UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Implementação em sistema embarcado de método para estimação de orientação utilizando filtro de Kalman, sensores inerciais e magnetômetro.

Autora: Jéssica Bohn da Vida

Orientador: Prof. Dr. Evandro Luís Linhari Rodrigues

São Carlos

2016

Jéssica Bohn da Vida

Implementação em sistema embarcado de método para estimação de orientação utilizando filtro de Kalman, sensores inerciais e magnetômetro.

Trabalho de Conclusão de Curso apresentado
à Escola de Engenharia de São Carlos, da
Universidade de São Paulo

Curso de Engenharia Elétrica

ORIENTADOR: Prof. Dr. Evandro Luís Linhari Rodrigues

São Carlos

2016

Bohn da Vida, Jéssica
B677i Implementação em sistema embarcado de método para estimação de orientação utilizando filtro de Kalman, sensores inerciais e magnetômetro / Jéssica Bohn da Vida; orientador Evandro Luis Linhari Rodrigues. São Carlos, 2016.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2016.

1. fusão de sensores. 2. sensores inerciais. 3. magnetômetro. 4. filtro de Kalman. 5. quaternions. 6. estimação de orientação. 7. sistema embarcado. 8. método de Gauss Newton. I. Título.

FOLHA DE APROVAÇÃO

Nome: Jéssica Bohn da Vida

Título: "Implementação em sistema embarcado de método para estimação de orientação utilizando filtro de Kalman"

Trabalho de Conclusão de Curso defendido e aprovado em 29/11/2016,

com NOTA 10,0 (DEZ, ZERO), **pela Comissão Julgadora:**

Prof. Associado Evandro Luis Linhari Rodrigues - Orientador - SEL/EESC/USP

Mestre André Luis Martins - Doutorando - SEL/EESC/USP

Mestre Tatiane Cristina da Costa Fernandes - Doutoranda - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado José Carlos de Melo Vieira Júnior

Dedicatória

Dedico este trabalho à minha família, amigos e a todos os responsáveis pela Escola de Engenharia de São Carlos que propiciam a formação de bons profissionais para o cenário brasileiro.

Jéssica Bohn da Vida.

Agradecimentos

Agradeço à minha mãe Márcia Bohn da Vida, meu pai Edson Martins da Vida e minha irmã Leticia Bohn da Vida por todo o apoio, motivação e confiança para continuar estudando.

Agradeço aos amigos pela partilha de conhecimento e pelas dificuldades superadas juntos durante toda a graduação. Em especial ao Alex Monteiro Sartin, Alexandre Moretti Bernado, Gabriel Camoese Salla, Leonardo Borges Farçoni, Marília Lírio Dourado, Nícolas dos Santos Rosa e Plínio Gonçalves Bueno Ferreira.

Também agradeço aos professores Evandro Luís Linhari Rodrigues e Rodrigo Andrade Ramos pela orientação fornecida no decorrer do desenvolvimento deste trabalho e durante o período acadêmico.

Jéssica Bohn da Vida.

Resumo

Este trabalho contém o desenvolvimento e implementação em sistema embarcado de método para estimação de orientação no espaço tridimensional utilizando a versão clássica de filtro de Kalman para sistemas lineares como algoritmo de fusão de sensores inerciais e magnetômetro, e utilizando *quaternions* para representação de rotação. A linearização da matriz de observação do filtro é feita utilizando-se o algoritmo de Gauss Newton. Através deste, calcula-se um quatérnion que representa a rotação do acelerômetro e do magnetômetro entre a posição de referência e a posição atual. Este quatérnion é então aplicado como entrada de medições no filtro em substituição aos dados desses sensores. Além da implementação do método descrito, também foi desenvolvida uma aplicação auxiliar para telemetria que permite visualizar graficamente dados de interesse, como leituras dos sensores e saída do sistema. Como resultados, observou-se um bom desempenho do método utilizado em termos de tempo de processamento e estabilidade da saída. Porém, o algoritmo de Gauss Newton limitou o intervalo de rotação para o qual o sistema converge em $\pm\frac{\pi}{2}$ rad, o que é indesejado para o projeto, tornando necessária sua alteração. Os sensores inerciais e magnetômetro de baixo custo utilizados, quando calibrados, tiveram desempenho suficiente em termos de taxa de ruído e sensibilidade, porém dever-se-á implementar um tratamento adequado de acelerações lineares que interferem na aquisição da aceleração da gravidade.

Palavras-Chave: fusão de sensores, sensores inerciais, magnetômetro , filtro de Kalman, quatérnion, estimação de orientação, método de Gauss Newton, sistema embarcado.

Abstract

This document contains the development, implementation in embedded system and test of a method for orientation estimation in 3D space utilizing the classic version of the Kalman Filter for linear systems as inertial sensor and magnetometer fusion algorithm and utilizing quaternions for rotation representation. The linearization of the observation matrix is accomplished utilizing the Gauss Newton algorithm, which calculates a quaternion that represents the rotation of the accelerometer and magnetometer from the reference frame to the current frame. This quaternion is applied as input of the filter in replace of those sensors measurements. In addition to the described method, an auxiliar application for Windows was also developed, providing a data of interest graphical visualization, such as sensors measurements and system output. The method utilized shown a good processing performance and output stability. However, the Gauss Newton algorithm limited the rotation intervale of converge of the system. The low cost inertial and magnetometer sensors utilized in the project, after calibration, had a sufficient performance in terms of noise rate and sensibility, but linear accelerations treatment will be necessary once there is a significant interference in the gravity acceleration aquisition.

Keywords: sensor fusion, inertial sensors, magnetometer, Kalman filter, quaternion, orientation estimation, Gauss Newton method, embedded system.

Lista de figuras

Lista de tabelas

Siglas

IMU	*Inertial measurement unit* - Unidade de Medida Inercial
MIMU	*Magnetic and inertial measurement unit* - Unidade de Medida Inercial e Magnética
MEMS	*Micro electrical mechanical system* - Sistema Micro Mecânico-Elétrico
DoF	*Degrees of freedom* - Graus de liberdade
USB	*Universal Serial Bus* - Barramento Serial Universal
API	*Application Programming Interface* - Interface do Programa de Aplicação
IDE	*Integrated Development Environment* - Ambiente de Desenvolvimento Integrado
KFA	*Kalman Filter Algorithm* - Algoritmo do Filtro de Kalman
GNA	*Gauss Newton Algorithm* - Algoritmo de Gauss Newton
MEMS	*Microelectromechanical System* - Sistema micro-eletro-mecânico
DMP	*Digital Motion Processor* - Processador Digital de Movimento
I2C	*Inter-Integrated Circuit* - Circuito Inter-Integrado
SPI	*Serial Peripheral Protocol* - Protocolo Serial Periférico
FIFO	*First in, First Out* - Registrador do tipo fila (primeiro a entrar, primeiro a sair)
ADC	*Analog to Digital Converter* - Conversor analógico digital

Conteúdo

Capítulo 1

Introdução

Dispositivos eletrônicos chamados *wearable* (que podem ser "vestidos") estão aos poucos saindo do meio acadêmico e alcançando usuários comuns. O desenvolvimento destes dispositivos, capazes de monitorar e interpretar atividades do corpo humano e a partir delas permitir o controle de outros equipamentos eletrônicos, é um tema bastante atual e ainda pouco explorado, sendo muito atraente para projetos de inovação e percursor de uma grande mudança no relacionamento homem-máquina.

Dois pontos principais tornam o desenvolvimento de *wearables* bastante favorável atualmente. O primeiro diz respeito à disponibilidade de plataformas embarcadas com grande capacidade de processamento, da ordem de centenas de MHz ou superior, de pequenas dimensões e custo acessível. Isto também permite a redução do custo de projeto do ponto de vista de sensoriamento, uma vez que é possível utilizar sensores de menor robustez e, portanto, menor custo, e fazer o tratamento dos dados adquiridos via *software*, com algoritmos de natureza estatística por exemplo. Vários métodos de tratamento de dados estão disponíveis na literatura, apresentando diferentes aplicações, níveis de complexidade e precisão, com destaque para o filtro de Kalman discreto, caracterizado pelo baixo custo computacional e ótima estimação de variáveis aleatórias. O segundo ponto está relacionado à disponibilidade de dispositivos de sensoriamento com boa exatidão e alto nível de integração (vários sensores em um único *chip*). A partir disto é possível projetar um dispositivo de uso agradável para o usuário e com grande variedade de aplicações [1] [2] [3] [4].

Neste trabalho são utilizados sensores inerciais e magnetômetro, todos de baixo custo, filtro de Kalman como algoritmo de fusão sensorial e tratamento das leituras, representação de rotação utilizando *quaternions* e algoritmo numérico de Gauss Newton para linearização da matriz de Observação de Kalman.

1.1 Motivação e requisitos de projeto

Este trabalho abrange a primeira etapa do projeto "Luva Inteligente", que tem como intuito o desenvolvimento de um dispositivo *wearable* capaz de identificar gestos específicos do usuário, traduzindo-os para comandos que poderão ser utilizados para controlar sistemas eletrônicos. Para compor o sistema completo, além desta primeira etapa, tem-se o desenvolvimento de uma inteligência artificial que deve receber o conjunto de dados de orientação dos dedos e mão do usuário e identificar um comando válido. A partir disto, pode-se definir alguns requisitos para o sistema de estimação de orientação a ser utilizado.

- precisão de posicionamento angular mínima necessária para que se possa identificar padrões de gesto (da ordem de 10^{-1});
- independência do ambiente e de aparatos fixos, para melhor experiência do usuário;
- baixo custo;
- tamanho reduzido do *hardware* para que possa ser "vestido";
- baixo consumo de energia;
- taxa de atualização alta o suficiente para que o usuário perceba a resposta do sistema de forma praticamente instantânea (da ordem de 100ms a 200ms);
- ausência de limitação do intervalo de rotação, propiciando liberdade de movimento;

1.2 Objetivos

Este trabalho tem como objetivos o desenvolvimento e implementação em *hardware* embarcado de um método de estimação de orientação utilizando filtro de Kalman para fusão de dados de acelerômetro , giroscópio e magnetômetro , além da avaliação da viabilidade de utilização deste para detecção de movimentos dos dedos e da mão do usuário.

Capítulo 2

Embasamento Teórico

2.1 Estimação de orientação utilizando sensores inerciais e magnetômetro

Este trabalho tem o intuito de implementar um método estimação de orientação de baixo custo que possa ser aplicado para captar gestos do usuário. Desta forma, optou-se por utilizar sensores mais baratos (com maior taxa de ruído e menor exatidão), o que exige tratamento via *software* das leituras coletadas.

Além disso, para que seja possível estimar orientação de forma totalmente embarcada utiliza-se o conceito de fusão de sensores (*sensor fusion*), ou seja, a utilização conjunta dos dados adquiridos a fim de se obter uma informação mais confiável do que a que seria fornecida por cada sensor separadamente [5]. Neste trabalho, a fusão dos sensores é realizada como descrito a seguir [6].

2.1.1 Considerações sobre os sensores

Giroscópio [1]

O giroscópio é responsável pela medição do deslocamento angular do objeto e dá informação de orientação a partir de uma referência calibrada quando o sistema é iniciado.

- Sensor de velocidade angular;
- Baseado no Efeito Coriolis, que ocorre quando um objeto se move em um referencial que está em rotação. Neste caso, uma "força inercial" age sobre o objeto, a qual dá

[1] Informações obtidas em [7].

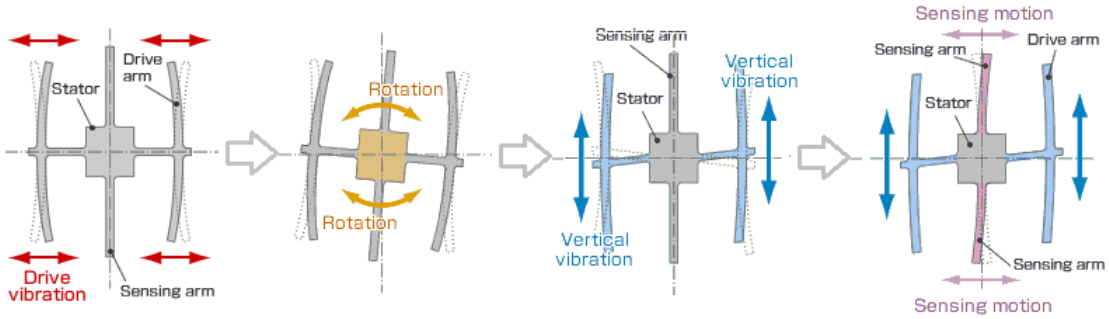


Figura 2.1: Efeito Coriolis em giroscópio MEMS [8].

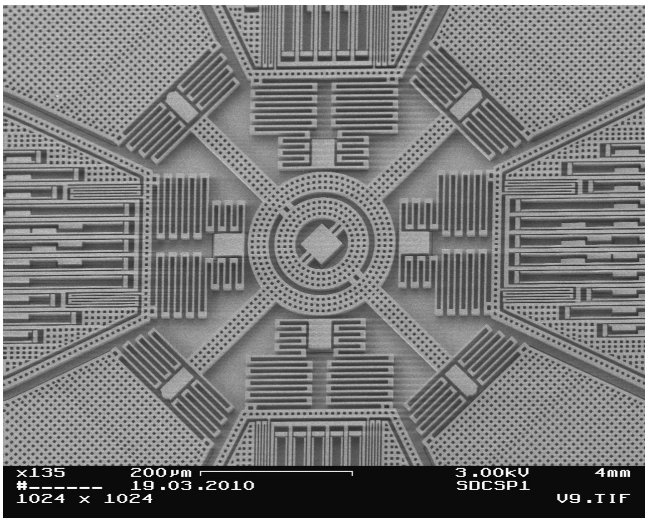


Figura 2.2: Exemplo MEMS [9].

informação sobre a velocidade angular do referencial, como mostrado na figura 2.1;

- Giroscópio MEMS é composto por braços vibrantes, ilustrados na figura 2.2;
- Mede sua própria movimentação e não um referencial externo;
- Deficiência: apresenta o efeito de *drift* (deriva) devido à integração realizada para se obter a informação de ângulo a partir das leituras de velocidade angular. Uma vez que ocorrem pequenas vibrações no sistema, caracterizadas como um ruído branco com média não nula, quando da integração das leituras obtém-se um valor sempre crescente mesmo com o sistema parado (figura 2.8).

Acelerômetro[1]

O acelerômetro fornece uma referência com relação à aceleração gravitacional da Terra (rotações em torno dos eixos x e y).

- Sensor de aceleração;

- Pode ser pensado como uma massa em uma caixa ou ainda como uma massa amortecida por molas (figura 2.3);
- A chamada massa sísmica balança quando o dispositivo é movimentado;
- Capta qualquer tipo de movimento (linear ou angular);
- Em queda livre, mede zero já que não há movimento relativo entre a massa e o referencial “caixa”;
- Deficiência: muito sensível a pequenas trepidações. Uma possível solução para melhorar a relação sinal-ruído é utilização de um filtro passa-baixa, porém isto torna o sistema lento, como mostra a figura 2.7;

É importante observar que a ordem de grandeza das acelerações lineares provindas do movimento da mão do usuário é muito inferior à aceleração gravitacional.

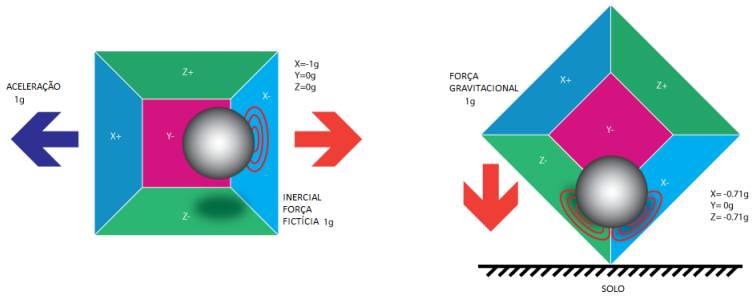


Figura 2.3: Efeito da aceleração na massa sísmica [10]

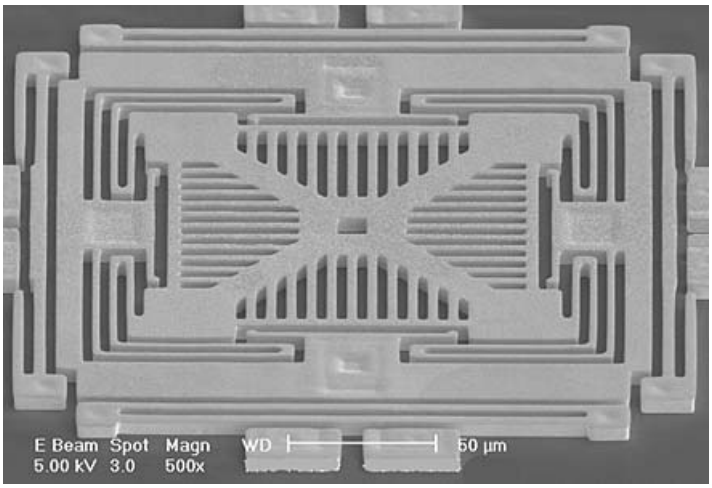


Figura 2.4: Exemplo de acelerômetro MEMS [11].

Magnetômetro[1]

o magnetômetro fornece uma referência com relação ao campo magnético da Terra (rotação em torno do eixo z ou *heading*).

- Sensor de campo magnético;
- Extremamente sensível;
- Baseia-se no efeito Hall, o qual descreve a deflexão da corrente elétrica que flui por um condutor quando este está imerso em um campo magnético perpendicular, como mostrado nas figuras 2.5 e 2.6;
- Não há necessidade de inserção de material magnético no silício;
- Deficiência: muito sensível a pequenas trepidações. Uma possível solução para melhorar a relação sinal-ruído é utilização de um filtro passa-baixa, porém isto torna o sistema lento, como pode ser visto na figura 2.7;

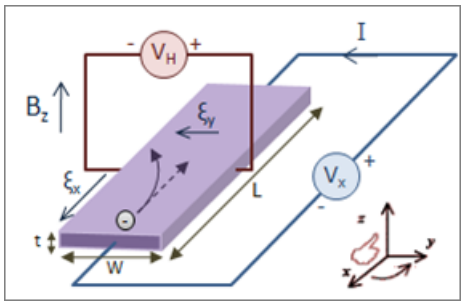


Figura 2.5: Efeito Hall - exemplo 1 [12].

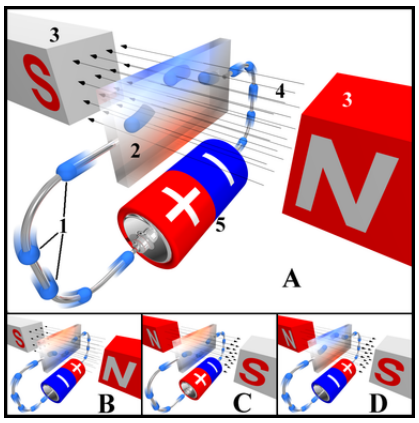


Figura 2.6: Efeito Hall - exemplo 2 [13].

Os três sensores analisados apresentam problemas e não são confiáveis para serem utilizados individualmente. A partir disso, vem a ideia de se fazer a fusão dos dados (*sensor fusion*) destes sensores de forma a se obter informações mais confiáveis sobre do sistema. A utilização de sensores inerciais e magnetômetro apresenta uma redundância da informação de orientação (referências globais e medição local de movimento).

Devido às pequenas dimensões dos sensores e à facilidade para embarcar o sistema (baixo consumo de energia e independência do ambiente), a utilização destes sensores para estimação de orientação de membros humanos é bastante atraente e vem sendo explorada cada vez mais [1] [3] [16].

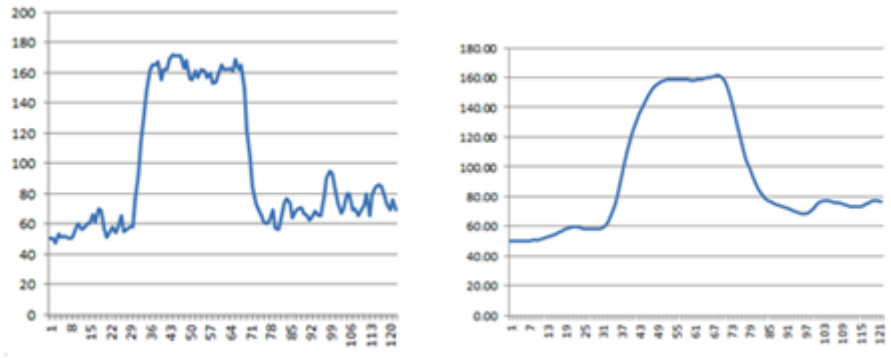


Figura 2.7: Aplicação de filtro passa-baixa nos dados do acelerômetro e magnetômetro [14].

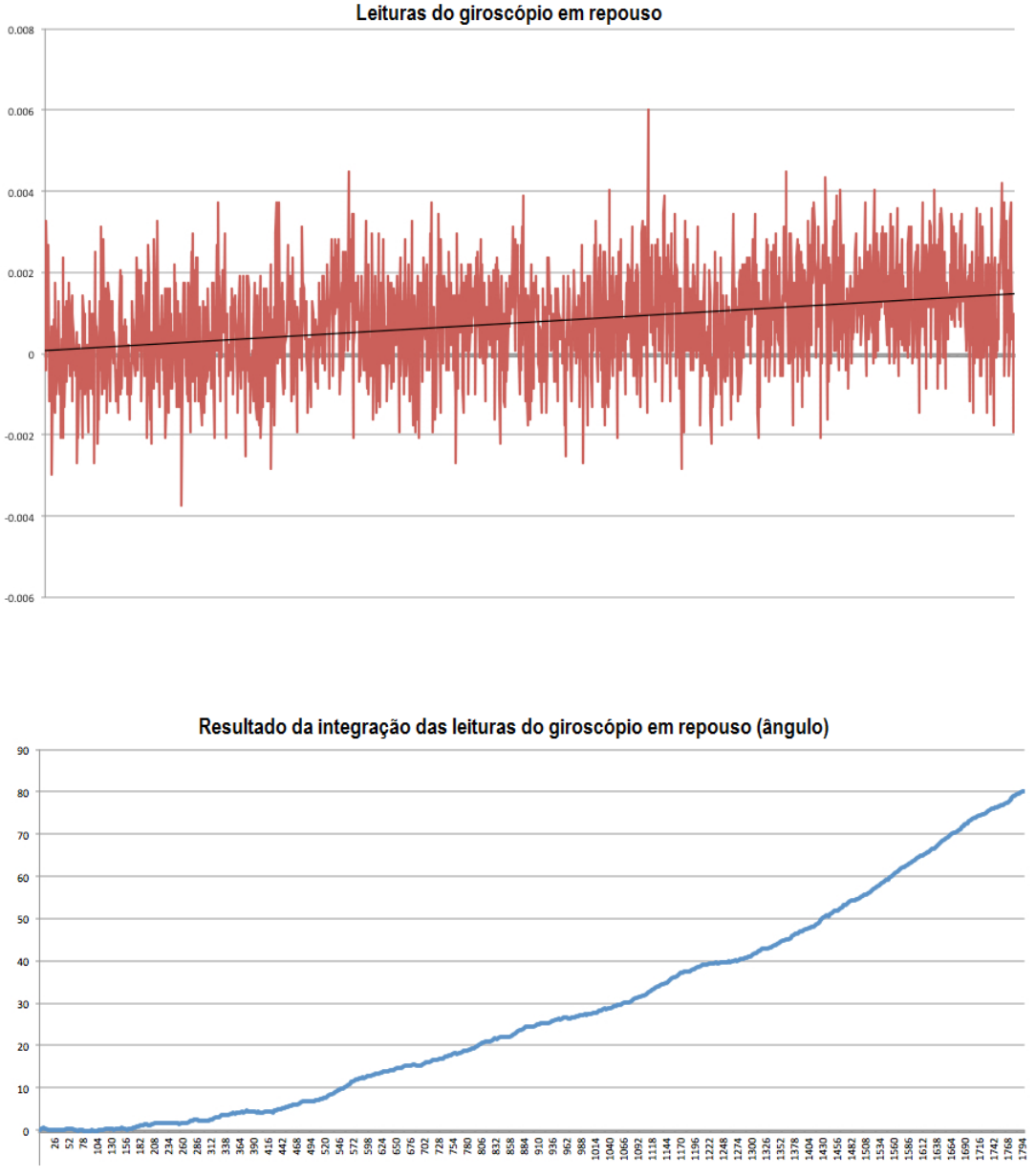


Figura 2.8: Efeito de *drift* nos dados do giroscópio [15].

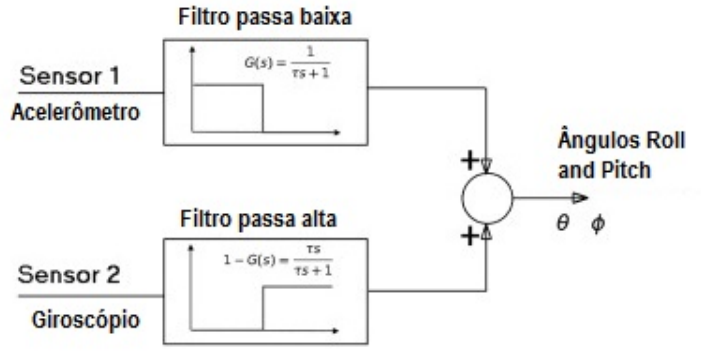


Figura 2.9: Diagrama de blocos do Filtro Complementar [18].

2.2 Métodos para fusão de sensores

Existem muitos algoritmos que realizam a fusão de dados, desde bastante simples até com alto nível de sofisticação, sendo necessária uma análise de custo computacional *versus* precisão dos resultados e tempo de processamento para a escolha do mais adequado. A seguir, são analisados dois exemplos.

2.2.1 Filtro Complementar

O filtro Complementar é composto por um algoritmo bastante simples e tem como objetivo utilizar dados de giroscópio e acelerômetro para estimar a orientação do objeto. Para isto, são utilizados um filtro passa-baixa e um filtro passa-alta que se "complementam" [17].

O filtro passa-baixa é utilizado para remover picos de dinâmica muita rápida que aparecem na saída do acelerômetro devido à vibrações ou inclinações no sistema. Já o filtro passa-alta é utilizado para remover o *drift* (seção 2.1.1) de dinâmica lenta que aparece nos dados obtidos do giroscópio.

Esta composição caracteriza o processamento inicial das leituras dos sensores. A partir disso, os dados são adicionados e compõe uma informação de posição angular ou orientação do objeto, como mostrado na figura 2.9. A figura 2.10 mostra o diagrama de Bode da composição dos dois filtros.

O filtro Complementar é um algoritmo de baixa complexidade e pouco robusto [1] uma vez que utiliza os dados crus dos sensores, os quais tem natureza bastante ruidosa. Não há alguma forma de redundância ou análise estatística da tendência de movimento e portanto leituras inconsistentes não são tratadas, como ocorre no filtro de Kalman, descrito a seguir.

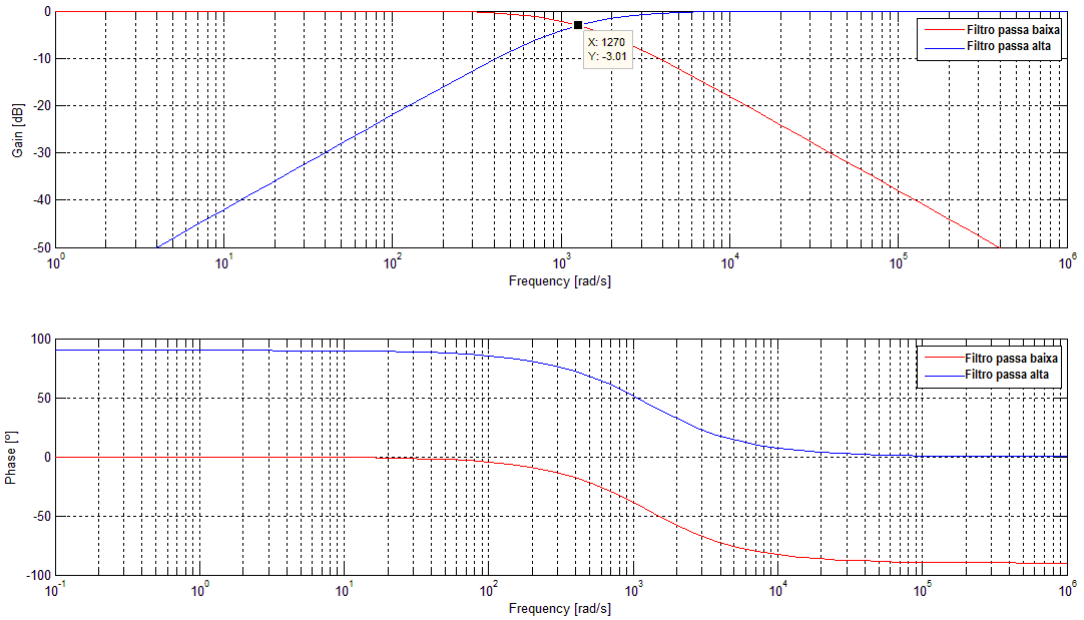


Figura 2.10: Diagrama de Bode dos filtros passa-alta e passa-baixa, ambos com frequência de corte de 200Hz, obtido via Matlab.

2.2.2 Filtro de Kalman

Desenvolvido por Rudolf E. Kalman nos anos 60, ainda hoje é um dos métodos mais comumente utilizados para fusão de dados. Isto se deve ao reduzido custo computacional do algoritmo e por ser considerado um estimador ótimo para sistemas unidimensionais e lineares com erro de distribuição gaussiana [19] [20].

As aplicações do filtro de Kalman estão relacionadas à estimativa de variáveis aleatórias de interesse, seja quando da utilização de sensores ruidosos ou quando da necessidade de antecipação dessas variáveis (em sistemas de alta velocidade por exemplo). A utilização do filtro também está intimamente relacionada ao desenvolvimento de sistemas de posicionamento global, navegação assistida e veículos autônomos.

O algoritmo é composto por um conjunto de equações que proveem um método recursivo capaz de realizar a estimativa das variáveis de interesse por meio do conceito de **probabilidade condicional** ou teorema de Bayes, como mostrado na equação 2.1, onde a função densidade de probabilidade do evento A ($p(A)$) depende do evento B (probabilidade de A dado B $p(A|B)$). Neste algoritmo, o qual também é uma forma de controle realimentado, a predição da variável de interesse é feita em um dado instante de tempo, sendo que tais valores são corrigidos a partir de medições (geralmente ruidosas) do sistema [19] [21], como representado na figura 2.11.

Figura 2.11: Diagrama de blocos do algoritmo de Kalman [19]

$$p(A\,|\,B) \;=\; \frac{p(B\,|\,A)p(A)}{p(B)} \tag{2.1}$$

Para o filtro de Kalman, tem-se que:

$$p(x_k\,|\,z_k) \;\sim\; N(\hat{x}_k, P_k) \tag{2.2}$$

Na equação 2.2 a função densidade de probabilidade da variável de interesse (x_k) depende das medições realizadas no sistema (z_k) e tem distribuição gaussiana cujo valor médio é o estimador que se quer calcular ($\hat{x}_k$) e cuja variância é o erro quadrático deste estimador, dado pela matriz P_k. No caso de múltiplas variáveis, a matriz P_k compõe a matriz de variâncias cruzadas ou covariâncias, como mostrado na equação 2.3, em que $E[\ ]$ é o operador Esperança Matemática. O algoritmo funciona de forma a reduzir o erro quadrático ou a covariância do estimador e, consequentemente, aproximando-o do valor real x_k [19].

$$P_k = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] \tag{2.3}$$

Para tanto, as variáveis de interesse a serem estimadas são tratadas como estados de um processo dinâmico linear modelado por uma equação diferencial estocástica [2] . O modelo em espaço de estados do processo, em tempo discreto, é mostrado na equação 2.4.

$$\hat{x}_k = A\hat{x}_{k-1} + Bu_{k-1} + w_{k-1} \tag{2.4}$$

$$p(w) \sim N(0, Q), \quad Q_k = E[(w_k)(w_k)^T] \tag{2.5}$$

[2]. Um processo estocástico ou aleatório é um conjunto de variáveis aleatórias que representam a evolução de um sistema no tempo. Este processo pode evoluir por infinitas direções a partir de quaisquer condições iniciais. Processos estocásticos podem ser considerados o oposto de processos determinísticos, que evoluem de uma única maneira e são descritos por equações diferenciais ordinárias. Na derivação do filtro de Kalman, sinais aleatórios de interesse são considerados como as saídas de um sistema dinâmico linear excitado por ruído branco [20].

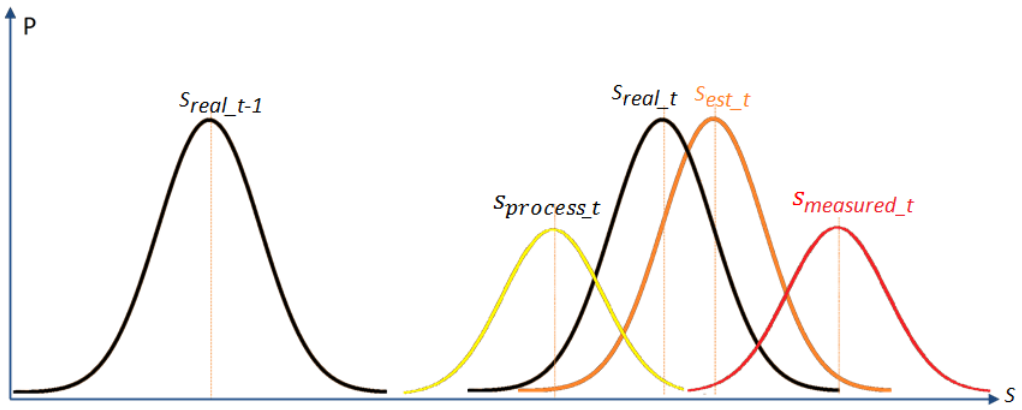


Figura 2.12: Representação do algoritmo de Kalman. s representa a variável de interesse e P a probabilidade.

onde $\hat{x}$ é o vetor de estados estimados, A é a matriz de estados, B é a matriz que relaciona as entradas de controle com os estados do processo (matriz de controle) e u é vetor de entradas conhecidas de controle. O vetor w contém o ruído ou erro no modelo do processo, assumido como sendo de distribuição normal com média zero e matriz de covariâncias Q [19] [21], como mostrado na equação 2.5.

As observações ou medições do sistema, variáveis aleatórias observadas [3] , são modeladas de acordo a equação 2.6, onde z é o vetor de medições, H é a matriz de observação que relaciona os estados com as medições (matriz de observação) e ν é o vetor contendo o ruído nas medições (ruído dos sensores), também assumido como sendo de distribuição normal com média zero e matriz de covariâncias R. É importante notar que para que se possa aplicar o filtro, o sistema deve ser observável.

$$\hat{z}_k = H\hat{x}_k + \nu_k \tag{2.6}$$

$$p(\nu) \sim N(0,R), \quad R_k = E[(\nu_k)(\nu_k)^T] \tag{2.7}$$

A figura 2.12 ilustra o funcionamento do algoritmo de Kalman utilizando as funções densidade de probabilidade do modelo de processo (equação 2.4) e das medições do sistema (equação 2.6) para compor uma função densidade de probabilidade, mais confiável que as duas anteriores isoladamente, para o estimador de estado.

[3] O filtro de Kalman utiliza o modelo estocástico em espaço probabilístico de estados chamado modelo oculto de Markov (HMM - *Hidden Markov Model*) [22] [23] no qual tem-se um conjunto de saídas observáveis $y_k \sim p(y_k|x_k)$ e um conjunto de estados não-observáveis $x_k \sim p(x_k|x_{k-1})$ de um sistema. Quando utiliza-se a distribuição gaussiana como função densidade de probabilidade para o conjunto de variáveis de estado x, o sistema é modelado como $x_k = Ax_{k-1} + q_k$ e $y_k = Hx_k + r_k$.

Comparativamente ao filtro Complementar, o filtro de Kalman apresenta grande robustez com relação à leituras inconsistentes (ruído) devido à sua natureza estatística. Apesar de apresentar maior complexidade computacional, existem versões digitais do algoritmo de Kalman otimizadas para a implementação em sistemas embarcados e que necessitem de baixo tempo de processamento [19], como mostrado a seguir.

Algoritmo do Filtro de Kalman

A seguir tem-se o algoritmo completo do filtro de Kalman discreto na forma de processo de "predição" e "correção" [19]. Para o primeiro, estados preditos são denotados com $^-$. A saída do segundo é o estimador da variável de interesse.

Fase de Predição

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \qquad (2.8)$$

$$\hat{P}_k^- = A\hat{P}_{k-1}A^T + Q \qquad (2.9)$$

Fase de Correção

$$K_k = \frac{P_{k-1}^- H^T}{H P_{k-1}^- H^T + R} \qquad (2.10)$$

$$\hat{x}_k = \hat{x}_k^- + K\left(z_k - H\hat{x}_k^-\right) \qquad (2.11)$$

$$P_k = (I - K_K H)P_k^- \qquad (2.12)$$

A fase de predição do filtro utiliza a modelagem do sistema, que não é perfeita, para fazer a *predição* do estimador do estado de interesse. Além disso, nesta etapa também é realizada a predição da matriz P de covariâncias do estimador.

Na fase de correção, tem-se a entrada das medições do sistema e a *correção* do estimador do estado. Para isto, primeiramente faz-se o cálculo do ganho de Kalman K, responsável por ponderar entre o valor predito e o valor medido para o estimador. Este é o passo mais importante do algoritmo e pode ser interpretado da seguinte forma (equação 2.10) [24].

- Quanto maior a covariância da medição obtida (matriz R) relativamente à covariância do estimador (matriz P), menos confiável é a medição, menor será K e, portanto, menor será a influência da medição no valor do estimador no passo k;

- Por outro lado, quanto menor a covariância da medição relativamente à covariância do estimador, maior será K e maior será a contribuição da medição para o valor do estimador no passo k, já que esta medição é mais confiável.

Para que o sistema seja bem representado e o filtro convirja de maneira coerente com o sistema físico é necessário ajustar os valores da matriz Q de covariâncias. Este ajuste

geralmente é feito de forma empírica, uma vez que o comportamento do sistema não é totalmente conhecido. Para valores muito grandes de Q, as estimações de estado tornam-se muito ruidosas devido ao aumento de P, do ganho de Kalman e da influência das medições no estimador. Portanto deve-se ajustar o maior valor de Q sem que as estimações de estado tornem-se muito ruidosas.

A parcela $(z_k - H\hat{x}_k^-)$, equação 2.11, também é chamada residual ou termo de correção e representa o quanto o valor predito $\hat{x}^-$ deve ser ajustado, a partir da medição obtida, para obter-se o estimador ótimo $\hat{x}$ no passo k.

2.3 Métodos de representação de orientação

Existem vários métodos para descrição de posição e orientação de objetos no espaço tridimensional. A seguir serão analisados os métodos mais comumente utilizados, suas vantagens e desvantagens visando o custo computacional para aplicação em sistemas embarcados e a liberdade de movimento (graus de liberdade).

2.3.1 Matriz de orientação

Matrizes de orientação compõe uma ferramenta matemática utilizada para realizar operações de rotação de vetores. Dado um vetor inicial (ponto de partida) e uma matriz de rotação, obtém-se a posição do ponto que desloca-se na superfície de uma esfera de raio constante e igual ao módulo do vetor inicial. Portanto, a matriz de orientação não contém informação de translação, apresentando três graus de liberdade [25].

$$R_x(\alpha) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & sin(\alpha) \\ 0 & -sin(\alpha) & cos(\alpha) \end{vmatrix}, \quad R_y(\beta) = \begin{vmatrix} cos(\beta) & 0 & -sin(\beta) \\ 0 & 1 & 0 \\ sin(\beta) & 0 & cos(\beta) \end{vmatrix}, \quad R_z(\gamma) = \begin{vmatrix} sin(\gamma) & cos(\gamma) & 0 \\ cos(\gamma) & -sin(\gamma) & 0 \\ 0 & 0 & 1 \end{vmatrix} \tag{2.13}$$

Para o espaço tridimensional, matrizes de orientação tem dimensão 3x3 e podem representar uma composição de rotações. Para um sistema de coordenadas cartesiano, cujos eixos compõe a referência da operação, a matriz de orientação que representa rotações consecutivas em *x, y e z* é mostrada na equação 2.14.

$$R_{x,y,z}(\alpha,\beta,\gamma) = R_x(\alpha)\,R_y(\beta)\,R_z(\gamma) \rightarrow \tag{2.14}$$

$$R_{x,y,z}(\alpha,\beta,\gamma) = \begin{vmatrix} cos(\gamma)cos(\beta) & cos(\beta)sin(\gamma) & -sin(\beta) \\ cos(\gamma)sin(\alpha)sin(\beta) - cos(\alpha)sin(\gamma) & cos(\alpha)cos(\gamma) + sin(\alpha)sin(\gamma)sin(\beta) & cos(\beta)sin(\alpha) \\ sin(\alpha)sin(\gamma) + cos(\alpha)cos(\gamma)sin(\beta) & cos(\alpha)sin(\gamma)sin(\beta) - cos(\gamma)sin(\alpha) & cos(\beta)cos(\beta) \end{vmatrix}$$

Portanto, o vetor rotacionado é dado pela equação 2.15.

$$v_{rot}(x,y,z) = R_{x,y,z}(\alpha,\beta,\gamma)v_{ini}(x,y,z) \tag{2.15}$$

Observa-se que o custo computacional para utilizar a matriz de orientação é bastante alto, tanto para o armazenamento (matriz 3x3) quanto para a obtenção da informação de orientação do objeto (produto matricial).

2.3.2 Ângulos de Euler

Ângulos de Euler são bastante utilizados para representação de orientação de objetos, principalmente em aplicações de aviação. De acordo com o Teorema de Euler, pode-se representar a orientação do objeto pela composição de três rotações. Partindo-se de uma situação em que eixos do objeto (x,y,z) estão alinhados com os eixos do referencial inicial (x_0,y_0,z_0), são realizadas rotações em torno de (x,y,z) até que o objeto atinja sua posição final. Os ângulos de rotação nos eixos x,y,z chamados, respectivamente, *roll*, *pitch* e *yaw*. As figuras 2.13 e 2.14 exemplificam a utilização de ângulos de Euler.

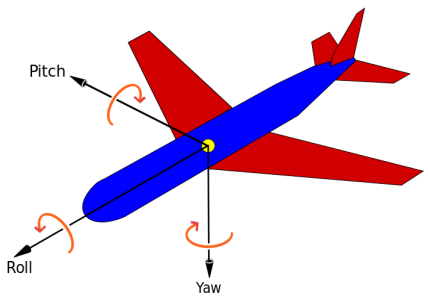


Figura 2.13: Ângulos de Euler - exemplo 1 [26]

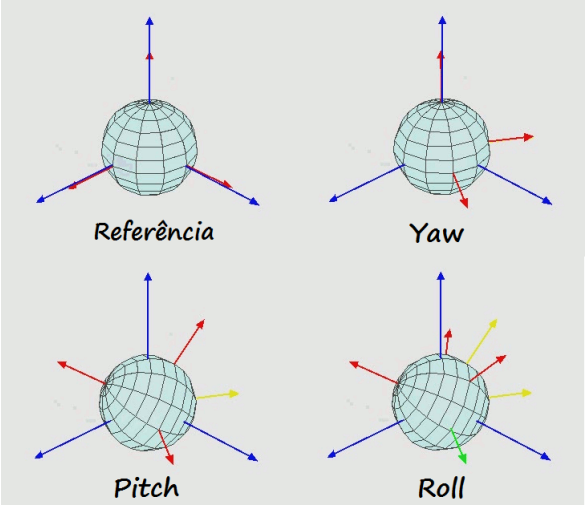


Figura 2.14: Ângulos de Euler - exemplo 2 [26]

Para se obter a posição do objeto a partir do referencial utilizando ângulos de Euler, utiliza-se a mesma operação matricial descrita na seção 2.3.1. Portanto, o custo computacional é praticamente o mesmo. Porém a utilização desta forma de representação facilita o armazenamento e manuseio da informação em algoritmos.

Gimbal lock

A forma de representação por ângulos de Euler apresenta deficiência para determinada orientação, que varia dependendo da sequência de rotações utilizada no sistema. Quando o objeto atinge esta orientação crítica, ocorre a perda de um grau de liberdade e o objeto pode movimentar-se em um espaço bidimensional degenerado. Isto pode ser observado utilizando-se por exemplo uma rotação de $\frac{\pi}{2}$ para *pitch* (θ) em um sistema *xyz*, como mostrado na equação 2.16.

$$R_{xyz}(\theta=\tfrac{\pi}{2}) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{vmatrix} \begin{vmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{vmatrix} \begin{vmatrix} sin(\gamma) & cos(\gamma) & 0 \\ cos(\gamma) & -sin(\gamma) & 0 \\ 0 & 0 & 1 \end{vmatrix} \rightarrow$$

$$R_{xyz}(\theta=\frac{\pi}{2}) = \begin{vmatrix} 0 & 0 & 1 \\ sin(\phi+\gamma) & cos(\phi+\gamma) & 0 \\ -cos(\phi+\gamma) & sin(\phi+\gamma) & 0 \end{vmatrix} \tag{2.16}$$

Observa-se que após o *pitch*, quaisquer valores de *yaw* e *roll* resultam em uma rotação na direção *z*, como se o objeto estivesse "preso" em um espaço representado pelo eixo *z* e por um eixo comsposto por *x* e *y*.

Em sistemas com movimentação limitada, o problema do *gimbal lock* pode ser ignorado. Porém, em sistemas que podem realizar rotações maiores que $\frac{\pi}{2}$ em qualquer um dos eixos

esta questão precisa ser contornada. Existem várias técnicas utilizadas para reestabelecer o rastreamento do objeto nestes casos e muitas delas estão baseadas na utilização de outros sensores, como magnetômetro por exemplo, que fornecem uma informação de referência com relação ao referencial global.

2.3.3 Vetor e ângulo

A forma de representação por vetor e ângulo utiliza um eixo de rotação *n* em torno do qual realiza-se uma rotação de ângulo θ seguindo-se a Regra da Mão Direita. O vetor *n* é tridimensional e com módulo unitário. Muitas vezes utiliza-se a notação em que a informação de rotação é armazenada em um vetor de 4 elementos, em que os três primeiros elementos representam o eixo *n* e o quarto elemento representa o ângulo de rotação.

Este método de representação é o mais fundamental e dele derivam as outras formas de representação citadas neste trabalho.

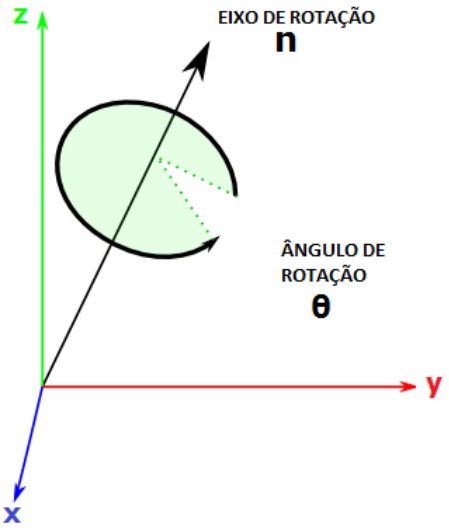


Figura 2.15: Representação de orientação por vetor e ângulo em 3D [27]

2.3.4 *Quaternions*

Quaternions são uma extrapolação de números complexos para o espaço tridimensional. A álgebra de *quaternions* foi introduzida por William Hamilton em 1843 com o intuito de aplicar a ideia de que, se números complexos representavam pontos no espaço bidimensional, o mesmo poderia ser feito para pontos no espaço tridimensional (figura 2.16) [6] [28] [29].

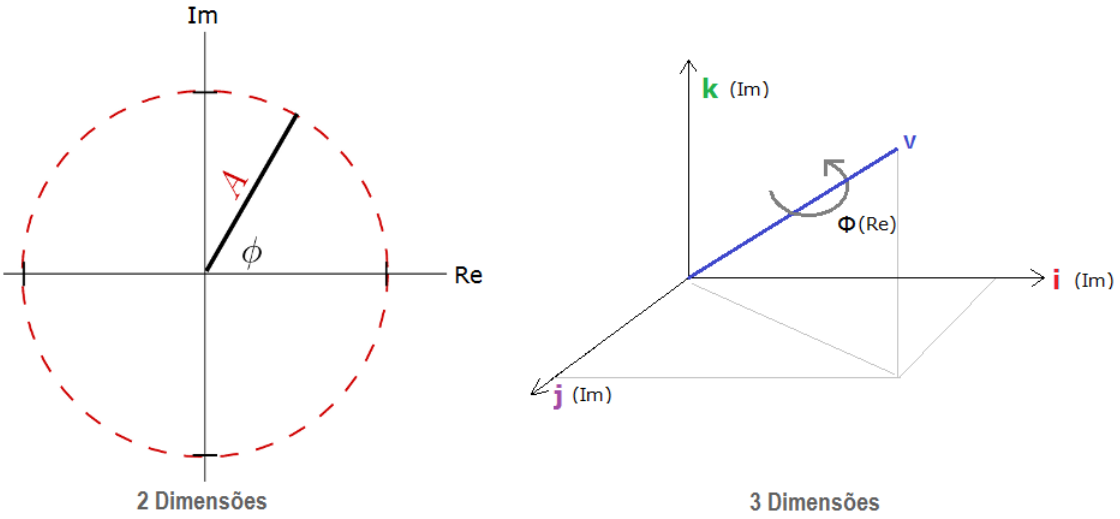


Figura 2.16: Representação de orientação por *quaternions*.

Assemelham-se bastante à forma de representação vetor e ângulo. Um quatérnion é definido, de forma genérica, como mostrado na equação 2.17, onde i, j e k são números imaginários que satisfazem as relações mostradas em 2.18.

$$\bar{q} = iq_1 + jq_2 + kq_3 + q_4 \tag{2.17}$$

$$\begin{gathered} i^2 = -1, \quad j^2 = -1, \quad k^2 = -1, \\ -ij = ji = k, \quad -jk = kj = i, \quad -ki = ik = j \end{gathered} \tag{2.18}$$

O *quaternion* é composto por uma parte real, q_4 e uma parte vetorial $q = \boldsymbol{i}q_1 + \boldsymbol{j}q_2 + \boldsymbol{k}q_3$. Quando as relações mostradas em 2.19 são obedecidas, diz-se que $\bar{q}$ é um *quaternion* de rotação em torno do vetor unitário $\hat{k}$ e de ângulo θ. O *quaternion* de rotação deve ser unitário, como mostrado em 2.20. Os *quaternions* $\bar{q}$ e $-\bar{q}$ descrevem uma rotação para a mesma posição final em um dado sistema de coordenadas (a representação vetor e ângulo não é única), sendo que o primeiro descreve o menor caminho para a posição final.

$$\bar{q} = \begin{bmatrix} q \\ q_4 \end{bmatrix}, \quad q = \begin{bmatrix} \boldsymbol{k_x} sin(\frac{\theta}{2}) \\ \boldsymbol{k_y} sin(\frac{\theta}{2}) \\ \boldsymbol{k_z} sin(\frac{\theta}{2}) \end{bmatrix} = \hat{\boldsymbol{k}} sin(\theta/2), \quad q_4 = cos(\theta/2) \tag{2.19}$$

$$|\bar{q}| = \sqrt{\bar{q}^T \bar{q}} = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2} = 1 \tag{2.20}$$

A rotação inversa é descrita pelo *quaternion* inverso ou conjugado, equação 2.21. Pode-se ainda utilizar a notação fasorial através da relação Euler do mesmo modo que para números

complexos convencionais, como mostrado na equação 2.22.

$$\bar{q}^* = \begin{bmatrix} -\boldsymbol{q} \\ q_4 \end{bmatrix} = \begin{bmatrix} -\hat{\boldsymbol{k}} sin(\theta/2) \\ cos(\theta/2) \end{bmatrix} \tag{2.21}$$

$$\bar{q} = e^{\frac{\theta}{2}(\boldsymbol{i}+\boldsymbol{j}+\boldsymbol{k})} = cos(\theta/2) + (\boldsymbol{i}+\boldsymbol{j}+\boldsymbol{k})sin(\theta/2) \tag{2.22}$$

Com relação ao método de ângulos de Euler, *quaternions* apresentam como vantagem a não ocorrência de *gimbal lock* e são equivalentes em termos de simplicidade de representação da orientação de objetos. Além da orientação, *quaternions* podem ser utilizados para representar uma rotação de maneira bastante enxuta (vetor de quatro posições). Portanto, são amplamente utilizados em aplicações onde há grande liberdade de movimento, por exemplo em computação gráfica, visão computacional, robótica, entre outros.

A álgebra de *quaternions* apresenta algumas características que devem ser conhecidas para sua utilização correta. Algumas destas características são mostradas a seguir.

Produto vetorial

$$\boldsymbol{q} \times \boldsymbol{p} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ q_1 & q_2 & q_3 \\ p_1 & p_2 & p_3 \end{vmatrix} = \begin{bmatrix} q_2p_3 - q_3p_2 \\ q_3p_1 - q_1p_3 \\ q_1p_2 - q_2p_1 \end{bmatrix} = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \lfloor \boldsymbol{q} \times \rfloor \boldsymbol{p} \tag{2.23}$$

Multiplicação de *quaternions*

A multiplicação pode ser utilizada para se obter um *quaternion* resultante que combina as rotações representadas por cada *quaternion* inicial, individualmente. É importante observar a ordem de aplicação das rotações. Na equação 2.24, a rotação representada por $\bar{p}$ seria aplicada primeiro.

$$\bar{q} \otimes \bar{p} = (\boldsymbol{i}q_1 + \boldsymbol{j}q_2 + \boldsymbol{k}q_3 + q_4)(\boldsymbol{i}p_1 + \boldsymbol{j}p_2 + \boldsymbol{k}p_3 + p_4) \rightarrow$$

$$\bar{q} \otimes \bar{p} = \begin{bmatrix} q_4p_1 + q_3p_2 - q_2p_3 + q_1p_4 \\ -q_3p_1 + q_4p_2 + q_1p_3 + q_2p_4 \\ q_2p_1 - q_1p_2 + q_4p_3 + q_3p_4 \\ -q_1p_1 - q_2p_2 - q_3p_3 + q_4p_4 \end{bmatrix} = \begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \rightarrow$$

$$\bar{q} \otimes \bar{p} = \begin{bmatrix} q_4 I_{3x3} - \lfloor \boldsymbol{q} \times \rfloor & \boldsymbol{q} \\ -\boldsymbol{q}^T & q_4 \end{bmatrix} \begin{bmatrix} \boldsymbol{p} \\ p_4 \end{bmatrix} \tag{2.24}$$

Representação de vetores

Um vetor tridimensional *v* pode ser representado na forma de *quaternion* $\bar{v}$ fazendo-se $\theta = 2\pi$, o que resulta em uma parte real nula.

$$\bar{v} = \begin{bmatrix} \boldsymbol{v_x} sin(\frac{\theta}{2}) \\ \boldsymbol{v_y} sin(\frac{\theta}{2}) \\ \boldsymbol{v_z} sin(\frac{\theta}{2}) \\ cos(\frac{\theta}{2}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{v_x} \\ \boldsymbol{v_y} \\ \boldsymbol{v_z} \\ 0 \end{bmatrix} \tag{2.25}$$

Rotação de vetores

A operação de rotação de um vetor da posição *a* para *b* pode ser feita utilizando-se um *quaternion* de rotação como mostrado na equação 2.26.

$${}^b v = {}^b_a\bar{q} \ {}^a v \ {}^b_a\bar{q}^* \tag{2.26}$$

Uma vez que a parte real da notação em *quaternion* de *v* é nula, a rotação pode ser realizada através da matriz ${}^b_a R_{3x3}$ mostrada em 2.27. O *quaternion* de rotação deve ter módulo unitário.

$${}^b v = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1 q_2 + q_3 q_4) & 2(q_1 q_3 - q_2 q_4) \\ 2(q_1 q_2 - q_3 q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2 q_3 + q_1 q_4) \\ 2(q_1 q_3 + q_2 q_4) & 2(q_2 q_3 - q_1 q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \begin{bmatrix} {}^a\boldsymbol{v_x} \\ {}^a\boldsymbol{v_y} \\ {}^a\boldsymbol{v_z} \end{bmatrix} \tag{2.27}$$

É importante destacar que para a notação de *quaternion* utilizada neste trabalho, como mostrado em 2.19, tem-se uma rotação positiva no sentido horário, utilizando um sistema de coordenadas Cartesiano representado pela mão esquerda [28]. Este tipo de notação não convencional é comumente utilizado em computação gráfica, com o qual a origem do sistema é posicionada no canto superior esquerdo, como mostrado na figura 2.17.

Multiplicação entre *quaternion* e vetor

Como na álgebra de *quaternion*s a multiplicação não é comutativa, na multiplicação entre *quaternion* e vetor aparecem duas matrizes características, a matriz Ω, utilizada para cálculo

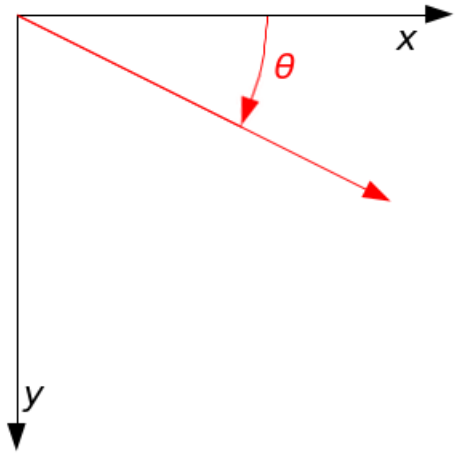


Figura 2.17: Rotação seguindo a mão esquerda, positiva no sentido horário, com eixo z apontando em direção ao observador [30].

da derivada, e a matriz Ξ. A relação entre estas é mostrada na equação 2.29, onde ω é um vetor tridimensional.

$$\Omega(\boldsymbol{\omega}) = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix}, \quad \Xi(\bar{q}) = \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix} \tag{2.28}$$

$$\Omega(\boldsymbol{\omega})\,\bar{q} = \Xi(\bar{q})\,\boldsymbol{\omega} = \bar{p} \tag{2.29}$$

Taxa de variação no tempo

No cálculo da derivada do *quaternion* com relação ao tempo, mostrada na equação 2.30, o vetor $\boldsymbol{\omega}$ contém as velocidades angulares do *quaternion* nos eixos $\boldsymbol{i}$, $\boldsymbol{j}$ e $\boldsymbol{k}$.

$$\dot{\bar{q}}(t) = \frac{1}{2}\begin{bmatrix} \omega \\ 0 \end{bmatrix} \otimes \bar{q} = \begin{bmatrix} -\lfloor \boldsymbol{\omega} \times \rfloor & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix} \bar{q} \rightarrow \tag{2.30}$$

$$\dot{\bar{q}}(t) = \frac{1}{2}\Omega(\boldsymbol{\omega})\bar{q} = \frac{1}{2}\Xi(\bar{q})\,\boldsymbol{\omega}, \tag{2.31}$$

$$\boldsymbol{\omega} = \lim_{\Delta t \to 0} \frac{\Delta \theta}{\Delta t}$$

Capítulo 3

Materiais e Métodos

3.1 Estimação de orientação com sensores inerciais e magnetômetro

A partir das características dos algoritmos de fusão de dados vistas na seção 2.2, optou-se por utilizar o filtro de Kalman (KFA - *Kalman Filter Algorithm*) para este projeto. Uma vez que os dados de saída do sistema serão, em trabalhos futuros, utilizados para reconhecimento de padrões de gestos do usuário, deseja-se que o sistema seja robusto e que a saída seja estável e fiel ao movimento realizado.

3.1.1 Filtro de Kalman utilizando *quaternions*

Dado o problema do *gimbal lock* apresentado pelo método de ângulos de Euler para representação de orientação, descrito na seção 2.3.2, e considerando-se que por este ser o projeto de um dispositivo *wearable* que deve permitir a maior liberdade de movimento e conforto para o usuário, optou-se por utilizar o método de representação por *quaternions* neste trabalho.

Para aplicar o KFA, como descrito na seção anterior, a um sistema é necessário, primeiramente, definir quais estados serão estimados e quais as medições disponíveis, garantindo-se a observabilidade. A partir disso, deve-se calcular as seguintes matrizes para a implementação:

- A (matriz de estados);
- B (matriz de entradas de controle);
- Q (matriz de covariância de processo);
- R (matriz de covariância de medição);

- H (matriz de observação);

Matriz A

O estado que se deseja estimar é o *quaternion* de rotação que leva do referencial do objeto para o referencial global (referência). Assim, é possível conhecer a orientação atual do objeto realizando-se a rotação por *quaternion*, como descrito na seção 2.3.4, do vetor da posição de referência. Desta forma, a matriz de estados A é obtida a partir da equação 2.31 de movimento do *quaternion*.

Para utilização da equação 2.31 é necessário discretizá-la com relação ao tempo. Isto é feito considerando-se um processo de amostragem com **integrador de ordem zero**, ou seja, o valor de ω é considerado constante durante o período de amostragem T_s. Assim, obtêm-se a equação 3.1, que pode ser escrita em série de Taylor, equação 3.2, considerando apenas três termos significativos [28] [31].

$$\bar{q}_{k+1} = e^{\left(\frac{1}{2}\Omega(\boldsymbol{\omega})\,T_s\right)}\,\bar{q}_k \rightarrow \tag{3.1}$$

$$\bar{q}_{k+1} = \left[I_{4x4} + \frac{1}{2}\Omega(\boldsymbol{\omega})\,T_s - \frac{1}{8}\Delta\theta^2\,I_{4x4}\right]\bar{q}_k \tag{3.2}$$

$$\Delta\theta^2 = \left(\omega_i^2 + \omega_j^2 + \omega_k^2\right)T_s^2$$

Portanto, tem-se que:

$$\hat{x} = \bar{q} \;\rightarrow\; \begin{bmatrix}\hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\end{bmatrix} = \begin{bmatrix}q_1\\ q_2\\ q_3\\ q_4\end{bmatrix} \tag{3.3}$$

$$A = \left[I_{4x4} + \frac{1}{2}\Omega(\boldsymbol{\omega})\,T_s - \frac{1}{8}\Delta\theta^2\,I_{4x4}\right] \tag{3.4}$$

É importante observar que o estado a ser estimado tem uma característica de transição **linear** com relação ao estado anterior, já que a matriz A tem apenas termos constantes para um dado passo k.

Outro ponto importante é que os dados do giroscópio são utilizados na matriz de estados e não no vetor de medições z do sistema. Desta forma, o vetor z é composto utilizando-se os dados do acelerômetro e magnetômetro.

Matriz B

Como não existem entradas de controle para o sistema, a matriz B é nula.

Matriz Q

Inicialmente considerada como mostrado na equação 3.5. Esta matriz deve ser ajustada como descrito em [3] e [31].

$$Q = \begin{bmatrix} 10^{-6} & 0 & 0 & 0 \\ 0 & 10^{-6} & 0 & 0 \\ 0 & 0 & 10^{-6} & 0 \\ 0 & 0 & 0 & 10^{-6} \end{bmatrix} \tag{3.5}$$

Matriz R

Inicialmente considerada como mostrado na equação 3.6. Esta matriz também deve ser ajustada de acordo com o comportamento dos dados dos sensores utilizados. Para este trabalho, utilizou-se como base [3] e [31].

$$R = \begin{bmatrix} 5.10^{-5} & 0 & 0 & 0 \\ 0 & 5.10^{-5} & 0 & 0 \\ 0 & 0 & 5.10^{-5} & 0 \\ 0 & 0 & 0 & 5.10^{-5} \end{bmatrix} \tag{3.6}$$

Matriz H

Levando em consideração que os sensores utilizados para o vetor de medições dão informação de aceleração e campo magnético, é necessário estabelecer relações destes valores com os estados do sistema, que são os quatro elementos do *quaternion* de rotação. Esta é uma tarefa bastante complicada e pouco intuitiva, figura 3.2. É possível obter um conjunto de equações que represente estas relações, porém a matriz H se tornaria não linear [32].

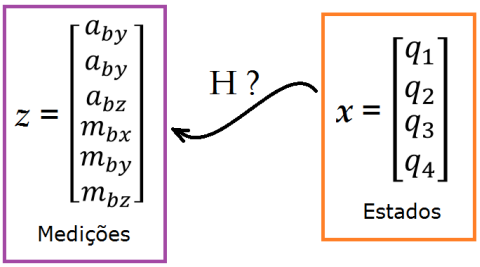


Figura 3.1: Como entrar com as medições no KFA.

Para que o KFA não perca a eficiência quando aplicado a *quaternion*s, é interessante obter um "*quaternion* medido" separadamente. Desta forma, ao invés de o vetor de medições z ser composto pelos dados dos sensores diretamente, como sugerido na figura 3.1, este deve ser composto pelos elementos de um *quaternion* calculado a partir dos dados dos sensores. Sendo assim, pode-se encontrar um *quaternion* de rotação que representará as medições do magnetômetro e acelerômentro no KFA. A principal vantagem desta abordagem é que a matriz H torna-se uma I_{4x4}, equação 3.7, uma vez que a relação entre os elementos dos quatérnios de estado e de medição é direta [1][2] [31] [3] [32].

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

3.1.2 Método de Gauss Newton

Para calcular o *quaternion* de medição a partir dos dados do acelerômetro e magnetômetro, pode-se buscar pontos de mínimo da equação de erro quadrático 3.8, onde ${}^{G}y$ é um vetor de seis posições com as leituras do acelerômetro e do magnetômetro na posição de referência (*global frame*) e ${}^{B}y$ é um vetor também de seis posições que contém as leituras atuais dos mesmos (*body frame*). A matriz M_{6x6}, equação 3.9, é composta por duas matrizes de rotação por *quaternion* ${}^{B}_{G}R_{3x3}$, equação 2.27, que levam os valores medidos do *global frame* para o *body frame*.

$${}^{B}Q = \varepsilon^{T}\varepsilon = ({}^{B}y - M\,{}^{G}y)^{T}({}^{B}y - M\,{}^{G}y) \tag{3.8}$$

$$M = \begin{bmatrix} {}^{B}_{G}R_{3x3} & 0 \\ 0 & {}^{B}_{G}R_{3x3} \end{bmatrix} \tag{3.9}$$

Tendo como base [2] [32], optou-se por utilizar o método numérico de **Gauss Newton** (GNA - *Gauss Newton Algorithm*), que permite calcular pontos de mínimo locais da equação de erro quadrático utilizando apenas derivadas de primeira ordem. Este algoritmo, apesar de possuir um raio de convergência limitado, e portanto dependente do chute inicial, converge em poucas iterações e com boa precisão.

Sendo assim, o vetor erro $\varepsilon(q_k)$ para o *quaternion* no passo k pode ser descrito como mostrado na equação 3.10. A matriz de derivadas, matriz Jacobiana, é dada por 3.11 e

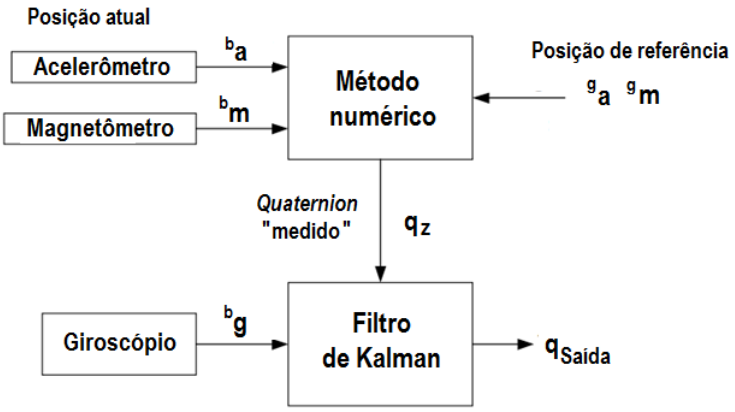


Figura 3.2: Diagrama de blocos da obtenção do *quaternion* medido e aplicação no KFA [32].

equação de iteração do método é dada por 3.12.

$$\varepsilon(q_k) = \begin{bmatrix} {}^{B}a_x \\ {}^{B}a_y \\ {}^{B}a_z \\ {}^{B}m_x \\ {}^{B}m_y \\ {}^{B}m_z \end{bmatrix} - \begin{bmatrix} \begin{bmatrix} {}^{B}_{G}R_{3x3} \end{bmatrix} & 0 \\ 0 & \begin{bmatrix} {}^{B}_{G}R_{3x3} \end{bmatrix} \end{bmatrix} \begin{bmatrix} {}^{G}a_x \\ {}^{G}a_y \\ {}^{G}a_z \\ {}^{G}m_x \\ {}^{G}m_y \\ {}^{G}m_z \end{bmatrix} \tag{3.10}$$

$$J(q_k) = \begin{bmatrix} \frac{\partial \varepsilon_1(q_k)}{\partial q1} & \cdots & \frac{\partial \varepsilon_1(q_k)}{\partial q4} \\ \frac{\partial \varepsilon_2(q_k)}{\partial q1} & & \vdots \\ \vdots & \ddots & \vdots \\ \frac{\partial \varepsilon_6(q_k)}{\partial q1} & \cdots & \frac{\partial \varepsilon_6(q_k)}{\partial q4} \end{bmatrix}_{6x4} \tag{3.11}$$

$$q_{k+1} = q_k - [(J(q_k)^T J(q_k))^{-1} J(q_k)^T \varepsilon(q_k)] \tag{3.12}$$

3.2 Módulo MPU9250

Escolheu-se para utilização no projeto o módulo MPU9250, da IvenSense, devido à várias características que atendem aos requisitos do projeto listados na seção 1.1, tais como: encapsulamento em um único chip, com pequenas dimensções, dos três sensores utilizados na aplicação; comunicação SPI com velocidade superior à I_2C e presença de FIFO, o que reduz o tempo de processamento; custo acessível em torno de R$30,00; liberdade de movimento, já que os três sensores são de três eixos; resolução da ordem de 10^{-2} e ainda à presença de

Figura 3.3: Módulo MPU9250 [33].

DMP (Digital Motion Processor) que pode ser utilizado para complementar os recursos do projeto.

O *pinout* do módulo utilizado é mostrado na figura 3.3.

3.2.1 Características

O MPU9250 é um módulo *multi-chip* composto por [34]

- giroscópio de 3-eixos com escalas de ±250, ±500, ±1000, and ±2000°/s e ADC de 16 bits (precisão mínima de 0.061°/s);
- acelerômetro de 3-eixos com escalas de ±2g, ±4g, ±8g e ±16g, g a aceleração da gravidade;
- magnetômetro de 3-eixos(AK8963) com escala de ±4800µT;
- sensor de temperatura;
- filtros digitais programáveis (passa-baixa);
- DMP (*Digital Motion Processor*) que realiza *Motion Fusion* e fornece informações de orientação em ângulos de Euler ou *quaternion*s;
- comunicação I_2C de 400kHz e SPI de 1MHz (os sensores e registradores de interrupção podem ser lidos em até 20MHz);

Tabela 3.1: Comandos para leitura e escrita via I_2C

Função	Comandos
Leitura $(data = I_2C\_readReg)$	$I_2C\_start(slaveAddress)$ I_2C_ack I_2C_stop $data = I_2C\_receive$
Escrita $(I_2C\_writeReg(data))$	I_2C_start I_2C_ack $I_2C\_send(data)$

- barramentos de comunicação principal (para comunicação com o *master*) e secundário (para leitura de até 4 sensores *slave* que podem ser conectados ao módulos);
- 3 conversores AD de 16 bits para cada um dos sensores;
- tensão de alimentação de 2.4V a 3.6V;
- FIFO (*first in first out buffer*) para automação das leituras de sensores;

3.2.2 Leitura e escrita de registradores

Para a configuração inicial do MPU9250 é necessário comunicar-se via I_2C. Para isto, foi desenvolvida uma biblioteca de comunicação, tendo como base a biblioteca *stm32f4xx_i2c* disponibilizada pela ST, que contém métodos para configuração de registradores para inicialização, geração de sinais de *start* (i_2c_start) e *stop* (i_2c_stop), sinal de *acknowledge* (i_2c_ack), configuração e gerenciamento de interrupções, e envio (i_2c_send) e recebimento ($i_2c\_receive$) de dados. Além da comunicação, a biblioteca desenvolvida também possui método para configuração dos pinos do ST32F407VG para utilização da I_2C.

A partir disto foi desenvolvida uma segunda biblioteca para escrita e leitura de registradores do módulo. A tabela 3.1 resume os comandos de leitura e escrita via I_2C e a tabela 3.2 resume a sequência de comandos para leitura e escrita de registradores do módulo.

O MPU9250 também permite realizar leitura e escrita de vários registradores em sequência, o que é uma boa opção no caso da obtenção das leituras do acelerômetro e giroscópio , acarretando na leitura de 12 registradores sem a necessidade de sinais de *stop* e *start*. Esta característica também é utilizada para a configuração do acelerômetro e giroscópio , onde

Tabela 3.2: Comandos para leitura e escrita em registradores do MPU9250

Função	Comandos
Leitura de registrador	$I_2C\_writeReg(address)$ $data = I_2C\_readReg$
Escrita em registrador	$I_2C\_writeReg(address)$ $I_2C\_writeReg(data)$

Tabela 3.3: Comandos para leitura e escrita do magnetômetro .

Função	Comandos
Leitura de registrador	$I_2C\_writeReg(address)$ $data = I_2C\_readReg$ $I_2C\_writeReg(status2Address)$ $data = I_2C\_readReg$
Escrita em registrador	$I_2C\_writeReg(mag\_address)$ $I_2C\_writeReg(data)$

escreve-se em quatro registradores em sequência.

Já a leitura e escrita do magnetômetro é realizada independentemente, uma vez que, por se tratar de um dispositivo que não compartilha da mesma pastilha que o acelerômetro e giroscópio , seu endereço para comunicação é diferente. A tabela 3.3 resume a sequência de comandos que deve ser realizada para escrita e leitura do magnetômetro .

Primeiramente é necessário habilitar no módulo o *bypass* do barramento I_2C principal, permitindo a conexão entre os pinos de dados (SDA) e *clock* (SCLK) e o barramento I_2C auxiliar, onde está conectado o magnetômetro . A figura 3.4 ilustra a organização e acesso aos blocos do módulo. É importante destacar que após a leitura dos registradores de medições é necessário ler também o registrador Status 2 do magnetômetro .

3.2.3 Inicialização

A inicialização do módulo deve conter, no mínimo, os seguintes passos:

- Confirmação do endereço para comunicação I_2C através da leitura do registrador WHO_AM_I;
- Ativação do módulo através da configuração do registrador PWR_MNGM_1;

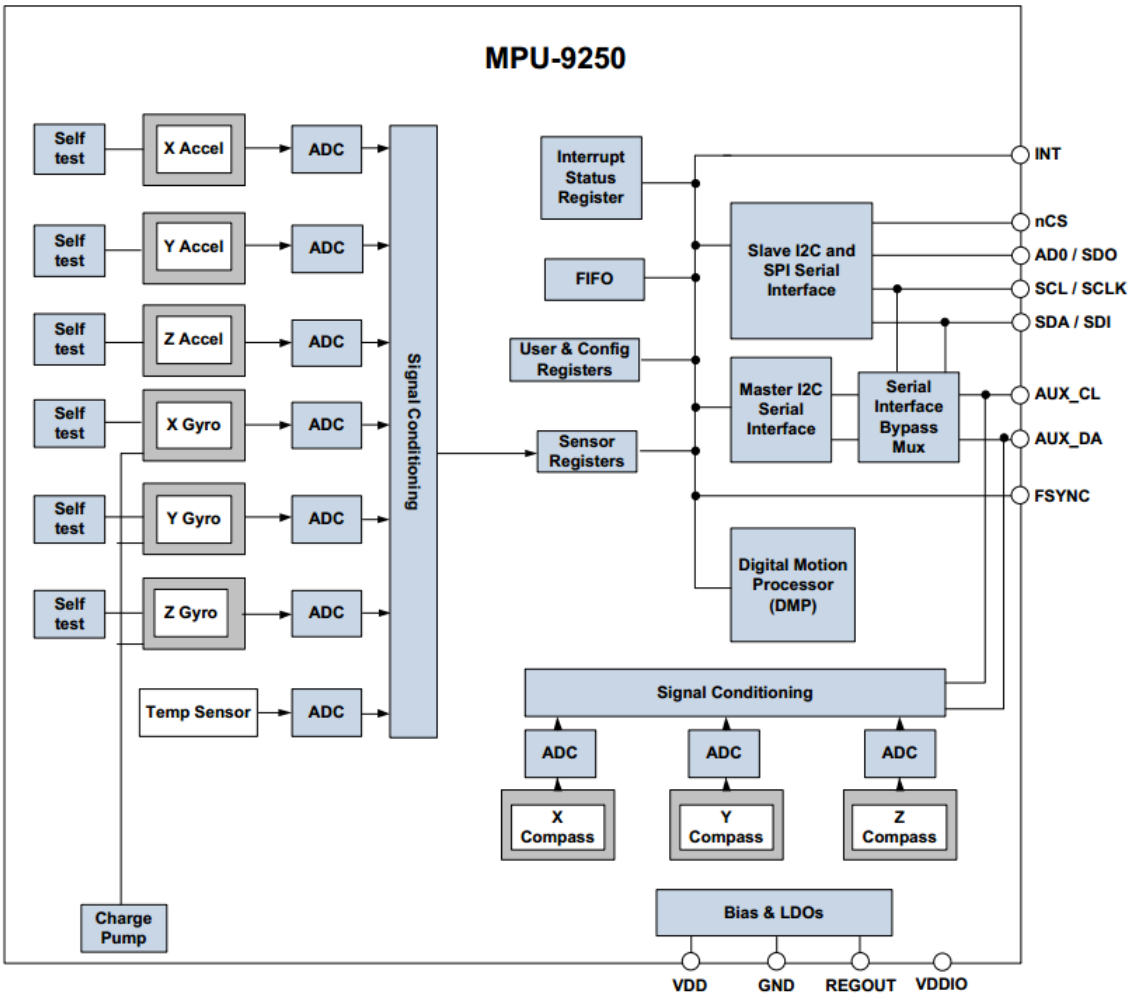


Figura 3.4: Acesso ao barramento I_2C auxiliar para comunicação com magnetômetro [35].

- Configuração de escalas e frequência de corte do filtro passa baixa para acelerômetro e giroscópio ;
- Configuração da resolução desejada pra o magnetômetro (14 ou 16bits);

Ao inicializar o módulo é necessário aguardar por no mínimo 35ms, tempo de inicialização do giroscópio (acelerômetro e magnetômetro tem tempos menores).

3.2.4 Conversão dos dados

A conversão dos dados lidos para valores físicos é feita de acordo com a equação 3.13, onde S_{sens} é a sensibilidade da escala utilizada, com unidades:

- $[^\circ]/s/LSB$ para o giroscópio;
- g/LSB para o acelerômetro;
- $\mu T/LSB$ para o magnetômetro;

A sensibilidade pode ser calculada dividindo-se o valor máximo da escala utilizada (S_{max}) pelo valor máximo do conversor AD (16bits), considerando 1 bit de sinal.

$$x_{phy} = \frac{x_{meas} S_{max}}{32767} = x_{meas} S_{sens} \tag{3.13}$$

A partir das leituras do giroscópio, pode-se também obter informação de ângulo, realizando-se uma integração discreta. A forma mais simples é utilizando interpolação linear, onde a leitura de velocidade angular no instante n (ω_n) é considerada constante durante o período de amostragem T_s, como mostrado na equação 3.14.

$$\alpha_k = \sum_{n=1}^{k} \omega_n T_s \tag{3.14}$$

3.2.5 Calibração dos sensores

Giroscópio

No caso do giroscópio, o ajuste de *offset* é feito sempre que o sistema é iniciado. Para isso, com a placa em repouso, são adquiridas 50 amostras de velocidade angular e posteriormente calcula-se a média entre a maior e a menor leituras obtidas, evitando-se o cálculo com valores muito grandes como no caso da utilização de média de todas as leituras. A média resultante

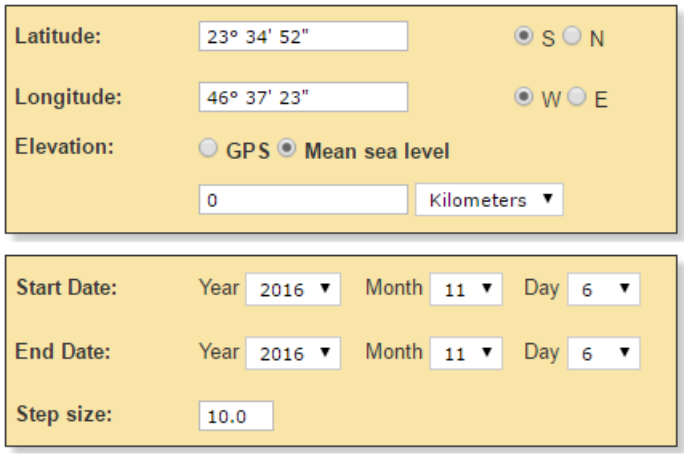
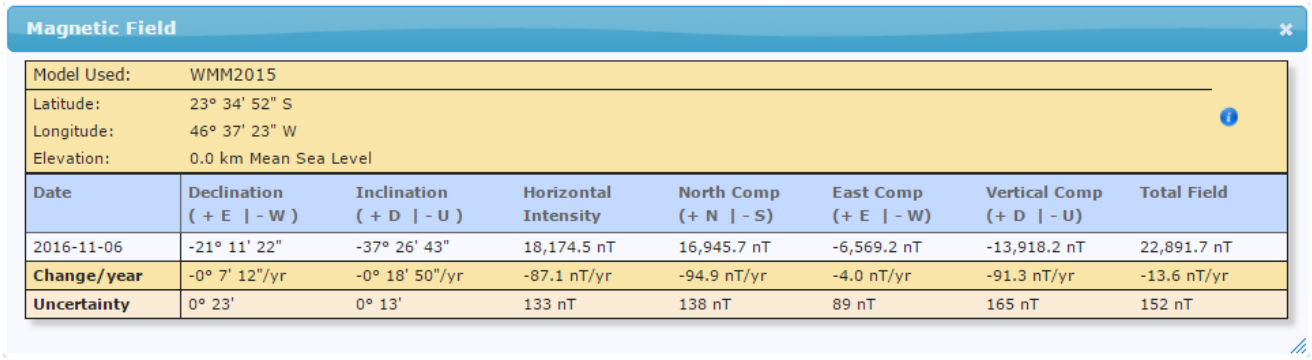


Figura 3.5: Dados de entrada para cálculo do campo magnético da Terra na cidade de São Paulo - SP [36].



Magnetic Field

Model Used:	WMM2015
Latitude:	23° 34' 52" S
Longitude:	46° 37' 23" W
Elevation:	0.0 km Mean Sea Level

Date	Declination (+ E | - W)	Inclination (+ D | - U)	Horizontal Intensity	North Comp (+ N | - S)	East Comp (+ E | - W)	Vertical Comp (+ D | - U)	Total Field
2016-11-06	-21° 11' 22"	-37° 26' 43"	18,174.5 nT	16,945.7 nT	-6,569.2 nT	-13,918.2 nT	22,891.7 nT
Change/year	-0° 7' 12"/yr	-0° 18' 50"/yr	-87.1 nT/yr	-94.9 nT/yr	-4.0 nT/yr	-91.3 nT/yr	-13.6 nT/yr
Uncertainty	0° 23'	0° 13'	133 nT	138 nT	89 nT	165 nT	152 nT

Figura 3.6: Campo magnético da Terra calculado na cidade de São Paulo - SP [36].

é então subtraída das medições futuras, uma vez que deseja-se obter velocidade angular nula em repouso.

$$m_{Offset} = \frac{m_{max} + m_{min}}{2} \tag{3.15}$$

Magnetômetro

Já para o magnetômetro, tanto o ajuste de *offset* quanto de escala devem ser feitos de tal forma que as medições estejam contidas em uma em uma esfera de raio igual ao campo magnético da Terra em uma dada localização. Existem diversas ferramentas disponíveis que permitem calcular o campo magnético da Terra, sua inclinação (ângulo com relação ao plano horizontal) e declinação (ângulo com relação ao norte verdadeiro), figura 3.7. Em [36] pode-se obter estas informações através de uma ferramenta do governo norte americano que utiliza o mais recente *WMM* (*World Magnetic Model*). Os valores obtidos podem ser vistos na figura 3.8.

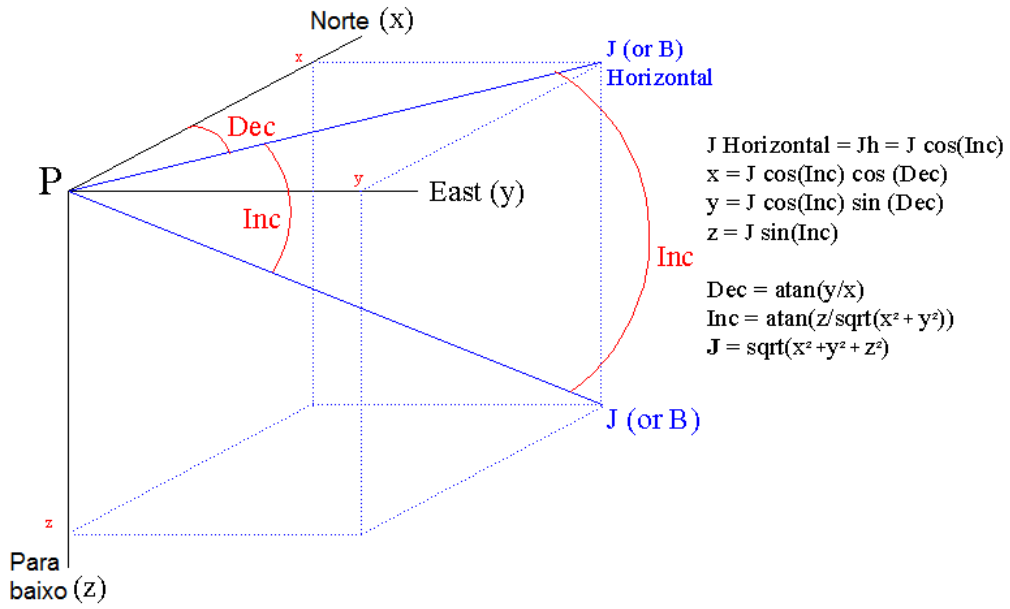


Figura 3.7: Ângulo de de inclinação e declinação [37].

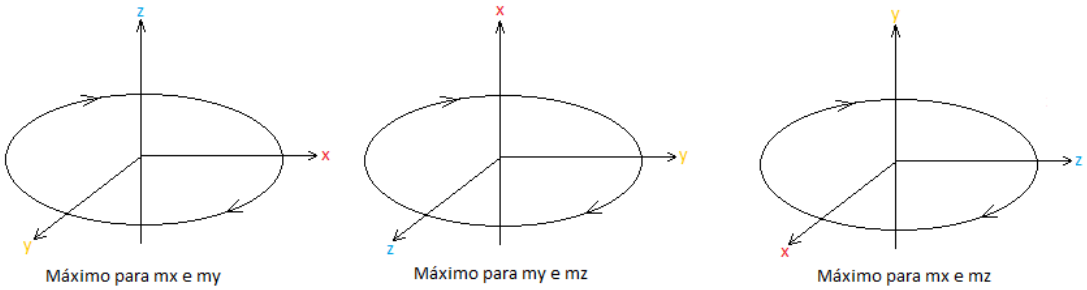


Figura 3.8: Rotações para calibração do magnetômetro.

O cálculo do *offset* e da escala para o magnetomêtro foi realizado separadamente à rotina do KFA. Foram adquiridas medições rotacionando a placa em 360° em torno de cada eixo, individualmente, mantendo-o alinhado com a vertical. A partir das medições obtidas, pode-se encontrar os valores máximo e mínimo para cada eixo. Primeiramente, foi feito o ajuste de *offset* como mostrado na equação 3.15. E então, realizou-se o o cálculo da escala (s_{adj}), dividindo-se $18.17\mu T$, máxima componente horizontal esperada, pelos valores medidos já sem *offset*. Por fim, o ajuste das medições pode ser feito como mostrado na equação 3.16.

$$m_{calib} = s_{adj}\left(m_{raw} - m_{offset}\right) \tag{3.16}$$

3.2.6 Escalas e filtro passa-baixa

Giroscópio

FCHOICE		DLPF_CFG	Gyroscope			Temperature Sensor	
<1>	<0>		Bandwidth (Hz)	Delay (ms)	Fs (kHz)	Bandwidth (Hz)	Delay (ms)
x	0	x	8800	0.064	32	4000	0.04
0	1	x	3600	0.11	32	4000	0.04
1	1	0	250	0.97	8	4000	0.04
1	1	1	184	2.9	1	188	1.9
1	1	2	92	3.9	1	98	2.8
1	1	3	41	5.9	1	42	4.8
1	1	4	20	9.9	1	20	8.3
1	1	5	10	17.85	1	10	13.4
1	1	6	5	33.48	1	5	18.6
1	1	7	3600	0.17	8	4000	0.04

Figura 3.9: Tempo entre medições do giroscópio com relação a frequência de corte do filtro passa-baixa do MPU9250 [35].

Para observar os valores de máxima velocidade angular e máxima frequência (máxima variação de velocidade angular) e, assim, poder definir a melhor frequência de corte para filtro passa-baixa do MPU9250 para que não houvesse perda de informação, foram obtidas medições do giroscópio durante a realização de gestos do usuário, segurando-se a placa na mão direita (usuário destro) e rotacionando-se rapidamente (máxima velocidade conseguida pelo usuário) o punho nos sentidos horário e antihorário, na direção do antebraço. O sensor foi configurado com frequências de corte de 184Hz, 41Hz e 5Hz e os resultados obtidos são mostrados na seção 4.2.2.

É importante ressaltar que quanto menor a frequência de corte utilizada, maior deve ser o *delay* entre novas aquisições, como mostrado na figura 3.9, obtida do mapa de registradores do módulo [35]. Portanto, deve-se ponderar entre melhor redução de oscilações nas medidas e tempo de processamento do algoritmo.

Acelerômetro

Para este trabalho, a máxima aceleração a ser medida é a própria aceleração da gravidade, 1g, uma vez que para gestos do usuário as acelerações lineares percebidas são insignificantes. Portanto, optando-se por uma escala maior , de +-8g, pode-se reduzir a sensibilidade por LSB do sensor e reduzir oscilações indesejadas.

Também para o acelerômetro é importante observar o *delay* entre aquisições para cada frequência de corte do filtro passa-baixa, mostrado na figura 3.10.

Accelerometer Data Rates and Bandwidths (Normal Mode)

ACCEL_FCHOICE	A_DLPF_CFG	Output			
		Bandwidth (Hz)	Delay (ms)	Noise Density (ug/rtHz)	Rate (kHz)
0	X	1.13 K	0.75	250	4
1	0	460	1.94	250	1
1	1	184	5.80	250	1
1	2	92	7.80	250	1
1	3	41	11.80	250	1
1	4	20	19.80	250	1
1	5	10	35.70	250	1
1	6	5	66.96	250	1
1	7	460	1.94	250	1

Figura 3.10: Tempo entre medições do acelerômetro com relação a frequência de corte do filtro passa-baixa do MPU9250 [35].

3.2.7 Alinhamento dos eixos e sentido de rotação

Para a convergência do GNA e do KF, é de fundamental importância garantir que os eixos dos sensores utilizados estejam alinhados corretamente e, além disso, que o sentido positivo de rotação do giroscópio esteja em acordo com o sentido de rotação e notação de *quaternion* adotados para este trabalho. Tendo como base a figura 3.11 e o sistema de coordenadas descrito na seção 2.3.4, é possível observar algumas modificações necessárias, descritas a seguir:

- Giroscópio: troca dos eixos x e y e inversão do sentido de rotação;
- Acelerômetro: troca dos eixos x e y;
- Magnetômetro: inversão do sentido positivo do eixo z;

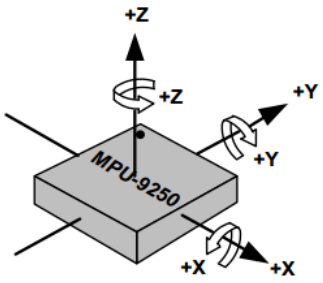


Figure 4. Orientation of Axes of Sensitivity and Polarity of Rotation for Accelerometer and Gyroscope

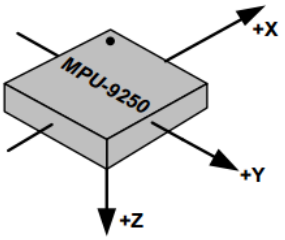


Figure 5. Orientation of Axes of Sensitivity for Compass

Figura 3.11: Alinhamento dos eixos dos sensores no MPU9250 [34].

3.3 Placa de desenvolvimento STM32F4Discovery

Para implementação do projeto, utilizou-se a placa de desenvolvimento STM32F4 Discovery, com as seguintes caracteríscticas [38].

- microcontrolador ST32F407VG ARM Cortex -M4 [39] :
  - FPU (Floatinf Point Unit);
  - 1Mbyte de memória *flash*;
  - 192 Kbyte de RAM;
  - clock de 168MHz;
- programador e *in-circuit debugger* ST-Link;
- porta COM virtual;

O *pinout* da placa é mostrado na figura 3.3. Utilizou-se a IDE CooCox CoIDE, gratuita e com suporte para a família ARM Cortex -M4.

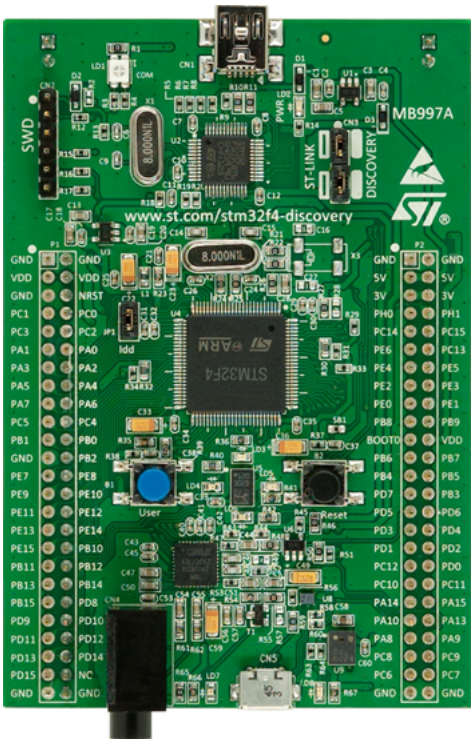


Figura 3.12: Placa de desenvolvimento utilizada [40].

3.4 Implementação

O algoritmo completo é composto pelas etapas enumeradas a seguir. A figura 3.13 ilustra o fluxo da rotina criada.

- Inicialização, configuração e obtenção das medições de referência dos sensores;
- Obtenção de medições dos sensores no *body frame* (comunicação I_2C);
- Cálculo do *quaternion* de medições pelo GNA;
- Cálculo do *quaternion* de saída pelo KFA;
- Envio dos dados de processamento para visualização no *desktop*, via módulo bluetooth (comunicação UART);

Para a implementação, utilizou-se linguagem C e compilador GCC C/C++ e GDB-based *debugger*, já presentes na CooCox CoIDE.

Os princípios de modularidade, rigor e formalidade e abstração da Engenharia de Software foram considerados durante o desenvolvimento, de tal forma que se obtivesse um padrão na nomenclatura de variáveis e criação de métodos (padronização de argumentos e retornos),

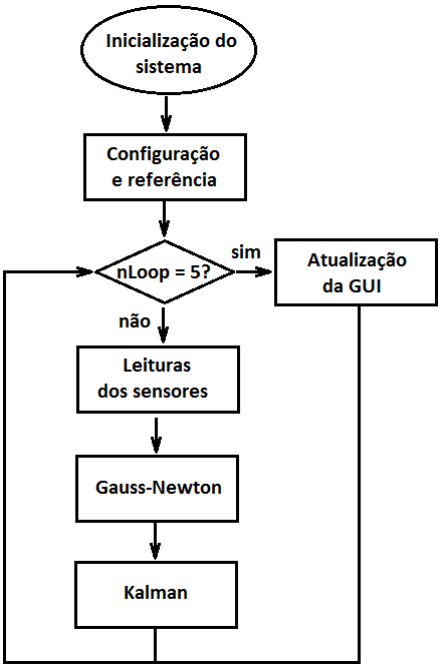


Figura 3.13: Fluxograma da rotina de estimação de orientação.

facilidade para encapsulamento destes e para realização de testes. Mais detalhes referentes a implementação de cada etapa estão descritas nas seções a seguir.

3.4.1 Biblioteca para MPU9250

Juntamente com os métodos de escrita e leitura de registradores, descritas na seção 3.2.2, a biblioteca desenvolvida para o módulo também contém métodos para leitura dos dados dos sensores (individuais ou em conjunto) com e sem aplicação de média, configuração e inicialização do módulo, e conversão das leituras para grandezas físicas.

Criou-se uma *struct MPU9250_t* contendo variáveis de interesse para o módulo utilizado. Os métodos da biblioteca sempre recebem uma variável desta *struct*. Em alguns métodos, tem-se também outros argumentos como número de amostras para cálculo da média ou valores de configurações desejadas. As principais configurações e os respectivos registradores são definidos no arquivo .h da biblioteca com nomes auto-explicativos, o que facilita a utilização.

Leituras de referência

Para a obtenção das leituras de referência dos sensores, são realizadas dez aquisições em sequência, com intervalo de 10ms entre cada uma, e então calcula-se a média entre os valores de máximo e mínimo obtidos.

Leituras de body frame

As leituras seguintes são utilizadas diretamente, sem aplicação de média, uma vez que já se utiliza o filtro passa baixa configurado com 41Hz de frequência de corte para o acelerômetro e giroscópio . Ao magnetômetro não há aplicação de filtro.

3.4.2 Bibliotecas para Gauss Newton e Kalman

A implementação de ambos os algoritmos foi fortemente embasada no conjunto de métodos da biblioteca CMSIS/DSP_Lib para cálculo matricial (MatrixFunctions) disponibilizado pela própria ST, que faz usp do FPU (*Floating Point Unit*) do microcontrolador. Os métodos incluem cálculo de adição, subtração, multiplicação, inversão, transposição e operações com números complexos.

Nesta biblioteca, uma matriz é representada por uma *struct* (arm_matrix_instance_f32) com os seguintes campos:

- número de linhas ($uint16\_t$);
- número de colunas ($uint16\_t$);
- ponteiro para o vetor de dados ($float32\_t*$);

Para utilizar uma variável deste tipo, é necessário inicializá-la enviando o número de linhas, de colunas e o ponteiro para o vetor de dados. Por isso, para toda matriz, deve-se criar também um vetor de $float32\_t*$. Para a utilização da maioria dos métodos de operações, deve-se enviar como argumento as matrizes a serem operadas e a matriz onde se deseja armazenar o resultado. Obtém-se como retorno um *typedef* de status da operação.

Foram desenvolvidas duas bibliotecas, uma para cada método, cada qual contendo uma *struct* com variáveis de interesse e um conjunto de matrizes utilizadas, como descrito nas seções 3.1.1 e 3.1.2. Além disso, cada biblioteca também contém um método para inicialização das matrizes, um método para cálculo de módulo de vetor e um método com o algoritmo propriamente dito.

O critério de parada utilizado para o GNA diz respeito ao valor de ajuste calculado para a próxima iteração do método. Assim, caso o ajuste entre os *quaternion*s k e $k+1$ seja $< 10^{-2}$, então o *loop* é interrompido. O máximo número de iterações utilizado foi de 15, uma vez que após isso o resultado não varia e não há redução do erro.

Tabela 3.4: Tabela de medições ideais para rotação em torno do eixo x

Ângulo em x [rad]	Acelerômetro [g]	Magnetômetro [$\mu T/18.174$]	Giroscópio [rad/s]
Referência	[0, 0, 1]	[0, 1, 0]	[3.927, 0, 0]
$\frac{\pi}{4}$	[0, 0.707, 0.707]	[0, 0.707, -0.707]	[3.927, 0, 0]
$\frac{\pi}{2}$	[0, 0.999, 0]	[0, 0, -0.999]	[3.927, 0, 0]
$\frac{3\pi}{4}$	[0, 0.707, -0.707]	[0, -0.707, -0.707]	[3.927, 0, 0]
π	[0, -8e-8, -1]	[0, -1, -8e-8]	[3.927, 0, 0]
$\frac{5\pi}{4}$	[0, -0.707, -0.707]	[0, -0.707, 0.707]	[3.927, 0, 0]
$\frac{3\pi}{2}$	[0, -0.707, 0.707]	[0, 0.707, 0.707]	[3.927, 0, 0]
2π	[0, 1.75e-7, 1]	[0, 1, 1.75e-7]	[3.927, 0, 0]

3.4.3 Teste de algoritmo

Para garantir a correta implementação do sistema como um todo, foram realizados testes com valores conhecidos e ideais de medições, simulando rotações de $\frac{\pi}{4} rad$ (sentido horário positivo) em torno do eixo x, como mostrado na tabela 3.4.

Os valores das medições ideias foram obtidos através da rotação por *quaternion* dos valores de referência, escolhidos arbitrariamente para o magnetômetro e acelerômetro . Tais valores equivalem ao posicionamento da placa de forma paralela ao plano horizontal, com eixo y apontando para o ponto de máximo da componente horizontal do campo magnético da Terra e z com valor máximo da gravidade. O valor ideal para o giroscópio foi obtido através da equação 3.17, para cada eixo.

$$G = \frac{\triangle \theta}{Ts} \tag{3.17}$$

sendo Ts o período de amostragem, igual ao tempo de rotina do código.

A comparação entre as saídas esperadas e obtidas tanto para GNA quanto para KFA pode ser vista na seção 4.

3.5 Aplicação auxiliar de desenvolvimento

Para melhor acompanhar o funcionamento das várias etapas do sistema proposto, houve a necessidade de desenvolvimento de uma ferramenta para visualização das leituras dos sensores e resultados dos algoritmos utilizados em tempo real. Outro objetivo desejado para a

ferramenta auxiliar seria a representação gráfica do deslocamento angular da placa de desenvolvimento.

3.5.1 Windows Forms

Optou-se, primeiramente, por implementar o algoritmo completo em Matlab para realização de testes preliminares e ajustes necessários. Desta forma também poderiam ser utilizados recursos gráficos de maneira simples, porém com uma experiência não tão boa para o usuário e dependência da instalação do *software* na máquina a ser utilizada.

Posteriormente, devido a facilidade de implementação, ao conhecimento básico prévio do assunto e, principalmente, a portabilidade, optou-se por utilizar a biblioteca de classes gráficas *Windows Forms*, em linguagem C# , que provê uma plataforma de desenvolvimento para aplicações com GUI (*Graphical User Interface*) de maneira direta e com resultados bastante complexos. Utilizou-se a IDE Microsoft Visual Studio Community 2013, de distribuição gratuita. Outra grande vantagem da utilização do *Windows Forms* é a vasta quantidade de material disponível sobre o assunto.

Para a renderização gráfica 3D, optou-se por utilizar a API OpenGL, também bastante difundida e de rápido processamento. A versão em C# da API apresenta relativa facilidade de implementação e possibilita desenvolver projeções gráficas de alta complexidade e a importação de modelos (vértices) em vários formatos, o que permite ainda fazer a integração com *softwares* de modelagem 3D, como Blender, também gratuito.

3.5.2 Comunicação com o sistema embarcado

Para a comunicação entre a aplicação Windows e o sistema embarcado, optou-se pela utilização do módulo Bluetooth, devido principalmente à facilidade de implementação em ambos os lados da comunicação, pela quantidade reduzida de dados a serem transmitidos nesta primeira etapa do projeto, pela característica de curtas distâncias (WPAN - *wireless personal area network*) e por atender satisfatoriamente as necessidades de velocidade de transmissão.

Através da porta COM emulada no sistema operacional pelo *driver* RFCOMM Protocol TDI, o qual implementa a camada de protocolo Bluetooth RFCOMM [https://msdn.microsoft.com/en-us/library/aa939805(v=winembedded.5).aspx], pode-se comunicar com o dispositivo Bluetooth de forma serial e sem a necessidade de implementação da pilha de comunicação, configurando-se apenas o *Baud Rate*, tamanho da palavra e bits de paridade e de parada.

```
void USART_sendFloat_word8(float32_t *fdata){
    uint8_t *data, size;

    data = (uint8_t*)fdata;
    size = sizeof(data)/sizeof(uint8_t);

    for(uint8_t i=0; i<size; i++){
        while(USART_GetFlagStatus(USART6, USART_FLAG_TXE) == RESET);
        USART_SendData(USART6, (uint16_t)(*(data+i)));
    }
}
```

Figura 3.14: Envio de dados *float* pela UART.

A biblioteca de classes *System.IO*, para .NET Framework, possibilita tais configurações, a transmissão/recepção de dados, definição de tamanho de *buffer* e ainda a utilização de interrupções.

Outro importante ponto com relação à comunicação entre plataforma embarcada e *desktop* diz respeito à transmissão de dados tipo *float* via serial. A técnica utilizada para isso é descrita no trecho de código da figura 3.14. Desta forma, o ponteiro para *uint8_t* aponta para o endereço do *float* a ser transmitido e seu conteúdo é desconhecido. Cada *float* a ser enviado resulta em um vetor de 4*bytes*.

Na aplicação Windows é necessário montar novamente o conjunto de 4 *bytes* em um *float*. Para tanto, a biblioteca utilizada possui uma função pronta para tal tarefa, *System.BitConverter.ToSingle(buffer, offset)*, bastando apenas inserir como argumento o vetor de *bytes* e o *offset* desejado. Foram realizados vários testes de transmissão com valores de tipo *float* conhecidos para garantir o correto funcionamento da comunicação.

Foi desenvolvida uma biblioteca para comunicação UART, embasada na biblioteca *stm32f4xx_usart* da própria ST, com métodos de escrita e leitura de *byte*, múltiplos *bytes*, *float* e múltiplos *float* e configuração dos pinos do ST32F407VG .

3.5.3 Módulo Bluetooth HC05

O módulo Bluetooth utilizado no projeto, como descrito na seção 3.5.2, foi o HC05, devido tanto ao custo acessível quanto pela facilidade de utilização. O *pinout* da placa pode ser visto na figura 3.15.

Tem-se as seguintes características [42]:

- Interface UART com BAUD programável;
- Antena integrada;

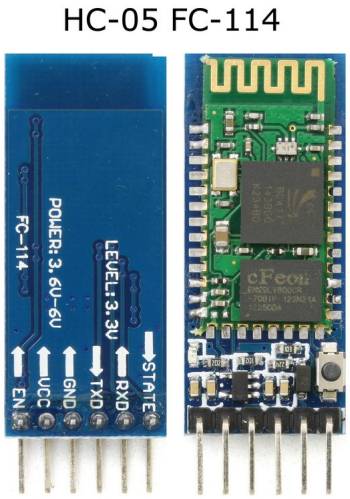


Figura 3.15: Módulo Bluetooth HC05 [41].

- Auto-conexão com o útimo dispositivo;
- Firmware para utilização de comandos AT;

Capítulo 4

Resultados e discussões

4.1 Aplicação auxiliar

A interface de usuário da aplicação desenvolvida pode ser vista na figura 4.1. Nela estão presentes:

1. gráficos das medições dos sensores;
2. gráfico dos ângulos Yaw, Pitch e Roll resultantes;
3. gráfico dos ângulos obtidos pela integração direta do giroscópio;
4. valores numéricos das medições dos sensores;
5. valores numéricos do *quaternion* resultante do algoritmo;
6. valores numéricos dos ângulos Yaw, Pitch e Roll resultantes;
7. quantidade de iterações para convergência do GNA (limitado em 15);
8. valor de ajuste da última iteração do GNA;
9. tempos de processamento tanto para o microcontrolador quanto para o sistema operacional;
10. botões para iniciar e parar a transferência de dados e para abrir a janela de renderização 3D;

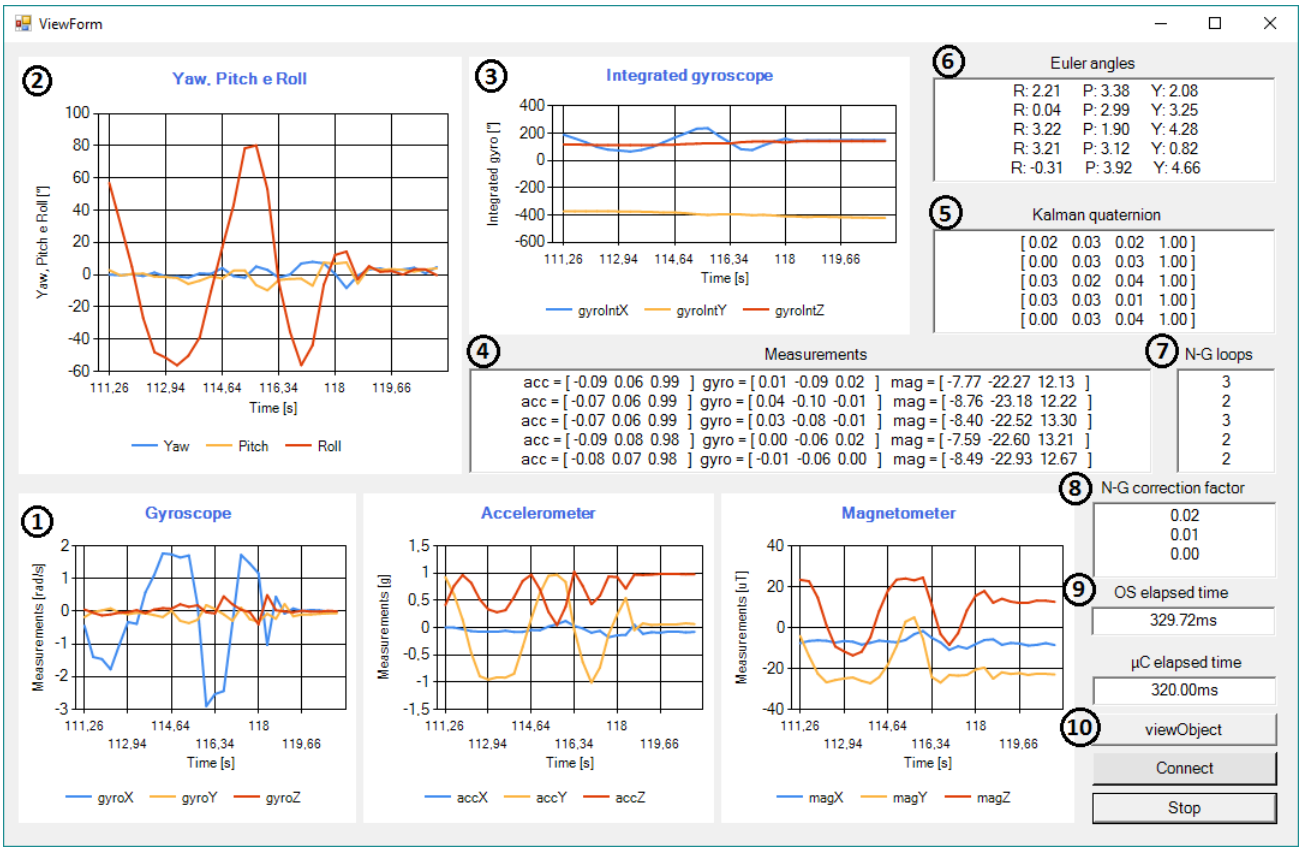


Figura 4.1: Interface da aplicação Windows desenvolvida.

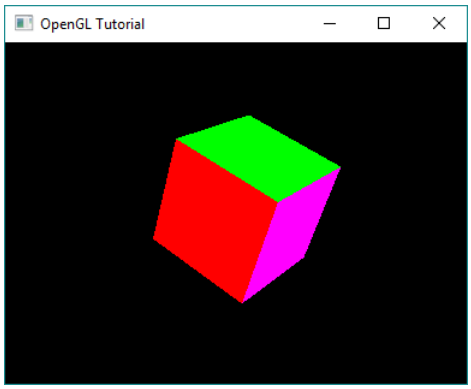


Figura 4.2: Janela de renderização criada com OpenGL.

Tabela 4.1: Tabela de medições obtidas em repouso, após a calibração dos sensores

	Giroscópio [rad/s]	Magnetômetro [μT]	Acelerômetro [g]
Máximo	[-0.03, 0.06, 0.02]	[-14.60, 5.56, 18.26]	[0.01, 0.02, 1]
Mínimo	[-0.01, 0.03, 0.01]	[-12.87, 3.28, 15.79]	[0, 0.01, 0.98]
Médio	[-0.02, 0.045, 0.015]	[-13.73, 4.42, 17.03]	[0.005, 0.015, 0.99]
Esperado	[0, 0, 0]	[_ , _, 13.92]	[0, 0, 1]

4.2 Módulo MPU9250

4.2.1 Calibração dos sensores

As figuras 4.3 e 4.4 mostram as leituras obtidas dos sensores com a placa em repouso, posicionada paralelamente ao plano horizontal, e rotacionando em torno do eixo z, respectivamente, após a calibração.

Para a primeira situação, é possível observar que as leituras do giroscópio estão bastante próximas de zero, sofrendo pequenas oscilações devido à sensibilidade do sensor. As leituras do acelerômetro estão coerentes, indicando aceleração de 1g no eixo z e aproximadamente zero em x e y. Já as leituras do magnetômetro também foram coerentes com o esperado. O valor obtido para o eixo z foi cerca de 15% maior que a componente vertical do campo magnético da Terra, de $13.92\mu T$. Os valores nos eixos x e y foram inferiores à componente horizontal do mesmo, sendo o vetor resultante de aproximadamente $14\mu T$, um pouco distante do esperado $18,17\mu T$. A tabela 4.1 mostra os valores de máximo e mínimo obtidos nesta situação.

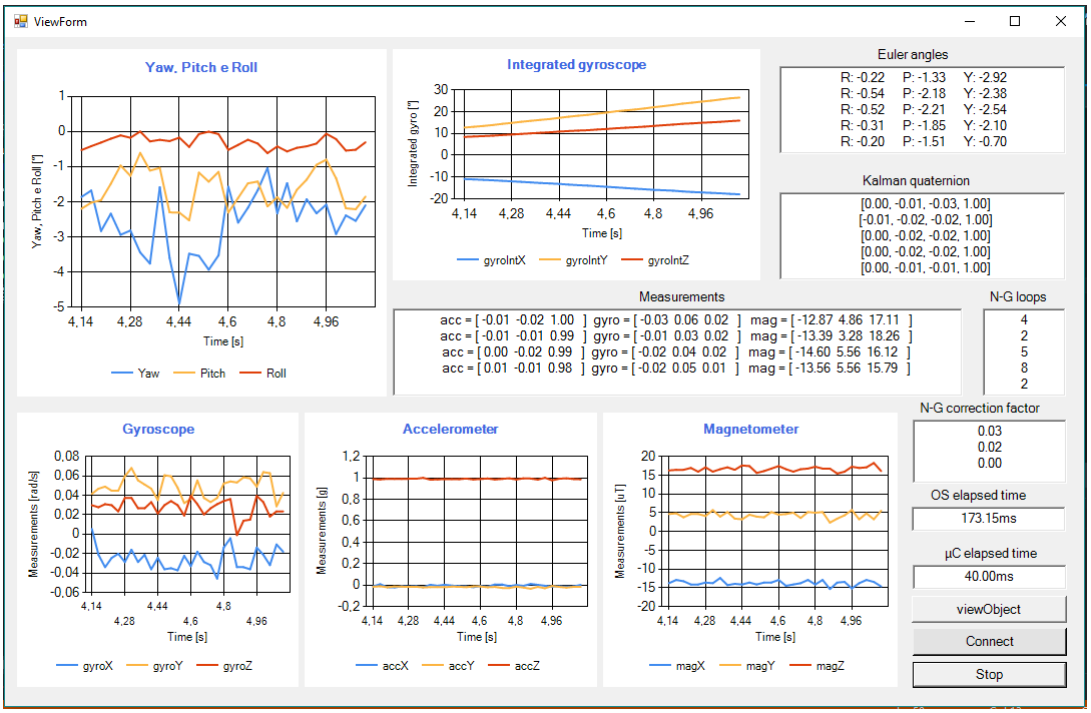


Figura 4.3: Leituras obtidas em repouso após calibração.

Com relação à segunda situação, obteve-se velocidades angulares maiores que zero, positivas para rotação no sentido horário (observar o campo dos ângulos yaw, pitch e roll na tela da ferramenta de *debug*) e negativas para anti horário, o que corresponde ao esperado de acordo com o sistema de coordenadas adotado.

O acelerômetro apresentou pequenas variações de leitura em x e y devido à movimentação da placa e sensibilidade do sensor. Em z, manteve-se 1g como esperado. Observa-se da figura 4.4 que a aceleração linear lida não é desprezível como inicialmente considerado, atingindo valores de até $\pm 0.5g$ mesmo com movimentos suaves da placa. Esta característica pode ter influência sobre a convergência do sistema e deve ser reavaliada.

Para o magnetômetro, a componente z superou os valores esperados, atingindo até $20\mu T$. As componentes x e y mantiveram-se menores que $18\mu T$, como esperado para o plano horizontal.

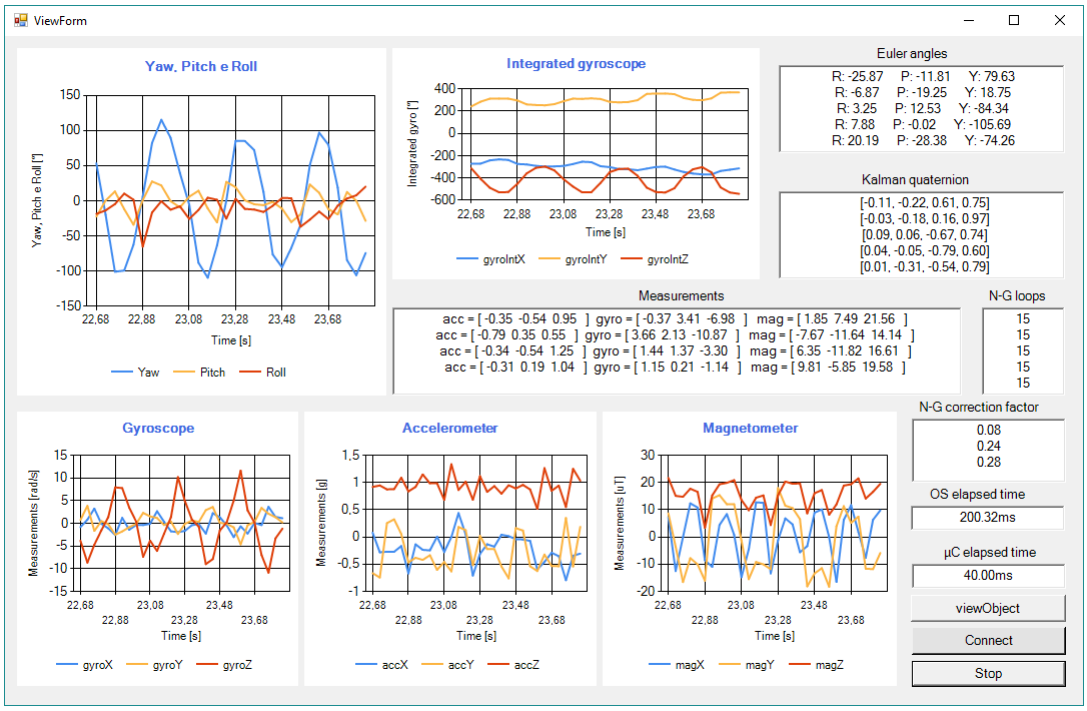


Figura 4.4: Leituras obtidas rotacionando em torno de z após calibração.

4.2.2 Escala e filtro passa-baixa

Os resultados obtidos para máxima velocidade e máxima variação de velocidade de gesto, medidas com o giroscópio configurado como descrito na seção 3.2 são mostrados nas figuras 4.5, 4.6 e 4.7.

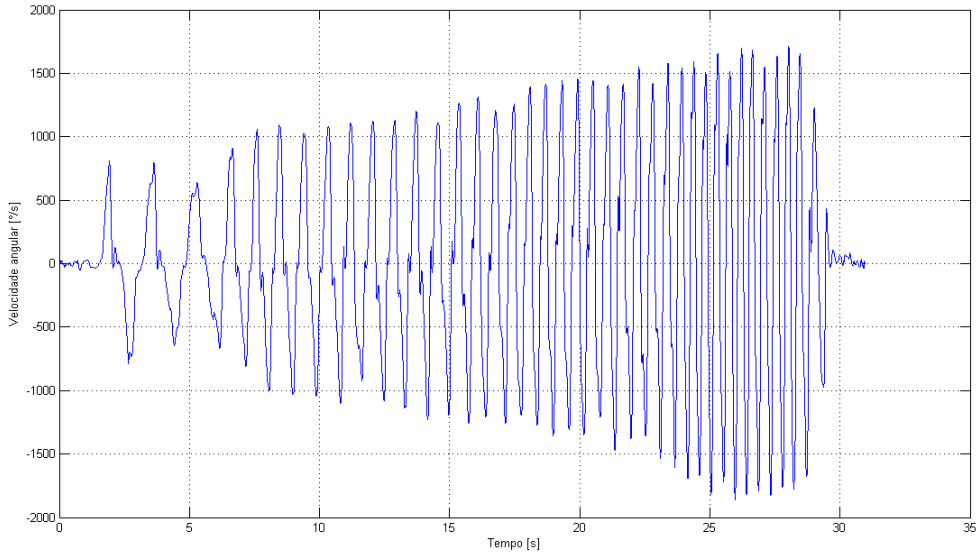


Figura 4.5: Dados do eixo y do giroscópio configurado com frequência de corte de 184Hz.

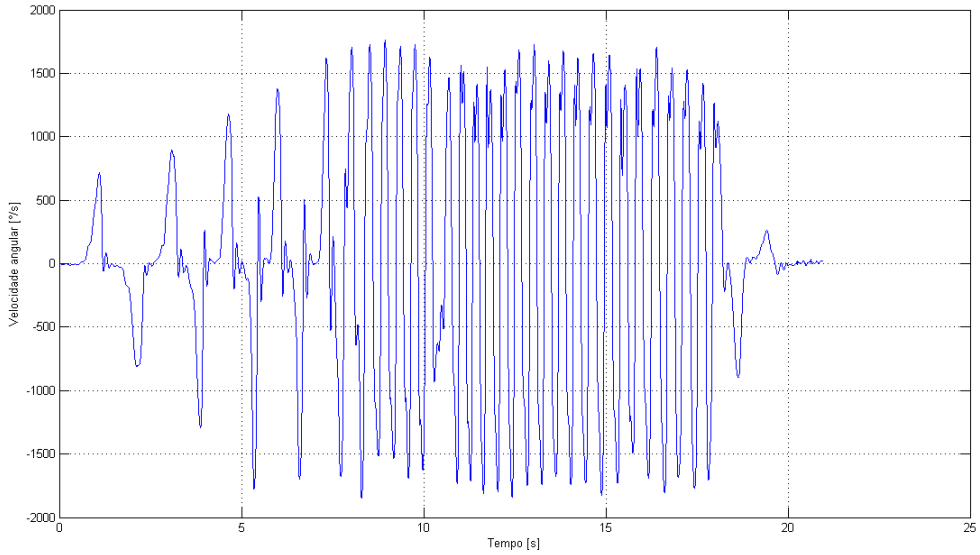


Figura 4.6: Dados do eixo y do giroscópio configurado com frequência de corte de 41Hz.

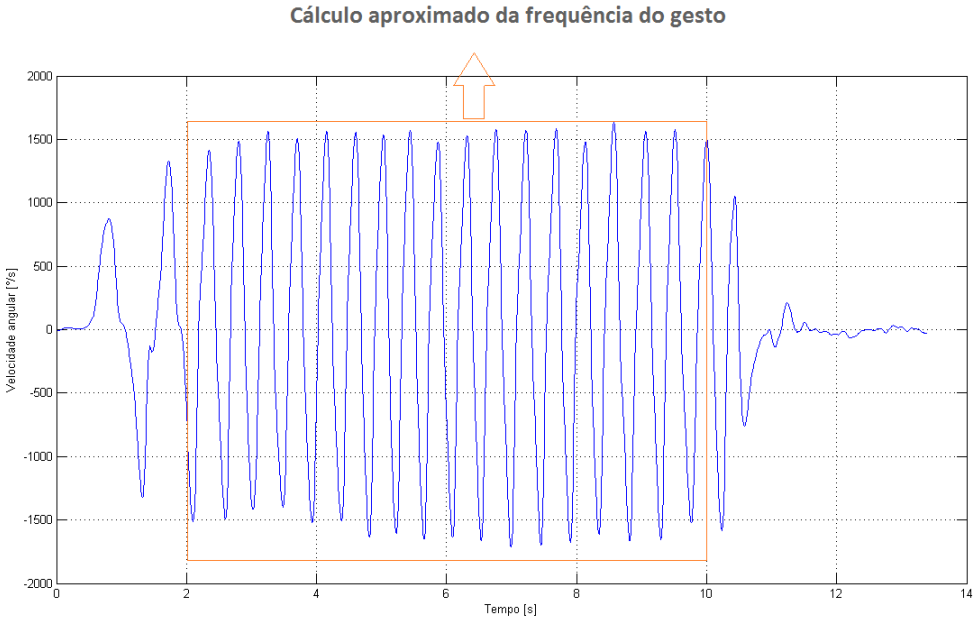


Figura 4.7: Dados do eixo y do giroscópio configurado com frequência de corte de 5Hz.

Pode-se inferir das figuras 4.5, 4.6 e 4.7 que a máxima frequência do gesto realizado fica em torno de 2 a 2,4Hz. Vale ressaltar que estas frequências foram atingidas com demasiado esforço do usuário e que, portanto, não seriam atingidas no caso ideal de utilização do dispositivo proposto.

Também é possível observar que a taxa de variação da velocidade angular sofre grandes alterações principalmente quando a frequência de corte do filtro passa-baixa é mais elevada, figuras 4.5, onde aparecem oscilações de alta frequência somadas à oscilação principal do gesto. Isto ocorre devido à excessiva sensibilidade do sensor que capta mínimas trepidações e gera leituras abruptas. A redução da frequência de corte ameniza este comportamento indesejado, como na figura 4.7.

Assim, pode-se concluir que com uma frequência de corte de 5Hz as leituras obtidas tem uma melhor relação sinal-ruído e não há perda de informação, já que tanto a frequência de movimento quanto a velocidade angular máxima não são alteradas. Porém, além da redução do ruído, existe a preocupação com o *delay* para obtenção de novas leituras que pode afetar drasticamente o resultado final do sistema. Optou-se portanto, por utilizar a frequência de corte de 41Hz, que apresenta *delay* de aproximadamente 10ms, como mostrado na tabela da figura 3.9, o que permite um equilíbrio entre eliminação de ruído e tempo de processamento. A mesma frequência de corte foi adotada para o acelerômetro.

Observa-se também que a escala de 2000°/s ou 34.9rad/s utilizada deve ser mantida, uma

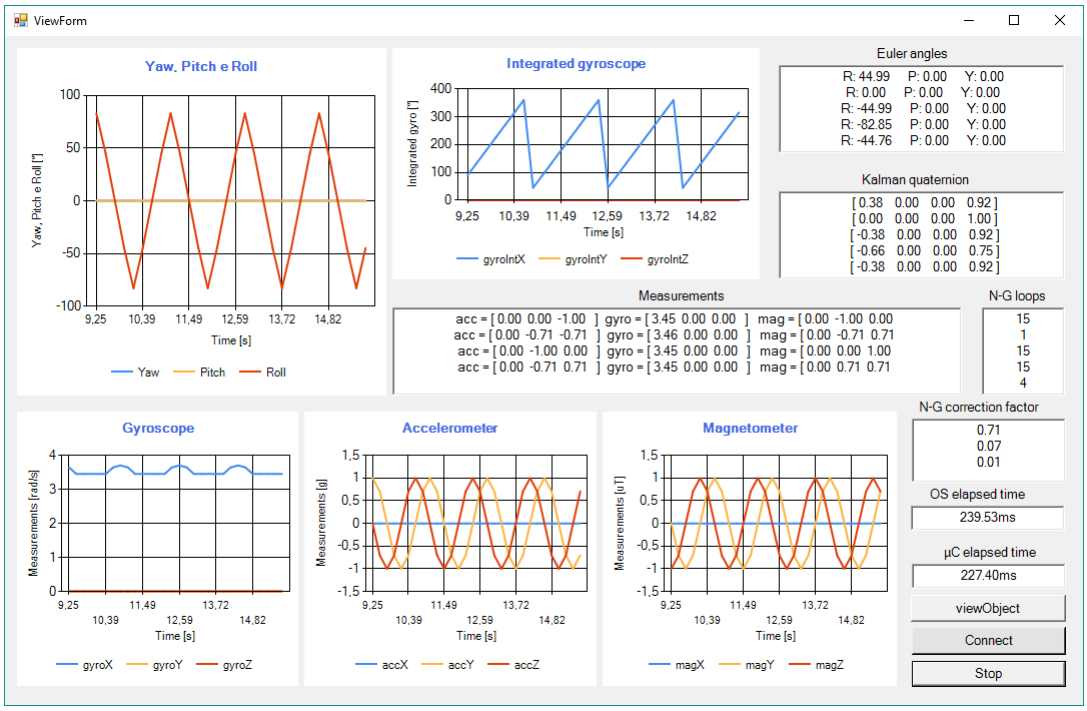


Figura 4.8: Resultados obtidos a partir de medições teóricas ideais com rotação no sentido horário.

vez que atingiu-se valores de velocidade angular superiores ao valor máximo da escala imediatamente inferior, de 1000°/s.

4.3 Convergência do algoritmo de Gauss Newton

A partir da tabela 3.4 de dados de entrada, aplicou-se o algoritmo implementado para estimar o *quaternion* que levasse os valores de *global frame* para o *body frame*, ou seja, o *quaternion* de "ida". A tabela 4.2 mostra os *quaternion*s obtido e esperado (calculado a partir da equação 2.19) para o ângulo de rotação aplicado e o número de iterações realizadas. A tabela 4.3 mostra a conversão do *quaternion* obtido para ângulos Yaw, Pitch e Roll. É importante observar que o chute inicial utilizado foi o *quaternion* de rotação por ângulo zero ($q_{inicial} = [0,0,0,1]$).

As figuras 4.8 e 4.9 mostram a visualização do resultados através da aplicação desenvolvida.

Observando-se os resultados obtidos é possível concluir que a convergência do GNA é limitado para um intervalo de $\pm\frac{\pi}{2}$ rad, uma vez que, para todos os casos em que o ângulo de rotação excedeu este intervalo, o *quaternion* resultante não foi correto. Pode-se notar ainda que para os próprios limites, $\frac{\pi}{2}$ e $-\frac{\pi}{2}$, a precisão do resultado não é boa, havendo um erro de

Tabela 4.2: Tabela de resultados de Gauss Newton para medições ideais.

Ângulo em x [rad]	*quaternion* esperado	*quaternion* obtido	Iterações
Referência	[0, 0, 0, 1]	[-2.18e-8, 0, 0, 1]	1
$\frac{\pi}{4}$	[0.382, 0, 0, 0.9238]	[0.382, 0, 0, 0.9238]	4
$\frac{\pi}{2}$	[0.707, 0, 0, 0.707]	[0.648, 0, 0, 0.768]	15
$\frac{3\pi}{4}$	[0.9238, 0, 0, 0.382]	[0.382, 0, 0, 0.924]	15
π	[1, 0, 0, -4.37e-8]	[-2.18e-8, 0, 0, 1]	1
$\frac{5\pi}{4}$	[0.9238, 0, 0, -0.382]	[-0.382, 0, 0, 0.9238]	15
$\frac{3\pi}{2}$	[0.707, 0, 0, -0.707]	[-0.648, 0, 0, 0.768]	15
$\frac{7\pi}{4}$	[0.382, 0, 0, -0.9238]	[-0.382, 0, 0, 0.9238]	4
2π	[8.74e-8, 0, 0, -1]	[-2.18e-8, 0, 0, 1]	1
$-\frac{\pi}{4}$	[-0.382, 0, 0, 0.9238]	[-0.382, 0, 0, 0.9238]	4
$-\frac{\pi}{2}$	[-0.707, 0, 0, 0.707]	[-0.648, 0, 0, 0.768]	15
$-\frac{3\pi}{4}$	[-0.9238, 0, 0, 0.382]	[-0.382, 0, 0, 0.924]	15
$-\pi$	[-1, 0, 0, -4.37e-8]	[2.18e-8, 0, 0, 1]	1
$-\frac{5\pi}{4}$	[-0.9238, 0, 0, -0.382]	[0.382, 0, 0, 0.9238]	15
$-\frac{3\pi}{2}$	[-0.707, 0, 0, -0.707]	[0.648, 0, 0, 0.768]	15
$-\frac{7\pi}{4}$	[-0.382, 0, 0, -0.9238]	[0.382, 0, 0, 0.9238]	4
-2π	[8.74e-8, 0, 0, -1]	[-2.18e-8, 0, 0, 1]	1

Tabela 4.3: Tabela de ângulos YPR obtidos de Gauss Newton para medições ideais.

Ângulo em x [rad]	YPR [°] desejado	YPR [°] estimado	YPR [°] convertido
Referência	[0,0,0]	[0,0,0]	[0,0,0]
$\frac{\pi}{4}$	[0,0,45]	[0,0,45]	[0,0,45]
$\frac{\pi}{2}$	[0,0,90]	[0,0,79.61]	[0,0,90]
$\frac{3\pi}{4}$	[0,0,135]	[0,0,45]	[0,0,45]
π	[0,0,180]	[0,0,0]	[0,0,0]
$\frac{5\pi}{4}$	[0,0,225]	[0,0,-45]	[0,0,-45]
$\frac{3\pi}{2}$	[0,0,270]	[0,0,-79.61]	[0,0,-90]
$\frac{7\pi}{4}$	[0,0,315]	[0,0,-45]	[0,0,-45]
2π	[0,0,360]	[0,0,0]	[0,0,0]
$\frac{\pi}{4}$	[0,0,-45]	[0,0,-45]	[0,0,-45]
$\frac{\pi}{2}$	[0,0,-90]	[0,0,-79.61]	[0,0,-90]
$\frac{3\pi}{4}$	[0,0,-135]	[0,0,-45]	[0,0,-45]
π	[0,0,-180]	[0,0,0]	[0,0,0]
$\frac{5\pi}{4}$	[0,0,-225]	[0,0,45]	[0,0,45]
$\frac{3\pi}{2}$	[0,0,-270]	[0,0,79.61]	[0,0,90]
$\frac{7\pi}{4}$	[0,0,-315]	[0,0,45]	[0,0,45]
2π	[0,0,-360]	[0,0,0]	[0,0,0]

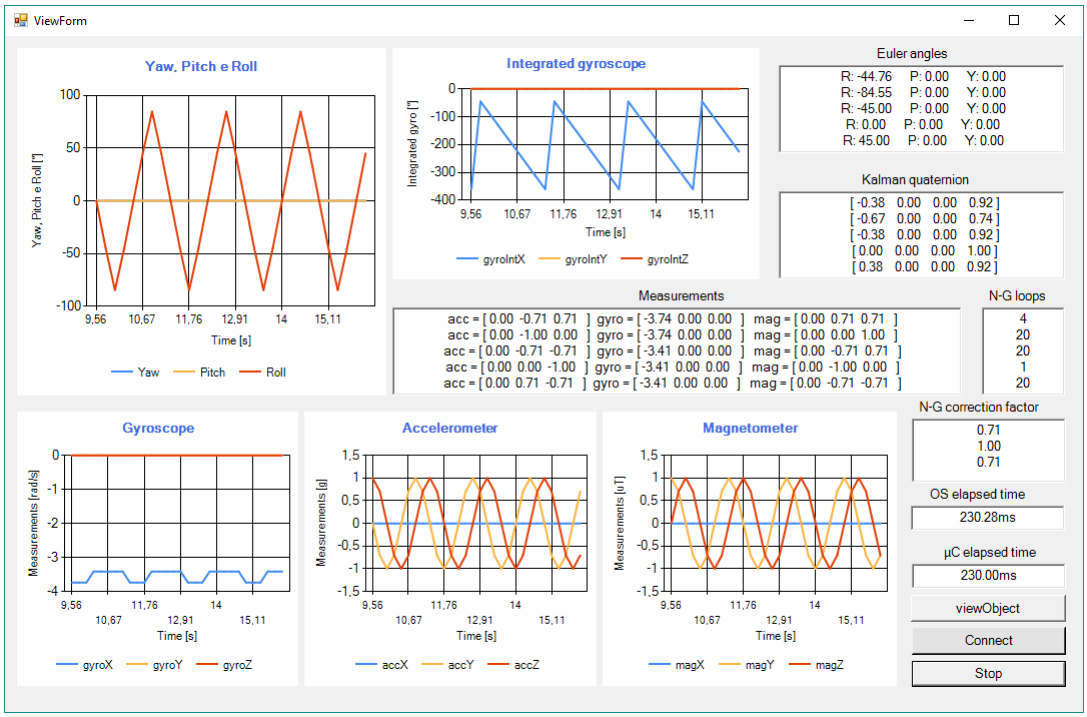


Figura 4.9: Resultados obtidos a partir de medições teóricas ideais com rotação no sentido anti horário.

$\simeq 8°$.

Também é possível notar um aumento significativo no número de iterações quando o ângulo de rotação atinge valores próximos ao limite. Devido à ambiguidade entre ângulos no 1°Q e 2°Q, e ângulos no 3°Q e 4°Q, a utilização do GNA desta forma prejudica o resultado final do sistema, cujo intuito é eliminar o problema de limitação de deslocamento angular.

Uma forma encontrada para garantir que a saída do GNA seja o mais próximo possível do valor esperado é fazer com que o *quaternion* medido represente sempre um ângulo contido no 1°Q ou 4°Q ($< \frac{\pi}{2}$). Para tanto, uma possível solução seria utilizar como referência não mais as medições obtidas no *global frame* e sim as medições obtidas no *body frame* do passo $k-1$. Desta forma, a equação a ser solucionada seria 4.1.

$$\varepsilon(q_k) = \begin{bmatrix} {}^{B}_{x}a_k \\ {}^{B}_{y}a_k \\ {}^{B}_{z}a_k \\ {}^{B}_{x}m_k \\ {}^{B}_{y}m_k \\ {}^{B}_{z}m_k \end{bmatrix} - \begin{bmatrix} \begin{bmatrix} & & \\ & {}^{k}_{k-1}R_{3x3} & \\ & & \end{bmatrix} & 0 \\ 0 & \begin{bmatrix} & & \\ & {}^{k}_{k-1}R_{3x3} & \\ & & \end{bmatrix} \end{bmatrix} \begin{bmatrix} {}^{B}_{x}a_{k-1} \\ {}^{B}_{y}a_{k-1} \\ {}^{B}_{z}a_{k-1} \\ {}^{B}_{x}m_{k-1} \\ {}^{B}_{y}m_{k-1} \\ {}^{B}_{z}m_{k-1} \end{bmatrix} \tag{4.1}$$

Uma vez que a máxima velocidade angular para o sistema é de 2000°/s, escala máxima

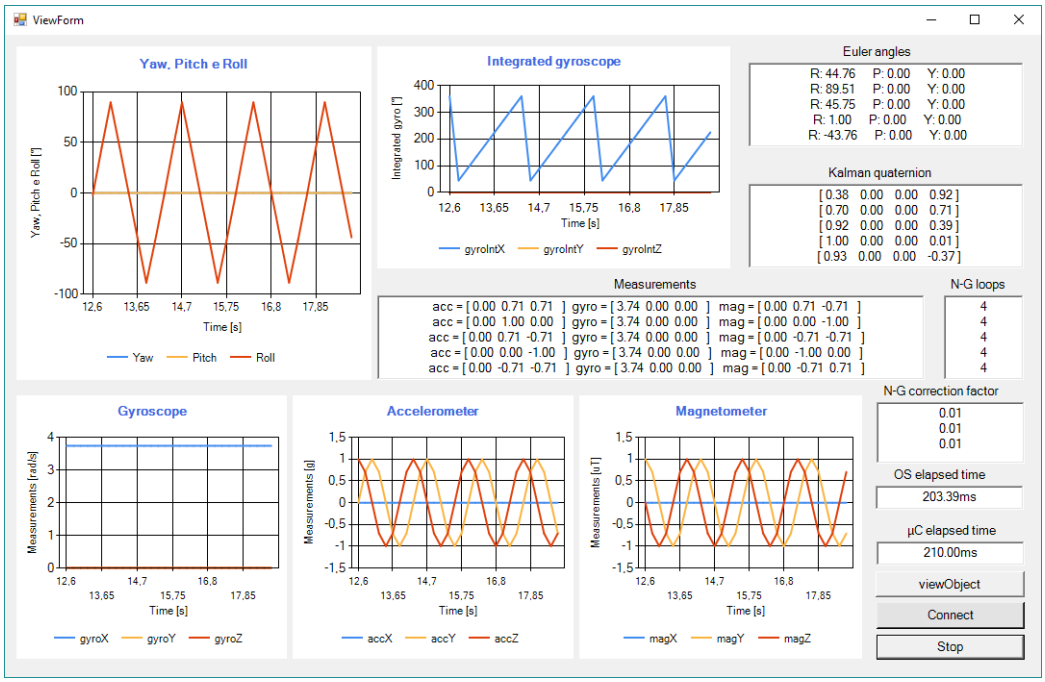


Figura 4.10: Saída do sistema completo obtida utilizando-se as medições ideais com passo $k-1$ como referência.

do gisroscópio, e sabendo-se que o tempo de rotina do algoritmo completo varia de 20ms a 40ms, como detalhado na seção 4.5, a partir da equação 4.2 é possível calcular o máximo deslocamento angular entre os passos k e $k+1$.

$$\Delta\theta = \omega T_s \rightarrow \Delta\theta \leq 80 \tag{4.2}$$

Nesta nova abordagem proposta, o *quaternion* de saída não mais poderia ser utilizado diretamente no algoritmo de Kalman, uma vez que deseja-se obter um *quaternion* que represente a movimentação angular atual com relação à posição de referência. Assim, é necessário acumular em um *quaternion* medido (q_z) as rotações calculadas pelo Gauss Newton. Isto pode ser feito multiplicando-se o *quaternion* de saída no passo k pelo de saída no passo $k-1$, e assim sucessivamente. A figura 4.10 mostra os resultados obtidos nesta situação com as medições ideais da tabela 3.4. As figuras 4.11 e 4.12 mostram resultados obtidos utilizando-se medições reais dos sensores.

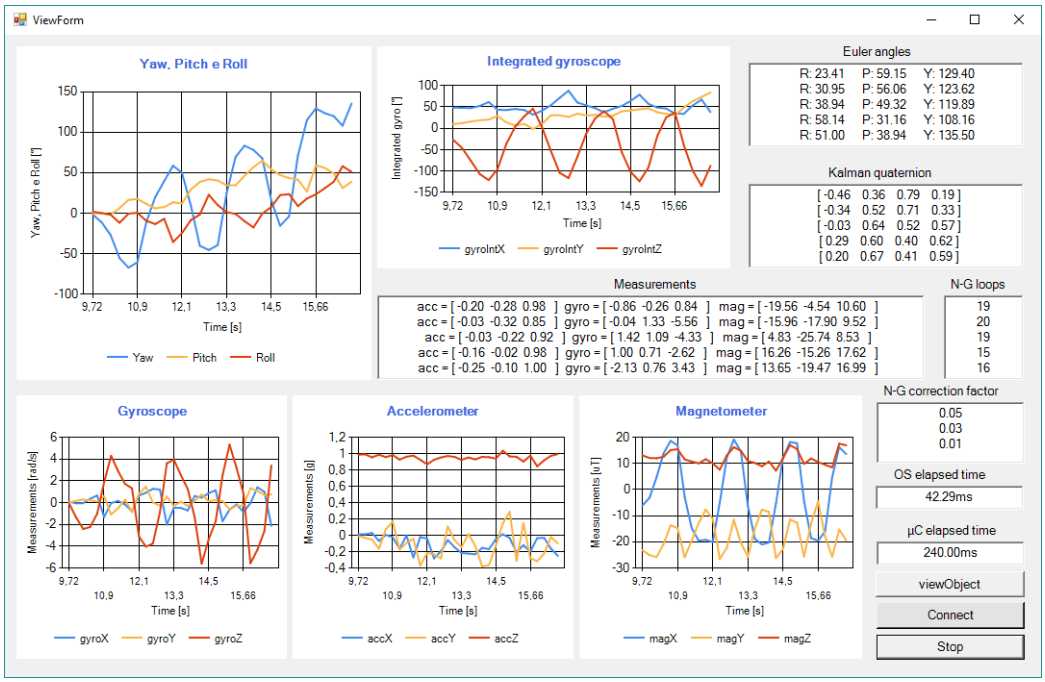


Figura 4.11: Saída do sistema completo obtida utilizando-se as medições reais com passo $k-1$ como referência.

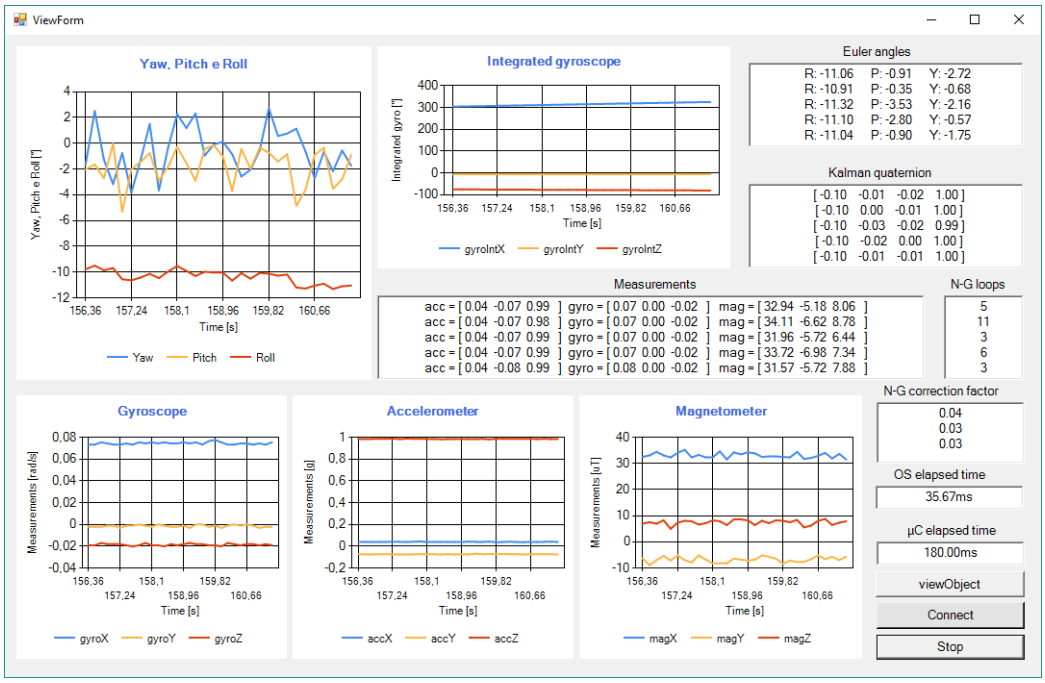


Figura 4.12: Saída do sistema completo obtida utilizando-se as medições reais com passo $k-1$ como referência.

Fica evidente que, na prática, devido a propagação de erro, não é possível utilizar esta abordagem para contornar o problema da limitação do intervalo de convergência. Na figura 4.12 é possível ver que mesmo com a placa em repouso, em pouco mais de 2min de aquisições já há *drift* de $\simeq$-10°.

4.4 Convergência do algoritmo do Filtro de Kalman

Também a partir da tabela 3.4 de dados de entrada, aplicou-se o KFA implementado para estimar o *quaternion* de "ida" do *globlal frame* para o *body frame*. A tabela 4.4 mostra os *quaternion*s obtido e esperado para o ângulo de rotação aplicado. A tabela 4.3 mostra a conversão do *quaternion* obtido para ângulos Yaw, Pitch e Roll. É importante observar que o *quaternion* de medições (q_z) utilizado foi igual ao *quaternion* desejado em cada situação.

Tabela 4.4: Tabela de resultados de Kalman para medições ideais.

Ângulo em x [rad]	*quaternion* desejado	*quaternion* obtido	Iterações
Referência	[0, 0, 0, 1]	[0.00, 0, 0, 1.00]	1
$\frac{\pi}{4}$	[0.382, 0, 0, 0.9238]	[0.38, 0, 0, 0.92]	4
$\frac{\pi}{2}$	[0.707, 0, 0, 0.707]	[0.71, 0 0, 0.71]	15
$\frac{3\pi}{4}$	[0.9238, 0, 0, 0.382]	[0.92, 0, 0, 0.38]	15
π	[1, 0, 0, -4.37e-8]	[1.00, 0, 0, 0.00]	1
$\frac{5\pi}{4}$	[0.9238, 0, 0, -0.382]	[0.92, 0, 0, -0.38]	15
$\frac{3\pi}{2}$	[0.707, 0, 0, -0.707]	[0.71, 0, 0, -0.71]	15
$\frac{7\pi}{4}$	[0.382, 0, 0, -0.9238]	[0.38, 0, 0, -0.92]	4
2π	[8.74e-8, 0, 0, -1]	[0, 0, 0, -1.00]	1
$-\frac{\pi}{4}$	[-0.382, 0, 0, 0.9238]	[-0.38, 0, 0, 0.92]	4
$-\frac{\pi}{2}$	[-0.707, 0, 0, 0.707]	[-0.73, 0, 0, 0.69]	15
$-\frac{3\pi}{4}$	[-0.9238, 0, 0, 0.382]	[-0.95, 0, 0, 0.30]	15
$-\pi$	[-1, 0, 0, -4.37e-8]	[-0.99, 0, 0, 0.08]	1
$-\frac{5\pi}{4}$	[-0.9238, 0, 0, -0.382]	[-0.91, 0, 0, -0.4]	15
$-\frac{3\pi}{2}$	[-0.707, 0, 0, -0.707]	[0.73, 0, 0, -0.69]	15
$-\frac{7\pi}{4}$	[-0.382, 0, 0, -0.9238]	[0.46, 0, 0, -0.88]	4
-2π	[8.74e-8, 0, 0, -1]	[-0.11, 0, 0, -0.99]	1

Tabela 4.5: Tabela de ângulos YPR obtidos de Kalman para medições ideais.

Ângulo em x [rad]	YPR [°] desejado	YPR [°] calculado
Referência	[0,0,0]	[0,0,0]
$\frac{\pi}{4}$	[0,0,45]	[0,0,45]
$\frac{\pi}{2}$	[0,0,90]	[0,0,90.01]
$\frac{3\pi}{4}$	[0,0,135]	[0,0,45]
π	[0,0,180]	[0,0,0]
$\frac{5\pi}{4}$	[0,0,225]	[0,0,-45]
$\frac{3\pi}{2}$	[0,0,270]	[0,0,-90.01]
$\frac{7\pi}{4}$	[0,0,315]	[0,0,-45]
2π	[0,0,360]	[0,0,0]
$\frac{\pi}{4}$	[0,0,-45]	[0,0,-45.00]
$\frac{\pi}{2}$	[0,0,-90]	[0,0,-90.01]
$\frac{3\pi}{4}$	[0,0,-135]	[0,0,-45.00]
π	[0,0,-180]	[0,0,0]
$\frac{5\pi}{4}$	[0,0,-225]	[0,0,45]
$\frac{3\pi}{2}$	[0,0,-270]	[0,0,90.01]
$\frac{7\pi}{4}$	[0,0,-315]	[0,0,45.00]
2π	[0,0,-360]	[0,0,0]

Pode-se observar da tabela 4.4 que o filtro de Kalman não possui limitação para o intervalo de convergência, uma vez que o *quaternion* por ele estimado ficou bastante próximo do desejado para o ciclo completo, como o esperado teoricamente pelo método representação de rotação adotado.

Porém ao converter para os ângulos Yaw, Pitch e Roll perde-se um grau de liberdade (4D para 3D) e há ambiguidade entre os ângulos do 1°Q e 2°Q, e entre os ângulos do 3°Q e 4°Q. Uma vez que esta conversão é feita apenas para efeito de visualização e não é utilizada no sistema, não há impactos negativos nos resultados.

4.5 Sistema completo para estimação de orientação

A seguir são feitas algumas considerações sobre o funcionamento conjunto dos dois algoritmos:

- O *quaternion* estimado em ambos os algoritmos deve relacionar *body frame* e o *global frame* da mesma forma, ou seja, ambos devem calcular um *quaternion* de "ida" ou de "volta". Para este trabalho adotou-se o primeiro;
- A medição de referência utilizada para o GNA deve ser global para que não haja propagação de erro entre as estimações;
- A cada ciclo completo de estimação são realizados *n loops* do GNA, com $n \leq n_{max}$;
- A utilização de critério de parada com relação ao erro para o GNA permite reduzir significativamente o tempo de rotina sempre que a estimação converge rapidamente;
- Os valores iniciais de *quaternion* utilizados em ambos os algoritmos equivalem ao *quaternion* estimado no passo $k-1$;

Os resultados de estimação obtidos para movimentação da placa em torno dos eixos x, y e z, respectivamente, respeitando o intervalo de convergência encontrado para o GNA, podem ser vistos nas figuras 4.13, 4.14 e 4.15.

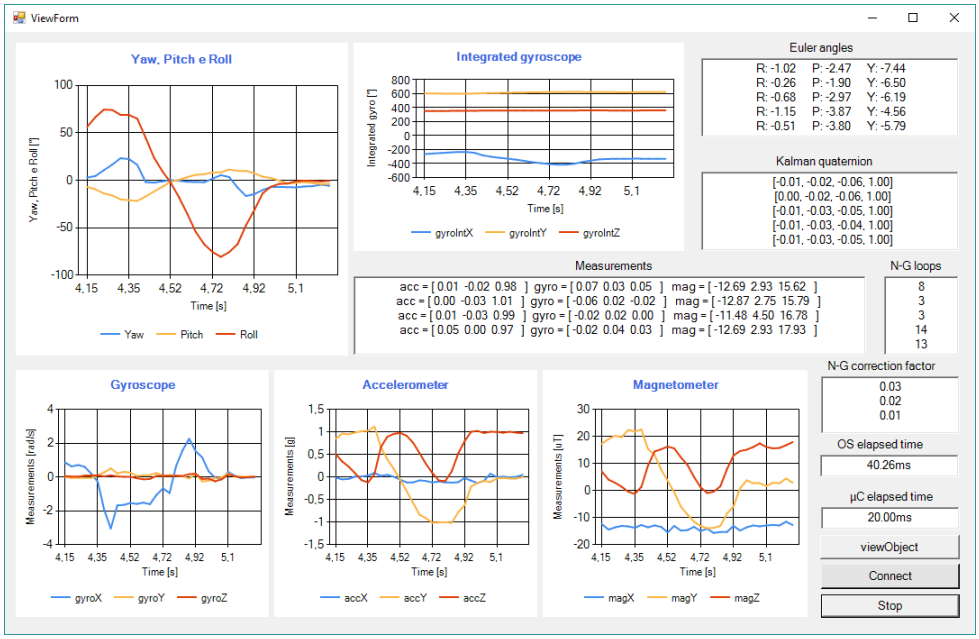


Figura 4.13: Saída do sistema completo para rotação em torno de x.

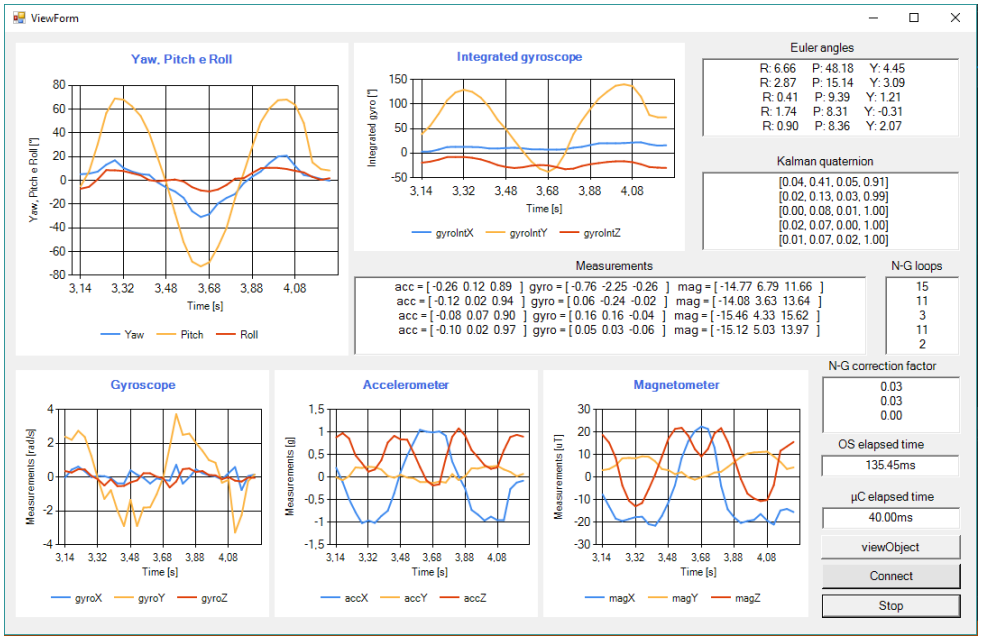


Figura 4.14: Saída do sistema completo para rotação em torno de y.

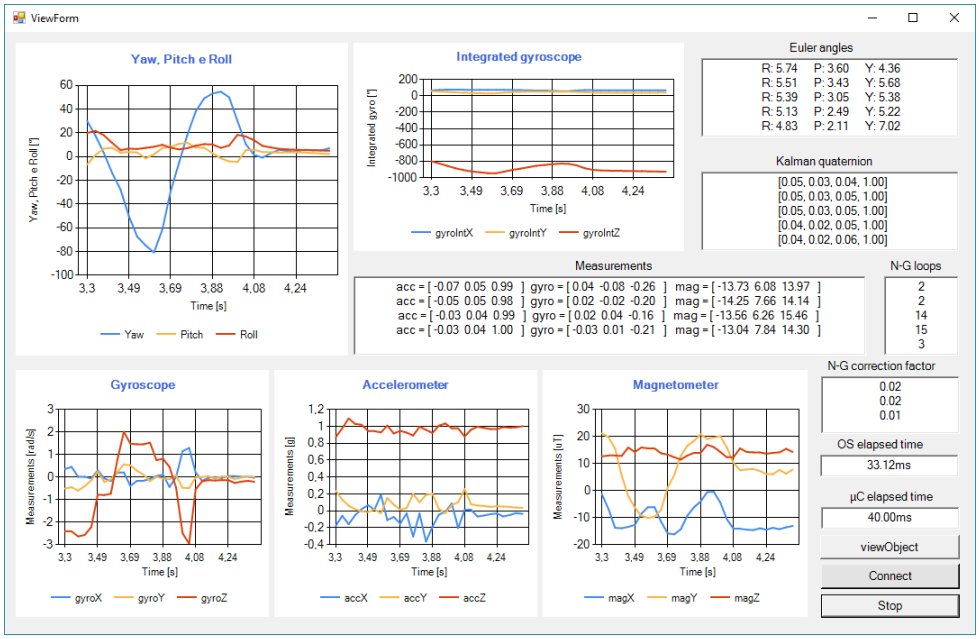


Figura 4.15: Saída do sistema completo para rotação em torno de z.

Pode-se observar que a limitação do intervalo de convergência gerada pelo GNA fica bem evidente, uma vez que não se consegue atingir ângulos maiores que $\pi/2$ rad mesmo com a placa rotacionada deste ângulo. Porém, para ângulos entre $\pm\frac{\pi}{2}$ rad obteve-se uma resposta muito satisfatória tanto em termos de estabilidade quanto em coerência com a rotação da placa de desenvolvimento.

Comparando-se os gráficos dos ângulos Yaw, Pitch e Roll obtidos na saída do filtro e pela integração das leituras do giroscópio, figura 4.15, é possível observar que, apesar da variação angular fornecida pelo segundo ser mais próxima do valor real, de aproximadamente 180° contra 160° no primeiro, a presença de *drift* torna inviável sua utilização. Na figura 4.3, em apenas 5s de execução já se tem aproximadamente 10° de deslocamento mesmo com a placa em repouso. Já para a saída filtrada, como mostrado na figura 4.15, após o fim do movimento, os três ângulo de rotação retornam rapidamente para zero.

Capítulo 5

Conclusões

O método utilizado para estimação de orientação mostrou-se adequado dados os requisitos de projeto. Devido ao reduzido custo computacional tanto do GNA, por utilizar apenas derivada de primeira ordem (matriz Jacobiana), quanto do KFA, principalmente pela linearização da matriz de observação fez com que se obtivesse um bom desempenho com o *hardware* de médio custo utilizado (microcontrolador ST32F407VG).

O filtro de Kalman aplicado a *quaternions* solucionou o problema de *Gimbal lock*, porém a utilização do GNA prejudicou a resposta dos sistema em termos de intervalo de rotação, o qual ficou limitado a $\pm\frac{\pi}{2}$ rad.

A saída do sistema mostrou-se robusta com relação à trepidações e ao *ripple* das leituras, observando-se uma oscilação média de aproximadamente $\pm 2°$. Observou-se que há oscilações significativas nos eixos x e y quando da rotação em torno do eixo z e vice-versa. Isto pode ser causado em parte pela imperfeição do movimento manual realizado e também pela má calibração dos sensores ou ruído nas leituras.

Tendo como base o gráfico da figura 4.6, no qual utilizou-se filtro passa baixa de 41Hz para o giroscópio , e tendo-se calculado a frequência máxima aproximada do gesto do usuário, de 2.4Hz, pode-se concluir que o tempo de amostragem de 20ms a 40ms conseguido foi satisfatório, resultando em no mínimo 10 amostras por ciclo, o que é suficiente para caracterização do sinal e bem a acima da frequência de Nysquist. Retirando-se *delay* utilizado poderia-se reduzir ainda mais tempo entre amostragem, para um intervalo de 10ms a 30ms (33Hz a 100Hz).

A precisão de orientação percebida pelo usuário ao rotacionar a placa também foi bastante satisfatória, com exceção de ângulos próximos do limite de convergência do GNA, onde obteve-se um erro de aproximadamente 10°, e oscilações abruptas em todos os eixos para os

casos em que se ultrapassou esse limite.

A calibração adequada dos sensores, principalmente do magnetômetro , mostrou-se fundamental para a correta convergência do método utilizado. Mesmo este não tendo atingido os valores esperados de leituras, com com erro de aproximadamente $\pm 5\mu T$, obteve-se uma melhora significativa na saída do sistema após a calibração.

Observou-se que as acelerações provocadas pela movimentação da mão do usuário não são desprezíveis e, portanto, devem ser tratadas. Com isto, pode-se resolver o problema de interferência entre os eixos.

5.1 Trabalhos futuros

Como trabalhos futuros, tem-se tanto melhorias da implementação realizada quanto a continuidade do projeto "Luva Inteligente". Pode-se citar os seguintes exemplos: medição da variância real dos sensores para correção da matriz R do filtro de Kalman, o desenvolvimento de um processo de calibração para o magnetômetro na interface da aplicação para Windows, a utilização de comunicação SPI para leitura dos sensores a fim de reduzir o tempo de processamento, a utilização de comunicação Wi-Fi para tornar o sistema mais versátil inclusive para aplicações IoT, a implementação do método utilizando-se 4 unidades inerciais que seriam posicionadas para detectar a orientação dos dedos do usuário e, por fim a implementação da inteligência artificial para identificação do gesto.

Referências bibliográficas

[1] Zunyi Tang, Masaki Sekine, Toshiyo Tamura, Noriko Tanaka, Masaki Yoshida, and Wenxi Chen. Measurement and estimation of 3d orientation using magnetic and inertial sensors. *Advanced Biomedical Engineering*, 4(0):135–143, 2015.

[2] Xiaoping Yun, Mariano Lizarraga, Eric R Bachmann, and Robert B McGhee. An improved quaternion-based kalman filter for real-time tracking of rigid body orientation. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1074–1079. IEEE, 2003.

[3] Xiaoping Yun and Eric R Bachmann. Design, implementation, and experimental results of a quaternion-based kalman filter for human body motion tracking. *IEEE transactions on Robotics*, 22(6):1216–1227, 2006.

[4] Angelo M Sabatini. Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing. *IEEE Transactions on Biomedical Engineering*, 53(7):1346–1356, 2006.

[5] Roberto S Inoue and Marco H Terra. Robust recursive kalman filtering for attitude estimation. *IFAC Proceedings Volumes*, 44(1):1108–1113, 2011.

[6] Aida Makni, Hassen Fourati, and Alain Y Kibangou. Adaptive kalman filter for mems-imu based attitude estimation under external acceleration and parsimonious use of gyroscopes. In *Control Conference (ECC), 2014 European*, pages 1379–1384. IEEE, 2014.

[7] Sensor fusion on android devices: A revolution in motion processing - googletechtalks. `https://www.youtube.com/watch?v=C7JQ7Rpwn2k`, Acesso em 15 de Março de 2016.

[8] Efeito coriolis. `http://www5.epsondevice.com/en/information/technical_info/gyro/`, Acesso em: 07 de Abril de 2016.

[9] Mems. http://www.pcbheaven.com/opendir/index.php?show=4109sg7239pu0d6729c0, Acesso em: 06 de Abril de 2016.

[10] Massa sísmica. http://www.starlino.com/imu_guide.html, Acesso em: 07 de Abril de 2016.

[11] Acelerômetro mems. https://www.quora.com/Robotics-What-is-the-concept-behind-gyroscopes-and-accelerometers, Acesso em: 07 de Abril de 2016.

[12] Efeito hall 1. en.wikipedia.org/wiki/Hall_effect, Acesso em: 07 de Abril de 2016.

[13] Efeito hall 2. https://pt.wikipedia.org/wiki/Efeito_Hall, Acesso em: 07 de Abril de 2016.

[14] Aplicação de filtro passa baixa. http://devcenter.wintellect.com/jprosise/using-the-accelerometer-in-silverlight-for-windows-phone, Acesso em: 07 de Abril de 2016.

[15] Aplicação de filtro passa baixa em giroscópio. https://roboticdreams.wordpress.com/category/arduino/, Acesso em: 07 de Abril de 2016.

[16] Brendan O'Flynn, Javier Torres Sanchez, James Connolly, Joan Condell, Kevin Curran, Philip Gardiner, and Barry Downes. Integrated smart glove for hand motion monitoring. 2015.

[17] Walter T Higgins. A comparison of complementary and kalman filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 11(3):321–325, 1975.

[18] A complementary filter block diagram. https://www.researchgate.net/figure/275718992_fig4_A-complementary-filter-block-diagram, Acesso em 17 de Abril de 2016.

[19] An introduction to the kalman filter. http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf, Acesso em 17 de Março de 2016.

[20] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[21] State estimation with kalman filter. http://techteach.no/fag/seky3322/0708/kalmanfilter/kalmanfilter.pdf, Acesso em 11 de Março de 2016.

[22] Hidden markov model kalman filter. http://bme.ccny.cuny.edu/faculty/parra/teaching/biomed-dsp/class10.pdf, Acesso em 11 de Junho de 2016.

[23] Derivations of the discrete-time kalman filter. http://webee.technion.ac.il/people/shimkin/Estimation09/ch4_KFderiv.pdf, Acesso em 11 de Junho de 2016.

[24] Tutorial: Kalman filter with matlab example part2. https://www.youtube.com/watch?v=NT7nYv9Ri2Y, Acesso em 11 de Março de 2016.

[25] Representação de orientação. http://www.mathworks.com/help/physmod/sm/mech/gs/representations-of-body-orientation.html, Acesso em 09 de Junho de 2016.

[26] Ângulos de euler. https://en.wikipedia.org/wiki/Euler_angles, Acesso em: 07 de Abril de 2016.

[27] Eixo de rotação. http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-17-quaternions/, Acesso em: 09 de Junho de 2016.

[28] Nikolas Trawny and Stergios I Roumeliotis. Indirect kalman filter for 3d attitude estimation. *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, 2:2005, 2005.

[29] William Rowan Hamilton. *Elements of quaternions*. Longmans, Green, & Company, 1866.

[30] Sentido de rotação. https://en.wikipedia.org/wiki/Rotation_matrix, Acesso em: 09 de Junho de 2016.

[31] Li Wang, Zheng Zhang, and Ping Sun. Quaternion-based kalman filter for ahrs using an adaptive-step gradient descent algorithm. *International Journal of Advanced Robotic Systems*, 12, 2015.

[32] João Luís Marins, Xiaoping Yun, Eric R Bachmann, Robert B McGhee, and Michael J Zyda. An extended kalman filter for quaternion-based orientation estimation using marg sensors. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 4, pages 2003–2011. IEEE, 2001.

[33] Mpu9250. http://www.naylampmechatronics.com/882-thickbox_default/modulo-mpu9250-acelerometro-giroscopio-magnetometro-i2c.jpg, Acesso em: 10 de Novembro de 2016.

[34] Mpu-9250 register map and descriptions revision 1.4. https://cdn.sparkfun.com/assets/learn_tutorials/5/5/0/MPU-9250-Register-Map.pdf, Acesso em 17 de Abril de 2016.

[35] Mpu-9250 product specification revision 1.0. https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf, Acesso em 17 de Abril de 2016.

[36] Magnetic field calculators. http://www.ngdc.noaa.gov/geomag-web/#igrfwmm, Acesso em 06 de Novembro de 2016.

[37] Inclinação e declinação. http://hs.umt.edu/geosciences/faculty/sheriff/courses/439-applied-magnetics/default.php, Acesso em 06 de Novembro de 2016.

[38] St32f4discovery. http://www.st.com/en/evaluation-tools/stm32f4discovery.html, Acesso em 05 de Julho de 2016.

[39] St32f407vg *Datasheet*. http://www.st.com/en/microcontrollers/stm32f407vg.html, Acesso em 05 de Julho de 2016.

[40] Stm32f4 discovery. http://embedded-lab.com/blog/wp-content/uploads/2015/02/STM32F4-Discovery-Board.jpg, Acesso em 06 de Novembro de 2016.

[41] Módulo bluetooth. http://www.martyncurrey.com/wp-content/uploads/2015/08/HC-05-FC-114-HC-06-FC-114_1200.jpg, Acesso em 06 de Novembro de 2016.

[42] Hc05 *Datasheet*. http://www.electronicaestudio.com/docs/istd016A.pdf, Acesso em 10 de Setembro de 2016.

Apêndice A

Códigos e bibliotecas

Códigos e bibliotecas desenvolvidos podem ser obtidos em:
`https://github.com/jebohnvida/KalmanFilter`