Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma Metodologia de Integração de Dispositivos de Interação Homem-Máquina ao Google Sketchup

Célio Fonseca Latorraca

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Marcus Vinicius Lamar

Coorientador
Emerson Lopes Machado

Brasília
2013

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Maristela Terto de Holanda

Banca examinadora composta por:

Prof. Marcus Vinicius Lamar (Orientador) — CIC/UnB
Emerson Lopes Machado — CIC/UnB
Flávio de Barros Vidal — CIC/UnB

CIP — Catalogação Internacional na Publicação

Latorraca, Célio Fonseca.

Uma Metodologia de Integração de Dispositivos de Interação Homem-Máquina ao Google Sketchup / Célio Fonseca Latorraca. Brasília : UnB, 2013.
107 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2013.

1. imersão, 2. realidade, 3. virtual, 4. imu, 5. filtro, 6. passa-baixas, 7. kalman, 8. sketchup

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma Metodologia de Integração de Dispositivos de Interação Homem-Máquina ao Google Sketchup

Célio Fonseca Latorraca

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Marcus Vinicius Lamar (Orientador)
CIC/UnB

Emerson Lopes Machado
CIC/UnB

Flávio de Barros Vidal
CIC/UnB

Prof. Maristela Terto de Holanda
Coordenador do Bacharelado em Ciência da Computação

Brasília, 20 de fevereiro de 2013

Dedicatória

Dedico à minha família e aos meus amigos, por acreditarem em mim e ficarem ao meu lado nos momentos mais difíceis de dúvidas e incertezas.

Agradecimentos

Agradeço a Deus que me deu coragem e perseverança para cumprir esta minha tarefa.

Ao meu orientador Prof. Marcus Vinicius Lamar pela condução e correção desta monografia.

À minha mãe Maria Celi pela ajuda na finalização ortográfica deste trabalho.

Ao meu coorientador e amigo Emerson Lopes Machado pela idéia do assunto e apoio no decorrer do projeto.

Aos meus amigos, em especial a minha namorada Karina Ornelas, que não reclamaram da minha ausência neste momento de conclusão de curso.

Resumo

Este projeto trata-se de uma metodologia para integração de um dispositivo de interação Homem-Máquina com o Google Sketchup, onde uma placa proprietária conhecida como Razor foi escolhida para aquisição dos dados de movimentação da cabeça do usuário. Esta placa utiliza uma Unidade de Medição Inercial (IMU) com três sensores: girômetro, acelerômetro e magnetômetro. Os filtros de fusão de sensores: Filtro de Kalman e DCM, foram analisados para fundir os dados capturados pela IMU. Para diminuição dos ruídos nos sinais, filtros passa-baixas: FIR e IIR, foram projetados, implementados e testados com os dados reais do projeto. O *plugin*, criado para o Google Sketchup, rotaciona a câmera do *software* de acordo com os dados lidos pelo módulo de aquisição. Assim, foi possível obter com esse projeto, um protótipo que viabilizasse a imersão de usuários em um ambiente de realidade virtual, simulado pelo Google Sketchup.

Palavras-chave: imersão, realidade, virtual, imu, filtro, passa-baixas, kalman, sketchup

Abstract

In this project is proposed a methodology for a human-machine interaction integration device through Google Sketchup interface. Such metodology uses a proprietary board known as Razor, choosen for the aquisition of the head's user position data. The board uses an Inercial Measurement Unit (IMU) with three sensors: gyrometer, accelerometer and magnetometer. The sensors filters of fusion - Kalman filter and DCM - were analised to merge the data captured by the IMU. To attenuate noise, lowpass filters - FIR and IIR - were projected, implemented and tested with real data of the project. The plugin, created for Google Sketchup, rotates the camera of software according to data read from acquisition module. Insomuch, was possible to obtain with this project a prototype that enables the immersion of users within a virtual reality environment, simulated by Google Sketchup.

Keywords: immersion, reality, virtual, imu, filter, low-pass, kalman, sketchup

Sumário

Lista de Figuras

Capítulo 1

Introdução

Neste capítulo é apresentada a especificação geral do problema abordado, sua relevância e as áreas de pesquisas relacionadas.

1.1 Definição do Problema

Embora muitos acreditem que o assunto da imersão virtual é um fenômeno novo, seus fundamentos são vistos na história da arte antiga, em imagens que faziam com que o espectador se sentisse imerso no mundo ali apresentado [16]. A imersão em um ambiente de realidade virtual faz com que o usuário sinta, ou tenha a sensação, que ele faz parte daquele mundo. Uma experiência dessa, dependendo do grau de imersão, leva o espectador a esquecer do mundo real e se focar apenas no ambiente virtual em que se encontra [33].

Em 1968, Ivan Sutherland criou o primeiro sistema de realidade virtual [25]. Nele, eram usados dispositivos para monitorar o movimento da cabeça e, dessa forma, permitir que os usuários ocupassem o mesmo ambiente que outros objetos virtuais. Como aplicações em potencial, a realidade virtual abrange tanto áreas da arquitetura, simulação de ambientes e entretenimento, quanto treinamento militar [25].

Neste trabalho, será definido "realidade virtual"como um sistema que permite ao usuário movimentar sua cabeça e ver a sua reprodução virtual agir da mesma maneira, interagindo com o ambiente artificial em que se encontra imerso.

A imersão virtual é uma área de pesquisa que busca aumentar a interação do usuário com um ambiente remoto artificial. Ela também é vista como uma parte de um tópico mais abrangente que é a Telepresença [3].

Telepresença é a percepção de estar fisicamente em um ambiente mediado por computador. A telepresença é retratada no filme Matrix (1999), onde as pessoas que estavam imersas no ambiente virtual não interagiam com o mundo real. A interação acontecia apenas entre os componentes do mundo virtual. A teleoperação e a realidade virtual são os principais aspectos dos ambientes artificiais, em que ambas envolvem um agente humano interagindo com um ambiente virtual, mediado por um computador. Na teleoperação, o mundo remoto é real e o usuário interage com ele através de vídeos remotos, manipuladores robóticos, entre outros dispositivos controlados remotamente [4]. Agora, quando se fala em realidade virtual, a diferença é que, nessa perspectiva, o mundo remoto é um ambiente simulado pelo computador. Ambos aspectos dão a mesma experiência ao usuário, sendo a única diferença o mundo experimentado [3].

Para imergir o usuário em um mundo artifical, é preciso monitorar a sua movimentação, para, enfim, projetar sua representação no ambiente virtual. Para isso, utiliza-se uma Unidade de Medida Inercial (IMU) para rastrear os movimentos feitos pela cabeça do agente. Uma IMU consiste-se em um sistema com diversos sensores, que capturam informações da movimentação da cabeça do agente e os interpreta. [26]. Cada um desses dispositivos tem suas peculiaridades. O acelerômetro é muito sensível a vibrações e não é muito preciso, já o girômetro tem problemas com *drift* (propagação de erro ao medir repetidas vezes o ângulo de visão), onde a imagem de desloca, fazendo com que a posição inicial não seja a que realmente se iniciou a movimentação.

Para resolver os problemas encontrados nos diferentes sensores existentes, testes foram feitos com o intuito de minimizar os erros e melhorar a precisão dos sinais lidos pos eles. Aplicação de filtros que integram diversos sensores são os mais utilizados, como é caso dos filtros de Ângulo Complementar [2].

Os principais dispositivos de interação Homem-Máquina, que buscam imergir um usuário em ambiente de realidade virtual, estão focados em áreas mais restritas. As forças armadas, por exemplo, utilizam essas técnicas para aplicar em seus soldados várias formas de treinamento militar.

Já existem dispositivos de *headtracking* no mercado, mas para integrá-los com os programas existentes, é necessário que haja um grande gasto na integração. Com isso, viu-se necessário a criação de uma metodologia para facilitar essa integração entre dispositivos de interação Homem-Máquina com *softwares* em geral, tornando a utilização desse tipo de componente mais interessante.

1.2 Objetivo

Este trabalho visa:

- Propor uma metodologia de integração de dispostivos de interação Homem-Máquina com o Google Sketchup, visando simplificar a integração entre outros dispositivos, que monitoram a movimentação do usuário, e sistemas computacionais, aplicando essa tecnologia ao cotidiano das pessoas;
- Criar um protótipo de *headtracking* para rastrear a movimentação do usuário e interagir com o ambiente artificial projetado no Sketchup;
- Apresentar uma interface configurável para que seja possível modificar os componentes utilizados pelo sistema. Tanto o dispositivo de *headtracking* quanto o *software* para apresentação, podem ser configurados e adequados às necessidades da aplicação.

No capítulo 2 é apresentado a fundamentação teórica para entender melhor os conceitos utilizados nesta monografia. No capítulo 3 é tratado a metodologia proposta para a solução do problema abordado por este projeto. No capítulo 4 é mostrado os resultados obtidos ao executar a metodologia que foi proposta. Finalmente, no capítulo 5 é apresentado as conclusões desse trabalho.

Capítulo 2

Fundamentação Teórica

Neste capítulo será abordada uma visão da literatura em realidade virtual, mais precisamente a imersão na realidade virtual. O foco dado ao assunto será voltado para a parte tecnológica da imersão virtual, e não para o lado filosófico e artístico do tema.

2.1 Realidade Virtual

A realidade virtual também é conhecida como virtualidade. A comunidade científica vem trabalhando no campo da Realidade Virtual (VR - do inglês, *Virtual Reality*) há décadas, e vem reconhecendo essa área como uma poderosa forma de interação entre o homem e o computador.

Antes de definir o que é VR, é preciso falar o que não é. Muitas vezes as pesquisas a confundem com a telepresença, que diz respeito a imersão em um ambiente real remoto. A telepresença é muito útil em Telerobótica, onde se busca controlar robôs à distância, sendo que o conhecimento do ambiente em volta do robô, é imprescindível. Outro assunto comumente confundido é a Realidade Aumentada (AR - do inglês, *Augmented reality*), onde objetos, ou textos computacionais são projetados sobre alguma imagem real. Ambas, telepresença e AR, incorporam informações extras a objetos do mundo real, então nenhuma delas é considerada uma Realidade Virtual em termos científicos.

Cientistas entraram no mundo jornalístico e artístico na tentativa de definir esse campo. Tende-se a referenciar realidade virtual em termos dos dispositivos usados, e não do seu propósito ou da sua funcionalidade. O público geral normalmente associa a realidade virtual aos dispositivos usados (como *head-mounted displays*) e luvas que capturam movimentação do usuário, visto que estes foram os primeiros dispositivos usados em simulações de realidade virtual.

Realidade virtual pode ser entendida como a simulação de um ambiente real em um sistema computacional. Com isso, pode-se dizer que um dos principais objetivos ao se criar um ambiente de realidade virtual é recriar o maior número de aspectos do mundo real, buscando ser o mais próximo possível da realidade.

2.2 Imersão Virtual

Em um ambiente de realidade virtual, um usuário tem a experiência da imersão, ou a sensação de estar dentro, ou de ser uma parte daquele mundo. Ele também pode interagir com esse ambiente de diversas formas. A combinação da sensação de imersão e a interatividade é chamada de telepresença.

O cientista da computação Jonathan Steuer [32] define a imersão como: "o tanto que uma pessoa se sente presente em um ambiente mediado, em vez do ambiente físico". Em outras palavras, a experiência proporcionada ao usuário de um sistema de VR, faz com que sua existência dentro do ambiente virtual se sobreponha ao ambiente real.

A imersão virtual pode ser dividida em dois componentes principais [32]: profundidade e a amplitude da informação.

Profundidade da informação se refere à quantidade e qualidade dos dados nos sinais que o usuário recebe ao interagir com um ambiente virtual. Para o usuário, isso pode significar a resolução do *display*, a complexidade dos gráficos do ambiente, a sofisticação do sistema de áudio, entre outras [32].

Amplitude da informação pode ser entendida como sendo o número de dimensões sensoriais que são estimuladas ao mesmo tempo. Uma experiência em ambiente virtual possui uma alta amplitude de informações se estimular todos os sentidos do usuário [2].

A maioria das experiências em ambientes virtuais priorizam os componentes de visão e a audição. Porém, vários estudos vêm buscando formas de se incorporar o sentido do toque ao usuário, por exemplo. São sistemas que aplicam uma força de *feedback* quando ocorre o contato com algum objeto no ambiente simulado.

Para que uma imersão seja efetiva, é necessário que o usuário possa explorar o que aparenta ser um ambiente virtual de tamanho real e ainda mudar sua perspectiva continuamente, ao se movimentar pelo ambiente.

2.3 Sistema de Navegação Inercial

Um Sistema de Navegação Inercial ou INS (do inglês, *Inertial Navigation System*) é um dispositivo que calcula constantemente a posição, orientação e velocidade de um objeto em movimentação sem a necessidade de referências externas. Essas medições são feitas por meio de algoritmos que calculam esses atributos, de acordo com os dados medidos, sem que seja necessário a utilização de dados de GPS ou outro tipo de dispositivo externo [36].

Um INS utiliza-se das informações coletadas por uma Unidade de Medição Inercial ou IMU (do inglês, *Inertial Measurement Unit*) para determinar a posição e a movimentação de um equipamento a partir de sua posição inicial [5]. A IMU que mede os dados de aceleração e de rotação do objeto e, quando tratados, cria-se um Sistema de Navegação Inercial.

Qualquer sistema de navegação inercial sofre com o problema do *drift*[1], onde pequenos erros na medição da aceleração e da direção geram grandes erros ao decorrer das medições. Isso se deve ao fato de que o INS utiliza os sinais medidos anteriormente para calcular a atual configuração, o que faz com que esses erros se propaguem e interfiram nos dados que se seguem.

[1] Propagação de erro na medição

Para evitar problemas como o *drift*, além de eliminar ruídos nas medições, técnicas para filtragem desses sinais foram criadas. Mais adiante, alguns dos principais métodos de otimização dos sinais colhidos serão apresentados. Métodos para fusão de sensores, por exemplo, utilizam dados do sistema que estão monitorando para aprimorar a precisão e evitar que erros de medição afetem os valores observados pela IMU.

2.3.1 Unidade de Medição Inercial

A IMU é uma unidade de detecção de mudanças na aceleração e na rotação aplicadas ao objeto observado. São utilizados um ou mais acelerômetros para medir o grau de variação da aceleração, e um ou mais girômetros para detectar as mudanças na rotação.

Os acelerômetros são posicionados a fim de que seus eixos de medição fiquem ortogonais uns aos outros e, com isso, medir a Força-G[2] aplicada ao corpo, como é mostrado na figura 2.1. Os girômetros também são dispostos ortogonalmente para medir a variação da rotação nas três dimensões do eixo cartesiano, como na figura 2.2.

A IMU é uma das partes principais de um INS, visto que é essa unidade que faz as medições das variações de aceleração e rotação sofridas pelo veículo que está utilizando um INS para navegação. Uma IMU pode ser composta por diferentes tipos de acelerômetros e girômetros, incluindo a possibilidade de utilizar sensores que medem mais de uma dimensão, diminuindo a quantidade de sensores dentro da IMU e reduzindo o seu custo.

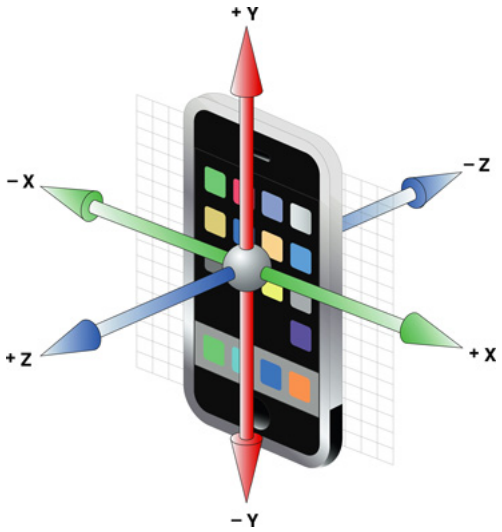

Figura 2.1: Eixos medidos por um acelerômetro. [19]

Os principais sensores utilizados em uma IMU são:

Girômetro

Girômetro é um sensor que mede mudanças na rotação em relação a um valor inicial. Essas mudanças se referenciam aos atributos *pitch*, *roll* e *yaw*[3], e dão a noção de qual

[2] A Força-G é a aceleração aplicada a um objeto e que não é produzida pela gravidade.

[3] Esses são os três parâmetros críticos, principalmente usados na aviação, que dizem respeito aos ângulos de rotação das três dimensões do centro de massa do sensor.

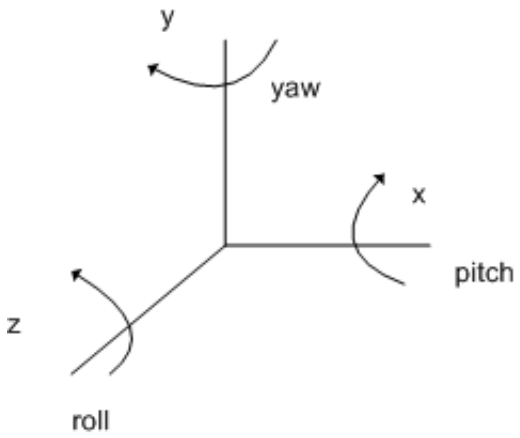

Figura 2.2: Representação das variações *pitch*, *roll* e *yaw*. [24]

direção o objeto observado está sendo rotacionado. Esses atributos dizem respeito a variação nos três eixos cartesianos.

A figura 2.3 apresenta um tipo de giroscópio que utiliza um disco com eixo no centro para medir as mudanças da velocidade angular de um objeto.

Figura 2.3: Representação de um giroscópio. [6]

Neste projeto foi utilizado um girômetro eletrônico, conhecido como girômetro MEMS (do inglês, *Micro Electro-Mechanical System*). Esse tipo de sensor utiliza o conceito da força inercial de Coriolis. Uma explicação mais detalhada pode ser encontrada em [13].

Acelerômetro

O acelerômetro é um dispositivo que mede a variação das forças aplicadas a um objeto. O sensor captura apenas mudanças a partir de seu estado de repouso, logo, a força da

gravidade é desconsiderada nos cálculos, como se estivesse em queda livre [17]. Esses sensores são muitas vezes utilizados para medir tanto a direção quanto a velocidade de determinado objeto.

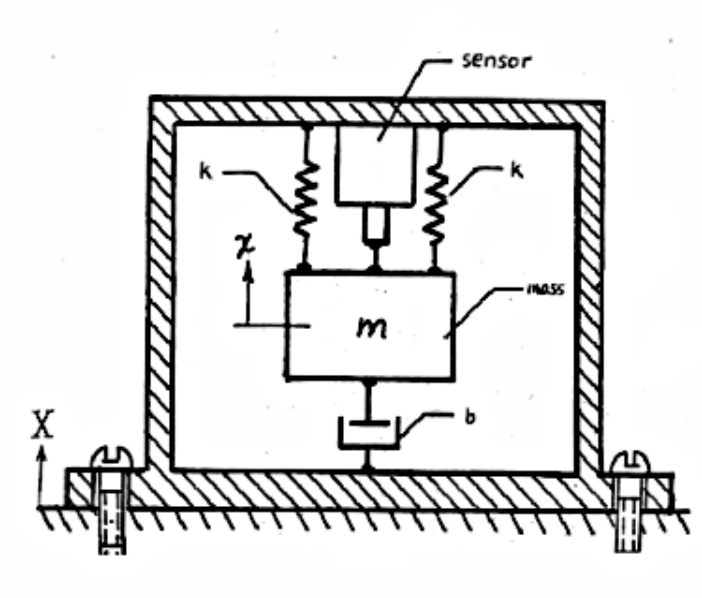

Figura 2.4: Representação de um acelerômetro. [30]

A figura 2.4 representa uma das muitas formas de acelerômetro. Esse acelerômetro é composto por uma massa de prova, que é movimentada à medida que o corpo sofre qualquer tipo de aceleração, e por molas que mantêm essa massa suspensa. É relativa a essa massa de prova, que é calculada a intensidade e a direção da aceleração sofrida pelo dispositivo.

2.3.2 Aplicações

Aeronaves e navios, por exemplo, fazem grande uso de INS's para se posicionarem e saberem a velocidade ou a direção em que estão se movimentando. Normalmente, não são utilizados isoladamente, visto que erros na medida podem se propagar e gerar grandes diferenças nas medidas seguintes. Com isso, esses sistemas normalmente trabalham juntos com GPS's e outros dispositivos, que tornam as medidas muito mais precisas.

Além desses grandes grupos de aplicações, todo sistema que precisa de uma orientação, ou que precisa saber como está se movimentando, utiliza um INS para esse fim. Com o barateamento dos componentes, esses dispositivos já se encontram, inclusive, em aparelhos como *headset*[4], que buscam medir as movimentações da cabeça do usuário.

2.4 Filtro Passa-baixas

Por definição, um filtro passa-baixas deixa passar os sinais que tenham frequência abaixo de um limiar, e dificulta a passagem dos sinais que ultrapassem essa frequência de

[4]Dispositivo multimídia utilizado na cabeça. Podem ser compostos por fones, microfones, *displays* de vídeos, entre outros.

corte. Os sinais de alta frequência tem sua amplitude reduzida para que não influenciem tanto nos valores medidos [21].

Há várias aplicações de filtros passa-baixas. São usados, por exemplo, em circuitos eletrônicos (para áudio), na conversão de sinais analógicos para digital, entre outras. Esse tipo de filtro provê uma forma mais suave de sinal, removendo as rápidas flutuações e deixando as tendências lentas predominarem. A figura 2.5 mostra um sinal com alta frequência, e a figura 2.6 apresenta o sinal após a utilização de um filtro passa-baixas.

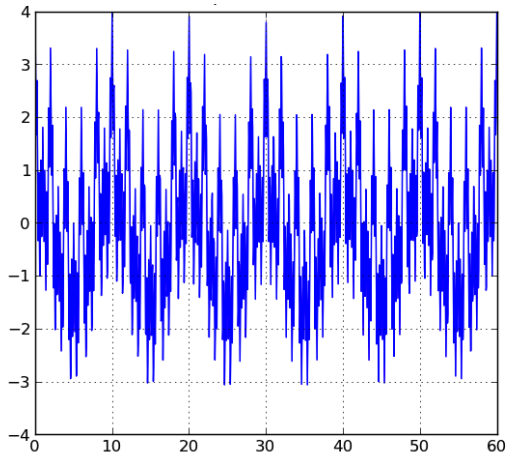

Figura 2.5: Sinal com alta frequência.

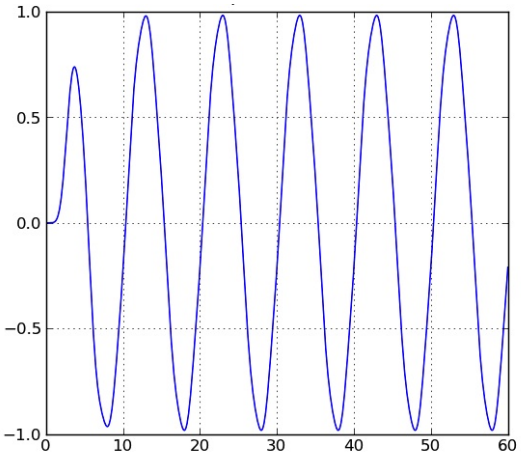

Figura 2.6: Sinal após a aplicação de um filtro passa-baixas.

Existem alguns tipos de básicos de filtros digitais, como por exemplo os filtros FIR (do inglês, *Finite Impulse Response*) e os filtros IIR (do inglês, *Infinite impulse response*).

2.4.1 Filtro Ideal

Filtros ideais eliminam complemamente todas as frequências acima da frequência de corte, enquanto permitem que as frequências abaixo permanecem inalteradas. A região de transição nos filtros práticos não existe [29].

Esse tipo de filtro pode ser criado matematicamente, mas não é possível implementá-lo na prática. A função de sincronização estende-se ao infinito. Para que um filtro se tornasse ideal, seria necessário que este prevesse o futuro e que possuísse um conhecimento infinito do passado, para realiar a convolução [29]. Seria possível apenas em sinais previamente gravados ou perfeitamente cíclicos, visto que dessa forma o filtro teria como prever os dados futuros e teria conhecimento de todos os dados anteriores.

2.4.2 Filtro FIR

São filtros onde a sua resposta ao impulso tem duração finita, pois ele tende a zero em tempo finito. Um filtro de ordem N, discretizado no tempo, utiliza as $N+1$ amostras anteriores para calcular o valor da próxima medida.

Para um filtro FIR discretizado no tempo, a sua saída é uma soma ponderada do valor atual e as N amostras anteriores de entrada. Pode-se observar a operação que é feita através da equação a seguir:

$$y[n] = b_0x[n] + b_1x[n-1] + \cdots + b_Nx[n-N] = \sum_{i=0}^{N} b_ix[n-i] \tag{2.1}$$

Onde:

- $x[n]$ é o sinal de entrada;
- $y[n]$ é o sinal de saída;
- b_i são os coeficientes do filtro;
- N é a ordem do filtro.

Algumas propriedades desse tipo de filtro o torna muito útil em muitos casos.

- Não utilizam realimentação (ou do inglês *feedback*), logo sua implementação é mais simples;
- São inerentemente estáveis. Isso se deve ao fato de não ser necessário a realimentação;
- Podem ser facilmente projetados para ter fase linear [5], apenas modificando os coeficientes utilizados pelo filtro.

2.4.3 Filtro IIR

Ao contrário dos filtros FIR, a resposta ao impulso dos filtros IIR tende a zero em um tempo infinito. Para sinais em tempo discreto, a função de transferência é derivada. A maioria desses filtros são descritos e implementados em termos de uma equação de diferenças, que define como o sinal de saída está relacionado com o de entrada:

$$y[n] = \frac{1}{a_0}(b_0x[n] + b_1x[n-1] + \cdots + b_Px[x-P] - a_1y[n-1] - a_2y[n-2] - \cdots - a_Qy[n-Q]) \tag{2.2}$$

Onde:

- P é a ordem da alimentação do filtro (*input*);
- b_i são os coeficientes da alimentação do filtro;
- Q é a ordem da realimentação do filtro (*output*);
- a_i são os coeficientes da realimentação do filtro;
- x[n] é o sinal de entrada;
- y[n] é o sinal de saída.

[5] De forma simples, a forma do sinal se mantém a mesma após a filtragem. Logo, é possível indentificar o sinal mesmo após a execução do filtro.

Como nesse caso é necessário a realimentação, sua implementação às vezes se torna um pouco mais complexa do que um filtro FIR, podendo se tornar instável. A vantagem desse tipo de filtro é a possibilidade de implementar filtros com alta taxa de rejeição [6].

2.5 Fusão de Sensores

Há muita confusão nas termologias usadas nos sistemas de fusão. Termos como, "fusão de sensores" (*sensor fusion*), "fusão de dados" (*data fusion*), "fusão de informação" (*information fusion*), "fusão de dados de multisensores" (*multi-sensor data fusion*), entre outros, são vastamente utilizados para denominar as técnicas e tecnologias relacionadas aos sistemas de fusão.

Wald [34] propôs o termo "fusão de dados" (*data fusion*), para ser usado como termo principal dos sistemas de fusão. Porém, embora esse termo seja de fácil compreensão, o significado exato dele pode variar de autor para autor.

Há vários livros clássicos de fusão como "*Multisensor Data Fusion*" de Waltz e Llinas [35], que sugerem uma extensão no termo, "fusão de dados de multisensores" (*multi-sensor data fusion*). Porém, mesmo nesses livros, o termo "fusão de sensores" (*data fusion*) é mencionado como sendo equivalente ao termo que estão propondo.

Para evitar confusões no significado, os estudiosos decidiram utilizar o termo "fusão de informação" (*information fusion*) para tratar da fusão de qualquer tipo de dado [8].

Definindo um subconjunto da "fusão de informação", temos o termo "fusão de sensores" (*sensor fusion*), que é introduzido como sendo:

A fusão de sensores diz respeito a fusão de dados derivados de sensores que geram um resultado, de algum modo, mais significante que os dados brutos, quando coletados individualmente. Essa significância que foi dita anteriormente, representa a melhoria na precisão, na robustez, na completude, e outros fatos que fazem os dados serem mais úteis do que os que foram coletados isoladamente [18].

Quando se mede uma variável em particular, um único tipo de sensor pode não fornecer todos os requisitos de desempenho necessários para se utilizar essa variável. Normalmente, combinam-se vários sensores no sistema de medida, e utiliza-se as melhores qualidades de cada dispositivo para se obter o valor da variável em questão.

Tem-se que, os sensores complementam uns aos outros e por isso são chamados de *Complementary Filters*, ou Filtros Complementares. A Fusão de Sensores é apenas a denominação mais nova que vem sendo utilizada para integrar sensores distintos, e obter informações mais precisas e confiáveis. A implementação mais conhecida, o Filtro de Kalman [20], é que vem sendo comumente utilizada em INS's.

Com a combinação de dados de sensores diferentes, consegue-se também mais informações do ambiente que estamos lidando. É o caso de se poder monitorar aspectos mais amplos de um sistema, ou aspectos diferentes de uma área geográfica qualquer, por exemplo.

Quando se refere a robustez, há várias formas de aumentá-la. Pode-se utilizar vários sensores iguais, e aplicar técnicas de estatística para aprimorar os dados obtidos, ou ainda

[6] Taxa de dados filtrados corretamente com uma janela menor de amostras necessárias, onde sinais acima da frequência de corte são atenuados e sinais com frequência menor que o Fc são liberados.

evitar falhas em sensores, e substituir um dispositivo defeituoso por outro em perfeito estado.

A idéia de utilizar a fusão de sensores, se deve ao fato de ter-se a necessidade de combinar a saída do acelerômetro com as leituras feitas pelo girômetro, e assim obter uma boa estimativa da orientação do objeto. Com isso, conseguimos compensar o *drift* gerado pelas leituras do girômetro e a baixa dinâmica dos dados do acelerômetro.

As fontes dos dados para um processo de fusão não precisam ser originadas de dispositivos idênticos. É possível distinguir entre fusão direta (*direct fusion*) e fusão indireta (*indirect fusion*) [23]. Fusão direta, é a fusão dos dados de um conjunto de sensores homogêneos ou heterogêneos, sensores sensíveis e valores históricos de dados sensoriais. Já a fusão de sensores indireta, utiliza a informação dessas fontes como conhecimento sobre o ambiente ou a entrada do sistema. Com isso, tem-se que o termo, "fusão de sensores", descreve uma fusão direta, e o termo "fusão de informação", também se refere a uma fusão indireta.

2.5.1 Motivação

Os sistemas que utilizam métodos de fusão de sensores, esperam obter maiores benefícios com relação aos sistemas que utilizam sensores únicos. Os sensores geralmente sofrem de alguns problemas, como os que serão listados a seguir:

- **Privação de sensor**: Quando um sensor falha, resulta em perdas na percepção do objeto desejado.
- **Espaço de cobertura limitado**: Com apenas um sensor, a região observada pode se tornar bastante restrita. Como é o caso de se utilizar um sensor de luminosidade, que apenas estima a intensidade da luz perto do sensor, e não no ambiente como um todo.
- **Tempo de cobertura limitado**: Alguns sensores precisam de um tempo específico para executar a leitura e transmitir os dados observados. Com isso, a frequência de amostragem acaba sendo limitada a um valor máximo relativo ao tempo de *set-up* do dispositivo.
- **Imprecisão**: As medidas de sensores individuais são limitadas à precisão do elemento empregado no dispositivo.
- **Incerteza**: A incerteza, ao contrário da Imprecisão, depende do objeto que está sendo observado, ao invés de depender do dispositivo. A incerteza aumenta quando o sensor não consegue, por exemplo, medir todos os atributos relevantes da sua percepção, ou quando a observação é ambígua. Um sistema, de apenas um sensor, não consegue diminuir sua incerteza com relação a sua percepção, devido à sua limitada visão do objeto [14].

Pode-se tomar como exemplo, um sensor de estacionamento instalado na parte traseira de um carro. Esse sensor tem apenas a "visão" da parte de trás do carro, o que significa que seu espaço de cobertura é limitado. E, se o sensor fizer suas medidas a cada segundo, temos um limite temporal. Pode acontecer do sensor medir de forma errada o tempo que o ultrassom levou para voltar, ou então as ondas não estão direcionadas corretamente.

Uma solução para esses problemas listados, é a fusão de sensores. É possível compensar os erros que são gerados quando um sensor falha. Para isso, é necesseario construir uma unidade de tolerância a falhas. Há várias formas de se construir uma unidade desse tipo. A redundância de sensores, por exemplo, pode evitar que o sistema deixe de receber os dados da área coberta pelos sensores. Utiliza-se um conjunto de sensores que podem ser do mesmo tipo (redundância) ou de tipos distintos que se complementam, e geram uma medida mais precisa do que se estivessem sendo usados isoladamente.

A utilização de métodos para a fusão de sensores, tanto homogêneos quanto heterogêneos, pode fornecer as seguintes vantagens ao sistema de medida [28]:

- **Robustez e confiabilidade**: A redundância dos dados obtidos quando se utiliza multisensores, faz com que o sistema forneça informação mesmo quando há falha em algum de seus sensores.
- **Espaço de cobertura expandido**: Um sensor pode observar coisas que o outro não pode e, com isso, pode medir dados que os outros sensores não conseguiriam.
- **Aumento da confiança**: A medida de um sensor é confirmada com a medida de outros sensores que estão cobrindo a mesma área.
- **Ambiguidade e incerteza reduzida**: Informações conjuntas reduzem o conjunto de interpretações ambíguas do valor medido.
- **Robustez contra interferências**: Aumentando a dimensionalidade do espaço observado (como é o caso de se utilizar um acelerômetro para medir a movimentação e um girômetro para medir a direção), faz com que o sistema se torne menos suscetível às interferências.
- **Melhoria na resolução**: Quando várias medidas independentes de uma mesma grandeza são fundidas, a resolução do resultado obtido é melhor do que a medida de um único sensor isoladamente. Ela se torna mais precisa e confiável.

Uma outra vantagem de fundir dados de sensores, é a possibilidade de reduzir a complexidade dos sistemas. Em sistemas tradicionais, os sensores alimentam a aplicação com os dados brutos sem nenhum tratamento. Essas aplicações têm que lidar com inúmeras imprecisões, ambiguidades e dados incompletos. Já em sistemas que utilizam dados pré-processados por métodos de fusão, a entrada pode ser generalizada e com isso independe de quais sensores estão sendo utilizados, facilitando a implementação da aplicação e ainda possibilitando modificações no sistema de sensores utilizado, como o número e o tipo desses sensores, sem que seja necessário modificar a aplicação [12].

2.5.2 Tipos

A fusão de sensores pode ser categorizada de três formas principais.

Categorização baseada no modelo de três níveis

O processo de fusão pode ser categorizado em três níveis distintos:

- **Fusão de baixo nível** (ou *fusão de dados brutos*): Combina várias fontes de dados brutos para produzir um novo valor, que se espera ser mais informativo que as entradas.
- **Fusão de nível intermediário** (ou *fusão a nível de recursos*): Combina várias características como bordas, cantos, linhas, etc. em um mapa de recursos, que pode, então, ser utilizado para segmentação e detecção.
- **Fusão de alto nível** (ou *fusão de decisão*): Combina decisões de vários sistemas especialistas. Votação, lógica fuzzy e métodos estatísticos são utilizados nas técnicas de fusão de decisão.

Categorização baseada na entrada/saída

Dasarathy [7] propôs uma refinação na categorização baseada no modelo de três níveis. Ele categoriza o processo de fusão abstraindo o nível das entradas e saídas dos processos.

A necessidade de se criar esse tipo de categorização, se deve ao fato de existirem paradigmas de fusão de sensores, em que as entradas ou as saídas do processo de fusão pertençam a diferentes níveis. Um exemplo é a seleção e extração de características, onde, uma vez que os dados brutos de entrada são processados, os resultados acabam por pertencer ao nível de recurso. Ou seja, não são utilizados os dados brutos, como vieram do sensor. Mesmo utilizando-os para obtenção de caracteríscas específicas, como bordas e cantos em uma imagem, esses dados já foram processados anteriormente, evitando os vários problemas vistos que os sensores apresentam, como visto anteriormente.

Para evitar esse tipo de problema, Dasarathy extendeu a visão de três níveis para cinco categorias de fusão, que são definidas pelas suas entradas/saídas. A figura 2.7 mostra a relação entre o modelo de três níveis e a categorização proposta por Dasarathy.

Categorização Baseada na Configuração dos Sensores

Redes de sensores podem também ser categorizados por sua configuração. Durrant-Whyte [9] distingue três tipos de configuração de sensores:

- **Complementar**: Uma configuração de sensores é dita complementar, se os sensores não dependem uns dos outros diretamente, mas podem ser combinados para obter uma melhor visão do fenômeno em observação. Isso resolve o problema com dados incompletos de sensores. Geralmente, fundir dados complementares é simples, desde que esses dados dos sensores possam ser unidos uns aos outros. Um exemplo desse tipo de configuração é a IMU, que utiliza um girômetro para capturar o grau de rotação (em torno do eixo da força gravitacional) e um acelerômetro para medir a aceleração sofrida por um objeto. Esses dados então são fundidos, e é possível saber a orientação da movimentação do objeto em análise.

  Na figura 2.8 os sensores S_2 e S_3 podem ser vistos como uma representação de uma configuração complementar, visto que estão observando partes distintas do ambiente.

- **Competitivo**: Uma configuração de sensores é dita competitiva se cada sensor entrega medidas independentes de uma mesma propriedade. Há duas formas de se

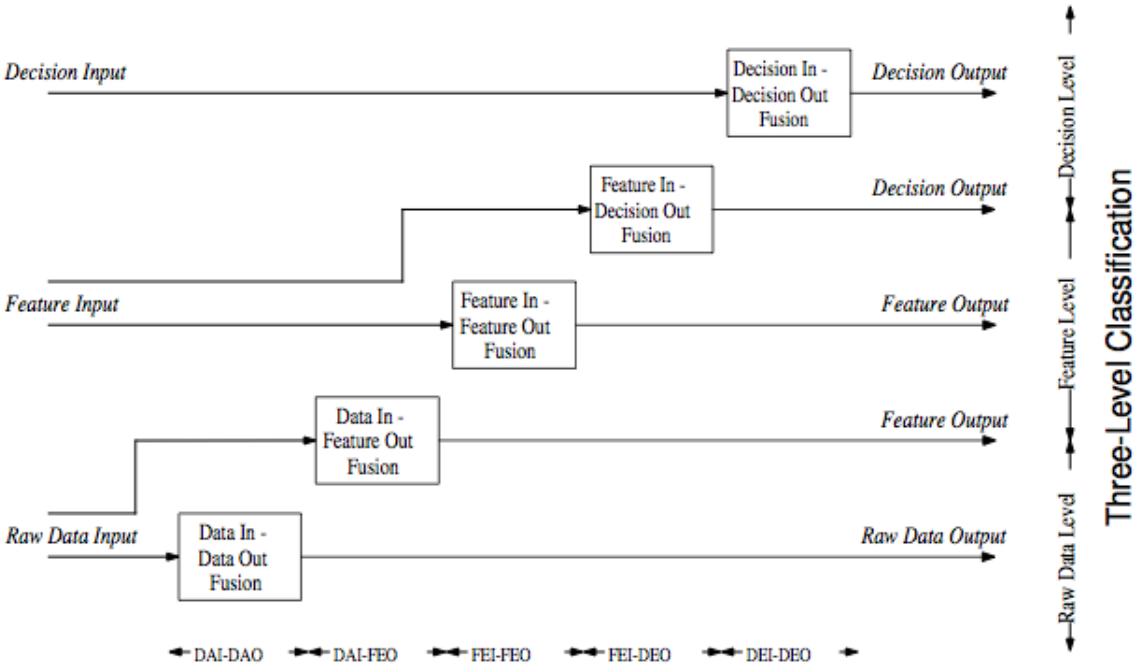

Figura 2.7: Categorização baseada na entrada/saída [7]

realizar essa configuração, sendo uma delas fundindo os dados obtidos por diferentes sensores, ou então fundindo as medidas de um mesmo sensor que foram capturadas em diferentes momentos. Esse tipo de configuração também é conhecido como uma configuração *redundante*.

Um caso especial da configuração competitiva são os sistemas de *Tolerância a falhas*, que necessitam de uma especificação exata do serviço e dos modos de falha do sistema. Em caso de falha, o sistema deve continuar a fornecer o serviço que se propôs a oferecer. Além disso, é possível aumentar a robustez do sistema ao se utilizar essa configuração. Um exemplo da utilização dessa configuração, seria reduzir ruídos combinando duas imagens sobrepostas.

Os sensores S_1 e S_2 da figura 2.8 representam uma configuração competitiva, onde ambos os sensores observam redundantemente a mesma propriedade de um objeto no espaço observado.

- **Cooperativo**: Uma rede de sensores é dita cooperativa, se utilizar a informação de cada sensor para derivar uma nova informação, que não seria acessível a um único sensor isoladamente. Um exemplo desse tipo de configuração, é a visão estereoscópica, onde se combinam imagens bidimensionais de duas câmeras com pontos de vista ligeiramente diferentes, em uma imagem tridimensional da cena observada.

  Esse tipo de configuração é o mais difícil de ser projetado, visto que o resultado é bastante sensível a imprecisões em cada um dos sensores que participam da medida. Em contraste com a fusão competitiva, a fusão de sensores cooperativos geralmente reduz a precisão e a confiabilidade dos resultados.

Os sensores S_4 e S_5 da figura 2.8 representam uma configuração cooperativa. Ambos sensores observam o mesmo objeto, mas a suas medidas serão combinadas e formarão uma visão diferente do objecto. Essa visão não poderia ser gerada utilizando os sensores isoladamente.

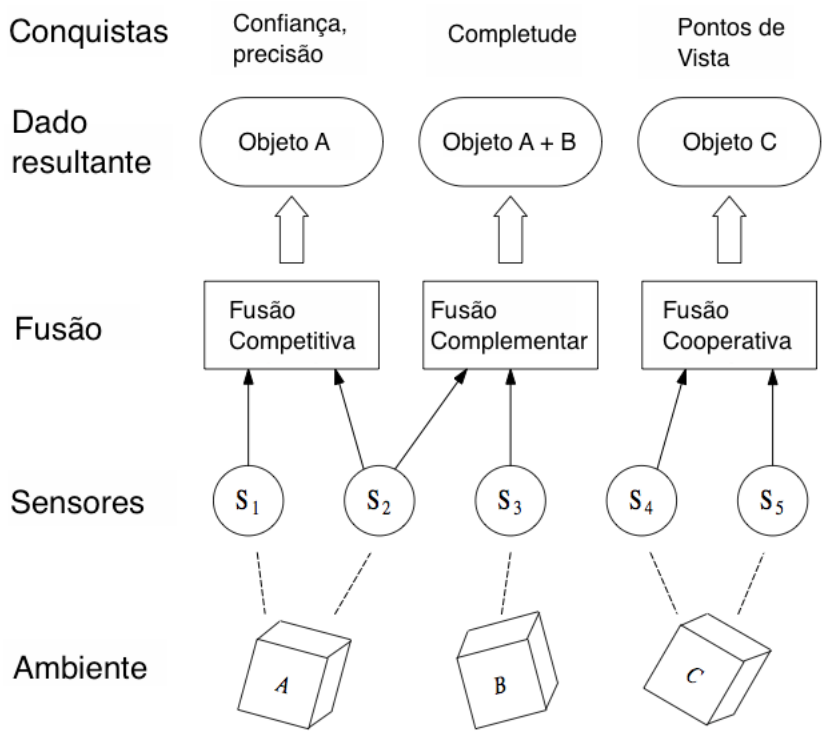

Figura 2.8: Configuração de sensores: Complementar, competitiva e cooperativa.

Essas três categorizações não são mutuamente exclusivas. Várias aplicações utilizam aspectos de mais de uma dessas configurações. Um bom exemplo disso, são sistemas de monitoramente por câmera, onde exista regiões cobertas por duas ou mais câmeras, os sensores podem estar configurados como competitivo ou cooperativo. Agora, em regiões em que apenas uma câmera consegue observar, o sensor fica configurado como complementar.

2.5.3 Algorítmos

Neste trabalho, uma configuração complementar será utilizada para a rede de sensores. A IMU é composta por três sensores (girômetro, acelerômetro e magnetômetro) que observam aspectos diferentes do ambiente e, com isso, é possível unir suas informações e criar uma nova, mais robusta e confiável.

A idéia principal de um filtro que utiliza a configuração complementar (do inglês, *angle complementary*) para a fusão dos dados é obter uma boa estimativa para a orientação do objeto observado. A fusão desses dados, compensa a deficiência do girômetro de propagação de erros nas medições, e ainda a deficiência do acelerômetro, de não ser muito preciso [2], por exemplo.

Uma das principais implementações utilizadas é o Filtro de Kalman. Porém, neste projeto será utilizada uma outra implementação, conhecida como *Direction Cosine Matrix* ou DCM.

Um dos principais algoritmos de fusão de sensores, utilizando a configuração complementar, é o filtro de Kalman [20]. Porém, nesse trabalho será abordado a filtragem do tipo DCM (do inglês, *Direction Cosine Matrix*).

Filtro DCM

O método de filtragem DCM consiste na utilização de uma matriz dos cossenos da direção, onde os sensores (como o girômetro, magnetômetro e acelerômetro) alteram os elementos dessa matriz. Com os dados da matriz, é possível determinar a direção de uma movimentação.

Visto como um algoritmo, o DCM funciona da seguinte forma [27]:

- O girômetro é usado como fonte primária para a informação da orientação;
  - Os dados obtidos informa a variação da direção sofrida pelo objeto.
- Reconhecendo que os erros numéricos vão gradualmente violando a restrição de ortogonalidade que o DCM deve satisfazer, pequenos e regulares ajustes são feitos nos elementos da matriz, com a finalidade de manter a ortogonalidade dos eixos;
- Os erros numéricos, o *drift* do girômetro e o seu *offset* são gradualmente acumulados nos elementos da matriz DCM. Um vetor de referência é utilizado para detectar os erros, e então é calculado o erro para controlar o *feedback* negativo entre os erros detectados e as entradas do girômetro usados no primeiro passo, eliminando os erros, antes que eles se acumulem.

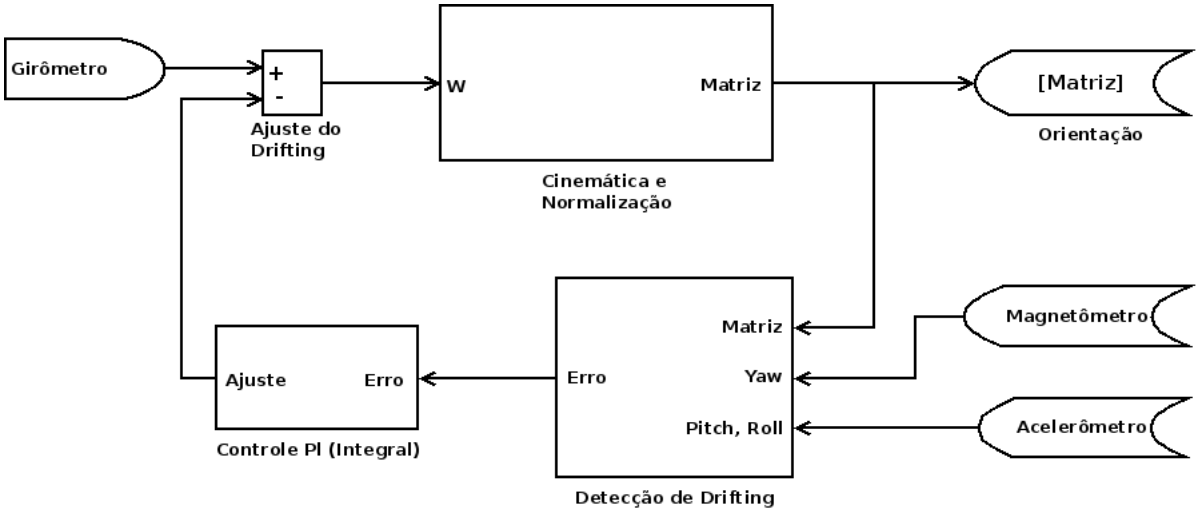

Figura 2.9: Esquema de funcionamento do filtro DCM. [27]

- Acelerômetro é usado para detectar os erros nos eixos *pitch* e *roll*, já o magnetômetro detecta erros no eixo *yaw*.

O esquema do processo pode ser observado na figura 2.9.

É possível perceber o ciclo de funcionamento da fusão dos dados utilizando o método de DCM. O girômetro captura a orientação inicial e os outros sensores são usados para corrigir e melhorar os dados iniciais. A matriz é utilizada para manter a referência da posição anterior. Com os erros calculados, a correção é feita e aplicada aos dados adquiridos pelo girômetro.

Capítulo 3

Metodologia Proposta

A proposta para solução do problema de integrar um dispositivo de interação humano-computador ao Google Sketchup, envolveu uma abordagem modular, visto que, a ideia era propor uma forma fácil de integrar o dispositivo com diversos outros sistemas computacionais.

Com a ideia de separar o sistema em partes que pudessem ser substituídas, dividiu-se o projeto em quatro módulos: de aquisição, de processamento, de comunicação e de apresentação, conforme apresentado na figura 3.1.

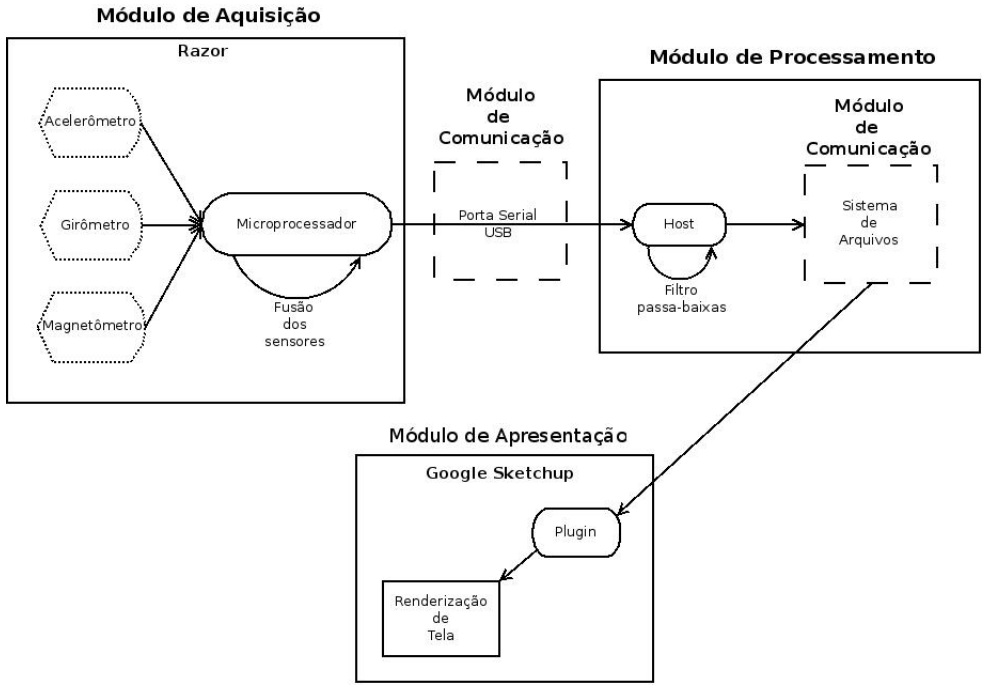

Figura 3.1: Sistema baseado em módulos proposto.

O módulo de aquisição se comunica com os três sensores utilizados: girômetro, acelerômetro e magnetômetro. Ao adquirir os dados, um pré-processamento é feito para fundir as medidas em apenas três valores: *yaw*, *pitch* e *roll*. O módulo de comunicação, que utiliza uma porta serial, é utilizado para transmitir os dados para o computador. A próxima etapa envolve o módulo de processamento, no qual são aplicados algoritmos de

otimização do sinal recebido (como um filtro passa-baixas frequências), para a melhor interação do dispositivo com o software integrado. Por último, utiliza-se outro módulo de comunicação para enviar os dados para um *plugin* interno ao Sketchup, que utiliza-se dessa informação, para movimentar a visão da câmera no aplicativo.

3.1 Escolha do Hardware

O primeiro passo para construção do protótipo do sistema foi a escolha do hardware a ser utilizado para a aquisição dos dados de orientação espacial.

Inicialmente, foi feito um estudo de caso para descobrir se os componentes encontrados no controle remoto do Wii poderiam ser usados para a captura de movimento da cabeça do usuário. Viu-se que esse controle possuía dois sensores que podem compor uma IMU, o acelerômetro e o girômetro.

Após uma primeira implementação do protótipo, a abordagem principal do projeto (apresentada a seguir) foi decidida e implementada.

3.1.1 Componentes do WiiMotionPlus

A começar pela escolha dos sensores, um controle do vídeo game Wii foi desmontado. Os únicos sensores que o controle continha eram um girômetro e um acelerômetro. Com os sensores escolhidos, algumas formas de comunicação com o computador foram testadas.

Girômetro

O primeiro dos sensores testados se mostrou funcional. Os dados capturados mostravam a direção do movimento e, com isso, foi possível movimentar a câmera no Google Sketchup.

Porém, alguns problemas logo foram observados:

- **Drifting**: assim como em todos os sensores, erros nas medidas podem ocorrer e, como o dado adquirido é a variação angular do objeto, tem-se que o acúmulo desses erros acaba gerando um deslocamento indesejado na imagem projetada. São esses acumulos de erros que geram o conhecido *drift* (deslocamento da imagem);
- **Tempo**: o girômetro informa a variação angular do objeto com relação a medida anterior. Logo, para mover a câmera no Sketchup, foi necessário saber a frequência das medições (como será visto no seção 3.3).

Acelerômetro

Esse sensor informa a intensidade da aceleração que o objeto está sofrendo no momento da medição. No entando, somente com ele, não é possível calcular para qual direção o objeto observado está virando, sabe-se apenas para qual direção ele está se movimentando, de forma escalar.

Portanto, a utilidade do acelerômetro no sistema é utilizar seus dados juntamente com o girômetro, e com isso diminuir o problema de *drifting* que este sofre.

Também pode-se verificar as seguintes dificuldades ao se utilizar o acelerômetro:

- **Baixa dinâmica**: Esse tipo de sensor é mais sensível a movimentações bruscas. Por esse motivo, os dados mensurados não necessariamente informam o que realmente aconteceu com o objeto;
- **Tempo**: Os sensores capturam as informações no momento que é requisitado, logo, é preciso utilizar a informação do período das medições para saber o tempo que essa aceleração está sendo aplicada ao objeto.

Comunicação

Para fazer a comunicação com os sensores, utilizou-se uma plataforma Arduino [37] conectada a cada sensor, enviando dados via porta serial USB para o computador *host*.

O *firmware* foi criado a fim de se comunicar com os sensores retirados do controle do Wii e enviar os dados para o computador *host*. Este foi instalado na Arduino. A cada requisição, as medições de cada um dos dois componentes são capturadas. As grandezas medidas estão no formato *float* (IEEE 754) de precisão simples (quatro bytes).

O protocolo criado para enviar os dados para o *host* foi:

- Inicialmente envia os dados do Girômetro;
  - A ordem de envio é: *Yaw*, *Pitch* e *Roll*;
  - Cada inteiro é enviado byte a byte, sendo o byte mais significativo do número, o primeiro a ser enviado;
- E a seguir, envia os dados do Acelerômetro;
  - A ordem de envio é: x-axis, y-axis e z-axis;
  - Assim como o girômetro, cada número é enviado byte a byte, sendo o byte mais significativo, o primeiro a ser enviado.

Conclusão preliminar

Com a utilização apenas dos dados do Girômetro, as informações de rotação que foram adquiridas não eram suficientes para proporcionar uma boa experiência ao usuário. Seria necessário fundir os dados de mais de um sensor para melhorar a precisão da orientação do objeto observado.

Para melhorar essa experiência, foi necessário a utilização de mais um sensor, deixando o sistema ainda com três graus de liberdade (porém, utilizando mais de um sensor para observar o mesmo atributo) e tornando as medidas muito mais precisas e adequadas para obter a rotação angular do objeto.

Foi na busca desse novo componente, o magnetômetro, que surgiu a oportunidade de se utilizar uma placa que já possuía integrado os três sensores que eram necessários, e ainda utiliza o mesmo microprocessador contido na plataforma Arduino.

Por esse motivo, optou-se por trocar o *hardware*, e passar a usar a nova placa da Razor [10] para obtenção dos dados de posicionamento do objeto e fusão dos sensores. Outra vantagem de se usar a Razor, é que seu *firmware* já possui um filtro de fusão DCM implementado.

3.1.2 Razor IMU

A placa da Razor IMU [10] possui três sensores: um girômetro, um acelerômetro e um magnetômetro. Logo, com sua utilização, é possível obter um sistema que proporcione medidas muito mais precisas e confiáveis. Além disso, seu *firmware* já possui um filtro DCM implementado, o que possibilita a aquisição de dados já fundidos.

Os dados gerados com fusão dos sinais obtidos pelos três sensores é exatamente o que precisa ser enviado ao Google Sketchup para controle da câmera. Os sensores informam a variação do objeto em relação a medição anterior. Após a fusão, esses dados não mais dizem respeito à essa variação. O resultado da fusão é a posição atual do objeto, logo é possível descobrir qual a posição e direção atual a partir de um ponto inicial. Com isso, tem-se que os dados mostram a posição absoluta, e não a diferença com relação a posição anterior.

Firmware

O *firmware* é fornecido no próprio site da Sparkfun [11]. Como o microcontrolador é o mesmo da plataforma Arduino, a placa foi conectada ao *host* através da porta serial USB e então foi feito o *upload* do *firmware*.

A etapa inicial para utilizar a Razor foi fazer testes e descobrir como configurar e usar todas as funcionalidades implementadas pelo *firmware*.

Utilizando a própria IDE da Arduino, foi aberta uma porta serial para comunicação com a placa. Por padrão, a Razor já vem configurada para apresentar os dados em forma de texto, o que pode ser observado na figura 3.2.

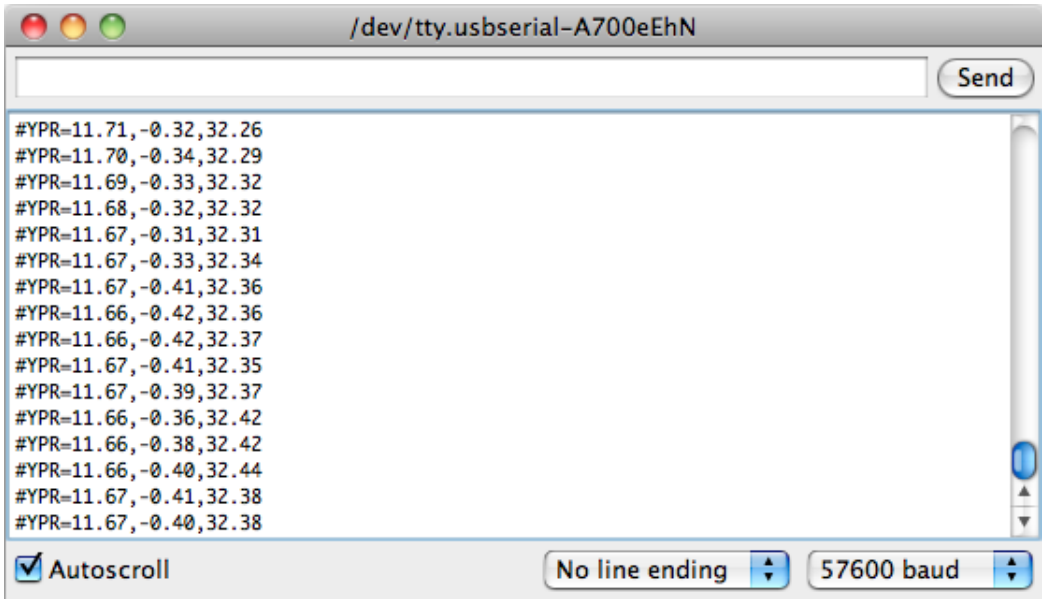

Figura 3.2: Saída padrão da placa Razor.

Após testar o funcionamento do Razor e visto que a placa realmente estava capturando as informações desejadas, foi então feita uma lista com os comandos aceitos pelo *firmware*:

- #**o**<**param**> - Aplica parâmetros de saída;

  - #**o0** - Desabilita o *stream* contínuo de saída;
  - #**o1** - Habilita o *stream* contínuo de saída;
  - #**ob** - Envia os ângulos em formato binário (*yaw*/*pitch*/*roll* são representados como floats);
  - #**ot** - Envia os ângulos em formato texto;
  - #**os** - Quando em modo de calibração, apresenta os dados referentes a cada sensor, em forma texto;
  - #**oc** - Entra em modo de calibração;
  - #**on** - Quando em modo de calibração, vai para o próximo sensor a ser calibrado;
  - #**oe0** - Desabilita a saída de erro;
  - #**oe1** - Habilita a saída de erro;

- #**f** - Faz a requisição de uma janela de dados. Útil quando a saída contínua de dados é desabilitada e atualizações são necessárias apenas em grandes intervalos de tempo.

- #**s**<**xy**> - Faz a requisição de um *token* de sincronização. Útil para verificar em qual momento se encontra o envio dos dados em um *stream* contínuo ou para testar se a placa está respondendo. A placa irá enviar "#SYNCH<xy>\r\n" como *token*.

Com esses testes iniciais, foi verificado que o *firmware* e a placa Razor estavam funcionando como o esperado. O próximo passo foi testar um código fornecido na página da Sparkfun para integração da placa com o *software* Processing [15].

Basicamente o código cria um bloco vermelho com uma seta verde acoplada. Esse bloco é girado de acordo com os dados coletados pela placa. Logo, é feita uma comunicação com a Razor através de uma porta serial e, com os dados de *yaw*, *pitch* e *roll* já fundidos, é feita a rotação do objeto. Com isso, é possível verificar o funcionamento da placa Razor e sua integração com o *Processing*.

O código de teste cria uma janela com o bloco se movimentando juntamente com a placa, como mostrado na figura 3.3.

O Processing mostou a precisão dos dados que estavam sendo capturados, porém percebeu-se a necessidade de fazer a calibragem dos sensores. Isso foi observado pelo fato de que algumas amplitudes de movimentação não estavam sendo consideradas corretamente.

Os sensores podem ter seus valores afetados não só pelo material ou pelo desgaste do sensor, como também pela sua posição geográfica. Alguns dos problemas são:

- **Acelerômetro**: a força gravitacional varia dependendo do local que se utiliza esse sensor;

- **Magnetômetro**: o campo magnético pode ser diferente em determinados lugares do mundo. Além disso, materiais que estejam conduzindo corrente elétrica podem interferir, quando utilizados perto do sensor.

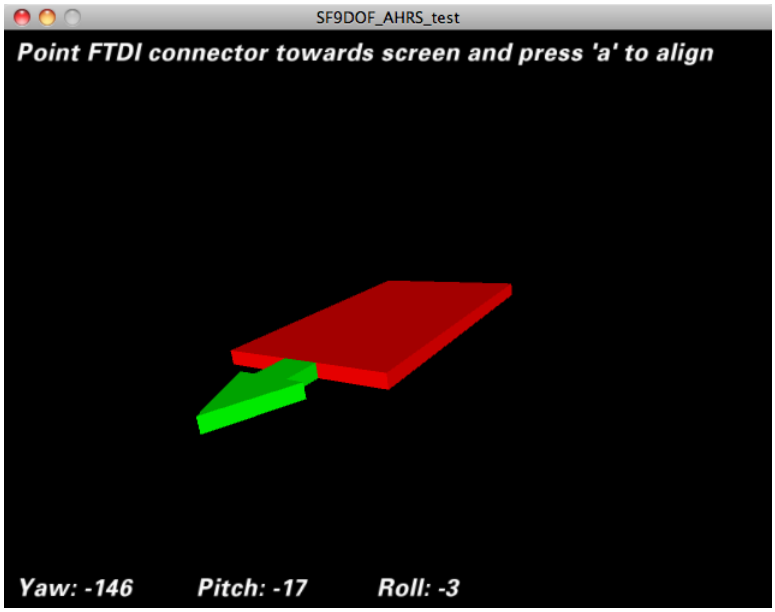

Figura 3.3: Tela do Processing com o bloco orientado de acordo com a posição da placa.

Por causa dessas interferências, utilizou-se o modo de calibragem do *firmware* (#oc) para configurar o dispositivo da melhor forma possível.

No modo de calibragem, é necessário movimentar o dispositivo para que ele chegue no máximo e no mínimo de seus valores para cada um dos eixos dos seus sensores, contemplando os três graus de liberdade que o dispositivo mensura.

Comunicação

O componente é ligado à porta serial USB do computador e este inicia a comunicação com os sensores, informando qual a forma que deseja configurar o *firmware*.

O *firmware*, quando configurado como saída binária, envia cada uma das três medidas (*yaw*, *pitch* e *roll*) em formato float de precisão simples (quatro bytes). O *firmware* da placa trata os bytes no formato *big-endian*[1].

3.2 Processamento

Este é o módulo que interliga o sistema de aquisição de dados com o *plugin* criado para o Google Sketchup (módulo de apresentação). Esse módulo foi todo implementado na linguagem de programação Ruby.

Inicialmente é estabelecida a comunicação com os sensores e então configurada a forma que se deseja obter as informações de posicionamento.

Empiricamente, chegou-se a melhor forma de configurar o *firmware*, visto que alguns problemas foram enfrentados no decorrer do projeto:

- #**ob**: hablita a saída como binária;

[1] O byte mais significativo é enviado por último.

  - Foi necessário converter os números para *little-endian*[2] e então montar os bytes como um float novamente;
  - O processador utilizado no desenvolvimento e teste desse projeto foi: Intel Core I7 - 2,66 GHz;

- #**o0**: desabilita o *stream* contínuo. É necessário solicitar ao dispositivo novas informações;
  - Como os dados são processados e enviados para o módulo de apresentação (no caso o Google Sketchup), o *stream* contínuo faz com que os sensores enviem novos dados antes que o módulo de processamento esteja pronto para recebê-los. Isso acaba gerando uma fila de dados inúteis no buffer da porta serial USB e, com isso, dados antigos são retornados (por causa do buffer), gerando atraso na movimentação da câmera;
- #**oe0**: Desabilita mensagens de erro, que são úteis apenas para debug.

Assim como pode ser visto no código de inicialização:

```
// dsl.rb#initialize
# Params for the serial port
port_str = "/dev/tty.usbserial"
baud_rate = 57600
data_bits = 8
stop_bits = 1
parity = SerialPort::NONE

@sp = SerialPort.new port_str, baud_rate, data_bits, stop_bits, parity

# Set Razor output parameters
@sp.write "#ob"   # Turn on binary output
@sp.write "#o0"   # Turn off continuous streaming output
@sp.write "#oe0"  # Disable error message output

# Synch with Razor
@sp.flush         # Clear input buffer up to here
@sp.write "#s00"  # Request synch token
```

Antes de iniciar a aquisisão dos dados, o *token* de sincronia é requisitado (comando "#s00") e ainda é feito um *flush* na portal serial. Isso evita que os dados sejam capturados fora de sincronia, gerando informações incorretas.

Ao pedir o *token* de sincronia, a porta serial USB é monitorada até que a saída "#SYNCH00\r\n" seja capturada.

Quando o módulo de processamento está sincronizado com o módulo de aquisição, inicia-se o *loop* infinito que: requisita os dados dos sensores, processa e filtra os dados e, por fim, os envia para o módulo de apresentação.

O sistema é completamente parametrizado para que seja possível alterar o que é executado em cada módulo. Nessa implementação, é possível alterar o filtro passa-baixas

[2] O byte mais significativo é recebido primeiro.

utilizado (para testes de qual o melhor filtro a ser utilizado nesse sistema) e ainda o módulo de apresentação (*software* integrado).

O *loop* principal do sistema:

```
// dsl.rb
# Loop forever
while true
  # Wait for the token to synch
  unless @synched
    @synched = readToken "#SYNCH00\r\n"
    next
  end

  low_pass_filter = read_values low_pass_filter

  print_values low_pass_filter, output

  if (@options[:verbose])
    actual_time = Time.now
    period = actual_time - last_time
    puts "Period(ms) = #{format("%.6f", (period) * 1000)}\t\tYaw:
        #{format("%.2f", low_pass_filter.yaw)}\t\tPitch: #{format("%.2f",
        low_pass_filter.pitch)}\t\tRoll: #{format("%.2f", low_pass_filter.roll)}"
    last_time = actual_time
    @max_period = period if period > @max_period
  end
end
```

3.2.1 Requisição dos dados

Escreve-se na porta serial o comando "#f", que requisita ao *firmware* novos dados. Em seguida, doze bytes de resposta são lidos e transformados, quatro a quatro, nos floats *yaw*, *pitch* e *roll*, respectivamente.

3.2.2 Filtragem

Verificou-se que as medições possuiam ruídos de alta frequência, mesmo quando o sensor estava parado. Variações significantes nos valores medidos podiam ser observados.

Alguns filtros digitais foram testados para verificação de qual melhor se comportaria no sistema. Eles são utilizados para eliminar, ou minimizar, os ruídos ou movimentações bruscas coletadas pelos sensores. Ruídos normalmente possuem uma frequência alta, logo utilizou-se filtro passa-baixas para tal finalidade.

Três tipos de filtros digitais foram testados, do mais simples ao mais complexo.

Média móvel

O primeiro a ser testado foi o filtro de média móvel com ordem dez. Esse filtro basicamente guarda as dez medidas anteriores e faz uma média desses valores. Isso faz

com que as mudanças rápidas (frequência alta) sejam minimizadas e as modificações lentas (frequência baixa) predominem.

O código desse filtro foi o mais simples:

```
// moving_average_filter.rb
1.  def moving_average array
2.     average = 0
3.
4.     current_value = array.last
5.     past_value = current_value
6.
7.     array.reverse.each do |value|
8.       diff = current_value - past_value
9.       if (diff < -180)
10.         current_value += 360
11.      elsif (diff > 180)
12.         current_value -= 360
13.      end
14.      average = average + current_value
15.
16.      # walk on the queue
17.      past_value = current_value
18.      current_value = value
19.    end
20.
21.    returned = (average / array.size) % 360
22.    if returned == 180
23.      return 179.99999
24.    elsif returned == -180
25.      return -179.99999
26.    else
27.      returned
28.    end
29. end
```

No início da iteração (linhas 8-13) é possível observar algumas verificações feitas na diferença do valor atual com o anterior.

Isso foi feito pelo fato dos sensores do Razor terem uma amplitude de -179,9 a 179,9. Uma medida de -179,9 e outra de 179,9 gerariam uma média igual a zero, mas esses valores são próximos (menos de um grau de diferença).

Para resolver esse problema, é verificado se a diferença entre os valores é maior que 180 (ou menor de -180). Caso seja, é adicionado (ou retirado) uma volta inteira de 360 graus ao valor a ser filtrado. Com isso, um valor de -179,9 ficaria realmente próximo ao seu valor "vizinho" 179,9.

Por fim, é aplicado um módulo de 360 na média criada pelo filtro, mantendo os dados em apenas uma volta com relação ao eixo inicial.

Entretanto, um filtro de ordem dez causa um atraso considerável. Esse problema faz com que a experiência do usuário não seja muito boa. Por esse motivo, outro tipo de filtro

foi testado.

FIR - Window Gaussian

Em seguida, foi testado um filtro FIR de fase linear projetado utilizando uma janela gaussiana de ordem cinco. Esse filtro faz uma média ponderada, onde cada um dos dados anteriores tem um peso diferente na cálculo do valor filtrado de saída.

```
// window_gaussian_filter.rb
def window_gaussian array
  average = 0

  current_value = array.last
  past_value = current_value

  array.reverse.each_with_index do |value,index|
    i = @filter_order - index - 1

    diff = current_value - past_value
    if (diff < -180)
        current_value += 360
    elsif (diff > 180)
        current_value -= 360
    end
    average = average + (current_value * @coef[i])

    # walk on the queue
    past_value = current_value
    current_value = value
  end

  returned = average % 360
  if returned == 180
    return 179.99999
  elsif returned == -180
    return -179.99999
  else
    returned
  end
end
```

O código implementado é semelhante ao da média móvel, porém pondera-se as amostras utilizando os coeficientes do filtro ao invés de simplesmente fazer a média dos valores.

IIR - Butterworth

Por último, um filtro IIR (do inglês *Infinite Impulse Response*) de ordem quatro do tipo Butterworth foi testado. Esse tipo de filtro guarda as últimas entradas e as últimas

saídas geradas pelo filtro. Com isso, apenas quatro dados anteriores (de entrada e de saída) são necessários para filtrar as amostras e obter os resultados desejados.

Como é possível ver, o código do filtro IIR utiliza dois buffers: um para armazenar as entradas e outro para as saídas.

```
// iir.rb
def iir in_buffer, out_buffer, input
  # Insert new value into Input Array
  in_buffer.unshift input

  # Truncate Input Array to have right number of elements
  in_buffer.pop if (in_buffer.size > @filter_order+1)

  out = 0.0;
  for i in 0..@filter_order
    out += @b[i] * in_buffer[i];
  end
  for i in 0...@filter_order
    out -= @a[i+1] * out_buffer[i];
  end

  # Insert new value into Output Array
  out_buffer.unshift out

  # Truncate Output Array to have right number of elements
  out_buffer.pop if (out_buffer.size > @filter_order)

  out
end
```

Por ter a menor ordem, o filtro IIR Butterworth é o que resulta no menor atraso. No entanto, sua fase não sendo linear pode interferir na usabilidade.

O filtro sendo não linear, o atraso gerado pelo filtro será diferente para frequências distintas. Isso implica que, se o usuário estiver mexendo a cabeça de um lado para o outro, o atraso será diferente para cada velocidade ele escolher, o que pode prejudicar na experiência do usuário.

3.2.3 Comunicação com Sketchup

Uma vez processados e filtrados, os dados são disponibilizados para o módulo de apresentação. Essa comunicação é feita através do sistema de arquivos do sistema operacional.

Um arquivo de *input* é criado na inicialização do módulo de processamento. Esse módulo escreve os ângulos no arquivo em formato texto, separando-os por vírgula, enquanto o *plugin* interno do Sketchup lê os dados do arquivo para movimentar a câmera.

Para que essa comunicação possa ser estabelecida, é necessário implementar uma lógica de *lock*, onde o processo que está utilizando o arquivo o impede de ser acessado por qualquer outro processo até que tenha terminado de utilizá-lo. Isso evita problemas, como

o caso do Sketchup tentar ler os dados ao mesmo tempo que o módulo de processamento os escreve.

3.3 Apresentação - Google Sketchup Plugin

Foi criado um *plugin*, utilizando a API do Google Sketchup, para fazer a integração do módulo de processamento com o módulo de apresentação. Esse *plugin* tem duas funções principais: iniciar a animação com dados dos sensores; executar um *offset* para posicionar a câmera no local correto. Essa API utiliza Ruby como linguagem de programação.

O *plugin* foi feito para ser chamado a todo o momento que se deseja criar o *nextFrame* a ser renderizado na tela. Com isso, a todo ciclo de renderização da tela, uma nova posição dos sensores é lida, fazendo com que a câmera seja corretamente rotacionada.

O *plugin* lê o arquivo de *input* e cada ângulo lido é transformado em radianos, pois a API do Sketchup utiliza o ângulo em radianos para fazer a rotação da câmera. Além disso, a API ainda necessita da posição do "olho" da câmera (posição da lente da câmera) e do eixo que será feita a rotação para executar essa rotação.

Os ângulos capturados e filtrados informam a orientação da cabeça do usuário. Para a criação de uma nova cena, dois vetores são necessários: o "alvo" e o "up", além de um ponto (tridimensional) referente ao "olho" da câmera, como ilustrado na figura 3.4.

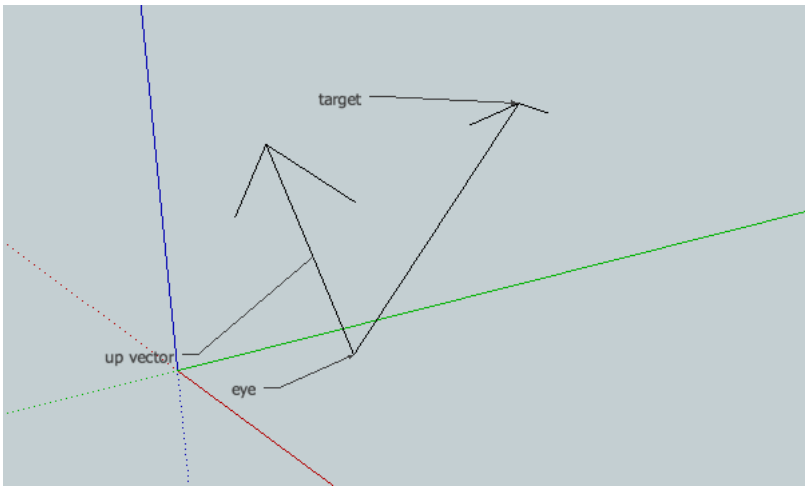

Figura 3.4: Esquema apresentando os vetores "Alvo" e "Up", e o ponto "Olho". [31]

- **Alvo**: Esse vetor informa a direção que a câmera está "apontando". Logo, esse é o principal vetor quando se deseja movimentar a visão que o usuário tem ao utilizar o sistema;
- **Up**: É o vetor que sempre "aponta" para cima. Com isso, é possível saber onde está a parte de cima da câmera mesmo ela estando inclinada;
- **Olho**: Este ponto mostra onde o olho se encontra na imagem. Ou seja, é ele que diz a que altura o olho está, por exemplo.

A API do Sketchup impede a execução de todas as rotações ao mesmo tempo. Logo, como cada um dos dados (*yaw*, *pitch* e *roll*) dizem respeito à rotação em cada eixo cartesiano, as rotações são aplicadas individualmente e, somente após as três movimentações serem executadas é que o "*nextFrame*" é renderizado para o usuário.

- **Yaw**: Diz respeito ao ângulo de rotação horizontal. Para isso é feita a rotação no eixo Y;
- **Pitch**: Informa o ângulo de rotação vertical. A rotação então é aplicada no eixo X;
- **Roll**: Ângulo referente ao giro do sensor. Move-se o alvo da câmera pelo eixo Z.

O *offset* implementado serve para reposicionar a câmera no centro da imagem, e fazer com que os próximos valores lidos a rotacionem a partir dessa posição marcada como inicial.

O código do *plugin* do Sketchup pode ser visto no apêndice A.

O protótipo foi desenvolvido e o modelo proposto mostrou-se funcional. Sem o uso da fusão dos dados dos sensores, a experiência não era boa, visto que estava muito sensível e, após um pequeno tempo de utilização, o erros já começavam a aparecer (como é o caso do *drift*).

Com a fusão dos dados coletados, o módulo de aquisição ficou completo, retornando dados robustos e confiáveis.

O módulo de Processamento é totalmente parametrizado, logo, é possível adaptá-lo facilmente. Isso foi feito ao se testar os vários tipos de filtro passa-baixas. Esses filtros tornaram a experiência do usuário muito melhor, pois são eles que impedem que ruídos e pequenas movimentações deixem a tela se movimentando incessantemente.

O módulo de apresentação foi implementado utilizando a API do Google Sketchup. Então, foi possível integrar perfeitamente com a ferramenta desejada. Não foi necessária nenhuma adaptação dos dados para que funcionasse a integração.

O Sketchup consiste de um cenário virutal utilizado para criação e visualização de projetos arquitetônicos. O *plugin* foi todo construído para lidar com esse ambiente virtual, que pode ser modificado pelo usuário ao se construir o projeto de arquitetura.

Capítulo 4

Resultados Obtidos

Neste capítulo são apresentados os resultados obtidos e as análises de desempenho do protótipo.

4.1 Aquisição dos Sinais

A figura 4.1 mostra a placa Razor utilizada para a aquisição dos dados. O *firmware* instalado é configurável e dispõe de um filtro DCM para a fusão dos sensores, como apresentado na seção 3.1.2. Com isso, é possível adquirir os dados já fundidos e prontos para serem usados.

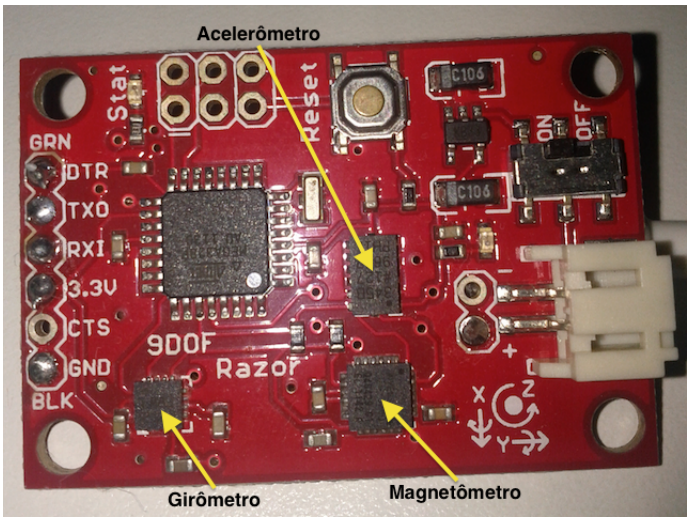

Figura 4.1: A placa Razor utilizada para aquisição dos sinais.

Os dados capturados através da Razor são enviados para o computador *host* via portal serial USB, onde é iniciada a fase de pré-processamento do sinal.

4.2 Pré-processamento

Os sinais fundidos são robustos mas, para uma aplicação que busca apresentar visualmente o resultados dessas movimentações, esses dados precisam ser filtrados novamente.

Por esse motivo, foram feitos alguns testes com filtros passa-baixas para minimizar os ruídos e as movimentações de alta frequência.

Inicialmente os filtros foram projetados utilizando o MatLab. O melhor de cada um dos tipos de filtro (IIR e FIR) foram implementados para verificar sua utilidade no sistema.

4.2.1 Design dos filtros passa-baixas digitais

Foram escolhidos três filtros de todos possíveis através do MatLab. O filtro de média móvel foi escolhido pela sua simplicidade, o melhor FIR foi o projetado usando janela gaussiana e o IIR que mais se destacou foi o filtro utilizando a aproximação de Butterworth.

Foram definidos, empiricamente, os requerimentos dos filtros passa-baixas digitais. Dada a frequência de amostragem (Fs), e com -20dB de atenuação na frequência de corte (Fc), foi possível projetar os filtros para atender a esses requisitos.

Janela Gaussiana

O MatLab provê uma ferramenta de projeto de filtros. Todos os filtros FIR foram testados utilizando frequência de amostragem (Fs = 10Hz) e frequência de corte (Fc = 3Hz).

Como é possível ver na figura 4.2, a resposta do filtro se mantém acima de -20dB antes da Fc e ao passar dessa frequência, nenhum sinal ultrapassa essa intensidade.

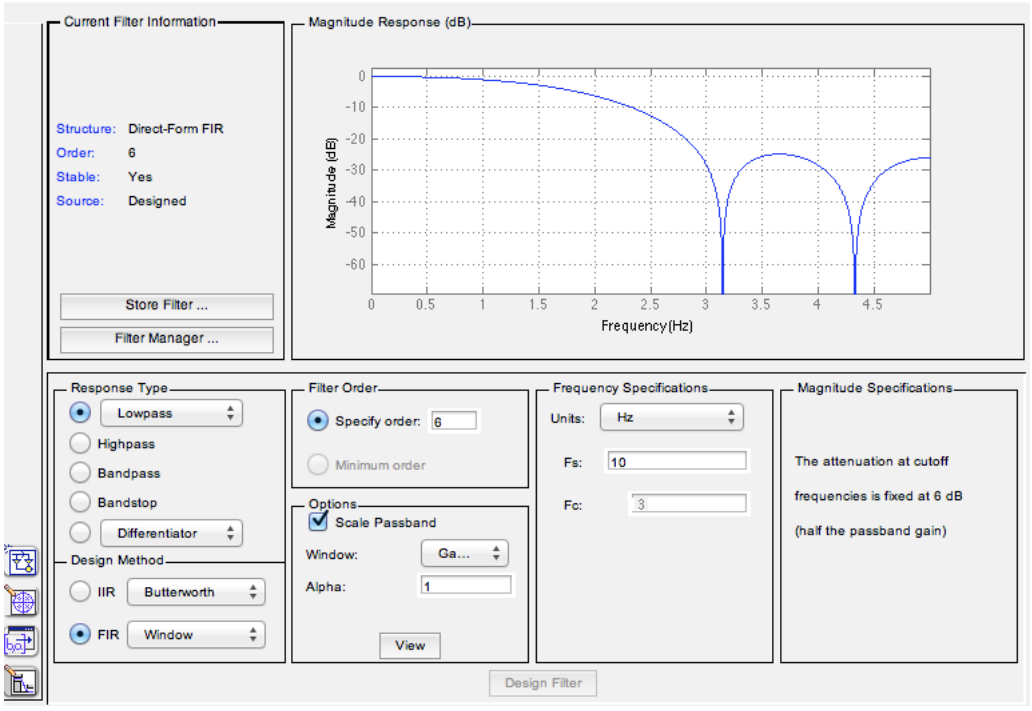

Figura 4.2: Design de um filtro FIR passa-baixas com janela gaussiana.

A melhor configuração encontrada para um filtro FIR foi uma janela gaussiana com alfa igual a 1 e ordem 5. Após ser projetado, os coeficientes do filtro foram extraídos e implementados no módulo de processamento.

Os coeficientes utilizados são:

- coef[0] = 0,000000000000000008832775598370409578262;
- coef[1] = 0,157440207956691974322538385422376450151;
- coef[2] = 0,342559792043307997921885998948710039258;
- coef[3] = 0,342559792043307997921885998948710039258;
- coef[4] = 0,157440207956691974322538385422376450151;
- coef[5] = 0,000000000000000008832775598370409578262.

Butterworth

Assim como os filtros FIR, todos do tipo IIR, que estão presentes no MatLab, foram modelados para verificar qual seria escolhido. A frequência de amostragem também foi fixada em (Fs = 10Hz), a frequência permitida foi definida como (Fpass = 2Hz) e a frequência de corte como (Fstop = 3).

Nesse caso, a ideia também é manter a resposta do filtro o mais próximo do 0dB até o momento que se ultrapassa a Fstop, onde a resposta do filtro deve se manter abaixo de -20dB.

No gráfico da figura 4.3, pode-se verificar a resposta desse filtro se mantém muito mais próximo de 0dB antes de alcançar a Fstop do que os filtros FIR. Tem-se que, após a frequência de corte, a resposta do filtro é sempre menor que -20dB, e, antes, ele se mantém o mais próximo do ideal.

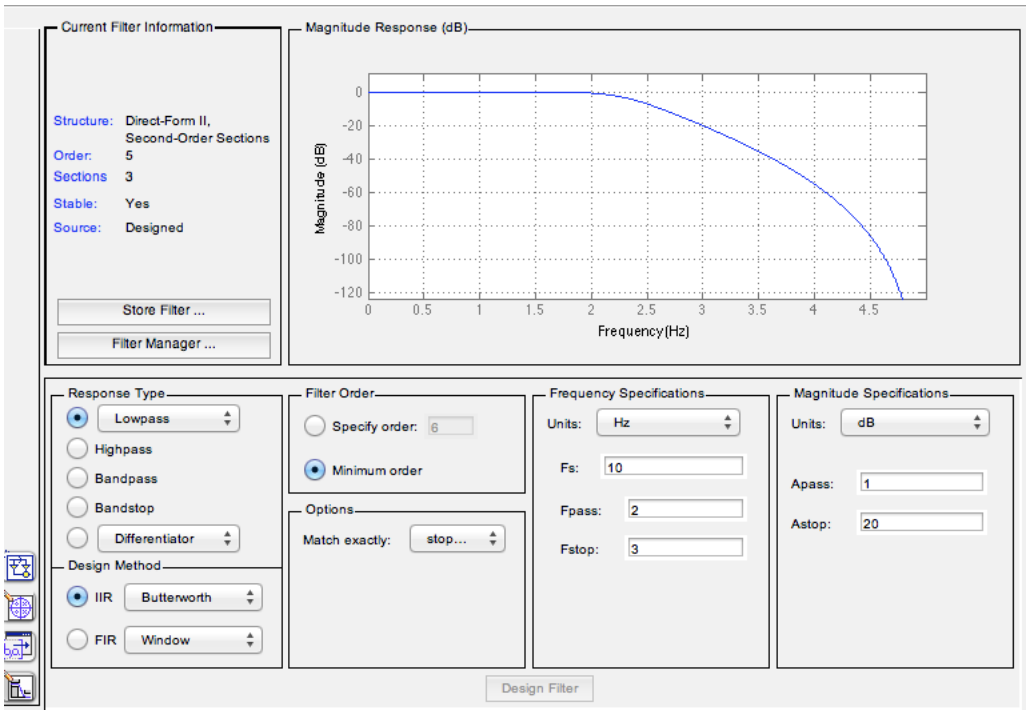

Figura 4.3: Design de um filtro IIR passa-baixas Butterworth.

A configuração utilizada para obter tal resultado foi a de um filtro do tipo Butterworth de ordem 4. Os coeficientes foram projetados e o filtro foi implementado para testá-lo com os dados reais do projeto.

Os coeficientes utilizados são:

- Coeficientes de entrada:
  - b[0] = 0,028;
  - b[1] = 0,053;
  - b[2] = 0,071;
  - b[3] = 0,053;
  - b[4] = 0,028;
- Coeficientes de saída:
  - a[0] = 1,000;
  - a[1] = −2,026;
  - a[2] = 2,148;
  - a[3] = −1,159;
  - a[4] = 0,279.

Implementação

Os filtros passa-baixas foram implementados no módulo de processamento. Para verificar qual era o melhor, o sensor foi deixado parado e então 50 amostras foram capturadas. Os gráficos 4.4, 4.5 e 4.6 foram feitos para comparação dos sinais após a aplicação de cada filtro.

Utilizando o gráfico 4.6 como exemplo é possível verificar alguns pontos importantes:

- O sinal sem filtro oscila bastante, mesmo quando movimentações de baixa frequência são notadas, como é o caso das amostras de 40 à 50;
- A janela gaussiana minimiza as mudanças de alta frequência, ou seja, mudanças que não se mantém são minimizadas (como visto nas amostras de 43 à 49), já modificações que perduram por mais tempo (frequência baixa), influenciam mais no resultado, como é o caso dessa movimentação de noventa graus observada;
- Como apresentado na figura 4.2, o filtro FIR com janela gaussiana reduz os sinais mesmo antes da frequência de corte. Isso pode ser verificado ao analisar as amostras como um todo, onde a saída do filtro leva um tempo maior para sofrer a influência de mudanças reais (de baixa frequência);
- O filtro IIR Butterworth consegue filtrar mudanças de alta frequência (amostras de 43 à 49) e ainda não apresenta um atraso tão grande para sofrer a influência de uma mudança de baixa frequência. Isso é confirmado ao observar o *design* desse filtro 4.3, no qual apenas próximo a frequência de corte que o sinal é reduzido.

Esses gráficos de comparação mostraram que o filtro IIR do tipo Butterworth foi o mais apropriado para ser utilizado, visto que esse filtra mais efetivamente mudanças de alta frequência, mas, por sua vez, não causa significativo atraso na saída.

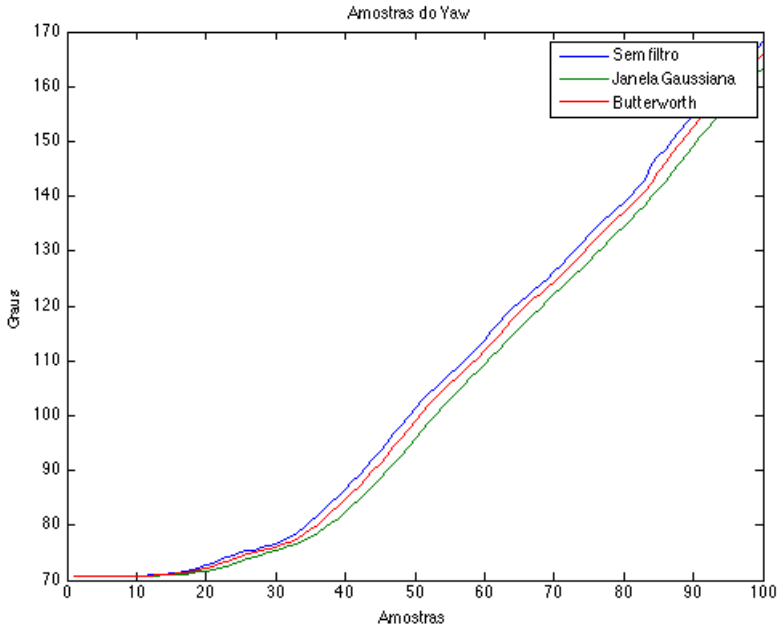

Figura 4.4: Gráfico de comparação dos dados Yaw.

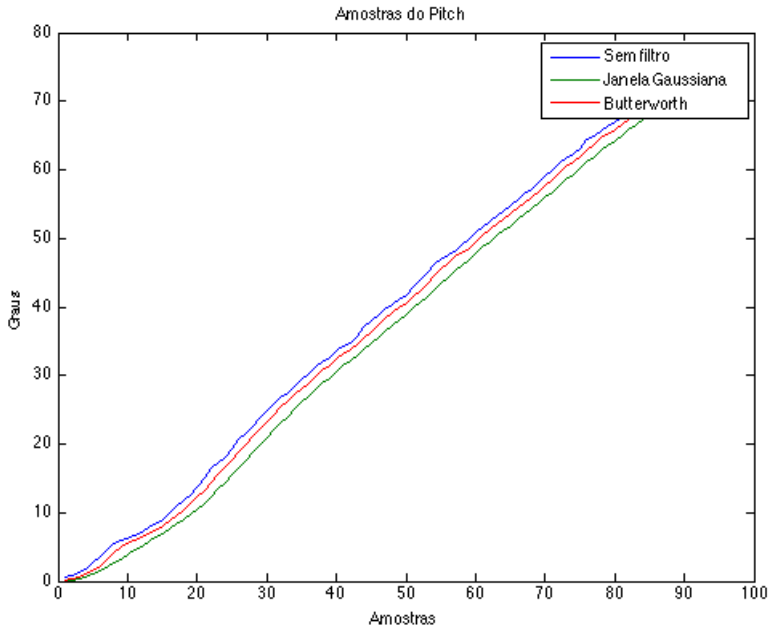

Figura 4.5: Gráfico de comparação dos dados Pitch.

4.3 Módulo de Processamento

Para torná-lo modularizado, foi necessário parametrizar o código. Quando executado em modo "verbose" é possível ver os ângulos capturados e processados, além de verificar o período entre cada medição, mostrado na figura 4.7.

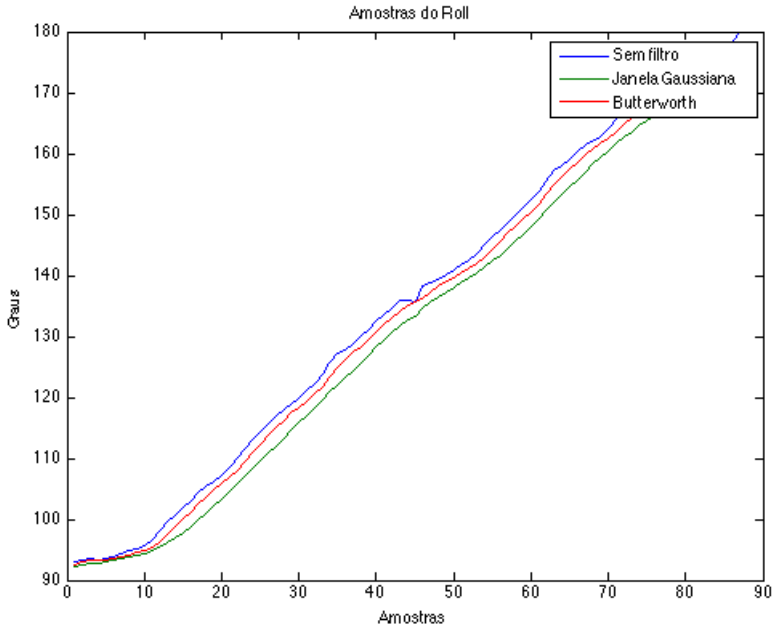

Figura 4.6: Gráfico de comparação dos dados Roll.

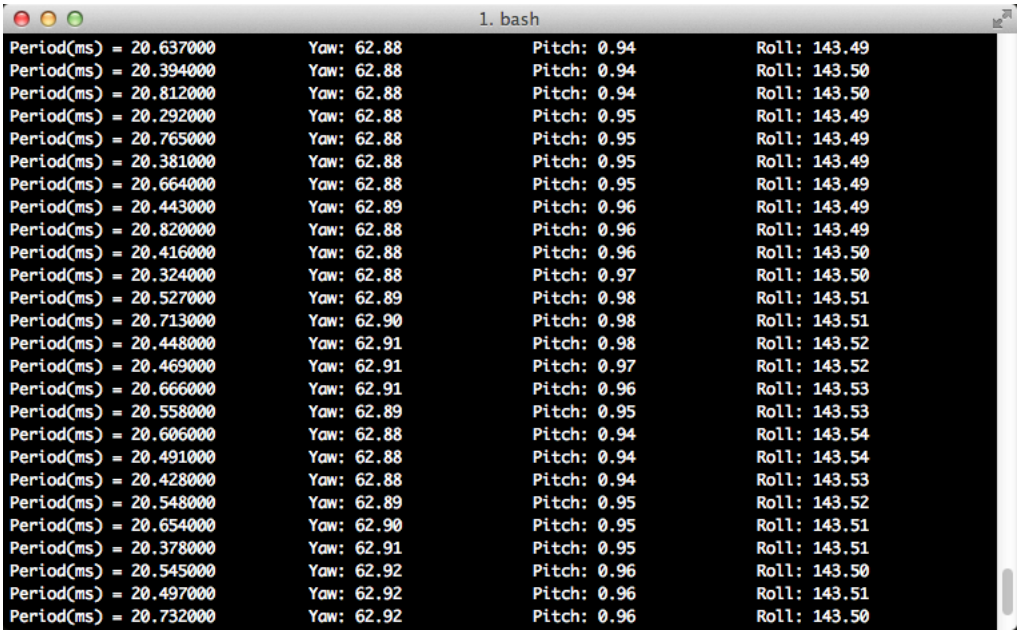

Figura 4.7: Saída para debug no módulo de Processamento.

Como é possível observar na figura 4.7, o período entre cada medida fica em torno a 20ms. Ou seja, a frequência de amostragem está por volta de 50Hz.

Assim sendo, no projeto dos filtros, a utilização dos parâmetros Fs=10Hz e Fc=3Hz corresponde à aplicação de um filtro passa-baixas de frequência de corte de aproximadamente 15Hz. Foi possível observar bons resultados visuais que aumentaram a sensação de imersão para o usuário.

4.4 Google Sketchup

A comunicação entre o módulo de processamento e o *plugin* criado para o Sketchup é feita por meio de sistema de arquivos, utilizando um arquivo chamado "input.txt".

O módulo de processamento limpa o arquivo a cada iteração e adiciona uma única linha contendo os dados *yaw*, *pitch* e *roll* separados por vírgula.

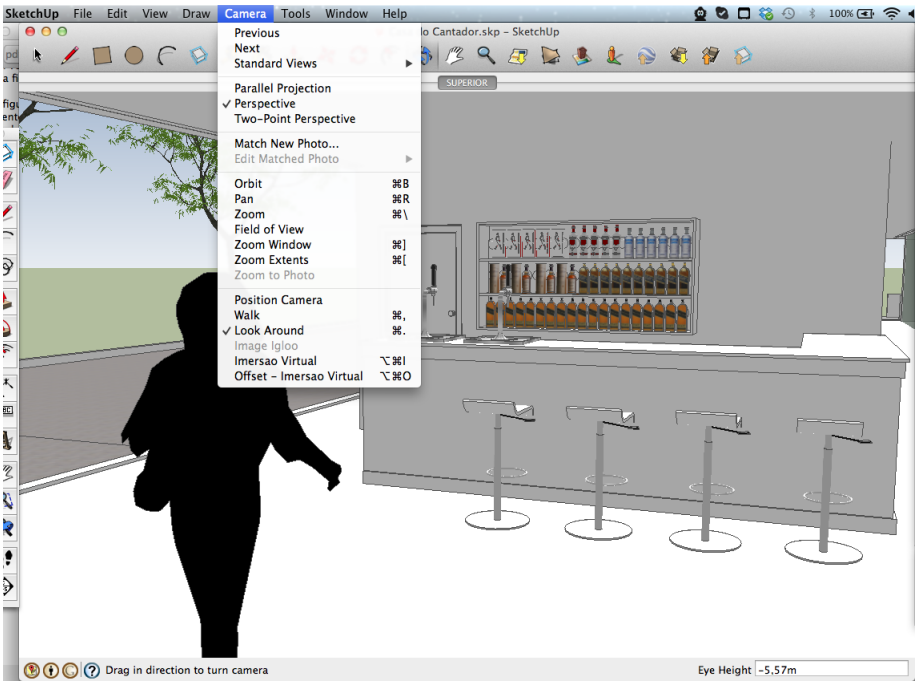

Figura 4.8: Tela do Sketchup com chamada do plugin aberta.

A figura 4.8 apresenta a tela do Google Sketchup e o menu, onde o *plugin* de Imersão Virtual é ativado. No mesmo menu é possível ver o *plugin Offset* para posicionamento correto da câmera.

Alguns problemas foram enfrentados na comunicação. Foi necessário utilizar um sistema de *lock* de arquivo para evitar que o módulo de processamento alterasse os dados, ao mesmo tempo em que o Sketchup os está lendo para renderização da tela.

4.5 Avaliação Final

Com o protótipo criado, foi possível observar as reações do usuário ao interagir com o ambiente virtual onde estava imerso, projetado pelo Google Sketchup, como pode ser observado na figura 4.9.

Contando com a interface de movimentação da câmera que o Sketchup provê, o projeto proporcionou uma ótima experiência ao usuário, onde é possível observar o ambiente a sua volta, sem a necessidade de girar o *mouse* inúmeras vezes, como é o caso da interface padrão de movimentação do Sketchup.

A integração com o Sketchup foi interessante, pelo fato dessa ferramenta ser utilizada para projetos de arquitetura. Muitos desses projetos são repletos de detalhes, e a utilização do protótipo possibilita ao usuário imergir nesse ambiente e visualizar esses detalhes, muito

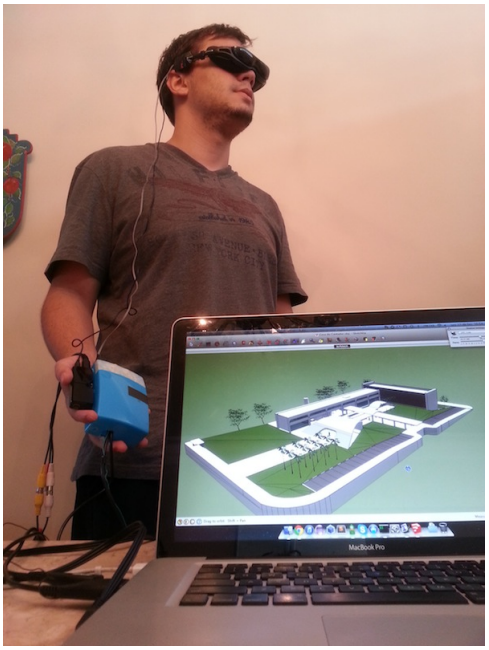

Figura 4.9: Protótipo sendo utilizado por um usuário, e na tela do computador a visão projetada.

melhor do que vê-los na tela do computador. Um bom exemplo disso é a figura 4.8 que mostra a câmera dento de um dos cômodos de um projeto real criado no Google Sketchup.

O sistema se comunica com a placa de aquisição de dados Razor através de porta serial USB. Logo, é necessário a utilização de um cabo para que a placa se comunique com o computador *host*. O *firmware* da Razor já provê a possibilidade de se comunicar via *bluetooth*, porém, isso não foi implementado.

A escolha de se utilizar comunicação por cabo deve-se ao fato de que os óculos não poderiam se comunicar via *wireless* sem a utilização de outros dispositivos, devido a alta taxa de transmissão necessária para apresentar a tela do computador. Com a necessidade de se utilizar o cabo S-Video para transmitir a imagem, um cabo USB é passado juntamente com esse outro para a transmissão dos dados da Razor, visto que ambos se conectam ao *host*.

Cerca de quatro usuários utilizaram o sistema, e foi relatado que a experiência foi bem satisfatória. Como se trata de um projeto arquitetônico, os usuários comentaram da facilidade de se observar detalhes como se estivessem no local já construído.

O único detalhe que não foi bem apreciado por eles foi a falta de textura do projeto criado no Sketchup, o que diminui a sensação de imersão, já que era visível a falta de semelhança com o mundo real. Vale ressaltar que esse problema não está relacionado ao sistema proposto, mas sim à aplicação do Google Sketchup. A melhor forma de se resolver esse problema é criar projetos com mais texturas e tentar torná-lo o mais próximo do mundo real.

O Google Sketchup possui uma forma de adicionar textura, mas essa função não é viável para aplicações em tempo real. Essa funcionalidade torna o projeto muito lento, o que o torna inviável para esse projeto.

Capítulo 5

Conclusão

Este trabalho apresentou uma proposta de integração de um dispositivo IMS ao Google Sketchup. Um modelo baseado em módulos foi apresentado com o intuito de facilitar as modificações futuras na implementação dos componentes utilizados pelo sistema.

Um protótipo foi criado para rastrear a movimentação da cabeça do usuário via IMS, utilizando fusão de sensores (girômetro, acelerômetro e magnetômetro), e movimentar a câmera dentro da ferramenta do Google Sketchup, e por fim, apresentar a imagem nos óculos que projetam a tela do computador.

As reações dos usuários, imersos no ambiente virtual criado, foram monitoradas, e notou-se que a utilização do sistema facilita=ou a observação de detalhes de projetos arquitetônicos criados no Sketchup. Porém, para um maior grau de imersão, é necessário que esse projeto seja feito o mais próximo do real, utilizando texturas e outros objetos que o tornem realista.

Vários desafios foram enfrentados no decorrer deste trabalho. Foi necessário aprender a utilizar uma comunicação via porta serial e a criar um protocolo para que essa comunicação fosse bem estabelecida. Além disso, o uso de filtros de fusão sensores e o filtros passa-baixas digitais fizeram necessário um estudo maior de vários conceitos matemáticos bem úteis ao se trabalhar com processamento de sinais. A utilização da API do Google Sketchup precisou ser estudo a fundo para implementar a rotação da câmera a partir da orientação da cabeça do usuário.

Algumas propostas de melhorias e trabalhos futuros podem ser listadas:

- Utilizar as informações do acelerômetro para se deslocar na cena do Google Sketchup;
- Fazer o sistema ser sem fio, facilitando a movimentação do usuário por um cenário virtual;
- Integração com outros módulos de apresentação. Jogos de computador, consoles de vídeo game, câmeras acopladas a robôs, entre outros exemplos que poderiam ser integrados;
- Utilizar um óculos com maior resolução ou com possibilidade de visão em 3 dimensões.

Referências

[1] Arastoo, Aflatoon, and Buqrath. Kalman Filters I. http://techpanacea.blogspot.com.br/2011/03/kalman-filters-i.html, Fevereiro 2013.

[2] A. J. Baerveldt and R. Klang. A low-cost and low-weight attitude estimation system for an autonomous helicopter. In *Intelligent Engineering Systems, 1997. INES '97. Proceedings., 1997 IEEE International Conference on*, pages 391 –395, Setembro 1997. 2, 4, 15

[3] W. S. Bainbridge. *Berkshire encyclopedia of human-computer interaction*, volume 2. Berkshire Publishing Group LLC, 2004. 1

[4] A. S. Barbosa. *Infraestrutura da Telepresença 3D e Teleoperação*. PhD thesis, UFRJ, 2009. 1

[5] A. P. L. Bó. Desenvolvimento de um sistema de localização 3D para aplicação em robôs aéreos. Master's thesis, Universidade de Brasília, Brasília, Distrito Federal, Julho 2007. 4

[6] P. Coho. Body balancing. http://thewhyaboutthis.com/2013/02/05/body-balancing/, Fevereiro 2013. vii, 6

[7] B V Dasarathy. Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38, 1997. vii, 13, 14

[8] Belur V. Dasarathy. Information fusion - what, where, why, when, and how? editorial. *Information Fusion*, 2(2):75–76, 2001. 10

[9] H.F. Durrant-Whyte and S Utete. Reliability in decentralised data fusion networks. *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI94)*, pages 215–221, 1994. 13

[10] Sparkfun Electronics. 9 Degrees of Freedom - Razor IMU. https://www.sparkfun.com/products/10736, Fevereiro 2013. 20, 21

[11] Sparkfun Electronics. Sparkfun Electronics. https://www.sparkfun.com/, Fevereiro 2013. 21

[12] W. Elmenreich and S. Pitzek. Using Sensor Fusion in a Time-Triggered Network. Proceedings of the 27th Annual conference of the IEEE Industrial Electronics Society, 2001. 12

[13] J. Esfandyari, R. De Nuccio, G. Xu, and STMicroelectronics. Introduction to mems gyroscopes. http://www.electroiq.com/articles/stm/2010/11/introduction-to-mems-gyroscopes.html, Fevereiro 2013. 6

[14] K. E. Foote and D. J. Huebner. Error, Accuracy, and Precision. Technical report, The Geographer's Craft Project, Department of Geography, University of Texas at Austin, 1995. 11

[15] B. Fry and C. Reas. Processing. http://processing.org/, Fevereiro 2013. 22

[16] O. Grau. *Virtual Art - From Illusion to Immersion*. MIT Press/Leonardo Books, Cambridge, MA, 2003. 1

[17] K. W. Hayward and L. G. Stephenson. Method and system of determining altitude of flying object, 10 2003. 7

[18] T. Henderson, M. Dekhil, R. Kessler, and M. Griss. Sensor fusion. In Bruno Siciliano and Kimon Valavanis, editors, *Control Problems in Robotics and Automation*, volume 230 of *Lecture Notes in Control and Information Sciences*, pages 193–207. Springer Berlin / Heidelberg, 1998. 10.1007/BFb0015084. 10

[19] Apple Inc. Uiacceleration class reference. http://developer.apple.com/library/IOS/documentation/UIKit/Reference/UIAcceleration_Class/index.html, Fevereiro 2013. vii, 5

[20] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960. 10, 16

[21] T. R. Kuphaldt. Low-pass filters. http://www.allaboutcircuits.com/vol_2/chpt_8/2.html, Julho 2011. 8

[22] P. S. Maybeck. *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*. 1979.

[23] G. T. McKee. *Multisensor Fusion for Computer Vision*, chapter What can be fused?, pages 71–84. Nato Advanced Studies Institute Series F, 1993. 11

[24] Microsoft. Matrix.rotateyawpitchroll(single,single,single). http://msdn.microsoft.com/en-us/library/windows/desktop/bb281738(v=vs.85).aspx, Fevereiro 2013. vii, 6

[25] R. Pausch, D. Proffitt, and G. Williams. Quantifying immersion in virtual reality. *ACM Press*, Agosto 1997. 1

[26] R. Pausch, D. Proffitt, and G. Williams. The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications. *IEEE Robotics and Automation Society*, Outubro 2001. 2

[27] W. Premerlani and P. Bizard. Direction cosine matrix imu: Theory. http://gentlenav.googlecode.com/files/DCMDraft2.pdf, Maio 2009. vii, 16

[28] R. E. Salustiano. Aplicação de técnicas de fusão de sensores no monitoramento de ambientes. Master's thesis, UNICAMP, Campinas, SP, Janeiro 2006. 12

[29] Julius O. Smith. *Spectral Audio Signal Processing*. http://ccrma.stanford.edu/~jos/sasp/, Março 2013. 8

[30] Stanford. Accelerometer overview. http://www.stanford.edu/class/me220/data/lectures/lect08/lect_4.html, Fevereiro 2013. vii, 7

[31] Stefan. Creating Stereo Images in Sketchup. http://www.stefan-motz.de/wordpress/2013/01/creating-stereo-images-in-sketchup/, Fevereiro 2013. vii, 29

[32] Jonathan Steuer. Defining virtual reality: Dimensions determining telepresence. *Journal of Communication*, 42(4):73–93, 1992. 4

[33] J. Strickland. How Virtual Reality Works. http://electronics.howstuffworks.com/gadgets/other-gadgets/virtual-reality.htm, Junho 2007. 1

[34] L. Wald. A European proposal for terms of reference in data fusion. In *International Archives of Photogrammetry and Remote Sensing*, volume XXXII, pages 651–654, Budapest, Hungary, 1998. 10

[35] E. Waltz and J. Llinas. *Multisensor Data Fusion*. Artech House, Norwood, Massachusetts, 1990. 10

[36] O. J. Woodman, C O. J. Woodman, and O. J. Woodman. An introduction to inertial navigation, 2007. 4

[37] ©Arduino. Arduino Home Page. http://www.arduino.cc/, Julho 2011. 20

Apêndice A

Código do Plugin do Google Sketchup

```
// imersao_virtual.rb
class ImersaoVirtual

  YAW_CONST = 0
  PITCH_CONST = 1
  ROLL_CONST = 2

  def initialize
    # Store the initial target
    @initial_target = Sketchup.active_model.active_view.camera.target
    @initial_up = Sketchup.active_model.active_view.camera.up
    @initial_eye = Sketchup.active_model.active_view.camera.eye

    @@yaw0 = false
    @@pitch0 = false
    @@roll0 = false
  end

  def self.reset_initial_values
    @@yaw0 = false
    @@pitch0 = false
    @@roll0 = false
  end

  def nextFrame(view)
    begin
      # Directory name where input file is placed
      dir_name = File.expand_path(File.dirname(__FILE__))

      File.open(dir_name+"/input.txt", "r") do |file|
        file.flock File::LOCK_EX

        # Get rotation values from file
        rotations = file.readline
```

```
# Write 0 to file, requesting new data
File.open(dir_name+"/input.txt", "w") do |file2|
  file2.write "0"
end

# Camera infos
camera = view.camera
eye = view.camera.eye
up = view.camera.up

if (rotations and rotations != "0")
  rotations = rotations.split ","

  # Reset to initial values
  camera.set @initial_eye, @initial_target, @initial_up

  rotations.each_with_index do |rotation,index|
    target = camera.target
    up = view.camera.up

    # Calculare rotation in radians
    rotation = rotation.to_f

    # Make the transformation
    if index == YAW_CONST
      @@yaw0 = rotation unless @@yaw0

      t = Geom::Transformation.rotation(eye, camera.yaxis,
          celsius_to_radians(rotation-@@yaw0))
      target.transform!(t)
    elsif index == PITCH_CONST
      @@pitch0 = rotation unless @@pitch0

      t = Geom::Transformation.rotation(eye, camera.xaxis,
          celsius_to_radians(rotation-@@pitch0))
      target.transform!(t)
    elsif index == ROLL_CONST
      @@roll0 = rotation unless @@roll0

      t = Geom::Transformation.rotation(eye, camera.zaxis,
          celsius_to_radians(rotation-@@roll0))
      up.transform!(t)
    end

    # Apply the transformation on camera
    camera.set eye, target, up
  end

else
```

```
          # Execute the transformation that does nothing
          # Needed to keep loop execution of nextFrame in Sketchup plugin
          t = Geom::Transformation.rotation(eye, camera.yaxis, 0)

          target = camera.target
          target.transform!(t)
          camera.set eye, target, up
        end
      end

    rescue
      # Rescue exception when target are paralelal to up vector
    end

    return true
  end

  def celsius_to_radians celsius
    (Math::PI/180) * celsius
  end
end

# This adds an item to the Camera menu to activate our custom animation.
UI.menu("Camera").add_item("Imersao Virtual") {
  Sketchup.active_model.active_view.animation = ImersaoVirtual.new
}

UI.menu("Camera").add_item("Offset - Imersao Virtual") {
  ImersaoVirtual.reset_initial_values
}
```