3^T ({}^wP - {}^wT_c)} \Rightarrow \\ \Rightarrow x_{im} &= o_x - \frac{f R_1^T ({}^wP - {}^wT_c)}{s R_3^T ({}^wP - {}^wT_c)}, \end{aligned} \quad (3.16)$$

e

$$\begin{aligned} -(y_{im} - o_y)s &= f \frac{R_2^T ({}^wP - {}^wT_c)}{R_3^T ({}^wP - {}^wT_c)} \Rightarrow \\ \Rightarrow y_{im} &= o_y - \frac{f R_2^T ({}^wP - {}^wT_c)}{s R_3^T ({}^wP - {}^wT_c)}. \end{aligned} \quad (3.17)$$

Para uma trajetória qualquer, considere inicialmente que a câmera está na posição ${}^wT_c^T = [x_t \ y_t \ z_t]$ e com mesma orientação das coordenadas do mundo. Um objeto qualquer na posição wP é visto pela câmera na posição:

$${}^cP = {}^cR_w ({}^wP - {}^wT_c) = \begin{bmatrix} {}^wX - x_t \\ {}^wY - y_t \\ {}^wZ - z_t \end{bmatrix},$$

e de (3.16) e (3.17),

$$x_{im} = o_x - \frac{f}{s} \frac{({}^wX - x_t)}{({}^wZ - z_t)}, \quad y_{im} = o_y - \frac{f}{s} \frac{({}^wY - y_t)}{({}^wZ - z_t)}.$$

Após Δt pequeno o conjunto veículo, câmera e acelerômetro sofrem um deslocamento mudando-os para a posição $\begin{bmatrix} x_{t+1} & y_{t+1} & z_{t+1} \end{bmatrix}^T$ e sofre rotação no eixo Y de $\Delta\theta$. Logo, de (3.7):

$$\begin{aligned} {}^cP &= \begin{bmatrix} \cos\Delta\theta & 0 & -\sin\Delta\theta \\ 0 & 1 & 0 \\ \sin\Delta\theta & 0 & \cos\Delta\theta \end{bmatrix} \begin{bmatrix} {}^wX - x_{t+1} \\ {}^wY - y_{t+1} \\ {}^wZ - z_{t+1} \end{bmatrix} \Rightarrow \\ \Rightarrow {}^cP &= \begin{bmatrix} ({}^wX - x_{t+1})\cos\Delta\theta + ({}^wZ - z_{t+1})\sin\Delta\theta \\ {}^wY - y_{t+1} \\ ({}^wX - x_{t+1})\sin\Delta\theta + ({}^wZ - z_{t+1})\cos\Delta\theta \end{bmatrix}. \end{aligned}$$

Considerando a aproximação $\sin(\Delta\theta) = \Delta\theta$ e $\cos(\Delta\theta) = 1$, tem-se

$${}^cP \cong \begin{bmatrix} ({}^wX - x_{t+1}) + \Delta\theta({}^wZ - z_{t+1}) \\ {}^wY - y_{t+1} \\ ({}^wZ - z_{t+1}) + \Delta\theta({}^wX - x_{t+1}) \end{bmatrix}.$$

Aplicando o resultado acima em (3.13) e (3.14), tem-se

$$x'_{im} = o_x - \frac{f}{s} \frac{{}^wX - x_{t+1} - \Delta\theta({}^wZ - z_{t+1})}{{}^wZ - z_{t+1} - \Delta\theta({}^wX - x_{t+1})}, \quad y'_{im} = o_y - \frac{f}{s} \frac{{}^wY - y_{t+1}}{{}^wZ - z_{t+1} - \Delta\theta({}^wX - x_{t+1})},$$

$\Delta\theta$ é obtido com a aplicação de giroscópios, os quais são capazes de detectar velocidade angular do dispositivo sob teste. No escopo desse trabalho, utiliza-se somente acelerômetros, i.e. $\Delta\theta \rightarrow 0$ e $\Delta\phi \rightarrow 0$. Assim, para o exemplo acima o deslocamento resulta em:

$$\begin{aligned} \Delta x_{im} &= x'_{im} - x_{im} = \frac{f}{s} \left[\frac{{}^wX - x_{t+1}}{{}^wZ - z_{t+1}} - \frac{({}^wX - x_t)}{({}^wZ - z_t)} \right], \\ \Delta y_{im} &= y'_{im} - y_{im} = \frac{f}{s} \left[\frac{{}^wY - y_{t+1}}{{}^wZ - z_{t+1}} - \frac{({}^wY - y_t)}{({}^wZ - z_t)} \right], \end{aligned}$$

considerando que ${}^wZ \gg \Delta z_t$, o deslocamento será

$$\Delta x_{im} = K (x_{t+1} - x_t), \quad \Delta y_{im} = K (y_{t+1} - y_t), \quad (3.18)$$

sendo K uma constante da relação foco, tamanho do *pixel* e distância entre a origem e a posição da câmera no eixo z do sistemas de coordenadas do mundo.

Haverá perda de sensibilidade de mudança de orientação e acumulação de erro de deslocamento quando o veículo executa trajetória com rotação devido a aproximação $\Delta\theta \rightarrow 0$ ou $\Delta\phi \rightarrow 0$. No entanto, o acelerômetro detecta vibrações com larga faixa de amplitudes, ao contrário do método TSS que busca comparações sobre deslocamentos fixos $\lceil w/2 \rceil$, $\lceil w/4 \rceil$ e $\lceil w/8 \rceil$, sendo w também fixo. A parte crítica do algoritmo TSS está na primeira etapa em que resulta um coeficiente de correlação ρ de valor baixo. Este trabalho propõe a substituição da primeira etapa do algoritmo TSS pelos dados fornecidos pelo acelerômetro em (3.18). As etapas 2 e 3 do algoritmo TSS devem encontrar o deslocamento da câmera com precisão eliminando o erro causado pelas incertezas das medidas do acelerômetro e pelas aproximações $\Delta\theta = 0$ e $\Delta\phi = 0$.

A integração do conjunto de dados coletados pelo acelerômetro e algoritmo TSS está descrito no fluxograma da Figura [1.1].

O acelerômetro fornece dados de aceleração em paralelo à aquisição de quadros pela câmera. No computador é calculada a posição x_{im}, y_{im} do acelerômetro no plano XY a 180 Hz através da acumulação de Δx_{im} e Δy_{im} em (3.18). A cada seis amostras de x_{im} e y_{im} , só uma é usada no TSS para sincronizar com a aquisição de quadros à 30 Hz . A Figura [3.6] mostra um exemplo da aplicação do algoritmo TSS com a primeira etapa fornecida por dados do acelerômetro em substituição ao método utilizado na Seção 2.5.2 e Figura [2.6].

Figura 3.6: Exemplo de aplicação do algoritmo TSS com a primeira etapa fornecida pelos cálculos dos dados do acelerômetro. A figura indica o deslocamento $(\Delta l_p, \Delta c_p) = (\Delta y_{im} + 3 - 1, \Delta x_{im} - 3 - 1) = (\Delta y_{im} + 2, \Delta x_{im} - 4)$.

Assim, de (2.6) e (3.18) a posição resultante do deslocamento é

$$\begin{aligned}x_{TSS_{k+1}} &= (\Delta x_{im_{k+1}} + \Delta c_{g_{k+1}}) + x_{TSS_k}, \\y_{TSS_{k+1}} &= (\Delta y_{im_{k+1}} + \Delta l_{g_{k+1}}) + y_{TSS_k}.\end{aligned}\tag{3.19}$$

Capítulo 4

Filtragem

Este capítulo apresenta a estrutura dos filtros utilizados para amortecer a vibração do vídeo digital. Entre eles estão: filtro IIR adaptativo [31], filtro média móvel, filtro de Kalman, filtro robusto BDU [40] e Filtro de Kalman robusto recursivo [40]. Um estudo comparativo é feito para avaliar o desempenho dos filtros robustos para sistemas de estabilização de vídeo. Os resultados são apresentados na Seção 5.3.

4.1 Filtro média móvel

O filtro média móvel opera calculando a média de uma série de pontos do sinal de entrada para produzir uma sinal amortecido. Este filtro é definido pela equação:

$$\mathbf{y}_k = \frac{1}{D} \sum_{d=0}^{D-1} \mathbf{z}_{k-d}, \quad (4.1)$$

sendo $\mathbf{y}$ o sinal filtrado, $\mathbf{z}$ é o sinal de entrada e D indica a ordem do filtro. A Figura 4.1 mostra um exemplo de aplicação do filtro.

Como proposto em [35], neste trabalho foi testado um filtro média móvel de ordem cinco ($D = 5$).

4.2 Filtro IIR adaptativo

Nesta seção é apresentado um filtro IIR adaptativo proposto em [31]. O filtro tem a seguinte estrutura:

Figura 4.1: Exemplo de um filtro média móvel. Em (a) há um pulso retangular imerso em ruído. Em (b) e (c) é mostrado o sinal quando um filtro média móvel de ordem 11 e 51 foram respectivamente aplicados. A figura foi extraída da referência [34].

Passo 1:

$$\mathbf{s}_{k+1} = \mathbf{z}_{k+1} - \mathbf{y}_k,$$

$$\theta = \left| \frac{\langle \mathbf{s}_{k+1}, \mathbf{y}_k \rangle}{\sqrt{\langle \mathbf{s}_{k+1}, \mathbf{s}_{k+1} \rangle} \sqrt{\langle \mathbf{y}_k, \mathbf{y}_k \rangle}} \right|,$$

Se $\mathbf{s}_{k+1} > T_1$ e $\mathbf{s}_k > T_2$ e $\mathbf{s}_{k-1} > T_3$, ou $\mathbf{s}_{k+1} < -T_1$ e $\mathbf{s}_k < -T_2$ e $\mathbf{s}_{k-1} < -T_3$,

Então $b = 0.8$;

Senão,

Então $b = 0.1$;

Passo 2:

$$\beta = \min(\theta, b),$$

$$\alpha = 1 - \beta,$$

$$\mathbf{y}_{k+1} = \alpha \mathbf{y}_k + \beta \mathbf{z}_{k+1},$$

sendo α o coeficiente de amortecimento, $\langle X, Y \rangle$ o produto escalar entre X e Y , e T_1 , T_2 e T_3 são *thresholds* predefinidos pelo projetista.

4.3 Filtro de Kalman

Considere o seguinte sistema dinâmico:

$$\mathbf{x}_{k+1} = F_k \mathbf{x}_k + G_k \mathbf{w}_k \quad (4.2)$$

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{v}_k \quad (4.3)$$

sendo que $\mathbf{x}_k \in \mathbf{R}^2$ é a variável semi-estado que descreve o comportamento interno do sistema, $\mathbf{z}_k$ é o sinal observado, i.e. a posição do quadro no instante k estimado em (2.7). As matrizes F_k , G_k e H_k são assumidas conhecidas. Os valores adotados para essas matrizes são:

$$F_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad G_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad H_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

os quais foram referenciadas de [49]. Em [49] é proposto a aplicação de filtragem robusta para acompanhar e filtrar posições imprecisas de uma mão em movimento dentro do campo visual de uma câmera. $\mathbf{w}_k$ e $\mathbf{v}_k$ denotam processos brancos de média zero descorrelacionados e ruídos de medidas que satisfazem em:

$$E \left(\begin{bmatrix} x_0 \\ w_i \\ v_i \end{bmatrix}, \begin{bmatrix} x_o \\ w_j \\ v_j \end{bmatrix}^T \right) = \begin{bmatrix} \Pi & 0 & 0 \\ 0 & Q\delta_{ij} & 0 \\ 0 & 0 & R\delta_{ij} \end{bmatrix}. \quad (4.4)$$

O filtro de Kalman clássico, [36], é implementado pelo seguinte algoritmo

$$\begin{aligned} \hat{\mathbf{x}}_{k+1|k} &= F_k \hat{\mathbf{x}}_{k|k}, \\ P_{k+1|k} &= F_k P_{k|k} F_k^T + G_k Q_k G_k^T, \\ K_{k+1} &= P_{k+1|k} H_{k+1}^T [H_{k+1} P_{k+1|k} H_{k+1}^T + R_{k+1}]^{-1}, \\ P_{k+1|k+1} &= [I - K_{k+1} H_{k+1}] P_{k+1|k}, \\ \hat{\mathbf{x}}_{k+1|k+1} &= \hat{\mathbf{x}}_{k+1|k} + K_{k+1} [\mathbf{z}_{k+1} - H_k \hat{\mathbf{x}}_{k+1|k}], \end{aligned}$$

sendo $\hat{\mathbf{x}}_{k+1|k}$ a estimativa preditora e $\hat{\mathbf{x}}_{k+1|k+1}$ a estimativa filtrada. As matrizes de covariância são

$$Q = \begin{bmatrix} 8.6 & 0 \\ 0 & 8.6 \end{bmatrix}, \quad R = \begin{bmatrix} 64 & 0 \\ 0 & 64 \end{bmatrix}.$$

R foi definido de acordo com a variância média entre os experimentos e Q foi definido através do algoritmo genético, veja Seção 4.6.

4.4 Filtro BDU

Nesta seção é apresentado o filtro BDU proposto em [40]. Para isto considere o seguinte sistema dinâmico com incertezas:

$$\mathbf{x}_{k+1} = (F_k + \delta F_k) \mathbf{x}_k + (G_k + \delta G_k) \mathbf{w}_k \quad (4.5)$$

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{v}_k \quad (4.6)$$

sendo suas perturbações modeladas como:

$$[\delta F_k \ \delta G_k] = M \Delta_k [N_F \ N_G], \quad (4.7)$$

com $\|\Delta_k\| < 1$. O modelo em (4.7) permite que o projetista restrinja as fontes de distorção selecionando as entradas $\{N_F \ N_G\}$ apropriadamente [38]. Este filtro robusto procura minimizar o erro de estimativa para o pior caso possível criado pela faixa de incertezas δF_k e δG_k . As estimativas $\{\hat{\mathbf{x}}_{k+1|k}, \hat{\mathbf{x}}_{k+1|k+1}\}$ no algoritmo que segue, podem ser obtidas por interações recursivas entre os passos 2 e 3. Para o caso quando $\hat{\lambda} = 0$, os passos 2 e 3 são reduzidos para o tempo padrão e para as equações de atualização de medida do filtro de Kalman. O algoritmo do filtro robusto é descrito a seguir:

Passo 1:

Se $H_{k+1}M_k = 0$, então configure $\hat{\lambda}_k = 0$ (filtro não robusto). Caso contrário, selecione α_f (tipicamente $0 \leq \alpha_f$) e configure $\hat{\lambda}_k = (1 + \alpha_f) \cdot \|M_k^T H_{k+1}^T R_{k+1}^{-1} H_{k+1} M_k\|$.

Passo 2:

Substitua $\{Q_k, R_k, P_{k|k}, G_k, F_k\}$ por:

$$\begin{aligned} \hat{Q}_k^{-1} &= Q_k^{-1} + \hat{\lambda}_k N_{G_k}^T \left[I + \hat{\lambda}_k N_{F_k} P_{k|k} N_{F_k}^T \right]^{-1} N_{G_k}, \\ \hat{R}_{k+1} &= R_k - \hat{\lambda}_k^{-1} H_{k+1} M_k M_k^T H_{k+1}^T, \\ \hat{P}_{k|k} &= \left(P_{k|k}^{-1} + \hat{\lambda}_k N_{F_k}^T N_{F_k} \right)^{-1}, \\ \hat{G}_k &= G_k - \hat{\lambda}_k F_k \hat{P}_{k|k} N_{F_k}^T N_{G_k}, \\ \hat{F}_k &= (F_k - \hat{\lambda}_k \hat{G}_k \hat{Q}_{ik} N_{G_k}^T N_{F_k}) (I - \hat{\lambda}_k \hat{P}_{k|k} N_{F_k}^T N_{F_k}), \end{aligned}$$

Passo 3:

Atualize $\{\hat{\mathbf{x}}_{k|k}, P_{k|k}\}$ como segue:

$$\begin{aligned}
\hat{\mathbf{x}}_{k+1|k} &= \hat{F}_k \hat{\mathbf{x}}_{k|k}, \\
P_{k+1|k} &= F_k \hat{P}_{k|k} F_k^T + \hat{G}_k \hat{Q}_k \hat{G}_k^T, \\
K_{k+1} &= P_{k+1|k} H_{k+1}^T \left[H_{k+1} P_{k+1|k} H_{k+1}^T + \hat{R}_{k+1} \right]^{-1}, \\
P_{k+1|k+1} &= [I - K_{k+1} H_{k+1}] P_{k+1|k}, \\
\hat{\mathbf{x}}_{k+1|k+1} &= \hat{\mathbf{x}}_{k+1|k} + K_{k+1} [\mathbf{z}_{k+1} - H_k \hat{\mathbf{x}}_{k+1|k}].
\end{aligned}$$

Q e R foram iniciados com os mesmos valores adotados no filtro de Kalman e, utilizando o algoritmo genético da Seção 4.6, foram definidos:

$$M = 1.01, \quad \alpha_f = 0.5, \quad N_F = \begin{bmatrix} 0.016 & 0 \\ 0 & 0.016 \end{bmatrix}, \quad N_G = \begin{bmatrix} 0.016 & 0 \\ 0 & 0.016 \end{bmatrix}.$$

4.5 Filtro de Kalman robusto recursivo

O Filtro de Kalman Robusto Recursivo, FKRR, proposto em [43], é uma extensão do filtro robusto apresentado em [41]. Considere o sistema com incertezas paramétricas modeladas por (4.8):

$$\begin{bmatrix} \delta F_k & \delta G_k \\ \delta H_k & 0 \end{bmatrix} = \begin{bmatrix} M_{1_k} & 0 \\ 0 & M_{2_k} \end{bmatrix} \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} \begin{bmatrix} N_{F_k} & N_{G_k} \\ N_{H_k} & 0 \end{bmatrix} \quad (4.8)$$

Δ_1 e Δ_2 são termos aleatórios cujas normas são definidas por $|\Delta| < 1$, $j = 1, 2$. As incertezas δF_k , δG_k e δH_k permitem restringir as fontes de distorções através da seleção apropriada das entradas $\{M_{1_k}, M_{2_k}\}$ e $\{N_{F_k}, N_{G_k}, N_{H_k}\}$. O FKRR proposto em [41] é definido pelas equações (4.9) e (4.10):

$$\begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\mathbf{x}}_{k+1|k} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_{k|k-1} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ I & 0 \\ 0 & 0 \\ 0 & I \end{bmatrix}^T \begin{bmatrix} P_{k|k-1} & 0 & 0 & 0 & I & 0 & 0 \\ 0 & \mathcal{R}_k & 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & \mathcal{F}_k & \mathcal{G}_k & \mathcal{E}_k \\ 0 & 0 & 0 & 0 & N_{\mathcal{F},k} & N_{\mathcal{G},k} & N_{\mathcal{E},k} \\ I & 0 & \mathcal{F}_k^T & N_{\mathcal{F},k}^T & 0 & 0 & 0 \\ 0 & 0 & \mathcal{G}_k^T & N_{\mathcal{G},k}^T & 0 & 0 & 0 \\ 0 & 0 & \mathcal{E}_k^T & N_{\mathcal{E},k}^T & 0 & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ \mathbf{b}_k \\ N_{\mathbf{b}_k} \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad (4.9)$$

$$P_{k|k-1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ I \end{bmatrix}^T \begin{bmatrix} P_{k|k-1} & 0 & 0 & 0 & I & 0 & 0 \\ 0 & \mathcal{R}_k & 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & \mathcal{F}_k & \mathcal{G}_k & \mathcal{E}_k \\ 0 & 0 & 0 & 0 & N_{\mathcal{F},k} & N_{\mathcal{G},k} & N_{\mathcal{E},k} \\ I & 0 & \mathcal{F}_k^T & N_{\mathcal{F},k}^T & 0 & 0 & 0 \\ 0 & I & \mathcal{G}_k^T & N_{\mathcal{G},k}^T & 0 & 0 & 0 \\ 0 & 0 & \mathcal{E}_k^T & N_{\mathcal{E},k}^T & 0 & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -I \end{bmatrix} \quad (4.10)$$

sendo

$$\begin{aligned} \mathcal{F}_k &= \begin{bmatrix} F_k \\ H_k \end{bmatrix}, \quad \mathcal{G}_k = \begin{bmatrix} G_k & 0 \\ 0 & I \end{bmatrix}, \quad \mathcal{E}_k = \begin{bmatrix} -I \\ 0 \end{bmatrix}, \quad \mathcal{R}_k = \begin{bmatrix} Q_k & 0 \\ 0 & R_{k+1} \end{bmatrix}, \\ N_{\mathcal{F}_k} &= \begin{bmatrix} N_{F_k} \\ N_{H_k} \end{bmatrix}, \quad N_{\mathcal{G}_k} = \begin{bmatrix} N_{G_k} & 0 \\ 0 & 0 \end{bmatrix}, \quad N_{\mathcal{E}_k} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\ \mathbf{b}_k &= \begin{bmatrix} -F_k \hat{\mathbf{x}}_{k|k-1} \\ \mathbf{z}_k - H_k \hat{\mathbf{x}}_{k|k-1} \end{bmatrix}, \quad N_{\mathbf{b}_k} = \begin{bmatrix} -N_{F_k} \hat{\mathbf{x}}_{k|k-1} \\ -N_{H_k} \hat{\mathbf{x}}_{k|k-1} \end{bmatrix}. \end{aligned} \quad (4.11)$$

A condição de existência deste filtro será garantida se

$$\begin{bmatrix} \mathcal{F}_k & \mathcal{G}_k & \mathcal{E}_k \\ N_{\mathcal{F},k} & N_{\mathcal{G},k} & N_{\mathcal{E},k} \end{bmatrix}$$

tiver posto linha pleno. Observe que nesta abordagem o FKRR não depende de parâmetros auxiliares para ser ajustado. Depende somente de parâmetros e ponderações de matrizes que são conhecidos *a-priori*.

Q , R , N_{F_k} e N_{G_k} foram iniciados com os mesmos valores adotados no filtro de Kalman e BDU e, utilizando o algoritmo genético foi definido:

$$N_H = \begin{bmatrix} 0.005 & 0 \\ 0 & 0.005 \end{bmatrix}.$$

4.6 Algoritmo genético

O algoritmo genético foi proposto para encontrar valores de parâmetros dos filtros quando não é possível obtê-los através de modelos de espaço de estado. Esse algoritmo executa exaustivamente a busca de valores para parâmetros $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$ e seleciona valores adequados para estes de forma que a área A seja reduzida ao mínimo. A área A é o resultado da soma da diferença absoluta entre cada $f_k(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$ e f'_k dentro de uma amostra de tempo T , veja a equação (4.12).

$$A = \frac{1}{T} \sum_{k=1}^T |f_k(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n) - f'_k| \quad (4.12)$$

f' é uma curva de referência representando a movimentação dos quadros livre de vibração. $f(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$ é a curva resultado após aplicação do filtro que contém os parâmetros $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$ que se deseja encontrar. O algoritmo abaixo descreve detalhadamente como o algoritmo genético deve ser executado.

Algorithm 1: Algoritmo genético

Result: catch suit parameters for Kalman and robust filters

initialization;

```

for  $\varepsilon_1 \leftarrow initial\_value_1$  to  $final\_value_1$  do
    for  $\varepsilon_2 \leftarrow initial\_value_2$  to  $final\_value_2$  do
         $\vdots$ 
        for  $\varepsilon_n \leftarrow initial\_value_n$  to  $final\_value_n$  do
             $A_{current} \leftarrow CALC\_AREA(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n);$ 
            if  $A_{current} < A_{previous}$  then
                 $A_{minimum} \leftarrow A_{current};$ 
                 $A_{previous} \leftarrow A_{current};$ 
return  $A_{minimum}$  parameters;

```

$CALC_AREA(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$: função descrita em (4.12).

A Figura 4.2 mostra um exemplo da representação da curva com vibração, curva de referência f' , curva resultado da filtragem $f(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$ e a área A .

Figura 4.2: Gráfico para representação das variáveis do algoritmo genético.

A curva de referência f' é adquirida através da interpolação polinomial na forma de Newton, como descrito abaixo:

$$P_n(t) = f[t_0] + \sum_{k=1}^n f[t_0, t_1, \dots, t_k] (t - t_0) (t - t_1) \dots (t - t_{k-1}), \quad (4.13)$$

sendo n a ordem do polinômio, $f[t_0] = f(t_0)$ e

$$f[t_0, t_1, \dots, t_n] = \frac{f[t_1, t_2, \dots, t_n] - f[t_0, t_1, \dots, t_{n-1}]}{t_n - t_0}. \quad (4.14)$$

Neste trabalho, o objetivo do algoritmo genético é selecionar valores para os parâmetros Q_k para o filtro de Kalman, M , α_f , N_G e N_F para o filtro BDU, N_H para o FKRR.

A busca dos parâmetros M , α_f , N_G e N_F para o filtro BDU é feita considerando R_k e Q_k iguais aos adotados para o filtro de Kalman. A busca do parâmetro N_H para o FKRR é feita considerando R_k e Q_k iguais aos parâmetros utilizados no filtro de Kalman e N_G e N_F iguais aos parâmetros utilizados no filtro BDU.

Para melhorar a estimativa dos valores dos parâmetros encontrados pelo algoritmo genético, o mesmo procedimento foi executado para 50 experimentos. A média dos valores dos parâmetros encontrados em todos os experimentos foi adotada como resultado da busca.

Capítulo 5

Experimentos e resultados

Este capítulo mostra a configuração que foi utilizada para executar os testes sobre o sistema proposto. Também é apresentado os resultados individualmente do algoritmo de visão computacional *three-step search*, o conjunto do acelerômetro com o algoritmo TSS adaptado, e o sistema completo para avaliar o comportamento dos filtros.

5.1 Configuração, testes e resultados do algoritmo TSS

Para verificar a eficácia e eficiência do algoritmo TSS foi proposto editar um vídeo adicionando vibração aos quadros artificialmente. Esse método consiste em produzir um vídeo no qual todos os *frames* contém a mesma imagem. No entanto, para cada *frame*, a origem $(i, j) = (1, 1)$ da imagem é deslocada aleatoriamente, assim reproduzindo uma vibração. Esse deslocamento aleatório foi modelado através da função `c*randn()`, a qual retorna valores com variância $\sigma_X^2 = \{10, 20, 40, 80\}$ e média $\mu = 0$. O algoritmo da página seguinte resume como é aplicado esse método.

Após o vídeo ser editado contendo uma vibração conhecida, o algoritmo TSS é testado comparando o deslocamento detectado e o deslocamento inserido. O valor de `c` foi alterado para testar a eficiência do algoritmo TSS em função do valor de w , o qual delimita a região de busca, veja Seção 2.5.2. O resultado de um dos experimentos pode ser visto na Figura 5.1, que mostra o GMV da componente vertical da detecção e vibração artificial pelas curvas vermelha e preta, respectivamente. Para este caso, o erro, calculado pela equação (2.8), foi de apenas 6%. Para mais resultados, veja a Tabela 2.2.

Algorithm 2: Reprodução de vídeo com vibração a partir de uma imagem

```

catch image;
make video = null;
while video not completed do
    newframe  $\leftarrow$  image;
    CHANGE_ORIGIN(newframe);
    CUT_MARGINS(newframe);
    video.framek  $\leftarrow$  newframe;
return video;
  
```

CHANGE_ORIGIN() é a função `c*randn()` aplicado à todos os *pixels* do quadro.

CUT_MARGINS() corta as margens do quadro.

Para experimentos em ambiente real foi colocada uma câmera comum, ver Figura A.4 nos anexos, sobre o painel de um carro. A câmera foi conectado a um computador *Raspberry Pi* pela interface USB (*Universal Serial Bus*). Os experimentos foram realizados através do compilador `g++`⁴ e das bibliotecas `openCV`⁵ no *Raspberry Pi*. Os gráficos da Figura 5.2 mostram os valores de deslocamento na componente vertical encontrado em um dos experimentos.

Figura 5.1: Resultados da estimativa de movimento quando aplica-se vibração artificial.

⁴Programa que compila o código fonte em C++ para um arquivo em binário para a máquina executar o programa.

texto: http://www.vietspring.org/cpp_linux/gpp.html.

⁵Significa *Open Source Computer Vision Library*

Figura 5.2: Resultados da estimativa de movimento em um caso real.

5.2 Configuração, testes e resultados do conjunto acelerômetro e câmera

O dispositivo utilizado para coletar dados de aceleração foi o MMA7361, para detalhes veja o Anexo A. O MMA7361 foi conectado a um circuito contendo um filtro ativo passa-baixas *Butterworth* de frequência de corte 200 Hz . O sinal de saída do circuito é carregado ao microcontrolador Arduino Uno para:

- Coletar amostras a 180 Hz ;
- Conversão analógica/digital;
- Executar uma última etapa do pré-filtro, veja Seção A.1;
- Fazer a integração numérica para obtenção de x_t e y_t .

Os testes foram feitos diretamente em ambiente real colocando a câmera, acelerômetro, circuito, microcontrolador e um computador dentro de um carro. Os dados fornecidos pelo microcontrolador e a câmera foram gravados no computador para executar o algoritmo TSS adaptado no MATLAB.

Na Figura 5.3, a curva em azul mostra a posição da câmera na componente horizontal em uma janela de 800 *frames* de um dos experimentos em que o veículo executou uma trajetória retilínea. No entanto, a curva em azul sofre uma flutuação sobre o valor médio da vibração ao longo do tempo indicando que o veículo teria feito curvas. As variáveis do algoritmo TSS foram analisadas e verificou-se que $\rho_{k,k-1}$ está com valor baixo para todos os blocos, sendo mais baixo para os blocos

distantes do centro dos quadros. Apesar das etapas 2 e 3 do algoritmo TSS tentarem corrigir os valores de Δx_{im} e Δy_{im} , as amostras destes acumulam erro devido ao uso recursivo de amostras anteriores nas equações em (3.4) e (3.5). Portanto, as variáveis x_t e y_t nas equações (3.4) e (3.5) devem ser corrigidos com os valores x_{TSS_k} e y_{TSS_k} de (2.7), tornando-se assim um sistema de malha fechada. A estrutura do sistema corrigido está como apresentado na Figura 5.4. As equações (3.4) e (3.5) são corrigidas como seguem:

$$x_{t+1} = x'_{TSS_k} + v_{x,t}T + a_{x,t}\frac{T^2}{2}, \quad (5.1)$$

$$y_{t+1} = y'_{TSS_k} + v_{y,t}T + a_{y,t}\frac{T^2}{2}, \quad (5.2)$$

x'_{TSS_k} e y'_{TSS_k} representam as variáveis x_{TSS_k} e y_{TSS_k} após conversão para o sistema de coordenadas do mundo através de (3.6).

Outro problema surge devido ao fluxo ótico intenso nas margens dos quadros, causando uma discrepância entre a estimativa de movimento pelo acelerômetro e pelo algoritmo TSS. Desta forma, os blocos utilizados para fazer a estimativa de movimento através do algoritmo TSS adaptado foram reduzidos em 74%, utilizando somente os blocos localizados ao centro dos quadros.

Figura 5.3: Resposta da posição da câmera afetada por vibração. A curva em azul mostra o resultado para o sistema proposto em malha aberta e a curva em preto mostra o resultado para o sistema proposto em malha fechada.

O sinal de cor preta na Figura 5.3 mostra o resultado após a correção das equações (3.4) e (3.5) com a recorrência de x'_{TSS_k} e y'_{TSS_k} fornecido do resultado dos dados do algoritmo de visão.

A confiabilidade do sistema proposto foi avaliada comparando-se os $\rho_{k,k-1}$ desse sistema e do algoritmo TSS apresentado na Seção 2.5.2. Em média, $\rho_{k,k-1}$ dos experimentos resultou em um aumento de 11% sobre o $\rho_{k,k-1}$ do algoritmo TSS puro.

Figura 5.4: Atualização do diagrama de blocos do sistema proposto, apresentado na Figura 1.1.

5.3 Filtragem

As próximas figuras mostram gráficos das componentes horizontal e vertical em quatro ambientes para realização dos experimentos usando a configuração citada na Seção 5.2.

A Figura 5.5 e Figura 5.7 mostram as componentes horizontal e vertical de um experimento realizado com o veículo sobre asfalto durante uma curva de 90 graus executada em quatro segundos. A Figura 5.8 e Figura 5.9 mostram as componentes horizontal e vertical no momento em que o veículo passa sobre uma ondulação transversal.

Nas figuras 5.5 e 5.9, a mudança de posição varia substancialmente devido à trajetória que a câmera exerce em seu plano $i \times j$. Nesses dois casos, a resposta ao filtro média móvel e IIR têm baixa eficiência por não acompanharem instantaneamente a trajetória da câmera sobre seu plano $i \times j$. Já o filtro de Kalman, filtro BDU e FKRR mostram melhor desempenho ao acompanhar devidamente a movimentação da câmera e suavização da vibração.

Para os outros casos, veja Figura 5.7 e Figura 5.8, a câmera não exerce movimento intencional nas componentes vertical e horizontal. O filtro média móvel teve o mais baixo desempenho devido ao fator $D = 5$ adotado. O amortecimento da vibração será maior com o aumento de D . No entanto, a complexidade computacional aumentará e os resultados serão menos eficientes para os

casos da câmera exercer movimento intencional, como nos casos das figuras 5.5 e 5.9. Os resultados apresentados pelo filtro IIR adaptativo não são robustos o suficiente por causa dos *thresholds* T_1 , T_2 e T_3 escolhidos que afetam diferentemente nos resultados para cada experimento. O filtro de Kalman mostrou bons resultados comparado aos dois filtros anteriores em razão da uniformidade de deslocamento independente das diferentes amplitudes de deslocamento dos quadros ao longo do tempo. Já o resultado com filtro BDU e FKRR apresentam dinâmica de movimento parecidos com os resultados obtidos com o filtro de Kalman. As curvas dos resultados do filtro BDU e FKRR ficam mais próximas da curva de referência f' , que representa o movimento intencional da câmera livre de vibração.

Para mais resultados e exemplos, outros gráficos foram plotados através de outros experimentos. Veja as figuras 5.10, 5.11, 5.12 e 5.13. A comparação técnica entre os filtros foi realizada com o cálculo de (5.3) sobre $T = 50$ experimentos. Veja Figura 5.14.

$$\varepsilon \|x_k - \hat{x}_k\| \approx \frac{1}{T} \sum_{j=1}^T \|x_k^{(j)} - \hat{x}_k^{(j)}\|, \quad (5.3)$$

O filtro BDU obteve redução de erro de 1,9% sobre os resultados do filtro de Kalman, enquanto que o FKRR obteve 3,7% sobre os resultados do filtro de Kalman.

Para visualizar os resultados de alguns desses experimentos, veja os links listados abaixo.

<https://youtu.be/pzg5Gvf9k5w>

<https://youtu.be/XPYPLVvWXBc>

<https://youtu.be/V-ckZsIGPrk>

<https://youtu.be/9yXX6I7TwU4>

<https://youtu.be/zHS16fbIQEs>

Figura 5.5: Componente horizontal do deslocamento dos quadros em *pixels* para o experimento em que o veículo faz uma curva de 90 graus.

Figura 5.6: Zoom da Figura 5.5 entre os instantes 3,5 e 4,5 segundos.

Figura 5.7: Componente vertical do deslocamento dos quadros em *pixels* para o experimento em que o veículo faz uma curva de 90 graus.

Figura 5.8: Componente horizontal do deslocamento dos quadros em *pixels* para o experimento em que o veículo atravessa uma ondulação transversal.

Figura 5.9: Componente vertical do deslocamento dos quadros em *pixels* para o experimento em que o veículo atravessa uma ondulação transversal.

Figura 5.10: Componente horizontal do deslocamento dos quadros em *pixels* para o experimento em que o veículo executa movimentos em zigue-zague.

Figura 5.11: Componente vertical do deslocamento dos quadros em *pixels* para o experimento em que o veículo executa movimentos em zigue-zague.

Figura 5.12: Componente horizontal do deslocamento dos quadros em *pixels* para o experimento em que o veículo executa curva leve e constante.

Figura 5.13: Componente vertical do deslocamento dos quadros em *pixels* para o experimento em que o veículo executa curva leve e constante.

Figura 5.14: Variância dos filtros. Análise com $T = 50$ experimentos.

Capítulo 6

Conclusões

Neste trabalho foi desenvolvido um sistema de estabilização digital de vídeo integrado com o algoritmo TSS adaptado para receber informações de um acelerômetro que está acoplado atrás da câmera. O algoritmo TSS foi apresentado e comparado a outros algoritmos de melhor desempenho, mas estes têm maior custo computacional. O sistema proposto reduziu o custo computacional em 95% devido a remoção da terceira etapa e a redução do número de blocos no algoritmo TSS adaptativo. Foi visto um ganho médio de 11% na precisão de detecção de movimento sobre os resultados usando somente o algoritmo TSS.

Também foram apresentados o filtro média móvel, o filtro IIR adaptativo, o filtro de Kalman, o filtro BDU e o FKRR. O filtro média móvel e o filtro IIR adaptativo apresentam bom desempenho comparados aos outros filtros em experimentos específicos. Porém, seus parâmetros são inadequados quando o padrão de comportamento do movimento global da câmera muda devido ao ambiente de teste.

O algoritmo genético contribuiu para identificar valores mais adequados dos parâmetros do filtro de Kalman, filtro BDU e FKRR para melhorar sua eficiência. Um estudo comparativo mostrou que o filtro BDU e FKRR obtiveram média de redução de erro de 1,9% e 3,7%, respectivamente, sobre o filtro de Kalman.

Com base no desenvolvimento técnico discutido nesta dissertação e nos resultados obtidos propõe-se sua extensão nos tópicos:

- Implementação do algoritmo de estimativa de movimento em FPGA (*Field Programmable Gate Array*) para fornecer alto nível de paralelismo. Os benefícios proporcionam mais eficiente e redução do esforço computacional pela criação de blocos lógicos específicos;

- Estudar outros métodos ou aprimorar o algoritmo TSS que exerça detecção de movimentos de rotação para embarcar em veículos aéreos;
- Adaptação do sistema proposto para aplicação em visão estereo, podendo também fazer estimativa da distância entre as câmeras e o objeto em observação.

Referências Bibliográficas

- [1] Mies, G. (2010). Military robots of the present and the future *Atlantic Association for Research in the Mathematical Sciences*, vol. 9, no. 0, pp 125–137.
- [2] Antebi, Liran (2013). Who Will Stop the Robots? *The Institute for National Security Studies (INSS) Military and Strategic Affairs, September 2013*, vol. 5, No. 2, pp. 61–77.
- [3] Xiufeng He, Nanjing, C., Yang Le & Xiao, W. (2013). MEMS IMU and two-antenna GPS integration navigation system using interval adaptive Kalman filter. *IEEE Aerospace and Electronic Systems Magazine*, vol. 28, no. 10, pp 22–28.
- [4] Diamant, R. & Yunye Jin. (2013). A Machine Learning Approach for Dead-Reckoning Navigation at Sea Using a Single Accelerometer. *IEEE Journal of Oceanic Engineering*, vol. 39, no. 4, pp. 672–684.
- [5] Lee Chong Wan, Sebastian, P. & Voon, Y.V. (2009). Stereo Vision Tracking System. *ICFCC 2009. International Conference on Future Computer and Communication*, pp. 487–491.
- [6] Ma'sum, M.A., Arrofi, M.K., Jati, G. & Arifin, F. (2013). Simulation of intelligent Unmanned Aerial Vehicle (UAV) For military surveillance. *International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 161–166.
- [7] Folkesson, J., Leonard, J., Leederkerken, J. & Williams, R. (2007). Feature tracking for underwater navigation using sonar. IROS 2007. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3678–3684.
- [8] Xuan Song, Huijing Zhao, Jinshi Cui & Shao Xiaowei. (2010). Fusion of laser and vision for multiple targets tracking via on-line learning. *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1050–4729.

- [9] Deyle, T., Nguyen, H., Reynolds, M. & Kemp, C.C. (2009). RF vision: RFID receive signal strength indicator (RSSI) images for sensor fusion and mobile manipulation. *IROS 2009. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5553–5560.
- [10] Hirayama, T. (2013). The evolution of CMOS image sensors. *IEEE Asian Solid-State Circuits Conference*, pp. 5–8.
- [11] Micron Technology, (2006). The Evolution of Digital Imaging: From CCD to CMOS. A Micron White Paper.
- [12] Gouveia, L. C. P. & Choubey, B. (2014) On Evolution of CMOS Image Sensors. *Proceedings of the 8th International Conference on Sensing Technology*, pp. 89–94.
- [13] Roggen, D., Jenny R, de la Hamette, P., & Troster, G. (2007). Mapping by seeing: wearable vision-based dead-reckoning, and closing the loop. *EuroSSC 2007, lecture notes in computer science*, pp. 29–45.
- [14] Ballard, D. H. & Brown, C. M. (1982). Computer Vision. Prentice Hall.
- [15] Bonato, V. (2008). Proposta de uma arquitetura de hardware em FPGA implementada para o SLAM com multi-câmeras aplicada à robótica móvel. Tese (Doutorado), ICMC, Universidade de São Paulo, São Carlos.
- [16] Jia, Z., Balasuriya, A., & Challa, S. (2009). Vision based target tracking for autonomous land vehicles navigation: a brief survey. *Academic Journal - United Technologies Research Center*, vol. 2, no. 1, pp. 32–42.
- [17] He, Q., Chu, C. H., & Camargo, A. (2012). Architectural Building Detection and Tracking under Rural Environment in Video Sequences Taken by Unmanned Aircraft System (UAS). *Worldcomp'12*.
- [18] Axis Communications, (2014). Image stabilization – improving camera usability. An Axis Communications White paper.
http://www.axis.com/files/whitepaper/wp_image_stabilization_57110_en_1406_lo.pdf
- [19] Kim, J. N. & Shoi, T. S. (1998). A fast three-step search algorithm with minimun checking points using unimodal error surface assumption. *IEEE Trans. on Consumer Electronics*, vol. 44, No. 3, pp. 638–648.
- [20] Li, R., Aeng, B., & Liou, M. L. (1994). A new three-step search algorithm for block motion estimation. *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 438–442.

- [21] Ko, S.-J., Lee, S.-H., Jeon, S.-W., & Kang, E.-S. (1999). Fast digital image stabilizer based on gray-coded bit-plane matching. *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 598–603.
- [22] Tham, J. Y., Ranganath, S., Ranganath, M., & and Kassim, A. A. (1998). A novel unrestricted center-biased diamond search algorithm for block motion estimation. *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 4, pp. 369–377.
- [23] Ley, L. H. (2006). Modeling the New Three Step Search Algorithm for Motion Estimation in Image Sequence in MATLAB. Monograph (Bachelor Degree), Faculty of Electronic Engineering, Kolej Universiti Teknikal Kebangsaan Malaysia.
- [24] Identificação e Transformação de Valores Aberrantes como Medida de Confiabilidade do Método das Diferenças para Estimativa de Fluxo Óptico em Sequências de Imagens. Tese (Mestrado), FEEC, Universidade Estadual de Campinas.
- [25] Oliveira Neto, V. J., Chavez, G. C., & Menotti, D. (2012). Towards License Plate Recognition: Comparing Moving Objects Segmentation Approaches. *The 2012 International Conference on Image Processing, Computer Vision, and Pattern Recognition*, pp. 447–453.
- [26] Ryu, Y. G., Roh, H. C., Kim, S. J., Ho An, K., & Chung M. J. (2009). Digital Image Stabilization for Humanoid Eyes Inspired by Human VOR System. *Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics December*, pp. 19–23.
- [27] Windau, J. & Itti, L. (2011). Multilayer real-time video image stabilization. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2397–2402.
- [28] Nützi, G, Weiss, S., Scaramuzza, D., & Siegwart, R. (2011). Fusion of IMU and Vision for Absolute Scale Estimation in Monocular SLAM. *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1-4, pp. 287–299.
- [29] Bell, S., Troccoli, A. & Pulli, K. (2014). A Non-Linear Filter for Gyroscope-Based Video Stabilization. *Computer Vision – ECCV 2014 Lecture Notes in Computer Science*, vol. 8692, pp. 294–308.
- [30] Karpenko, A., Jacobs, D., Baek, J. & Levoy, M. (2011). Digital Video Stabilization and Rolling Shutter Correction using Gyroscopes. *A Stanford University Computer Science Tech Report CSTR 2011-03*.

- [31] Yang, S. H. & Jheng, F. M. (2006). An adaptive image stabilization technique. In Proceedings of the *IEEE International Conference on Systems, Man, and Cybernetics* (SMC2006), pp. 1968–1973.
- [32] Tanakian, M. J., Rezaei, M., & Mohanna, F. (2010). Digital Video Stabilization System by Adaptive Motion Vector Validation and Filtering. *First International Conference on Communications Engineering*, pp. 165–168.
- [33] Erturk, S. & Dennis, T. J. (2000). Image sequence stabilisation based on DFT filtering. *IEE-Proc. Vision, Image and Signal Processing*, vol. 147, no. 2, pp. 95–102.
- [34] Smith, W. (1998). The Scientist and Engineer’s Guide to Digital Signal Processing, chapter 15. California Technical.
- [35] Silvestre, A. C. (2007). Estabilização digital em tempo real de imagens em sequência de vídeos. Tese (Mestrado), Departamento de Engenharia Elétrica e de Computação, Universidade de São Paulo, São Carlos.
- [36] Brown, R. G. (1996). Introduction to Random Signals and Applied Kalman Filtering. Wiley, 3 edition.
- [37] Erturk, S. (2002). Real-Time Digital Image Stabilization Using Kalman Filters. *Real-Time Imaging*, Elsevier, vol. 8, no. 4, pp. 317–328.
- [38] Wang, C., Kim, J.-H., Byun, K.-Y., Ni, J., & Ko, S.-J. (2009). Robust Digital Image Stabilization Using the Kalman Filter. *IEEE Transactions on Consumer Electronics*, vol. 55, no. 1, pp. 6–14.
- [39] Patti, A. J., Tekalp, A. M., & Sezan, M. I. (1998) A new motion-compensated reduced-order model Kalman filter for space-varying restoration of progressive and interlaced video. *IEEE Transactions on Image Processing*, no. 7: pp. 543–554.
- [40] Sayed, A. H. (2001). A framework for state-space estimation with uncertain models. *IEEE Transaction on Automatic Control*, vol. 46, pp. 998–1013.
- [41] Terra, M. H., Ishihara, J. Y. & Inoue R. S. (2011). A new approach to robust linear filtering problems. *18th IFAC World Congress*, pp. 1174–1179.
- [42] Ventura, J. M. V. (2011). Estudo de Coordenação de Robôs Móveis com Obstáculos. Tese (Mestrado), Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

- [43] Inoue R. S. (2012). Controle robusto descentralizado de movimentos coordenados de robôs heterogêneos. Tese (Doutorado), Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.
- [44] Inoue R. S. (2012). Acompanhamento de trajetória de alvos móveis utilizando filtro robusto. *19 Simpósio Internacional de Iniciação Científica (SIICUSP)*.
- [45] Trucco, E. & Verri, A. (1998). Introductory Techniques for 3-D Computer Vision. Prentice Hall.
- [46] National Television System Committee (1951–1953), [Report and Reports of Panel No. 11, 11-A, 12-19, with Some supplementary references cited in the Reports, and the Petition for adoption of transmission standards for color television before the Federal Communications Commission, n.p., 1953], 17 v. illus., diags., tables. 28 cm. LC Control No.:54021386 Library of Congress Online Catalog.
- [47] Inman D. J. (2007) Engineering vibrations, 3rd edn. Pearson International Education (Prentice Hall), Upper Saddle River.
- [48] Hazan, A., Verleysen, M., Cottrell, M. & Lacaille, J.(2010). Trajectory Clustering for Vibration Detection in Aircraft Engines. *Advances in Data Mining. Applications and Theoretical Aspects Lecture Notes in Computer Science*, vol. 6171, pp. 362–375.
- [49] Dominguez, S. M., Keaton, T., & Sayed, A. H. (2001). Comparison of Robust Estimation and Kalman Filtering Applied to Fingertip Tracking in Human-Machine Interfaces. *Asilomar Conference on Signals Systems and Computers*, Vol. 1, pp. 342–346.
- [50] Oliveira, H. M. (2010). Engenharia de Telecomunicações. Departamento de Eletrônica & Sistemas, Universidade Federal de Pernambuco.
- [51] Won-Chul Bang, Wook Chang, Kyeong-Ho Kang, Eun-Seok Choi, Potanin, A., & Dong-Yoon Kim. (2003). Self-contained Spatial Input Device for Wearable Computers. *In Proceedings of the 7th IEEE International Symposium on Wearable Computers (ISWC'03)*, pp. 26–34.
- [52] Shannon, C. E. (1948). A mathematical theory of communication. *Bell Sys. Tech. J.*, vol. 27, pp. 379–423, 623–56.

Apêndice A

Modelagem do acelerômetro

A aceleração fornecida pelos acelerômetro é dada por

$$a = a_a + \delta a_a - b_a - w_a \quad (\text{A.1})$$

onde a_a é a aceleração medida conforme movimentação do acelerômetro, δa_a é o termo incerto de a_a , b_a é a polarização do acelerômetro e w_a o ruído Gaussiano branco das medidas do acelerômetro. O termo incerto δa_a é solucionado durante os passos 2 e 3 do algoritmo TSS adaptado apresentado no **Capítulo 3**, Seção 3.3. A polarização é medida com o acelerômetro parado na posição em que é fixado atrás da câmera. A variação da polarização nos experimentos é desprezada, uma vez que o plano XY do acelerômetro não é inclinado sobre o eixo X e Y do mesmo. O ruído Gaussiano provindo de w_a e da polarização b_a resulta em uma leve flutuação dos valores do acelerômetro quando ele está parado. Para contornar esse problema, foi adicionado um circuito com um filtro *butterworth* passa baixas com frequência de corte 200 Hz [50]. O circuito da Figura A.1 foi conectada entre os pinos do dispositivo MMA7361 e o microcontrolador Arduino Uno, veja **Anexo A**.

A leitura e transmissão dos dados do acelerômetro é feita como segue abaixo:

```
// # Connection :  
// #           x  -> Analog pin 0  
// #           y  -> Analog pin 1  
// #           z  -> Analog pin 2  
// #  
  
void setup()  
{  
    Serial.begin(19200); // 19200 bps
```

```

}
void loop()
{
  int x,y,z;
  x=analogRead(0);
  y=analogRead(1);
  z=analogRead(2);
  Serial.print("x= ");
  Serial.print(x,DEC);
  Serial.print(' ');
  Serial.print("y= ");
  Serial.print(y,DEC);
  Serial.print(' ');
  Serial.print("z= ");
  Serial.println(z,DEC);
  delay(); //ajustar frequencia desejada
}

```

A.1 Detecção de movimento

Em [51] foi proposto um método para reduzir o erro estocástico presente em qualquer unidade de medidas inerciais. O conceito em questão indica que velocidade e aceleração são nulos se o acelerômetro estiver parado, i.e. identificar se o acelerômetro está se movendo de fato. O método é comparar o desvio padrão da aceleração de sua estrutura $|\mathbf{A}_b|, \sigma_{|\mathbf{A}_b|}^S(t)$, a um *threshold* σ_{th} , como descrito no algoritmo abaixo. S é uma janela de amostras e t é o instante de tempo.

Algorithm 3: Detecção de perturbação do acelerômetro

```

initialization;
if  $\sigma_{|\mathbf{A}_b|}^S(t) < \sigma_{th}$  then
  | Stationary;
else
  | Moving;

```

A.2 Pré-filtro

Foi implementado um filtro Butterworth de segunda ordem com frequência de corte de 200 Hz para selecionar os dados que interessam do acelerômetro. O filtro Butterworth foi escolhido devido a sua resposta em frequência não ter ondulações, ou *ripple*. A função de transferência no domínio da frequência é como segue abaixo:

$$H(s) = \frac{(2\pi f_c)^2}{s^2 + 2\pi \frac{f_c}{Q} + (2\pi f_c)^2} \quad (\text{A.2})$$

onde f_c é a frequência de corte. $Q = 0,707$ é o fator de qualidade do filtro Butterworth. A figura abaixo mostra o esquemático contendo os componentes utilizados para implementar o filtro.

Figura A.1: Circuito filtro Butterworth de segunda ordem e frequência de corte 200 Hz.

Apêndice B

Códigos

```
function LMVs = TSS3( doub_c, doub_p )
% Aplicacao do algoritmo three-step search para w=7 ou 8;
% cframe e o frame atual; pframe e o frame anterior;
% Os frames tem formato double da componente Green de uma imagem;
% A saida retorna um vetor com dimensoes: linhas (sentido Y), colunas
%(sendido X).

tam1=size(doub_c,1)/16;
tam2=size(doub_c,2)/16;
aux=zeros(tam1,tam2,3);    % info de todos os lmvs: linhas por colunas
                           % por x,y,pxy

aux2=0;
orient=zeros(1,2);         %auxilio na orientacao e etapa do algoritmo,
                           %recebe Linha, coluna
e=[4 2 1];                 %somador por etapa (1 2 e 3)
m=1;                       %numero da etapa

for k=3:(tam1-2)
    bloco_v=(k-1)*16+1;
    for l=3:(tam2-2)
        bloco_h=(l-1)*16+1;
        p_lc=zeros(1,9);
        aux_vetor=zeros(1,2);
        pmedia=0;          %media do bloco
        den2=0;            %variaveis auxiliares para matriz de correlacao
%testar MSE
        test_mse=my_mse(doub_c(bloco_v:(bloco_v+16),bloco_h:(bloco_h+16)));
        if( test_mse > 0.12 )
```

```

for m=1:3                                %estagios do tss
    cmedia=zeros(1,9);
    num=zeros(1,9);
    den1=zeros(1,9);
    for i=bloco_v:(bloco_v+16)            %percorre linha (y)
        iv=i+aux_vetor(1);
        for j=bloco_h:(bloco_h+16)        %percorre coluna (x)
            jv=j+aux_vetor(2);
            %posicao1->central :
            cmedia(1)=cmedia(1) + doub_c(iv,jv);
            %posicao2->noroeste :
            cmedia(2)=cmedia(2) + doub_c(iv-e(m),jv-e(m));
            %posicao3->norte :
            cmedia(3)=cmedia(3) + doub_c(iv-e(m),jv);
            %posicao4->nordeste :
            cmedia(4)=cmedia(4) + doub_c(iv-e(m),jv+e(m));
            %posicao5->oeste :
            cmedia(5)=cmedia(5) + doub_c(iv,jv-e(m));
            %posicao6->leste :
            cmedia(6)=cmedia(6) + doub_c(iv,jv+e(m));
            %posicao7->sudoeste :
            cmedia(7)=cmedia(7) + doub_c(iv+e(m),jv-e(m));
            %posicao8->sul :
            cmedia(8)=cmedia(8) + doub_c(iv+e(m),jv);
            %posicao3->sudeste :
            cmedia(9)=cmedia(9) + doub_c(iv+e(m),jv+e(m));
            if (m==1)
                pmedia=pmedia + doub_p(i,j);
            end
        end
    end
    cmedia=cmedia/256;
    if (m==1)
        pmedia=pmedia/256;
    end
    for i=bloco_v:(bloco_v+16)
        iv=i+aux_vetor(1);
        for j=bloco_h:(bloco_h+16)
            jv=j+aux_vetor(2);
            num(1)=num(1)+(doub_c(iv,jv)-cmedia(1))*(doub_p(i,j)-pmedia);
            num(2)=num(2)+(doub_c(iv-e(m),jv-e(m))-cmedia(2))*(doub_p(i,j)-pmedia);
            num(3)=num(3)+(doub_c(iv-e(m),jv)-cmedia(3))*(doub_p(i,j)-pmedia);

```

```

num(4)=num(4)+(doub_c(iv-e(m),jv+e(m))-cmedia(4))*(doub_p(i,j)-pmedia);
num(5)=num(5)+(doub_c(iv,jv-e(m))-cmedia(5))*(doub_p(i,j)-pmedia);
num(6)=num(6)+(doub_c(iv,jv+e(m))-cmedia(6))*(doub_p(i,j)-pmedia);
num(7)=num(7)+(doub_c(iv+e(m),jv-e(m))-cmedia(7))*(doub_p(i,j)-pmedia);
num(8)=num(8)+(doub_c(iv+e(m),jv)-cmedia(8))*(doub_p(i,j)-pmedia);
num(9)=num(9)+(doub_c(iv+e(m),jv+e(m))-cmedia(9))*(doub_p(i,j)-pmedia);
den1(1)=den1(1)+(doub_c(iv,jv)-cmedia(1))^2;
den1(2)=den1(2)+(doub_c(iv-e(m),jv-e(m))-cmedia(2))^2;
den1(3)=den1(3)+(doub_c(iv-e(m),jv)-cmedia(3))^2;
den1(4)=den1(4)+(doub_c(iv-e(m),jv+e(m))-cmedia(4))^2;
den1(5)=den1(5)+(doub_c(iv,jv-e(m))-cmedia(5))^2;
den1(6)=den1(6)+(doub_c(iv,jv+e(m))-cmedia(6))^2;
den1(7)=den1(7)+(doub_c(iv+e(m),jv-e(m))-cmedia(7))^2;
den1(8)=den1(8)+(doub_c(iv+e(m),jv)-cmedia(8))^2;
den1(9)=den1(9)+(doub_c(iv+e(m),jv+e(m))-cmedia(9))^2;
if(m==1)
    den2=den2 + (doub_p(i,j)-pmedia)^2;
end
end
end
p_lc=num./ (sqrt(den1)*sqrt(den2));
[orient aux2]=orient_TSS(p_lc,m);
aux_vetor=aux_vetor+orient;
end
aux(k,l,1)=aux_vetor(1); %linha (Y)
aux(k,l,2)=aux_vetor(2); %coluna (X)
aux(k,l,3)=aux2;
else
    aux(k,l,1)=100; %fora
    aux(k,l,2)=100; %fora
    aux(k,l,3)=aux2;
end
end
end
LMVs=aux; %retorna orientacao da linha (Y) por coluna (X)

```

```
function saida = my_mse( mat1616 )

% Calcula o MSE entre os elementos adjacentes da matriz;
% A entrada e uma matriz 16 por 16 do tipo double.

aux1=0;
aux2=0;

for i=1:16
    for j=2:16
        aux1=aux1 + abs(mat1616(i,j)-mat1616(i,j-1));
    end
end
for j=1:16
    for i=2:16
        aux2=aux2 + abs(mat1616(i,j)-mat1616(i-1,j));
    end
end
saida=sqrt(aux1+aux2)/350;

end
```

```

function [vec_lc aux] = orient_TSS( vetor , etapa )
% Funcao compara os dados do VETOR, e retorna o maior valor e a orientacao
% Detailed explanation goes here

maior=vetor(1);
indice=1;
aux_orient = zeros(1,2);
e=[4 2 1];          %somador por etapa (1 2 e 3)
for i=2:9
    if(maior < vetor(i))
        maior = vetor(i);
        indice=i;
    end
end
switch indice
    case 2
        aux_orient=aux_orient-1;
    case 3
        aux_orient=[-1 0];
    case 4
        aux_orient=[-1 1];
    case 5
        aux_orient=[0 -1];
    case 6
        aux_orient=[0 1];
    case 7
        aux_orient=[1 -1];
    case 8
        aux_orient=[1 0];
    case 9
        aux_orient=aux_orient+1;
end
if(etapa==1)
    aux_orient=e(1)*aux_orient;      %aux_orient=4*aux_orient;
elseif(etapa==2)
    aux_orient=e(2)*aux_orient;      %aux_orient=2*aux_orient;
end
vec_lc = aux_orient;
aux = maior;
end

```

```
function [GMVlc n_valid] = validar_lmv( vetor3d )
%UNTITLED Summary of this function goes here
% Detailed explanation goes here
aux_l=0;
aux_c=0;
k=0;
tam1=size(vetor3d,1)-2;
tam2=size(vetor3d,2)-2;
for i=3:tam1
    for j=3:tam2
        if (vetor3d(i,j,1)<100)
            aux_l=aux_l + vetor3d(i,j,3)*vetor3d(i,j,1);
            aux_c=aux_c + vetor3d(i,j,3)*vetor3d(i,j,2);
            k=k + vetor3d(i,j,3);
        end
    end
end
if(k>0)
    GMVlc=[aux_l/k; aux_c/k];
else
    GMVlc=[0; 0];
end
n_valid=k;
end
```

```
function saida = fmm( ent )
%UNTITLED3 Summary of this function goes here
% Detailed explanation goes here
y=zeros(1,200);
for i=2:200
    if (i==2)
        y(i)=(ent(i)+ent(i-1))/2;
    elseif (i==3)
        y(i)=(ent(i)+ent(i-1)+ent(i-2))/3;
    elseif (i==4)
        y(i)=(ent(i)+ent(i-1)+ent(i-2)+ent(i-3))/4;
    else
        y(i)=(ent(i)+ent(i-1)+ent(i-2)+ent(i-3)+ent(i-4))/5;
    end
end
saida=y;

end
```

```
function saida = iir( ent )
%UNTITLED2 Summary of this function goes here
% Detailed explanation goes here
y=zeros(1,200);
dv=zeros(1,200);
y(1)=ent(1);
for i=2:200
    if (i==2)
        y(2)=ent(2);
        dv(2)=ent(2)-y(1);
    elseif (i==3)
        y(3)=ent(3);
        dv(3)=ent(3)-y(2);
    else
        dv(i)=ent(i)-y(i-1);
        if ( abs(dv(i))>4 && (abs(dv(i-1))>3) && (abs(dv(i-2))>3) )
            y(i)=0.2*y(i-1)+0.8*ent(i);
        else
            y(i)=0.9*y(i-1)+0.1*ent(i);
        end
    end
end
saida=y;

end
```

```

function estado = kalman( medida,nframes )
%UNTITLED Summary of this function goes here

f=[1 0; 0 1]; g=[1 0; 0 1]; h=[1 0; 0 1]; q=[8.6 0; 0 8.6]; r=[64 0; 0 64];
xp=zeros(2,1,nframes); xa=xp;
p=zeros(2,2,nframes); pa=p; kk=p;
xa(:,1,1)=[0; 0];
pa(:,1,1)=[50 0; 0 50];

for j=2:nframes
    %propagacao
    xp(:,1,j)=f*xa(:,1,j-1);
    p(:,1,j)=f*pa(:,1,j-1)+g*q*g;
    %atualizacao
    kk(:,1,j)=p(:,1,j)*h*inv(h*p(:,1,j)+r);
    pa(:,1,j)=p(:,1,j)-kk(:,1,j)*h*p(:,1,j);
    xa(:,1,j)=xp(:,1,j)+kk(:,1,j)*(medida(:,1,j)-h*xp(:,1,j)));
end
estado=xa;
end

```

```

function estado = bdu( medida,nframes )
%UNTITLED Summary of this function goes here

i=eye(2);
h=[1 0; 0 1]; r=[8^2 0; 0 8^2];
xp=zeros(2,1,nframes); xa=xp;
p=zeros(2,2,nframes); pa=p; kk=p;
f=p; g=p; q=p; p_est=p;
nf=[0.016 0; 0 0.016]; ng=[0.016 0; 0 0.016];
%estimativas iniciais
f(:,:,1)=[1 0; 0 1]; g(:,:,1)=[1 0; 0 1]; q(:,:,1)=[8.6 0; 0 8.6];
pa(:,:,1)=[50 0; 0 50]; xa(:,1,1)=[0;0];
M=1.01;
aux=M*h*inv(r)*h*M;
lambda=1.5*norm(aux);
R=r-h*M*M*h/lambda;
for j=2:nframes
    p_est(:,:,j)=inv(pa(:,:,j-1))+lambda*nf*nf;
    p_est(:,:,j)=inv(p_est(:,:,j));
    g(:,:,j)=g(:,:,1)-lambda*f(:,:,1)*p_est(:,:,j)*nf*ng;
    q(:,:,j)=inv(q(:,:,1))+lambda*ng*inv(i+lambda*nf*pa(:,:,j-1)*nf)*ng;
    q(:,:,j)=inv(q(:,:,j));
    f(:,:,j)= ( f(:,:,1)-lambda*g(:,:,j)*q(:,:,j)*ng*nf)*(i-lambda*p_est(j)*nf*nf);
    %propagacao
    xp(:,1,j)=f(:,:,j)*xa(:,1,j-1);
    p(:,:,j)=f(:,:,j)*p_est(:,:,j)*f(:,:,j)+g(:,:,j)*q(:,:,j)*g(:,:,j);
    %atualizacao
    kk(:,:,j)=p(:,:,j)*h*inv(h*p(:,:,j)*h+R);
    pa(:,:,j)=p(:,:,j)-kk(:,:,j)*h*p(:,:,j);
    xa(:,1,j)=xp(:,1,j)+kk(:,:,j)*(medida(:,1,j)-h*xp(:,1,j)));
end
estado=xa;
end

```

```

function estado = fkrr( medida,nframes )
%UNTITLED Summary of this function goes here
%Filtro de kalman robusto recursivo

o=zeros(2,2); i=eye(2);
i44=eye(4); o24=[o o]; o42=o24'; o44=[o42 o42];
%*****
f=i; g=i; h=i; q=90*i; r=(60^2)*i;
nf=0.0001*i; ng=0.0001*i; nh=0.0001*i;

ff=[f;h]; gg=[g o;o i]; ee=[-i;o];
nff=[nf;nh]; ngg=[ng o; o o]; nee=[o;o]; rr=[q o;o r];
% tentar usar nh diferente de nf.
xp=zeros(2,1,nframes); p=zeros(2,2,nframes); xa=xp;
p(:, :, 1)=50*i; xa(:, 1, 1)=[0;0];

%bk=[-f*xa(:, :, j);medida(:, :, j)-h*xa(:, :, j)];
%Nbk=[-nf*xa(:, :, j);0;0];

A2=[o42 rr o44 o44 o42 i44 o42; o42 o44 o44 o44 ff gg ee; o42 o44 o44 i44 nff ngg nee];
A2=[A2;i o24 ff' nff' o o24 o; o42 i44 gg' ngg' o42 o44 o42; o o24 ee' nee' o o24 o];
%A2 e fixo (nao varia)

A1=[p(:, :, 1) o24 o24 o24 i o24 o];
A=[A1;A2];
B=inv(A);

%e=10;
L1=[o o24 o24 o24 i o24 o]; L2=[o o24 o24 o24 o o24 i]; L4=[o;o;o;o;o;o;o;o;o;o;-i];

for j=2:nframes
    bk=[-f*xa(:, 1, j-1); medida(:, 1, j) - h*xp(:, 1, j-1)];
    nb=[-nf*xa(:, 1, j-1);-nh*xp(:, 1, j-1)];
    L3=[0;0;0;0;0;0;bk;nb;0;0;0;0;0;0;0;0];
    xp(:, 1, j)=L2*B*L3;
    p(:, :, j)=L2*B*L4;
    A1=[p(:, :, j) o24 o24 o24 i o24 o];
    A=[A1;A2];
    B=inv(A);
    xa(:, 1, j)=xp(:, 1, j) + L1*B*L3;
end

```

```
estado=xa;
```

```
end
```

```

function saida = alg_gen_kalman( medida , aprox , r , nframes )

% A saida e o Q;
% medida e o sinal com vibracao;
% aprox e o sinal que se aproxima do esperado;
% o algoritmo procura encontrar um valor para Q de forma que o filtro retorne uma
%curva proxima da curva aprox.

f=[1 0; 0 1]; g=[1 0; 0 1]; h=[1 0; 0 1];
q=0.1:0.1:10; Q=[q(1) 0; 0 q(1)]; Q_aux=Q;      %100 valores possiveis
R=[r^2 0; 0 r^2];

Err_ant = 0;
Err = 0;

for i=1:100
    Q=[q(i) 0; 0 q(i)];
    xp=zeros(2,1,nframes); xa=xp;
    p=zeros(2,2,nframes); pa=p; kk=p;
    xa(:,1,1)=[0; 0];
    pa(:,1,1)=[50 0; 0 50];
    integ = [0;0];
    for j=2:nframes
        %propagacao
        xp(:,1,j)=f*xa(:,1,j-1);
        p(:,1,j)=f*pa(:,1,j-1)+g*Q*g;
        %atualizacao
        kk(:,1,j)=p(:,1,j)*h*inv(h*p(:,1,j)+R);
        pa(:,1,j)=p(:,1,j)-kk(:,1,j)*h*p(:,1,j);
        xa(:,1,j)=xp(:,1,j)+kk(:,1,j)*(medida(:,1,j)-h*xp(:,1,j)));

        integ = integ + abs(xa(:,1,j)-aprox(:,1,j));
    end
    integ=integ/nframes;
    Err = norm(integ);
    if (i==1)
        Err_ant = Err;
        Q_aux=Q;
    elseif (i>1 && Err<Err_ant)
        Err_ant = Err;
        Q_aux=Q;
    end
end

```

```
end  
saida=Q_aux;  
end
```

```

function [ngnf_est m_est] = alg_gen_bdu( medida , aprox , r_init , q_init, nframes )
%UNTITLED Summary of this function goes here
% As saidas sao NF e NG;
% medida e o sinal com vibracao;
% aprox e o sinal que se aproxima do esperado;
% o algoritmo procura encontrar um valor para Q de forma que o filtro retorne uma
%curva proxima da curva aprox.

h=[1 0; 0 1];
r=[r_init^2 0; 0 r_init^2];
nf=0.01:0.01:1;
%NF=[nf(1) 0; 0 nf(1)];
%ng=nf;
%NG=[ng(1) 0; 0 ng(1)];
dk=0.3:0.1:0.9;
M=1./dk; %nove valores
%aux=M*h*inv(r)*h*M;
%lambda=1.5*norm(aux);
Err_ant = 0;
%Err = 0;
ngnf_aux=0;
m_aux=0;

for k=1:7
    lambda=1.5*norm(M(k)*h*inv(r)*h*M(k));
    R=r-(1/lambda)*h*M(k)*M(k)*h;
    for i=1:100
        xp=zeros(2,1,nframes); xa=xp;
        p=zeros(2,2,nframes); pa=p; kk=p;
        f=zeros(2,2,nframes); g=p; q=p; p_est=p;
        f(:,:,1)=[1 0; 0 1]; g(:,:,1)=[1 0; 0 1]; q(:,:,1)=[q_init 0; 0 q_init];
        xa(:,1,1)=[1040; 0];
        pa(:,1,1)=[50 0; 0 50];
        NF=[nf(i) 0; 0 nf(i)];
        NG=[ng(i) 0; 0 ng(i)];
        integ = [0;0];
        for j=2:nframes
            p_est(:,:,j)=inv(pa(:,:,j-1))+lambda*NF*NF;
            p_est(:,:,j)=inv(p_est(:,:,j));
            g(:,:,j)=g(:,:,1)-lambda*f(:,:,1)*p_est(:,:,j)*NF*NG;
            q(:,:,j)=inv(q(:,:,1))+lambda*NG*inv(i+lambda*NF*pa(:,:,j-1)*NF)*NG;
        end
    end
end

```

```

    q(:, :, j) = inv(q(:, :, j));
    f(:, :, j) = ( f(:, :, 1) - lambda*g(:, :, j)*q(:, :, j)*NG*NF)*(i - lambda*p_est(j)*NF*NF);
    %propagacao
    xp(:, 1, j) = f(:, :, j)*xa(:, 1, j-1);
    p(:, :, j) = f(:, :, j)*p_est(:, :, j)*f(:, :, j) + g(:, :, j)*q(:, :, j)*g(:, :, j);
    %atualizacao
    kk(:, :, j) = p(:, :, j)*h*inv(h*p(:, :, j)*h+R);
    pa(:, :, j) = p(:, :, j) - kk(:, :, j)*h*p(:, :, j);
    xa(:, 1, j) = xp(:, 1, j) + kk(:, :, j)*(medida(:, 1, j) - h*xp(:, 1, j));

    integ = integ + abs(xa(:, 1, j) - aprox(:, 1, j));
end

integ = integ / nframes;
Err = norm(integ);
if (k==1 && i==1)
    Err_ant = Err;
    m_aux = M(k);
    ngnf_aux = nf(i);
elseif (Err < Err_ant)
    Err_ant = Err;
    m_aux = M(k);
    ngnf_aux = nf(i);
end
end

ngnf_est = ngnf_aux;
m_est = m_aux;
end

```

```

function [q_est r_est integral]= alg_gen_fkrr2( medida , aprox , nframes )

% A saida e o NH;
% avaliar parte final da curva

o=zeros(2,2); i=eye(2);
i44=eye(4); o24=[o o]; o42=o24'; o44=[o42 o42];
%*****
f=i; g=i; h=i; %q=q_init*i; r=r_init^2*i;
nf=0.0005*i; ng=0.0005*i; nh=0.0005*i;
q=88:0.1:91.9;
r=51:60;
r=r.^2;

ff=[f;h]; gg=[g o;o i]; ee=[-i;o];
ngg=[ng o; o o]; nee=[o;o]; %rr=[q o;o r];
nff=[nf;nh];

Err_ant = 0;
%Err = 0;
q_aux=0;
r_aux=0;

L1=[o o24 o24 o24 i o24 o]; L2=[o o24 o24 o24 o o24 i]; L4=[o;o;o;o;o;o;o;o;o;o;-i];

for k2=1:10
    for k=1:40
        xp=zeros(2,1,nframes); p=zeros(2,2,nframes); xa=xp;
        p(:,:,1)=50*i; xa(:,1,1)=[1100;0];

        rr=[q(k)*i o;o r(k2)*i];
A2=[o42 rr o44 o44 o42 i44 o42; o42 o44 o44 o44 ff gg ee; o42 o44 o44 i44 nff ngg nee];
A3=[A2;i o24 ff' nff' o o24 o; o42 i44 gg' ngg' o42 o44 o42; o o24 ee' nee' o o24 o];

        A1=[p(:,:,1) o24 o24 o24 i o24 o];
        A=[A1;A3];
        B=inv(A);

        integ = [0;0];
        for j=2:nframes

```

```

    bk=[-f*xa(:,1,j-1); medida(:,1,j) - h*xp(:,1,j-1)];
    nb=[-nf*xa(:,1,j-1);-nh*i*xp(:,1,j-1)];
    L3=[0;0;0;0;0;0;bk;nb;0;0;0;0;0;0;0;0];
    xp(:,1,j)=L2*B*L3;
    p(:, :, j)=L2*B*L4; %o problema estah aqui
    A1=[p(:, :, j) o24 o24 o24 i o24 o];
    A=[A1;A3];
    B=inv(A);
    xa(:,1,j)=xp(:,1,j) + L1*B*L3;

    integ = integ + abs(xa(:,1,j)-aprox(:,1,j));

end

integ=integ/nframes;
Err = norm(integ);
if (k==1 && k2==1)
    Err_ant = Err;
    q_aux=q(k);
    r_aux=r(k2);

elseif (Err<Err_ant)
    Err_ant = Err;
    r_aux=r(k2);
    q_aux=q(k);
end
end

end

q_est = q_aux;
r_est=sqrt(r_aux);
integral = Err_ant;
end

```

Anexo A

Equipamentos

A.1 MMA7361

O acelerômetro MMA7361 foi utilizado para detectar a vibração da câmera, veja a Figura A1. Segue abaixo as especificações do MMA7361:

- Baixo consumo de corrente: $400\ \mu$. No modo *Sleep*: $3\ \mu A$;
- Alta sensibilidade: $800\ mV/g$ no modo $1.5g$;
- Seleção de sensibilidade($\pm 1.5g$, $\pm 6g$);
- Resposta rápida ao acionamento: $0.5\ ms$;
- Dimensões: $28mm \times 17mm$.

Figura A.1: Acelerômetro MMA7361. Figura retirada do manual do fabricante.

A Tabela A.1 mostra a definição dos pinos do MMA7361.

Tabela A.1: Descrição dos pinos do MMA7361.

Pino	Nome	Tipo	Descrição
1	5V	P	Alimentação de 5V
2	3V3	P	Alimentação de 3.3V
3	GND	GND	<i>ground</i>
4	g-select	I	Modo teste
5	Selftest	I	Modo teste
6	X	A	Dados de aceleração na componente X
7	Y	A	Dados de aceleração na componente Y
8	Z	A	Dados de aceleração na componente Z
9	sleep	I	Modo <i>sleep</i>
10	0g-detect	O	Linear <i>freefall</i>

A: Saída analógica I: Entrada digital O: Saída digital

A interface é feita diretamente com o Arduino, veja próxima seção.

A.2 Arduino Uno

O Arduino Uno® é um microcontrolador baseado no microprocessador ATmega 328 16 MHz. O Arduino Uno tem capacidade de fazer interface de 14 entradas e saídas digitais, 6 entradas analógicas e comunicação serial por uma porta USB (*Universal Serial Bus*). O ATmega 328 tem capacidade de fazer a conversão analógica digital à 10 KHz, memória RAM de 32 kb, SRAM de 2 kb, EEPROM de 1 kb e consumo de corrente de 50 mA. Veja a Figura A2.

A aquisição de dados do acelerômetro é feito pelo Arduino Uno, o qual faz a conversão analógica digital e transmite o sinal digitalizado para o computador.

A.3 Raspberry Pi

O *Raspberry Pi*® é uma mini-computador desenvolvido no Reino Unido pela *Raspberry Pi Foundation* com a intenção de estimular o ensino da ciência da computação básica nas escolas. O *Raspberry Pi* é baseado no sistema *Broadcom* BCM2835 em um chip, o qual inclui um processador

Figura A.2: Microcontrolador Arduino Uno. Figura retirada do manual do fabricante.

ARM1176JZF-S 700 MHz, unidade de processamento gráfico *VideoCore IV*, e 512 *megabytes* de memória RAM. É necessário um cartão de memória SD para instalação do sistema operacional e armazenamento de dados.

Figura A.3: Raspberry Pi. Foto: <http://www.pcmag.com.br/us/article2/0,2817,2407058,00.asp>

A.4 Câmera

A Figura A4 mostra a câmera utilizada nos experimentos. Essa contém um sensor de imagem CMOS, transmite quadros à taxa de 30 Hz com resolução configurável de 352×288 à 2560×1920 correspondente à 5 mega *pixels* e transmite dados por interface USB. A distância focal é 8 *cm* e o peso total é 40 *g*.

Figura A.4: *Mini PC-camera*. Foto retirada do fabricante.