Universidade de São Paulo–USP
Escola de Engenharia de São Carlos
Departamento de Engenharia Mecânica
Programa de Pós-Graduação em Engenharia Mecânica

Luís Guilherme Duenhas Silva

Auxílio à navegação autônoma por meio de uma plataforma Android

São Carlos
2018

Luís Guilherme Duenhas Silva

Auxílio à navegação autônoma por meio de uma plataforma Android

Dissertação de mestrado apresentada ao Programa de Engenharia Mecânica da Escola de Engenharia de São Carlos como parte dos requisitos para a obtenção do título de Mestre em Ciências.

Área de concentração: Dinâmica de maquinas e sistemas

Orientador: Prof Mario Luiz Tronco

São Carlos
2018

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Duenhas Silva, Luís Guilherme
D586a Auxílio a navegação autônoma por meio de uma plataforma Android / Luís Guilherme Duenhas Silva; orientador Mario Luiz Tronco. São Carlos, 2018.

Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia Mecânica e Área de Concentração em Dinâmica das Máquinas e Sistemas -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2018.

1. Robótica móvel. 2. Navegação Autônoma. 3. Android. 4. Sensores Inerciais. 5. Visão Computacional. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE JULGAMENTO

Candidato: Engenheiro **LUIS GUILHERME DUENHAS SILVA.**

Título da dissertação: "Auxílio à navegação autônoma por meio de uma plataforma Android".

Data da defesa: 20/04/2018.

Comissão Julgadora:	**Resultado:**
Prof. Associado **Mario Luiz Tronco** **(Orientador)** (Escola de Engenharia de São Carlos/EESC)	Aprovado
Prof. Dr. **Gustavo Franco Barbosa** (Universidade Federal de São Carlos/UFSCar)	Aprovado
Prof. Dr. **Emerson Carlos Pedrino** (Universidade Federal de São Carlos/UFSCar)	Aprovado

Coordenador do Programa de Pós-Graduação em Engenharia Mecânica:
Prof. Associado **Gherhardt Ribatski**

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Luís Fernando Costa Alberto**

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

Agradecimentos

Agradeço a minha esposa Giovanna pelo amor, carinho e compreensão em todos os momentos, sendo eles fáceis ou difíceis, de alegria ou superação, sempre ao meu lado.

Agradeço a minha mãe Heloísa, meu pai José Antônio e a minha irmã Ana Luíse pelo apoio, educação e, principalmente, por sempre acreditarem em mim.

Agradeço aos professores e funcionários da USP São Carlos que me acompanharam durante todo o processo de aprendizagem, desde a graduação.

Agradeço a USP como instituição que me proporcionou aprendizado para toda a vida.

E agradeço a Deus por poder viver na sua companhia.

"A parte mais importante do progresso,
é o desejo de progredir."
(Lucius Annaeus Seneca)

Resumo

Silva, Luis G. D. **Auxílio à navegação autônoma por meio de uma plataforma Android**. 97 p. Dissertação de mestrado – Escola de Engenharia de São Carlos, Universidade de São Paulo, 2018.

Sistemas embarcados têm se tornado cada vez mais complexos e habituais, não sendo mais meros cumpridores de funções dedicadas e sim agregando funcionalidades cada vez mais variadas. Dentre esses, os que utilizam o sistema operacional Android têm se destacado, devido sua versatilidade, poder de processamento, pluralidade de sensores e principalmente baixo custo. Sendo usualmente utilizados em plataformas multifuncionais e que contém diversos sensores embarcados, como acelerômetro, bússola magnética, giroscópio e câmera. Este trabalho teve como objetivo apresentar um estudo sobre a utilização de uma plataforma Android para o auxílio à navegação autônoma de sistemas robóticos, por meio do emprego dos sensores embarcados supracitados, da comunicação direta do sistema robótico com a plataforma Android de forma a melhorar a estimativa de orientação e localização e do processamento inerente à plataforma, de modo a realizar a localização do sistema robótico em um ambiente previamente conhecido. Para tanto, além de avaliar a efetividade dos sensores, também foram investigados diferentes métodos de fusão sensorial e diferentes métodos de visão computacional de forma a comparar suas entregas e suas demandas. Os resultados encontrados corroboram a ideia de que a plataforma Android é uma oportunidade de baixo custo para auxílio à navegação autônoma, já que por meio de sensores inerciais e óptico, normalmente integrados à placa com esse sistema operacional, é possível navegar autonomamente um sistema robótico de um ponto a outro de um ambiente previamente conhecido.

Palavras-chave: Robótica Móvel. Navegação Autônoma. Android. Sensores Inerciais. Visão Computacional.

Abstract

Silva, Luis G. D. **Autonomous navigation assistance through an Android platform**. 97 p. Master Thesis – São Carlos School of Engineering, University of São Paulo, 2018.

Embedded systems have become more and more complex and habitual, being no longer mere fulfillers of dedicated tasks, but also earning more and more varied functionalities. Among these, the devices with Android operating system have stood out, being usually used in multifunctional platforms and that contains several sensors, like accelerometers, magnetic compass, gyroscope and camera. This work presents a study on the use of an Android platform to aid in the autonomous navigation of robotic systems, through the use of abovementioned embedded sensors and the processing inherent in the platform in order to perform the location of the robotic system in a previously known environment. Therefore, besides evaluating the effectiveness of the sensors, different methods of sensorial fusion were investigated and also different methods of computer vision in order to compare their deliveries and their demands. The findings corroborate the idea that the Android platform is a low-cost opportunity for autonomous navigation, being possible to autonomous navigate a robotic system from one point to another from a previously known environment.

Keywords: Mobile Robotics. Autonomous Navigation. Android. Inertial Sensors. Computer Vision.

Lista de ilustrações

Sumário

Capítulo 1

Introdução

1.1 Contextualização

Sistemas embarcados possuem vasta aplicação e a cada dia têm se tornado cada vez mais complexos e habituais. Em meio ao surgimento dos intitulados "*smartphones*", em 2007, a Google Inc., empresa multinacional de serviços *online* e *software*, surpreendeu o mundo com o lançamento do Android, uma plataforma abrangente, aberta e livre para dispositivos móveis que vai muito além de um simples sistema operacional.

Desde seu lançamento a plataforma se modificou muito, principalmente impulsionada pela política de incentivo ao livre desenvolvimento de aplicações e pelo próprio desenvolvimento dos dispositivos móveis, o que permitiu agregar funcionalidades cada vez mais variadas. Atualmente são inúmeras as aplicações da plataforma, notavelmente inseridas em distintos segmentos do mercado. Alguns exemplos são mostrados na Figura 1, a seguir. Estes sistemas incluem eletrônicos como *smartphones*, *tablets* e computadores, eletrodomésticos, como como televisores e geladeiras, sistemas domésticos, como controles de temperatura e vigilância, uso pessoal, como óculos e relógios inteligentes, sistemas automotivos, como sistemas de navegação, controles de bordo e multimídia, sistemas de placas de desenvolvimento aplicado, como os projetos Udoo e até mesmo aplicações em robótica, como robôs de entretenimento e robôs de desenvolvimento.

Em paralelo a essa realidade, o contínuo desenvolvimento da robótica móvel vem ganhando um amplo destaque e espaço na sociedade de um modo geral. A visão de robôs como sendo robôs industriais e braços mecânicos, vem aos poucos compartilhando espaço com os robôs móveis, os quais são responsáveis por se deslocar no ambiente em que se encontram, realizando trabalhos que exigem precisão, destreza e que são repetitivos ou mesmo insalubres.

Algumas das áreas com serviços sendo realizados por robôs móveis são mostradas na Figura 2, e incluem: exploração de ambientes impróprios para humanos, com sistemas de prospecção e aeroespaciais, militares, com sistemas de monitoramento aéreo remoto e de transporte de carga, agrícolas e industriais, com veículos de operação autônomos,

automobilística, com veículos de transporte autônomos, domésticos, com aspiradores de pó e cortadores de grama e entretenimento, com veículos aéreos não tripulados e robôs humanóides.

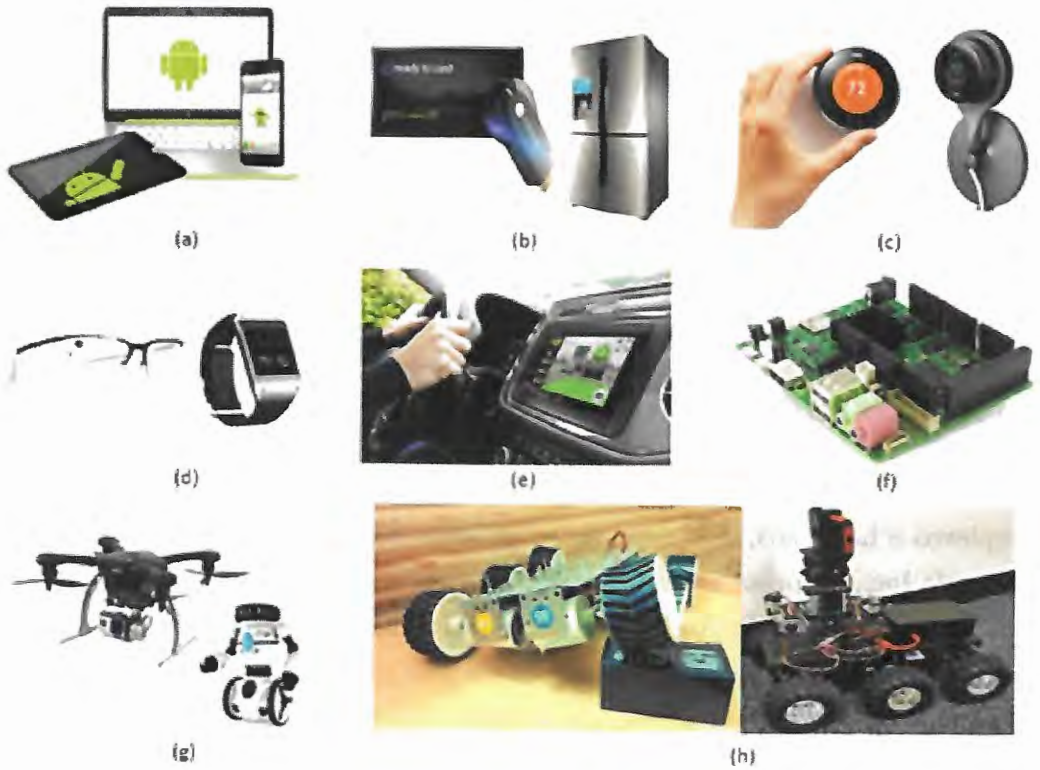

Figura 1 – Exemplos de aplicações Android. (a) Eletrônicos (b) Eletrodomésticos (c) sistemas domésticos (d) uso pessoal (e) sistemas automotivos (f) placas de desenvolvimento aplicado (g) robôs de entretenimento (h) robôs de desenvolvimento.

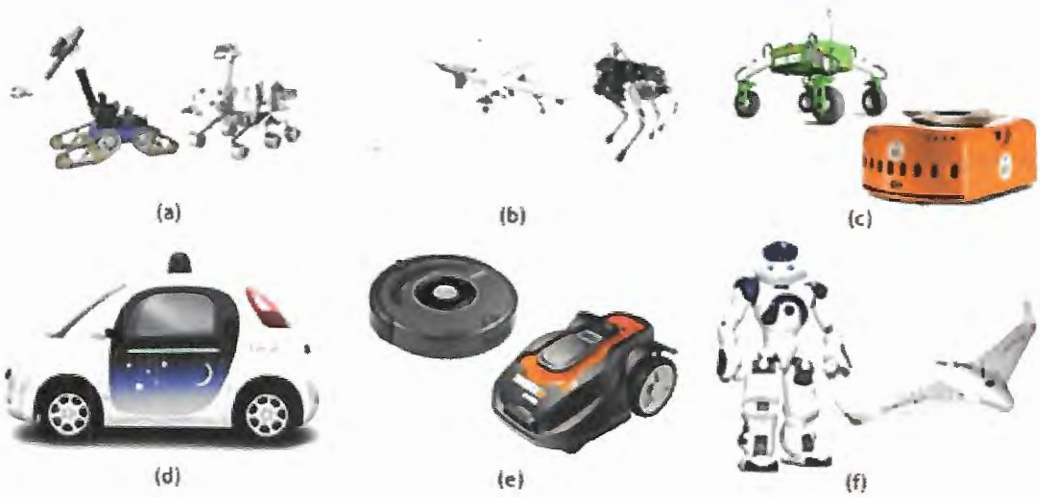

Figura 2 – Áreas de Aplicação de Robôs Móves: (a) Exploração, (b) Militar, (c) Agrícola e Industrial, (d) Automobilística, (e) Doméstica e (f) Entretenimento.

1.2 Motivação

A robótica é um catalisador da relação da ciência com a economia, criando valor por meio da dinamização da atividade econômica. Entretanto, apesar da robótica já ser uma realidade crescente no âmbito do setor secundário da economia, a robótica móvel é um tema pouco explorado na área.

Ambientes de manufatura, ou mesmo de produção agrícola, são altamente dinâmicos e necessitam ser extremamente eficientes para que a empresa possa ser competitiva. Nesse sentido, o desenvolvimento de robôs móveis associados à essência da produção podem ser elementos chave para aumentar a produtividade e reduzir o desperdício de recursos e tempo. Como visto na seção anterior, o grande avanço nas pesquisas de robôs móveis e na plataforma Android vem despertando a criação de novas possibilidades e aplicações cada vez menos prováveis para a concepção humana.

Assim sendo, as possibilidades de criar novos produtos ou serviços customizados e de baixo custo são inúmeras, o que fortalece a motivação de trabalhos em ambos os campos de pesquisa e consequentemente na intersecção dos mesmos. Dessa forma, o presente trabalho pode contribuir para o desenvolvimento de um sistema de auxílio à navegação autônoma preciso, confiável, flexível para os requisitos específicos de cada aplicação e mais acessível financeiramente.

1.3 Problemas

Sistemas autônomos são sistemas dotados de propriedades que permitem a execução de determinadas tarefas sem a necessidade de interferência externa durante o processo. Assim, o problema básico da navegação autônoma de veículos é a determinação de uma trajetória, de modo que o veículo possa navegar em segurança (sem colidir com obstáculos) até alcançar uma ou mais posições pré-estabelecidas no ambiente (FABRO, 1996).

Desta forma, sistemas de navegação autônomos devem ser capazes de definir uma sequência de ações a serem tomadas por robôs móveis, dotados de um conjunto limitado de sensores (FIGUEIREDO, 1999). E intimamente relacionados à autonomia dos sistemas robóticos encontram-se três paradigmas primitivos: sentir, planejar e atuar, os quais estão relacionados a forma como os dados são coletados pelos sensores e então processados (THRUN et al., 2005).

Leonard e Durrant-Whyte (DURRANT, 1992) sumarizaram uma maneira generalista de definir o problema da navegação autônoma de robôs móveis por meio de três perguntas básicas, as quais devem ser respondidas pelos seus algoritmos de controle:

- **"Onde estou?"** , pergunta que relaciona o sistema robótico com a estratégia de localização ou, como supracitado, com o ato de "sentir".

- **"Para onde vou?"** , pergunta que relaciona o sistema robótico com o seu objetivo ou tarefa.

- **"Como posso chegar lá?"** , pergunta que relaciona o sistema robótico com a estratégia de navegação, ato de "planejar".

A estratégia de localização resulta na posição que o sistema robótico possui em relação ao ambiente e que pode ser calculada baseada em informações do próprio ambiente coletadas através de sensores (MACHARET, 2009).

O processo de localização é de extrema importância, uma vez que fornece o embasamento necessário para a navegação em si. Quanto maior a precisão e certeza da localização do sistema robótico, mais assertiva será a navegação, razão para a constante busca por sensores, filtros e fusões de informações mais acuradas e robustas.

A estratégia de navegação é então responsável por integrar a localização atual do robô, com a localização de destino, podendo ou não ser auxiliada por um mapa do ambiente previamente definido. Assim, o sistema robótico verifica os possíveis caminhos que levam até a posição desejada sempre atento aos obstáculos, os quais devem ser evitados (THRUN, 2002).

Por fim, o ato de "atuar" é realizado baseando-se nas decisões tomadas pela estratégia de navegação e reflete na alteração da posição do robô, assim exigindo uma reavaliação da estratégia de localização, que por sua vez realimenta a estratégia de navegação, assim completando o ciclo.

1.4 Proposição

Com base nas seções anteriores, neste trabalho é apresentado um estudo sobre métodos de coleta e tratamento das informações dos sensores de uma plataforma Android visando realizar a localização em tempo real de um sistema robótico em um ambiente previamente mapeado.

Essas informações serão comunicadas ao sistema robótico, através de módulos comuns à plataforma Android, como *Wi-Fi*, *Bluetooth* e comunicação cabeada *Universal Serial Bus*, USB, de modo a permitir que o mesmo navegue autonomamente, realizando as tarefas para as quais foi programado. Portanto, estratégias de navegação não serão abordadas neste trabalho.

CAPÍTULO 2

Estado da arte

Nesta seção é apresentada uma revisão bibliográfica das principais pesquisas relacionadas a este trabalho: robótica móvel, aplicações com sistema operacional Android, visão computacional e fusões sensoriais. Nos últimos anos houve um crescimento vertiginoso na intersecção dessas áreas de pesquisa.

Na robótica móvel, podem ser observados esforços no sentido de se obter resultados mais precisos, mais robustos e, principalmente, com menor custo de implementação, como apresentado por Kothari (KOTHARI, 2011) e Seljanko (SELJANKO, 2013).

Já no ramo da visão computacional, foi demonstrado por Leandro Couto sistemas de localização robótica de veículos autônomos baseado em pontos de referência (COUTO, 2012). Através de um método detector-descritor utilizando-se espaço de escala em um mapa visual memorizado previamente.

Nesse mesmo ramo, muitos são os trabalhos voltados para o desenvolvimento de realidade aumentada, no qual o trabalho mais pertinente foi elaborado por Kato et al (KATO, 2000) e que diz respeito à criação da biblioteca ARToolKit, e amplamente aplicado por Bleser na navegação autônoma de robôs móveis (BLESER, 2009b).

No segmento da fusão sensorial o trabalho mais completo e promissor é de Alexander Pacha, que propôs a fusão sensorial de sensores virtuais (PACHA, 2013). Isto é, combinação de respostas previamente processadas e oriundas de diversos sensores físicos.

A tendência nas pesquisas sobre o sistema operacional Android é a exploração de novas aplicações, assim tomando as mais diversas vertentes. Nesse sentido, há trabalhos mostrando a utilização de dispositivos Android para o auxílio a navegação (CARRERA, 2017), bem como desenvolvimento de múltiplos robôs interagindo entre si por meio do sistema operacional Android (GUNAWAN, 2017). Não menos importante, há ainda o desenvolvimento de robôs para uso pessoal também utilizando a plataforma Android (ONO, 2014).

2.1 Sistema operacional Android

Desenvolvido pela Open Handset Alliance no final de 2008, um consórcio de mais de 30 empresas do setor de tecnologia e comunicação, e gerido pela Google Inc., o Android é uma plataforma móvel livre, aberta e completa, apresentando características incorporadas de outros sistemas, além de inúmeros conceitos inovadores para o segmento móvel (WEISS, 2008).

Baseado no sistema operacional Linux e com uma interface de usuário baseada na manipulação direta, consiste em um pacote de *software* que inclui um sistema operacional, um *middleware* e aplicativos de características essenciais e específicas. dentre os quais se encontram o emulador e um *debbuger*, juntamente com toda a biblioteca de desenvolvimento e ferramentas para a criação de aplicativos.

O Android tem como objetivos principais a oportunidade de personalização das aplicações e componentes presentes em seu sistema e, a possibilidade de desenvolvimento rápido e moderno de aplicações pessoais, corporativas e de pesquisa, permitindo assim a desenvolvedores e empresas criarem aplicativos que se apliquem melhor ao cotidiano dos usuários, ampliando suas habilidades, e, dependendo da configuração de *hardware* do dispositivo, permitindo a realização de atividades específicas.

Além de administrar e desenvolver, a Google disponibiliza um site voltado unicamente para os desenvolvedores, https://developer.android.com, o qual contém toda a API do sistema, e outro site, https://source.android.com, o qual contém todo o código fonte da plataforma. Também é fornecido um ambiente de desenvolvimento integrado, chamado de Android Studio, https://developer.android.com/studio/, baseado na IDE Eclipse, mas com foco no desenvolvimento específco de aplicações Android. Dessa forma, permitindo que programadores escrevam livremente, adaptando, personalizando e assim contribuindo para o desenvolvimento comum.

Com essa série de atributos o Android se tornou o sistema operacional móvel mais utilizado do mundo, estando presente em mais dispositivos móveis do que os competidores somados. Sendo extremamente popular entre empresas de tecnologia que busca uma aplicação de baixo custo, personalizável e para dispositivos de alta tecnologia. Então, devido todas as características supracitadas, esse trabalho será focado especificamente no Sistema Operacional Android e não em outros como o iOS ou Windows.

2.1.1 Arquitetura

A estrutura do sistema operacional Android é basicamente um pilha de *software* dividida em 5 camadas: *Kernel*, *Hardware Abstraction Layer*, Biliotecas nativas e *Android Runtime*, *Android Framework* e Aplicativos. Cada qual com módulos internos especializados em sua tarefa particular, conforme ilustrado na Figura 3.

Figura 3 – Estrutura da plataforma Android.

Kernel

Os dispositivos móveis, assim como os computadores, possuem arquiteturas distintas entre si. Logo, para que um sistema consiga operar com diferentes tipos de *hardware*, se fazem necessários os sistemas operacionais, os quais são os responsáveis por gerenciar o processador, a memória e outros dispositivos de entrada e saída, além de fornecer uma interface amigável para o utilizador.

Assim, no nível mais baixo da arquitetura do Android encontra-se o *kernel* do Linux versão 3.4, removido de alguns componentes típicos da plataforma Linux, como a biblioteca GNU e acrescido de pequenas melhorias vitais para as funcionalidades de uma plataforma embarcada, como a *Binder Intern-Process Communication*, que permite que os APIs de alto nível interajam diretamente com os serviços do sistema Android. Essas modificações geram muita discussão se o Android é uma distribuição Linux ou não, mas isso não impacta o desenvolvimento de aplicações, já que ambas as plataformas são livres e gratuitas.

Os objetivos deste nível são o controle dos serviços relacionados à segurança, gerência de memória e processos, pilhas de rede e manutenção dos modelos de drivers. De um modo geral, esse nível garante que o desenvolvedor não terá de se preocupar em como acessar ou gerenciar componentes específicos do dispositivo, criando assim uma abstração entre o *hardware* e o *software*.

Camada de Abstração de Hadware

A camada de abstração de *hardware*, também conhecida como HAL, *Hardware Abstraction Layer*, define uma interface padrão para fornecedores de *hardware*. Assim, permitindo a implementação de funcionalidades sem afetar ou necessitar modificações no sistema.

Para tanto, os mais diversos *hardwares*, como sensores, devem ter seus respectivos drivers compilados em módulos HAL, sob a extensão ".so", os quais serão carregados pelo sistema Android no momento apropriado, permitindo que as demais camadas da estrutura não necessitem acessar diretamente o *Kernel*, o que aumenta o desempenho e a modularidade do sistema.

Bibliotecas Nativas

Durante o desenvolvimento de programas, existem rotinas básicas que muitas vezes são comuns em muitos *softwares*, que não são o foco principal do desenvolvedor. Logo, para evitar-se a reescrita dessas funções e rotinas, existem as bibliotecas.

Escritas na linguagem de programação C++, as bibliotecas podem ser acessadas apenas através da camada de *framework* dos aplicativos, e contemplam módulos que servem como base para as aplicações.

Android Runtime

Paralelamente às bibliotecas, encontra-se a máquina virtual Android, *Android Runtime*, desenvolvida para atender às necessidades de um sistema que rode com limitações de bateria, memória e processamento. Nesse contexto, se fazem presentes tanto a Máquina Virtual Dalvik (MVD), proveniente da Máquina Virtual Java (JVM), baseada em registradores e desenvolvida pela Google, quanto as principais bibliotecas Java para entrada e saída de dados, coleções e outros.

O *Android Runtime* foi completamente escrito em Java e constitui a base para os aplicativos, permitindo aos desenvolvedores o acesso aos mesmos recursos utilizados pelas aplicações do sistema, podendo fazer uso destes seguindo seu livre arbítrio, mediante algumas restrições de segurança e de recursos dispostos por outras aplicações

Android Framework

Além das bibliotecas, os desenvolvedores têm a sua disposição diversos outros componentes que o Android disponibiliza como, provedor de conteúdo, gerenciador de janela, recursos e atividades, permitindo a criação de listas, grades, caixas de texto, botões e outros tipos de controle.

A possibilidade de acesso e modificação dos componentes do Android é um diferencial em relação às outras plataformas para dispositivos móveis. Tais componentes permitem que as aplicações desenvolvidas possam se relacionar com o sistema do celular, adaptando assim o sistema às necessidades do usuário.

Aplicações

Por fim, a camada de aplicativos é o ambiente de contato com o usuário e contempla as aplicações como cliente de e-mail, envio e recebimento de mensagens, calendários, mapas, navegador *Web*, contatos, entre outros.

2.1.2 Fundamentos das Aplicações

Os aplicativos Android são escritos em linguagem Java e após compilados ficam presentes em um arquivo do tipo *.apk. Esse arquivo pode ser instalado através da ferramenta chamada AAPT (*Android Asset Package Tool*), que faz o gerenciamento dos pacotes e instala a aplicação no sistema (DEVELOPER, 2011).

Cada aplicativo Android é executado em seu próprio processo Linux e dentro de sua própria máquina virtual. Um aplicativo recebe por padrão uma ID exclusiva, como os aplicativos Linux, assim, cada processo é invisível ao outro. Também é possível que mais de uma aplicação compartilhe o mesmo ID, para que haja compartilhamento dos mesmos recursos, arquivos e máquina virtual.

Outra diferença entre os aplicativos Android e os de outros sistemas é que cada aplicação pode realizar chamadas para outras a qualquer momento durante sua execução. Podendo a aplicação requerer uma chamada de outro processo tanto em qualquer momento quanto em diversos pontos de partida. Portanto, um aplicativo Android não possui uma função "*main*()".

2.1.3 Desenvolvimento

O desenvolvimento de uma aplicação para Android consiste em utilizar um conjunto de códigos, classes, funções e métodos, desenvolvidos justamente para dar suporte a esse processo. A Google disponibiliza gratuitamente o Kit de Desenvolvimento (SDK) do Android, o qual possui as ferramentas e APIs necessárias.

A construção de aplicativos pode ser realizada com as mesmas ferramentas utilizadas no desenvolvimento de aplicativos Java e, ainda através do framework de desenvolvimento do Android é possível encontrar bibliotecas e funcionalidades específicas da plataforma.

Assim, uma vez analisado todo o conceito do sistema operacional, desenvolveu-se um código capaz de ler os dados dos sensores e performar as respectivas fusões sensoriais, apresentadas no Capítulo 3, além de também ser capaz de executar as bibliotecas de

visão computacional, apresentadas no Capítulo 4. Todo esse desenvolvimento deu origem a um aplicativo para dispositivos com sistema operacional Android, apresentado na Seção 3.4

Visando aprimorar a eficiência do código, o aplicativo foi codificado também na camada de bibliotecas, as quais são escritas em C/C++, além obviamente na camada de aplicações, a qual é escrita em Java.

Inúmeras são as possibilidades de otimização do código, por exemplo remover aplicações desnecessárias ao projeto, como controle de luminosidade de tela, gestão de bateria ou sincronização de mensagens. Porém, como será apresentado nos capítulos seguintes a aplicação obteve sucesso, mesmo sem essas otimizações. Isso se deve ao elevado poder de processamento das plataformas com sistema operacional Android disponíveis hoje em dia.

2.2 Sensores

Dispositivos móveis com o sistema operacional Android geralmente contam com uma ampla variedade de sensores, para os mais diversos propósitos e aplicações. Dentre os mais comuns estão os sensores ópticos, como câmeras e sensores de proximidade e de luminosidade, sensores de posicionamento global e sensores inerciais, como acelerômetro, bússola magnética e giroscópio.

Este capítulo descreve os sensores utilizados no projeto, apresentando suas funcionalidades, características e restrições intrínsecas.

2.2.1 GPS

O *Global Positioning System*, GPS, é um sistema de sensoriamento composto por três segmentos: espacial, controle e usuário. Ele é capaz de retornar a posição global do usuário (latitude, longitude e altitude).

O segmento espacial desse sistema de sensoriamento consiste em 24 satélites orbitando em 6 órbitas sob uma inclinação de 55° em relação a linha do equador. As órbitas são organizadas de tal forma que a todo tempo tem-se em qualquer ponto do planeta a visualização de pelo menos 6 satélites.

O segmento de controle é composto por estações e antenas, os quais são utilizados para verificar, controlar e corrigir, caso necessário, as órbitas e os relógios dos satélites.

O segmento do usuário consiste de um dispositivo contendo uma antena, capaz de receber as mensagens enviadas pelos satélites, além de *hardware* para realizar o processamento das informações, de modo a calcular a exata localização, e uma interface, de modo a exibir os resultados para o usuário.

Os sensores GPS são capazes de calcular a correta localização através da triangulação das informações dos diferentes satélites. As mensagens enviadas pelos satélites contém a

exata posição orbital do satélite e o exato momento em que a mensagem foi enviada. O receptor compara então o tempo necessário para receber a mensagem e realiza os cálculos para definir sua posição global.

A Figura 4 e as Equações 1, 2 e 3 representam de maneira ilustrativa o esquemático e os correspondentes cálculos que são realizados para calcular a exata localização do dispositivo.

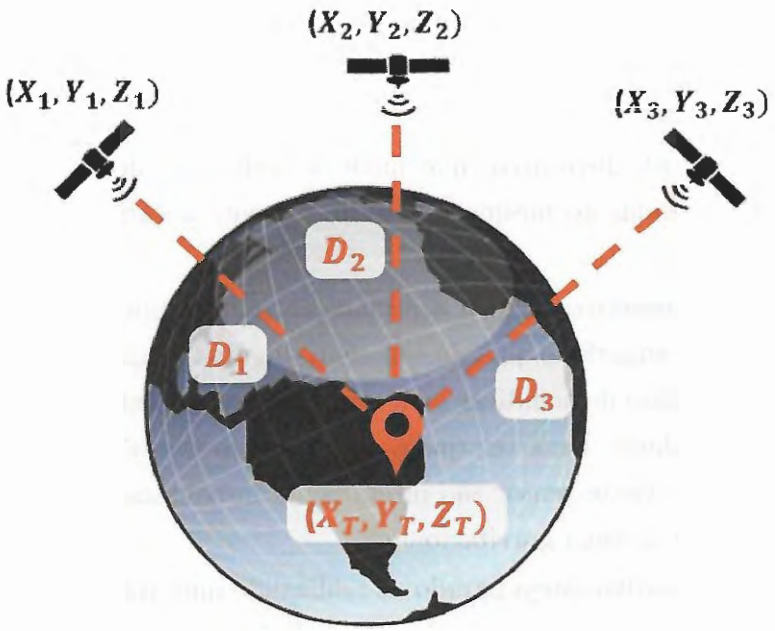

Figura 4 – Esquemático do método de localização realizado pelo GPS.

$$\sqrt{(x_t - x_1)^2 + (y_t - y_1)^2 + (z_t - z_1)^2} + c * \Delta t = D_1 \tag{1}$$

$$\sqrt{(x_t - x_2)^2 + (y_t - y_1)^2 + (z_t - z_2)^2} + c * \Delta t = D_2 \tag{2}$$

$$\sqrt{(x_t - x_3)^2 + (y_t - y_1)^2 + (z_t - z_3)^2} + c * \Delta t = D_3 \tag{3}$$

Nas quais, c é a constante da velocidade da luz e Δ t é o passo, discretização, do relógio receptor. Essa tecnologia é amplamente difundida nas mais diversas aplicações, entretanto requer uma "visão" direta dos satélites. Ambientes internos ou outros objetos que possam bloquear o caminho da mensagem interferem no processo de comunicação, gerando erros nas estimativas de posição.

Outra fonte de erro comum para esse sensoriamento é a inconsistência dos clocks dos receptores. A variação ou baixa discriminação na contagem do tempo pelo receptor

influencia diretamente no cálculo do tempo entre o envio e o recebimento da mensagem, o que atrapalha na estimativa de posição.

Uma vez que os dispositivos em estudo não são focados para o uso desse tipo de sensoriamento, o erro comum para a localização via GPS de dispositivos Android é de um raio de 5 metros. E o intuito do projeto é o desenvolvimento de uma tecnologia que possa ser usada tanto interna quanto externamente, portanto, não será foco desse trabalho o estudo desse sensor.

2.2.2 Acelerômetro

Um acelerômetro é um dispositivo que mede a aceleração do dispositivo por meio das forças externas aplicadas ao mesmo e não diretamente a variação de velocidade do dispositivo no tempo.

Geralmente os acelerômetros medem a aceleração sensoriando o quanto uma massa específica pressiona uma superfície quando sob efeito de uma força externa. Desse modo, um acelerômetro em repouso deve indicar uma aceleração de aproximadamente 9,81 m/s^2, relativo à força da gravidade. Uma vez que o acelerômetro também é capaz de detectar a gravidade, as respostas deste sensor são diretamente impactadas pela posição angular do dispositivo em relação a força gravitacional.

E a menos que o dispositivo esteja parado ou realizando uma trajetória linear com uma aceleração muito constante, é praticamente impossível separar somente com os dados do acelerômetro qual é o componente da força gravitacional e os demais componentes, oriundos das forças de aceleração gerados pelo movimento do dispositivo.

2.2.3 Bússola Magnética

Um sensor magnético é um instrumento capaz de sensoriar campos magnéticos que cercam o instrumento, inclusive o campo magnético terrestre, daí o nome bússola magnética.

Esses sensores, também conhecidos como magnetômetros, podem ser de dois tipos: escalares, os quais medem somente a magnitude da somatória dos campos magnéticos, ou vetoriais, os quais além da magnitude também tem a capacidade de medir a direção do campo gravitacional, em relação a orientação espacial do dispositivo.

Apesar de ser capaz de medir a direção do campo magnético, o sensor usa como referência o próprio dispositivo, assim tornando-se altamente dependente da orientação angular do mesmo frente ao espaço tridimensional.

E a mais forte fonte de ruído e variação na resposta desse tipo de sensor é a contaminação magnética do sinal causada por componentes ferromagnéticos nas proximidades do sensor.

2.2.4 Giroscópio

Giroscópio é um dispositivo, que baseado nos princípios da conservação do momento angular, é capaz de manter a orientação angular mesmo frente a movimentação do dispositivo e assim medir a orientação ou as velocidades angulares aplicadas ao dispositivo.

Giroscópio Mecânico

Um giroscópio convencional, mecânico, consiste em uma roda montada em um suporte bi-apoiado de forma a ser possível rotacionar-se em todos os três eixos, como apresentado na Figura 5

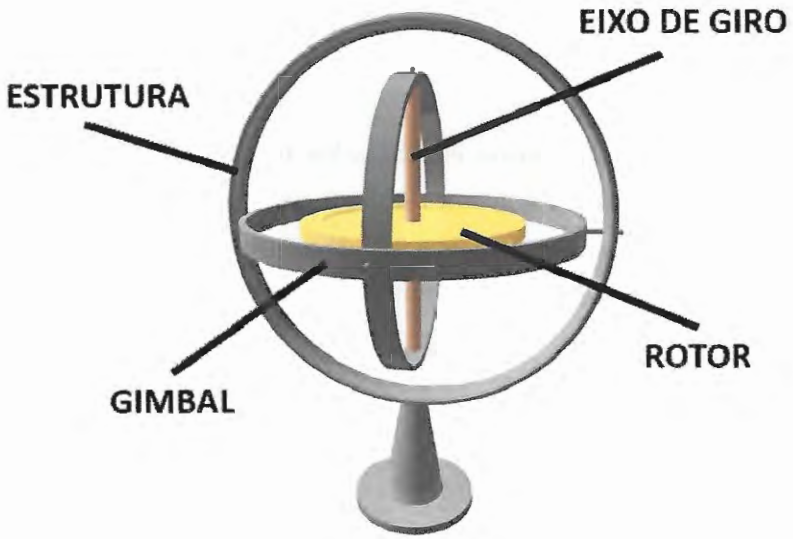

Figura 5 – Giroscópio mecânico convencional.

Como efeito da conservação do momento angular, a roda resiste às alterações de orientação. Assim, quando um giroscópio mecânico é submetido a uma rotação, a roda irá permanecer em uma orientação global constante, mas o ângulo entre os apoios adjacentes irá mudar. Esse ângulo, por sua vez, pode ser medido, de forma a definir a movimentação que o dispositivo foi submetida.

A principal desvantagem desse tipo de giroscópio é que eles contêm partes móveis, e partes móveis causam atrito, que por sua vez faz com que a resposta seja desviada ao longo do tempo (WOODMAN, 2007).

Giroscópio Micro-eletro-mecânico

Um giroscópio micro-eletro-mecânico é constituído por um sistema integrado composto por poucas peças construídas por microusinagem que fazem uso do efeito Coriolis para medir a velocidade angular do sistema.

O efeito Coriolis afirma que em um sistema, com uma massa m, em rotação, com uma velocidade angular e uma velocidade v, experimenta a força f, segundo a Equação 4

$$f = -2 * m * (\omega \times v) \tag{4}$$

A principal desvantagem desse tipo de giroscópio é que para determinar a orientação angular relativa do sistema é necessário integrar as respostas ao longo do tempo, o que pode gerar desvios caso o dispositivo esteja parado e a resposta não seja exatamente nula.

Porém também é válido lembrar que, a menos que a informação de orientação seja calibrada na inicialização, esse sensor somente poderá fornecer informações de orientação relativa.

2.2.5 Sensores Virtuais

Além dos diversos sensores físicos encontrados nos dispositivos Android, as placas com sistema operacional Android contam também com uma gama de "sensores virtuais". Esses são resultados de derivações, filtros e fusões dos sensores físicos implementados via *software*, que por sua vez são capazes de entregar diferentes respostas e resultados mais estáveis e precisos.

Norte Magnético

O eixo de rotação do planeta Terra não coincide exatamente com o eixo magnético do mesmo. Isso faz com que a bússola magnética aponte para uma direção que não passa necessariamente pelo norte geográfico. Essa diferença é conhecida como declinação magnética e varia conforme a posição do sensor no planeta.

Logo, com base no Mapa Mundi Magnético fornecido pela Agência de Inteligência Geoespacial dos Estados Unidos da América, esse sensor virtual é capaz de corrigir as informações da bússola magnética, segundo a informação de localização fornecida pelo GPS do dispositivo, fornecendo a correta direção do polo norte magnético.

Gravidade

Esse sensor virtual é capaz de utilizar o acelerômetro e o giroscópio em conjunto de forma a sempre definir qual a correta orientação da força gravitacional.

Quando o dispositivo está parado, com respostas nulas no giroscópio, esse sensor assume que toda resposta do acelerômetro pertence a força da gravidade. E quando o dispositivo entra em movimento, o sensor utiliza a resposta do giroscópio para calcular a mudança na orientação e em seguida a nova orientação da gravidade.

Aceleração Linear

Uma vez calculada a correta orientação e magnitude da força gravitacional é possível remover essa variável da resposta do acelerômetro, obtendo portanto somente a real aceleração do dispositivo.

Esse sensor é responsável por performar essas contas e retornar a aceleração em m/s^2 nos três eixos coordenados, x, y e z.

Porém, esse sensor requer duas integrações o que demanda demasiado poder de processamento e pode resultar em um erro elevado, principalmente se a estimativa da orientação da força gravitacional não for precisa (BLESER, 2009a).

Orientação Abdoluta

Também conhecido como Rotation Vector, esse é o sensor mais robusto disponível no Android para a definição da orientação absoluta do dispositivo. Sua eficiência e robustez vem do fato que utiliza como entrada base os sensores: acelerômetro, bússola magnética e giroscópio, além de ter em seu código um filtro de Kalman.

Nessa fusão sensorial, o giroscópio é o principal contribuidor para a orientação do dispositivo, enquanto esse está em movimento. Por sua vez o acelerômetro fornece a correção da orientação em função da direção da gravidade e a bússola fornece a correção em termos da direção norte geográfica (PACHA, 2013).

Giroscópio Calibrado

Esse é o sensor mais preciso disponível no Android, porém, uma vez que só utiliza o giroscópio, não é capaz de definir a orientação absoluta do dispositivo, somente a orientação relativa do mesmo frente a orientação inicial.

Para entregar tal precisão o Android possui em seu código um filtro de Kalman que é capaz de definir com precisão a correta resposta do giroscópio, evitando ruídos estacionários e desvios intrínsecos a integração temporal necessária para obter-se a resposta da orientação.

Capítulo 3

Materiais e Métodos

O objetivo deste capítulo é apresentar e dar uma visão mais detalhada dos componentes de *software* e *hardware* utilizados nesse trabalho, bem como os métodos e técnicas investigadas e implementadas. Começando pelos fundamentos básicos de navegação inercial e visão computacional, passando pela plataforma robótica utilizada e encerrando no aplicativo desenvolvido.

A Figura 6 representa a lógica do sistema proposto nesse trabalho. Dado um robô móvel genérico, capaz de se mover autônomamente em um ambiente dotado de marcadores visuais e previamente mapeado, e um dispositivo Android posicionado fisicamente em sua carcaça, é possível estabelecer uma relação de comensalismo entre ambos e favorável ao robô móvel.

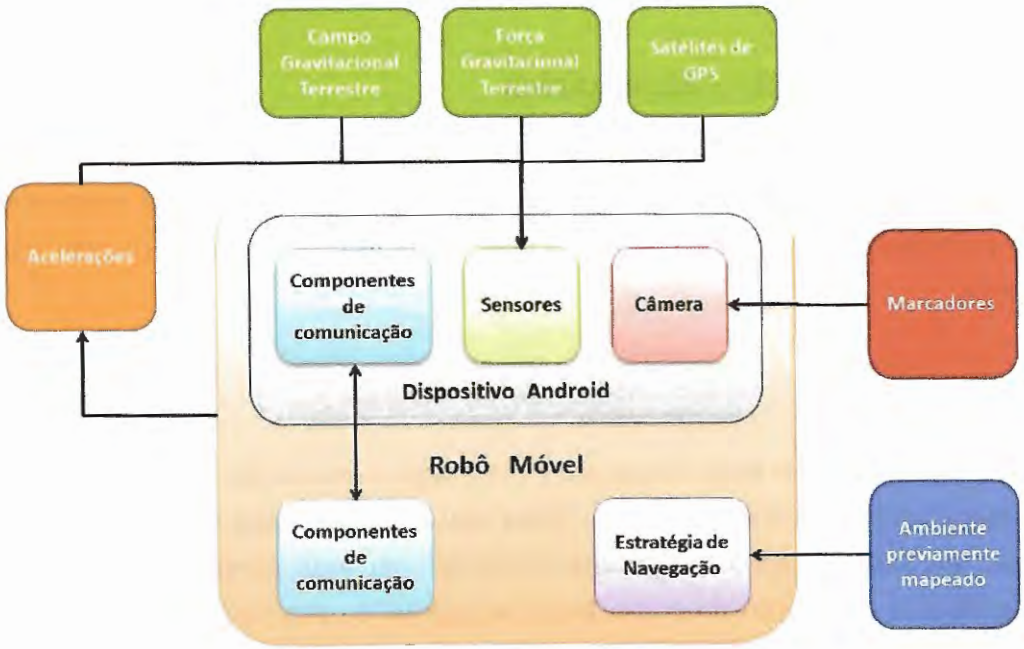

Figura 6 – Diagrama de blocos da proposição.

Isso porque o dispositivo Android pode ser programado de forma a coletar informações do meio ambiente, como acelerações, campos e forças gravitacionais, localizações globais e estímulos visuais, e em seguida processá-las, transformando-as em informações úteis para a navegação do robô móvel.

3.1 Navegação Inercial

Navegação inercial é uma técnica de navegação auto-contida, na qual informações provenientes de sensores como giroscópios, acelerômetros ou bússolas magnéticas são utilizados a fim de definir a posição e orientação de um objeto em relação a um ponto de partida definido (WOODMAN, 2007).

Assim, neste método a estimativa de localização é relativa, pois é feita com base nas localizações anteriores, o que a torna extremamente suscetível a erros, os quais são propagados e acumulados ao longo da navegação.

Normalmente se aplica uma estimativa da localização via odometria juntamente com a navegação inercial. No entanto, apesar de ser possível ser desenvolvida por meio de um sensor externo conectado ao dispositivo Android, não será alvo de estudo neste trabalho, pois entende-se que não é uma aplicação usual à sistemas Android e este tipo de sensor não está disponível em todos os sistemas robóticos.

Por fim, para realizar a localização relativa do sistema robótico em um ambiente previamente conhecido, serão alvo de estudo os sensores comumente disponíveis em dispositivos Android,: acelerômetro, giroscópio e bússola magnética, assim como a fusão de suas respostas.

3.1.1 Representação Matemática

Diversas são as maneiras possíveis de se estruturar as respostas dos sensores para representar a orientação de um corpo em um espaço tridimensional. Porém a representação escolhida pode impactar diretamente no processamento necessário para performar a fusão dos sensores.

Ângulos de Euler

Essa é a representação mais trivial para descrever a orientação de um corpo rígido em um espaço tridimensional euclidiano. Para tanto, são definidos dois sistemas de coordenadas, um anexado ao corpo girante e um fixo em uma referência global. Logo, para especificar a orientação do corpo frente a referência, são necessários três ângulos independentes, os chamados ângulos de Euler.

Essa representação tem o benefício de ser facilmente interpretada e permite realizar rotações em um eixo sem muito esforço, mas sofre de algumas limitações como o Gimbal

Lock e a impossibilidade de interpolar duas orientações. A Figura 7, a seguir, mostra tal representação, e a Figura 8 apresenta o Gimbal Lock.

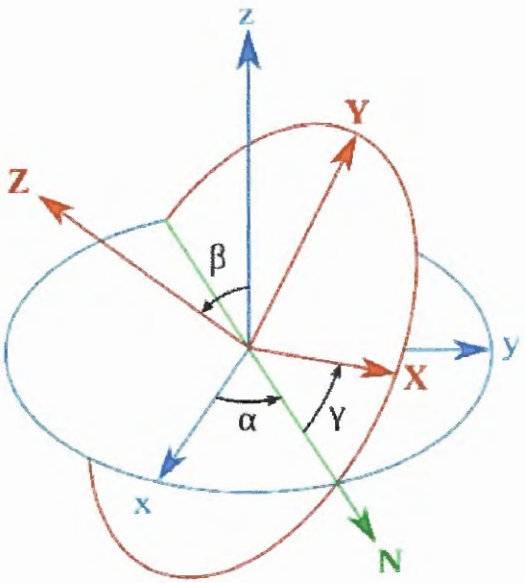

Figura 7 – Representação dos ângulos de Euler.

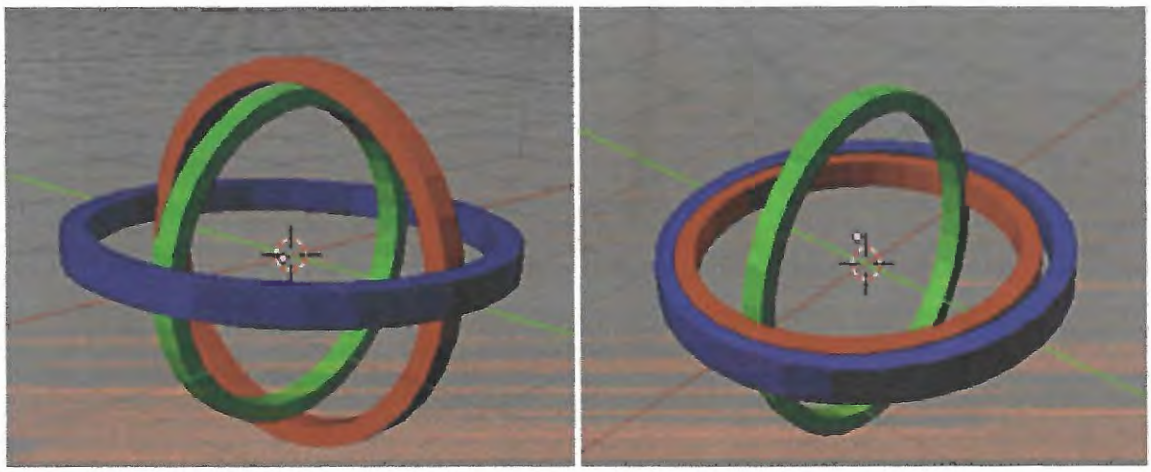

Figura 8 – Representação do Gimbal Lock.

Um exemplo simples do Gimbal Lock é executar uma rotação de 90° no eixo Y, aro vermelho da Figura 8. Nesse caso o eixo Y assume a mesma orientação do eixo Z, aro vermelho, assim, qualquer movimento posterior poderá ser explicado tanto pela rotação de Y quanto pela rotação de Z. O que acaba ocasionando problemas na cinemática dos objetos estudados.

Matriz de Rotação

Essa é a representação mais comum para manipulação gráfica em computadores e consiste em uma matriz 3x3, para um espaço tridimensional, que contém os dados necessários para fazer a transformada de um sistema de coordenada para outro, como apresentado pela Figura 9 e pelas Equações 5, 6, 7, 8 e 9.

Sua principal vantagem reside nas propriedades matemáticas às quais pode ser submetidas além da facilidade de ser processada paralelamente.

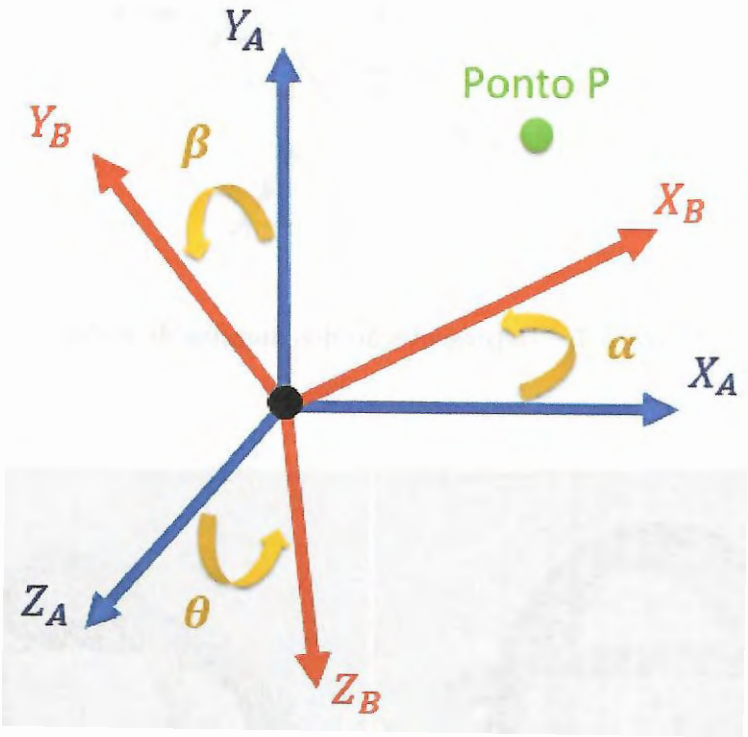

Figura 9 – Transformada de sistema de coordenadas.

Dado um ponto P, a transformação do sistema de coordenadas A para B compreende rotação nos três eixos coordenados, X, Y e Z, cada um com com sua respectiva rotação,α, β e θ, como apresentado na Equação 5.

$$\begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix} = R_x(\alpha).R_y(\beta).R_z(\theta).\begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} \tag{5}$$

As matrizes de transformação homogêneas podem ser representadas pelas Equações

6, 7 e 8, respectivamente para as rotações em X, Y e Z.

$$R_x(\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \tag{7}$$

$$R_z(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \tag{8}$$

Logo, temos que a Equação 9, representa a matriz de rotação do sistema de coordenadas A para o sistema de coordenadas B.

$$\begin{bmatrix} X_B \\ Y_B \\ Z_B \end{bmatrix} = \begin{bmatrix} c(\beta)c(\alpha) & s(\theta)s(\beta)c(\alpha) + c(\theta)s(\alpha) & -c(\theta)s(\beta)c(\alpha) + s(\theta)s(\alpha) \\ -c(\beta)s(\alpha) & -s(\theta)s(\beta)s(\alpha) + c(\theta)c(\alpha) & c(\theta)s(\beta)s(\alpha) + s(\theta)c(\alpha) \\ s(\beta) & -s(\theta)c(\beta) & c(\theta)c(\beta) \end{bmatrix} \begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix} \tag{9}$$

Porém, apesar de sua praticidade na implementação, essa representação é relativamente custosa frente ao custo computacional, pois inúmeras operações matemáticas são necessárias durante o seu processamento.

Quetérnions

Essa é a representação mais abrangente, na qual a rotação é especificada por meio de um ângulo de rotação, ω, e três eixos de rotação, que podem ser escritos em função dos versores i, j e k, como representado na Equação 10

$$Q = \omega + x \cdot i + y \cdot j + z \cdot k \tag{10}$$

À primeira vista os quatérnions parecem ser uma representação matemática complexa, porém a interpolação entre dois quatérnions é extremamente simples, por meio da chamada Interpolação Esférica Linear, SLERP em inglês, como representado na Equação 12

$$SLERP(Q_1, Q_2, p) = (1 - p)Q_1 \oplus pQ_2 \tag{11}$$

Como apresentado na Equação 10, um quatérnion é constituído por uma parte escalar, ω, e por um vetor, $x \cdot i + y \cdot j + z \cdot k$. Assim, dois quatérnions podem ser representados como pertencentes a uma esfera em um ambiente 3D, como apresentado na Figura 10.

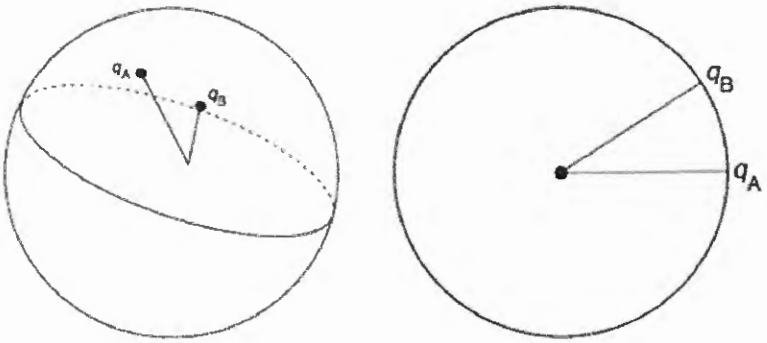

Figura 10 – Espaço esférico tridimensional que contém dois quatérnions.

Assim, denotando θ o ângulo entre os dois vetores, q_1 e q_2, pode-se chegar a trigonometria apresentada na Figura 11.

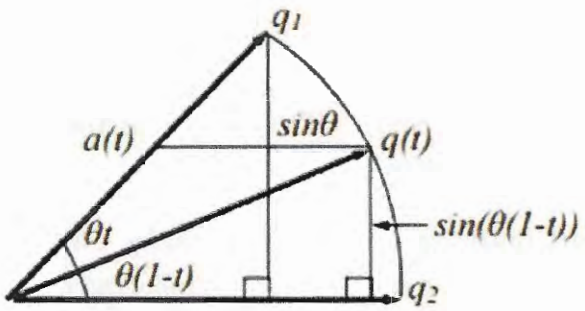

Figura 11 – Trigonometria para interpolação de dois quatérnions.

Logo, pode-se escrever a interpolação linear esférica entre dois quatérnions como:

$$q(t) = \frac{[\sin\theta(1-t)]}{\sin\theta} q_1 + \frac{\sin(\theta t)}{\sin\theta} q_2 \tag{12}$$

Desde que t esteja contido entre 0 e 1.

Assim, tem-se um interpolação com bons resultados e baixa necessidade de processamento. Por essa razão, essa será a representação utilizada neste trabalho

3.1.2 Fusão Sensorial

A fusão de sensores consiste da obtenção de leituras do ambiente através de mais de um tipo de sensor, e da combinação dos resultados adquiridos visando compensar deficiências de cada sensor e tirar proveito de uma maior quantidade de informação. Tem-se deste modo uma melhor aferição do ambiente, assim como a percepção humana consiste de um conjunto de informações de diversos sentidos, e não apenas de um (COUTO, 2012). Dessa forma espera-se obter uma maior precisão, acuracidade e robustez na definição da

posição e orientação do sistema robótico, do que seria possível se utilizados os sensores separadamente.

Filtro Complementar

Como apresentado na Seção 2.2, acelerômetro e bússola magnética podem ser utilizados para determinar a orientação global do dispositivo, porém ambos são imprecisos devido a ruídos oriundos de componentes metálicos próximos ao sensor, ou mesmo por acelerações do sistema robótico.

Sabe-se, também, que a acuracidade do giroscópio é maior do que os outros dois sensores, entretanto a integração necessária para determinar a orientação relativa ao longo do tempo pode resultar em um desvio as respostas, caso o ruído não tenha média 0, como apresentado nos gráficos da Figura 12, a seguir:

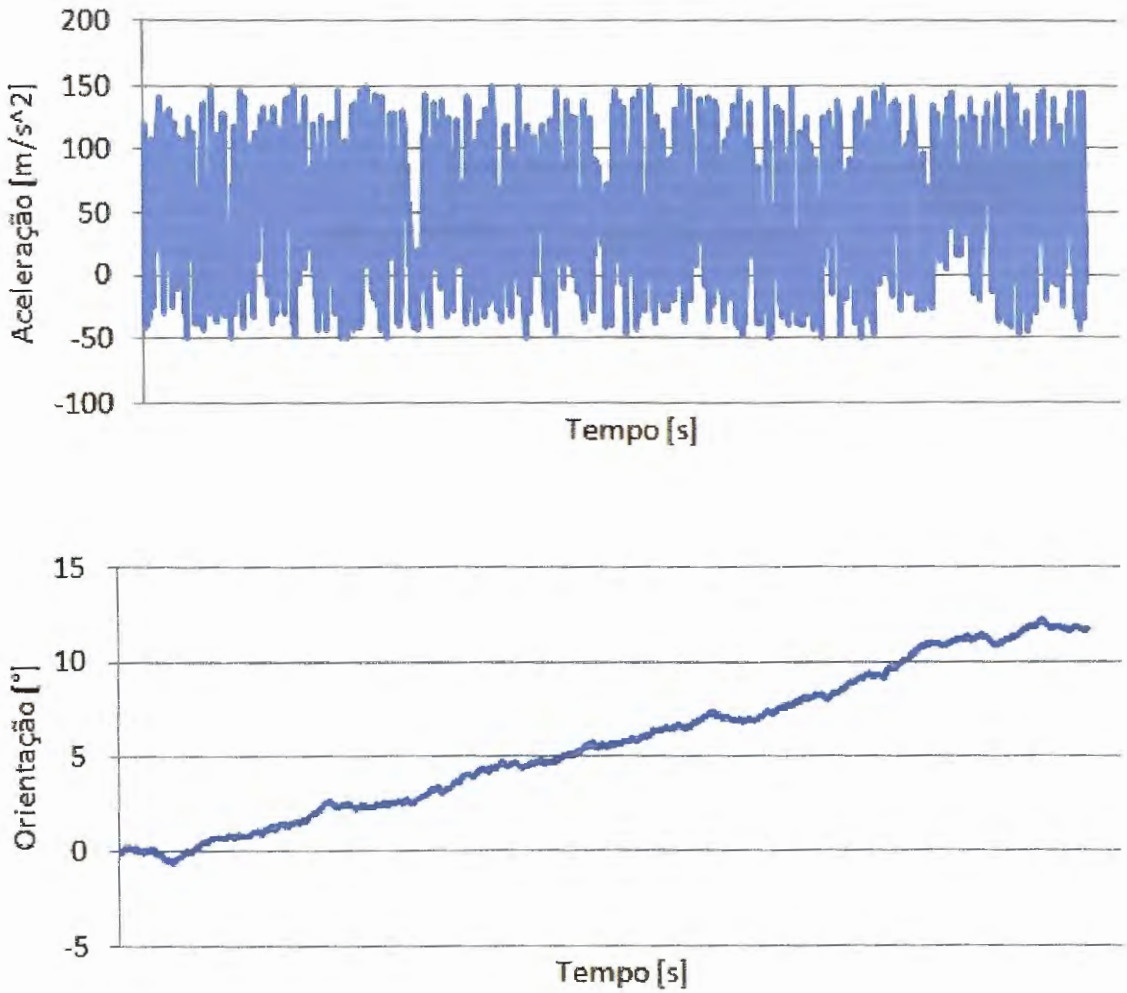

Figura 12 – Variação na orientação devido integração de ruído.

Assim, uma técnica simples mas eficiente de realizar uma fusão sensorial é por meio de um filtro complementar, como apresentado por Lawitzki (LAWITZKI, 2012), na qual é proposto um filtro passa-baixa para sinais muito ruidosos, calculando desta forma a média de um número grande de medições, e um filtro passa-alta para eliminar ruídos estacionários, deixando passar somente sinais realmente significativos.

A estrutura do filtro complementar estudado neste projeto pode ser representada pelo diagrama mostrado na Figura 13, a seguir:

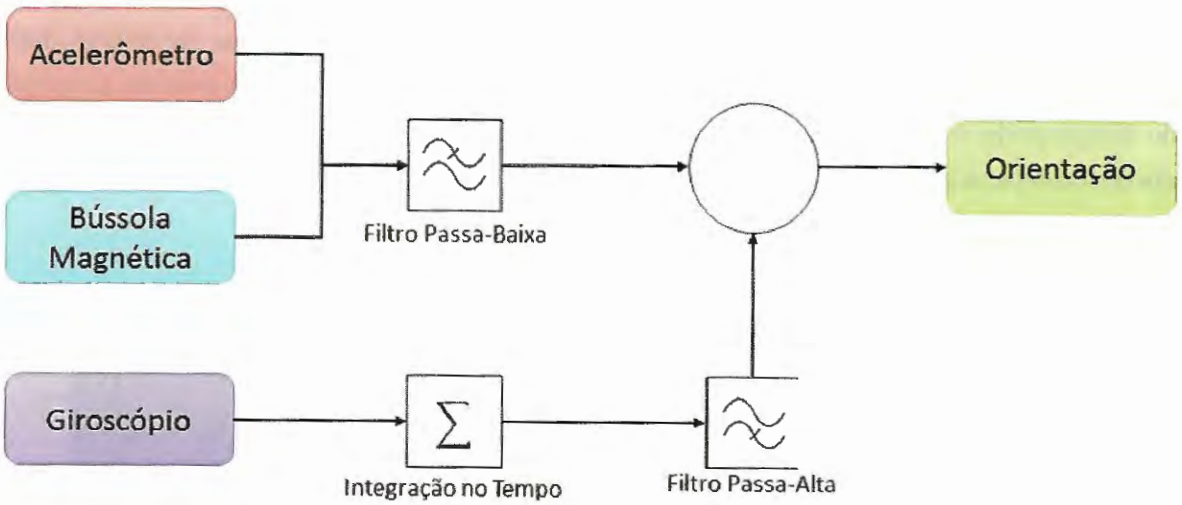

Figura 13 – Estrutura do filtro complementar.

Na Figura 14 é mostrado o comportamento ideal das etapas de funcionamento do filtro complementar proposto.

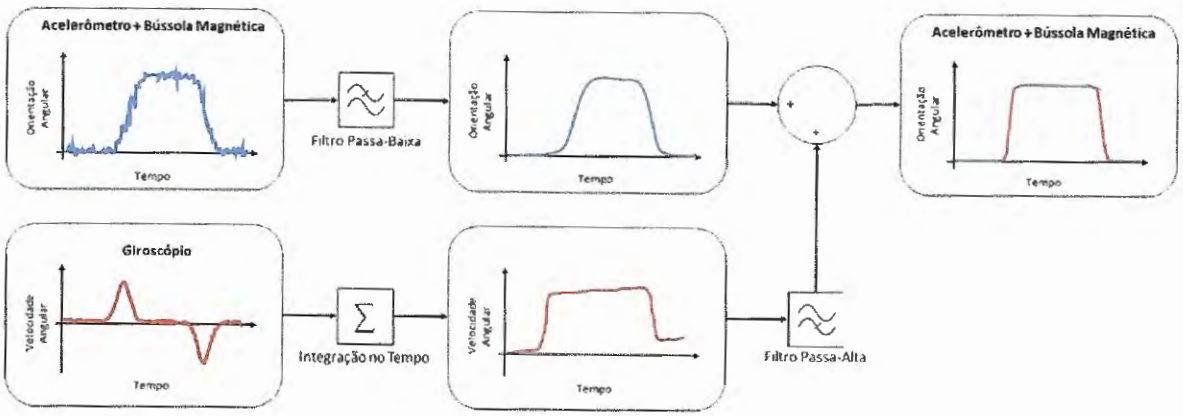

Figura 14 – Comportamento ideal do filtro complementar proposto.

Filtro de Kalman

O potencial e as propriedades que podem ser obtidos por meio de um filtro de Kalman são amplamente conhecidos e aplicados nos mais diversos sistemas da robótica móvel, sendo muito utilizado para endereçar o problema geral de tentar estimar um estado ou posição em um processo controlado com tempo discreto (WELCH, 2006).

A ideia principal por trás do filtro de Kalman é retroalimentar o modelo do sistema com uma previsão do estado futuro e atualizar essa informação com o valor medido quando disponível. Ambas as informações, previsão e medição, assim como suas incertezas, se complementam. Dessa forma o algoritmo do filtro pode ser representado como na Figura 13

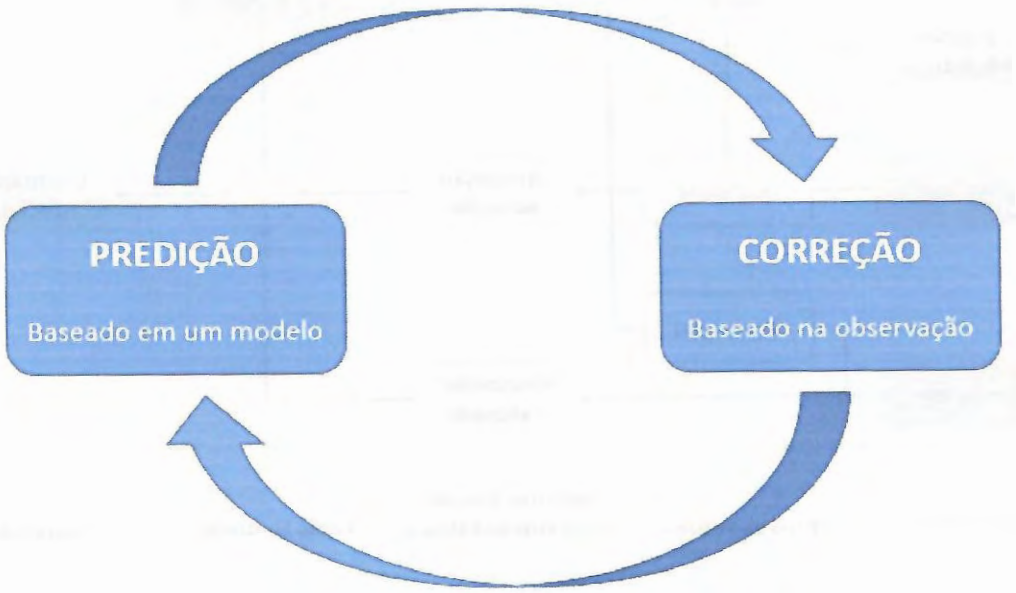

Figura 15 – Algoritmo do Filtro de Kalman.

As equações utilizadas no filtro de Kalman não serão abordadas neste documento, já que não foram alvo de estudo, uma vez que o sistema operacional Android já possui filtros de Kalman implementados para cada sensor específico.Mas são parte essencial da segunda estratégia de fusão sensorial, apresentada a seguir.

Fusão de Sensores Virtuais

Como apresentado na Seção 2.2, o Android possui disponível dois sensores virtuais, "Orientação Absoluta" e "Giroscópio Calibrado", com excelentes resultados graças a utilização de Filtros de Kalman. Assim sendo robustos a ruídos intrínsecos à aplicação estudada, o que é extremamente positivo.

O sensor "Giroscópio Calibrado" é capaz de fornecer uma orientação relativa do dispositivo de forma rápida e praticamente sem desvios ou ruídos, já o sensor "Orientação Absoluta" fornece a orientação global de forma precisa porém não tão rápida.

Portanto a ideia proposta e estudada é realizar uma fusão sensorial desses dois sensores. E uma vez que o dispositivo em Android também pode receber informações sobre a mecânica do robô, por exemplo, se o robô está parado ou se movimentando linearmente, ou fazendo uma curva, essa informação pode ser utilizada de forma a melhorar a fusão sensorial entre os sensores virtuais, como apresentado na Figura 16.

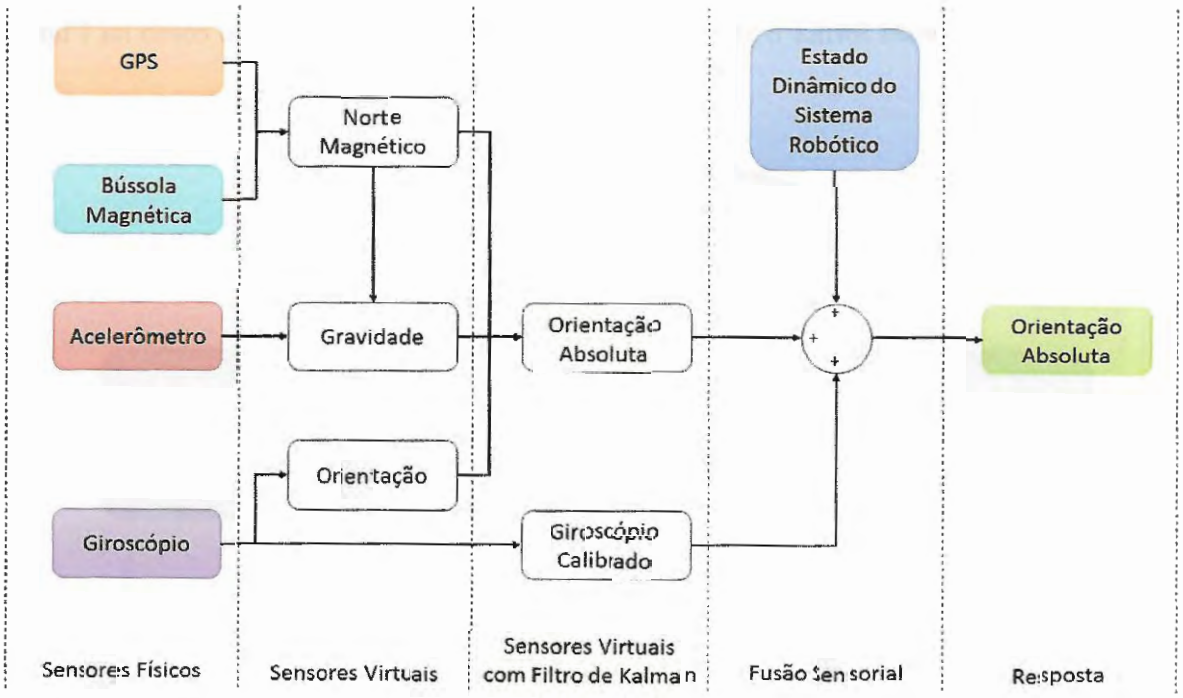

Figura 16 – Fusão sensorial entre sensores virtuais.

Uma vez que a representação padrão no Android para ambos os sensores "Giroscópio Calibrado" e "Orientação Absoluta" são quatérnions, a fusão sensorial pode ser feita facilmente por meio de uma interpolação SLERP, como representado na Equação 13, na qual o coeficiente "p" é diretamente relacionado ao estado dinâmico do robô.

$$Orientação_{Fusão_{Sensorial}} = SLERP(Orientação_{Giroscópio_{Calibrado}}, Orientação_{Orientação_{Absoluta}}, p) \tag{13}$$

Dessa forma a resposta da orientação pode ser mais rápida quando o robô está executando curvas, ou mais acurada quando o robô está parado ou se movimentando linearmente.

Um pseudocódigo é apresentado a seguir, na Figura 17.

```
INÍCIO
    Inicia Sensores Físicos (Giroscópio, Acelerômetro, Bússola Magnética, GPS);

    SE (Nova informação disponível)
        Velocidade_Angular <- Giroscópio;
        Aceleração <- Acelerômetro;
        Norte_Bússola <- Bússola Magnética;
        Latitude, Longitude <- GPS;
    FIM SE;

    Norte_Magnético <- Mapa_Mundi_Magnético(Norte_Bússola, Latitude, Longitude);

    SE (Aceleração == 9.81)
        Gravidade <- Subtração_Vetorial (Aceleração, Norte_Magnético);
    FIM SE;

    Orientação <- Integração_Temporal (Velocidade_Angular);

    Giroscópio_Calibrado <- Filtro_de_Kalman (Orientação);

    Orientação_Absoluta <- Filtro_de_Kalman (Norte_Magnético, Gravidade, Orientação);

    SE (Robô parado)
        Orientação_Final <- SLERP (0.1, Giroscópio_Calibrado, Orientação_Absoluta);
    FIM SE;

    SE (Robô se movendo linearmente)
        Orientação_Final <- SLERP (0.5, Giroscópio_Calibrado, Orientação_Absoluta);
    FIM SE;

    SE (Robô rotacionando)
        Orientação_Final <- SLERP (0.9, Giroscópio_Calibrado, Orientação_Absoluta);
    FIM SE;

    SE (Movimentação da Orientação_Absoluta >> Movimentação do Giroscópio_Calibrado)
        Orientação <- SLERP (1.0, Giroscópio_Calibrado, Orientação_Absoluta);
    FIM SE;
FIM;
```

Figura 17 – Pseudocódigo referente a implementação da fusão sensorial entre sensores virtuais.

3.2 Visão Computacionall

Visão computacional é um conceito abrangente que permite ao computador não somente receber imagens de um sensor óptico, mas também localizar, identificar e seguir objetos (Pacha, 2013). Os padrões a serem reconhecidos podem ser criados artificialmente por meio de formas geométricas, como a Figura 18, ou também podem ser objetos naturais, presentes no ambiente em que o sistema robótico opera. Para tanto diversas podem ser as técnicas de detecção, como a cor, forma, tamanho ou posição do mesmo.

Figura 18 – Ambiente com marcadores não naturais.

3.2.1 ARToolKit

ARToolKit (*Argumented Reality Toolkit*) é uma biblioteca multiplataforma com código aberto e gratuita apropriada para desenvolver aplicações de Realidade Aumentada (RA), desenvolvida inicialmente por Hirokazu Kato do Instituto de Ciência e Tecnologia de Nara, Japão, e em sua continuidade pelo *Human Interface Laboratory* (HIT Lab) da Universidade de Washington.

Essa biblioteca faz uso de técnicas de visão computacional para o reconhecimento de padrões e inserção dos objetos virtuais no ambiente real.

A biblioteca ARToolkit é construída em C/C++ e oferece suporte a programadores para o desenvolvimento de aplicações de Realidade Aumentada (RA), utilizando o rastreamento óptico, que implementa técnicas de visão computacional, para identificar e estimar em tempo real a posição e a orientação de um marcador em relação ao dispositivo de captura de vídeo. Assim, o cálculo da correlação entre os dados estimados do marcador real e sua imagem, possibilita posicionar objetos virtuais alinhados à imagem do marcador, permitindo o desenvolvimento rápido de uma vasta gama de aplicações de RA (KATO, 1999).

As principais funcionalidades e recursos da biblioteca são:

- Identificação dos marcadores;
- Definição de um sistema referencial cartesiano por marcador em relação à posição da câmera;
- Posição, orientação e rastreamento de um objeto baseado em uma única câmera;
- Rastreamento com marcadores relativamente simples;
- Calibração da câmera relativamente simples;
- Rápido o suficiente para aplicações em tempo real;
- Multi-plataforma (SGI IRIX, Linux, MacOS e Windows);

O ARToolKit fornece funções predefinidas que devem ser chamadas em uma ordem específica para o desenvolvimento de uma aplicação de RA. Embora também seja possível utilizar diferentes funções separadamente.

Visando minimizar as dependências com o sistema operacional o ARToolKit utiliza a OpenGL para parte da renderização, GLUT para manipular a aparência do aplicativo e dependências de *hardware* e a API padrão de cada plataforma, win32 no Windows por exemplo.

Na Figura 19, a seguir são mostradas as relações entre a aplicação desenvolvida, o ARToolkit e as Bibliotecas utilizadas.

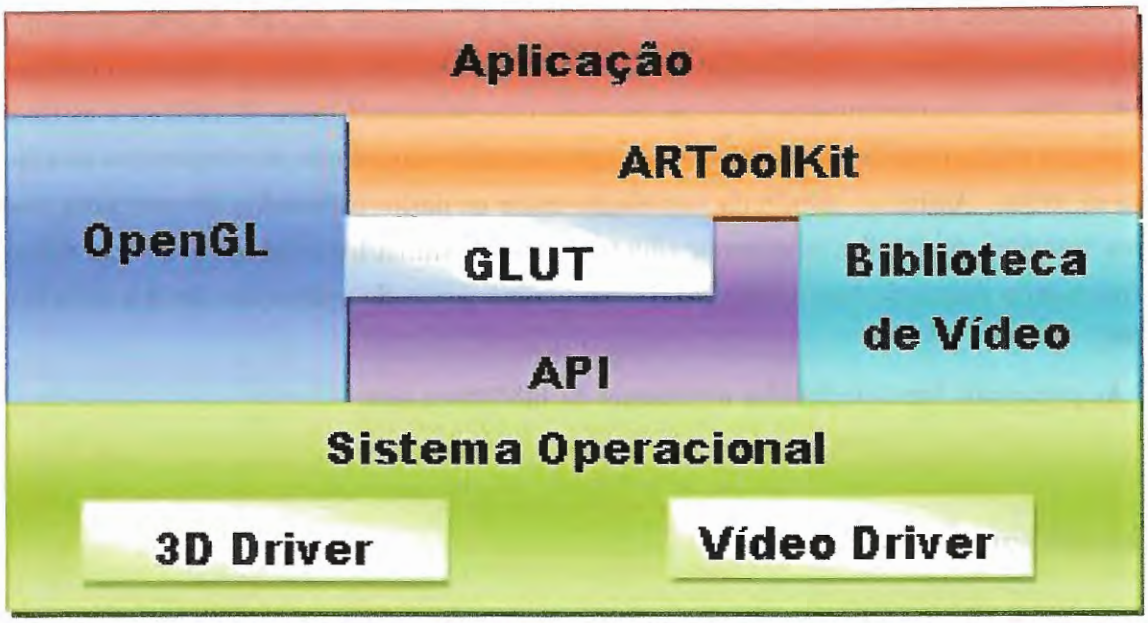

Figura 19 – Relações entre ARToolkit e as demais bibliotecas.

Dessa forma, o ARToolKit é constituido basicamente por 3 módulos:

- **"Módulo AR:"** módulo central com rotinas de rastreamento de marcadores, calibração e coleta de parâmetro;
- **"Módulo de vídeo:"** coleção de rotinas de vídeo para capturar os quadros de entrada de vídeo. Este é um invólucro em torno do padrão das rotinas de captura de vídeo;
- **"Módulo Gsub Lite:"** coleção de rotinas gráficas baseada no OpenGL e nas bibliotecas GLUT, independente de qualquer kit de ferramentas de janelas.

Os módulos se relacionam em uma sequência direta, para que se necessário sejam substituídos facilmente pelo desenvolvedor. Esta sequência é apresentada na Figura 20, a seguir.

Figura 20 – Módulos ARToolKit.

O rastreamento implementado no ARToolKit estima a posição de marcadores em relação à câmera, tornando possível desenvolver aplicações que necessitem conhecer a posição e orientação de elementos.

Os marcadores reconhecidos pelo ARToolkit consistem em duas figuras geométricas quadradas, uma preta maior que contém em seu interior uma branca menor. A menor por sua vez contém, no seu interior, símbolos para identificação do marcador, como mostrado na Figura 21. Os símbolos podem ou não ser figuras geométricas, porém quanto maior for a complexidade deste, maior será o tempo de processamento e possivelmente menor será a taxa de reconhecimento.

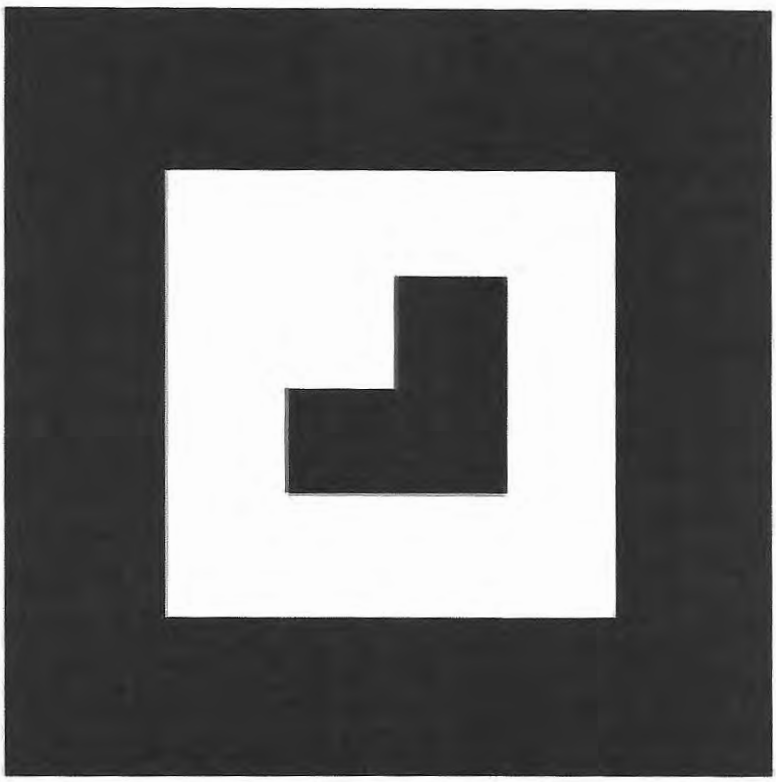

Figura 21 – Exemplo de um marcador padrão para a ARToolKit.

Para melhorar o reconhecimento do marcador, é recomendado manter-se uma moldura branca envolvendo as bordas do marcador, assim promovendo o contraste e viabilizando seu reconhecimento sobre superfícies de cores escuras.

O reconhecimento de padrões no ARToolKit identifica quatro vértices de regiões quadradas, contidas na imagem de vídeo, e compara os símbolos do seu interior com os gabaritos dos marcadores previamente cadastrados pelo usuário (CLAUS, 2005).

Uma vez que haja o reconhecimento de um marcador, o rastreamento do ARToolKit extrai algumas informações com relação a detecção e identificação dos marcadores, além de estimar sua posição e orientação.

Na Figura 22 são esquematizados os sistemas de coordenadas com os quais o ARToolKit trabalha, e utiliza para calcular os parâmetros necessários para a inserção de um objeto virtual na imagem real.

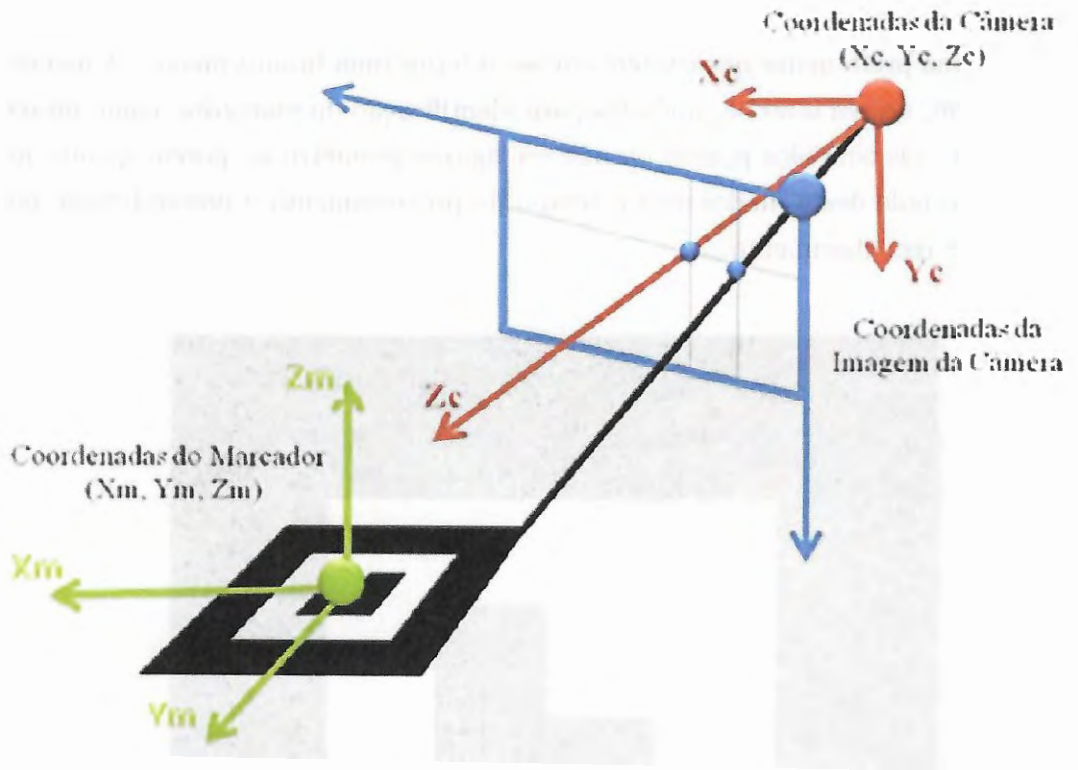

Figura 22 – Relação entre os sistemas de coordenadas do marcador e da câmera.

As coordenadas do marcador e as coordenadas da câmera se relacionam por intermédio de uma matriz 3x4 denominada "matriz de transformação". A multiplicação da matriz de transformação "T" por um ponto 3D no marcador (Xm, Ym, Zm), resulta no ponto correspondente no sistema de coordenadas da câmera (Xc, Yc, Zc).

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = T \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \tag{14}$$

Para cada elemento da matriz tem-se:

$$R_{11} = \cos\theta . \cos\varphi \tag{15}$$

$$R_{12} = -\cos\phi . \sin\varphi + \sin\phi . \sin\theta . \cos\varphi \tag{16}$$

$$R_{13} = \sin\phi . \sin\varphi + \cos\phi . \sin\theta . \cos\varphi \tag{17}$$

$$R_{21} = \cos\theta . \sin\varphi \tag{18}$$

$$R_{22} = \cos\phi.\cos\varphi + \sin\phi.\sin\theta.\sin\varphi \tag{19}$$

$$R_{23} = -\sin\phi.\cos\varphi + \cos\phi.\sin\theta.\sin\varphi \tag{20}$$

$$R_{31} = -\sin\theta \tag{21}$$

$$R_{32} = \sin\phi.\cos\theta \tag{22}$$

$$R_{33} = \cos\phi.\cos\theta \tag{23}$$

$$T_1 = X \tag{24}$$

$$T_2 = Y \tag{25}$$

$$T_3 = Z \tag{26}$$

Os ângulos de giro são apresentados na Figura 23.

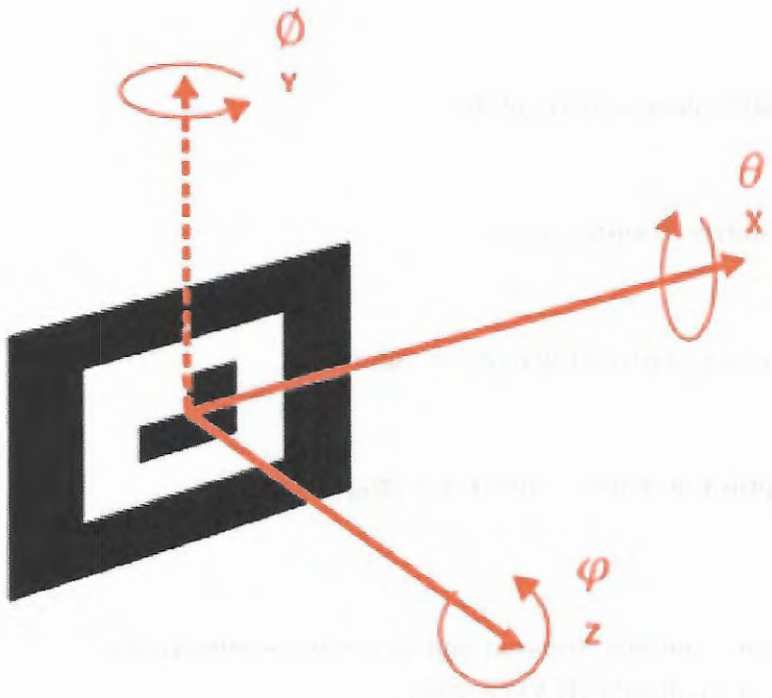

Figura 23 – Ângulos de giro do marcador.

Assim, é possível estimar o relacionamento entre as coordenadas 3D do mundo real e as coordenadas 2D da imagem.

Uma vez estimada a matriz transformação, a API "OpenGL" é utilizada para ajustar a câmera virtual, posicionar e desenhar o objeto virtual alinhado na visualização do marcador real (PIEKARSKI, 2002).

Basicamente para o desenvolvimento de uma aplicação de RA, Realidade Aumentada, utilizando o ARToolKit, deve-se seguir o seguinte procedimento:

1) Inicializar a aplicação;

1.a) Inicializar a configuração de vídeo;

1.b) Ler o arquivo de cadastramento de marcadores;

1.c) Ler os parâmetros da câmera;

2) Capturar um quadro de vídeo;

3) Detectar e identificar os marcadores;

4) Calcular a Matriz Transformação;

5) Desenhar o objeto virtual referente ao marcador;

6) Fechar a captura de vídeo e encerrar o programa.

Para a execução contínua do programa os passos de número 2 a 5 devem ser repetidos, até a aplicação ser finalizada (KATO, 2000).

O funcionamento padrão da biblioteca está descrito na Figura 24, desde a captura do vídeo até a inserção de um objeto virtual no mesmo.

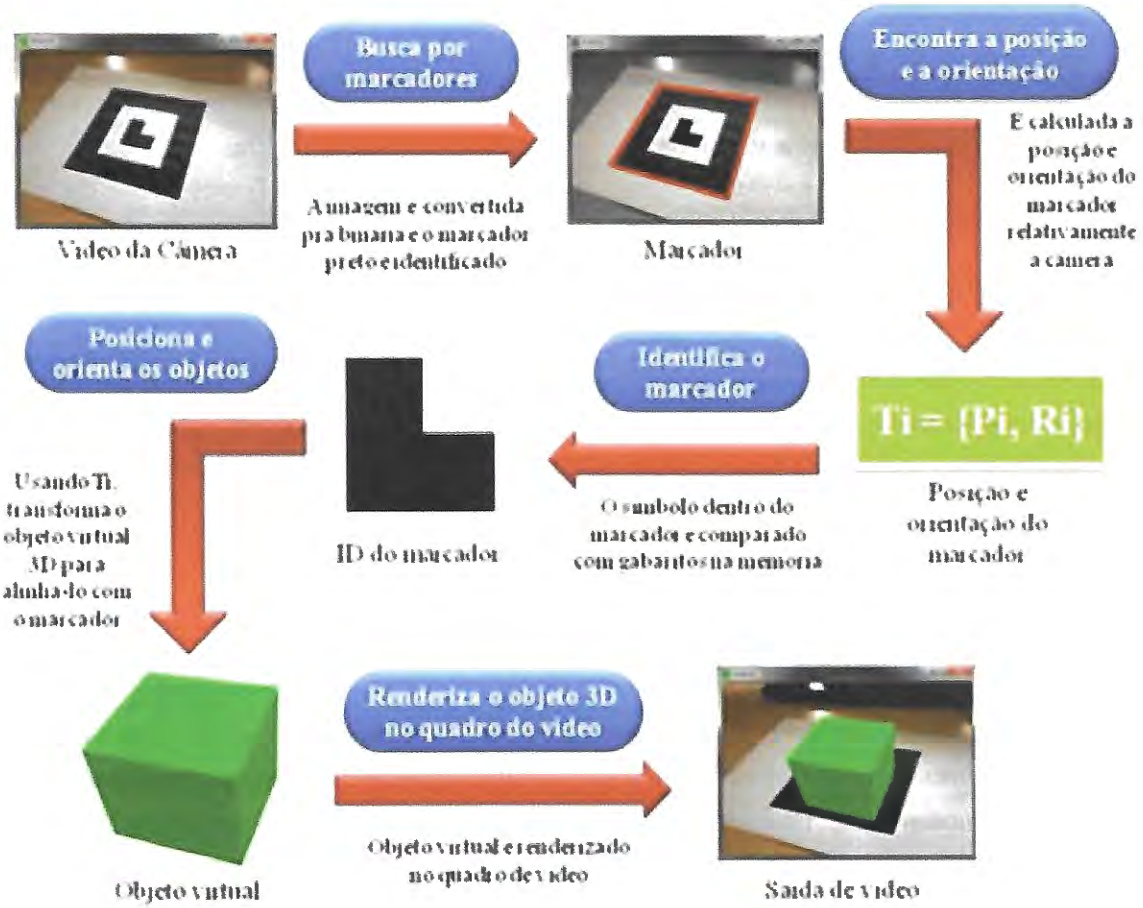

Figura 24 – Funcionamento da ARToolKit.

3.2.2 OpenCV

OpenCV é uma biblioteca multiplataforma com código aberto e gratuita, tanto para uso acadêmico quanto comercial, apropriada para desenvolver aplicações relacionadas a processamento de imagem e visão computacional de forma eficiente e em tempo real, desenvolvida e gerida pela Intel desde 2000.

Essa biblioteca tem em sua estrutura uma ampla gama de ferramentas para o tratamento de imagens, desde filtros de ruído até análise de movimentos e reconhecimento de padrões.

A biblioteca OpenCV foi construída originalmente em C e migrada em sua segunda versão para a linguagem C++, visando sempre não somente códigos abertos, mas também otimizados e portáteis, de forma que os desenvolvedores possam se valer dessa biblioteca de maneira fácil e intuitiva para a criação de novas aplicações e o desenvolvimento de pesquisas e inovações relacionadas.

As principais funcionalidades e recursos da biblioteca são:

- Manipulação de dados de imagens
- Entrada e saída de imagens e vídeos
- Manipulação de matrizes e vetores
- Rotinas de álgebra linear
- Estruturas de dados dinâmicos
- Processamento básico de imagens
- Calibração dos instrumentos de captura
- Reconhecimento de objetos
- Análise de movimento de objetos
- Desenvolvimento de interfaces

A OpenCV possui uma estrutura modular, na qual os pacotes compartilham as mesmas bibliotecas e funções base. Dessa forma, formam-se módulos, dos quais temos:

- **"Core:"** Módulo básico, responsável pela estrutura de dados, matemática básica e álgebra linear e funções básicas como: gerenciamento de memória e manipulação de erros;
- **"Imgproc:"** Módulo de processamento de imagens, incluindo filtros, transformações geométricas e conversões de cores;
- **"Video:"** Módulo de análise de vídeo, sendo responsável, por exemplo, pela análise de movimento de objetos;
- **"Calib3d:"** Módulo responsável pela calibração de câmeras, mono e estéreo;
- **"Objdetect:"** Módulo de detecção de objetos ou instâncias, como rostos, pessoas e carros;
- **"Highgui:"** Módulo de alto nível responsável pelas interfaces com o usuário e com a captura de vídeo;

O método de detecção e análise de movimento de marcadores escolhido consiste na definição de valores de cor para cada um dos marcadores. O sistema de cores utilizado foi o HSB, "hue, saturation and brightness", o qual é composto por parâmetros de matriz, saturação e brilho. Assim, devendo ser definido para cada marcador os três parâmetros, assim como os intervalos de variação aceitáveis.

A biblioteca OpenCV é utilizada então de forma a:

1) Identificar os pixels que estão dentro dos intervalos definidos como um marcador;

2) Converter imagem em binária, sendo 1 os pixels com cores do marcador e 0 os demais;

3) Pixels contíguos são definidos como parte de um mesmo objeto, assim identifica-se as bordas dos objetos;

4) Definição do marcador como sendo o objeto na imagem com a maior área;

5) Extracação de informações do marcador, como área e posição em relação ao sistema 2D da imagem;

Essa tratativa de definição por área é aplicada para reduzir o ruído inerente ao método. Entretanto, ainda sim o método é extremamente suscetível a variações na iluminação no ambiente, calibração de luminosidade da câmera e a presença de outros objetos com a mesma cor que o marcador.

3.3 Plataforma Robótica

Diversos são os projetos que envolvem veículos autônomos, como por exemplo o Agribot, plataforma robótica modular capaz de se locomover em ambientes típicos da área agrícola, ou a Empilhadeira Robótica, plataforma robótica modular capaz de se locomover em ambientes industriais e armazéns inteligentes.

Figura 25 – Exemplos de veículos autônomos. (a) Agribot (b) Empilhadeira Robótica.

Porém cada sistema robótico possui sua particularidade, sua restrição no movimento, nos acionamentos e nos requisitos. Como o principal objetivo do projeto é auxiliar a navegação autônoma de um sistema robótico genérico, desenvolveu-se um robô com o mínimo de restrições possíveis.

3.3.1 Estrutura Base

Quatro motores independentes com encoder foram posicionados em dois eixos paralelos e acoplados a rodas omnidirecionais mecanum. Cada roda mecanum é constituída por 7 roletes plásticos dispostos a 45°, como apresentado na Figura 19.

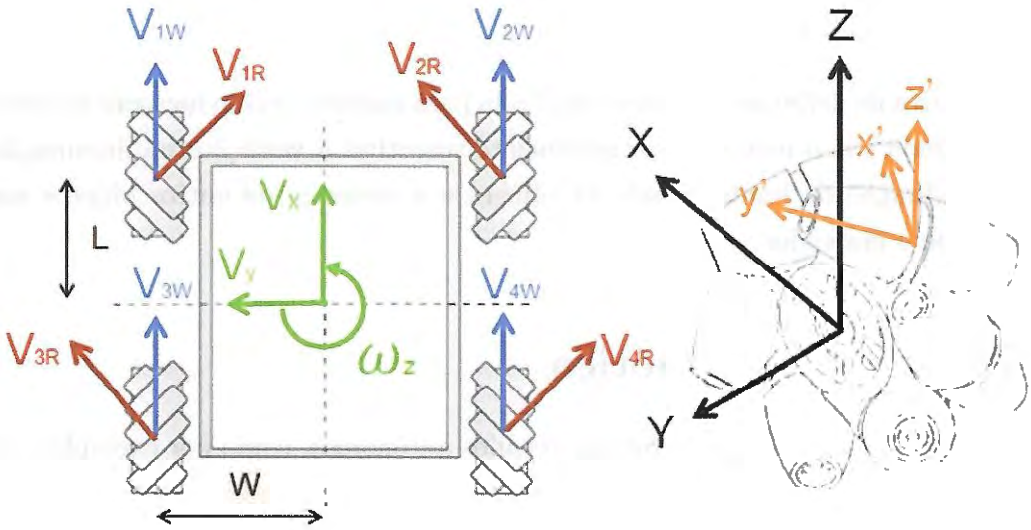

Figura 26 – Estrutura do chassis e das rodas com sua respectiva cinemática.

Por meio dessa estrutura é possível mover o sistema robótico em qualquer um dos três graus de liberdade disponíveis (x, y e ω). E a cinemática segue as equações apresentadas abaixo:

$$\begin{bmatrix} V_x \\ V_y \\ \omega_z \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -T & T & -T & T \end{bmatrix} \begin{bmatrix} V_{1w} \\ V_{2w} \\ V_{3w} \\ V_{4w} \end{bmatrix} \tag{27}$$

Onde:

$$T = \frac{1}{(W + L)} \tag{28}$$

$$V_{nW} = R_W . \dot{\theta}_n \tag{29}$$

As dimensões básicas do sistema robótico utilizado é apresentado na Figura 27, a seguir:

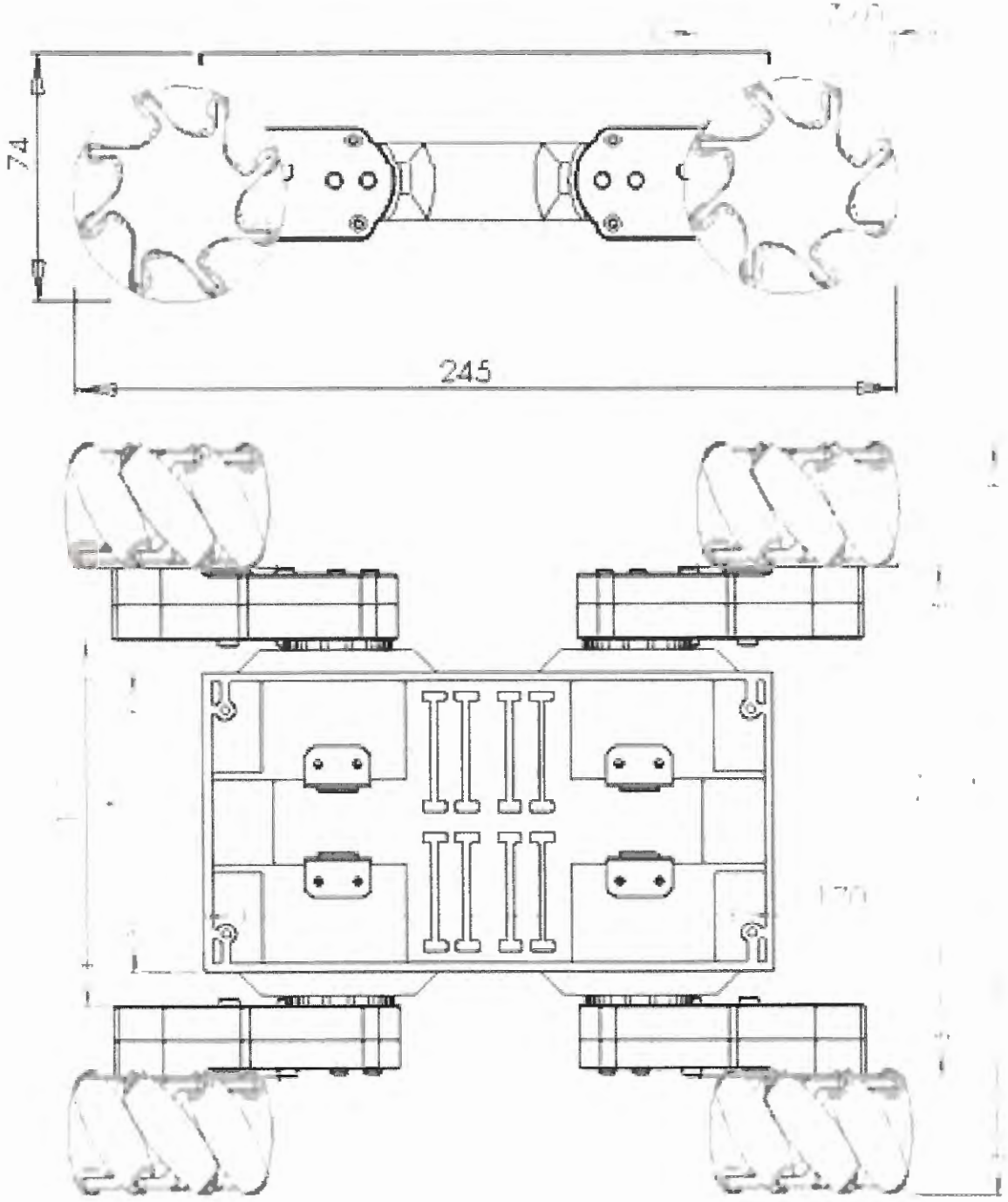

Figura 27 – Dimensões básicas do sistema robótico.

O robô também conta com um mecanismo pan-tilt que permite girar o dispositivo Android em relação ao sistema robótico. Esse movimento é importante para permitir que

a câmera da plataforma Android possa visualizar marcadores tanto na frente quanto nas laterais do robô, assim expandindo as possibilidades de navegação.

Por fim, o sistema robótico desenvolvido também contém alguns sensores de distância, infravermelhos e ultrasônicos, como apresentado na Figura 28, para que no caso de uma colisão iminente a rota possa ser alterada, assim desviando de um obstáculo não previsto.

Figura 28 – Sistema robótico e seus sensores.

3.3.2 Controle de Baixo Nível

Para realizar a comunicação entre o sistema robótico e o dispositivo Android utilizou-se um placa de desenvolvimento Arduino Mega ADK, a qual possui protocolo de comunicação por fio para placas da plataforma Android. E no intuito de permitir que o sistema robótico fosse navegado pela plataforma Android, um sistema de controle de baixo nível foi desenvolvido no Arduino, de forma a receber comandos de alto nível vindo do Android, como orientação de movimentação e velocidade. Um sistema de controle das rodas por meio de um controlador Proporcional Integral Derivativo – PID, baseado na velocidade não é eficaz para a estrutura robótica desenvolvida, porque uma vez que uma das rodas erre ou mesmo demore pra atingir a velocidade, o robô começará a se movimentar em uma direção diferente da desejada. Assim, foi implementado um controle PID baseado em velocidade em uma roda e três controles PID baseado em distância para as demais rodas, tendo como entrada a distância percorrida pela primeira roda e levando em consideração a cinemática do movimento a ser executado, como apresentado na Figura 29.

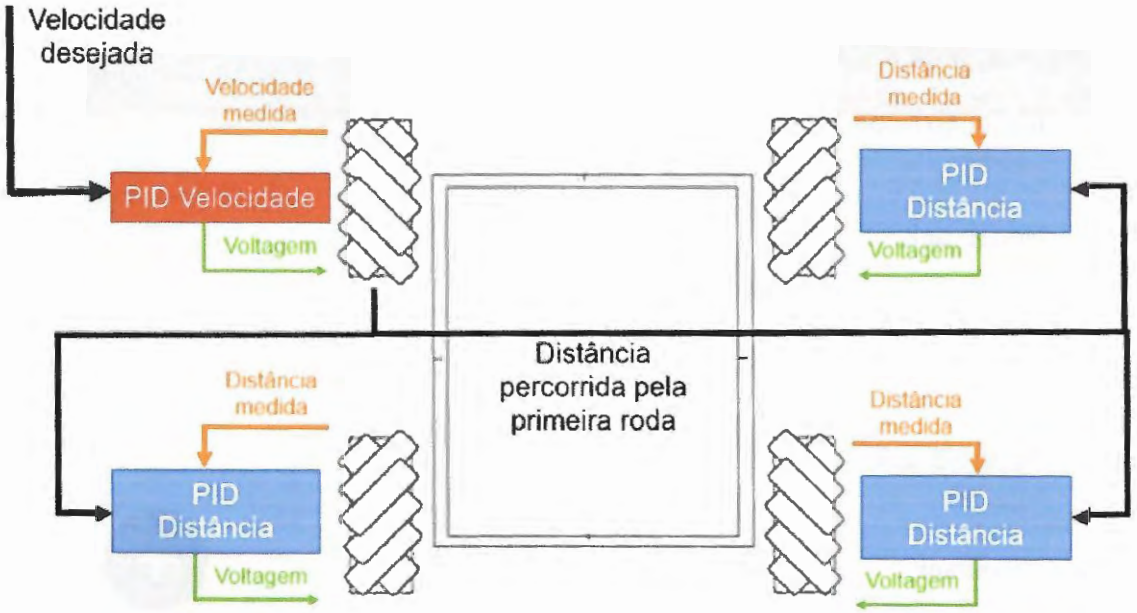

Figura 29 – Esquemático do controle de baixo nível.

3.4 Aplicativo

Unindo todas as técnicas estudadas e apresentadas anteriormente, desenvolveu-se um aplicativo para o sistema operacional Android capaz de extrair do ambiente as informações necessárias para o auxílio à navegação autônoma de um robô móvel e enviá-las por diversos modos de comunicação para o mesmo.

O aplicativo, nomeado de Teseu, pode ser encontrado no endereço: <https://drive.google.com/file/d/1pW49l_eR3JTxZZA7GSap19O6GizLvbiT/view?usp=sharing>.

Instalação

Uma vez baixado, o arquivo "Teseu.apk"deve ser movido para o dispositivo móvel. Preferencialmente para a pasta "Memória Interna / Download". Para instalar o aplicativo é necessário um aplicativo com acesso as pastas internas do dispositivo. A sugestão é o aplicativo *File Explorer*, disponível em <https://play.google.com/store/apps/details?id=com.mauriciotogneri.fileexplorer>. Deve-se então navegar até a pasta que contém o arquivo "Teseu.apk"a abrí-lo para realizar a instação, como apresentado nas Figuras 30, 31 e 32 respectivamente.

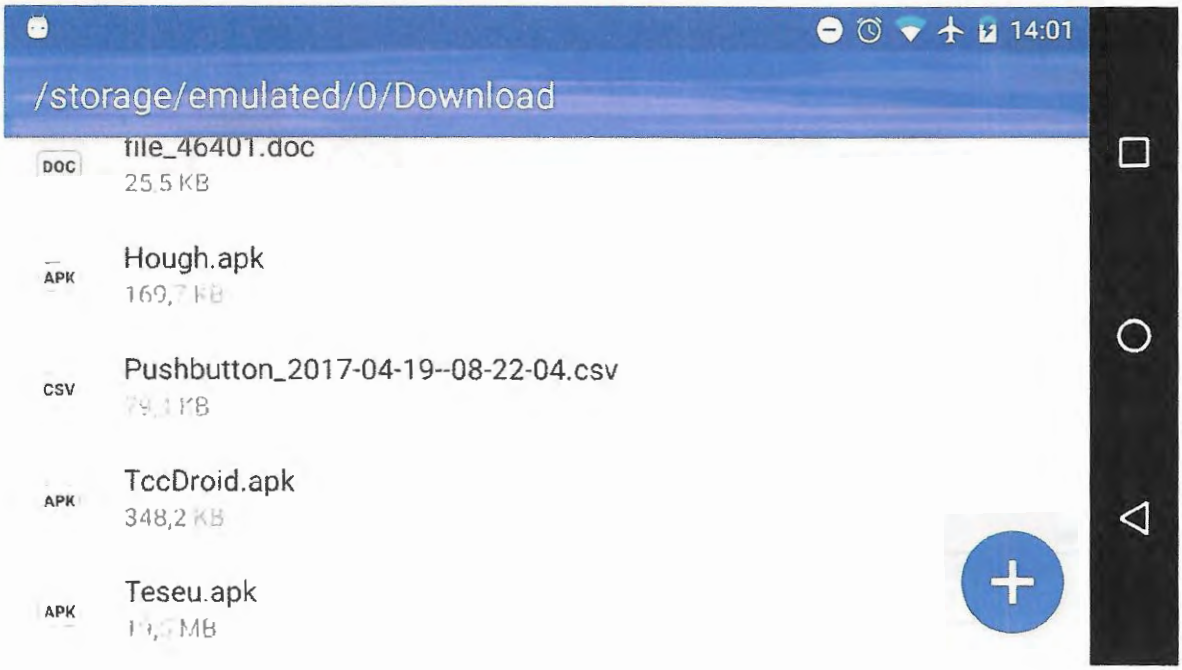

Figura 30 – Arquivo "Teseu.apk na memória interna do dispositivo visto por meio do aplicativo *File Explorer*.

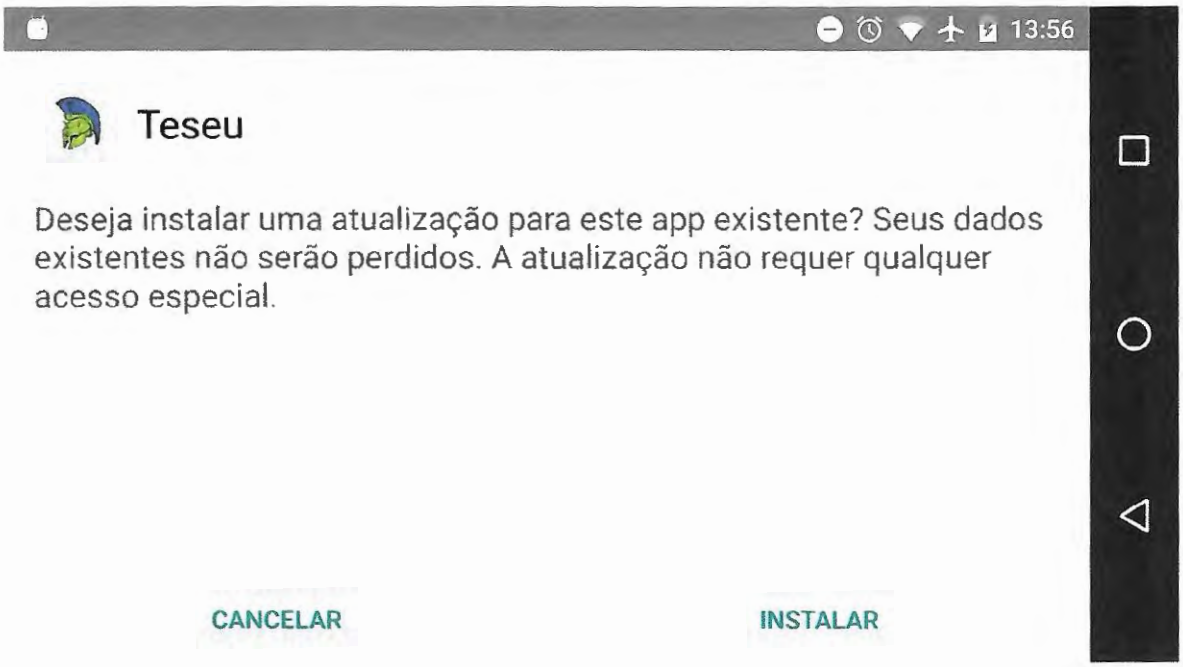

Figura 31 – Aplicativo Teseu aberto para ser instalado.

Figura 32 – Aplicativo Teseu instalado com sucesso.

O aplicativo pode então ser acessado pelo ícone no painel de aplicativos, como apresentado na Figura 33.

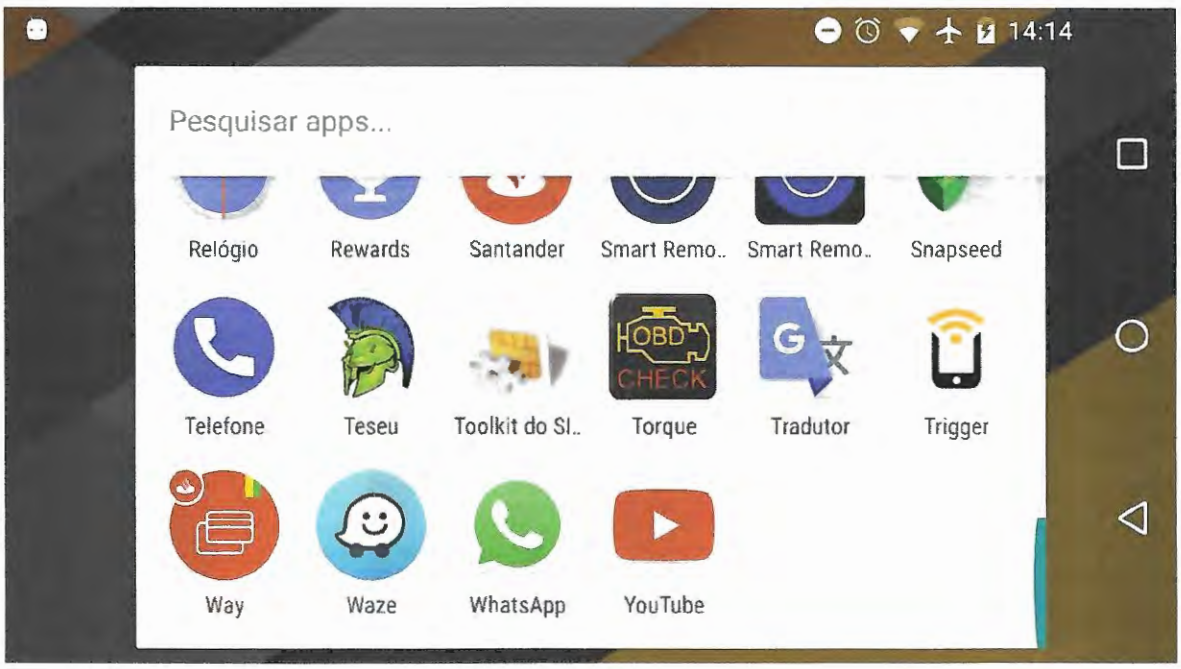

Figura 33 – Aplicativo Teseu no painel de aplicativos disponíveis no dispositivo Android.

Tela Inicial

A Figura 34 mostra a página principal do aplicativo Teseu. Nesta página é possível acessar informações sobre o aplicativo, determinar o modo de comunicação entre o dispositivo Android e o robô móvel, definir quais informações são necessárias para auxiliar a navegação autônoma do robô e iniciar a extração dos dados.

Figura 34 – Página principal do aplicativo Teseu.

Menu de Informações

Tocando no botão "Informação"o usuário tem acesso a um menu com informações detalhadas de cada configuração e opção possível de ser feita com o aplicativo.

Figura 35 – Menu de informações disponíveis no aplicativo Teseu.

3.4.1 Orientação

Caso seja de interesse do usuário, o aplicativo pode enviar para o robô móvel os dados relativos a orientação global do sistema, apresentado no Capítulo 3. Para tanto deve-se habilitar o botão "Orientação", como apresentado na Figura 36.

Figura 36 – Seleção para que o aplicativo processe a orientação do sistema.

Uma vez iniciada a execução do aplicativo com a seleção de orientação habilitada, o usuário passará para a tela de orientação, apresentada na Figura 37 e o envio de pacote de dados para o robô móvel será continua até que o usuário saia da tela de orientação.

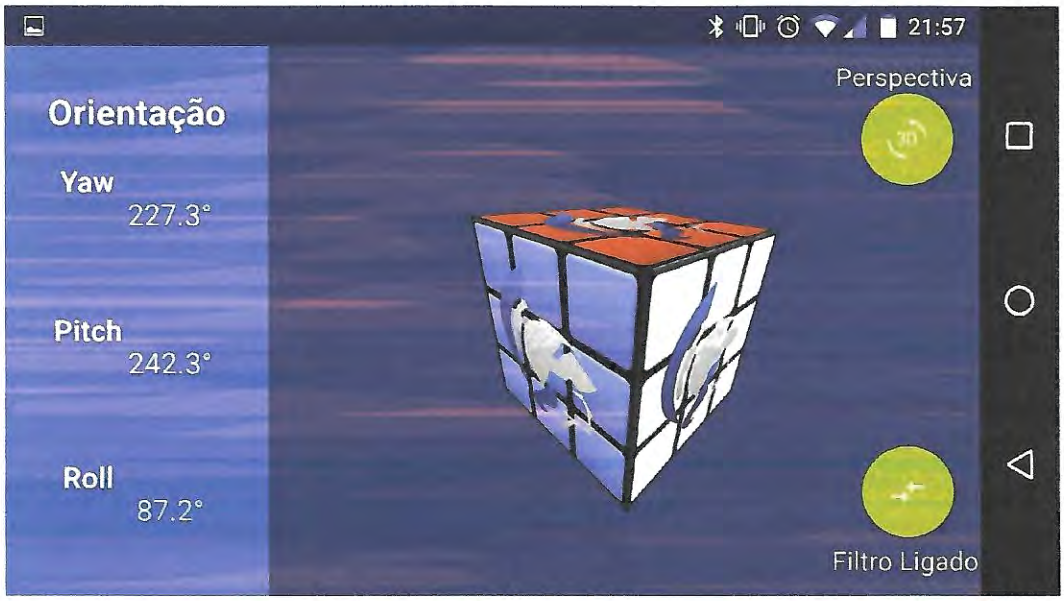

Figura 37 – Tela principal de orientação do aplicativo Teseu.

A esquerda são apresentados as orientações globais do sistema em Yaw, Pitch and Roll, representados na forma de ângulos de Euler, pois são mais facilmente compreendidos pelos usuários, mas essa exibição é independente da opção de formato de informação selecionada para envio ao sistema robótico.

No centro tem-se a representação de um cubo, o qual gira conforme o disposito móvel gira. Trata-se de uma forma gráfica de apresentar a orientação do sistema, mostrando como se houvesse um cubo estacionário no espaço e o dispositivo Android gira ao redor do mesmo.

A direita tem-se dois botões. O primeiro, "Perspectiva", muda a visualização gráfica de fora para dentro do cubo, como apresentado na Figura 38. Essa opção é lúdica e não impacta no desempenho do aplicativo, já que o único processamento extra necessário para realizar a operação é feito pela placa gráfica do mesmo.

Figura 38 – Apresentação gráfica da orientação do sistema.

O segundo botão a direita, "Filtro", liga e desliga a fusão sensorial desenvolvida nesse trabalho. Essa opção impacta significativamente no desempenho do aplicativo, portanto não deve ser desligada durante uma operação real. Só está disponível para que o usuário tenha a percepção do quanto a fusão sensorial é importante para uma boa determinação da orientação do sistema.

3.4.2 Localização

Caso seja de interesse do usuário, o aplicativo pode enviar para o robô móvel os dados relativos a localização do sistema baseado em marcadores específicos, apresentado no Capítulo 4. Para tanto deve-se usar o botão "Localização", o qual disponibiliza a localização por meio de marcadores, "ARToolkit", ou por cores, "OpenCV, como apresentado na Figura 39.

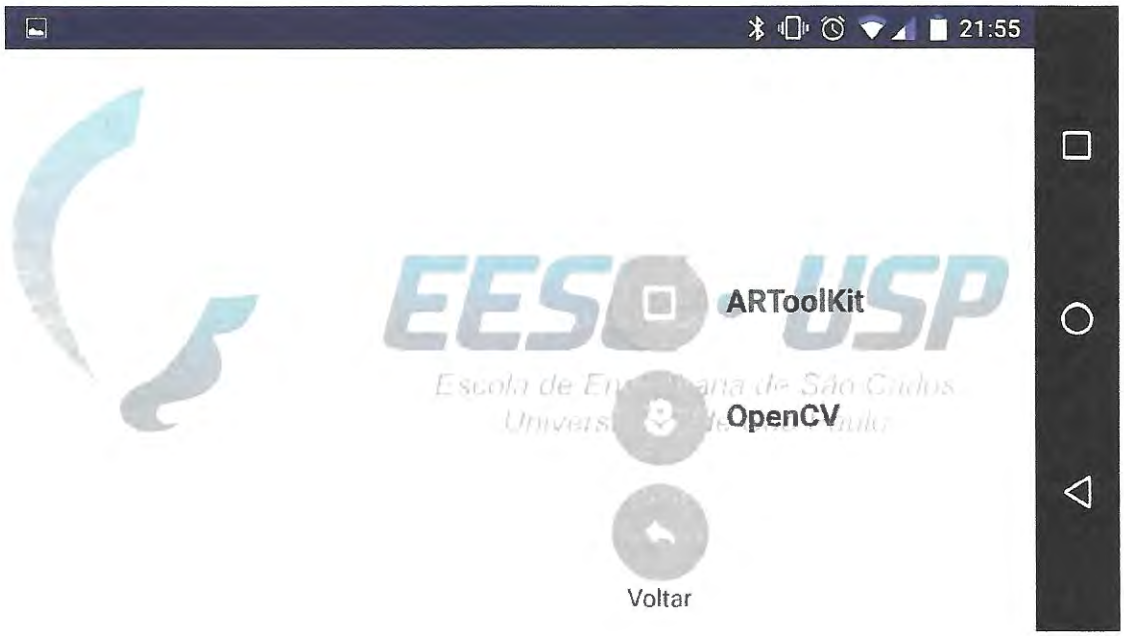

Figura 39 – Seleção dos modos de localização do aplicativo Teseu.

ARToolKit

Nesse modo de localização o aplicativo detecta macadores predefinidos por meio da biblioteca ARToolKit. Uma vez selecionado, o aplicativo levará o usuário para uma tela de apresentação das configurações da câmera atualmente selecionadas para a biblioteca ARToolKit, como apresentado na Figura 40.

Figura 40 – Configurações atuais do módulo ARToolKit.

Caso esteja de acordo com os interesses do usuário, ele deve apertar o botão à direita, "continuar". Caso contrário é possível alterar as configurações apertando o botão à esquerda, "configurar". O que levará ao usuário a tela apresentada na Figura 41.

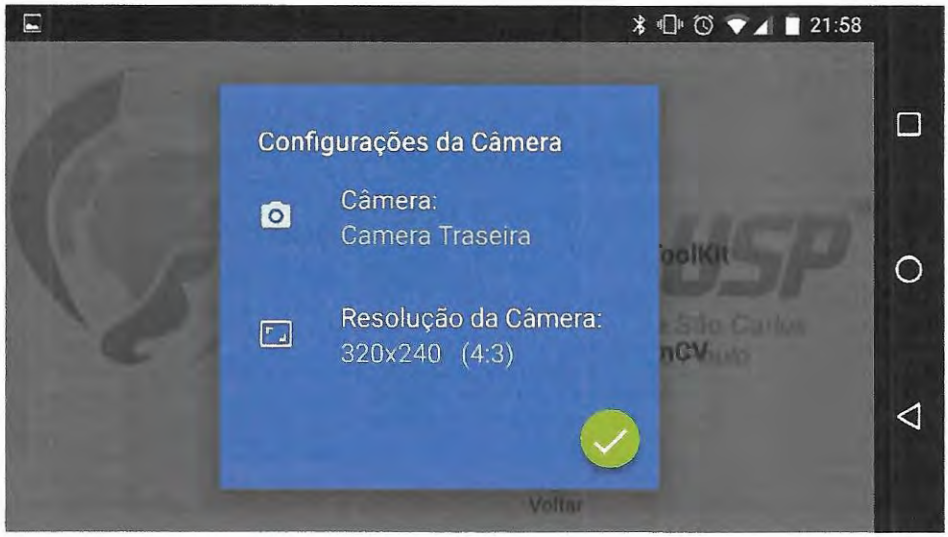

Figura 41 – Alterar configurações do módulo ARToolKit.

Tocando na configuração da câmera, uma nova tela será aberta, permitindo que o usuário selecione uma dentre as câmeras disponíveis pelo dispositivo Android, como apresentado na Figura 42.

Tocando na configuração da resolução, uma nova tela será aberta, permitindo que o usuário selecione uma dentre as resoluções disponíveis para a câmera selecionada, como apresentado na Figura 43.

Tocando no botão "continuar", será confirmada as configurações selecionadas.

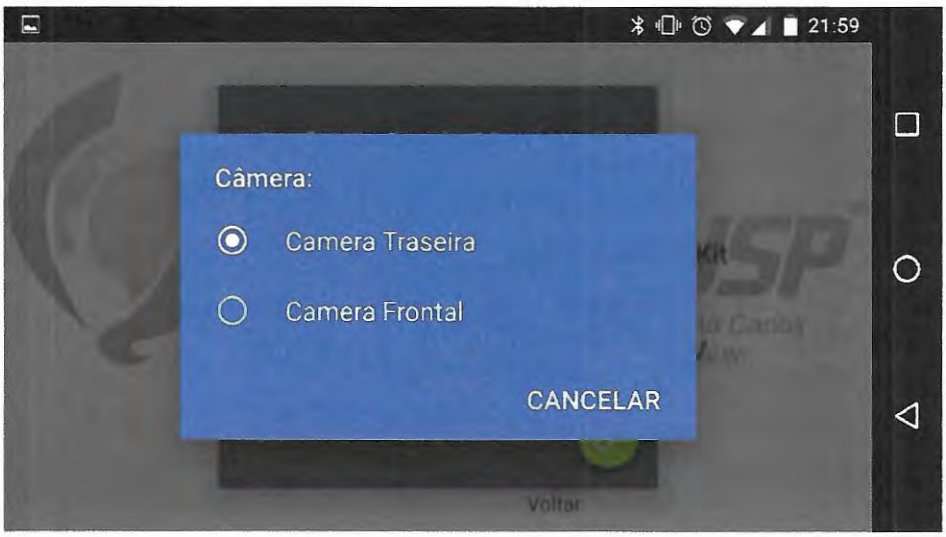

Figura 42 – Seleção da câmera para o módulo ARToolKit.

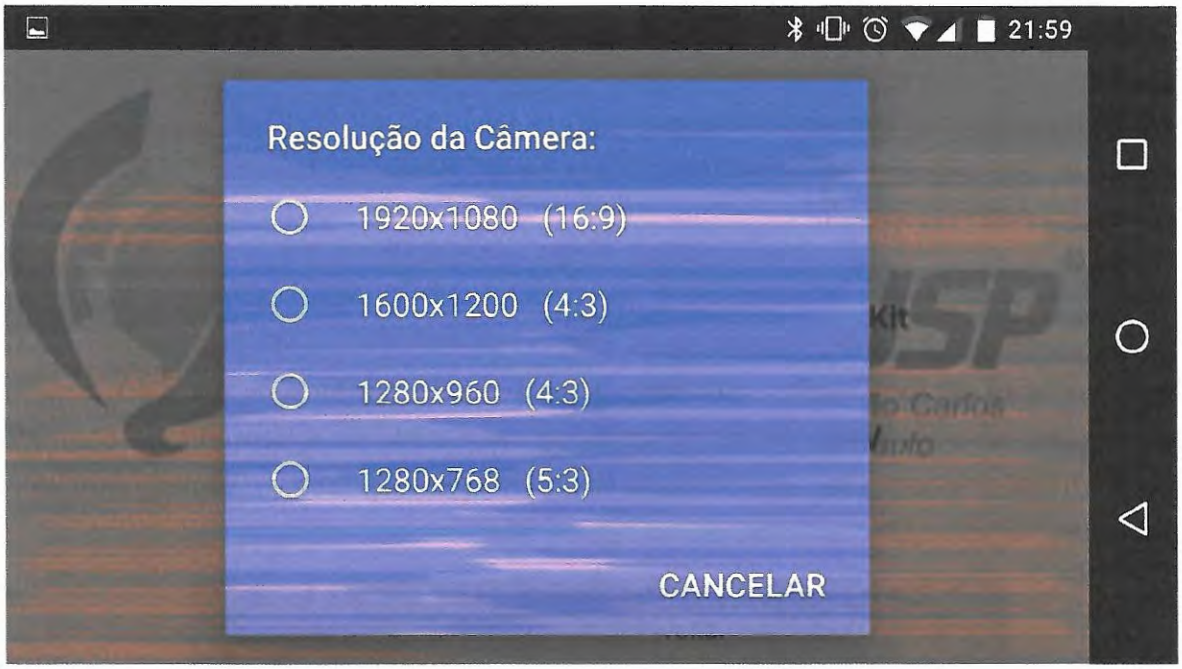

Figura 43 – Seleção da resolução da câmera para o módulo ARToolKit.

Uma vez iniciada a execução do aplicativo com a seleção de localização por ARToolKit habilitada, o usuário passará para a tela de localização ARToolKit, apresentada na Figura 44 e o envio de pacote de dados para o robô móvel será continua até que o usuário saia da tela de localização ARToolKit.

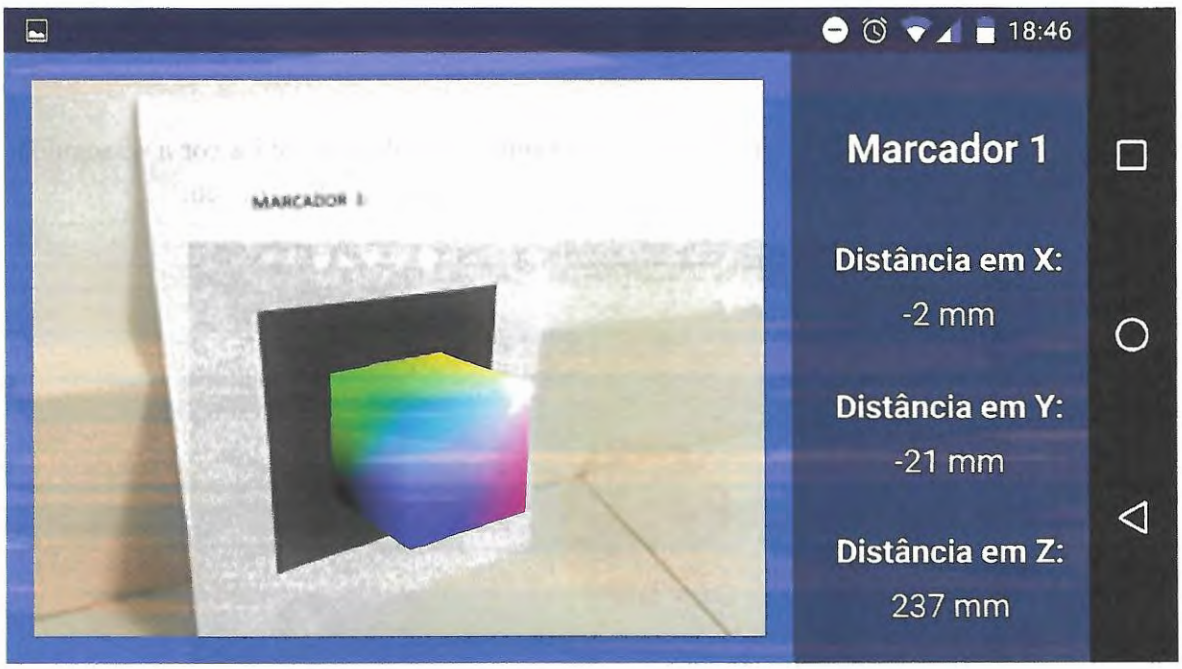

Figura 44 – Tela principal de localização via ARToolKit do aplicativo Tesen.

A direita são apresentados as distâncias em X, Y e Z entre o marcador e o dispositivo Android.

A esquerda é apresentado a imagem captura pela câmera. Quando um marcador é detectado um cubo é projetado sobre o marcador de forma a apresentar graficamente a orientação entre o marcador e o dispositivo.

OpenCV

Nesse modo de localização o aplicativo detecta objetos por meio de sua cor, portanto é necessário uma calibração da cor desejada em cada início da aplicação para diminuir ruídos devido a iluminação. Uma vez selecionado, o aplicativo levará o usuário para uma tela de calibração, como apresentado na Figura 45.

Figura 45 – Calibração do módulo OpenCV.

Nesta tela é esperado que o usuário toque na tela no local onde está a cor a serseguida, o que habilitará as funções de ajuste fino, como apresentado na Figura 46.

Figura 46 – Ajuste da calibração do módulo OpenCV.

Nesta, o usuário pode usar a barra na parte inferior para ampliar ou restringir a variação de cores aceitáveis para o objeto a ser detectado. O usuário também pode alterar a resolução da câmera e por fim deve tocar no botão "retornar"para voltar a tela principal do aplicativo.

Após finalizar a calibração da cor o usuário pode iniciar a execução do aplicativo.

Uma vez iniciada a execução do aplicativo com a seleção de localização por ARToolKit habilitada, o usuário passará para a tela de localização OpenCV, apresentada na Figura 47 e o envio de pacote de dados para o robô móvel será continua até que o usuário saia da tela de localização ARToolKit.

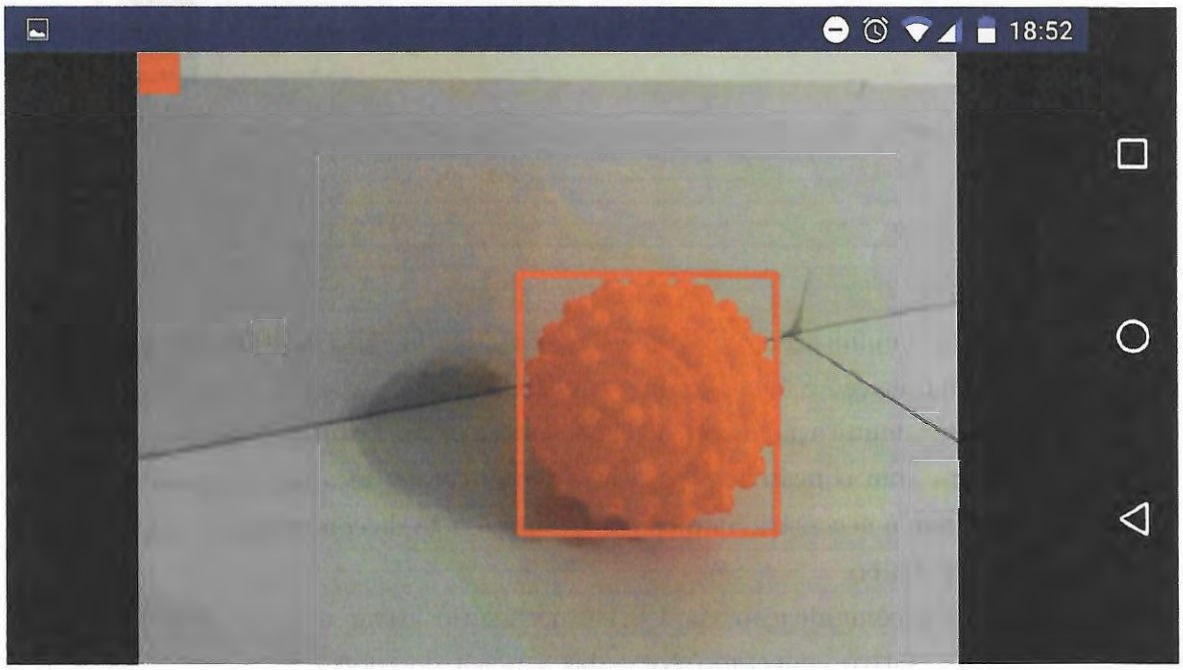

Figura 47 – Tela principal de localização via OpenCV do aplicativo Teseu.

Nesta tela é exibida a imagem capturada pela câmera e o objeto contendo a cor selecionada será cercado por um retângulo vermelho, como apresentado na Figura 47.

3.4.3 Comunicação

Dentre os modos de comunicação disponíveis para um dispositivo com sistema operacional Android, escolheu-se os principais para realizar a comunicação entre o dispositivo Android e o robô móvel. Assim permitindo uma ampla utilização do aplicativo desenvolvido nas mais diversas aplicações.

Tocando no botão "Comunicação"o usuário tem acesso às 3 opções de comunicação disponíveis: USB, Bluetooth e WiFi; além de também ter acesso á um modo que salva na memória interna os dados coletados pelo dispositivo.

Figura 48 – Menu de modos de comunicações disponíveis no aplicativo Teseu.

USB

Nesse modo de comunicação a informação é enviada via cabo diretamente para o robô móvel, logo a comunicação é bem rápida e confiável.

Para fazer essa comunicação direta utilizou-se uma placa Arduino ADK no robô móvel. A mesma já conta com o protocolo para esse tipo de comunicação. É possível utilizar outras placas, porém é necessário implementar o protocolo de comunicação, o que não era o objetivo deste trabalho.

Para habilitar a comunicação via USB é necessário ativar a opção "USB" no aplicativo, com o dispositivo conectado por cabo à placa do robô. Se a comunicação foir bem sucedida, o ícone "USB"se tornará azul, caso contrário permanecerá verde até que a comunicação seja feita, como apresentado na Figura 49.

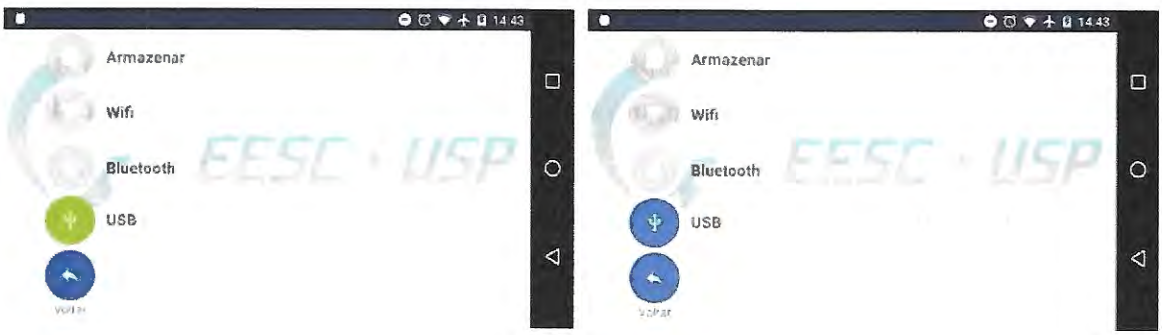

Figura 49 – Estado da comunicação USB.

Um exemplo do código do robô móvel está disponível em <https://drive.google.com/file/d/1uRNTcs4tLFM0WSAXAnFl5Y3_G27oc_SM/view?usp=sharing>.

Bluetooth

Nesse modo de comunicação a informação é enviada via protocolo Bluetooth para o robô móvel, logo a comunicação é bastante confiável, porém pode sofrer atrasos ou ficar mais lenta dependendo da distância entre os módulos bluetooth

Para habilitar a comunicação via Bluetooth é necessário ativar a função bluetooth no dispositivo Android e ativar a opção "Bluetooth" no aplicativo, e em seguida fazer a seleção do outro módulo bluetooth que participará da comunicação, como apresentado na Figura 50.

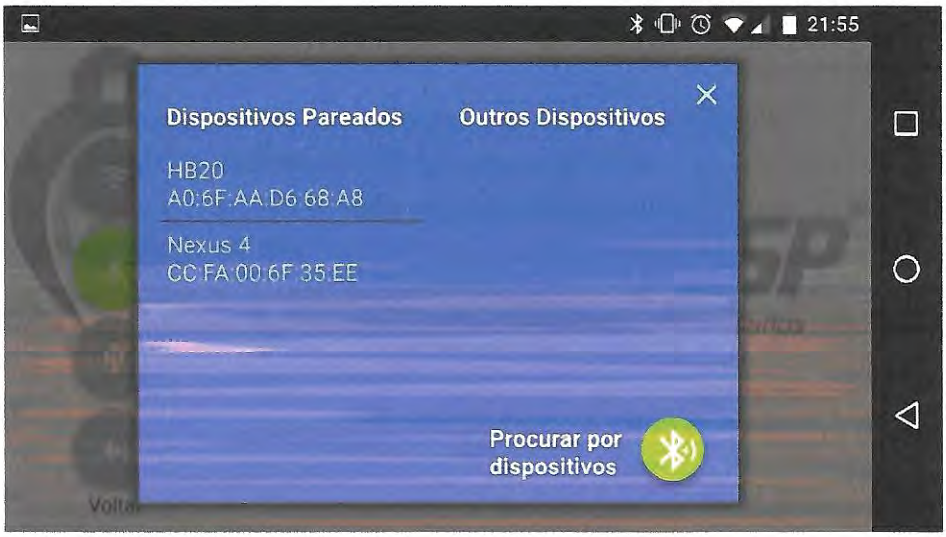

Figura 50 – Seleção do módulo Bluetooth que recebrá os dados coletados pelo dispositivo Android.

Muitos módulos permitem automaticamente a comunicação, nesse caso o símbolo da comunicação bluetooth se tornará azul.Caso o módulo bluetooth necessite autorizar a comunicação, o símbolo da comunicação bluetooth permanecerá verde até a autorização ser efetivada, como apresentado na Figura 51.

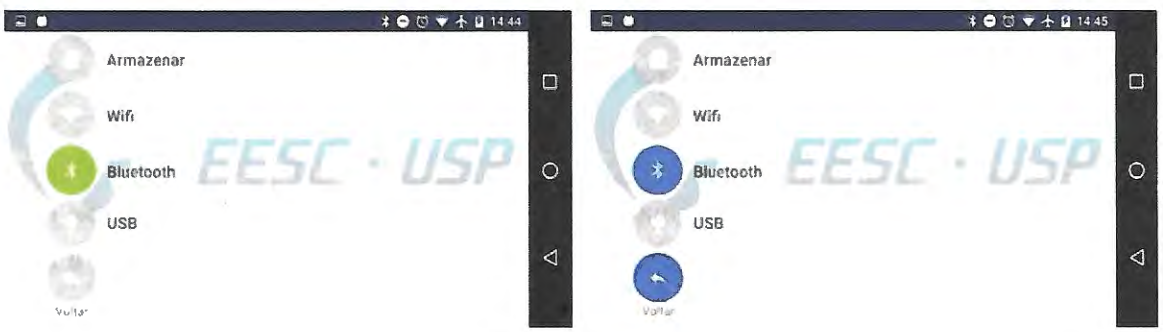

Figura 51 – Estado da comunicação Bluetooth.

Para fins de verificação da comunicação bluetooth é recomendado testar a comunicação em outro dispositivo Android, utilizando o aplicativo Bluetooth Terminal, disponível em <https://play.google.com/store/apps/details?id=ptah.apps.bluetoothterminal>.

WiFi

Nesse modo de comunicação a informação é enviada utilizando uma rede Wi Fi disponível através de um socket direto entre o dispositivo Android e o robô móvel. Logo, a comunicação é rápida, possível de ser feita com maiores distâncias entre os módulos emissores-receptores, porém alguns pacotes podem se perder no caminho, e estes não serão novamente enviados.

Para habilitar a comunicação via Wi-Fi é necessário conectar-se a rede Wi-Fi na qual está conectado o robô móvel e ativar a opção "WiFi" no aplicativo. Em seguida deve-se indicar o IP do robô móvel, como apresentado na Figura 52.

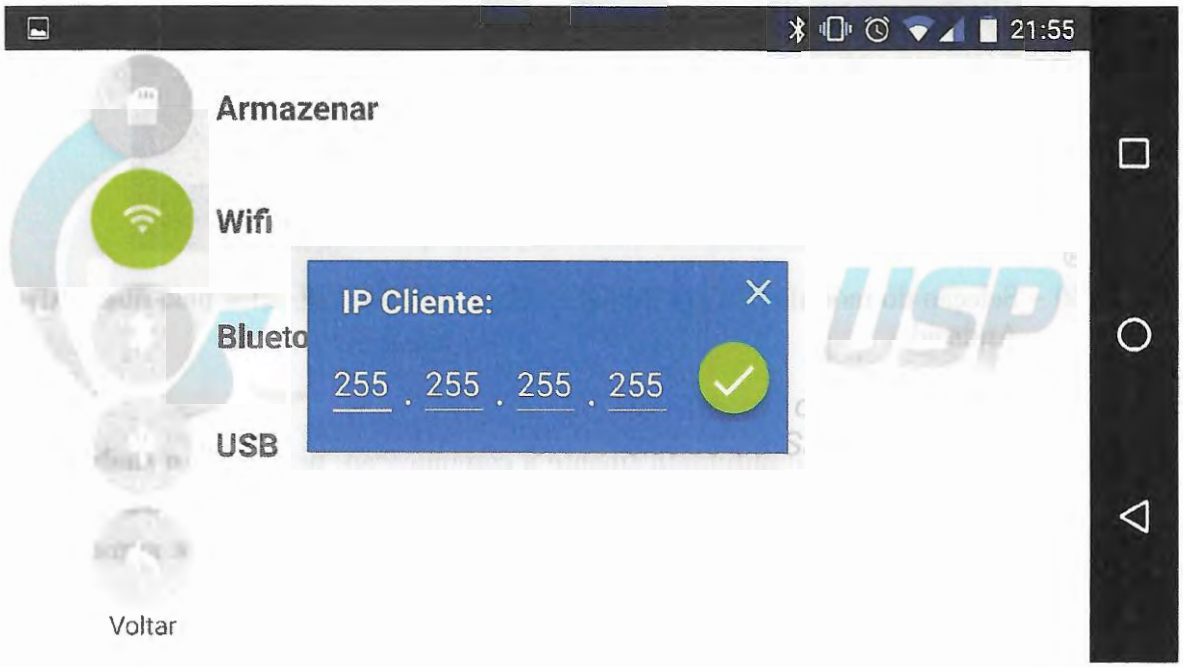

Figura 52 – Seleção do IP do robô móvel para a comunicação via WiFi.

Também é possível enviar as informações para mais de um módulo Wi-Fi, nesse caso deve-se ser indicado o IP 255.255.255.255. Assim, todos os dispositivos conectados à rede Wi-Fi poderão receber as informações enviadas pelo dispositivo Android.

A fim de verificar a comunicação Wi-Fi, um código para o sistema operacional Windows está disponível em <https://drive.google.com/file/d/12JzObGKZ4hcPKFq6R0M0Y8KPsnl7yZvv/view?usp=sharing>. Para executá-lo:

- Descompacte o arquivo zip em uma pasta conhecida
- Abrir o programa Executar do Windows
- Digitar: cmd
- Digitar: cd "caminho para a pasta usada na descompactação" / Server / bin
- Digitar: java server

O código em questão salva dois arquivos na pasta "caminho para a pasta usada na descompactação" / Server / bin. O primeiro, "arquivo0.txt", contém todos os dados enviados pelo dispositivo Android. O segundo, "arquivo1.txt", contém o último pacote de mensagens recebido. Assim facilmente outro programa pode estar continuamente acessando esse arquivo e coletando a última informação disponível.

O código encerrerá automaticamente sua execução se ficar 5 segundos sem receber nenhum novo pacote de dados.

Armazenamento Interno

Além dos três modos de comunicação apresentados anteriormente, um quarto modo foi implementado, visando aplicações que envolvem a coleta de dados mas precisam tomar decisões em tempo real, sendo pós-processadas.

Para tanto, esse modo deve ser habilitado por meio da opção "Armazenar" apresentada juntamente aos demais modos de comunicação, como apresentado na Figura 53.

Figura 53 – Armazenamento dos dados coletados na memória interna habilitado.

Os dados serão salvos em um arquivo em formato texto na memória interna do dispositivo móvel no endereço: Memória Interna / Android / Data / edu.eesc.usp.guiduenhas / files / Data da Coleta / Hora da Coleta .txt.

Ajustes da comunicação

Uma vez selecionada todas as configurações de comunicação e de modos de extração de dados, o usuário ao iniciar a execução do programa será enviado à uma tela de ajustes de comunicação, como apresentado na Figura 54.

Figura 54 – Tela de ajuste de comunicação do aplicativo Teseu.

Nesta tela o usuário pode ajustar:

- **Taxa de Amostragem:** Frequência de envio de dados entre o dispositivo Android e o robô móvel, de 1Hz a 100Hz.
- **Representação da Orientação:** A orientação pode ser enviada na forma de quatérnion ou ângulos de Euler.

  Se escolhido "Quaternion", o aplicativo enviará automaticamente os 4 valores da representação.

  Se escolhido "Angulos de Euler", o aplicativo enviará os ângulos selecionados na linha abaixo, Yaw, Pitch e Roll.

- **Dados de Localização:** Dentre os dados de localização extraídos pelo aplicativo, o usuário pode selecionar entre as distâncias X, Y e Z e os ângulos Yaw, Pitch e Roll.

Protocolo de comunicação

Para padronizar a comunicação entre o dispositivo móvel e o robô móvel, a comunicação é estabelecida por meio de pacote de dados que seguem o protocolo:

- Só são enviados os dados selecionados pelo usuário na tela de ajuste de comunicação;
- Todos os dados são separados por um caracter "espaço";
- Ao final de cada pacote de dados é enviado a sequência de caracteres: "@\n";
- O pacote de dados é construído na ordem: Dados de Orientação > Dados de Localização ARToolKit > Dados de Localização OpenCV;
- Os dados de orientação seguem a ordem: W > X > Y > Z, quando a representação por quatérnions é selecionada;
- Os valores do quaternion dos dados de orientação, se positivos, são compostos por: 2 caracteres > caracter ".": 4 caracteres;
- Os valores do quaternion dos dados de orientação, se negativos, são compostos por: caracter -» 1 caracter > caracter ".": 4 caracteres;
- Os dados de orientação seguem a ordem: Yaw > Pitch > Roll, quando a representação por angulos de Euler é selecionada;
- Os angulos de Euler dos dados de orientação são compostos por: 3 caracteres > caracter ".» 1 caracter;
- Os dados de localização do ARToolKit seguem a ordem: X > Y > Z > Yaw > Pitch > Roll;
- Os dados de distância de localização do ARToolKit, se positivos, são compostos por: 4 caracteres;
- Os dados de distância de localização do ARToolKit, se negativos, são compostos por: caracter -» 3 caracteres;
- Os dados de orientação de localização do ARToolKit são compostos por: 3 caracteres > caracter ".» 1 caracter;
- Os dados de localização do OpenCV seguem a ordem: X > Y > Largura > Altura;

- Os dados de localização do OpenCV, se positivos, são compostos por: 2 caracteres > caracter ".» 4 caracteres;
- Os dados de localização do OpenCV, se negativos, são compostos por: caracter -» 1 caracter > caracter ".» 4 caracteres;

Capítulo 4

Resultados Experimentais

Este capítulo apresenta os resultados obtidos pelos diversos métodos apresentados nos capítulos anteriores, comparando as suas eficiências.

Afim de validar as diferentes entregas de cada método, foram propostos 4 experimentos. O primeiro focado nos sensores inerciais, o segundo dividido em duas partes focado na detecção de marcadores pela biblioteca ARToolkit e OpenCV e o terceiro focado no auxílio a navegação autômoma de um sistema robótico.

4.1 Sensores Inerciais

Este primeiro teste visava avaliar e comparar os diferentes métodos de determinação de orientação. Para tanto, o sistema robótico foi controlado externamente por meio de um controle implementado utilizando os sensores de movimentação das rodas. E durante esse processo, a plataforma Android realizava a detecção da orientação através dos sensores: "Norte Magnético", "Filtro Complementar", "Orientação Absoluta", "Giroscópio Calibrado"e "Fusão de Sensores Virtuais".

A fim de se ter uma avaliação completa o sistema robótico foi programado de forma a realizar uma série de movimentos em sequência, movimentos retilíneos frontais, movimentos retilíneos laterais e movimentos rotacionais, sempre intercalados por momentos estacionários.

E visando avaliar a robustez e confiabilidade dos métodos propostos, o teste foi repetido 16 vezes, considerando alguns fatores de ruído como: Diferentes dispositivos Android, Diferentes velocidade de movimentação do sistema robótico, diferentes tempos de pausa e diferentes orientações iniciais. Como na árvore de amostragem apresentada na Figura 55, a seguir:

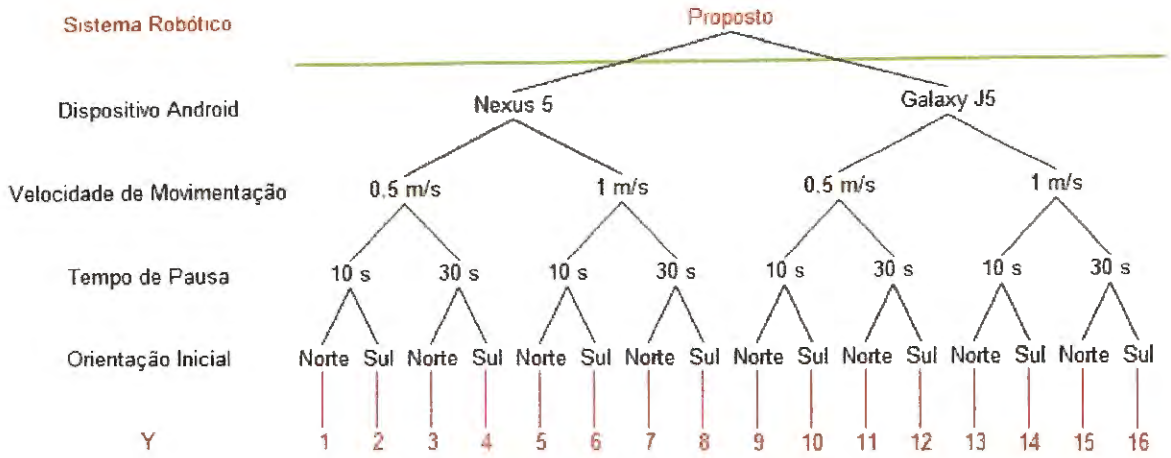

Figura 55 – Estratégia de amostragem para o primeiro experimento.

Na Figura 56 representa-se o gráfico de orientação teórico executado pelo sistema robótico neste experimento.

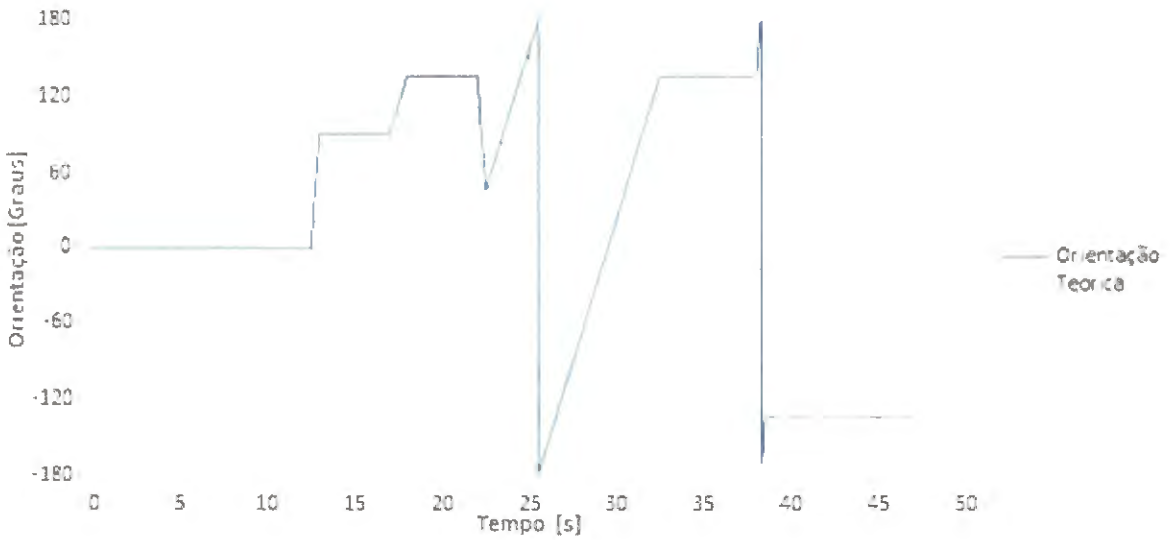

Figura 56 – Orientação teórica prevista para o primeiro experimento.

Neste podemos ver que o sistema robótico iniciava o experimento direcionado para a orientação 0°. Após percorrer alguns metros, medidos por meio dos encoders nas rodas, realizou uma rotação de 90° para a direita no seu próprio eixo. Depois seguiu em um movimento retilíneo lateralmente para a esquerda alguns metros e então começou um movimento curvilíneo também para a direita, até atingir a orientação de 135°, na qual permaneceu parado por alguns segundos. Então iniciou um movimento curvilíneo para a esquerda até atingir 45°, momento no qual realizou uma rotação lenta no próprio eixo, no sentido horário, até completar uma volta completa e se alinhar em 120°. Por fim, se

movimentou linearmente para trás por alguns metros e girou rapidamente 90° também no sentido horário.

Com fins comparativos, os dados coletados das 16 rodadas foram ajustados de forma a começar também na orientação 0°, e as médias de cada método é expressa nos próximos gráficos.

Sensor "Norte"

Na Figura 57 apresenta-se o gráfico com o comparativo entre a orientação teórica proposta pelo experimento e a resposta obtida pela utilização do sensor "Norte", apresentado na Seção 2.2.

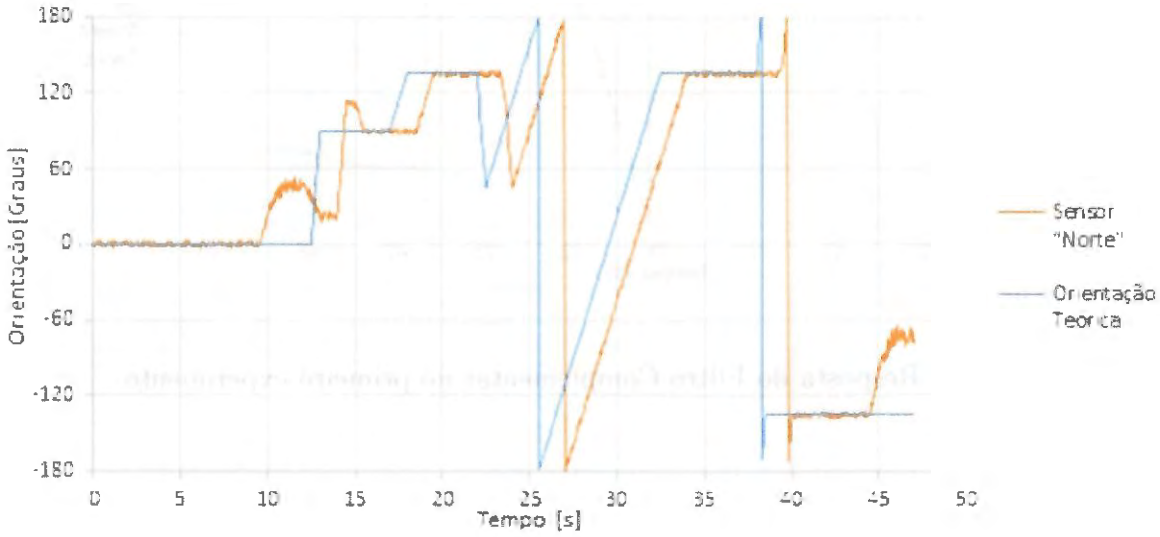

Figura 57 – Resposta do Sensor "Norte" no primeiro experimento.

É evidente que a utilização somente deste sensor não é uma solução robusta. Primeiramente porque a resposta do mesmo é defasada em relação a orientação do sistema robótico. Isso se deve à velocidade da resposta proveniente da bússola magnética, a qual não é muito alta.

Também é possível ver dois momentos, de 9 a 16 segundos e de 44 a 47 segundos, nos quais a resposta do sensor diverge significativamente da orientação do sistema robótico. Isso provavelmente se deve a materiais ferromagnéticos nas proximidades do mesmo. Lembrando que o concreto armado utilizado nas construções dos prédios é um forte contribuidor para o campo magnético no ambiente.

Por fim, o desvio padrão entre a resposta do sensor e a orientação teórica proposta pelo experimento foi de 2,08° já se desconsiderando os dois momentos onde os valores divergiram e a defasagem entre as respostas.

Filtro Complementar

Na Figura 58 apresenta-se o gráfico com o comparativo entre a orientação teórica proposta pelo experimento e a resposta obtida pela utilização do sensor "Filtro Complementar", apresentado no Capítulo 3.

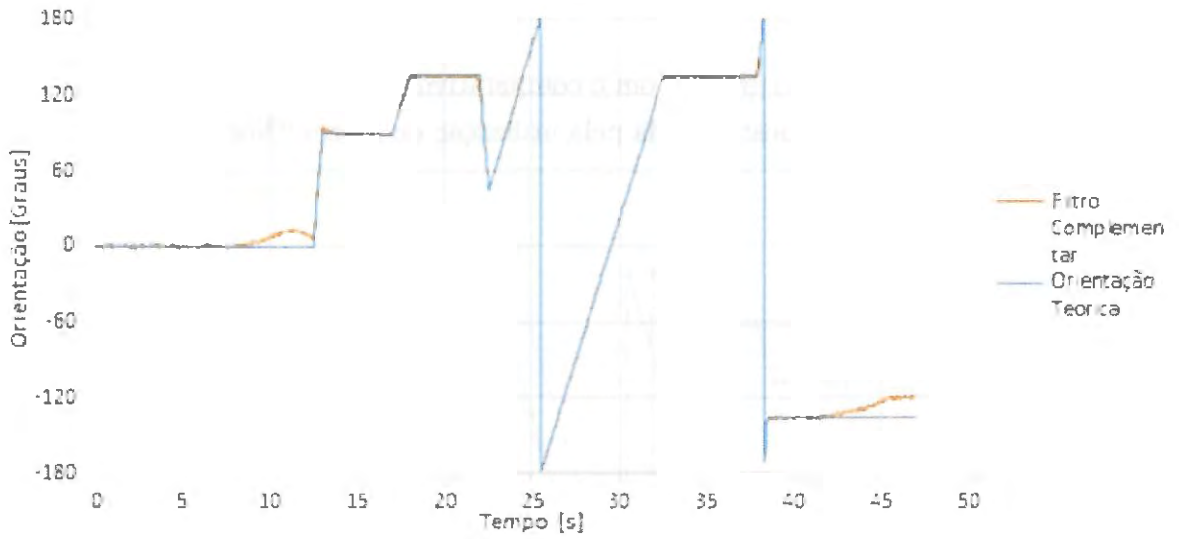

Figura 58 – Resposta do Filtro Complementar no primeiro experimento.

Pode-se notar claramente que a proposta de fusão sensorial por meio do filtro complementar é mais robusta que a utilização do sensor "Norte".

Entretanto ainda é evidente a presença de dois momentos de divergência dos valores. Isso se deve ao fato do sensor confiar na resposta da Bússola Magnética quando a resposta dela é constante, mesmo que seja oriunda de um ruído ambiental.

O desvio padrão entre a resposta do filtro complementar proposto e a orientação teórica proposta pelo experimento foi de 1,63° já se desconsiderando os dois momentos onde os valores divergiram.

Sensor "Orientação Absoluta"

Na Figura 59 apresenta-se o gráfico com o comparativo entre a orientação teórica proposta pelo experimento e a resposta obtida pela utilização do sensor "Orientação Absoluta", apresentado no Capítulo 3.

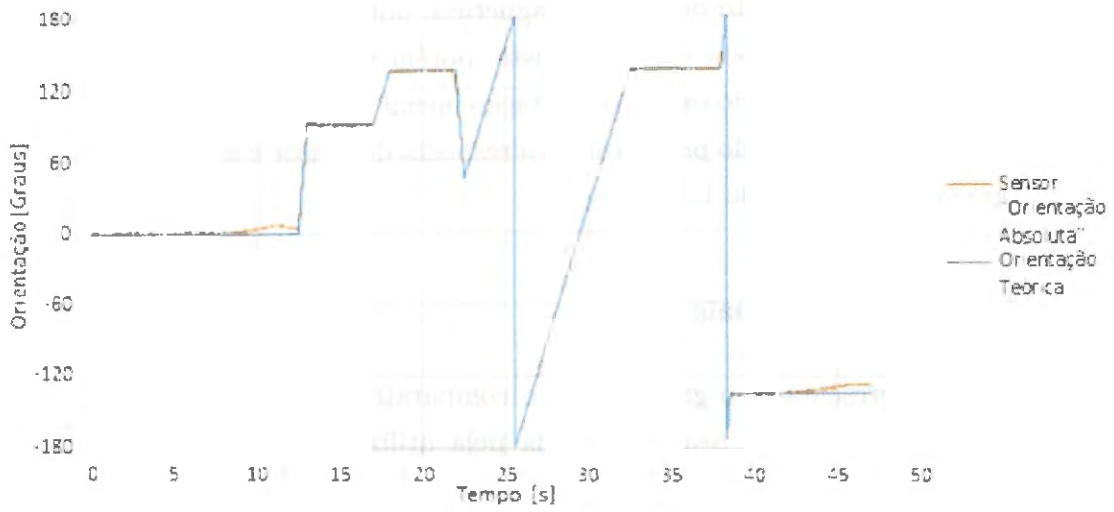

Figura 59 – Resposta do Sensor "Orientação Absoluta" no primeiro experimento.

Mesmo com essa fusão sensorial mais elaborada ainda é visível a presença de dois momentos de divergência dos valores. Isso se deve ao fato do sensor confiar na resposta da Bússola Magnética quando a resposta no Giroscópio é nula, ou seja, quando o robô está praticamente parado.

O desvio padrão entre a resposta do sensor e a orientação teórica proposta pelo experimento foi de 1,66", considerando até mesmo os dois momentos onde os valores divergiram.

Sensor "Giroscópio Calibrado"

Na Figura 60 apresenta-se o gráfico com o comparativo entre a orientação teórica proposta pelo experimento e a resposta obtida pela utilização do sensor "Giroscópio Calibrado", apresentado no Capítulo 3.

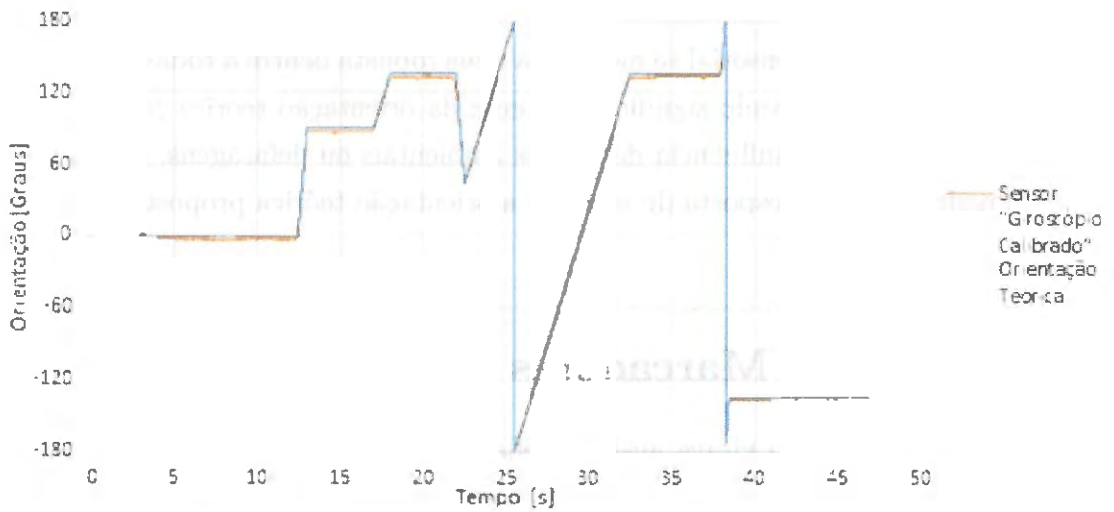

Figura 60 – Resposta do sensor "Giroscópio Calibrado" no primeiro experimento.

Esse sensor não sofre impacto de ruídos magnéticos pois só utiliza como sensor base o Giroscópio. Assim, sua resposta é mais robusta, porém vale-se lembrar de que é uma resposta relativa, sendo necessário saber a orientação inicial para que se saiba a orientação real do sistema robótico. O desvio padrão entre a resposta do sensor e a orientação teórica proposta pelo experimento foi de 1,47°.

Fusão dos Sensores Virtuais

Na Figura 61 apresenta-se o gráfico com o comparativo entre a orientação teórica proposta pelo experimento e a resposta obtida pela utilização da Fusão dos Sensores Virtuais, apresentada no Capítulo 3.

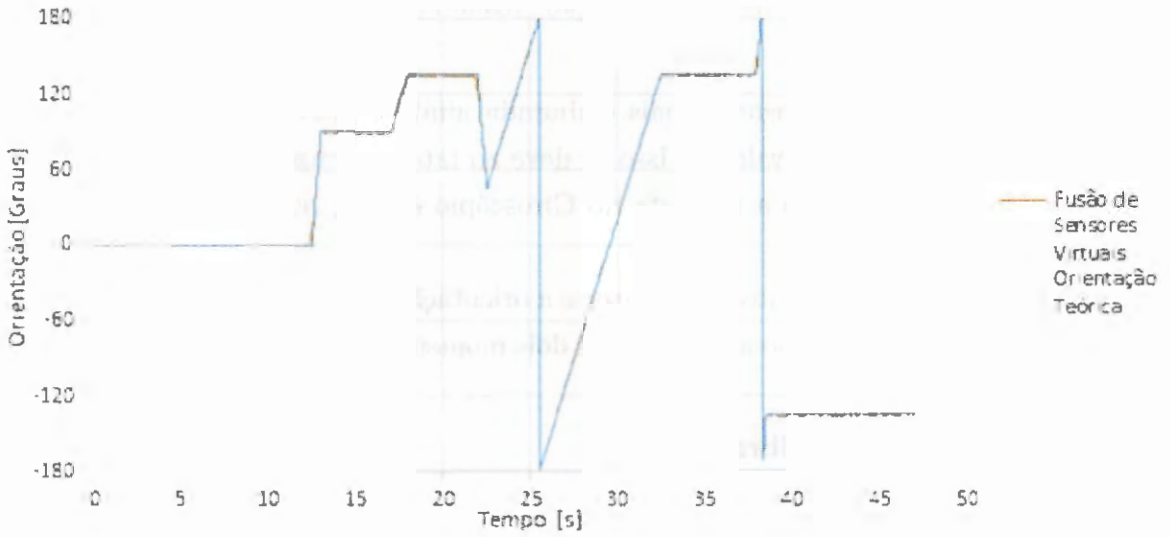

Figura 61 – Resposta da Fusão de Sensores Virtuais no primeiro experimento.

A utilização dessa fusão sensorial se mostrou a mais robusta dentre a todas as testadas, com as respostas se aproximando significativamente da orientação teórica proposta pelo experimento, e não sofrendo influência de fatores ambientais ou defasagens.

O desvio padrão entre a resposta do sensor e a orientação teórica proposta pelo experimento foi de 1,05°.

4.2 Detecção de Marcadores

Este segundo experimento visava avaliar a eficiência e robustez das metodologias de detecção e rastreamento de marcadores. Porém as bibliotecas foram avaliadas separadamente, uma vez que cada metodologia demanda um tipo de marcador específico, porém ambos visando a assertividade da detecção.

O sistema robótico foi programado a se movimentar linearmente para frente com base nas medidas dos sensores de movimentos nas rodas. A trajetória executada permitia que a câmera da plataforma Android foi exposta a 5 marcadores, como apresentado na Figura 62, cada um posicionado a uma distância específica do trajeto do sistema robótico, como apresentado na Tabela 1.

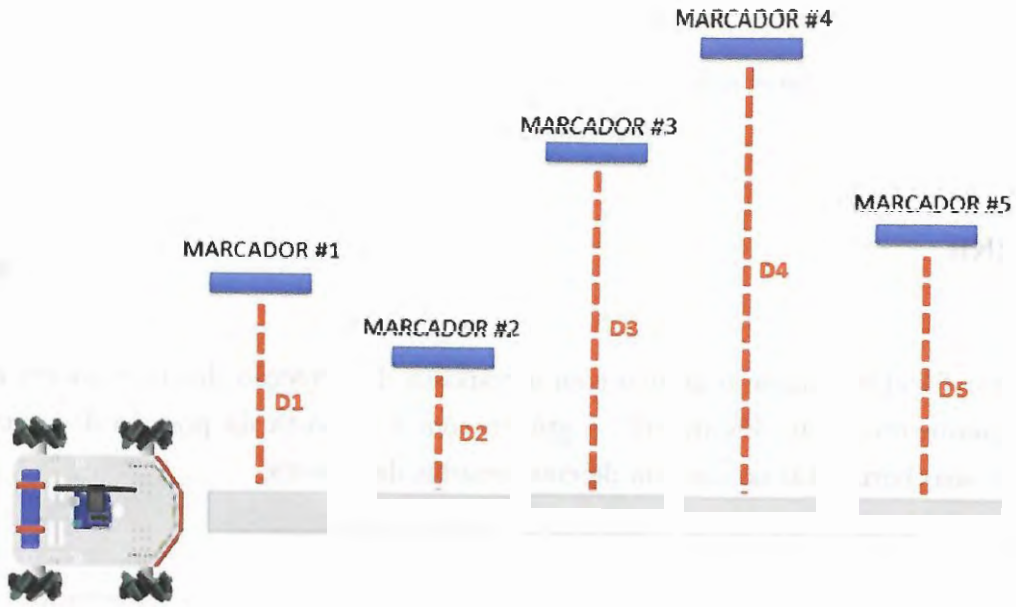

Figura 62 – Esquemático da estratégia aplicada no segundo experimento.

Tabela 1 – Distância dos marcadores

Marcador	Distância [mm]
#1	100
#2	75
#3	150
#4	225
#5	125

E visando avaliar a robustez e confiabilidade dos métodos propostos, o teste foi repetido 16 vezes, considerando alguns fatores de ruído como: Diferentes dispositivos Android, Diferentes velocidade de movimentação e Diferentes condições de iluminação, utilizando iluminação natural e artificial e utilizando o flash do dispositivo ou não. Como na árvore de amostragem apresentada na Figura 63, a seguir:

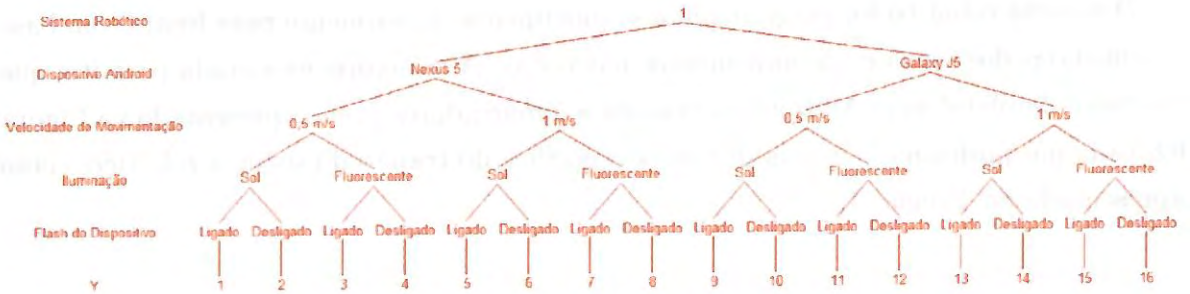

Figura 63 – Estratégia de amostragem para o segundo experimento.

ARToolKit

Na Figura 64 apresenta-se o gráfico com a resposta da detecção dos marcadores ao longo do experimento, e, na Figura 65, o gráfico com a resposta da posição do centro marcador no eixo horizontal no sistema de coordenadas da câmera.

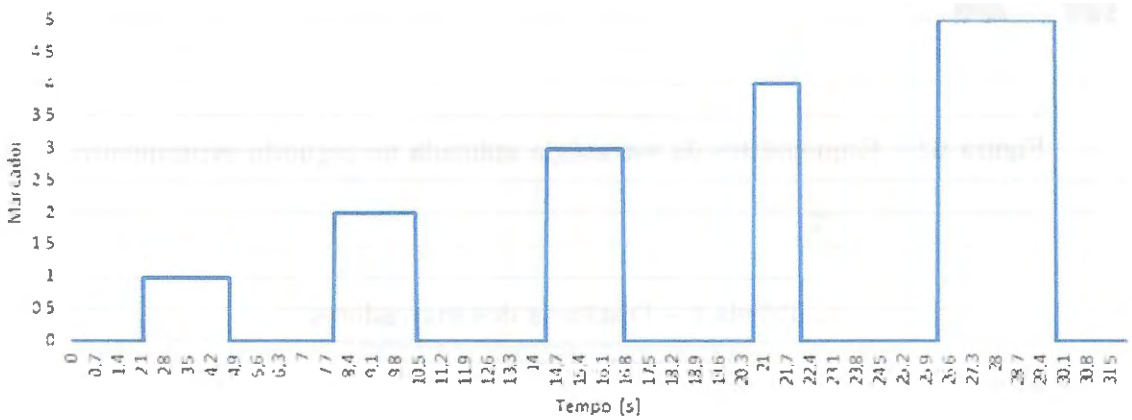

Figura 64 – Detecção dos marcadores com a ARToolKit no segundo experimento.

Neste, pode-se observar, que o ARToolKit foi capaz de reconhecer corretamente o experimento, pois com o passar do tempo o sistema robótico se moveu e os diferentes marcadores foram detectados pelo sistema Android.

Porém podemos notar que a detecção do marcador #4, o mais distante, demorou mais do que os demais para acontecer, uma vez que este marcador ficou menos tempo identificado do que os demais.

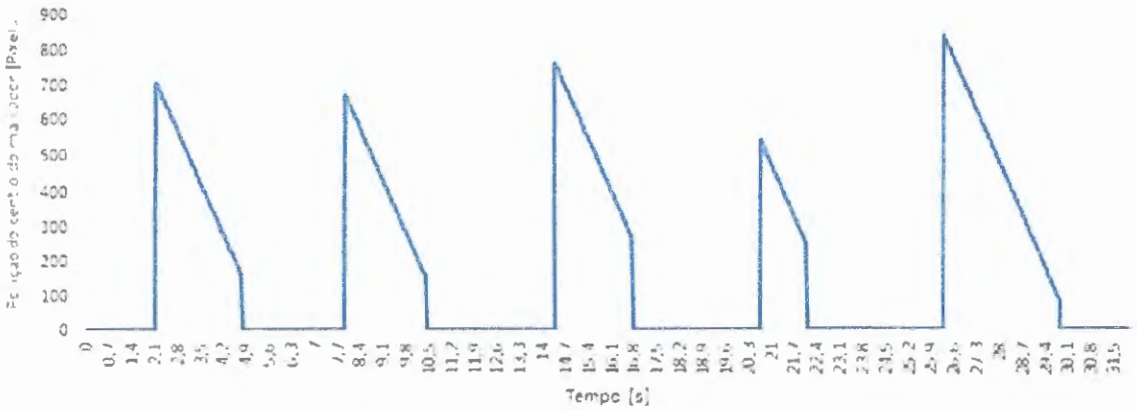

Figura 65 – Posição do centro do marcador no sistema de coordenadas da câmera com a ARToolKit.

Neste gráfico, pode-se ver a variação da posição do centro dos marcadores ao longo do tempo. Essa posição é medida em pixels e estruturada conforme o esquema apresentado na Figura 66.

Assim como pode ser visto na Figura 64, na Figura 65, também é possível perceber que o marcador #4, o mais distante, demorou mais tempo do que os demais para acontecer, uma vez que este marcador só foi identificado quando seu centro estava praticamente no centro da imagem capturada pela câmera.

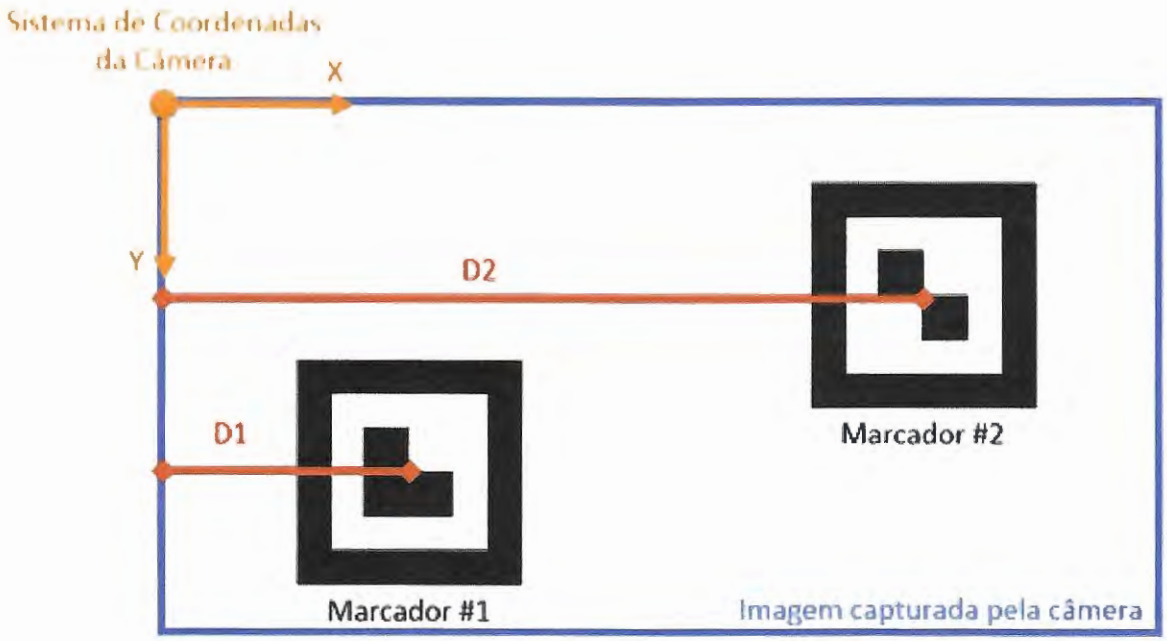

Figura 66 – Estrutura para medição da posição do marcador na imagem capturada pela câmera.

Devido às características matematicamente desenhadas dos marcadores, a ARToolkit também é capaz de detectar também a distância entre o sistema robótico e o marcador, gráfico apresentado pela Figura 67, a seguir.

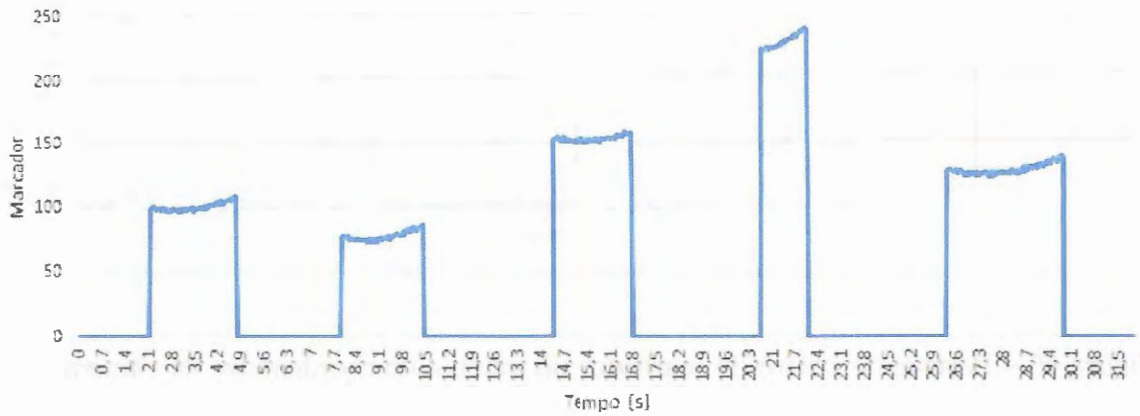

Figura 67 – Distância entre o sistema robótico e os marcadores no segundo experimento.

Pode-se observar que as respostas são bem próximas às distâncias reais, até mesmo sendo possível detectar a variação da distância conforme o sistema robótico de movimenta paralelamente ao marcador. Na Figura 68 exemplifica-se melhor esse efeito.

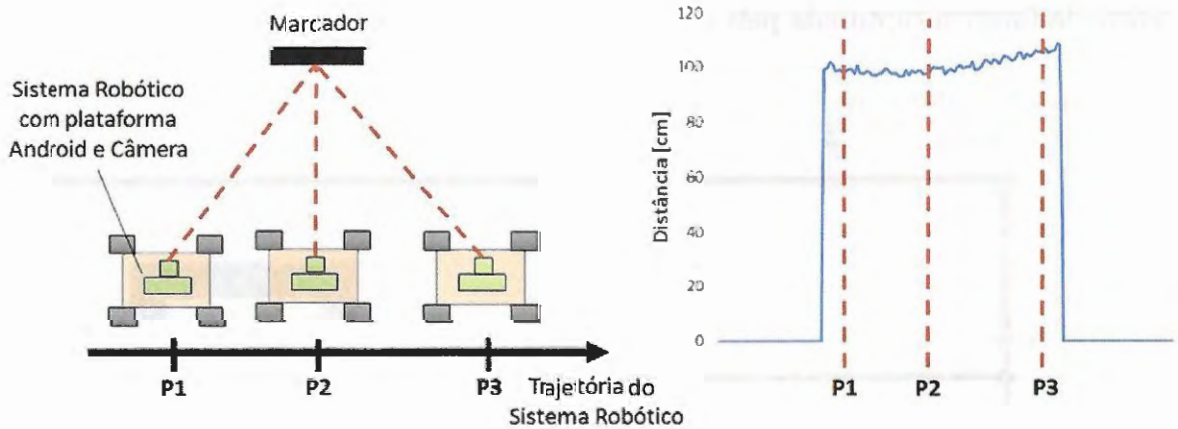

Figura 68 – Variação da distância durante o rastreamento de um marcador.

Em um primeiro momento, P1, o marcador se torna visível para o sistema robótico com a plataforma Android, e a distância entre ambos é medida pelo *software*. Em um segundo momento, P2, o marcador se torna perpendicular ao sistema de visão, e a distância entre ambos é a menor possível. Por fim, no último instante em que o marcador ainda é visível para o sistema robótico, P3, a distância é máxima. Portanto, a distância no instante é P2 é menor que a distância P1, que por sua vez é menor que a distância P3.

Teoricamente as distâncias medidas em P1 e P3 poderiam ter o mesmo valor, porém o *software* demanda algum tempo para realizar a primeira detecção do marcador, depois só necessita manter o rastreamento. Assim, normalmente teremos a distância P1 menor do que a distância P3.

OpenCV

Na Figura 69 apresenta-se o gráfico com a resposta da detecção dos marcadores ao longo do experimento, e na Figura 70, o gráfico com a resposta da posição do marcador no eixo horizontal da imagem.

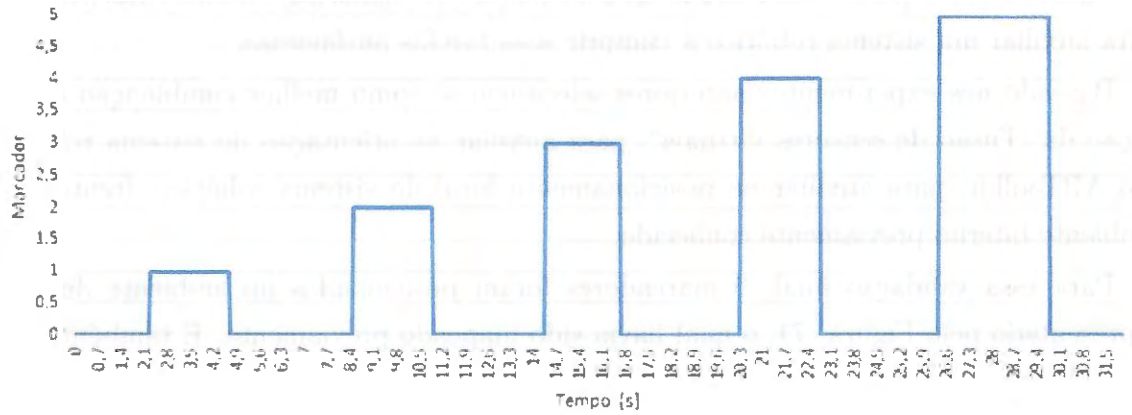

Figura 69 – Distância entre o sistema robótico e os marcadores no segundo experimento.

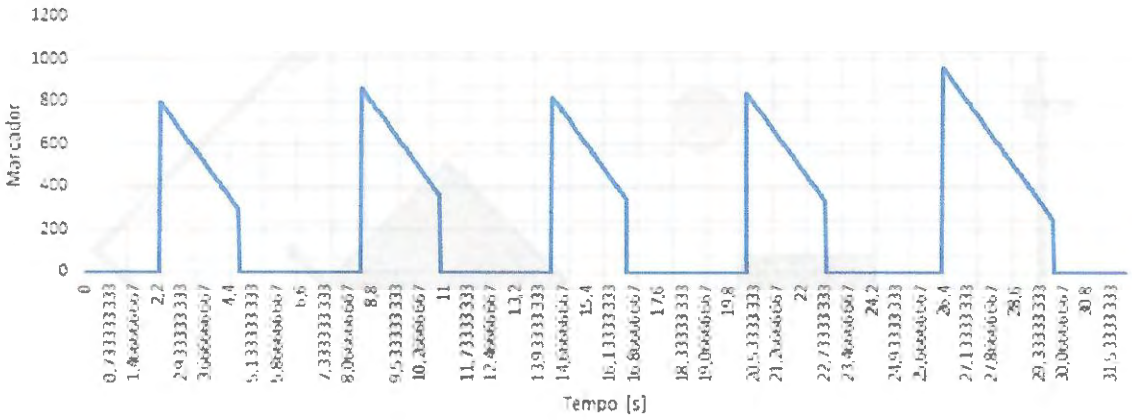

Figura 70 – Distância entre o sistema robótico e os marcadores no segundo experimento.

Em ambos os gráficos, apresentados nas Figuras 69 e 70, a OpenCV foi capaz de reconhecer e analisar os movimentos dos marcadores corretamente no experimento de uma

forma consistente independente da distância do marcador, fator que foi significativo para a ARToolKit.

Porém vale ressaltar que durante o experimento a biblioteca deixou de reconhecer alguns marcadores quando houve variação na iluminação ambiente. Assim reforçando a necessidade de performar uma rotina de calibração dos marcadores antes de iniciar a operação.

4.3 Navegação Autônoma

Este terceiro experimento visa avaliar a eficiência e robustez dos métodos desenvolvidos para auxiliar um sistema robótico a cumprir suas tarefas autônomas.

Baseado nos experimentos anteriores selecionou-se como melhor combinação a utilização da "Fusão de sensores virtuais", para auxiliar na orientação do sistema robótico, e o ARToolkit, para auxiliar no posicionamento local do sistema robótico, frente a um ambiente interno previamente conhecido.

Para essa validação final, 9 marcadores foram posicionados no ambiente de teste, representado pela Figura 71, o qual havia sido mapeado previamente. E também foram adicionados ao ambiente dois obstáculos não mapeados.

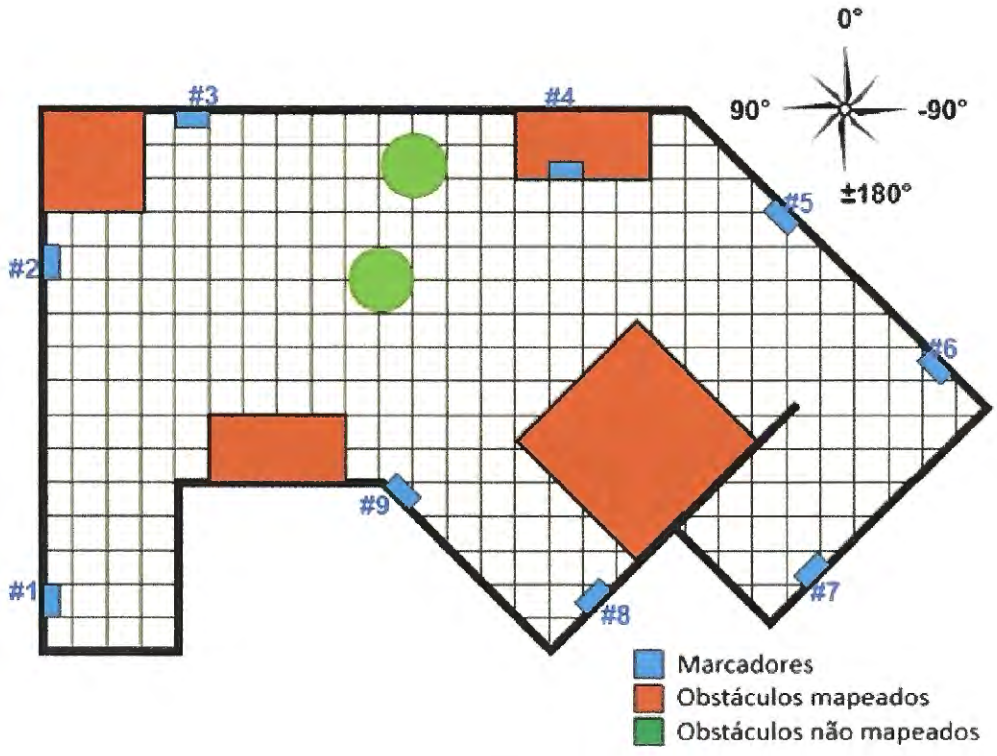

Figura 71 – Mapeamento do ambiente do terceiro experimento.

E uma lógica simples de navegação autônoma foi programada na plataforma Android, de forma a conduzir o robô entre dois marcadores contíguos.

Por exemplo, para o sistema robótico se movimentar do marcador #2 até o marcador #3, ele deve:

1) Com o dispositivo virado para a lateral esquerda do sistema robótico, alinhar à orientação de 90°;

2) Movimentar-se linearmente para a direita até que o sistema robótico esteja a 2 metros do marcador #2;

3) Rotacionar-se no próprio eixo, no sentido horário, até a orientação de 0°;

4) Movimentar-se linearmente para a frente até o marcador #3 estar centrado na câmera;

O experimento em questão requisitava que o sistema robótico se movimenta-se do marcador #1 até o marcador #7 e em seguida até o marcador #8. Para tanto, o sistema robótico deve se movimentar sequencialmente entre os marcadores, #1, #2, #3, #4, #5, #6, #7, #6, #5, #4, #9 e #8, como representado na Figura 72.

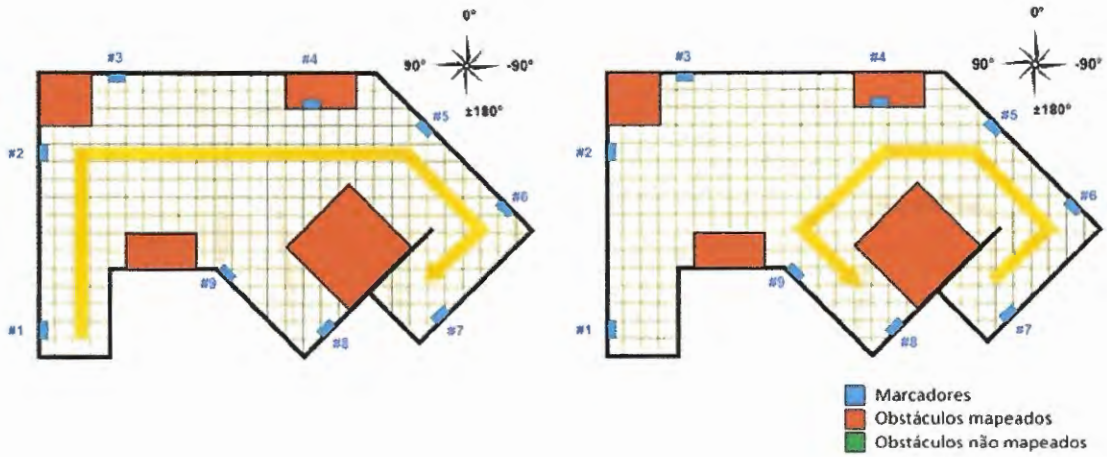

Figura 72 – Rota requisitada para o terceiro experimento.

E visando avaliar a robustez e confiabilidade dos métodos propostos, o teste foi repetido 12 vezes, considerando alguns fatores de ruído como: Diferentes dispositivos Android e Diferentes condições de iluminação, utilizando iluminação natual e artificial. Como na árvore de amostragem apresentada na Figura 73.

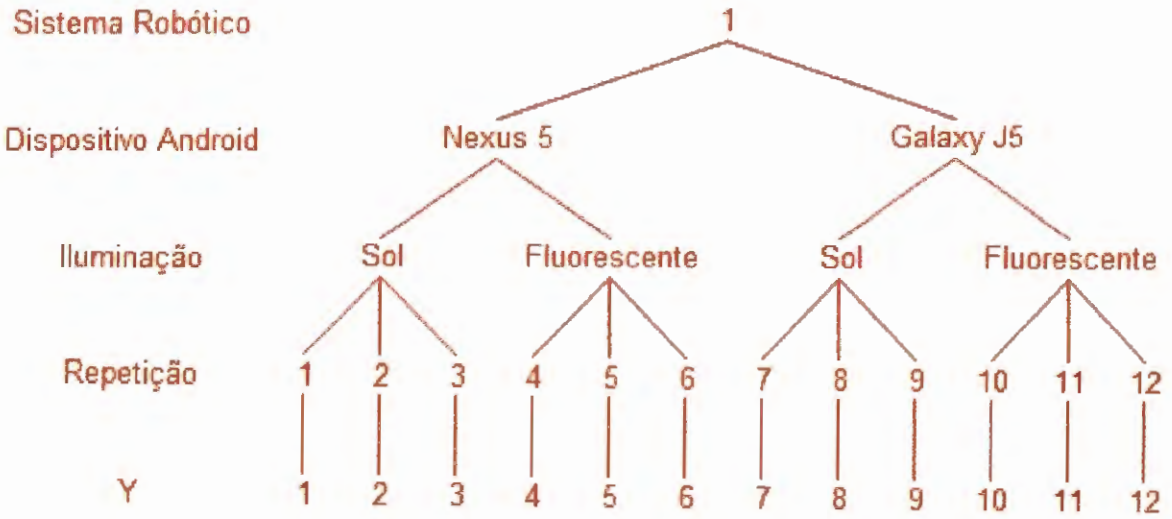

Figura 73 – Estratégia de amostragem para o terceiro experimento.

Orientação

Na Figura 74 apresenta-se o gráfico com a orientação teórica proposta pelo experimento e a orientação medida pela plataforma Android.

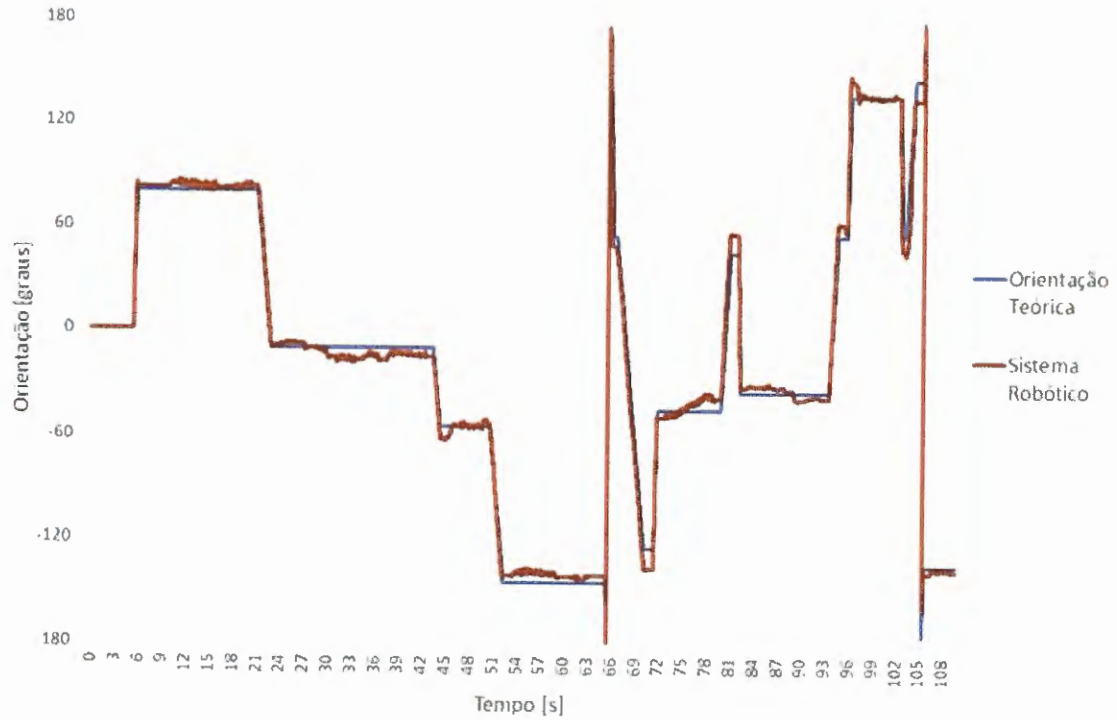

Figura 74 – Resposta da Fusão de Sensores Virtuais no terceiro experimento.

Como pode ser observado, as respostas da fusão sensorial se aproxima da orientação teórica proposta pelo experimento, porém com ruído elevado em alguns momentos. Isso se deve a alguns motivos:

- O controle do sistema robótico não está mais sendo performado pelo Arduino com base nas informações provenientes do conjunto de encoders das rodas e sim pelo Android, o que diminui significativamente a taxa de atualização do controle e ao fato de que o controle agora é performado para o sistema robótico como um todo e não separadamente para cada roda;
- A trajetória proposta pelo experimento demanda o mecanismo pan-tilt para movimentação do dispositivo Android em relação ao sistema robótico. Porém a utilização desse mecanismo habilita graus de liberdade entre o dispositivo Android e o sistema robótico, o que pode influenciar diretamente na malha de controle.

Posicionamento Local

Na Figura 75 apresenta-se o gráfico com a detecção dos marcadores pela plataforma Android.

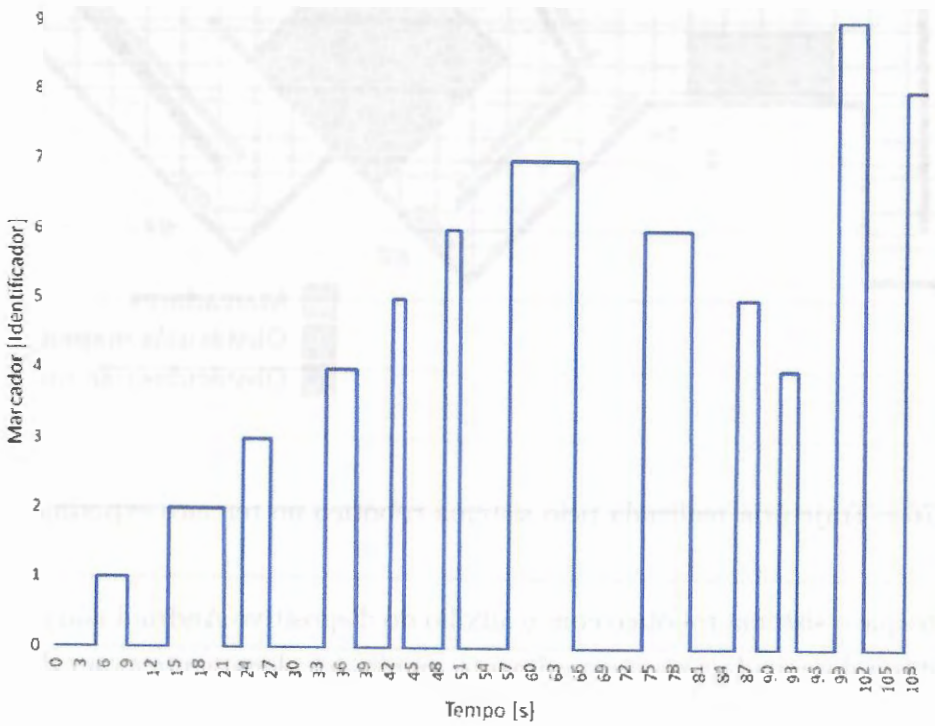

Figura 75 – Detecção dos marcadores no terceiro experimento.

Por meio desse gráfico fica evidente que a plataforma Android valendo-se da biblioteca ARToolkit é plenamente capaz de detectar corretamente marcadores no ambiente e tomar decisões corretas baseada nessas informações.

Resultado Global

Na Figura 76 apresenta-se uma visão esquemática da movimentação do sistema robótico no ambiente. Esse gráfico foi criado manualmente com base nos vídeos gravados durante a execução do experimento. Os vídeos em questão foram compilados e podem ser vistos pelo link: <https://youtu.be/G1YHjXgACYI>.

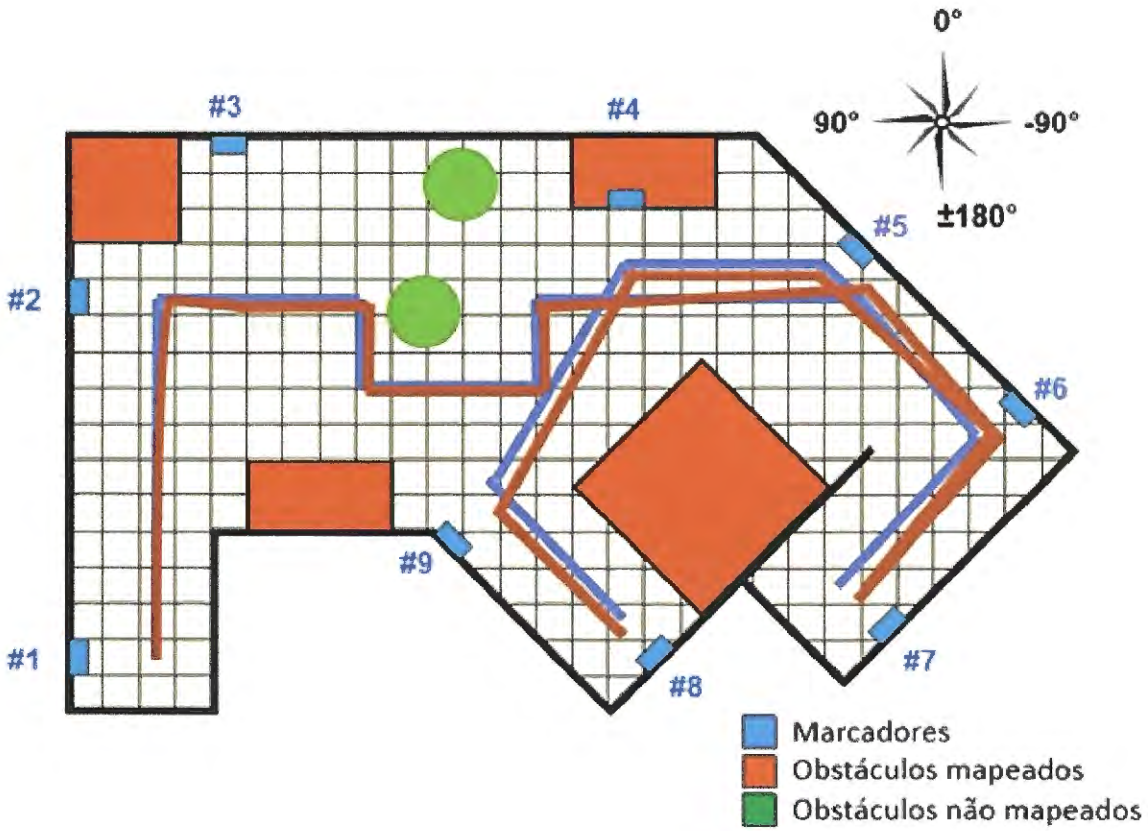

Figura 76 – Trajetória realizada pelo sistema robótico no terceiro experimento.

É evidente que o sistema robótico com o auxílio do dispositivo Android não performou a exata trajetória desenhada para o experimento, porém foi plenamente capaz de concluir as tarefas demandadas, sendo capaz de navegar autonomamente de um ponto de origem até duas posições alvos, até mesmo desviando de obstáculos não mapeados.

CAPÍTULO 5

Conclusões

De modo geral podemos considerar que este projeto atingiu seus objetivos, pois se verificou com sucesso que com as metodologias desenvolvidas foram capazes de realizar a localização do sistema robótico em um ambiente interno previamente conhecido e até mesmo controlar o sistema robótico, de forma a navegar o mesmo autonomamente entre pontos distintos do ambiente.

Para tanto, performou-se a localização relativa do sistema robótico em relação ao ambiente por meio da câmera incorporada à plataforma Android e marcadores bem definidos, previamente colocados em lugares estratégicos do ambiente. Também foi alvo de estudo métodos para a localização global do sistema robótico, através de sua orientação espacial, por meio dos sensores embutidos na plataforma Android. Sendo que uma das fusões sensoriais obteve resultados promissores e se mostrando robusta contra diversas fontes de ruído intrínsecas ao ambiente.

Por fim, a plataforma Android foi capaz de realizar todo o processamento necessário com uma frequência de atualização compatível com uma aplicação em tempo real. Porém como dito na proposição desse trabalho, a recomendação é que se use o aplicativo desenvolvido para enviar as informações para o robô móvel, como um sensor, e não comandá-lo. Isso não sobrecarrega o processamento dos filtros desenvolvidos e permite maior flexibilidade na implementação da navegação autônoma do robô.

5.1 Oportunidades de melhorias

Para reduzir o ruído visto no experimento final, aproximando ainda mais o robô de sua trajetória ideal algumas mudanças podem ser feitas, como:

- Utilizar um mecanismo mais robusto, que evite vibrações e desalinhamentos entre o dispositivo Android e o sistema robótico;
- Utilizar um mecanismo óptico, como um espelho hiperbólico, de forma que a câmera do dispositivo Android possa visualizar todo o ambiente ao redor do robô, e mantendo

suas coordenadas solidárias as coordenadas do sistema robótico;

- Utilizar mais de um dispositivo Android, de forma a cobrir todo o perímetro visual ao redor do sistema robótico;
- Implementar uma estratégia de navegação mais sofisticada e que utilize simultaneamente as informações de orientação e posição, de forma a permitir mais flexibilidade e robustez de movimentação ao sistema robótico.

Referências

BLESER, G., 2009, Advanced tracking through efficient image processing and visual inertial sensor fusion, Computers & Graphics.

BLESER, G., 2009, Towards Visual-Inertial SLAM for Mobile Augmented Reality, Fachbereich Informatik der Technischen Universität Kaiserslautern.

CARRERA J. et al. (2017). A real-time robust indoor tracking system in smartphones. Computer Communications.

CLAUS, D.; FITZGIBBON, A. W. (2005). Reliable Automatic Calibration of a Marker-Based Position Tracking System. Seventh IEEE Workshops on Application of Computer Vision.

COUTO, L. N., 2012, "Sistema para localização robótica de vepiculos autônomos baseado em visão computacional por pontos de referência", Instituto de Ciências Matemáticas e de Computação, USP, São Carlos, SP.

DEVELOPER. (2011). Android Developers SDK. Disponível em <http://developer.android.com/guide/topics/fundamentals.html>. Acesso em: 24 Mar. 2011.

DURRANT-WHYTE, H. F., LEONARD, J. J. (1992). Directed Sonar Navigation. Kluwer Academic Press, 1992.

FABRO, J. A., 1996, "Grupos neurais e sistemas nebulosos: aplicação à navegação autônoma", tese de mestrado, Faculdade de Engenharia Elétrica e de Computação, Unicamp, Campinas, SP.

FIGUEIREDO, M. (1999). Navegação Autônoma de Robôs. In: Fabian Viegas (FEEVALE). (Org.). VII Escola de Informática da SBC - Regional Sul. Novo Hamburgo: FEEVALE, v. , p. 74-106

GUNAWAN A. et al. (2017). Development of Affordable and Powerful Swarm Mobile Robot Based on Smartphone Android and IOIO board. 2nd International Conference on Computer Science and Computational Intelligence.

KATO, H. (1999). Marker Tracking and HDM Callibration for a Video-based Augmented Reality Conferencing System. Proceedings of the 2nd IEEE and ACM Internacional Workshop on Augmented Reality.

KATO, H.; BILLINGHURST, M.; POUPYREV, I. (2000). ARToolKit Version 2.33. HIT Lab, Universidade de Washington.

KOTHARI, N. et al. (2011). Robust Indoor Localization on a Commercial Smart-Phone. The Robotics Institute Carnegie-Mellon University Pittsburgh, Pennsylvania

LAWITZKI, P. (2012). Application of Dynamic Binatural Signals in Acoustic Games. Stuttgart Media University, Stuttgart, 2012.

MACHARET, D. G. (2009). Localização e mapeamento em terrenos irregulares utilizando robôs móveis. Dissertação (Mestrado) Univerdidade Federal de Minas Gerais, Belo Horizonte, 2009.

ONO K.; OGAWA H. (2014). Personal Robot Using Android Smartphone. International workshop on Innovations in Information and Communication Science and Technology.

PACHA, A., 2013, Sensor fusion for robust outdoor Augmented Reality tracking on mobile devices., Augsburg University.

PIEKARSKI, W.; Thomas, B. H. (2002). Using ARToolKit for 3D Hand Position Tracking in Mobile Outdoor Environments. 2nd Internacional Augmented Reality ToolKit Workshop 60

SELJANKO F. (2013). Low-Cost Electronic Equipment Architecture Proposal for Urban Search and Rescue Robot. International Conference on Mechatronics and Automation.

SHALA U.; RODRIGUEZ A. (2011). Indoor Positioning using Sensor-fusion in Android Devices. School of Health and Society Department Computer Science Embedded Systems

THRUN, S; BURGARD, W. & Fox, D. (2005). Probabilistic Robotic. The MIT Press.

THRUN, S. MONTEMERLO, M. & WEGBREIT, B. (2002). "FastSLAM: A factored solution to the simultaneous localization and mapping problem". In Proceedings of the AAAI National Conference on Artificial Intelligence.

WELCH, G; BISHOP, G. (2006). An Introduction to the Kalman Filter. University of North Carolina at Chapel Hill, 2006.

WOODMAN, OLIVER J. (2007). An introduction to inertial navigation. Technical Report, University of Cambridge.