Universidade federal de Goiás – UFG

Thiago Henrique de Oliveira

Fusão de sensores para robô da categoria VSSS

Brasil

2018

sistema de bibliotecas ufg

PRPG
PRÓ-REITORIA DE PÓS-GRADUAÇÃO

UFG

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

1. Identificação do material bibliográfico: **[ ] Dissertação** **[ ] Tese** [X] TCC

2. Identificação da Tese ou Dissertação:

Nome completo do autor: Thiago Henrique de Oliveira

Título do trabalho: Fusão de sensores para robô da categoria VSSS

3. Informações de acesso ao documento:

Concordа com a liberação total do documento [X] SIM [ ] NÃO1

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF da tese ou dissertação.

Thiago Henrique de Oliveira
Assinatura do(a) autor(a)2

Ciente e de acordo:
Marco Antonio Assfalk de Oliveira
Assinatura do(a) orientador(a)2 Data: 2018 / 12 / 21

1 Neste caso o documento será embargado por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Os dados do documento não serão disponibilizados durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

2 A assinatura deve ser escaneada.

Thiago Henrique de Oliveira

Fusão de sensores para robô da categoria VSSS

Projeto Final de curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Engenharia de Computação

Universidade federal de Goiás – UFG

Orientador: Marco Antonio Assfalk de Oliveira

Brasil

2018

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Oliveira, Thiago Henrique de
Fusão de sensores para robô da categoria VSSS [manuscrito] / Thiago Henrique de Oliveira. - 2018.
LV, 55 f.: il.

Orientador: Prof. Dr. Marco Antonio Assfalk de Oliveira.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Engenharia de Computação, Cidade de Goiás, 2018.
Bibliografia.
Inclui gráfico, tabelas, algoritmos.

1. Fusão de sensores. 2. EKF. 3. UKF. 4. VSSS. 5. Robotica. I. Oliveira, Marco Antonio Assfalk de, orient. II. Título.

CDU 621.3

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO
Coordenação de Estágio e Projeto Final

ATA DE AVALIAÇÃO DE PROJETO FINAL II

Aos 21 dias do mês de dezembro do ano de 2018, foi apresentado e defendido o Projeto Final II, intitulado Fusão de Sensores para Robô da categoria VSSS

perante a banca examinadora composta pelos membros:

1. Marco Antonio Assfalk de Oliveira, orientador e presidente;
2. Gisele Guimarães;
3. Lucas da Silva Assis.

Após a exposição do trabalho por parte do(s) autor(es), aluno(s) do curso de Engenharia de Computação, foram lhe(s) atribuídas as seguintes notas pelos membros da banca:

Nome do(a) Aluno(a)	Membro1	Membro2	Membro3
Thiago Henrique de Oliveira	10,0	10,0	10,0

Nada mais havendo a registrar, eu, Marco Antonio Assfalk de Oliveira, designado secretário "ad hoc" da banca examinadora, lavrei a presente Ata do ocorrido, a qual, lida e considerada conforme, vai assinada por mim e pelos membros da banca.

Goiânia, 21 de dezembro de 2018.

Marco Antonio Assfalk de Oliveira

Gisele Guimarães

Lucas da Silva Assis

Av. Universitária, no 1488, Quadra 86, Bloco A – 3o Piso – CEP: 74605-010 – Setor Leste Universitário – Goiânia - GO
Telefones: (62) 3209-6079 | (62) 3209-6070 - Fax: (62) 3209-6292 | email: secretaria@emc.ufg.br

Resumo

IEEE VSSS é uma categoria de competição de robótica em que cada equipe utiliza 3 robôs pequenos, tipicamente de duas rodas, observados por um sistema de visão computacional. Na categoria, os robôs tem o objetivo de jogar futebol, o que exige movimentos rápidos e precisos. Para controlá-los bem, é necessária uma boa estimativa da posição, orientação e velocidade linear e angular. O objetivo deste projeto é estimar bem estas variáveis, utilizando de fusão de sensores para lidar com ruído e falhas inerentes a cada sensor.
Para obter estes dados, os sensores utilizados foram uma IMU com giroscópio, magnetômetro e acelerômetro de 3 eixos, um encoder em cada roda e um sistema de visão computacional que mede posição e orientação do robô. O Filtro de Kalman Estendido (EKF) foi implementado para combinar a informação desses sensores, modelando a incerteza nas medidas de cada um.
O Unscented Kalman Filter (UKF) também foi implementado, comparando com o EKF os erros nas estimativas, desempenho de execução e outros fatores importantes.
Cada sensor foi configurado para medir as variáveis desejadas. Foi encontrado diversas distorções nos sensores, que foram corrigidas utilizando outros sensores como referência e através de métodos estatísticos. Os ruídos de cada um foram medidos e modelados nas matrizes do EKF para atingir uma boa rejeição de ruído.
Foram feitos vários testes para verificar a qualidade destas estimativas, e o resultado foi satisfatório. Houve boa eliminação de ruído e convergência rápida para o valor correto.

Palavras-chave: EKF. VSSS. Fusão de sensores.

Lista de ilustrações

Sumário

1 Introdução

IEEE VSSS é uma categoria de competição de robótica em que cada equipe utiliza 3 robôs pequenos, tipicamente de duas rodas, observados por um sistema de visão computacional. Na categoria, os robôs tem o objetivo de jogar futebol, guiados por um sistema de visão computacional e sensores opcionais dentro do robô. No robô utilizado neste projeto, os sensores disponíveis são uma IMU (unidade de medição inercial) com giroscópio, magnetômetro e acelerometro, um encoder em cada roda e o sistema de visão computacional que mede posição e orientação do robô.

Cada sensor tem vantagens e desvantagens: O giroscópio é rápido e preciso, mas não é capaz de corrigir eventuais erros na orientação por não ser capaz de medi-la diretamente, apenas velocidade angular. O encoder pode medir a velocidade de cada roda separadamente, o o que permite calcular tanto a velocidade linear quanto a angular, mas pode medir dados errados caso o robô derrape. O sistema de visão computacional disponível é capaz de medir posição e orientação diretamente, mas tem baixa taxa de atualização e não tem alta precisão.

Utilizando estes sensores de forma individual em um sistema de controle pode resultar em problemas de estabilidade. Por exemplo, um sistema de controle que utilize realimentação apenas com dados da odometria, mesmo que seja estável em condições normais, pode se tornar instavel caso haja derrapamentos.

Utilizando métodos de fusão sensorial, é possível combinar a informação desses sensores para obter estimativas muito superiores ao que seria obtido de cada um de forma individual. A possibilidade de utilizar sensores diferentes que medem a mesma variável possibilita estimar esta variável com ruído resultante menor do que o ruído de cada sensor individual. É possível levar em consideração o nível de ruído de cada sensor ao combinar os dados obtidos por eles.

O algorítmo utilizado neste projeto é o Filtro de Kalman Estendido (EKF), que modela a incerteza em cada valor medido como um ruído gaussiano, assim como a incerteza na estimativa obtida por ele. Os ruídos de cada um foram medidos e modelados nas matrizes do EKF para que o algorítmo possua boa rejeição de ruído. Cada sensor utilizado foi configurado e as distorções foram corrigidas utilizando outros sensores como referência ou através de métodos estatísticos. Também foi implementado o Unscented Kalman Filter (UKF), para compara-lo com o EKF, observando os erros nas estimativas, desempenho de execução e outros fatores.

2 Desenvolvimento

2.1 Filtro de Kalman Estendido

Há diversas variações do filtro de Kalman, adequados para sistemas diferentes. O Filtro de Kalman tradicional é compatível apenas com sistemas lineares, então não pode ser utilizado neste sistema. Já o filtro de Kalman estendido (Extended Kalman Filter - EKF) é compatível com sistemas não lineares, e por isso é utilizado neste projeto.

Diferente do Filtro de Kalman tradicional, o filtro de Kalman estendido utiliza equações não lineares para modelar a transição de estados e as medidas, e utiliza jacobianas dessas equações no lugar das matrizes A e B.

O algoritmo é dividido em duas etapas: predição e atualização. Na predição, é calculado o valor das variáveis de estado a partir do valor anterior, utilizando um modelo do sistema para prever os novos valores. Na atualização, os sensores são utilizados para corrigir o valor predito, melhorando a estimativa.

As equações da etapa de predição são:

$$\bar{\mu_t} = f(\mu_{t-1}, u) \tag{2.1}$$

$$\bar{\Sigma_t} = F_t * \Sigma_{t-1} * F_t^T + Q_t \tag{2.2}$$

As equações da etapa de atualização são:

$$K_t = \bar{\Sigma_t} * H_t^T * (H_t * \bar{\Sigma_t} * H_t^T + R_t)^{-1} \tag{2.3}$$

$$\mu_t = \bar{\mu} + K_t(z_t - h(\bar{\mu_t})) \tag{2.4}$$

$$\Sigma_t = (I - K_t * H_t) * \bar{\Sigma_t} \tag{2.5}$$

Em que μ e Σ_t são os parâmetros da distribuição normal (valor esperado e matriz de covariança) que modela a estimativa do EKF, u são as variáveis de controle, $\bar{\mu}$ e $\bar{\Sigma_t}$ são os parametros da gaussiana predita pelo modelo, K é o Ganho de Kalman e z_t são os valores medidos pelos sensores.

Para utilizar o algorítmo, devem ser definidas as equações e matrizes:

- Modelo do sistema $f(\mu_{t-1}, u)$ e o jacobiano F_t

- Modelo de medidas $h(\bar{\mu})$ e o jacobiano H_t
- Matriz de covariancia R_t
- Matriz de ruído de processo Q_t

2.2 Implementação

Como o algoritmo deve ser executado em um microcontrolador com processador ARM Cortex-M3, que não possui uma FPU, é necessário que seja implementado de forma altamente otimizada para atingir uma boa taxa de amostragem. O Filtro de Kalman Estendido (e posteriormente, o UKF) foi implementado utilizando a biblioteca Eigen (1), por oferecer operações de matrizes como multiplicação e inversa com alto desempenho. Ela é tipicamente utilizada em computadores x86, mas foi possível compila-la para ARM sem muitos problemas. O tempo gasto para calcular uma iteração do EKF no microcontrolador é de 0.894ms, enquanto o tempo total, incluindo o EKF e a medida de sensores, é de 1.033ms (frequência de 968Hz). O alto desempenho obtido permite tirar bom proveito da alta taxa de atualização dos sensores utilizados.

Outra biblioteca utilizada foi o MBED (2), que fornece um sistema operacional em tempo real (rtos), driver de I2C que foi utilizado para comunicação com os sensores, driver de UART utilizado para comunicar com o rádio xbee, e o driver QEI, que conta os pulsos medidos pelo encoder. Para comunicar com o computador via xbee, foi utilizado a biblioteca XBeeLib (3) fornecida pela fabricante, que é compatível com o MBED.

A implementação do EKF, drivers dos sensores, modelagem matemática, etc. está disponível no github do Pequi Mecânico (4).

2.3 Modelo matemático

O movimento do robô pode ser modelado como circular, mas tal modelo resulta equações mais complexas e não definidas para movimentos em linha reta, e por isso não é muito utilizado na literatura. Por isso, o modelo mais comum na literatura, e que é utilizado nesse trabalho, é o do uniciclo (5), em que o movimento é aproximado a uma reta na mesma direção e sentido da velocidade do robô. As equações utilizadas são:

$$\Delta x = \Delta S * \cos(\theta + \frac{\Delta\theta}{2}) \tag{2.6}$$

$$\Delta y = \Delta S * \sin(\theta + \frac{\Delta\theta}{2}) \tag{2.7}$$

A relação entre os deslocamentos e as velocidades linear e angular, para um curto intervalo de tempo, são:

$$\Delta S = v * \Delta t \tag{2.8}$$

$$\Delta\theta = \omega * \Delta t \tag{2.9}$$

Substituindo na equação anterior, pode se obter a equação de transição de estados:

$$x_t = x_{t-1} + v_t * \Delta t * \cos(\theta_{t-1} + \frac{\omega_t * \Delta t}{2}) \tag{2.10}$$

$$y_t = y_{t-1} + v_t * \Delta t * \sin(\theta_{t-1} + \frac{\omega_t * \Delta t}{2}) \tag{2.11}$$

$$\theta_t = \theta_{t-1} + \omega_t * \Delta t \tag{2.12}$$

A relação entre as velocidades do robô e a das rodas:

$$\omega = \frac{(v_r - v_l)}{L} \tag{2.13}$$

$$v = \frac{(v_r + v_l)}{2} \tag{2.14}$$

Onde v_r é a velocidade da roda direita e v_l é a velocidade da roda esquerda.

Para poder estimar também as velocidades linear e angular utilizando dados de vários sensores, elas foram consideradas também como variáveis de estado. A variação das velocidades são modeladas pela equação:

$$v_t = v_{t-1} + a * \Delta t \tag{2.15}$$

$$\omega_t = \omega_{t-1} + \alpha * \Delta t \tag{2.16}$$

Onde a é a aceleração linear e α é a aceleração angular. Desta forma, as acelerações são as variáveis de controle. Antes de adicionar o acelerômetro, elas eram calculadas numéricamente a partir das velocidades das rodas:

$$a_{roda} = \frac{v_{t,roda} - v_{t-1,roda}}{T} \tag{2.17}$$

$$a = \frac{(a_r + a_l)}{2} \tag{2.18}$$

$$\alpha = \frac{(a_r - a_l)}{L} \tag{2.19}$$

A equação 2.19 foi posteriormente substituida por leituras do acelerômetro presente na IMU utilizada. Para remover a aceleração da gravidade das leituras do acelerômetro, θ_y foi adicionado como variável de estado para estimar a inclinação do robô no eixo y. A estimativa de θ_y é descrita em mais detalhes na seção 2.8.

Para eliminar efeitos externos no magnetômetro, θ_{offset} foi adicionado como variável de estado para estimar o offset nas leituras devido a campos magnéticos não modelados, como a presença de materiais metálicos próximos ao robô. ω_y foi adicionada como variável de controle para calcular alterações em θ_y. A correção no magnetômetro é descrita em mais detalhes na seção 2.7

O vetor de variáveis de estado é:

$$\begin{bmatrix} x \\ y \\ \theta \\ v \\ \omega \\ \theta_{offset} \\ \theta_y \end{bmatrix}$$

A equação de transição de estados $f(\mu_{t-1}, u)$ é:

$$x_t = x_{t-1} + v_{t-1} * \Delta t * \cos(\theta_{t-1} + \frac{\omega_{t-1} * \Delta t}{2}) \tag{2.20}$$

$$y_t = y_{t-1} + v_{t-1} * \Delta t * \sin(\theta_{t-1} + \frac{\omega_{t-1} * \Delta t}{2}) \tag{2.21}$$

$$\theta_t = \theta_{t-1} + \omega_{t-1} * \Delta t \tag{2.22}$$

$$v_t = v_{t-1} + a * \Delta t \tag{2.23}$$

$$\omega_t = \omega_{t-1} + \alpha * \Delta t \tag{2.24}$$

$$\theta_{offset,t} = \theta_{offset,t-1} \tag{2.25}$$

$$\theta_{y,t} = \theta_{y,t-1} * \omega_{y,t-1} \tag{2.26}$$

Derivando cada equação em relação a cada variável de estado resulta no jacobiano F:

$$\begin{array}{cccccc} 1 & 0 & -v_{t-1}*\Delta t*\sin(\theta_{t-1}+\frac{\omega_{t-1}*\Delta t}{2}) & \Delta t*\cos(\theta_{t-1}+\frac{\omega_{t-1}*\Delta t}{2}) & -v_{t-1}*\Delta t*\sin(\theta_{t-1}+\frac{\omega_{t-1}*\Delta t}{2})*\frac{\Delta t}{2} & 0 \\ 0 & 1 & v_{t-1}*\Delta t*\cos(\theta_{t-1}+\frac{\omega_{t-1}*\Delta t}{2}) & \Delta t*\sin(\theta_{t-1}+\frac{\omega_{t-1}*\Delta t}{2}) & v_{t-1}*\Delta t*\cos(\theta_{t-1}+\frac{\omega_{t-1}*\Delta t}{2})*\frac{\Delta t}{2} & 0 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array} \tag{2.27}$$

O tempo entre dados medidos pela visão computacional é de 33ms devido ao uso de uma câmera de 30fps, enquanto os outros sensores são mais rápidos: giroscópio e

acelerômetro suportam taxas de atualização de até 1.66kHz e o magnetômetro de 80Hz. Para lidar com essas diferenças, além de diminuir o tempo gasto calculando o algorítmo, a etapa de atualização foi dividida em duas: utilizar apenas dados da IMU e encoder ou apenas dados da visão e magnetômetro (para correção de offset). Atualizações com IMU e encoder são modelados por uma equação $h(z_t)$ e jacobiano H, enquanto atualizações de visão utilizam $h_v(z_t)$ e H_v. A vantagem computacional é grande: as inversas calculadas são de matrizes 4x4, em vez de sempre inverter uma de 7x7.

O modelo de medidas $h(z_t)$ para o IMU e encoder é:

$$\theta_{magnetometro} = \theta_t + \theta_{offset} \tag{2.28}$$

$$\omega_{giroscopio} = \omega_t \tag{2.29}$$

$$v_l = v_t - \frac{\omega_t * L}{2} \tag{2.30}$$

$$v_r = v_t + \frac{\omega_t * L}{2} \tag{2.31}$$

E o jacobiano H é:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -\frac{L}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 & \frac{L}{2} & 0 & 0 \end{bmatrix}$$

O modelo de medidas $h_v(z_t)$ para a visão é:

$$x_{visão} = x_t \tag{2.32}$$

$$y_{visão} = y_t \tag{2.33}$$

$$\theta_{visão} = \theta_t \tag{2.34}$$

$$\theta_{mag\,offset\,t} = \theta_{offset\,t} \tag{2.35}$$

E o jacobiano H_v é:

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

2.4 Configuração da IMU

A IMU utilizada (6) contém dois circuitos integrados: o LSM6DS33, que contém um giroscópio e um acelerômetro de 3 eixos, e o LIS3MDL, que contém um magnetômetro. A comunicação com os sensores é feita através do protocolo I2C, permitindo ler ou escrever nos registradores internos dos sensores.

De acordo com o datasheet, para escrever em registradores deve ser enviado o endereço do registrador seguido do valor a ser escrito. O datasheet também descreve a funcionalidade de cada um e como configurá-los.

Os registradores relevantes e suas configurações são:

- CTRL1_C: Permite habilitar ou desabilitar cada eixo do giroscópio. Foi inicialmente configurado para habilitar apenas o eixo z, por ser o único necessário para medir a velocidade angular em um sistema de coordenadas 2D. Para corrigir os efeitos da aceleração da gravidade no acelerômetro, o eixo y foi habilitado posteriormete para uso na estimativa de inclinação do robô
- CTRL2_G: Permite selecionar a taxa de atualização do giroscópio e valor máximo. A taxa de atualização selecionada foi de 833Hz, que resulta num período de 1.2ms. Períodos menores não são necessários porque as medidas não seriam utilizadas, já que cada etapa do EKF demora aproximadamente 1ms para serem calculadas. Já o valor máximo selecionado foi de 2000° por segundo, ou 34.9 rad/s, que é o máximo permitido pelo sensor. Valores menores resultariam em saturações, porque o robô conseguiria se movimentar a velocidades angulares maiores que o sensor mede.
- CTRL_REG2: Configura a faixa de valores que o magnetômetro irá medir. Foi configurado para -4 a +4 Gauss, que é a menor faixa possível, porque a intensidade de campo magnético da terra é menor que 4 Gauss, e faixas menores permitem precisão maior nas medidas.
- CTRL_REG1: Configura a taxa de atualização do magnetômetro e habilita os eixos do sensor. Como o robô se movimenta em duas dimensões, foram habilitados apenas os eixos x e y necessários para calcular o ângulo do robô no eixo z. A taxa de atualização foi selecionada como 80Hz (período de 12.5ms) por ser a maior frequência possível. Como o período do sensor é maior que o tempo de calcular o EKF, algumas iterações do EKF não utilizarão o magnetômetro.
- CTRL9_XL: Habilita cada eixo do acelerômetro. Foi configurado para habilitar os 3 eixos.
- CTRL1_XL: Configura a taxa de atualização e a faixa de medida dos acelerômetro. A taixa de aceleração disponível é até 6.66kHz. Foi selecionado 1.66kHz, por ser

a maior taxa disponível que o microcontrolador consegue processar. A faixa do acelerômetro selecionada é de 4 vezes a aceleração da gravidade (39.2266 m/s^2), por ser suficiente para atender a aceleração do robô durante o seu movimento.

Cada eixo de cada sensor possui dois registradores de 8 bits, que armazenam o byte superior e o inferior da última medição do sensor, que é um número de 16 bits. Para lê-los, é enviado por I2C o endereço do registrador referente ao eixo, seguido do número de bytes que serão lidos, e o sensor responde com o valor armazenado nele.

Com o algoritmo de configurar os sensores e ler os dados pronto, falta apenas converter o valor lido para a unidade de medida correta. Para isso, é utilizado o valor máximo possível de cada sensor, conforme configurado anteriormente. A equação de conversão é:

$$valor_{real} = valor_{medido} * \frac{valorMáximo_{real}}{valorMáximo_{medido}} \tag{2.36}$$

Como todos os valores são de 16 bits, o $valorMáximo_{medido}$ é sempre:

$$\frac{2^{16}}{2} - 1 = 32767 \tag{2.37}$$

Já o valor máximo real depende de cada sensor. No giroscópio, como a velocidade angular máxima é de 34.9 rad/s, o valor medido em rad/s é dado por:

$$\alpha = valor_{medido} * \frac{34.9rad/s}{32767} \tag{2.38}$$

Já no magnetômetro, como o valor máximo da intensidade de campo magnético foi comfigurado para 4 Gauss, o valor em Gauss é dado por:

$$B = valor_{medido} * \frac{4Gauss}{32767} \tag{2.39}$$

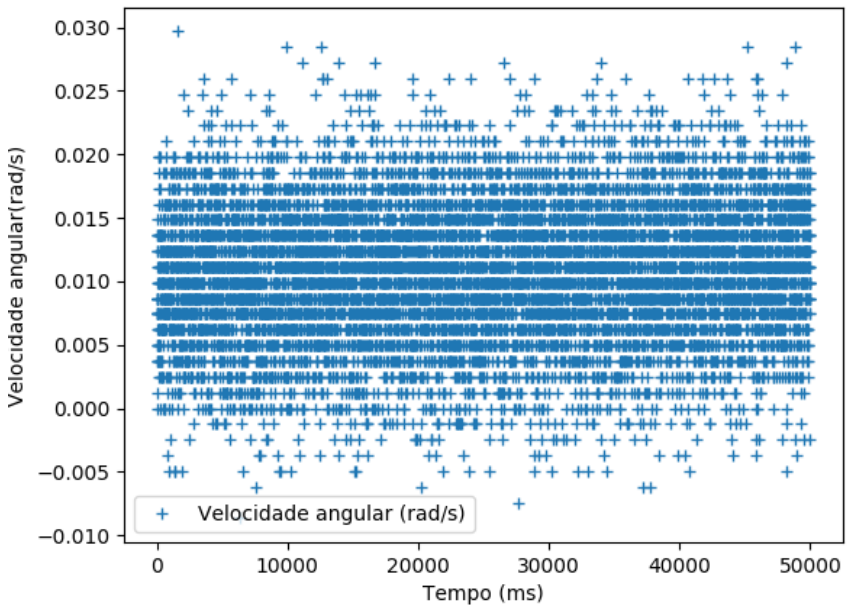
E no acelerômetro, com o valor máximo de 39.2266 m/s^2, o valor medido em m/s^2 é dado por:

$$a = valor_{medido} * \frac{39.2266m/s^2}{32767} \tag{2.40}$$

2.5 Calibração do giroscópio

Medindo a velocidade angular através do giroscópio com o robô parado resulta no seguinte gráfico:

Figura 1 – Velocidade angular medida pelo giroscópio com robô parado

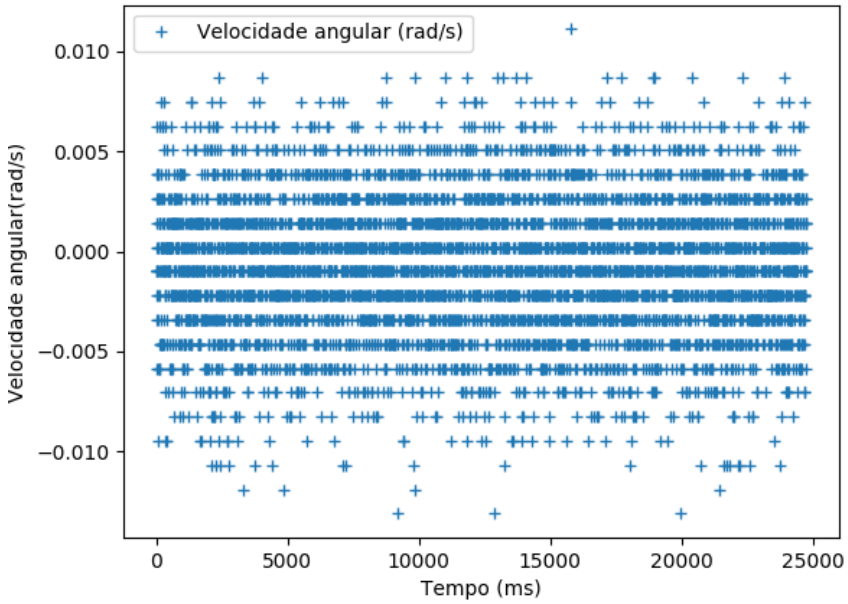


O valor médio não é zero, indicando a existência de um offset. Ao medir vários valores do offset e fazer média entre eles, o offset estimado é de 0.01064 rad/s. O offset é corrigido pela equação:

$$\omega = \omega_{\text{medido}} - \text{offset} \tag{2.41}$$

Entretanto, o offset não é sempre o mesmo, e por isso esta correção deve ser sempre realizada antes de utilizar o robô. Isso pode ser feito realizando amostras e calculando a média delas imediatamente após o robô ser ligado, enquanto ele está parado. Medindo ω novamente, utilizando a correção proposta:

Figura 2 – Velocidade angular com correção de offset, com o robô parado



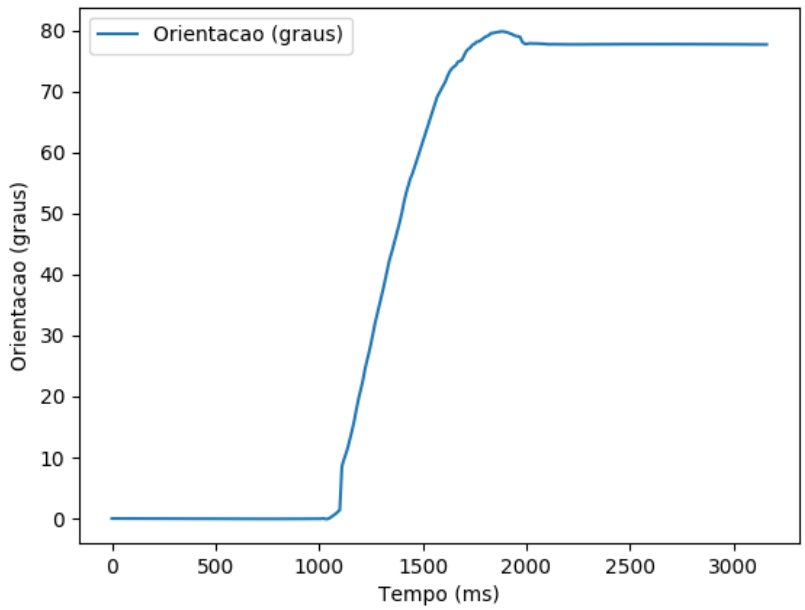
A média das leituras é aproximadamente 0 rad/s, o que indica que a correção foi bem sucedida.

Para testar o giroscópio, a velocidade angular medida por ele foi utilizada para calcular orientação, utilizando a equação:

$$\theta_t = \theta_{t-1} + (\omega_t - \text{offset}) * \Delta t \tag{2.42}$$

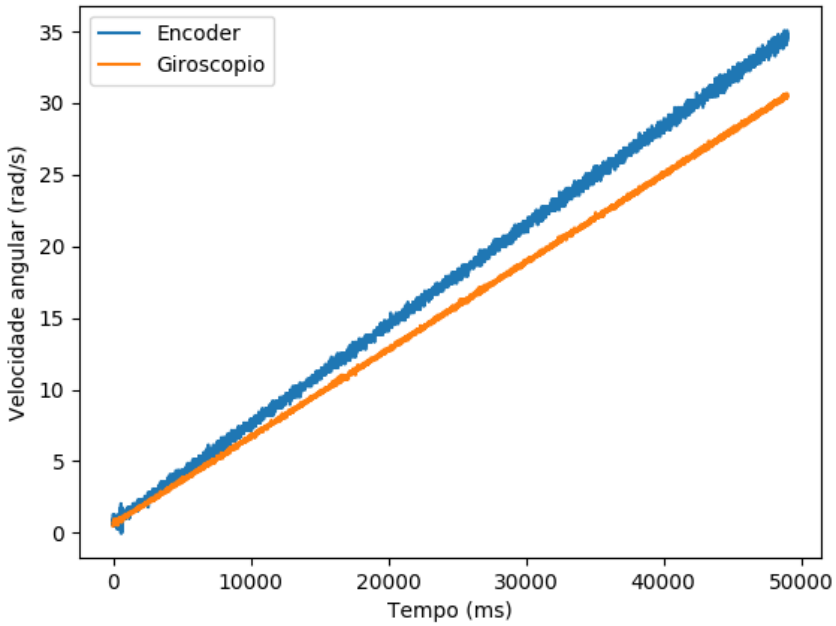
Ao girar o robô manualmente em 90°, foi observado o seguinte resultado:

Figura 3 – Orientação calculada ao girar 90°



O ângulo calculado no final do movimento foi de apenas 77 graus, o que indica que há alguma distorção na velocidade angular medida pelo giroscópio. Para descobrir que tipo de distorção está presente, os valores de velocidade angular foram medidos através do giroscópio e dos encoders enquanto o robô girando com velocidade crescente, com velocidade das rodas entre 0m/s e 1.15m/s . O resultado obtido foi:

Figura 4 – Velocidade angular medido por giroscópio e encoder



Pode-se perceber que há uma distorção linear no giroscópio, que resulta em medições menores que as reais. A distorção pode ser causada por imperfeições no sensor, eixo z não alinhado perfeitamente com eixo do robô, entre outros.

Para corrigir o problema, foi feito regressão linear nos valores medidos dos dois sensores, o que resultou nos seguintes coeficientes lineares:

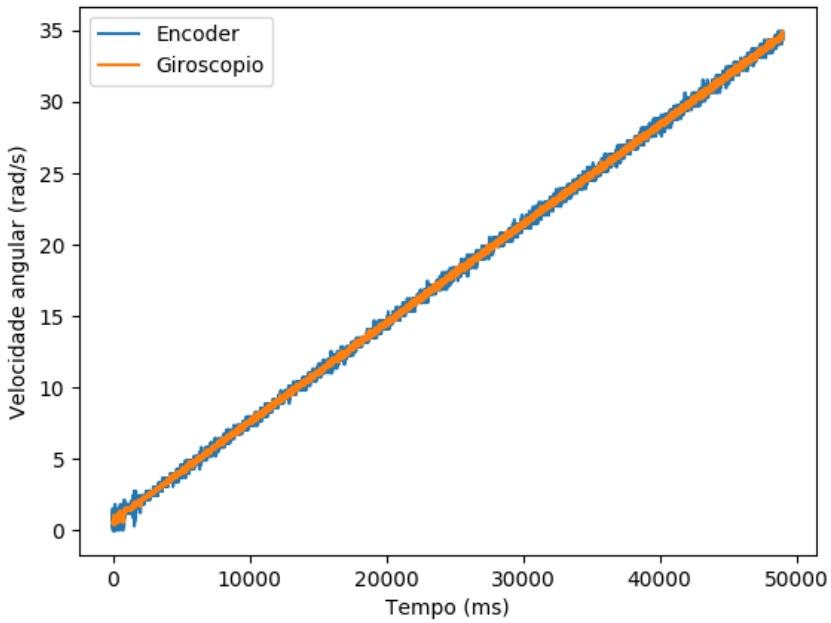
Sensor	Coeficiente linear
Encoder	0.00346722
Giroscópio	0.00298599

Estes valores foram então utilizados para corrigir a escala do giroscópio, calculando a velocidade angular correta por:

$$\omega = \omega_{medido} * \frac{coeficiente_{encorder}}{coeficente_{giroscopio}} \tag{2.43}$$

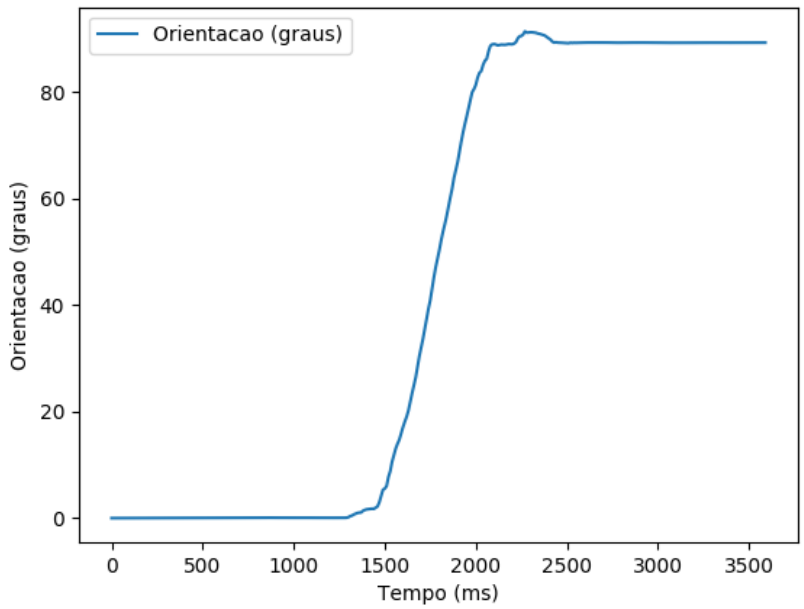
Repetindo a mesma medição, mas calculando a velocidade angular utilizando a correção definida anteriormente, resulta em:

Figura 5 – Velocidade angular medido por giroscópio e encoder, após a correção



As medidas dos dois sensores se alinharam, demonstrando que a distorção no giroscópio foi corrigida. Ao girar o robô em 90 graus, calculando o ângulo através das medições corrigidas do giroscópio, resultou no gráfico:

Figura 6 – Orientação calculada ao girar 90°, com correção da velocidade angular



Desta vez o sensor foi capaz de obter o ângulo correto, o que indica que o giroscópio está funcionando corretamente e pode ser incorporado na estimativa de ângulo da fusão de sensores.

2.6 Calibração do magnetômetro

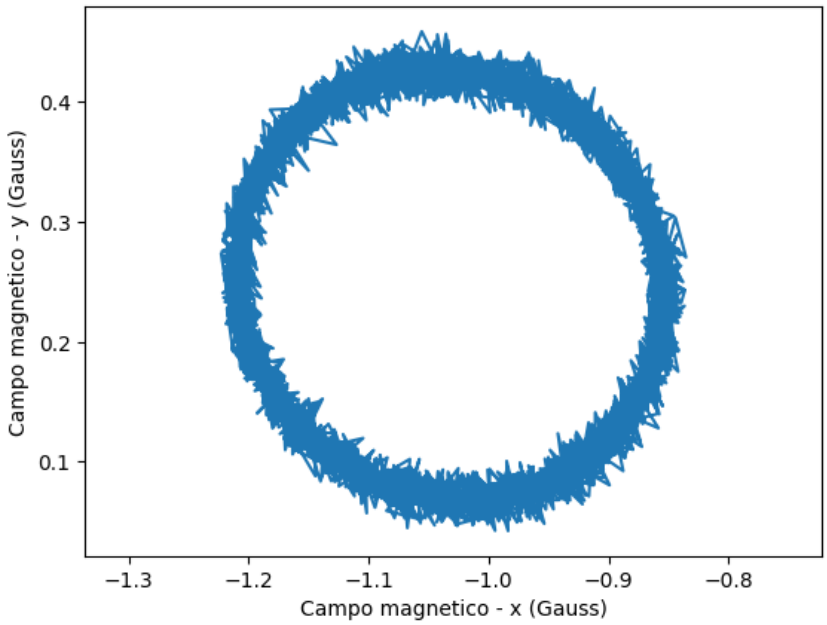
O valor medido pelo magnetômetro é a intensidade do campo magnético da terra, sendo necessário uma forma de calcular a orientação do robô a partir destas medições. Além disso, podem haver distorções no valor medido devido a partes metálicas e magnéticas no robô e devido a imperfeições no sensor, que devem ser corrigidas antes de serem utilizadas. Para calcular o ângulo, é utilizada a equação:

$$\theta = \arctan \frac{-B_y}{B_x} \tag{2.44}$$

Em que B_x e B_y são as intensidades do campo magnético nos eixos x e y.

Para observar possíveis distorções nos valores medidos, o robô foi girado continuamente em torno do próprio eixo, enquanto o magnetômetro media a intensidade do campo magnético nos eixos x e y. Ao traçar os valores dois dois eixos, o resultado é o seguinte:

Figura 7 – Elipse formada pelas medidas nos eixos x e em y do magnetômetro



Através do gráfico pode-se perceber que o centro do círculo não está na origem, o que indica a presença de um offset. Também é visível que o eixo y está um pouco mais alongado que o eixo x, exigindo a normalização dos valores. O valor real do campo magnético foi modelado como:

$$B = \frac{B_{medido}}{B_{máximo}} - \text{offset} \tag{2.45}$$

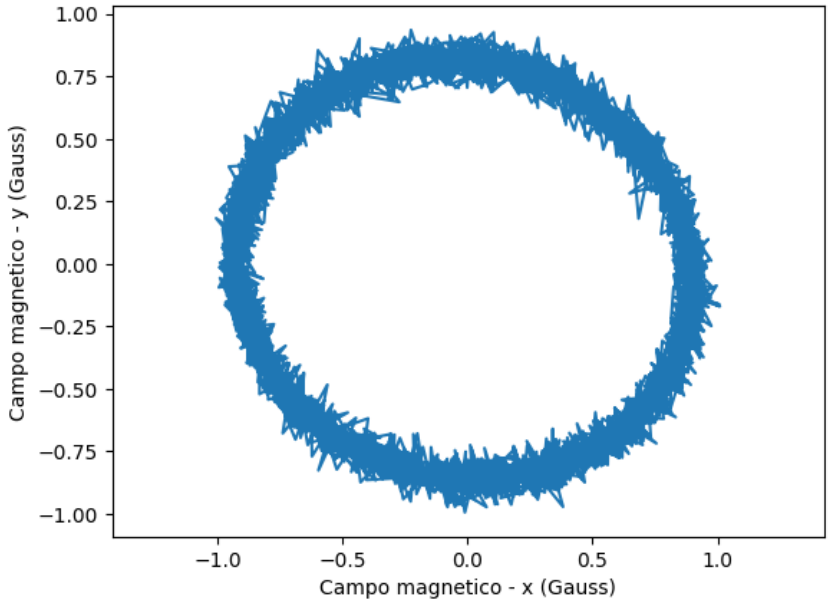
O offset elimina o efeito das partes metálicas e magnéticas no robô, enquanto a divisão pelo valor máximo normaliza as medidas, corrigindo diferenças de escala entre os eixos x e y do sensor. Desta forma, os valores medidos estão sempre entre 0 e 1. Zero significa que o eixo está perpendicular ao campo magnético da terra, e 1 significa que está alinhado com o campo.

O valor máximo e média de cada eixo no teste anterior é:

Eixo	Valor máximo	média
x	0.1877037	-1.02949175
y	0.1946520	0.24693936

O offset de cada eixo é a média dos valores medidos. Ao realizar o mesmo experimento, aplicando a correção descrita acima, foi obtido o seguinte gráfico:

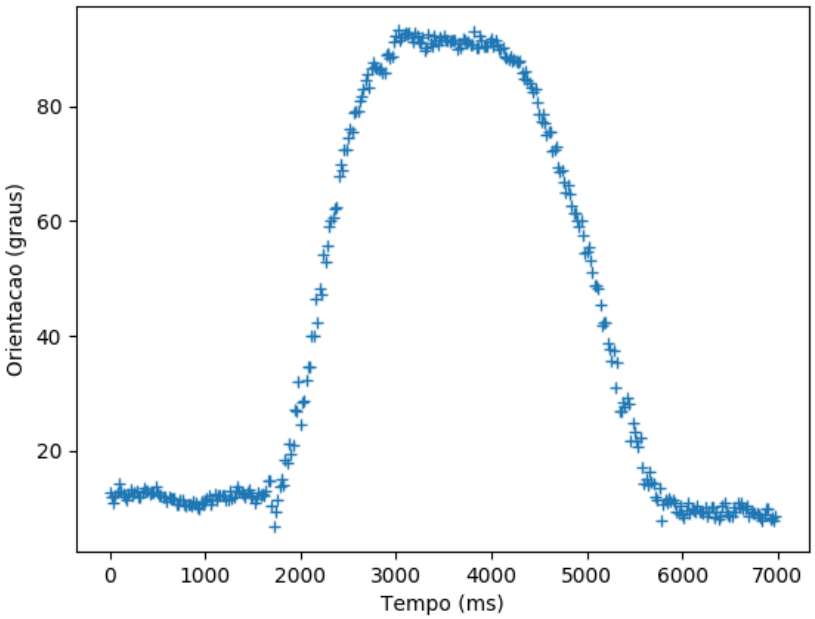
Figura 8 – Elipse formada pelas medidas nos eixos x e em y do magnetômetro, após correções



O raio do círculo é 1 e o centro está na origem, indicando que a correção foi bem sucedida.

Ao mover o robô aproximadamente a 90° e voltá-lo a orientação original, a orientação medida foi:

Figura 9 – Orientação do robô medida pelo magnetômetro



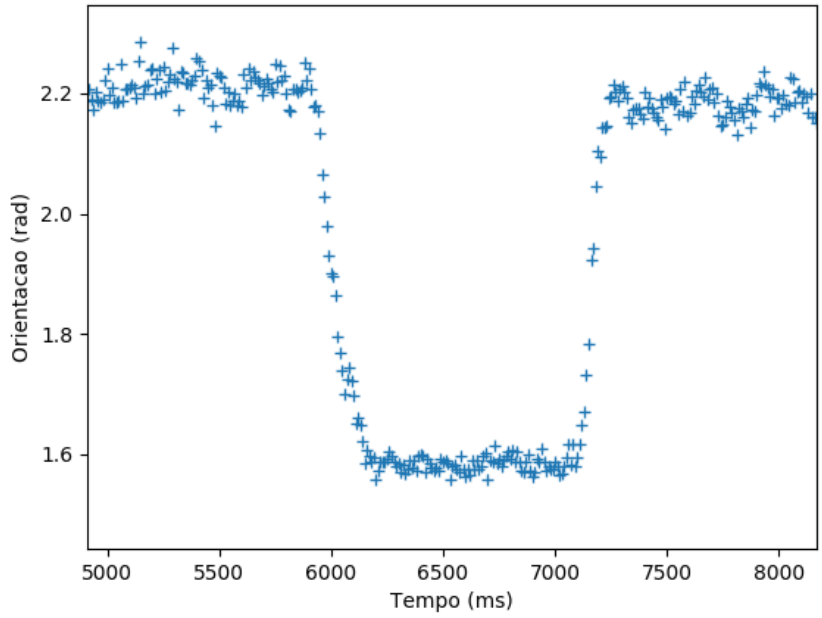
O angulo foi medido corretamente, confirmando que o modelo utilizado é capaz de

corrigir os problemas detectados.

2.7 Offsets do magnetômetro

Uma das desvantagens dos magnetômetros é que, como eles medem a intensidade do campo magnético, qualquer objeto metálico ou magnético interfere em suas leituras. Isso se torna um problema quando o robô deve ser utilizado próximo de outros robôs, como é o caso da aplicação estudada neste trabalho. Por isso, é necessário uma forma de impedir que estas interferências resultem em um erro na estimativa da orientação do robô. Para observar interferências, foi medido a orientação calculada por dados do magnetômero do robô parado, enquanto outro robô foi aproximado e afastado dele.

Figura 10 – Orientação do robô medida pelo magnetômetro devido a interferências



Para resolver este problema, o offset do magnetômetro foi adicionado as variáveis de estado do robô, para que o EKF possa estimá-lo. Dessa forma, o modelo de medida para leituras do magnetômetro passa a ser:

$$\theta_{mag} = \theta + \theta_{offset} \tag{2.46}$$

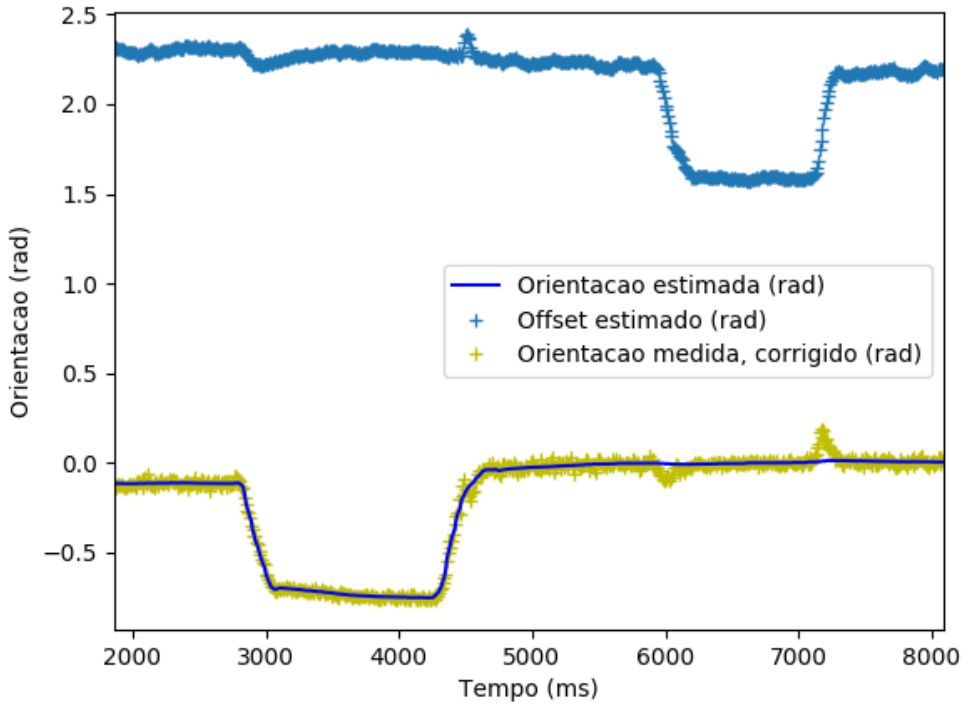
E ao ter dados da visão computacional disponíveis, o magnetômetro é lido e θ_{offset}, calculado pela equação e utilizado na etapa de atualização do EKF:

$$\theta_{offset} = \theta_{mag} - \theta_{visao} \tag{2.47}$$

Dessa forma, a visão computacional contribui com a estimativa do offset do magnetômetro. Com a correção de interferências externas implementada, foi estimar a orientação do robô.

No gráfico a seguir, o robô foi movimentando manualmente, e em seguida, outro robô foi aproximado dele. Foi obtido os resultados:

Figura 11 – Orientação do robô medida pelo magnetômetro, com e sem correção



Nesse gráfico, pode-se observar que apesar de as leitudas do magnetômetro terem sido afetadas pelo outro robô, a orientação estimada continuou correta, ignorando as influências externas.

2.8 Correções no acelerômetro

Deixando o robô parado, o acelerômetro mede a aceleração da gravidade:

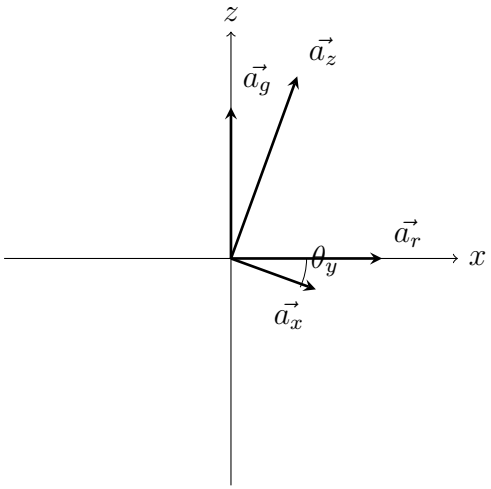
eixo	aceleração (m/s^2)
a_x	-0.525424
a_y	0.402307
a_z	9.774617
$\sqrt{a_x^2 + a_y^2 + a_z^2}$	9.796992

O robô pode se inclinar no eixo y, o que influencia nas medidas de aceleração dos eixos x e z. Inclinando o robô ao máximo, a aceleração medida foi:

eixo	aceleração (m/s^2)
a_x	0.105287
a_y	0.406665
a_z	9.792763
$\sqrt{a_x^2 + a_y^2 + a_z^2}$	9.801768

Pode-se observar que o modulo da aceleração é quase o mesmo, mas a componente x foi muito alterada devido a mudança no angulo entre o vetor da aceleração da gravidade e o eixo medido pelo sensor. Isso significa que o eixo x não pode ser utilizado diretamente como aceleração do robô, e que também que esse erro não é constante durante o movimento do robô. É necessário levar em consideração a inclinação do robô para corrigi-lo.

Os vetores relevantes podem ser modelados como:



Em que a_g é a aceleração da gravidade, a_r é a aceleração do robô, e a_x e a_z são as acelerações medidas pelos eixos x e z do robô. Calculando as projeções de a_g e a_r nos eixos a_x e a_z resulta nas equações:

$$a_x = a_r * \cos\theta_y - ag * \sin\theta_y \tag{2.48}$$

$$a_z = a_r * \sin\theta_y + ag * \cos\theta_y \tag{2.49}$$

E também vale a relação:

$$\sqrt{a_x^2 + a_z^2} = \sqrt{a_r^2 + a_g^2} \tag{2.50}$$

Mantando o robô parado para que a_r seja 0, é calculado a aceleração da gravidade pela equação:

$$a_g = \sqrt{a_x^2 + a_z^2} \tag{2.51}$$

E a inclinação do robô é obtida por:

$$\theta_y = \arccos\frac{a_z}{a_g} \tag{2.52}$$

Esses dois valores são calculados sempre que o robô é ligado, para que possam ser utilizados posteriormente para remover o efeito da gravidade em a_x, permitindo obter a_r. Como a inclinação θ_y não permanece constante durante todo o movimento do robô, ela é estimada pelo EKF através das medidas do eixo y do giroscópio:

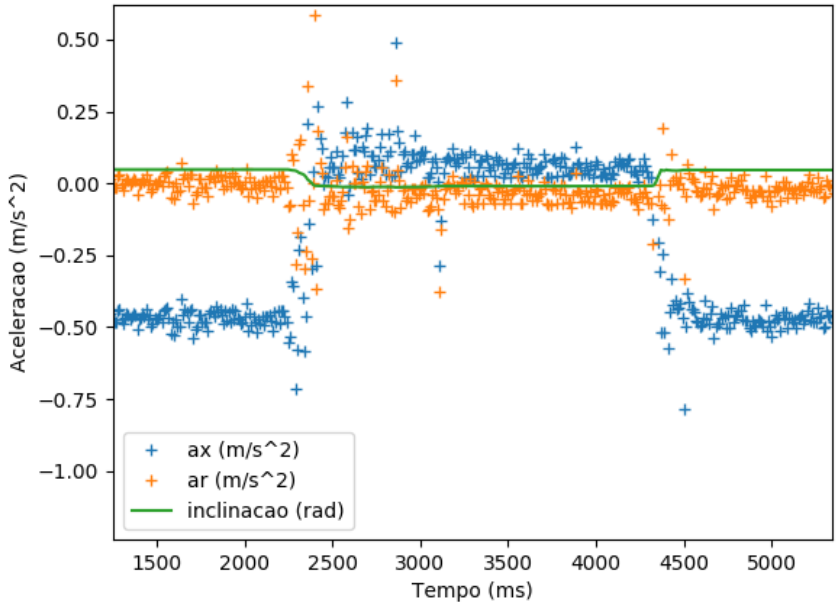
$$\theta_{y\,t} = \theta_{y\,t-1} + \omega_y * \Delta t \tag{2.53}$$

A partir de a_x medido pelo acelerômetro, θ_y estimado pelo EKF e a_g medido anteriormente, a aceleração real do robô sem a aceleração da gravidade é calculada:

$$a_r = \frac{a_x - a_g * \sin\theta_y}{\cos\theta_y} \tag{2.54}$$

Para testar a correção da componente da gravidade, os valores de a_x, a_r e θ_y foram medidos mantendo o robô parado, inclinando o robô manualmente:

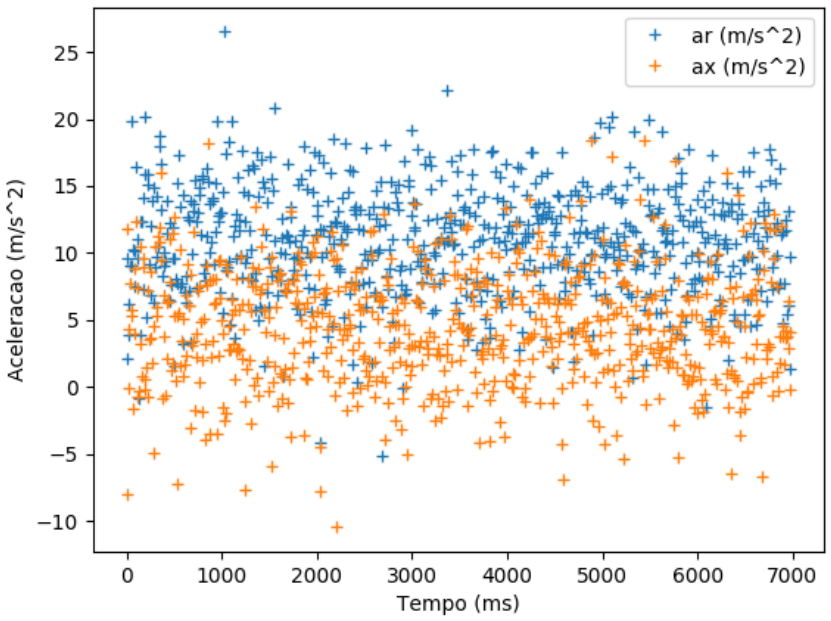
Figura 12 – Inclinação do robô e aceleração com e sem correção



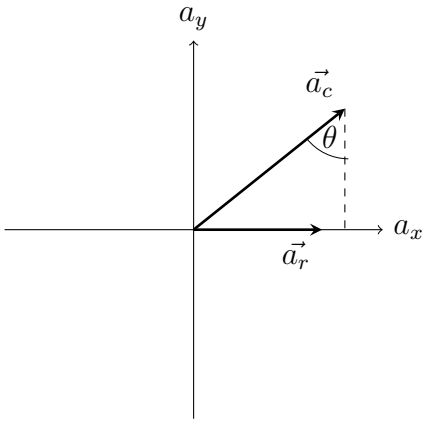
Como é visível no gráfico, enquando houve grandes alterações em a_x, a aceleração a_r não é muito influenciada pela inclinação do robô.

Ainda há outras acelerações que modificam o valor lido pelo giroscópio. Ao girar o robô em torno do próprio eixo com velocidade angular de 20 rad/s, o acelerômetro mede uma aceleração adicional nos eixos x e y:

Figura 13 – Aceleração em x e y com rotação constante a 20 rad/s



Os valores medidos de a_x e a_y tem média de 10.8527 e 4.45632 m/s^2, respectivamente. Estes valores são componentes da aceleração centrípeta que surgiu devido ao giro do robô. Para saber a aceleração linear na direção do movimento do robô, é preciso removê-la. As acelerações nos eixos x e y foram modeladas como:



em que a_c é a aceleração centrípeta e a_r é a aceleração real do robô. As acelerações no eixo x e y podem ser obtidas pelas equações:

$$a_x = a_c * \sin\theta + a_r \tag{2.55}$$

$$a_y = -a_c * \cos\theta \tag{2.56}$$

A aceleração centrípeta pode ser escrita em função da velocidade angular:

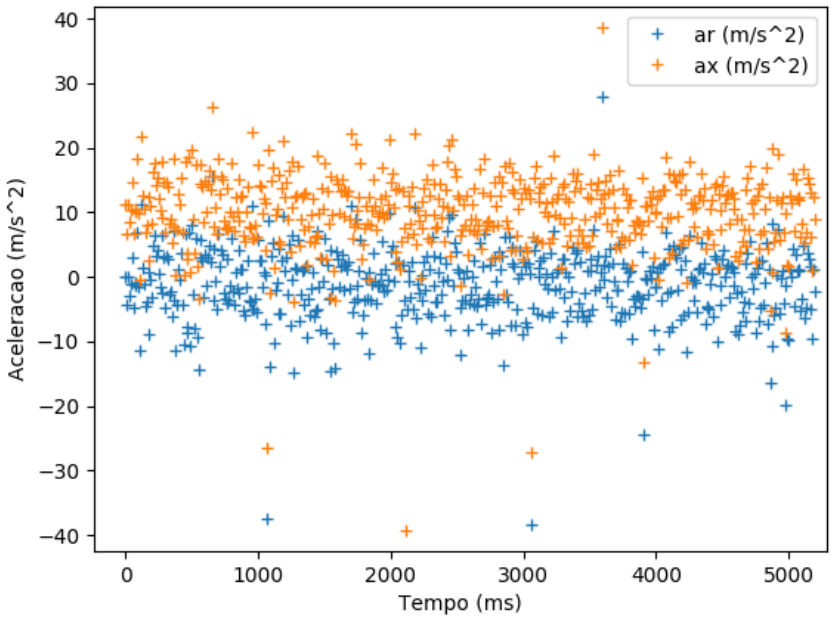
$$a_c = \omega^2 * r \tag{2.57}$$

E ω é medido pelo giroscópio. Dessa forma, a aceleração do robô é obtida por:

$$a_r = a_x - \omega^2 * r * \sin\theta \tag{2.58}$$

Girando o robô novamente a velocidade angular de 20 rad/s, medindo a aceleração no eixo x, com e sem correção da aceleração centrípeta:

Figura 14 – ax e ar com rotação constante a 20 rad/s

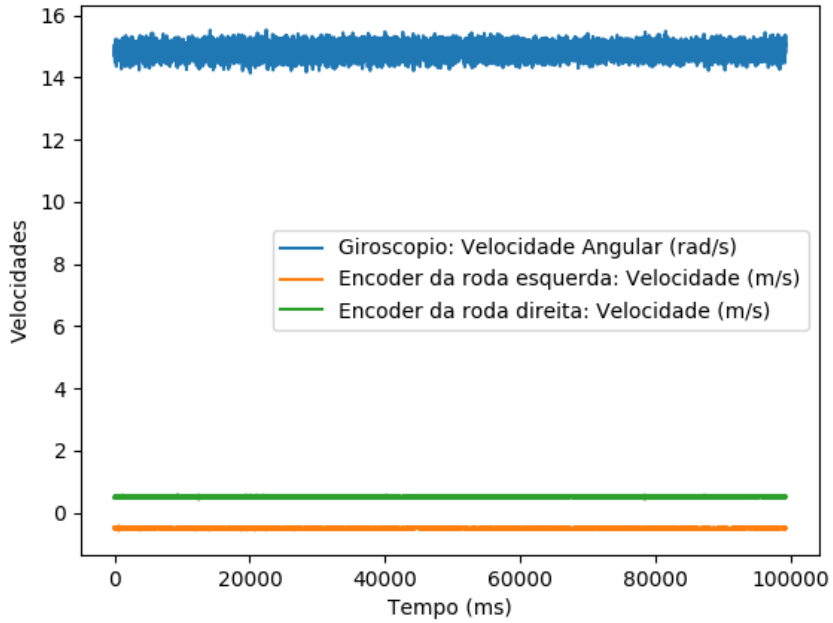


Mostrando que o efeito da aceleração centrípeta na medida da aceleração do robô foi reduzido.

2.9 Medida de covariância do encoder e giroscópio

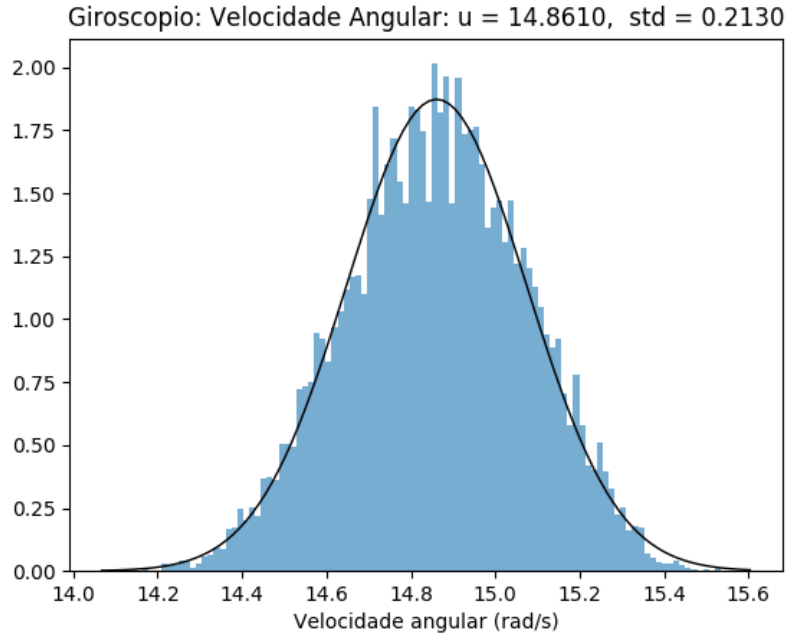
Para medir a covariância do encoder e giroscópio, aplicou-se 0.5m/s (velocidade típica) na roda direita e -0.5m/s na roda esquerda, o fazendo girar. O robô mediu a velocidade das rodas através do encoder e a velocidade angular através do giroscópio. O resultado foi:

Figura 15 – Velocidade angular e velocidades de cada roda



Utilizando as ferramentas do matplotlib (7) para montar um histograma e o scipy (8) para fazer o fitting em gaussiana dos valores medidos, pode-se observar o ruído do giroscópio:

Figura 16 – Gaussiana do giroscópio



O que indica que o ruído está muito próximo de um ruído gaussiano. Os histogramas das medidas de velocidade do encoder da roda direita e da esquerda são:

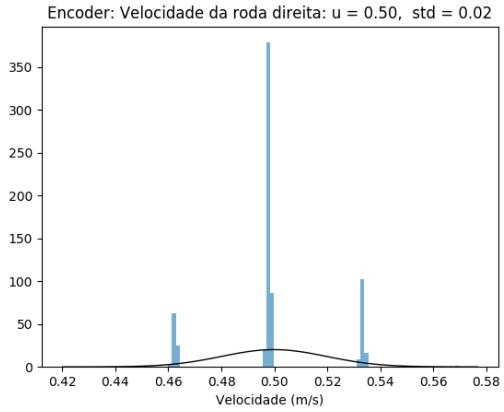


Figura 17 – Gaussiana do encoder da roda direita

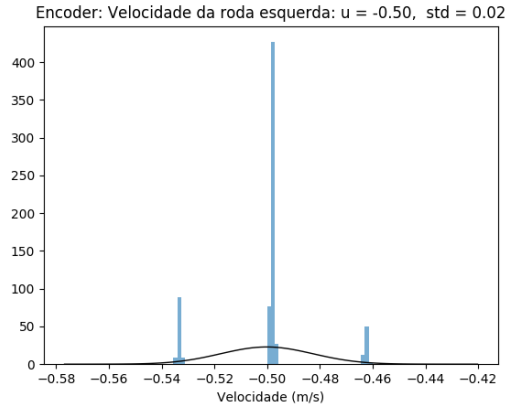


Figura 18 – Gaussiana do encoder da roda esquerda

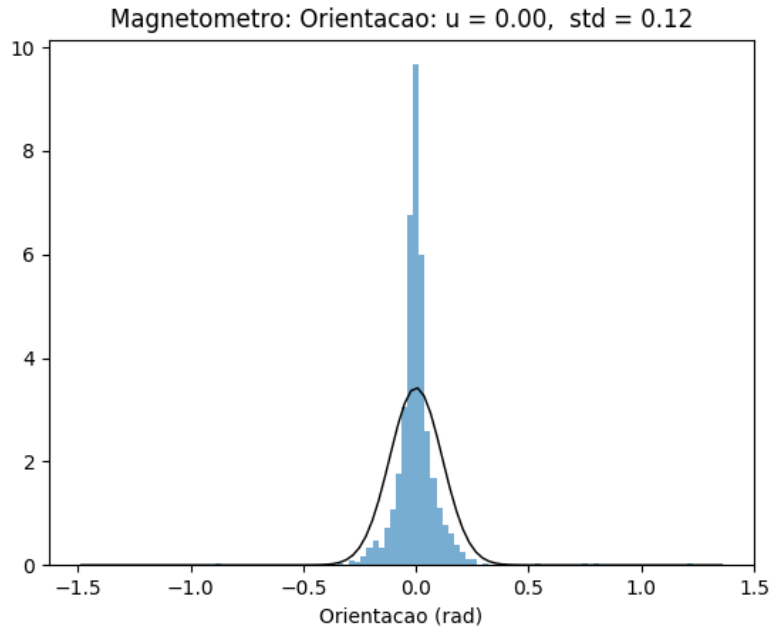
Neste gráfico pode ser visto que há uma baixa resolução nas medidas do encoder. Apesar destas falhas, o ruído do sensor ainda pode ser modelado por uma gaussiana. A tabela a seguir mostra a covariância de cada sensor:

Sensor	Covariância
Giroscópio	4.53759819e-02
Encoder da roda esquerda	3.05991867e-04
Encoder da roda direita	3.92135776e-04

2.10 Medida de covariância do magnetômetro

Para obter a covariância do magnetômetro, foram realizadas diversas medida com o robô parado. Utilizando novamente o matplotlib e o scipy para fazer o fitting em gaussiana dos valores medidos e montar o histograma resulta em:

Figura 19 – Gaussiana do magnetômetro



O magnetômetro também possui ruído aproximadamente gaussiano. A covariância encontrada foi 0.01356513.

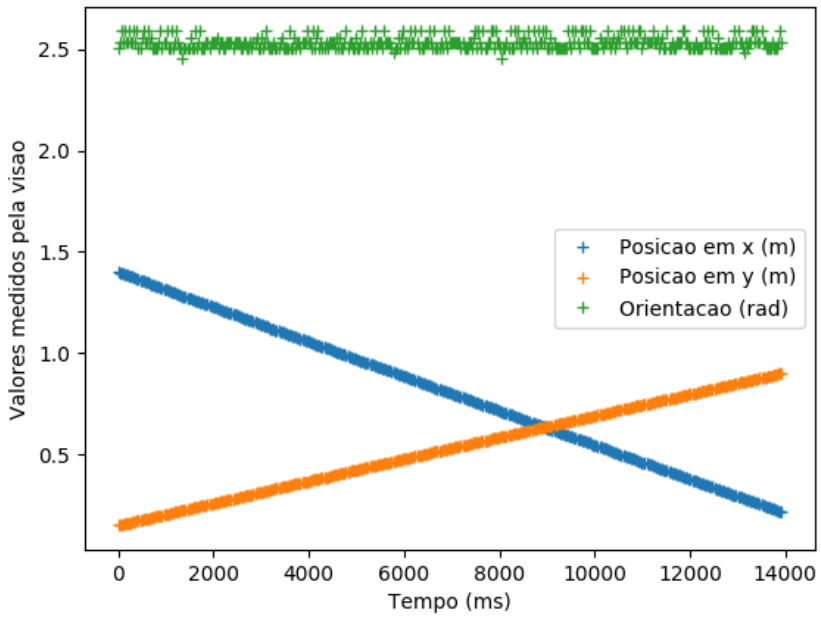
Utilizando os valores medidos dos encoders, giroscópio e magnetômetro, foi formada a matriz de covariância R:

$$\begin{bmatrix} 0.01356513 & 0 & 0 & 0 \\ 0 & 4.53759819e-02 & 0 & 0 \\ 0 & 0 & 3.05991867e-04 & 0 \\ 0 & 0 & 0 & 3.92135776e-04 \end{bmatrix}$$

2.11 Covariância dos dados da visão computacional

Aplicou-se 0.15m/s nas duas rodas para fazer o robô andar lentamente em linha reta em uma distância de aproximadamente 1 metro. Durante o movimento, foram armazenados os valores medidos pela visão computacional de posição (x, y) em metros e orientação (θ) em radianos. Os valore medidos podem ser vistos no gráfico:

Figura 20 – Posição e orientação medidos pela visão computacional, enquanto robô anda em linha reta



Para diferenciar a variação devido ao movimento do robô da variação devido ao ruído, foi calculado a regressão linear de x e y e, utilizando a equação de reta encontrada, foi subtraído o valor medido do valor ideal, mantendo apenas o ruido gaussiano branco. O histograma com fitting de gaussiana em x, y e θ são:

Figura 21 – Ruido de posição no eixo x da visão computacional

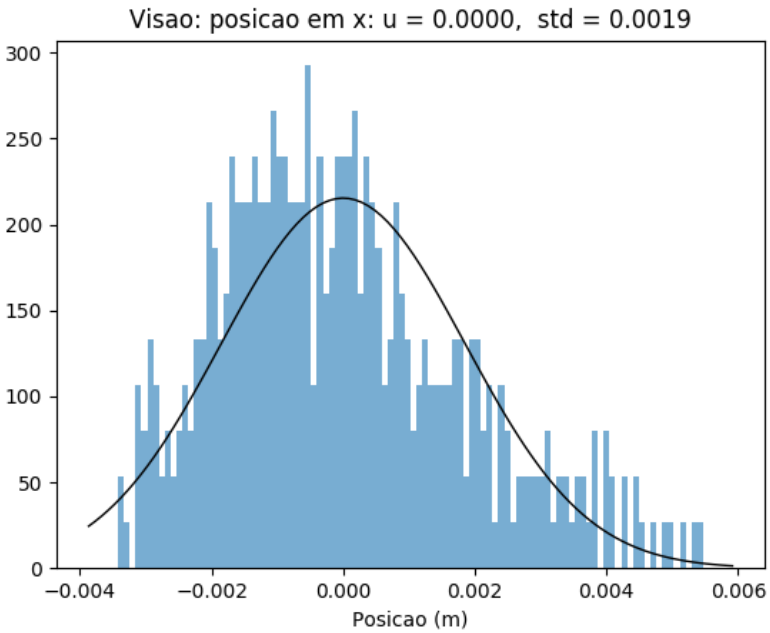


Figura 22 – Ruido de posição no eixo y da visão computacional

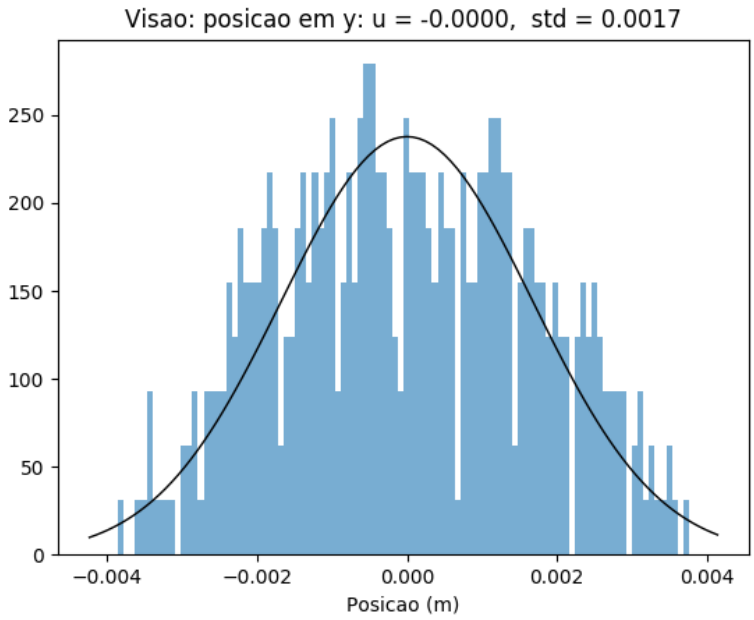
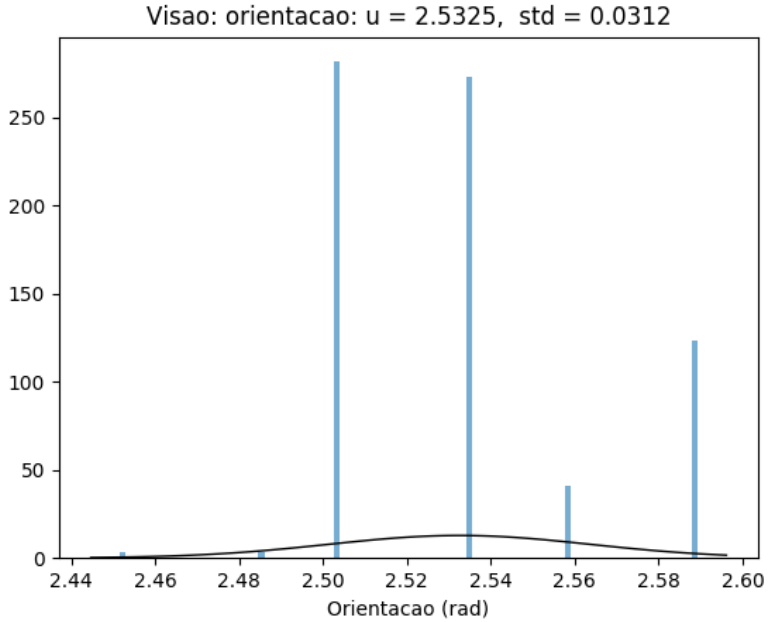


Figura 23 – Ruido de orientação da visão computacional



Estes resultados indicam que não há grande resoluçao na medição de orientação, mas as de posição foram muito próximas de uma gaussiana. Já a coviariancia do offset do magnetômetro é a soma da covariancia da visão somada a covariancia do magnetômetro,

resultando em 0.0145424. Utilizando estes testes, foi obtida a matriz de covariança R_v:

$$\begin{bmatrix} 3.44048681e-06 & 0 & 0 & 0 \\ 0 & 2.82211659e-06 & 0 & 0 \\ 0 & 0 & 9.77316323e-04 & 0 \\ 0 & 0 & 0 & 0.0145424 \end{bmatrix}$$

2.12 Matriz de ruído de processo Q_t

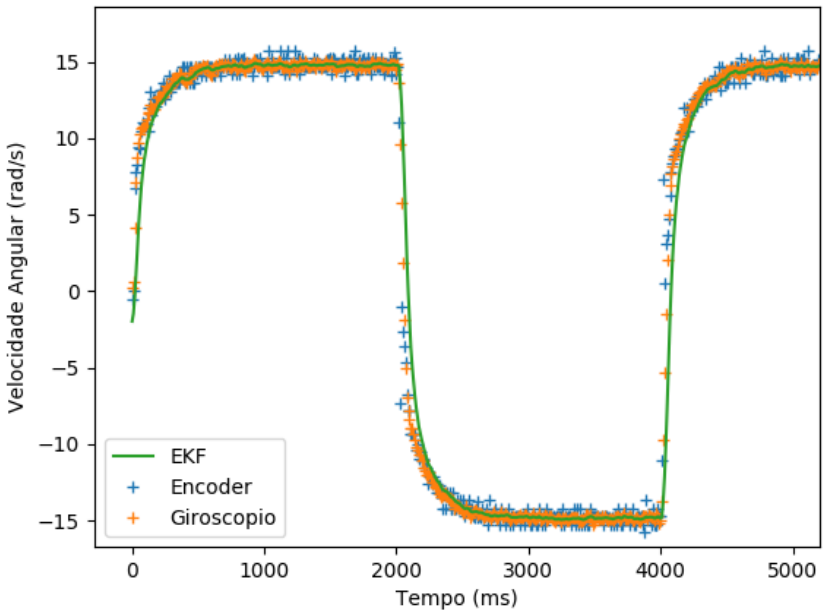
A matriz de ruído de processo Q foi modelada como:

$$\begin{bmatrix} Tk_x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & Tk_y & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Tk_\theta & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & Tk_v & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & Tk_\omega & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & Tk_{\theta_{offset}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Tk_{\theta_y} \end{bmatrix}$$

Onde T é o período de amostragem e k_i são constantes que devem ser ajustadas. Foi utilizado o período de amostragem porque quanto maior o tempo entre as iterações, maior é o erro no modelo.

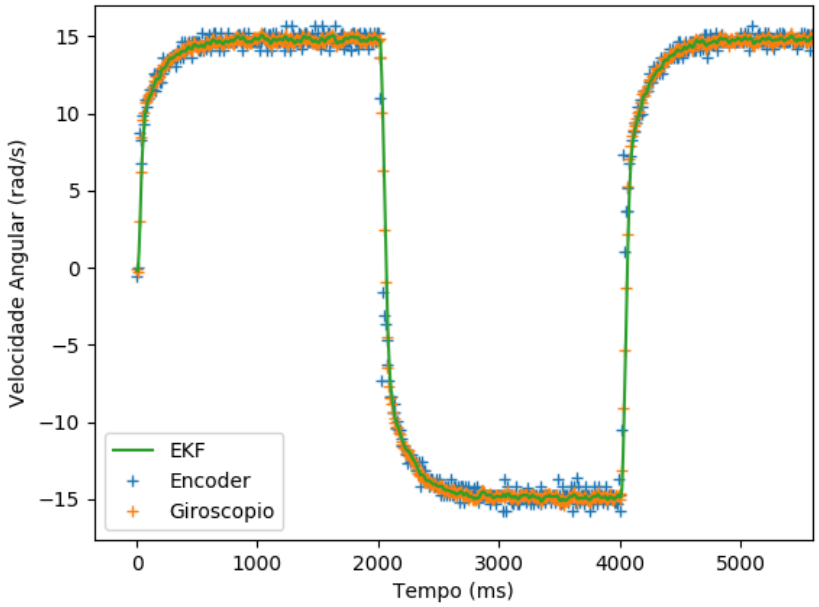
Quando a aceleração angular era modelada como constante, foi observado problemas com variações rápidas em ω. Para observar os efeitos de k_ω, os valores de velocidade angular foram medidas aplicado alternadamente 0.6m/s e -0.6m/s em cada roda. Com $k_\omega = 0.01$, a estimativa do EKF é:

Figura 24 – Velocidade angular estimada e medida, $k_\omega = 0.01$



Pode-se perceber que o filtro não conseguiu acompanhar bem as variações bruscas na velocidade angular. Este problema foi corrigido no modelo de aceleração constante utilizando um valor maior da constante, de $k_\omega = 0.1$:

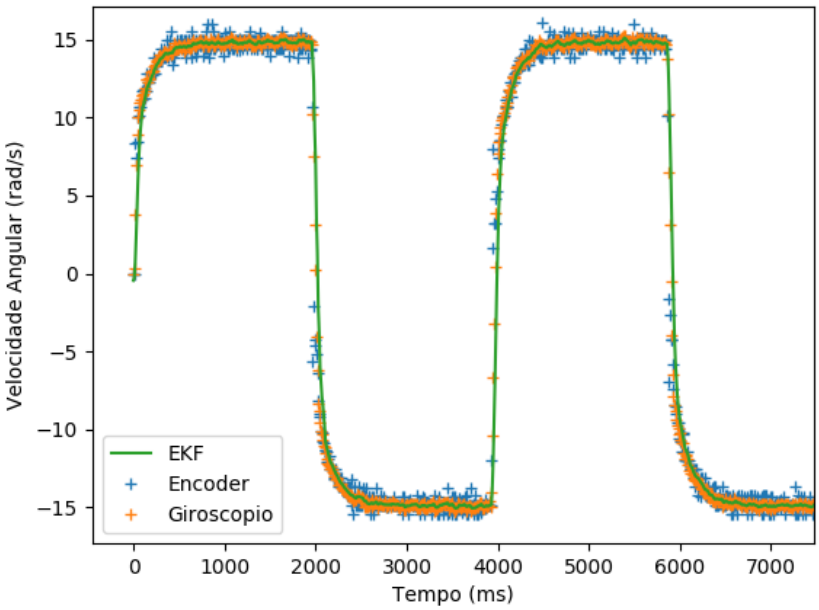
Figura 25 – Velocidade angular estimada e medida, $k_\omega = 0.1$



Neste novo teste, o ajuste no k_ω conseguiu melhorar muito a estimativa da velocidade angular mesmo durante as variações bruscas, por fazer o EKF confiar mais nas novas medidas e menos no valor predito a partir da estimativa anterior.

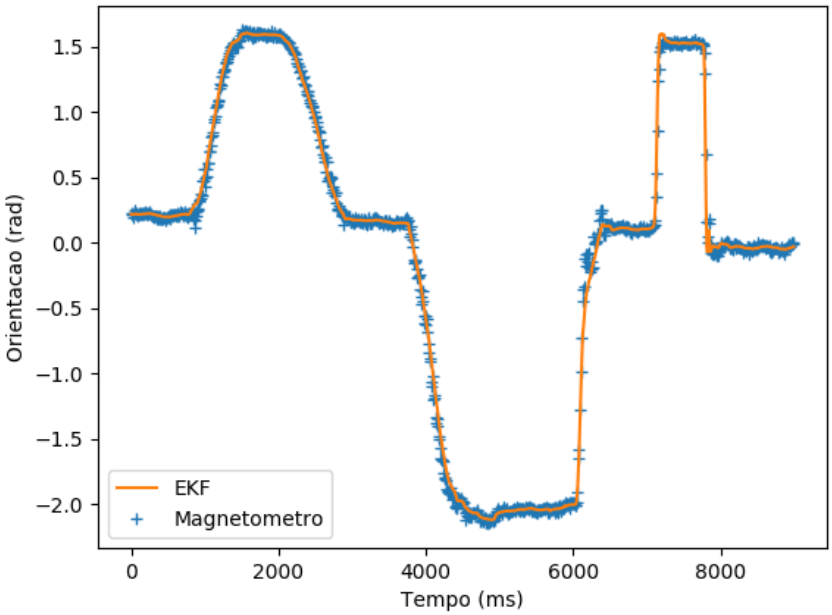
Quando a aceleração foi levada em consideração no modelo utilizando a derivação numérica das velocidades das rodas, os problemas foram corrigidos. Mesmo para valores baixos de k_ω, como 0.001, a estimativa conseguiu acompanhar corretamente os sensores:

Figura 26 – Velocidade angular estimada e medida, $k_\omega = 0.001$



A orientação foi bem estimada com $k_\theta = 0.001$, como pode ser vista no gráfico:

Figura 27 – Velocidade angular estimada e medida, $k_\theta = 0.001$



Neste teste, o robô foi girado manualmente por aproximadamente 90 graus, retornado para a orientação anterior, depois o movimento foi repetido na outra direção. Por fim, ele foi movimentado rápidamente, e o EKF continuou realizando boas estimativas.

As constantes $Tk_{\theta_{offset}}$ e Tk_{θ_y} foram ajustadas para 0.005 e 0.0001 respectivamente por tentativa e erro, ajustando seus valores até que as estimatimativas ficassem adequadas.

2.13 Unscented Kalman Filter

Unscented Kalman Filter (9), ou UKF, é uma alternativa ao EKF que lineariza melhor o sistema. Em vez de linearizar o sistema utilizando aproximação de Taylor de primeira ordem como o EKF faz, o UKF gera vários pontos (chamados de pontos sigma) próximos ao valor real, os passa pelo modelo do sistema e gera uma estimativa final, aplicando pesos a cada um desses pontos. Dessa forma, o UKF propaga a covariancia pelo modelo não linear alcançando precisão de terceira ordem para qualquer não linearidade.

Assim como o EKF, o UKF tem uma etapa de prediçao. Primeio, os pontos sigma são gerados:

$$X_0 = x \tag{2.59}$$

$$X_i = x + (\sqrt{(L+\lambda)\Sigma})_i, i = 0, .., L \tag{2.60}$$

$$X_i = x - (\sqrt{(L+\lambda)\Sigma})_{i-L}, i = L+1, .., 2L \tag{2.61}$$

Em que X é uma matriz de tamanho 2L + 1 x L, x é o vetor de variáveis de estado e X_i é a i-ésima coluna de X. Depois a predição é calculada para cada ponto sigma:

$$X_t = F(X_{t-1}) \tag{2.62}$$

E então os pontos sigmas preditos são utilizados para calcular a nova covariancia e as novas variáveis de estado:

$$x_t = \sum_{i=0}^{2L} W_i * X_{i,t} \tag{2.63}$$

$$\Sigma = \sum_{i=0}^{2L} W_i * [X_{i,t} - x_t][X_{i,t} - x_t]^T \tag{2.64}$$

A etapa de atualização é semelhante. Os pontos sigmas são gerados e calcula-se as leituras preditas através do modelo de medidas:

$$Y_t = H(X_t) \tag{2.65}$$

$$y_t = \sum_{i=0}^{2L} W_i * Y_{i,t} \tag{2.66}$$

E depois é calculado as variáveis de estado e suas covariancias:

$$\Sigma_{y,y} = \sum_{i=0}^{2L} W_i * [Y_{i,t} - y_t][Y_{i,t} - y_t]^T \tag{2.67}$$

$$\Sigma_{x,y} = \sum_{i=0}^{2L} W_i * [X_{i,t} - x_t][Y_{i,t} - y_t]^T \tag{2.68}$$

$$K = \Sigma_{x,y}\Sigma_{y,y}^{-1} \tag{2.69}$$

$$x_t = x_t + K(y_{medido,t} - y_t) \tag{2.70}$$

$$\Sigma_t = \Sigma_t - K\Sigma_{y,y}K^T \tag{2.71}$$

As constante λ define a distancia entre os pontos sigma e o centro, e as constantes W_i são pesos aplicados a cada ponto sigma.

2.14 Comparação como o UKF

O UKF requer mais memoria do microcontrolador, já que precisa armazenar uma matriz com todos os pontos sigmas durante sua execução, enquanto o EKF só armazena as variáves de estado, que é um vetor menor ((2L+1) x L para o UKF vs Lx1 para o EKF, em que L é o numero de variáveis de estado.

Também houveram mais dificuldades de implementá-lo no microcontrolador, devido ao calculo de raíz quadrada da matriz de covariancia, que causou erros de execução se as constantes não estivessem bem ajustadas. Esses erros eram devido a matriz de covariancia deixar de ser positiva semi-definida, o que tornava impossível calcular sua raíz quadrada. Esses problemas foram resolvidos por tentativa e erro, ajustando as constantes até não houvessem mais erros.

O UKF também é computacionalmente mais pesado. O tempo total gasto calculando uma etapa inteira de predição e atualização foi maior que o EKF:

Algoritmo	Tempo (ms)
EKF	1.3
UKF	6.3

Esse peso computacional é uma desvantagem comparado ao EKF caso seja utilizado em um sistema que exija um controlador com períodos menores. O controlador do robô utilizado tem um período de 10 ms, então isso não foi um problema. Esse peso computacional também resulta em valores menores lidos de cada sensor. Isso resulta na perda de muitas leituras do giroscópio, que é capaz de obter uma nova medida a cada 0.6 ms, e do acelerômetro.

Suas estimativas foram comparadas enquanto o robô realizava um movimento circular, aplicando uma velocidade na roda direita de 0.6 m/s e na roda direita de 0.5 m/s.

Os erros em x, y e θ foram medidos a cada atualização do EKF e do UKF. A variação do erro em θ foi:

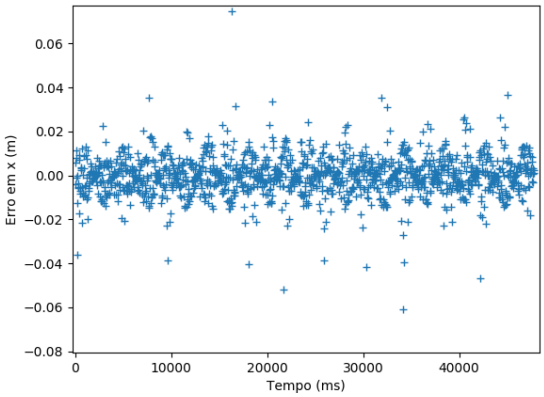


Figura 28 – Erro em x do UKF

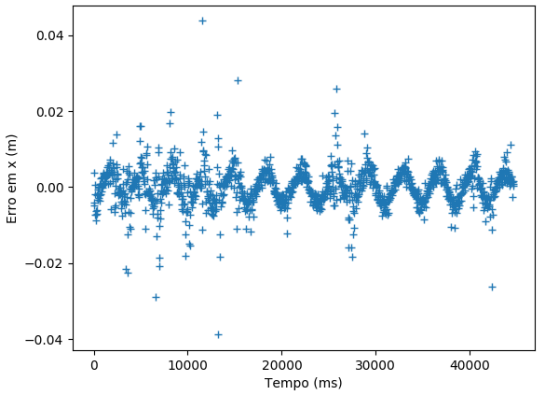


Figura 29 – Erro em x do EKF

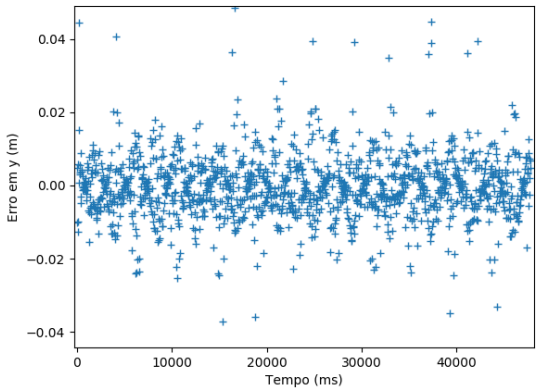


Figura 30 – Erro em y do UKF

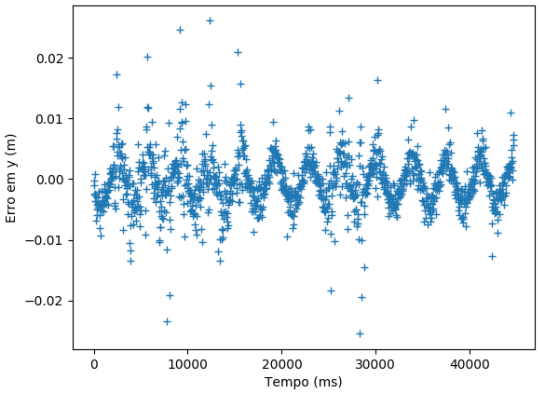


Figura 31 – Erro em y do EKF

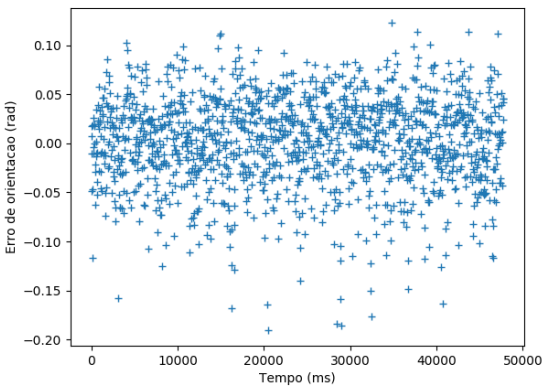


Figura 32 – Erro de orientação do UKF

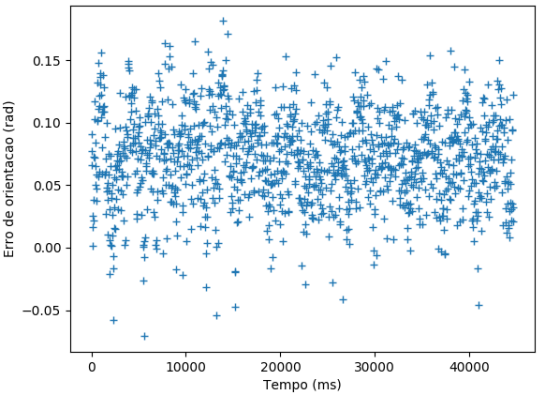


Figura 33 – Erro de orientação do EKF

A média dos erros observados foram:

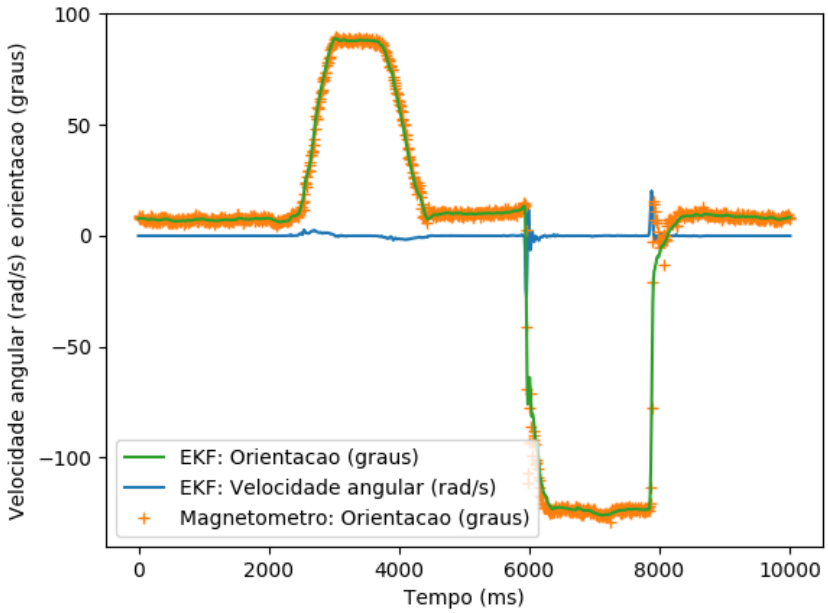
Variável	EKF	UKF
x (m)	0.0041313	0.0067911
y (m)	0.0033865	0.0064115
θ (rad)	0.0723554	0.0360838

O UKF estimou melhor a orientação, mas obteve resultados inferiores ao estimar a posição em x e y. A causa provável do UKF não ter obtido o desempenho superior em todas as estimativas é a menor taxa de atualização, que o faz corrigir suas estimativas 5 vezes menos frequentemente.

3 Resultados

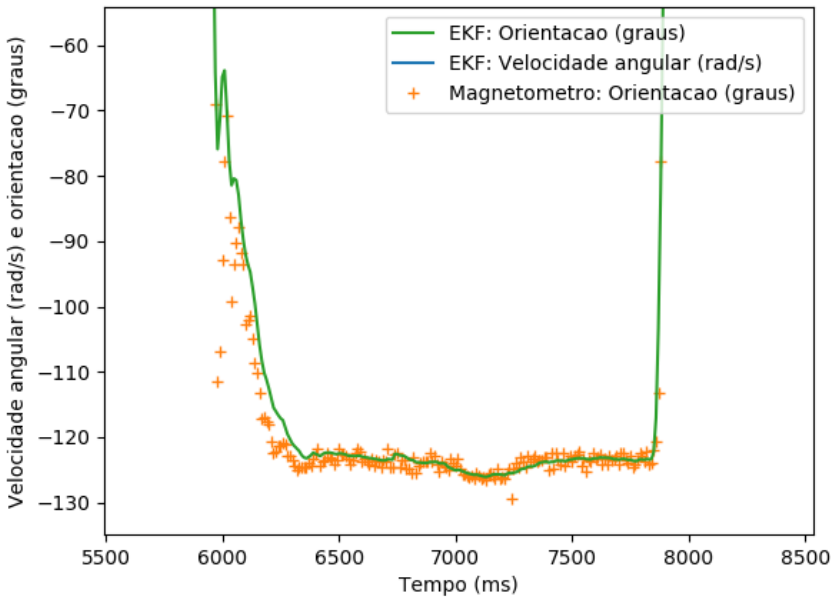
Movimentando o robô manualmente mostra que o EKF consegue estimar bem o ângulo do robô:

Figura 34 – Estimativa de orientação e velocidade angular



Na segunda vez que o robô foi girado, a velocidade angular aplicada foi maior que o giroscópio consegue medir. Apesar disso, o magnetômetro foi capaz de ajustar a estimativa de orientação corretamente. Visualizando a estimativa no segundo giro:

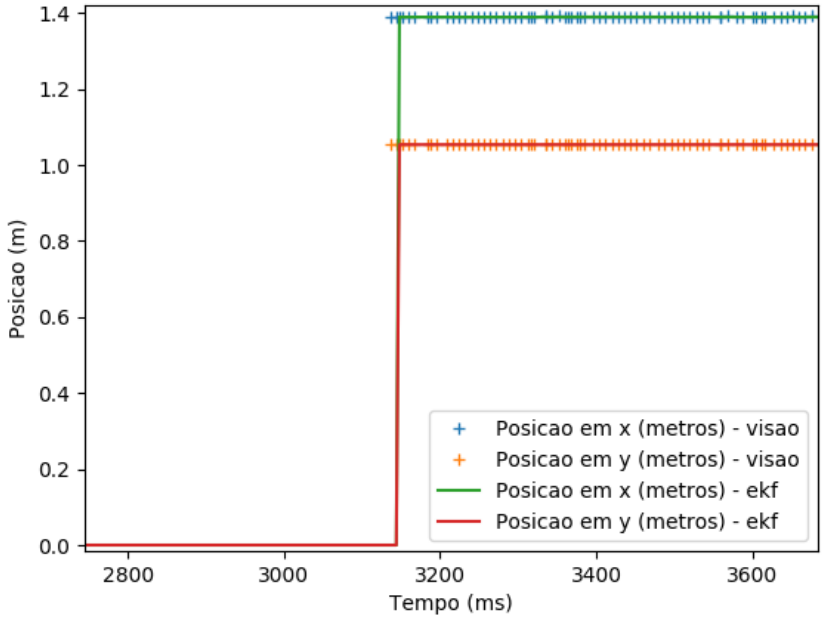
Figura 35 – Comparação dos valores medidos e estimados da orientação



É possível perceber o atrazo na estimativa devido a saturação do giroscópio, e apesar disso o valor estimado ainda converge para o valor real devido as correções do magnetômetro. Também é visível que a rejeição de ruídos resulta numa estimativa muito estável.

Ao utilizar os dados da visão computacional, o EKF converge rapidamente para a posição e orientação correta devido a alta covariancia inicial, mesmo com posição inicial em (0, 0, 0):

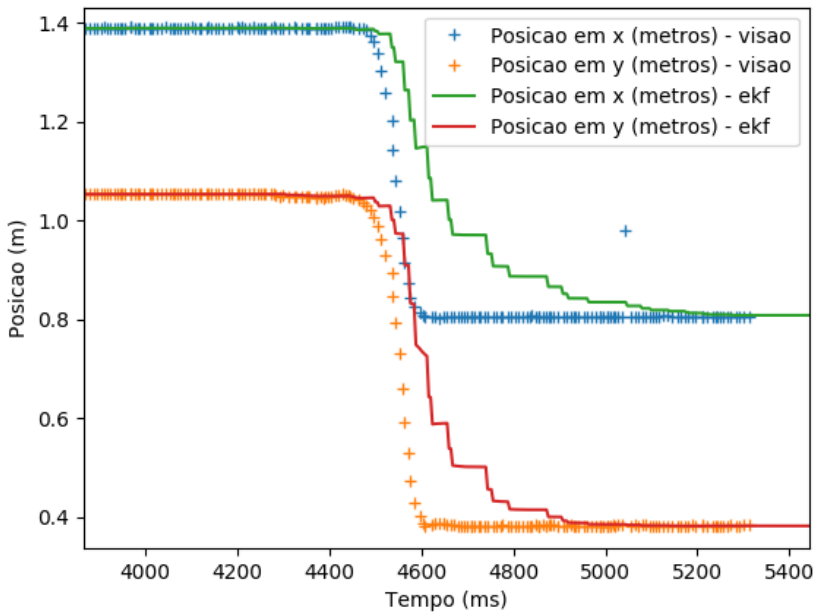
Figura 36 – Convergencia aos valores medidos pela visão computacional



No mesmo gráfico podem ser vistas algumas medidas incorretas de orientação provenientes da visão computacional, que não influenciaram muito na estimativa.

O robô foi movimentado manualmente sem que as rodas girem, de forma que o encoder não consiga detectar o movimento do robô. Após algumas medidas, a estimativa convergiu para o valor correto através das correções fornecidas pelos dados de visão:

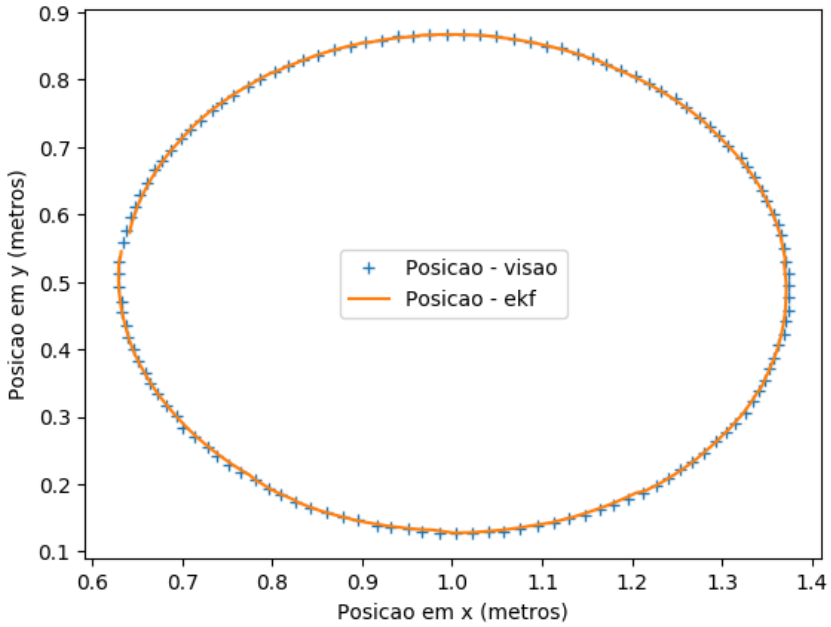
Figura 37 – Estimativa de posicão enquanto as rodas derrapam (tempo em dezenas de ms)



O que indica que derrapamentos não causarão erros permanentes nas estimativas.

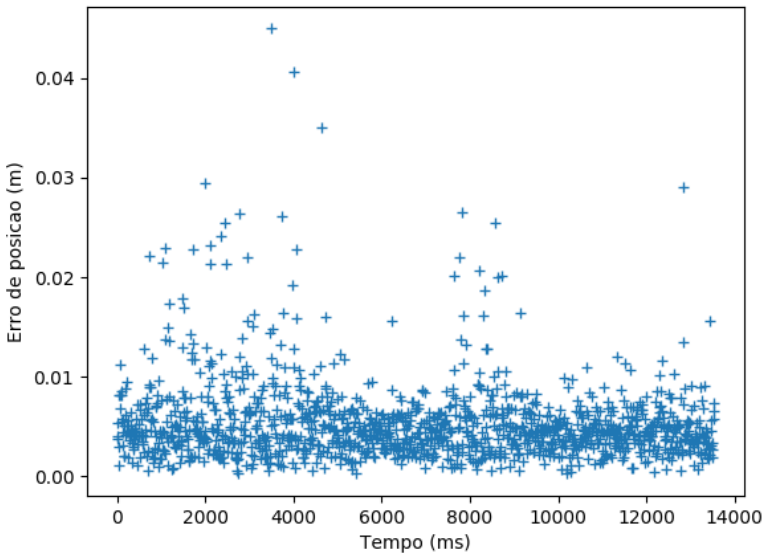
Ao fazer um movimento circular, as posições estimadas e da visão são:

Figura 38 – Medidas e estimativa de posição, movimento circular



O erro é pequeno. A distancia entre a posição estimada e a obtida pela visão é:

Figura 39 – Distancia entre estimativas e visão computacional



Com média de 0.532 cm.

Ao testar o sistema de controle já presente no robô, utilizando as estimativas do EKF em vez da odometria utilizada anteriormente, foi notado um aumento significativo da estabilidade do sistema. Sem o EKF, não era possível controlar o robô mandando novos comandos a velocidade máxima (33 ms), exigindo delays de 200 ms a cada comando. Isto indica que a obtenção de estimativas mais precisas e com menos erros causados por derrapamentos é uma boa solução para os problemas de controle descritos anteriormente.

3.1 Atendimento do cronograma

No planejamento inicial do primeiro semestre, foram definidas as seguintes atividades:

- Pesquisa bibliográfica e desenvolver modelo do robô
- Escrever algoritmo do Filtro de Kalman e testar no robô
- Estudar propriedades de cada sensor e configurá-los
- Medir ruído dos sensores e adicionar dados ao filtro
- Testar estimativas do algoritmo
- Fazer melhorias, corrigir problemas e testar outros modelos

Foi possível fazer todos a tempo, mas algumas foram mais difíceis e demoradas do que o esperado, enquanto outras foram mais fáceis. O desenvolvimento do modelo inicial do robô foi relativamente rápido, porque o robô já foi estudado pela literatura suficientemente bem. O modelo utilizado neste projeto é muito comun na literatura.

A primeira implementação do EKF e sua execução no robô, feita inicialmente com representação de ponto fixo, foi terminada mais rápido do que esperado, mas foi refeita posteriormente para alcançar precisão melhor, o que exigiu mais tempo.

Já a configuração dos sensores foi a parte que mais levou tempo. Não houve dificuldades para implementar a leitura dos sensores, mas a presença de diversas distorções nos sensores exigiu tempo adicional para corrigi-las. Cada sensor teve que ser testado e configurado multiplas vezes, encontrando problemas e os corrigindo.

As ultimas atividades de medir ruídos, testar as estimativas e fazer melhorias foram realizadas de forma iterativa. Algumas medidas de estimativas demonstraram bugs na implementação EKF, outras mostraram que algumas correções no modelo eram necessárias, e outras que era as calibrações de alguns sensores que foram melhoradas.

Já no segundo semestre, foi definida as atividades:

- Estudar acelerômetro e implementar leitura dos dados
- Modificar modelo do robô para adicionar dados de aceleração
- Corrigir distorções do acelerômetro e ajustar covariâncias
- Corrigir distorções do magnetômetro e testes
- Comparação com o UKF

Todas as atividades foram concluídas. A implementação da leitura dos dados do acelerômetro foi feita rapidamente, devido a sua similaridade com os outros sensores. Adicionar o acelerômetro não exigiu muitas mudanças iniciais ao modelo do robô, já que já eram utilizados dados de aceleração através da velocidade das rodas.

Já a correção de distorçoes do acelerômetro foi a parte mais problemática. A variação de inclinação do robô no eixo y dificultou a remoção dos efeitos da gravidade no acelerômetro, exigindo que a inclinação do robô também fosse estimada pelo EKF. Já os efeitos da aceleração centrípeta foram mais simples de remover devido a presença do giroscópio. No final, as distorções encontradas foram removidas e o acelerômetro ficou adequado ao uso no sistema.

A correção das distorções do magnetômetro exigiu apenas pequenas mudanças no modelo de medidas dos sensores e ajustes em constantes das matrizes de covariancia e de ruído de processo. As correções implementadas resolveram os problemas encontrados no primeiro semestre.

Já a implementação do UKF resultou em muitos problemas. O UKF exige mais memória e processamento do microcontrolador, o que precisou de mais cuidado em sua implementação. Além disso, o calculo de raíz quadrada da matriz de covariancia causava erros de execução se as constantes não fossem bem ajustadas, porque constantes mal ajustadas faziam a matriz de coviariancia deixar de ser positiva semi-definida. Apesar dos problemas, o UKF foi implementado e a comparação foi realizada.

3.2 Conclusão

O projeto realizado alcançou seus objetivos e conseguiu uma boa estimativa das variáveis de estado. Os valores estimados apresentam boa rejeição de ruído e rápida recuperação de leituras incorretas, resultando em estimativas mais robustas do que soluções com sensores trabalhando de forma individual.

Os sensores necessitaram de diversos ajustes, como a de remoção de interferências no magnetômetro, remoçao de componentes indesejadas nas medidas do acelerômetro

como a aceleração da gravidade e centrípeta, ajustes em distorções no giroscópio. Esses ajustes foram corrigidos, permitindo um bom uso dos sensores.

A comparação com o UKF obteve resultados mistos: a estimativa de algumas variáveis melhoraram, enquanto outras pioraram. Este resultado podem ser causado pelo desempenho inferior do microcontrolador ao calcular o UKF, e devem ser melhorados caso um microcontrolador de maior capacidade de processamento seja utilizado.

Referências

1 EIGEN. *Eigen*. Disponível em: <http://eigen.tuxfamily.org/index.php?title=Main_Page>. Acesso em: 1 jul. 2018. Citado na página 14.

2 ARM. *ARM Mbed OS is a platform operating system designed for the internet of things*. Disponível em: <https://github.com/ARMmbed/mbed-os>. Acesso em: 1 jul. 2018. Citado na página 14.

3 ARM. *Library to easily communicate with XBee modules*. Disponível em: <https://os.mbed.com/teams/Digi-International-Inc/code/XBeeLib/>. Acesso em: 1 jul. 2018. Citado na página 14.

4 HENRIQUE, T. *VSSS-Robot*. [S.l.]: GitHub, 2018. <https://github.com/PEQUI-MEC/VSSS-Robot>. Citado na página 14.

5 SIEGWART, R.; NOURBAKHSH, I. *Introduction to Autonomous Mobile Robots*. Bradford Book, 2004. (A Bradford book). ISBN 9780262195027. Disponível em: <https://books.google.com.br/books?id=gUbQ9_weg88C>. Citado na página 14.

6 POLOLU. *MinIMU-9 v5 Gyro, Accelerometer, and Compass*. Disponível em: <https://www.pololu.com/product/2738>. Acesso em: 1 jul. 2018. Citado na página 18.

7 MATPLOTLIB. *Python plotting*. Disponível em: <https://matplotlib.org/>. Acesso em: 1 jul. 2018. Citado na página 33.

8 SCIPY. *Scipy*. Disponível em: <https://www.scipy.org/>. Acesso em: 1 jul. 2018. Citado na página 33.

9 WAN, E. A.; MERWE, R. V. D. The unscented kalman filter for nonlinear estimation. In: IEEE. *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. [S.l.], 2000. p. 153–158. Citado na página 41.