



TRABALHO DE GRADUAÇÃO

**IMPLEMENTAÇÃO DE SISTEMA DE LOCALIZAÇÃO PARA
FUTEBOL DE ROBÔS BASEADO EM SENsoRES INERCIAIS**

Débora Ferreira dos Santos

Brasília, Dezembro de 2020



**ENGENHARIA
MECATRÔNICA**
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

IMPLEMENTAÇÃO DE SISTEMA DE LOCALIZAÇÃO PARA FUTEBOL DE ROBÔS BASEADO EM SENsoRES INERCIAIS

Débora Ferreira dos Santos

*Relatório submetido como requisito parcial de obtenção
de grau de Engenheiro de Controle e Automação*

Banca Examinadora

Profa. Mariana Costa Bernardes Matias, _____
FGA/UnB
Orientadora

Prof. Roberto de Souza Baptista, FGA/UnB _____
Examinador interno

Profa. Cláudia Patrícia Ochoa Diaz, FGA/UnB _____
Examinadora interna

Brasília, Dezembro de 2020

FICHA CATALOGRÁFICA

DÉBORA, FERREIRA DOS SANTOS

Implementação de sistema de localização para futebol de robôs baseado em sensores inerciais, [Distrito Federal] 2020.

ix, 58p., 210 x 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2020). Trabalho de Graduação – Universidade de Brasília, Faculdade de Tecnologia.

- | | |
|-----------------------|---------------------|
| 1. Localização | 2. Filtro de Kalman |
| 3. Sensores inerciais | 4. NAO |
| 5. RoboCup | |

I. Mecatrônica/FT/UnB

II. Implementação de sistema de localização para futebol de robôs baseado em sensores inerciais

REFERÊNCIA BIBLIOGRÁFICA

SANTOS, D.F., (2020). Implementação de sistema de localização para futebol de robôs baseado em sensores inerciais. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-n°01, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 67p.

CESSÃO DE DIREITOS

AUTOR: Débora Ferreira dos Santos

TÍTULO DO TRABALHO DE GRADUAÇÃO: Implementação de sistema de localização para futebol de robôs baseado em sensores inerciais.

GRAU: Engenheiro

ANO: 2020

É concedida à Universidade de Brasília permissão para reproduzir cópias deste Trabalho de Graduação e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desse Trabalho de Graduação pode ser reproduzida sem autorização por escrito do autor.

Débora Ferreira dos Santos

Universidade de Brasília (UnB) – Campus Darcy Ribeiro

70919-970 Brasília – DF – Brasil.

Agradecimentos

Primeiramente, gostaria de agradecer a minha orientadora, Mariana Bernardes, por toda a excelência técnica sempre pautada na sensibilidade. Obrigada pela compreensão, apoio e incentivo durante esses anos de tamanho aprendizado.

Agradeço ao Samuel por toda a confiança e por aceitar embarcar nesse tema comigo. Obrigada pela amizade para todas as horas e pela parceria fortalecida nesse período de elaboração dos nossos trabalhos.

À equipe UnBeatables, meu imenso carinho por todas as experiências proporcionadas. Sou muito feliz de ter conhecido cada um dos membros com quem trabalhei, viajei, competi e me diverti nos quatro anos em que fiz parte da equipe. Obrigada, Gabriel, Guilherme, Lívia, Natalia, Murilo e Paulo, por terem acreditado na minha capacidade de completar este trabalho e por todo o amparo oferecido principalmente nesses últimos meses. Obrigada, Eric, Felipe, Gildo, Henrique e Raphael, por todo o aprendizado, diversão e perrengues superados juntos. Com carinho, agradeço ao André pela força nos momentos mais delicados.

Eu não poderia ter ganhado melhores companheiras de curso que Juliana e Mariana. Obrigada por terem sido ombro, suporte, torcida e inspiração durante toda a graduação e por terem construído essa amizade que se estende fora da faculdade também.

Obrigada aos amigos que compartilharam a rotina da UnB comigo pelas trocas significativas que me permitiram uma formação profissional mais humana. Aos amigos de longa data, agradeço pelos constantes encorajamentos, mas, sobretudo, pelo prazer de poder ter crescido ao seu lado. Em especial, deixo meu agradecimento ao Caio e ao Victor pelo esforço em me auxiliar com os detalhes deste projeto.

Por fim, sou grata a minha família por toda a dedicação que colocaram na minha educação. À minha avó, agradeço pela compreensão com as ausências frequentes. Ao meu pai, pela paciência e pelo apoio às minhas escolhas. À minha mãe, por manter meu pé no chão e por jamais ter limitado as minhas possibilidades.

Débora Ferreira dos Santos

RESUMO

Na robótica móvel, a fim de completar certas tarefas, é necessário que o robô tenha conhecimento do ambiente em que está inserido e em que deve se locomover. No contexto de futebol de robôs, ele deve ser capaz de reconhecer características relevantes do campo de futebol e, a partir disso, determinar sua localização dentro desse campo. Este trabalho busca desenvolver um sistema de localização baseado em medições de unidade inercial aliadas à extração de informação das imagens da câmeras do robô NAO. O emprego do Filtro de Kalman Estendido permite essa fusão sensorial e, assim, são obtidas previsões e correções do estado do robô. O trabalho foi validado na plataforma NAO em um ambiente adaptado para reproduzir as particularidades do campo de futebol.

Palavras Chave: localização, Filtro de Kalman, fusão sensorial, NAO, sensores inerciais, RoboCup, robótica móvel

ABSTRACT

In order to complete certain tasks, a mobile robot must have knowledge of the environment in which it is inserted and must move. In the context of robot soccer, it must be able to recognize the soccer field features and then define its location within the field. This work aims to develop a localization system based on inertial unit measurements together with the information extracted from NAO robot camera images. The use of the Extended Kalman Filter allows sensor fusion and, thus, provides predictions and corrections of the robot's state. The work was validated on the NAO platform in an environment adapted to reproduce the particularities of the soccer field.

Keywords: localization, Kalman Filter, sensor fusion, NAO, inertial sensors, RoboCup, mobile robot

SUMÁRIO

1	Introdução.....	1
1.1	CONTEXTUALIZAÇÃO.....	1
1.2	MOTIVAÇÃO	2
1.3	DEFINIÇÃO DO PROBLEMA	3
1.3.1	CONFIGURAÇÃO DO AMBIENTE NA SPL	3
1.3.2	PLATAFORMA NAO	5
1.3.3	<i>Framework NAOQI</i>	7
1.3.4	CÓDIGO DA EQUIPE UNBEATABLES	7
1.3.5	MÉTODOS DE LOCALIZAÇÃO	8
1.4	OBJETIVOS DO PROJETO	10
1.5	APRESENTAÇÃO DO MANUSCRITO.....	10
2	Fundamentação	12
2.1	REPRESENTAÇÃO ESPACIAL	12
2.1.1	REPRESENTAÇÃO DA ROTAÇÃO.....	13
2.1.2	REPRESENTAÇÃO DA TRANSLAÇÃO	14
2.1.3	REPRESENTAÇÃO HOMOGÊNEA	14
2.2	ODOMETRIA	15
2.2.1	CÁLCULO DA MATRIZ DE ROTAÇÃO	16
2.2.2	CÁLCULO DA POSIÇÃO	18
2.2.3	ERROS NA ODOMETRIA	18
2.3	REPRESENTAÇÃO DA IMAGEM	18
2.4	MAPA	21
2.5	ESTIMATIVA DE ESTADO.....	22
2.5.1	FILTRO DE KALMAN	24
2.5.2	FILTRO DE KALMAN ESTENDIDO	25
3	Desenvolvimento.....	27
3.1	INTRODUÇÃO	27
3.2	MAPA	27
3.3	DETECÇÃO DE <i>landmarks</i>	28
3.4	ACESSO À MEMÓRIA	29
3.5	DADOS DOS SENSORES INERCIAIS	30

3.5.1	ERROS DO GIROSCÓPIO E ACELERÔMETRO	30
3.6	DADOS DA DETECÇÃO DE <i>landmarks</i>	33
3.7	PROJETO DO FILTRO	34
3.7.1	VARIÁVEIS DE ESTADO	34
3.7.2	FUNÇÃO TRANSIÇÃO DE ESTADO E FUNÇÃO DE CONTROLE	36
3.7.3	MATRIZ DE RUÍDO DE PROCESSO	36
3.7.4	FUNÇÃO DE MEDIDA	36
3.7.5	MATRIZ DE RUÍDO DE MEDIDA	37
3.7.6	CONDIÇÕES INICIAIS	38
3.8	IMPLEMENTAÇÃO DO FILTRO	39
3.8.1	ESTRUTURA BÁSICA DO CÓDIGO	39
3.8.2	CÁLCULO DAS MATRIZES JACOBIANAS	40
3.8.3	CÁLCULO DA COVARIÂNCIA NA FASE DE CORREÇÃO	40
4	Resultados.....	41
4.1	INTRODUÇÃO	41
4.2	AVALIAÇÃO DA PREDIÇÃO DE POSIÇÃO.....	43
4.3	AVALIAÇÃO DA PREDIÇÃO DE ORIENTAÇÃO.....	44
4.4	AVALIAÇÃO DA CORREÇÃO	46
4.5	RESULTADOS DO FILTRO EM AMBIENTE COM MÚLTIPLAS <i>landmarks</i>	48
5	Conclusões.....	52
5.1	PERSPECTIVAS FUTURAS	53
REFERÊNCIAS BIBLIOGRÁFICAS		54
Anexos.....		57
I	Programas utilizados.....	58

LISTA DE FIGURAS

1.1	Diagrama com as marcações do campo de futebol da SPL. Fonte: <i>RoboCup Technical Committee</i>	4
1.2	Dimensões do gol na SPL. Fonte: <i>RoboCup Technical Committee</i>	5
1.3	Sensores do robô NAO. Fonte: Adaptado de RobotLAB.....	5
1.4	Localização da câmera no NAO v4 e campos de visão. Fonte: <i>Softbank Robotics</i>	6
1.5	Diagrama do carregamento de módulos da NAOqi e respectivos métodos. Fonte: <i>Softbank Robotics</i>	7
1.6	Diagrama da estrutura do código da equipe UnBeatables. Fonte: Autora	8
2.1	Teste	19
2.2	Modelo <i>pinhole</i> da câmera. Fonte: BERNARDES, 2009, p. 22	19
2.3	Projeção de ponto na imagem seguindo modelo <i>pinhole</i> . Fonte: Adaptado de BERNARDES, 2009, p. 22	20
3.1	Interseções das linhas de marcação do campo indicadas pelos círculos tracejados. Fonte: Autora	28
3.2	Exemplos de padrões de Naomarks. Fonte: <i>Softbank Robotics</i>	29
3.3	Posição inicial de movimentação do robô NAO utilizada para a calibração dos sensores iniciais. Fonte: Autora	31
3.4	Fluxograma do processo de tratamento dos dados dos sensores iniciais. Fonte: Autora	32
3.5	Representação dos ângulos α e β da detecção de Naomarks. A bissetriz do campo de visão é apresentada em cor azul. As linhas verdes indicam o centro na imagem da Naomark detectada. Fonte: Autora	33
3.6	Representação das posições usadas para teste da detecção da <i>landmark</i> . Fonte: Autora	38
3.7	Diagrama da arquitetura do código implementado. Fonte: Autora.....	39
4.1	Ambiente de validação com o robô NAO observando uma Naomark afixada na parede. Fonte: Autora.....	41
4.2	Ambiente de validação construído com múltiplas Naomarks afixadas nos planos verticais. Fonte: Autora	42
4.3	Imagens, obtidas pelo Monitor, da câmera do robô posicionado em frente a uma Naomark. Fonte: Autora	43

4.4	Posição final do robô após comando de movimentação para aferição da predição de posição. Fonte: Autora.....	43
4.5	Resultados do filtro para x e y após teste de movimentação no eixo x para aferição da predição de posição. Fonte: Autora.....	44
4.6	Posição final do robô após comando de movimentação para aferição da predição de orientação. Fonte: Autora	45
4.7	Resultados do filtro para as projeções dos eixos x e y após teste de rotação em torno do eixo z para aferição da predição de orientação. Fonte: Autora.....	45
4.8	Posição final do robô após comando de movimentação para aferição da correção. Fonte: Autora	46
4.9	Resultados do filtro para x e y após teste de movimentação no eixo x para aferição da correção. Fonte: Autora	47
4.10	Resultados do filtro para as projeções dos eixos x e y após teste de movimentação no eixo x para aferição da correção. Fonte: Autora	48
4.11	Representação do ambiente de teste com múltiplas <i>landmarks</i> . Fonte: Autora	48
4.12	Representação tridimensional do ambiente de validação construído com múltiplas Naomarks afixadas nos planos verticais. Fonte: Autora	49
4.13	Resultados do filtro para x e y após teste de movimentação em ambiente com múltiplas <i>landmarks</i> . Fonte: Autora.....	49
4.14	Resultados do filtro para os eixos x e y após teste de movimentação em ambiente com múltiplas <i>landmarks</i> com faixa de valores limitada. Fonte: Autora	50
4.15	Resultados do filtro para as projeções dos eixos x e y após teste de movimentação em ambiente com múltiplas <i>landmarks</i> . Fonte: Autora	51

LISTA DE TABELAS

1.1	Medidas do campo de futebol da SPL	4
1.2	Especificações dos dispositivos dos robôs NAOs versão v4 e v6	6

LISTA DE SÍMBOLOS

Símbolos Latinos

p	coordenadas de um ponto	[m]
<i>x</i>	coordenada x de um ponto	[m]
<i>y</i>	coordenada y de um ponto	[m]
<i>z</i>	coordenada z de um ponto	[m]
<i>t</i>	instante de tempo	[s]
<i>B</i>	representação da integração da matriz Ω	[rad]
a	vetor aceleração	[m/s ²]
v	vetor velocidade	[m/s]
g	vetor aceleração da gravidade	[m/s ²]
<i>f</i>	distância focal da câmera	[m]
<i>k</i>	fator de escala da imagem no eixo <i>x</i>	[pixel/m]
<i>l</i>	fator de escala da imagem no eixo <i>y</i>	[pixel/m]
x	vetor de estado	[m]
mov	vetor de comando de movimentação	[m] e [rad]

Símbolos Gregos

θ	Ângulo de rotação	[rad]
ω	Vetor velocidade de rotação	[rad/s]
$\delta\phi$	Rotação em torno do eixo <i>x</i>	[rad/s]
$\delta\theta$	Rotação em torno do eixo <i>y</i>	[rad/s]
$\delta\psi$	Rotação em torno do eixo <i>z</i>	[rad/s]
Ψ	Matriz de representação de ângulos	[rad]
Ω	Vetor velocidade angular do sistema	[rad/s]
α	ângulo em torno do eixo <i>y</i> da câmera	[rad]
β	ângulo em torno do eixo <i>z</i> da câmera	[rad]
$\boldsymbol{\theta}$	Vetor de ângulo de rotação	[rad]
σ	desvio padrão	[m] e [rad]

Grupos Adimensionais

SE	Grupo euclídeo especial
SO	Grupo ortogonal especial
\mathbb{R}	Conjunto dos números reais
\mathbb{R}^n	Espaço vetorial real n-dimensional
G	Sistema de coordenadas global
L	Sistema de coordenadas local
\mathbf{R}	Matriz de rotação
\mathbf{r}	Vetores que compõem a matriz de rotação
r	Elementos que compõem a matriz de rotação
\mathbf{T}	Matriz de translação
\mathbf{H}	Matriz de representação homogênea
\mathbf{A}	Matriz de rotação do sistema local
\mathbf{I}	Matriz identidade
π	plano da imagem
O	origem do sistema de coordenadas
\mathbf{K}	matriz de calibração da câmera
p	probabilidade
Σ	matriz de covariância
μ	vetor média
\mathbf{z}	vetor de medidas
\mathbf{u}	vetor de sinais de controle

Subscritos

l	referente ao sistema de coordenadas local
x	referente ao eixo x
y	referente ao eixo y
z	referente ao eixo z
g	referente ao sistema de coordenadas global
i, j	índices
t	referente à translação
0	referente ao centro da imagem
$lmark$	referente a <i>landmark</i>
c	referente à câmera

Sobrescritos

-	Representação em coordenadas homogêneas
.	representação da taxa de variação
c	referente ao sistema de coordenadas da câmera
π	referente ao plano da imagem
i	referente ao sistema de coordenadas da imagem

Siglas

API	Interface de Programação de Aplicação
EDO	Equação diferencial ordinária
EKF	Filtro de Kalman Estendido
FDP	Função densidade de probabilidade
FIFA	Federação Internacional de Futebol
KF	Filtro de Kalman
LARA	Laboratório de Robótica e Automação
MCL	Localização Monte Carlo
SPL	<i>Standard Platform League</i>

Capítulo 1

Introdução

Ao longo da história, sempre foi interesse de estudo o desenvolvimento de dispositivos com algum nível de inteligência que pudessem auxiliar o trabalho humano. O anseio de se obter um invento que servisse ao homem aliado à necessidade de se substituir a força humana em atividades perigosas levaram ao desenvolvimento da robótica moderna [1]. Dentro desta área, um robô é definido como um sistema com sensores e atuadores capaz de autonomamente perceber o ambiente no qual está inserido e dele extrair informações, processá-las e então atuar nesse ambiente a fim de atingir um objetivo estabelecido [2, 3].

A robótica pode ser dividida em duas frentes de estudo de acordo com a mobilidade da estrutura do robô: robôs manipuladores e robôs móveis [1, 3]. Robôs manipuladores têm sua plataforma fixa durante toda a execução da tarefa e o volume e forma de seu espaço de trabalho são determinados pela estrutura do manipulador. Já os robôs móveis têm uma plataforma móvel que permite a atuação em um espaço que excede os limites de suas dimensões. O foco do estudo nessa subárea é a autonomia para a navegação.

Em algumas aplicações, a resposta sensorial pode ser suficiente para a navegação. O robô recebe do ambiente informações relacionadas à direção que ele deve tomar, como, por exemplo, a incidência de luz ou campo potencial [4]. No entanto, para outras aplicações, as informações do ambiente podem ser limitadas e não simbólicas. Nesse caso, pode ser necessário que o robô seja capaz de conhecer sua posição com relação a um referencial global, isto é, capaz de se localizar no ambiente em que está inserido para então se movimentar no espaço satisfatoriamente [2].

1.1 Contextualização

Durante o *Workshop on Grand Challenges in Artificial Intelligence*, realizado em Tóquio no ano de 1992, a discussão principal envolvia a concepção de projetos que apresentassem desafios técnicos disruptivos [5]. Tais projetos deveriam, durante sua execução, propiciar o desenvolvimento de pesquisas em robótica e inteligência artificial e, uma vez concluídos, ter a simbologia de um

grande marco alcançado nessas áreas. Como fruto dessa conferência, surgiu a *RoboCup*¹, um projeto que utiliza o futebol como uma forma viável de promoção de tecnologias com impactos social, econômico e científico relevantes [6]. O objetivo final dessa iniciativa é desenvolver um time de robôs que seja capaz de ganhar da seleção humana vencedora da Copa do Mundo da Federação Internacional de Futebol (FIFA)² no ano de 2050 [7].

O jogo de futebol é capaz de fornecer um ambiente de validação padronizado para o desenvolvimento de pesquisas e também de reproduzir as complexidades do mundo real [7]. Em uma partida do jogo, há múltiplos agentes atuando em um ambiente dinâmico e não determinístico de forma cooperativa, competitiva e também neutra, a partir de um controle distribuído. Além disso, a aquisição de informações do mundo pelo robô é limitada e não simbólica [8]. Assim, diversos campos de estudo podem ser explorados em uma única configuração principal.

A *RoboCup* conta com cinco ligas de futebol com variações particulares das regras e ambiente. Em uma delas, a *Standard Platform League* (SPL)³, adota-se, para todas as equipes, uma plataforma padronizada, atualmente o robô NAO fabricado pela *SoftBank Robotics* cujo *hardware* não deve sofrer alterações. O campo da partida é semelhante ao do futebol humano com marcações brancas e duas balizas em medidas reduzidas conforme determinação do livro de regras da liga. Os times são formados por cinco robôs que devem operar autonomamente, isto é, apesar de cooperarem entre si, devem ser capazes de separadamente processar dados e tomar decisões.

Por conta da alta complexidade do sistema que representa o jogo de futebol, para a navegação do robô no campo, uma navegação reativa não seria suficiente. Assim, é necessário que o robô se localize dentro do campo para, em seguida, poder se movimentar nele. Ainda, no contexto da SPL, o algoritmo de localização deve utilizar-se apenas dos sensores de que o NAO é dotado e de sua capacidade de processamento.

1.2 Motivação

A equipe UnBeatables, filiada ao Laboratório de Robótica e Automação (LARA), é formada por alunos da graduação da Universidade de Brasília e participa de competições na categoria SPL. O time participou de sua primeira competição em 2014 na *RoboCup* realizada em João Pessoa e, desde então, tem trabalhado a fim de desenvolver aspectos da arquitetura do código, percepção, comportamento, comunicação e movimentação dos robôs. Desde a aquisição dos robôs pelo LARA, além do desenvolvimento para a competição, também foram realizados trabalhos de graduação englobando problemas do ambiente da SPL e outras aplicações.

Em 2014, Carvalho [9] desenvolveu um sistema de controle dos movimentos de um humanoide para aplicação em teleoperação. Os movimentos humanos foram detectados a partir de um Kinect e o controle foi realizado por cinemática inversa baseada em quatérnios duais. A plataforma

¹O termo RoboCup é uma abreviação para Robot World Cup Initiative. Em português, Iniciativa da Copa do Mundo dos Robôs.

²Em francês, Fédération Internationale de Football Association, FIFA.

³Em português, Liga de Plataforma Padrão.

NAO foi utilizada para validação desse trabalho. Ainda no contexto da teleoperação, em 2016, Balbino [10] estendeu o trabalho de Carvalho incluindo a caminhada do teleoperador como um dos movimentos que o robô pudesse reproduzir. Também implementou a transmissão das imagens da câmera do NAO em formato de realidade virtual para possibilitar a imersão do teleoperador no ambiente do robô.

Em 2016, o comitê técnico da SPL propôs um desafio de comunicação entre dois robôs que não se baseasse em redes sem cabo [11]. Motivados por esse desafio, Faria e Rocha [12] implementaram ferramentas de comunicação baseadas em visão computacional no robô NAO. Acresceram ainda ao trabalho o desenvolvimento de métodos de controle cooperativo. Também em 2016, Resende [13] desenvolveu um algoritmo de localização no campo de futebol baseado em visão computacional e medidas inerciais. A implementação e validação por simulação foram realizadas em ambiente MATLAB.

Este trabalho tem como objetivo contribuir para o histórico de desenvolvimento da equipe UnBeatables a partir da construção de um módulo de localização a ser implementado no robô NAO e acoplado ao código da equipe para uso em competições.

1.3 Definição do problema

O presente trabalho utiliza o contexto de uma partida da SPL como ambiente de desenvolvimento de um sistema de localização a ser implementado e validado na plataforma NAO. Pretende-se ainda que esse módulo possa ser integrado ao código desenvolvido pela equipe UnBeatables e utilizado posteriormente nas competições das quais participa o time. Nas seções seguintes são detalhados o ambiente em que o robô deve realizar a localização, as especificações da plataforma NAO, as características do código de competição da equipe e as técnicas de localização da literatura.

1.3.1 Configuração do ambiente na SPL

A cada ano, as regras da *RoboCup* são alteradas para tornar seus jogos mais parecidos com uma partida entre humanos. Essas regras definem as dimensões e materiais do campo e bola, condições de iluminação e o próprio andamento da partida. As regras publicadas para o ano de 2019 [14] definem o uso de um campo de grama sintética verde com as marcações das linhas de meio de campo, círculo central, laterais, fundo, pequena área, cruz do pênalti e marca de meio de campo em cor branca conforme são apresentadas na Figura 1.1 e seguindo dimensões apresentadas na Tabela 1.1.

O livro de regras define ainda as características da bola utilizada na partida: uma bola branca e preta com 10 cm de diâmetro. Também determina que a iluminação do campo deve ser primordialmente natural, podendo, porém, ser utilizado um dispositivo de iluminação adicional a fim de garantir que não haja uma iluminação inferior a 300 lx ou que a razão entre as luminosidades da região mais iluminada e da menos iluminada não ultrapasse o valor de 10.

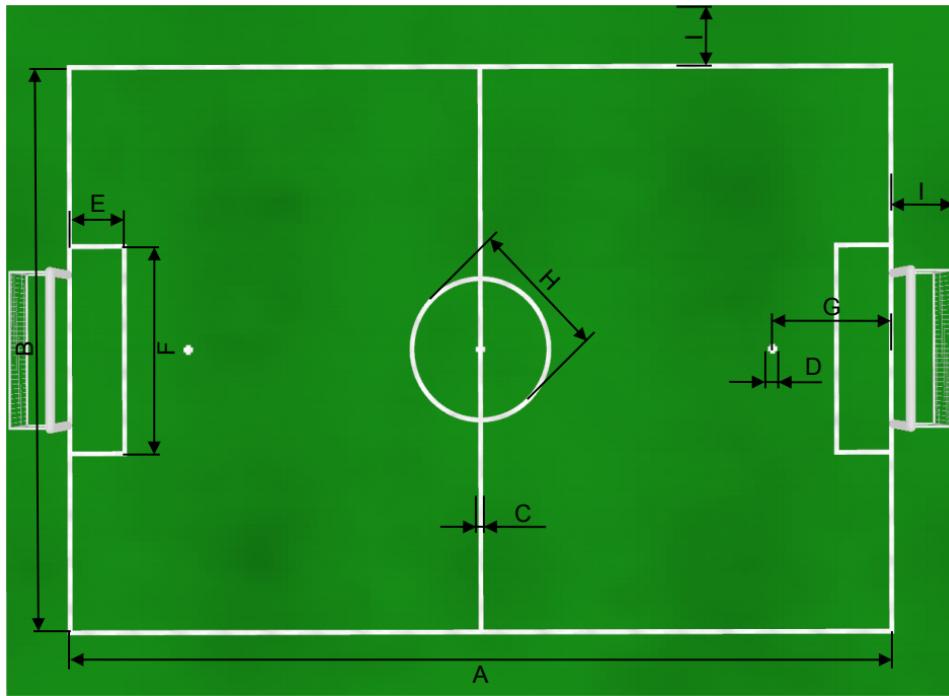
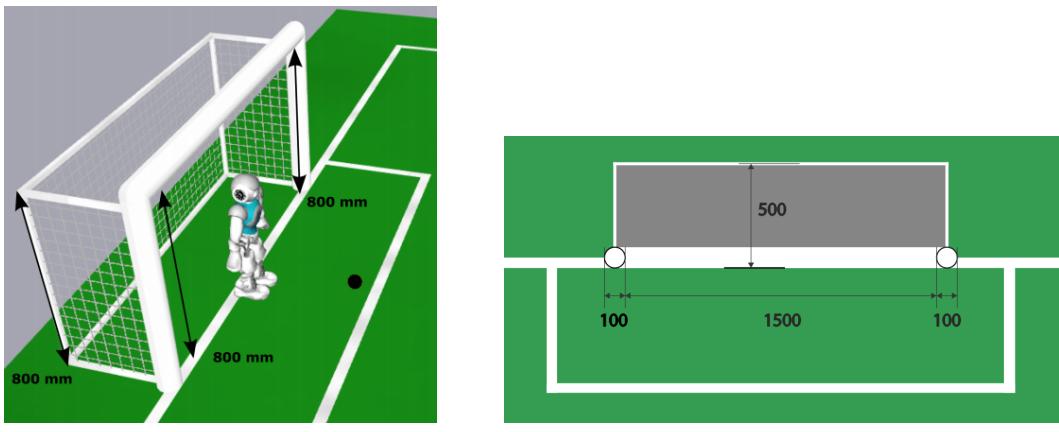


Figura 1.1: Diagrama com as marcações do campo de futebol da SPL. Fonte: *RoboCup Technical Committee*

Tabela 1.1: Medidas do campo de futebol da SPL

Símbolo	Descrição	Medida (mm)
A	Comprimento do campo	9000
B	Largura do campo	6000
C	Largura da linha	50
D	Marca do pênalti	100
E	Comprimento da área do pênalti	600
F	Largura da área do pênalti	2200
G	Distância da marca do pênalti	1300
H	Diâmetro do círculo de meio de campo	1500
I	Distância entre linha de campo e fim do gramado	700

Por fim, no que concerne à instalação do ambiente de jogo, também são definidas as características para a construção do gol. As balizas principais do gol devem ser feitas de cilindros brancos. A rede e a estrutura de suporte podem ter cores branca, cinza ou preta. As dimensões da estrutura completa e seu posicionamento no campo podem ser verificados na Figura 1.2.



(a) Aparência e dimensões do gol

(b) Vista superior com medidas em mm

Figura 1.2: Dimensões do gol na SPL. Fonte: *RoboCup Technical Committee*

1.3.2 Plataforma NAO

O NAO é um robô humanoide produzido pela empresa *SoftBank Robotics* e passou a ser utilizado como plataforma oficial da *RoboCup* a partir de 2008. A empresa comercializa uma edição específica para o uso em competições que tem um número reduzido de dispositivos em comparação com a versão acadêmica. A versão para a *RoboCup* conta com os seguintes dispositivos: LEDs; botões táteis e de contato na cabeça, peito, mãos e pés; alto-falantes e microfones na cabeça; sensores resistivos localizados nos pés do robô; sensores de posição em seus motores; dois pares de sonar localizados em seu peito; uma unidade inercial; e duas câmeras em sua cabeça. Uma representação do posicionamento de alguns desses sensores no robô pode ser verificada na Figura 1.3.

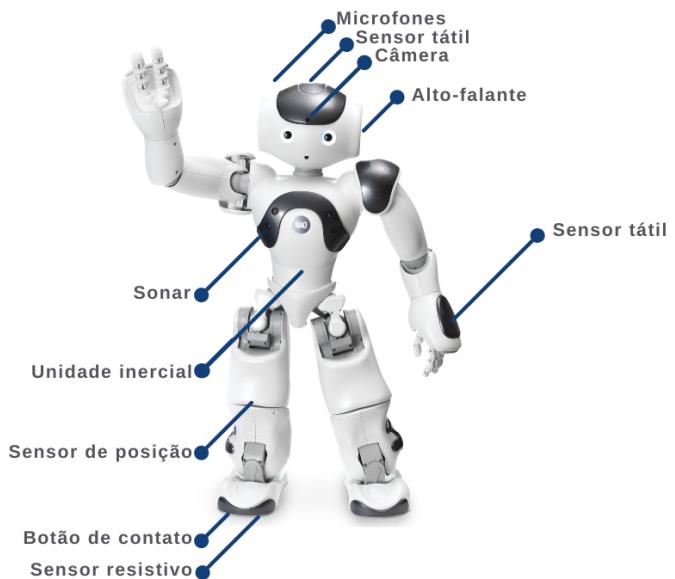


Figura 1.3: Sensores do robô NAO. Fonte: Adaptado de RobotLAB

A unidade inercial fica localizada no torso do robô e é composta por giroscópio e acelerômetro. Conta ainda com processador próprio capaz de fornecer o ângulo do torso a partir de algoritmo programado pela fabricante da plataforma, que realiza a fusão entre os dois sensores [15, 16]. As duas câmeras estão localizadas na cabeça do robô conforme indicado na Figura 1.4. O robô conta ainda com um processador Atom. Desde o primeiro lançamento, a plataforma vem sofrendo melhorias e novas versões foram disponibilizadas. As especificações de todos esses dispositivos variam entre as versões do NAO já lançadas. São apresentadas na Tabela 1.2 as especificações para as versões NAO v4 e NAO v6, as versões que o LARA atualmente tem.

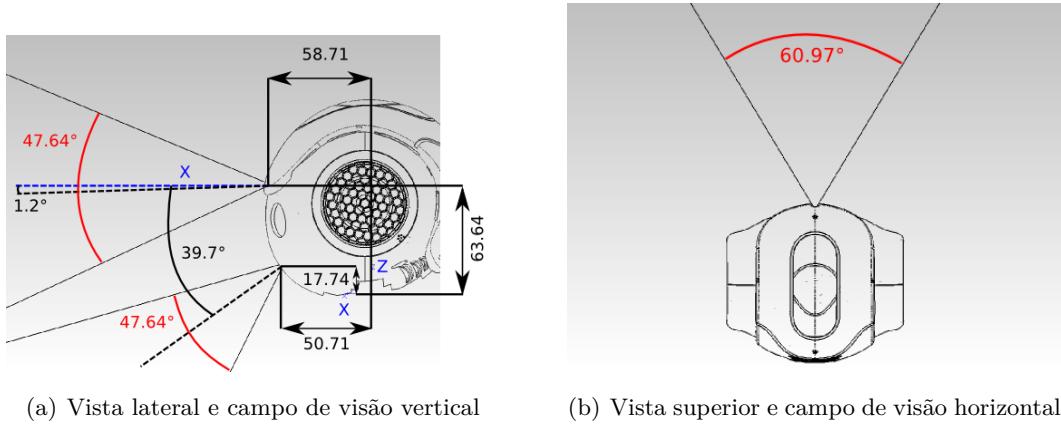


Figura 1.4: Localização da câmera no NAO v4 e campos de visão. Fonte: *Softbank Robotics*

Tabela 1.2: Especificações dos dispositivos dos robôs NAOs versão v4 e v6

Dispositivo	Parâmetro	V4	V6
Câmera	Resolução	1,22 Mp	5 Mp
	Pixels ativos (h x v)	1288 x 968	2592 x 1944
	Tamanho do pixel	1,9µm x 1,9µm	1,4µm x 1,4µm
	Saída da câmera	1280 x 960 (30fps)	640 x 480 (30fps) ou 2560 x 1920 (1fps)
	Tipo de foco	Fixo	Automático
	Campo de visão horizontal	60,9º	56,3º
	Campo de visão vertical	47,6º	43,7º
	Campo de visão diagonal	72,6º	67,4º
Giroscópio	Número de eixos	2	3
	Velocidade angular	~500º/s	~500º/s
	Posição (x,y,z)	(-0,008, 0,006, 0,029)	
Acelerômetro	Número de eixos	3	3
	Aceleração	~2 g	~2 g
	Posição (x,y,z)	(-0,008, 0,00606, 0,027)	
Placa-mãe	Processador	ATOM Z530	ATOM E3845
	RAM	1 GB	4 GB
	Frequência de relógio	1,6 GHz	1,91 GHz

1.3.3 Framework NAOqi

O NAO conta com uma distribuição do GNU/Linux baseada no Gentoo e adaptada para as especificidades do robô [15]. A fim de prover essas especificidades, o robô conta ainda com o *middleware* NAOqi de que dependem todos os comportamentos do robô. A NAOqi permite que o usuário se utilize de softwares pré-determinados para criar um código que cumpre um objetivo específico. Além disso, ela é responsável por gerenciar o fluxo de controle, ou seja, realizar as chamadas aos programas e às funções. Esse *framework* contém um conjunto de ferramentas particulares ao NAO como compiladores, bibliotecas e Interface de Programação de Aplicação (API)⁴. O uso de API facilita a comunicação em um ambiente distribuído com diversas bibliotecas para métodos de acesso e controle do *hardware* do robô [17]. São fornecidas pela *Softbank Robotics* APIs que vão desde as fundamentais, de acesso à memória e gerenciamento de comportamento por exemplo, até APIs de visão e movimento, entre outras. O carregamento dos módulos correspondentes às APIs é representado na Figura 1.5.

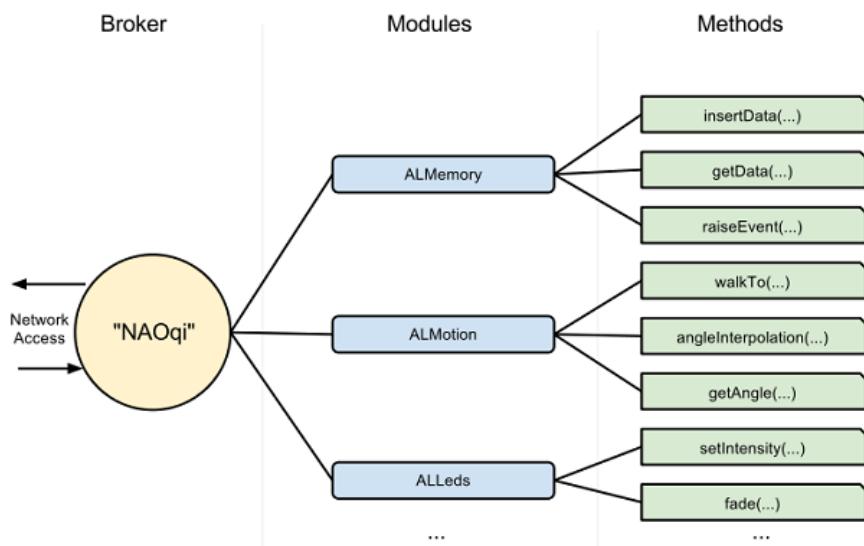


Figura 1.5: Diagrama do carregamento de módulos da NAOqi e respectivos métodos. Fonte: *Softbank Robotics*

1.3.4 Código da equipe UnBeatables

O código da equipe UnBeatables é desenvolvido utilizando os módulos disponibilizados pela fabricante e também outros módulos independentes. A arquitetura do código criada pelo time é baseada no uso de memória compartilhada entre as quatro *threads*⁵ responsáveis pelo gerenciamento da percepção, comportamento, locomoção e comunicação [18]. A detecção de bolas e características de interesse a partir da câmera é realizada dentro da *thread* de percepção. O com-

⁴Em inglês, Application Programming Interface, API.

⁵Em português, linha de execução. É a divisão de um processo em diversos fluxos de controle dentro de um programa.

portamento é baseado em uma máquina de estados que determina o próximo comportamento a ser tomado a partir do estado atual e da extração de informações da memória compartilhada, que foram atualizadas pelos outros módulos. Já a locomoção é responsável pelo controle dos movimentos do robô utilizando como entrada o comportamento determinado pela *thread* anterior. Por fim, a comunicação é responsável pela transmissão de mensagens entre os robôs e o computador principal da liga pelo qual o andamento do jogo é controlado. A Figura 1.6 representa a estrutura do código da UnBeatables, evidenciando as *threads* gerenciadas, suas funções e a utilização de serviços da NAOqi em seus métodos.

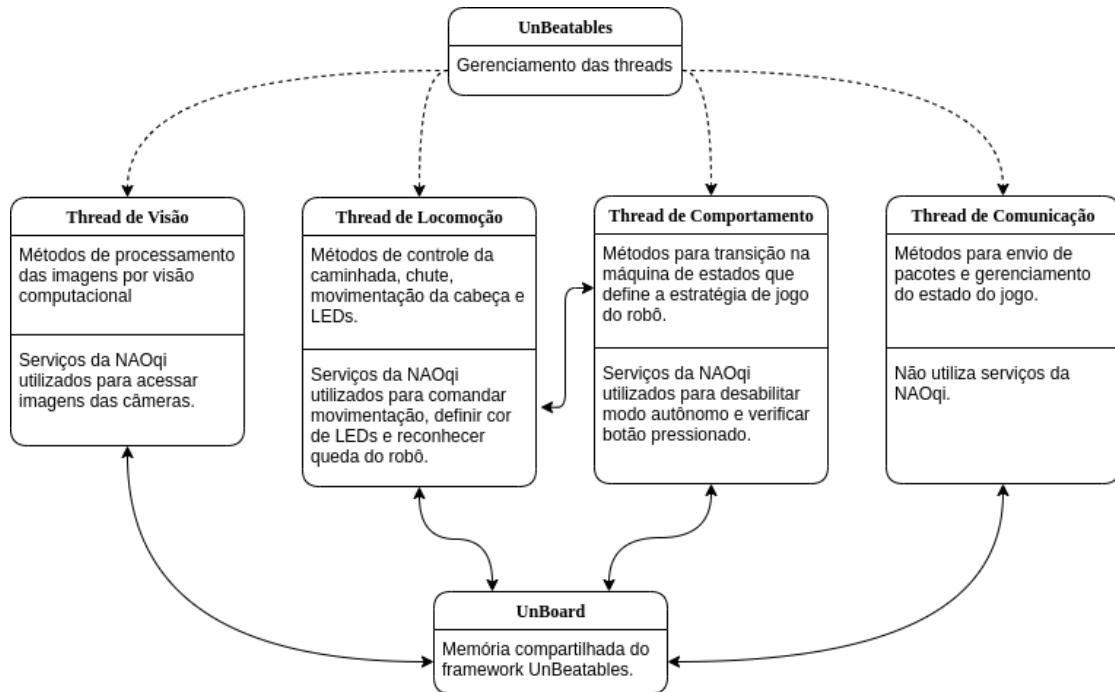


Figura 1.6: Diagrama da estrutura do código da equipe UnBeatables. Fonte: Autora

Por conta da modularização do código, sua manutenção se torna mais simples. A inserção de novas funcionalidades pode ser facilmente realizada pela implementação de um módulo independente, gerenciado dentro de alguma das *threads*, com a devida atualização de informações na memória. Outros módulos podem ter seus comportamentos alterados para possibilitar o uso dos dados providos pelo novo módulo.

1.3.5 Métodos de localização

Como é de conhecimento prévio o mapa relativo ao campo de futebol, o problema aponta para a determinação da posição de um robô no campo, tendo como base somente as leituras de seus próprios sensores. As informações dos sensores em robôs móveis não são dadas com relação a um referencial global, até mesmo porque, com a locomoção, as coordenadas desse referencial não coincidiriam com as coordenadas do robô. Os sensores são, na verdade, incrementais e a estimativa de um estado pode ser dada a partir do estado anterior [1]. A posição em um determinado momento de amostragem pode ser dada com relação à junção do estado anterior e a estimativa

da velocidade e direção de movimentação do robô [2]. Essa técnica para obtenção da posição do robô considerando os dados dos seus sensores internos e o conhecimento de estados anteriores é denominada odometria. Contudo, essa estimativa do estado é indireta e apresenta ruídos inerentes à medição, devido principalmente ao escorregamento do robô no gramado, por isso, a incerteza aumenta com o maior número de estados a serem estimados [3].

Desta maneira, é necessário que as informações dos sensores inerciais sejam complementadas. O uso de visão computacional proporciona o reconhecimento das características distintivas do campo. A partir do reconhecimento desses atributos, é possível fazer a projeção da imagem da câmera e calcular a distância do robô ao ponto de referência. Assim, a posição estimada pode ser corrigida por meio de fusão sensorial entre câmera e sensores inerciais, evitando o acúmulo crescente de erros da odometria. Embora existam técnicas que forneçam múltiplas hipóteses de localização sem diferenciação de probabilidade entre elas [19], a abordagem mais comum é a estimativa com mensuração da incerteza da posição do robô. Por conta da natureza das medições dos sensores que devem ser associadas, o modelo bayesiano é base de desenvolvimento de vários métodos para esse tipo de problema. A estimação bayesiana usa densidade de probabilidade para representar a estimativa de posição calculada. Essa abordagem é realizada em duas etapas. A primeira, a predição, está relacionada à movimentação do robô. Nessa fase, a posição do robô é obtida utilizando a informação da posição anterior e a leitura dos sensores inerciais. A predição corresponde à etapa em que há acúmulo de erro devido à odometria. A segunda fase é a de correção e está relacionada à percepção do ambiente pelo robô. As leituras dos sensores permitem encontrar a diferença entre a posição estimada pela movimentação e a posição encontrada pela percepção, levando a uma diminuição do erro associado aos cálculos [20].

Caso se tenha conhecimento da posição inicial do robô, pode-se utilizar uma função densidade de probabilidade gaussiana para representar a estimativa de sua postura. Nesse caso, pode ser adotado o Filtro de Kalman (KF)⁶. Esse filtro requer que os modelos de entrada do algoritmo de localização sejam lineares e é considerado um método robusto, acurado e ótimo [21]. Em situação não linear, pode-se utilizar a representação de variáveis pelo primeiro e segundo momento de sua função densidade em uma variação da abordagem denominada Filtro de Kalman Estendido (EKF)⁷ [22].

As demais abordagens probabilísticas partem da premissa de que não se conhece a posição inicial do robô. Diante disso, uma função densidade de probabilidade gaussiana não pode ser adotada. O método Markov utiliza um modelo multimodal para a crença da posição do robô. Isso possibilita a resolução do problema de localização global, isto é, o processo de localização sem conhecimento da posição inicial, e também possibilita a representação de ambiguidades [23]. Há duas abordagens para a localização Markov. A primeira utiliza mapas topológicos cuja baixa resolução tem implicações na acurácia das tarefas realizadas, sendo inaplicável a depender do objetivo desejado [21]. A outra abordagem utiliza um mapeamento em grade, que apresenta maior resolução, porém pode não ser uma aplicação realizável em tempo real por conta da alta demanda computacional e de memória [21, 22].

⁶Em inglês, Kalman Filter, KF.

⁷Em inglês, Extended Kalman Filter, EKF.

O método de Localização Monte Carlo (MCL)⁸ é uma variação do método Markov. Nele, um conjunto de amostragem é utilizado para aproximar distribuições de probabilidade. Por meio de um algoritmo on-line, a amostragem e reamostragem são feitas segundo uma sequência de importância relacionada a sua probabilidade [22]. Dessa forma, os cálculos são mantidos em regiões de provável interesse e o custo computacional é reduzido. Apresenta ainda resultados mais acurados e implementação mais simples em relação ao método anterior [21].

No âmbito da RoboCup, a localização Monte Carlo é amplamente utilizada nas diversas ligas, incluindo a SPL [24]. A maior parte das equipes na SPL utiliza um sistema de localização baseado em Monte Carlo devido a sua precisão, porém esse método exige configurações adequadas dos parâmetros referentes aos modelos [24]. Outra parcela significativa das equipes, porém, utiliza sistemas de localização baseados em Filtro de Kalman e variações, que possuem menor custo computacional quando comparados a Monte Carlo. [25]

1.4 Objetivos do projeto

O objetivo deste trabalho é desenvolver e implementar um sistema de localização para o robô NAO. O trabalho poderá ser utilizado futuramente em um campo de futebol da categoria SPL integrado ao código de competição da UnBeatables. Embora no momento de desenvolvimento não seja utilizado o *framework* da equipe, é importante que o código mantenha a mesma estrutura para a futura incorporação.

A predição da posição e orientação do robô pode ser dada a partir dos dados da unidade de medição inercial e a correção, a partir das câmeras. Para reproduzir os elementos do campo de futebol e também a detecção visual, deve ser construído um ambiente de validação e o respectivo módulo de processamento para extração de informações. O módulo de localização deve receber como entrada, além dos dados anteriormente mencionados, as medições dos sensores iniciais. A saída fornecida deve ser a estimativa do posicionamento do robô, considerando posição e orientação. A fusão sensorial deve ser feita a partir do uso de um Filtro de Kalman Estendido.

O desenvolvimento desse módulo deve considerar ainda a capacidade de processamento da plataforma e, portanto, deve ser executado sem causar prejuízo às demais rotinas do robô durante a partida de futebol.

1.5 Apresentação do manuscrito

O Capítulo 2 deste manuscrito introduz conceitos básicos de representação de pontos e corpos rígidos no espaço, uso dos sensores iniciais e estimativa de estados por meio de filtros bayesianos. No Capítulo 3, é detalhada a metodologia seguida para a elaboração do sistema de localização proposto neste trabalho. São apresentadas primeiramente as soluções para representação do mundo real, substituição dos elementos do campo de futebol, detecção de tais elementos, acesso a infor-

⁸Em inglês, Monte Carlo Localization, MCL.

mações do *hardware* do robô e também utilização dos dados de sensores e de medição. Definidas essas soluções, o Capítulo 3 é finalizado com a apresentação do projeto do filtro e detalhes de sua implementação. O Capítulo 4 é dedicado à apresentação dos resultados obtidos com o sistema desenvolvido e análise de seu desempenho. Por fim, o Capítulo 5 apresenta as conclusões finais do sistema implementado e propõe trabalhos futuros que podem ser feitos a partir deste projeto.

Capítulo 2

Fundamentação

Neste capítulo são explorados conceitos básicos e fundamentos matemáticos necessários para o desenvolvimento deste trabalho. São apresentadas a representação espacial de um objeto, os cálculos do processo de odometria, a representação da imagem, formas de mapeamento e, por fim, técnicas de estimativa de estados.

2.1 Representação espacial

Para o problema de localização é imprescindível definir uma forma de representação do robô no seu ambiente. No espaço tridimensional, um ponto pode ser facilmente representado por um vetor que o relaciona a um sistema de coordenadas cartesianas de referência [2]. Já um objeto é constituído por um conjunto de pontos. Considerando um corpo rígido, isto é, um corpo que não sofre deformações e, portanto, não tem sua dimensão nem sua forma alteradas, todos os pontos definidos nesse objeto mantêm entre si distâncias constantes. Para representar o corpo, portanto, pode-se definir um sistema de coordenadas local, geralmente localizado no centro de gravidade do objeto ou outro ponto de interesse para o problema. Assim, ao invés de monitorar a movimentação de cada um dos pontos individualmente, monitora-se apenas a movimentação desse novo sistema de coordenadas. Nessa configuração, pode-se recuperar a informação sobre o posicionamento de pontos específicos a partir da posição da origem do sistema e das relações conhecidas entre os pontos [2, 26].

O movimento de um corpo rígido pode ser representado por um mapeamento contínuo e único do objeto no espaço $SE(3)$, o que será melhor explanado posteriormente. Para que tal operação represente o deslocamento rígido, ela deve garantir não só a preservação das distâncias internas do objeto, mas também as orientações internas, pois, do contrário, haveria matematicamente a representação de reflexões que são fisicamente impossíveis ao corpo. Os dois requisitos são cumpridos a partir da utilização de um modelo que transforme um sistema de coordenadas ortonormal para outro sistema do mesmo tipo [26].

Por fim, as transformações referentes à movimentação do corpo rígido podem ser modela-

das como translação e rotação, que, por sua vez, podem ser parametrizadas de diversas formas. No restante desta seção, serão apresentados conceitos fundamentais de representação espacial e transformação de corpos rígidos com suporte em informações extraídas de [26].

2.1.1 Representação da rotação

Considerando um sistema de coordenadas global G e um sistema de coordenadas local L rotacionado em relação a G , um ponto \mathbf{p} , descrito em coordenadas locais como

$$\mathbf{p}_l = \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix}, \quad (2.1)$$

pode ser descrito em coordenadas globais a partir de uma rotação que relate os dois sistemas de coordenadas. Tal transformação é representada por uma matriz de rotação $\mathbf{R} \in SO(3)$ ¹. Assim, define-se a matriz de rotação como

$$\mathbf{R} = [\mathbf{r}_x \ \mathbf{r}_y \ \mathbf{r}_z], \quad (2.2)$$

em que \mathbf{r}_x , \mathbf{r}_y e \mathbf{r}_z são os vetores que descrevem cada um dos eixos cartesianos do sistema local com relação aos eixos do sistema global.

Assim, o ponto \mathbf{p} é descrito em coordenadas globais por

$$\mathbf{p}_g = \mathbf{R}_l^g \mathbf{p}_l, \quad (2.3)$$

em que \mathbf{R}_l^g representa a rotação do sistema L com relação ao sistema G dada por

$$\mathbf{R}_l^g = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix}, \quad (2.4)$$

em que, por fim, o termo r_{ij} representa a projeção do eixo j do sistema L sobre o eixo i do sistema G .

Uma matriz de rotação é dita elementar se a rotação for realizada em torno de um único eixo do sistema cartesiano. Considerando o caso de um robô que se movimenta apenas em uma superfície plana, ou seja, no plano xy , como no caso deste trabalho, sua rotação θ é representada por uma matriz de rotação elementar do eixo z dada por

$$\mathbf{R}_z = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.5)$$

¹Em inglês, Special Orthogonal Group, SO. Esse espaço compreende as matrizes ortogonais, aquelas cuja inversa se iguala à transposta, com a propriedade especial de possuírem determinante de valor igual a um.

2.1.2 Representação da translação

Para o caso da translação do sistema de coordenadas L com relação ao sistema G , a transformação $\mathbf{T}(t) \in \mathbb{R}^3, t \in [0, t]$, relaciona a descrição em coordenadas locais do ponto \mathbf{p} à descrição em coordenadas globais seguindo a relação

$$\mathbf{p}_g = \mathbf{p}_l + \mathbf{T}, \quad (2.6)$$

em que

$$\mathbf{T} = \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix}. \quad (2.7)$$

Considerando simultaneamente a translação e a rotação executadas sobre o sistema L , o ponto passa a ser descrito conforme

$$\mathbf{p}_g = \mathbf{R}_l^g \mathbf{p}_l + \mathbf{T}. \quad (2.8)$$

As matrizes de transformação da rotação e translação podem ser entendidas tanto como a transformação das coordenadas de um ponto como também a própria representação da orientação e posição de um sistema de coordenadas com relação a outro [26].

2.1.3 Representação homogênea

Uma forma de combinar o movimento de rotação e translação foi definida em 2.8. No entanto, uma representação mais compacta pode ser obtida utilizando a representação homogênea $\mathbf{H} \in SE(3)^2$. Esse espaço é originado do produto do espaço \mathbb{R}^3 , relativo à translação, e o espaço $SO(3)$, relativo à rotação. A praticidade de estarem os dois movimentos representados em uma única estrutura vem ao custo do aumento da dimensão da matriz para a quarta dimensão. Assim, os pontos passam a ser representados com um quarto elemento cujo valor atribuído é 1, conforme

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \quad (2.9)$$

Assim, pode-se definir a matriz homogênea H como

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0^{1x3} & 1 \end{bmatrix} \quad (2.10)$$

e a transformação do ponto \mathbf{p} entre coordenadas local e global é dada, finalmente, por

²Em inglês, Special Euclidean Group, SE.

$$\begin{aligned}\bar{\mathbf{p}}_g &= \mathbf{H} \bar{\mathbf{p}}_l \\ \begin{bmatrix} \mathbf{p}_g \\ 1 \end{bmatrix} &= \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0^{1x3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_l \\ 1 \end{bmatrix}\end{aligned}\tag{2.11}$$

em que $\bar{\mathbf{p}}_g$ e $\bar{\mathbf{p}}_l$ são as representações homogêneas do ponto \mathbf{p} .

2.2 Odometria

Quando um comando de atuação é executado, por conta dos ruídos dos próprios atuadores, há uma diferença entre o movimento comandado e o movimento efetuado. Assim, se um robô computasse sua posição final considerando apenas a sequência de passos comandados a partir de uma posição inicial conhecida, poderia ter uma crença errônea de seu real posicionamento [20]. A odometria é uma técnica de navegação inercial que se utiliza de dados provenientes dos sensores proprioceptivos, isto é, sensores que fornecem informações relativas ao estado interno do robô, para poder estimar sua posição e orientação com relação às condições iniciais conhecidas [3, 20, 27].

Os sistemas iniciais podem ser classificados em dois tipos: *Stable Platform Systems* e *Strapdown Systems* [27]. No primeiro sistema, os sensores são montados no dispositivo de forma que suas leituras sejam independentes dos movimentos do robô. Devido a esse isolamento, as informações obtidas por esses sensores são dadas diretamente em coordenadas globais. Já no segundo sistema, os sensores são rigidamente acoplados ao robô, reduzindo a complexidade mecânica de sua construção. As medidas passam, no entanto, a ser fornecidas com relação ao referencial local e isso gera, portanto, maior complexidade computacional para que elas possam ser transformadas em coordenadas globais.

A plataforma NAO utilizada neste trabalho contém um sistema de navegação inercial do tipo *Strapdown* formado por acelerômetro e giroscópio cujas leituras podem ser aplicadas ao modelo de movimento do robô para então determinar seu estado no momento da leitura. O giroscópio fornece a velocidade angular do corpo e dela pode-se extrair, por meio da integração dos sinais, a matriz de rotação do robô com relação ao referencial global. O acelerômetro, por sua vez, fornece a aceleração do corpo em coordenadas locais. Utilizando a matriz de rotação obtida anteriormente, tal aceleração pode ser calculada com relação às coordenadas globais e integrada uma vez para a obtenção da velocidade e mais uma vez para a posição do robô. Como os sinais dos sensores são lidos de forma discreta, um esquema de integração numérica deve ser definido. Para aplicações que não exigem alto grau de precisão, é possível aplicar um esquema de ordem baixa, como a regra dos retângulos [27].

No restante desta seção, serão demonstrados os cálculos necessários para obtenção da orientação e posição do robô a partir das leituras de seus sensores iniciais conforme visto em [27].

2.2.1 Cálculo da matriz de rotação

Como mencionado anteriormente, as informações dos sensores são dadas com relação ao referencial local, ou seja, o referencial do dispositivo. Com o uso do sinal de velocidade angular do giroscópio ω_l pode-se definir a orientação do robô com relação ao referencial global por meio de sua matriz de rotação. Seja \mathbf{R} essa matriz de rotação, a sua taxa de variação no instante t pode ser dada por

$$\dot{\mathbf{R}}(t) = \lim_{\delta t \rightarrow 0} \frac{\mathbf{R}(t + \delta t) - \mathbf{R}(t)}{\delta t}. \quad (2.12)$$

Pode-se reescrever o termo $\mathbf{R}(t + \delta t)$ como

$$\mathbf{R}(t + \delta t) = \mathbf{R}(t)\mathbf{A}(t), \quad (2.13)$$

em que $\mathbf{A}(t)$ representa a rotação desempenhada pelo sistema local entre os instantes t e $t + \delta t$. Caso os ângulos $\delta\phi$, $\delta\theta$, $\delta\psi$ das rotações realizadas ao redor dos eixos x , y e z , respectivamente, sejam pequenos, pode-se aplicar a aproximação para pequenos ângulos e considerar que seus valores são iguais aos seus senos e que seus cossenos valem um. Assim, tem-se que

$$\mathbf{A}(t) = \mathbf{I}\delta\Psi, \quad (2.14)$$

em que $\delta\Psi$ é dado por

$$\delta\Psi = \begin{pmatrix} 0 & -\delta\psi & \delta\theta \\ \delta\psi & 0 & -\delta\phi \\ -\delta\theta & \delta\phi & 0 \end{pmatrix}. \quad (2.15)$$

Substituindo (2.13) em (2.12), obtém-se

$$\dot{\mathbf{R}}(t) = \lim_{\delta t \rightarrow 0} \frac{\mathbf{R}(t)\mathbf{A}(t) - \mathbf{R}(t)}{\delta t}. \quad (2.16)$$

Desenvolvendo (2.16), tem-se

$$\dot{\mathbf{R}}(t) = \mathbf{R}(t) \lim_{\delta t \rightarrow 0} \frac{\delta\Psi}{\delta t}. \quad (2.17)$$

Definindo $\boldsymbol{\Omega}$ como a velocidade angular do sistema e aplicando a aproximação para ângulos pequenos, tem-se que

$$\boldsymbol{\Omega}(t) = \lim_{\delta t \rightarrow 0} \frac{\delta\Psi}{\delta t}$$

$$\boldsymbol{\Omega}(t) = \begin{pmatrix} 0 & -\omega_{l_z}(t) & \omega_{l_y}(t) \\ \omega_{l_z}(t) & 0 & -\omega_{l_x}(t) \\ -\omega_{l_y}(t) & \omega_{l_x}(t) & 0 \end{pmatrix}. \quad (2.18)$$

Assim, substituindo (2.18) em (2.17), obtém-se a equação diferencial ordinária (EDO) da matriz de rotação

$$\dot{\mathbf{R}}(t) = \mathbf{R}(t)\boldsymbol{\Omega}(t) \quad (2.19)$$

cuja solução é dada por

$$\mathbf{R}(t) = \mathbf{R}(0) \exp \int_0^t \boldsymbol{\Omega}(t) \delta t. \quad (2.20)$$

Para duas amostras no período entre t e $t + \delta t$, (2.20) pode ser reescrita como

$$\mathbf{R}(t + \delta t) = \mathbf{R}(t) \exp \int_t^{t+\delta t} \boldsymbol{\Omega}(t) \delta t. \quad (2.21)$$

Utilizando a regra dos retângulos na integração, tem-se que

$$\int_t^{t+\delta t} \boldsymbol{\Omega}(t) \delta t = \mathbf{B} = \begin{pmatrix} 0 & -\omega_{l_z} \delta t & \omega_{l_y} \delta t \\ \omega_{l_z} \delta t & 0 & -\omega_{l_y} \delta x \\ -\omega_{l_y} \delta t & \omega_{l_x} \delta t & 0 \end{pmatrix}. \quad (2.22)$$

Substituindo (2.22) em (2.21), obtém-se

$$\mathbf{R}(t + \delta t) = \mathbf{R}(t) \exp \mathbf{B}. \quad (2.23)$$

Aplicando a expansão de Taylor no termo $\exp \mathbf{B}$ em (2.23), tem-se

$$\mathbf{R}(t + \delta t) = \mathbf{R}(t) \left(\mathbf{I} + \mathbf{B} + \frac{\mathbf{B}^2}{2!} + \frac{\mathbf{B}^3}{3!} + \dots \right). \quad (2.24)$$

Ao se desenvolver \mathbf{B}^3 , encontra-se a relação

$$\mathbf{B}^3 = -\sigma^2 \mathbf{B}, \quad (2.25)$$

em que σ equivale a

$$\sigma = |\omega_l \delta t|. \quad (2.26)$$

Aplicando (2.25) em (2.24), obtém-se

$$\mathbf{R}(t + \delta t) = \mathbf{R}(t) \left(\mathbf{I} + \mathbf{B} + \frac{\mathbf{B}^2}{2!} - \frac{\sigma^2 \mathbf{B}}{3!} - \frac{\sigma^2 \mathbf{B}^2}{4!} + \dots \right). \quad (2.27)$$

Desenvolvendo (2.27), encontra-se a equação final para o cálculo da matriz de rotação do instante $t + \delta t$ a partir da matriz de rotação do instante t e das medidas do giroscópio, dada por

$$\mathbf{R}(t + \delta t) = \mathbf{R}(t) \left(\mathbf{I} + \frac{\sin \sigma}{\sigma} \mathbf{B} + \frac{1 - \cos \sigma}{\sigma^2} \mathbf{B}^2 \right). \quad (2.28)$$

2.2.2 Cálculo da posição

Utilizando a matriz de rotação $\mathbf{R}(t)$ encontrada anteriormente, a aceleração lida em coordenadas locais \mathbf{a}_l pode ser transformada em coordenadas globais \mathbf{a}_g conforme

$$\mathbf{a}_g(t) = \mathbf{R}(t)\mathbf{a}_l(t). \quad (2.29)$$

Uma vez que os sinais são convertidos às coordenadas globais, é necessário compensar a aceleração da gravidade ainda antes de integrar os sinais para obtenção da velocidade e da posição. Assim, as equações do movimento são dadas por

$$\begin{aligned} \mathbf{v}_g(t) &= \mathbf{v}_g(0) + \int_0^t (\mathbf{a}_g(t) - \mathbf{g}_g(t))\delta t \\ \mathbf{p}_g(t) &= \mathbf{p}_g(0) + \int_0^t \mathbf{v}_g(t)\delta t, \end{aligned} \quad (2.30)$$

em que \mathbf{v}_g , \mathbf{v}_l e \mathbf{s}_g representam, respectivamente, as velocidades global e local e a posição global do corpo.

Aplicando a regra dos retângulos na integração, as equações de movimento do robô para um período de amostragem δt são descritas por

$$\begin{aligned} \mathbf{v}_g(t + \delta t) &= \mathbf{v}_g(t) + \delta t \cdot (\mathbf{a}_g(t + \delta t) - \mathbf{g}_g) \\ \mathbf{p}_g(t + \delta t) &= \mathbf{p}_g(t) + \delta t \cdot \mathbf{v}_g(t + \delta t). \end{aligned} \quad (2.31)$$

2.2.3 Erros na odometria

O modelo encontrado é, no entanto, incompleto, pois são desconsiderados eventos físicos tais como irregularidade da superfície, inclinações, variação do ponto de contato do robô com o solo, escorregamento e até mesmo a resolução limitada e ruído dos sensores, além das aproximações realizadas no desenvolvimento do modelo, como a aproximação dos pequenos ângulos para obtenção da velocidade angular e a regra dos retângulos na integração. Todos esses pontos que não são considerados no modelo utilizado acarretam em erro entre a posição calculada e o movimento executado. Como visto em 2.30, as medidas são integradas e os erros, portanto, também o são. Isso mostra a necessidade de conduzir-se uma atualização da crença da posição do robô por meio de múltiplas leituras, o que pode ser realizado incluindo no cálculo da posição do robô informações obtidas de outros sensores, como a câmera [1, 20].

2.3 Representação da imagem

A câmera é um dispositivo óptico capaz de capturar uma imagem em seu material fotossensível. Um modelo simplificado da câmera, o modelo *pinhole*³, a considera como um equipamento

³Em português, câmera estenopeica.

contendo uma barreira à passagem de luz com uma única abertura por onde a luz do ambiente pode passar, gerando uma imagem invertida do objeto, conforme apresentado na Figura 2.1. Esse sistema limita o número de raios de luz que são capazes de atingir o plano da imagem e influenciar, portanto, na formação da imagem. Dessa forma, há uma projeção de um ponto no espaço em um único ponto na imagem. A lente encontrada em câmeras reais permite que mais raios de luz passem por sua abertura gerando uma imagem mais clara e nítida, apesar de exigir uma nova modelagem que inclua a distorção causada. No entanto, o modelo *pinhole*, apesar de simples, apresenta aproximações suficientemente adequadas para a aplicação desejada, sendo, portanto, adotado para este trabalho por sua simplicidade e razoável representação [28]. O modelo *pinhole* será melhor desenvolvido a seguir.

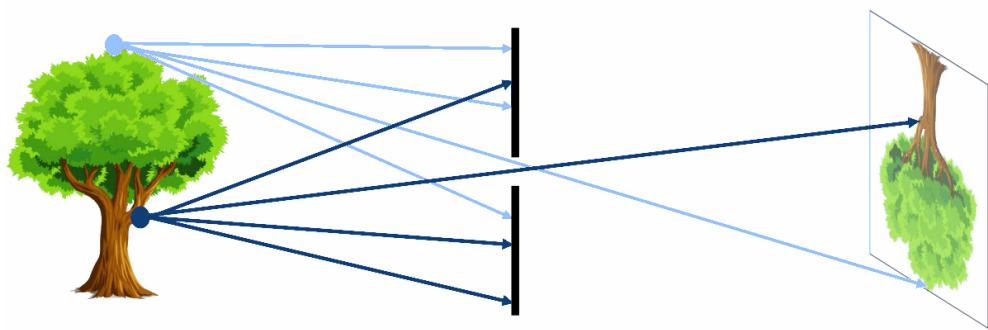


Figura 2.1: Representação do funcionamento de uma câmera no modelo *pinhole*. Fonte:
Adaptado de HATA e SAVARESE

O plano da imagem π é o plano no qual os objetos são projetados. O orifício por onde os raios de luz podem passar é denominado centro óptico. Nesse ponto, é definida a origem do sistema de coordenadas da câmera O^c cujo eixo z^c representa o eixo óptico, isto é, o eixo que passa perpendicularmente ao plano da imagem. A distância entre o plano π e a origem O^c é denominada distância focal f e é um parâmetro intrínseco do dispositivo. Para simplificar o modelo, pode-se considerar um plano virtual, também perpendicular ao eixo óptico e localizado a uma distância f do centro óptico, mas em sentido contrário ao plano da imagem. Assim, nesse novo plano, a imagem gerada mantém a mesma orientação do objeto no espaço. Esses parâmetros descritivos do modelo são indicados na Figura 2.2.

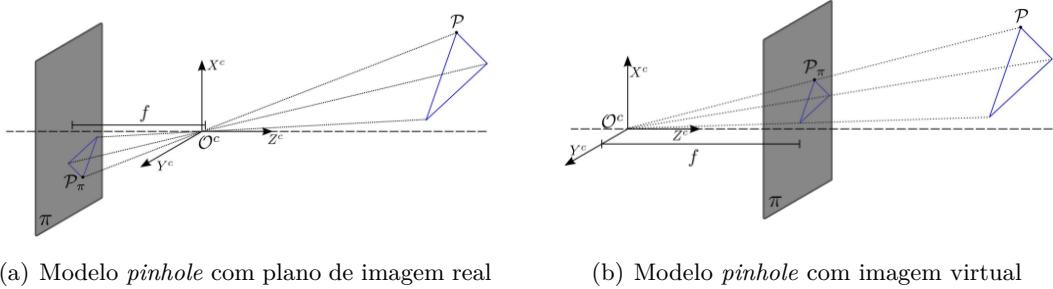


Figura 2.2: Modelo *pinhole* da câmera. Fonte: BERNARDES, 2009, p. 22

O ponto \mathbf{p} no espaço deve ser relacionado à sua representação na imagem para a obtenção de um modelo de projeção dos objetos no mundo real em uma imagem. Esses pontos podem ser descritos com relação ao sistema de coordenadas da câmera, assim, o ponto no espaço é dado por $\mathbf{p} = (x, y, z)^T$ e o ponto no plano da imagem é dado por $\mathbf{p}^\pi = (x^\pi, y^\pi, f)^T$. Por semelhança de triângulos, pode-se obter uma relação entre as coordenadas x^π e y^π e as coordenadas de \mathbf{p} . Assim, \mathbf{p}^π pode ser definido como

$$\mathbf{p}^\pi = \begin{pmatrix} xf \\ zf \\ yf \\ zf \\ f \end{pmatrix} = f \begin{pmatrix} x \\ z \\ y \\ z \\ 1 \end{pmatrix} = \frac{f}{z} \mathbf{p}. \quad (2.32)$$

A Figura 2.3 representa a projeção de um ponto \mathbf{p} na imagem detalhando um novo sistema de referência localizado na imagem, O^i , e os pixels que a formam.

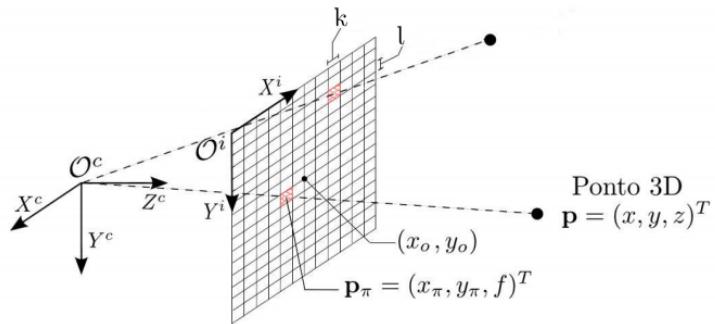


Figura 2.3: Projeção de ponto na imagem seguindo modelo *pinhole*. Fonte: Adaptado de BERNARDES, 2009, p. 22

Para transformar o ponto \mathbf{p}^π entre o sistema de coordenadas da câmera e da imagem, são necessárias uma rotação e uma translação aplicadas seguindo a relação

$$\mathbf{p}^i = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{p}^\pi + \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}, \quad (2.33)$$

em que \mathbf{p}^i representa o ponto referente ao sistema de coordenadas da imagem e x_0 e y_0 são as coordenadas do ponto da imagem no centro óptico, ou seja, do centro da imagem.

Ainda, é necessário ajustar as unidades de medida do modelo, pois o modelo até aqui derivado utiliza medidas métricas, mas, em uma imagem digital, um ponto é representado por pixels. Os pixels são geralmente retangulares e têm, portanto, um parâmetro escalar para cada um de seus lados: k e l , para o eixo x e y respectivamente [29]. Assim, para a adequação da unidade de medida, tem-se a inclusão da matriz de escala ao modelo de forma que

$$\mathbf{p}^i = \begin{pmatrix} k & 0 & 0 \\ 0 & l & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{p}^\pi + \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix}. \quad (2.34)$$

Unificando as matrizes de transformação e aplicando a relação definida na equação 2.32, obtém-se o modelo final de projeção da imagem com relação aos parâmetros intrínsecos da câmera dado por

$$\mathbf{p}^i = \mathbf{K} \frac{1}{z} \mathbf{p}, \quad (2.35)$$

em que \mathbf{K} é a matriz de calibração. Ela é definida como

$$\begin{pmatrix} -kf & 0 & x_0 \\ 0 & lf & y_0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (2.36)$$

em que f é a distância focal da câmera, k é o fator de escala da imagem no eixo x , l é o fator de escala no eixo y , e, finalmente, x_0 e y_0 são as coordenadas do centro óptico.

2.4 Mapa

No problema de localização, é necessário que o robô tenha um mapa do ambiente para que possa identificar o seu posicionamento a partir das leituras de seus sensores. A natureza do mapa depende das tarefas a serem desempenhadas e das limitações físicas do robô, ou seja, o ambiente no qual o robô está inserido, quais leituras são possíveis a partir de seus sensores e sua capacidade de processamento e de armazenamento de dados [30]. Tal conjuntura define como o robô codifica e armazena as informações extraídas do ambiente a fim de formar sua representação do mundo real [3]. As formas de representação do ambiente podem ser divididas em dois tipos de mapas: topológico e métrico [31].

O mapa topológico se baseia em pontos de referência do ambiente, que devem ser características visíveis cuja posição com relação ao referencial adotado é conhecida [2]. O objetivo desse mapa é mostrar a conexão entre distintos pontos de referência. Dessa forma, é comumente representado em formato de grafos e aplicado a problemas de navegação e também a problemas de localização interna, pois há possibilidade de se determinar padrões. O mapa topológico depende também da unicidade das características identificadas, tornando seu processamento complexo caso seus nós não sejam únicos [3].

Já o mapa métrico fornece medidas de representações geométricas com relação a um sistema de coordenadas global. É capaz de fornecer informações mais completas, pois a partir da caracterização do ambiente podem-se extrair informações como forma e tamanho dos objetos nela representados [30]. Assim, pontos de referência que poderiam ser interpretados de forma ambígua com a representação anterior passam a ser únicos com o mapa do tipo métrico. É uma representação potencialmente acurada, mas como o armazenamento total é proporcional à densidade de objetos no ambiente, tem um custo computacional alto. Assim, além da discretização do espaço representado, a escolha de características de relevante utilidade à tarefa a ser realizada permite a criação de uma representação mínima do mundo, propiciando a solução do problema de custo computacional [3].

Alguns pontos devem ser considerados na elaboração de um mapa, tais como precisão, tipo de características e complexidade. A precisão com que o robô realiza uma tarefa está relacionada à precisão de sua representação do mundo e também à sua capacidade de extrair do ambiente os pontos de referência indicados no mapa [20]. Além disso, em caso de necessidade de atualização do mapa, o robô deve manter seu mapa acurado e atualizado e efetuar operações com relação a este mapa [3]. Um modelo complexo pode exigir maior complexidade dos cálculos e trazer maiores custos processuais [20]. É também importante delimitar no mapa pontos de referência que o robô seja capaz de extrair do ambiente.

2.5 Estimativa de estado

O ambiente no qual o robô atua pode ser caracterizado por um estado. Da interação do robô com o ambiente podem ser discriminadas tanto as medidas dos sensores, que são as observações ruidosas que o robô consegue extrair do ambiente, quanto as ações de controle, que são os comandos para atuação do robô no ambiente. As informações de que o robô necessita para atuar no ambiente não podem ser diretamente extraídas dele, mas apenas inferidas dos dados dos sensores [32].

A estimativa de um estado a partir de tais dados e sua representação se tornam uma questão importante. O ambiente dinâmico, ambíguo e imprevisível, junto às limitações dos equipamentos, leva a incertezas na modelagem do sistema que podem ser representadas com uma abordagem probabilística. Ao adotar uma distribuição de probabilidade para representar a informação do ambiente, todas as hipóteses possíveis dentro do espaço amostral são consideradas. Essa abordagem torna o sistema mais robusto, pois permite lidar com hipóteses múltiplas e conflituosas que naturalmente surgem nesse tipo de ambiente [32].

Dessa forma, o estado, as medidas e as ações de controle podem ser representados por variáveis aleatórias. Em espaços contínuos, essas variáveis assumem uma função densidade de probabilidade (FDP). Para uma variável vetorial, a probabilidade $p(\mathbf{x})$ de ocorrência de um evento é dada pela distribuição multivariada descrita como

$$p(\mathbf{x}) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}, \quad (2.37)$$

em que $\boldsymbol{\mu}$ é o vetor média e Σ é a matriz de covariância, conforme visto em [32]. Toda as demais derivações desta seção também foram extraídas de [32].

A distribuição conjunta de duas variáveis pode ser calculada considerando as relações de dependência ou independência entre elas. O teorema de Bayes demonstra uma forma de calcular a probabilidade condicional $p(\mathbf{x} | \mathbf{z})$ a partir de sua inversa, $p(\mathbf{z} | \mathbf{x})$, conforme

$$p(\mathbf{x} | \mathbf{z}) = \frac{p(\mathbf{z} | \mathbf{x})p(\mathbf{x})}{p(\mathbf{z})}. \quad (2.38)$$

A equação definida em 2.38 pode ser entendida como o cálculo da probabilidade de ocorrência do estado \mathbf{x} dada a medida \mathbf{z} , ou seja, a própria estimativa de estado a partir das leituras dos

sensores do robô. No entanto, isso é feito utilizando a probabilidade de se obter a medida \mathbf{z} em um ambiente de configuração \mathbf{x} . Ainda, $p(\mathbf{x})$ representa o conhecimento a priori que se tem desse estado, antes de serem incorporados os dados dos sensores. Por fim, $p(\mathbf{z})$ é a probabilidade de ocorrência da medida \mathbf{z} e independe do valor de \mathbf{x} . Para valores discretos, o termo $\frac{1}{p(\mathbf{z})}$ pode ser entendido como um fator normalizador η , mais facilmente calculado por

$$\eta = \sum_{\mathbf{x}} p(\mathbf{z} | \mathbf{x})p(\mathbf{x}). \quad (2.39)$$

Para descrever o sistema estocástico, podem ser utilizados modelos generativos dos estados e medidas, isto é, modelos que mostrem como tais variáveis são geradas. Considerando que o estado do ambiente seja completo, ou seja, suficiente para descrever o resultado de todas as operações realizadas até o tempo t , a probabilidade de transição de estados depende apenas do estado anterior e da última ação de controle tomada. Já a medida no tempo t depende apenas das condições do ambiente nesse mesmo instante de tempo. Assim, os processos são regidos pela probabilidade de transição de estado e pela probabilidade de medida dos sensores dadas por

$$\begin{aligned} & p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t) \\ & p(\mathbf{z}_t | \mathbf{x}_t), \end{aligned} \quad (2.40)$$

em que \mathbf{x}_t é o estado no tempo t , \mathbf{x}_{t-1} é o estado anterior, \mathbf{u}_t é a ação de controle e \mathbf{z}_t a medida dos sensores no tempo t .

Como o robô é incapaz de saber o real estado do ambiente e pode apenas inferir dos dados dos sensores, o conhecimento que ele tem sobre o estado é denominado crença. Os modelos generativos apresentados em 2.40 são usados como base para a representação dessa crença. A crença que o robô tem, por exemplo, sobre sua posição no espaço, é obtida em duas etapas a partir do cálculo das distribuições de probabilidade condicional. Primeiramente, é obtida a estimativa por predição, indicada por $\bar{bel}(\mathbf{x})$, a partir da probabilidade de transição de estado. Depois, é calculada a correção da estimativa, $bel(\mathbf{x})$, utilizando a probabilidade de medida dos sensores. Aplicando a essas formulações o teorema de Bayes apresentado em 2.38, obtém-se as equações do Filtro de Bayes

$$\begin{aligned} \bar{bel}(\mathbf{x}_t) &= \int p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t) bel(\mathbf{x}_{t-1}) d\mathbf{x} \\ bel(\mathbf{x}_t) &= \eta p(\mathbf{z}_t | \mathbf{x}_t) \bar{bel}(\mathbf{x}_t). \end{aligned} \quad (2.41)$$

Esse filtro funciona de forma recursiva, utilizando informações do instante de tempo atual e a última estimativa, calculada no instante anterior. Por fim, para completar o modelo e o Filtro de Bayes, deve-se definir a distribuição do estado inicial. Assim, para um estado inicial desconhecido, a crença $bel(\mathbf{x}_0)$ é definida com uma distribuição uniforme e, para um estado inicial conhecido, a crença assume probabilidade igual a um em tal valor e zero em todos os demais valores.

O Filtro de Bayes foi derivado com a suposição de que o estado seja completo, no entanto, por conta de fenômenos não modelados e erros de aproximação, por exemplo, esse princípio é violado. Ainda assim, o filtro bayesiano se mostra robusto caso o estado seja definido de forma que as violações tenham efeito quase aleatório. Existem diversos algoritmos baseados no filtro de Bayes, com suposições específicas e, portanto, modelos distintos para a estimativa. A escolha do tipo de filtro leva em conta a eficiência computacional, a acurácia da aproximação e a facilidade da implementação do algoritmo [32]. No restante desta seção, serão exploradas duas formas de implementação desse filtro utilizando distribuições gaussianas.

2.5.1 Filtro de Kalman

Uma forma de implementar o filtro bayesiano é utilizando a representação gaussiana, ou seja, uma distribuição normal. Ela pode ser caracterizada pelo seu primeiro e segundo momentos, a média μ e a covariância Σ , e os demais momentos assumem valor zero. O uso de uma função unimodal leva à uma estimativa próxima do real estado do sistema.

O Filtro de Kalman foi originalmente desenvolvido para processos lineares contínuos. Para sua aplicação, assume-se que a crença inicial deve ser normalmente distribuída, seguindo a formulação em 2.37. Ainda, o estado \mathbf{x} e a medida \mathbf{z} devem ser lineares com ruído gaussiano de média zero. Ambos são representados por

$$\begin{aligned}\mathbf{x}_t &= \mathbf{A}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \varepsilon_t \\ \mathbf{z}_t &= \mathbf{C}_t \mathbf{x}_t + \delta_t,\end{aligned}\tag{2.42}$$

em que \mathbf{A} é a matriz de transição de estado, \mathbf{B} é a matriz das entradas de controle e \mathbf{C} é a matriz de medida e permitem a linearização das funções. Os termos ε_t e δ_t representam os ruídos associados ao estado e à medida.

As três restrições mencionadas acima garantem que a crença posterior calculada seja também gaussiana seguindo o mesmo formato da distribuição inicial. Aplicando 2.42 em 2.37, podem-se definir as probabilidade de transição de estado e de medida como

$$\begin{aligned}p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t) &= \det(2\pi\mathbf{Q}_t)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\mathbf{x}_t - \mathbf{A}_t \mathbf{x}_{t-1} - \mathbf{B}_t \mathbf{u}_t)^T \mathbf{Q}_t^{-1} (\mathbf{x}_t - \mathbf{A}_t \mathbf{x}_{t-1} - \mathbf{B}_t \mathbf{u}_t) \right\} \\ p(\mathbf{z}_t | \mathbf{x}_t) &= \det(2\pi\mathbf{R}_t)^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (\mathbf{z}_t - \mathbf{C}_t \mathbf{x}_t)^T \mathbf{R}_t^{-1} (\mathbf{z}_t - \mathbf{C}_t \mathbf{x}_t) \right\},\end{aligned}\tag{2.43}$$

em que \mathbf{R}_t e \mathbf{Q}_t representam as matrizes de covariância e os termos $\mathbf{A}_t \mathbf{x}_{t-1} - \mathbf{B}_t \mathbf{u}_t$ e $\mathbf{C}_t \mathbf{x}_t$ representam as médias.

O funcionamento do filtro se dá em duas etapas, predição e correção, da mesma maneira que o filtro de Bayes. A diferença se dá nos termos que são calculados a fim de representar a estimativa do estado. Com o Filtro de Kalman, calculam-se a média $\bar{\mathbf{x}}$ e covariância $\bar{\mathbf{P}}$ na etapa de predição seguindo

$$\begin{aligned}\bar{\mathbf{x}}_t &= \mathbf{A}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t \\ \bar{\mathbf{P}}_t &= \mathbf{A}_t \mathbf{P}_{t-1} \mathbf{A}_t^T + \mathbf{Q}_t.\end{aligned}\tag{2.44}$$

Já a etapa de correção compreende o cálculo do ganho de Kalman e da média e da covariância da correção.

$$\begin{aligned}\mathbf{K}_t &= \mathbf{P}_t \mathbf{C}_t^T (\mathbf{C}_t \bar{\mathbf{P}}_t \mathbf{C}_t^T + \mathbf{R}_t)^{-1} \\ \mathbf{x}_t &= \bar{\mathbf{x}}_t + \mathbf{K}_t (\mathbf{z}_t - \mathbf{C}_t \bar{\mathbf{x}}_t) \\ \mathbf{P}_t &= (\mathbf{I} - \mathbf{K}_t \mathbf{C}_t) \bar{\mathbf{P}}_t,\end{aligned}\tag{2.45}$$

em que \mathbf{K}_t representa o ganho de Kalman e é utilizado para ajustar o valor da média \mathbf{x} , regulando o peso que as novas medidas tem sobre a estimativa.

2.5.2 Filtro de Kalman Estendido

O requisito explorado anteriormente de que o sistema fosse regido por equações lineares é dificilmente cumprido em aplicações reais. Para transpassar as limitações da aplicabilidade do Filtro de Kalman, o estado e a medida podem ser modelados por funções não-lineares em uma nova abordagem denominada Filtro de Kalman Estendido. Dessa forma, o estado e a medida passam a ser descritos por

$$\begin{aligned}\mathbf{x}_t &= f(\mathbf{x}_{t-1}, \mathbf{u}_t) + \varepsilon_t \\ \mathbf{z}_t &= h(\mathbf{x}_t) + \delta_t,\end{aligned}\tag{2.46}$$

em que f e h representam funções não-lineares e ε_t e δ_t são os ruídos associados às variáveis.

O Filtro de Kalman Estendido exige a aplicação de aproximações visto que não há uma solução em forma fechada para 2.46. Assim, a linearização é feita por meio da expansão de Taylor de primeira ordem. Os pontos de linearização para cada uma das funções podem ser escolhidos a partir do estado mais provável no instante t de linearização, aproximado pelo valor da média na última iteração $\boldsymbol{\mu}_{t-1}$ para a função f e pelo valor $\bar{\boldsymbol{\mu}}_{t-1}$ para a função h . Assim, pela expansão de Taylor de primeira ordem, encontram-se as linearizações

$$\begin{aligned}f(\mathbf{u}_t, \mathbf{x}_{t-1}) &\approx f(\mathbf{u}_t, \boldsymbol{\mu}_{t-1}) + g'(\mathbf{u}_t, \boldsymbol{\mu}_{t-1})(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) \\ h(\mathbf{x}_t) &\approx h(\bar{\boldsymbol{\mu}}_t) + h'(\bar{\boldsymbol{\mu}}_t)(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t),\end{aligned}\tag{2.47}$$

em que os termos $g'(\mathbf{u}_t, \boldsymbol{\mu}_{t-1})$ e $h'(\bar{\boldsymbol{\mu}}_t)$ podem ser representados pelas matrizes Jacobianas \mathbf{F}_t e \mathbf{H}_t . Fazendo as substituições, obtém-se, por fim,

$$\begin{aligned}f(\mathbf{u}_t, \mathbf{x}_{t-1}) &= f(\mathbf{u}_t, \boldsymbol{\mu}_{t-1}) + \mathbf{F}_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) \\ h(\mathbf{x}_t) &= h(\bar{\boldsymbol{\mu}}_t) + \mathbf{H}_t(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t).\end{aligned}\tag{2.48}$$

Assim, as probabilidades de transição de estado e de medida pelos sensores são

$$\begin{aligned}
p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1}) &= \det(2\pi\mathbf{R}_t)^{-\frac{1}{2}} \\
&\exp \left\{ -\frac{1}{2}(\mathbf{x}_t - f(\mathbf{u}_t, \boldsymbol{\mu}_{t-1}) - \mathbf{F}_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}))^T \mathbf{R}_t^{-1} (\mathbf{x}_t - f(\mathbf{u}_t, \boldsymbol{\mu}_{t-1}) - \mathbf{F}_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})) \right\} \\
p(\mathbf{z}_t \mid \mathbf{x}_t) &= \det(2\pi\mathbf{Q}_t)^{-\frac{1}{2}} \\
&\exp \left\{ -\frac{1}{2}(\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t) - \mathbf{H}(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t))^T \mathbf{Q}_t^{-1} (\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t) - \mathbf{H}(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t)) \right\}.
\end{aligned} \tag{2.49}$$

As etapas de predição e correção mantém funcionamento análogo àquele verificado no Filtro de Kalman clássico. As equações de predição ajustadas para as novas suposições da abordagem estendida são

$$\begin{aligned}
\bar{\mathbf{x}}_t &= f(\mathbf{u}_t, \mathbf{x}_{t-1}) \\
\bar{\mathbf{P}}_t &= \mathbf{F}_t \mathbf{P}_{t-1} \mathbf{F}_t^T + \mathbf{Q}_t.
\end{aligned} \tag{2.50}$$

Por fim, as equações da correção são dadas por

$$\begin{aligned}
\mathbf{K}_t &= \mathbf{P}_t \mathbf{H}_t^T (\mathbf{H}_t \bar{\mathbf{P}}_t \mathbf{H}_t^T + \mathbf{R}_t)^{-1} \\
\mathbf{x}_t &= \bar{\mathbf{x}}_t + \mathbf{K}_t (\mathbf{z}_t - h(\bar{\mathbf{x}}_t)) \\
\mathbf{P}_t &= (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \bar{\mathbf{P}}_t.
\end{aligned} \tag{2.51}$$

O Filtro de Kalman Estendido é eficiente computacionalmente, tem implementação simples e se mostra um algoritmo robusto para estimativa de estados. Como é feita uma aproximação por linearização, a acurácia do resultado depende do grau de não linearidade da função a que é aplicada e também o grau de incerteza.

Capítulo 3

Desenvolvimento

Neste capítulo é descrita a metodologia adotada no desenvolvimento deste trabalho. São apresentadas a configuração do ambiente de teste e sua representação, bem como as adaptações realizadas a fim de permitir a simulação de um campo de futebol. Também são explicados os procedimentos para leitura dos dados dos sensores e seu processamento para aquisição de informação significativa. Por fim, é detalhado o projeto do Filtro de Kalman e sua implementação.

3.1 Introdução

Durante a partida de futebol, o robô se movimenta pelo campo ao mesmo tempo em que extrai informações do mundo. Conforme mencionado em seções anteriores, as informações relevantes para que o robô se localize no campo são extraídas das imagens capturadas pelas câmeras e de seus sensores iniciais. O modelo para esse sistema envolve não-linearidades. Portanto, adotou-se o filtro de Kalman Estendido uma vez que ele é capaz de linearizar as equações do modelo localmente. Nas próximas seções, serão detalhados os métodos desenvolvidos para a implementação do sistema de localização para o robô NAO.

3.2 Mapa

Para poder fazer a associação entre os seus dados internos e o posicionamento no mundo real, o robô precisa de uma representação do ambiente em que está atuando. Representar todos os elementos do campo de futebol de forma contínua seria inviável tanto com relação ao armazenamento quanto à complexidade da computação. Portanto, é importante definir a representação mínima suficiente para que o robô efetue a localização.

Os elementos característicos do ambiente são denominados *features*. As *features* escolhidas devem ser suficientemente informativas e, ao mesmo tempo, devem ser detectáveis pelo robô. As

interseções das linhas de marcação são elementos significativos nesse contexto, pois podem indicar posicionamento específico no campo. Os pontos onde ocorrem interseções das marcações do campo são apresentados na Figura 3.1. A detecção de intersecções pode ser feita aplicando técnicas de visão computacional às imagens do robô. Por último, a baixa densidade de interseções no ambiente torna viável computacionalmente a utilização de um mapa métrico.

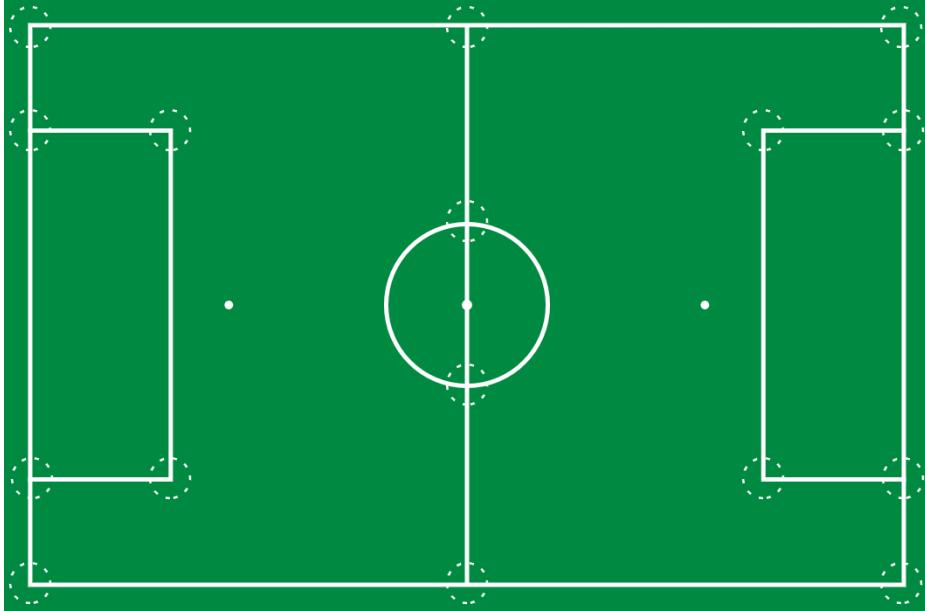


Figura 3.1: Interseções das linhas de marcação do campo indicadas pelos círculos tracejados.

Fonte: Autora

O mapa pode ser armazenado dentro do robô por meio de chaves, que representam cada uma das interseções, associadas a pares ordenados, que representam suas respectivas posições com relação a um referencial especificado. É necessário, então, definir a origem do sistema de coordenadas global. Neste trabalho, ela foi posicionada no escanteio inferior direito, no lado coincidente com a defesa.

A representação do mapa dessa maneira permite que o sistema seja avaliado com diferentes conjuntos de *landmarks* de forma a trabalhar a localização com informações parciais. Além disso, podem-se ajustar os valores de posicionamento de acordo com o redimensionamento que o campo possa sofrer.

3.3 Detecção de *landmarks*

No cenário de uma partida, utilizando o código de competição da equipe UnBeatables, a detecção de objetos na imagem é realizada por um módulo dedicado. Por meio de visão computacional, são registrados na memória compartilhada do *framework* o tipo de *feature* detectada e seu posicionamento na imagem. Assim, o módulo de localização pode acessar essas informações e utilizá-las como medição dentro do filtro. Esse sistema está sendo desenvolvido concomitantemente com este

trabalho como outro projeto de graduação. Portanto, no futuro, ambos trabalhos poderão ser acoplados para utilização em uma partida real.

Dada a indisponibilidade dos resultados do módulo de visão no momento de validação do presente trabalho, buscou-se uma alternativa ao reconhecimento das *features* do campo de futebol. Optou-se pela construção de um ambiente de validação substituindo as *features* por *landmarks* com padrões específicos de fácil reconhecimento. O *QR Code* é um tipo de *landmark* comumente utilizada em aplicações que exigem navegação em ambientes internos. No entanto, existem padrões de *landmark* criados especificamente para o uso com o robô NAO. Esses padrões são chamados de Naomark e são exemplificados na Figura 3.2. Dessa forma, uma implementação mais simples, capaz de simular o resultado do trabalho de visão computacional em progresso, pode ser realizada a fim de validar os resultados do filtro.

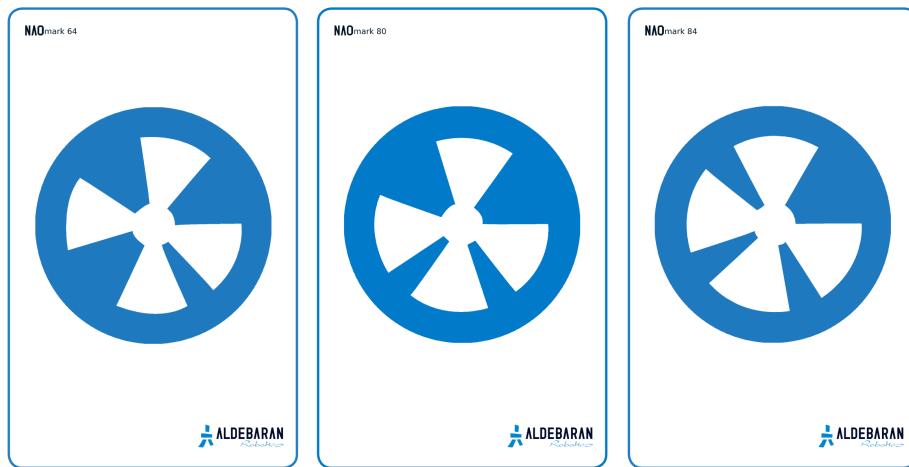


Figura 3.2: Exemplos de padrões de Naomarks. Fonte: *Softbank Robotics*

Os métodos de detecção das Naomarks são aplicados utilizando a API *ALLandMarkDetection*. As informações fornecidas por esse módulo são a marcação de tempo, o número de identificação da Naomark e o seu posicionamento descrito em parâmetros angulares.

3.4 Acesso à memória

A NAOqi conta com uma memória centralizada em que são armazenadas as informações relativas aos sensores e atuadores do robô. No caso de execução de aplicações orientadas a eventos, a memória pode ser utilizada também para salvar variáveis nomeadas. Tanto as informações da detecção das Naomarks quanto os dados dos sensores iniciais são acessados por essa memória.

A unidade inercial do robô NAO é composta por giroscópio e acelerômetro que fornecem, respectivamente, velocidade angular e aceleração, decompostas em cada uns dos três eixos cartesianos. A cada um dos sensores é atribuída uma chave identificadora. A leitura dos sensores é feita utilizando métodos da API *ALMemory* associados à chave específica de cada sensor. Assim, podem-se obter todos os valores relativos à aceleração linear e à velocidade angular nos eixos *x*, *y*

e z .

As informações da API *ALLAndMArkDetection* são publicadas e registradas em variável customizada da memória. O acesso a elas é feito diferenciadamente por se tratar de um módulo cujo controle é efetuado por eventos. Para a leitura desses dados, cria-se uma função *callback* que se subscreve a esse evento e recupera as informações da memória.

3.5 Dados dos sensores inerciais

Uma vez obtidas as leituras dos sensores inerciais, pode-se estimar a pose do robô. As formulações da odometria foram descritas detalhadamente na Seção 2.2. A velocidade angular é aplicada na Equação 2.28 para cômputo da matriz de rotação. Em seguida, a matriz de rotação é utilizada para transformar de coordenadas locais para coordenadas globais os valores lidos do acelerômetro, conforme Equação 2.29. Por fim, a aceleração global é utilizada para obtenção da velocidade e, consequintemente, da posição conforme as Equações 2.31.

3.5.1 Erros do giroscópio e acelerômetro

As causas de erros em giroscópios e acelerômetros envolvem erros sistemáticos, efeitos da temperatura e ruídos diversos [27]. Como os erros são integrados juntos às medições, o resultado da odometria apresenta uma grande divergência da realidade.

Dentre os erros citados, um dos mais significativos é o *bias*. O *bias* é a saída inesperada que o sensor fornece mesmo com o robô em completo repouso. O *bias* no giroscópio tem o efeito de acúmulo linear de erro na orientação, já o *bias* do acelerômetro causa um erro quadrático na posição. O *bias* é um erro sistemático que pode ser estimado por meio de calibração.

Para a calibração, as saídas dos sensores são lidas por um longo período de teste e a média desses valores determina o valor do *bias*. Para calibrar o giroscópio, é suficiente que o dispositivo permaneça estacionário independente de sua inclinação. Porém, como o acelerômetro sofre influência da aceleração da gravidade, a calibração requer que o dispositivo seja posicionado em uma superfície totalmente plana. Do contrário, as componentes da aceleração gravitacional em outros eixos podem ser medidas e podem afetar o *bias*. Para realizar a calibração do acelerômetro com sucesso, seria necessário o uso de uma plataforma que permitisse completo controle da inclinação [27, 33]. No entanto, para este trabalho, simplificou-se o processo e adotou-se como orientação neutra a posição inicial de movimentação do robô, apresentada na Figura 3.3. O robô, parado em sua posição inicial, foi submetido a esse experimento pelo período de cinco minutos. Foram encontrados o erro dos sensores e o desvio padrão associado, dados por

$$\begin{aligned}
\mathbf{a}_{bias} &= [0.4656 \quad 0.0783 \quad -10.6874]^T \\
\mathbf{a}_\sigma &= [0.0265 \quad 0.01567 \quad 0.0253]^T \\
\boldsymbol{\omega}_{bias} &= [3.8416 \times 10^{-5} \quad -6.7597 \times 10^{-4} \quad 2.9065 \times 10^{-5}]^T \\
\boldsymbol{\omega}_\sigma &= [0.0015 \quad 0.0031 \quad 0.0019]^T.
\end{aligned} \tag{3.1}$$



Figura 3.3: Posição inicial de movimentação do robô NAO utilizada para a calibração dos sensores inerciais. Fonte: Autora

Em seguida, foi implementada a compensação do *bias* do acelerômetro, \mathbf{a}_{bias} , e giroscópio, $\boldsymbol{\omega}_{bias}$ nas leitura dos sensores inerciais. Observou-se, no entanto, que, a cada reinicialização do robô, o valor compensado já não condizia mais com o real comportamento dos sensores. Para atenuar esse efeito, foi implementada uma rotina de calibração executada no momento de inicialização do robô.

Posteriormente, foram realizados testes iniciais para avaliação do resultado obtido por odometria. Foram observadas as saídas para o cômputo da odometria em três situações: robô parado; robô sendo girado manualmente de forma a tentar minimizar o impacto nas medições do acelerômetro; e finalmente com a movimentação do robô em linha reta. Obtiveram-se bons resultados para o cálculo da orientação do robô a partir dos dados do giroscópio. No entanto, os valores obtidos para o cálculo de posição divergiram rapidamente da realidade, alcançando a ordem de centenas de metros em apenas alguns segundos. As leituras do acelerômetro, mesmo após compensação do *bias*, apresentaram valores muito altos incongruentes com o modelo físico do robô.

A calibração dos sensores, realizada em condições estacionárias, é incapaz de amenizar os demais erros provenientes da dinâmica do sistema. O comportamento dinâmico do sistema pode ser melhor observado se as frequências de amostragem forem apropriadamente altas. Os sinais

são de alta frequência e, para acompanhar as alterações dos dados, o período de amostragem deve ser pequeno [33]. Em suma, é possível atenuar alguns dos erros associados às medições, mas o acúmulo de erros é inerente a esse tipo de estimativa. Por conta disso, emprega-se o Filtro de Kalman para a obtenção de estimativas mais acuradas.

Todo o processo de tratamento dos dados da unidade de medidas iniciais para obtenção da estimativa de posição e orientação está representado no diagrama da Figura 3.4. Vale enfatizar que esse diagrama representa o fluxo de operações para uma odometria inercial pura, que não é a abordagem final deste trabalho. Os dados obtidos são, na verdade, incorporados ao filtro e a implementação final difere da ilustrada no diagrama.

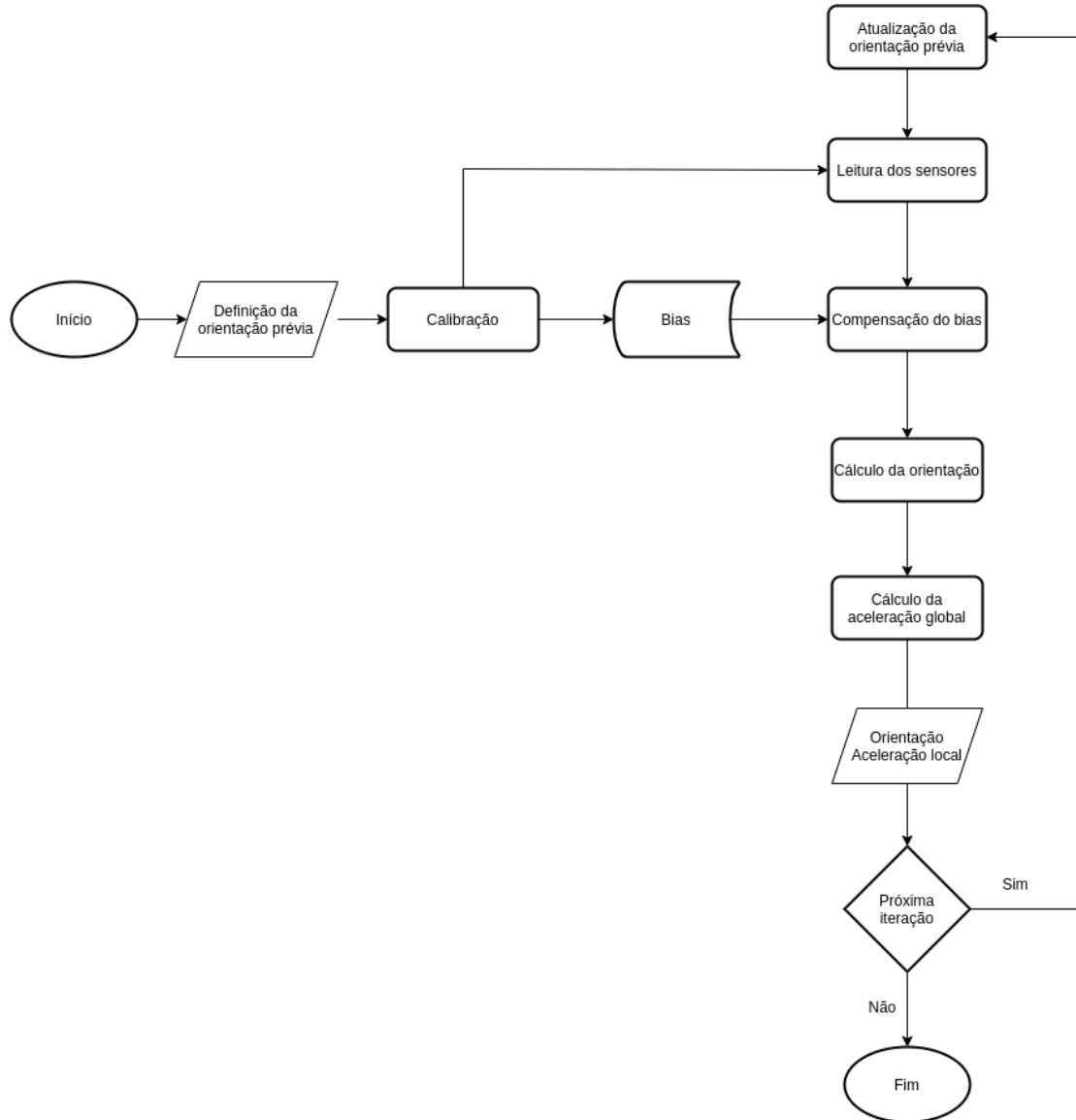
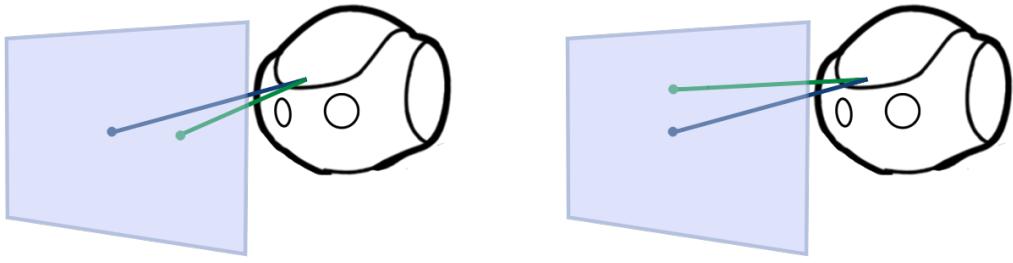


Figura 3.4: Fluxograma do processo de tratamento dos dados dos sensores iniciais. Fonte:
Autora

3.6 Dados da detecção de *landmarks*

O módulo de detecção de *landmarks* retorna, além do número identificador da Naomark e da marca temporal, parâmetros que descrevem a posição da Naomark na imagem com relação aos eixos da câmera. Os parâmetros α e β representam, respectivamente, os ângulos em torno dos eixos y e z no sistema de coordenadas tridimensional da câmera. Esse ângulos são formados entre a posição do centro da *landmark* e a bissetriz do campo de visão da câmera e são representados na Figura 3.5. Também é obtido, pela detecção, o diâmetro angular da marcação nos eixos x e y da imagem. A partir dessas propriedades, pode-se encontrar a posição da *landmark* com relação à câmera e, em seguida, com relação ao sistema de referência do robô.



(a) Naomark detectada com variação de orientação na horizontal. O ângulo entre as linhas verde e azul corresponde ao parâmetro α .

(b) Naomark detectada com variação de orientação na vertical. O ângulo entre as linhas verde e azul corresponde ao parâmetro β .

Figura 3.5: Representação dos ângulos α e β da detecção de Naomarks. A bissecriz do campo de visão é apresentada em cor azul. As linhas verdes indicam o centro na imagem da Naomark detectada. Fonte: Autora

Primeiramente, relacionando a largura real da *feature* com a largura detectada na imagem, pode-se obter a distância entre a *landmark* e a câmera do robô por meio de semelhança de triângulos. A distância d é dada, portanto, por

$$d = \frac{\varnothing}{2 \tan \frac{\delta}{2}} \quad (3.2)$$

em que \varnothing representa o diâmetro real em metros da *landmark* usada e δ representa o diâmetro angular medido pelo módulo de detecção.

As operações de translação e rotação são representadas, no módulo de detecção, por uma matriz homogênea simplificada: ela não possui a última linha de valores zero e um, que ajusta a matriz à quarta dimensão. A matriz é constituída apenas pela matriz de rotação \mathbf{R} e pela translação \mathbf{T} conforme

$$\mathbf{H}_{3x4} = \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{T}_{3x1} \end{bmatrix}. \quad (3.3)$$

Os ângulos α e β são utilizados para calcular a matriz homogênea referente à rotação que

orienta a câmera no sentido da *landmark*, representada por \mathbf{H}_r . Já a distância d previamente computada é utilizada para a obtenção da matriz homogênea referente à translação entre *landmark* e câmera, representada por \mathbf{H}_t . O posicionamento da *landmark* com relação à câmera, então, é descrito pela matriz \mathbf{H}_{cam}^{lmark} obtida a partir da multiplicação das matrizes anteriores seguindo a relação

$$\mathbf{H}_c^{lmark} = \mathbf{H}_r \mathbf{H}_t. \quad (3.4)$$

Ainda, as coordenadas podem ser transformadas do sistema cartesiano centrado na câmera para o sistema cartesiano local, que está localizado no plano xy entre os pés do robô. Para computar essa transformação entre câmera e robô, o método usa o posicionamentos das juntas para obter a matriz de transformação equivalente. Por fim, encontra-se a matriz \mathbf{H}_l^{lmark} , que relaciona a *landmark* ao espaço local, conforme

$$\mathbf{H}_l^{lmark} = \mathbf{H}_l^c \mathbf{H}_c^{lmark}. \quad (3.5)$$

Com os cálculos apresentados, é possível obter a distância no mundo real entre robô e objeto detectado a partir das medidas obtidas pelo processamento da imagem. Por outro lado, também pode-se fazer a relação inversa. É possível obter as respectivas medidas a partir do conhecimento de um ponto do mundo. Essa transformação permite que, na fase de correção do Filtro de Kalman, a predição do estado seja levada para o espaço de medições. Dessa forma é possível compará-la com as medidas fornecidas por visão computacional. Essa transformação é feita previamente ao cálculo do resíduo na etapa de predição do filtro cujo projeto será detalhado na próxima seção.

3.7 Projeto do filtro

O sistema físico equivalente ao NAO é um sistema dinâmico que pode ser modelado matematicamente por equações diferenciais. Uma representação mais conveniente é feita em espaço de estados. Para obter o melhor desempenho do filtro, é importante escolher as variáveis que melhor representam o sistema real. O restante desta seção apresenta os passos para definição de parâmetros do filtro.

3.7.1 Variáveis de estado

Como visto anteriormente, para acompanhar a movimentação do robô pelo campo, é necessário acompanhar a evolução do estado do sistema. Portanto, devem ser escolhidas como variáveis de estado aquelas que possam representar as coordenadas x e y do robô e sua orientação θ .

A escolha das variáveis de estado depende da interpretação que é dada aos sensores inerciais. Eles podem ser considerados como parte do modelo de movimento e, portanto, são incluídos no vetor de estado \mathbf{x} conforme

$$\mathbf{x} = [\mathbf{p}^T \ \mathbf{v}^T \ \mathbf{a}^T \ \boldsymbol{\theta}^T \ \boldsymbol{\omega}^T]^T, \quad (3.6)$$

em que \mathbf{p} , \mathbf{v} , \mathbf{a} são, respectivamente, os vetores posição, velocidade e aceleração linear e $\boldsymbol{\theta}$ e $\boldsymbol{\omega}$ são as representações da orientação e velocidade angular.

Sob outra perspectiva, os dados dos sensores inerciais podem ser utilizados como sinais de controle do modelo e são então retirados do estado do robô. O vetor de estados \mathbf{x} passa a ser representado como

$$\mathbf{x} = [\mathbf{p}^T \ \mathbf{v}^T \ \boldsymbol{\theta}^T]^T \quad (3.7)$$

em que $\boldsymbol{\theta}$ pode assumir diversas representações de orientação.

Para este trabalho, foi adotado o segundo modelo, mais compacto. O emprego dos sinais dos sensores inerciais como controle do sistema dinâmico permite a associação do ruído de processo com os próprios ruídos das medições desses sensores. O desenvolvimento do ruído de processo é apresentado detalhadamente na Seção 3.7.3.

Os termos do segundo modelo podem ainda ser interpretados de uma maneira mais intuitiva. Os elementos mínimos que devem constar como variáveis de estado são a posição em x e y além da representação de orientação do robô. Ainda, como a velocidade do robô não permanecerá constante durante sua movimentação, essa é outra informação relevante a ser incorporada ao filtro de forma a melhorar seu resultado.

Termos de ordem superior, como a aceleração, também estarão sujeitos a variação durante a movimentação no mundo real. No entanto, é importante respeitar a dinâmica real do sistema e não incluir termos de ordem superior deliberadamente. Caso os sensores inerciais fossem utilizados como sinal de entrada e a aceleração também fosse incluída na modelagem, o filtro interpretaria os ruídos inerentes às medições como aceleração realmente sofrida pelo robô. O filtro teria então comportamento de rastreamento dos ruídos e isso traria prejuízo ao seu desempenho [34].

Portanto o vetor de estados foi definido como

$$\mathbf{x} = [x \ \dot{x} \ y \ \dot{y} \ z \ \dot{z} \ \mathbf{R}_{1x9}]^T, \quad (3.8)$$

em que \mathbf{R}_{1x9} representa a matriz de rotação do sistema descrita em um vetor unidimensional. Embora não seja relevante para o problema acompanhar a posição e velocidade linear no eixo z , essas variáveis foram incluídas no modelo com o objetivo de permitir o melhor acompanhamento da evolução do estado durante o desenvolvimento do trabalho.

Por conveniência, no restante do trabalho, as representações do vetor de estados da Equação 3.8 e Equação 3.7 serão utilizadas de forma intercambiável, indicando apenas uma representação mais detalhada ou mais compacta.

3.7.2 Função transição de estado e função de controle

A função de transição de estado é responsável por atualizar o estado \mathbf{x} para o próximo instante de tempo, levando em consideração a influência que o sinal de controle tem sobre o resultado. Assim, a predição do estado é dada por

$$\bar{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}). \quad (3.9)$$

Utilizando as Equações 2.28 e 2.31, desenvolvidas na Seção 2.2, pode-se encontrar as equações de transição de estado, dadas por

$$\begin{bmatrix} \bar{\mathbf{p}} \\ \bar{\mathbf{v}} \\ \bar{\mathbf{R}} \end{bmatrix} = \begin{bmatrix} \mathbf{p} + \Delta t \mathbf{v} \\ \mathbf{v} + \Delta t (\mathbf{R} \mathbf{a} - \mathbf{g}) \\ \mathbf{R} \left(\mathbf{I} + \frac{\sin \sigma}{\sigma} \mathbf{B} + \frac{1 - \cos \sigma}{\sigma^2} \mathbf{B}^2 \right) \end{bmatrix}. \quad (3.10)$$

3.7.3 Matriz de ruído de processo

Na seção anterior, foi feita uma simplificação no modelo do sistema. Escorregamentos, por exemplo, não foram modelados. A matriz de ruído de processos cobre essas falhas do modelo aplicando ruído ao estado.

A escolha dos valores dessa matriz se reflete no comportamento do filtro. Para valores muito pequenos, a interpretação é de que o processo tem pouco erro e, por consequência, o filtro pode dar maior peso à predição. Para valores muito grandes, o filtro dá maior peso às medidas e o resultado passa a ser influenciado pelos ruídos das medidas. Essas situações podem levar à operação em modo subótimo ou até mesmo à divergência do filtro.

O ruído associado ao processo pode ser modelado como um ruído branco de média zero. No domínio discreto, por conveniência, pode-se considerar que o valor difere para cada período de amostragem, mas que permanece constante durante todo o período. Para esse modelo, é necessário fornecer o valor da variância do processo. Esse valor pode ser ajustado empiricamente a partir das observações dos erros dos sensores iniciais como descrito na Seção 3.5.1. Assim as variâncias relativas à aceleração linear e velocidade angular são adicionadas à matriz de ruído de processo a fim de permitir que o filtro se adeque às variações não previstas pelo modelo.

3.7.4 Função de medida

A função de medida realiza a transformação do espaço das variáveis de estado para o espaço das variáveis de medição. As medições do sistema são fornecidas pelo módulo de detecção de *landmarks*. A saída do módulo contém as distâncias entre o robô e a Naomark nos eixos x e y em coordenadas locais. A saída é descrita por

$$\mathbf{z} = \begin{bmatrix} dx \\ dy \end{bmatrix}. \quad (3.11)$$

A mudança de domínio do estado \mathbf{x} para o espaço de medidas é obtida por

$$\mathbf{h}_x = \begin{bmatrix} \mathbf{R}_{1,1} & \mathbf{R}_{1,2} \\ \mathbf{R}_{2,1} & \mathbf{R}_{2,2} \end{bmatrix} \begin{bmatrix} \mathbf{lmark}_x - x \\ \mathbf{lmark}_y - y \end{bmatrix}, \quad (3.12)$$

em que os termos $\mathbf{R}_{i,j}$ representam os elementos da matriz de rotação, que faz parte do estado \mathbf{x} . Para encontrar a medida correspondente ao estado, o posicionamento global da *landmark* deve ser extraído do mapa armazenado no robô.

3.7.5 Matriz de ruído de medida

As medições do sistema são as estimativas de distância nos eixos x e y fornecidas pelo módulo de detecção de *landmarks*. Portanto a matriz de ruído de medida \mathbf{R} é construída a partir dos valores de variância σ_x^2 e σ_y^2 conforme

$$\mathbf{R} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}. \quad (3.13)$$

Para avaliar a detecção de *landmarks*, o robô foi posicionado em diversas pontos de coordenadas conhecidas. Assim pode-se comparar as medições obtidas pelo módulo com as distâncias reais. A Figura 3.6 apresenta alguns das posições em que o robô foi colado. Foram testadas três configurações com diferentes distâncias. Na primeira, o eixo x do robô foi alinhado com a *landmark* de forma a mensurar o distanciamento em um único eixo. Na segunda configuração, o robô foi posicionado com seu eixo x ainda na mesma orientação, porém com deslocamento em y também. Na terceira, o robô foi orientado em direção a *landmark*. A diferença entre a distância real e a distância computada por visão foi calculada para cada um dos casos. O valor encontrado para a médida desses valores nos dois eixos foi de $\sigma = 0.0825$ m.

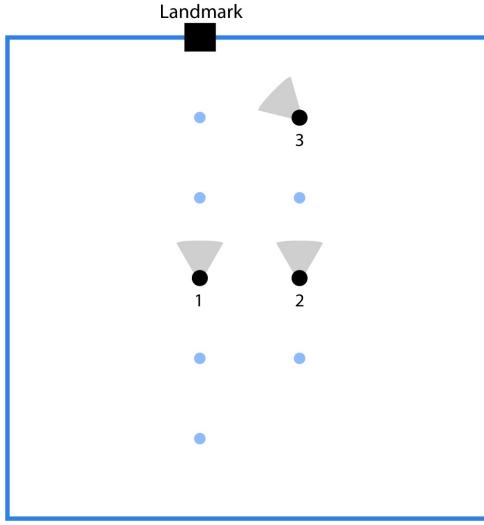


Figura 3.6: Representação das posições usadas para teste da detecção da *landmark*. Fonte:
Autora

Como o posicionamento do robô foi feito manualmente e, devido a sua forma, é difícil estabelecer um ponto específico de referência, os valores encontrados não representam corretamente o real erro associado às detecções. No entanto, são suficientemente apropriados para utilização no filtro. Assim a matriz de ruído de medida se torna

$$\mathbf{R} = \begin{bmatrix} 0.0825^2 & 0 \\ 0 & 0.0825^2 \end{bmatrix} \quad (3.14)$$

3.7.6 Condições iniciais

No início da partida, o robô pode ser posicionado próximo a pontos específicos cujas coordenadas são de conhecimento prévio. Assim, durante os testes, o robô foi sempre posicionado na origem do ambiente de validação e orientado com o sistema de coordenadas global. Pôde-se ainda definir uma alta confiança da posição inicial. O vetor de estados inicial é definido com os valores de \mathbf{x} conforme

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \\ \mathbf{R}_{1x9} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{I}_{1x9} \end{bmatrix} \quad (3.15)$$

em que \mathbf{I}_{1x9} é a representação em vetor da matriz identidade \mathbf{I}_{3x3} .

3.8 Implementação do filtro

O robô NAO permite implementações em linguagem Python e C++ em conjunto com sua *framework* NAOqi. Como o código de competição da equipe UnBeatables foi desenvolvido em Python, optou-se pelo uso dessa linguagem neste trabalho para facilitar posterior adaptação. Alguns detalhes da implementação são apresentados nas próximas seções.

3.8.1 Estrutura básica do código

Para que o robô pudesse realizar a movimentação em paralelo com a detecção de *landmarks* e o processo de localização, foram criadas *threads* para o gerenciamento do fluxo de execução. Já para o compartilhamento de informações entre os módulos, foi construído um módulo de memória compartilhada. Essa estrutura de código é semelhante a do *framework* da equipe UnBeatables, facilitando a futura reutilização deste trabalho no contexto da SPL.

Inicialmente, utilizou-se a biblioteca Filterpy¹ para confecção do filtro. Essa biblioteca tem implementados métodos de filtros bayesianos diversos [34]. No entanto, as ferramentas do robô não apresentaram compatibilidade com essa biblioteca e, portanto, não foi possível realizar a sua instalação. Ainda assim, os métodos encontrados nessa biblioteca foram utilizados de referência para o desenvolvimento do código deste trabalho. A Figura 3.7 apresenta um diagrama da arquitetura do código deste trabalho.

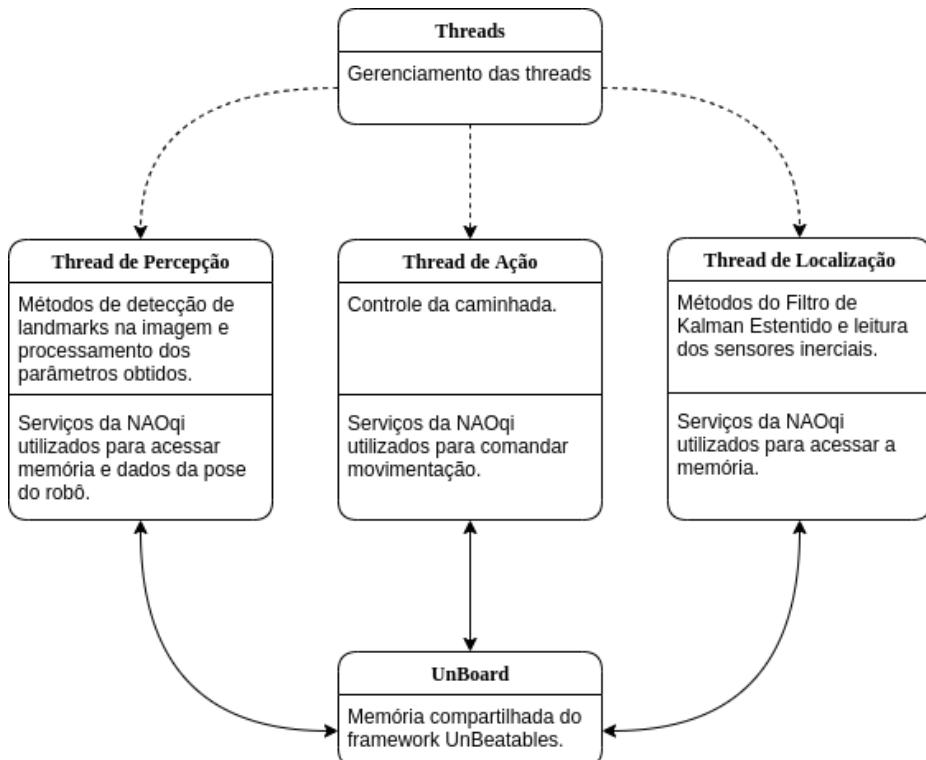


Figura 3.7: Diagrama da arquitetura do código implementado. Fonte: Autora

¹Documentação da biblioteca disponível em <https://filterpy.readthedocs.io/en/latest/>

3.8.2 Cálculo das matrizes Jacobianas

As funções de estado desenvolvidas para este trabalho se tornaram extremamente complexas e, para calcular as respectivas Jacobianas, considerou-se inicialmente o uso da biblioteca Sympy². No entanto, o robô apresentou incompatibilidade com a biblioteca e não foi possível sua utilização dentro da plataforma. A fim de encontrar uma solução temporária que permitisse o restante do desenvolvimento, a matriz Jacobiana foi criada considerando um modelo simplificado do sistema. Isso acarreta em erros no cálculo da matriz de covariância. Por conta dessa alternativa, o filtro deixa de operar otimamente e não é capaz de convergir, visto que não há a correta atualização dos valores associados ao erro da estimativa. Esse fator deve ser levado em consideração durante a análise dos resultados.

3.8.3 Cálculo da covariância na fase de correção

A matriz de covariância é atualizada na fase de correção a partir da relação

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \bar{\mathbf{P}}_t. \quad (3.16)$$

No entanto, a implementação adotada calcula a matriz de covariância como

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \bar{\mathbf{P}}_t (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t)^T + \mathbf{K} \mathbf{R} \mathbf{K}^T. \quad (3.17)$$

Essa abordagem é válida tanto para o filtro ótimo quanto para a execução subótima. Devido à assimetria do termo $\mathbf{I} - \mathbf{K}_t \mathbf{H}_t$, o cálculo tradicional pode levar a instabilidades numéricas, que são evitadas com a reformulação da equação [34] sem trazer prejuízo para o valor calculado.

²Documentação da biblioteca disponível em <https://docs.sympy.org/latest/index.html>

Capítulo 4

Resultados

Neste capítulo, o ambiente e alguns passos da validação são explanados. Os resultados obtidos com o sistema de localização são apresentados em etapas. Primeiramente, a análise do desempenho do filtro considerando apenas a execução da predição é realizada. Depois, são discutidos os resultados do filtro considerando execução da etapa de predição. Por fim, é apresentado o teste realizado com múltiplas landmarks no ambiente.

4.1 Introdução

Antes de prosseguir para os resultados do trabalho, faz-se necessário detalhar o ambiente de teste e explicitar demais detalhes. Diferentemente das *features* do campo de futebol que estão no plano *xy*, as *landmarks* impressas foram afixadas em plano vertical conforme verificado na Figura 4.1. Apesar da diferença de planos, não há prejuízos para a análise do filtro.



Figura 4.1: Ambiente de validação com o robô NAO observando uma Naomark afixada na parede. Fonte: Autora

O conjunto de *landmarks* utilizadas variou para cada teste. Toda as etapas de validação foram executadas em ambiente real. Uma das configurações de ambiente construídas pode ser visualizada na Figura 4.2. Essa configuração foi utilizada para o teste final e será retomada com mais detalhes na Seção 4.5.



Figura 4.2: Ambiente de validação construído com múltiplas Naomarks afixadas nos planos verticais. Fonte: Autora

A origem do sistema global também teve seu posicionamento alterado para conveniência da execução dos testes. No entanto, para todos os ensaios, o posicionamento inicial do robô foi considerado coincidente com a origem do sistema global e os eixos de coordenadas local foram alinhados aos globais. Nas imagens de representação dos movimentos, por simplificação, a posição inicial do robô está omitida, pois coincide sempre com a origem do sistema.

Vale notar que tanto o posicionamento inicial do robô quanto as medições finais, após sua caminhada, foram feitas por inspeção visual. Essas informações servem como diretriz para análise dos dados obtidos, mas suas medições, inexatas, não interferem no cômputo de desempenho do filtro.

Ainda, com o objetivo de acompanhar o desempenho do robô com relação a um comportamento esperado, optou-se por não controlar o robô por comandos de velocidade. A abordagem escolhida para o controle do robô foi a de utilizar um comando de movimentação dado por

$$\mathbf{mov} = \begin{pmatrix} x & y & \theta \end{pmatrix}, \quad (4.1)$$

em que x representa a posição desejada no eixo x , y representa a posição desejada no eixo y e θ a orientação desejada em torno do eixo z .

Por fim, para verificar os períodos de detecção bem sucedida de *landmarks*, as imagens da câmera do robô foram monitoradas durante toda a execução dos testes. O *software* Monitor, fornecido pela fabricante do NAO, foi utilizado para observar as imagens. Exemplos de imagens

visualizadas por esse programa são apresentadas na Figura 4.3. Em caso de detecção, além do círculo delimitador, o número identificador da Naomark também é escrito na imagem.

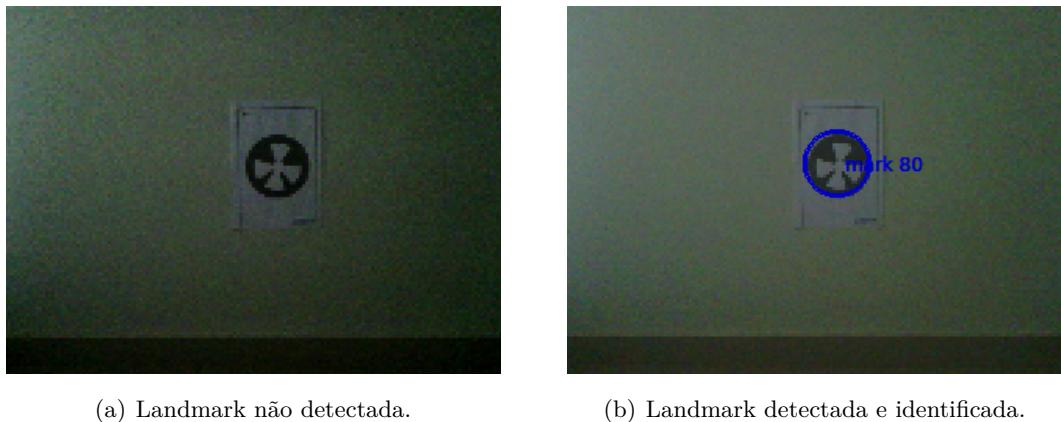


Figura 4.3: Imagens, obtidas pelo Monitor, da câmera do robô posicionado em frente a uma Naomark. Fonte: Autora

4.2 Avaliação da predição de posição

A fim de analisar o desempenho da odometria na etapa de predição, foram retiradas do ambiente todas as Naomarks, de forma que o filtro não realizasse correções das suas estimativas. Primeiro avaliaram-se os resultados da posição estimada em x e y . O comando de movimentação dado ao robô foi $\text{mov} = \begin{pmatrix} 2 & 0 & 0 \end{pmatrix}$ para um deslocamento de 2 m no eixo x apenas. Uma representação da posição final real do robô é apresentada na Figura 4.4.

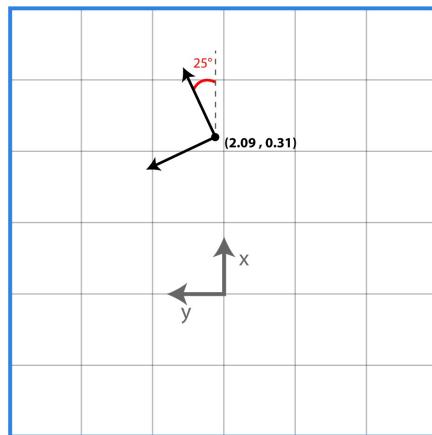


Figura 4.4: Posição final do robô após comando de movimentação para aferição da predição de posição. Fonte: Autora

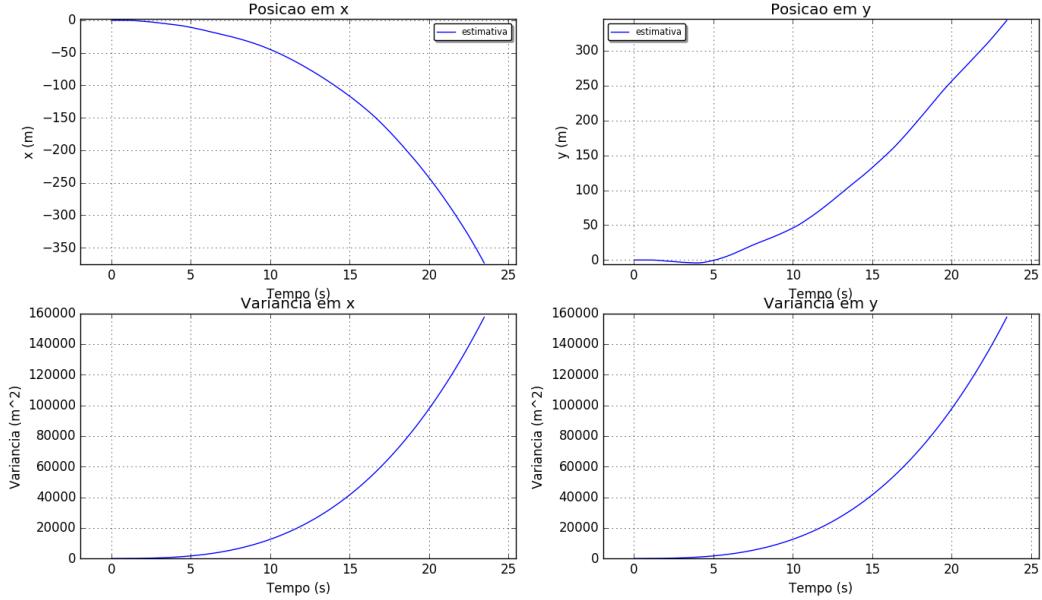


Figura 4.5: Resultados do filtro para x e y após teste de movimentação no eixo x para aferição da predição de posição. Fonte: Autora

Pode-se perceber que o robô se deslocou um pouco da posição final desejada devido aos escorregamentos durante sua caminhada. Apesar de a trajetória ter sido realizada sem distúrbios externos, o filtro foi incapaz de estimar a posição do robô satisfatoriamente. Os valores estimados para x e y podem ser observados na Figura 4.5 bem como a variância para cada uma dessas variáveis.

O acúmulo de erros na estimativa da posição é inerente ao modelo da odometria. A curva da posição tem perfil quadrático de acordo com o esperado para o acúmulo de erros do acelerômetro. O alto valor da variância indica a divergência do filtro e representa o grau de descrença do sistema sobre a posição estimada.

4.3 Avaliação da predição de orientação

Para avaliar o rastreamento da orientação a partir dos dados do giroscópio, o robô recebeu o comando $\mathbf{mov} = \begin{pmatrix} 0 & 0 & 3.1415 \end{pmatrix}$ duas vezes seguidas para completar uma volta em torno do seu eixo z . A Figura 4.6. ilustra a posição e orientação do robô no início e fim do teste. Percebe-se que há um pequeno deslocamento translacional e que o ângulo final obtido difere daquele comandado, como esperado devido aos escorregamentos.

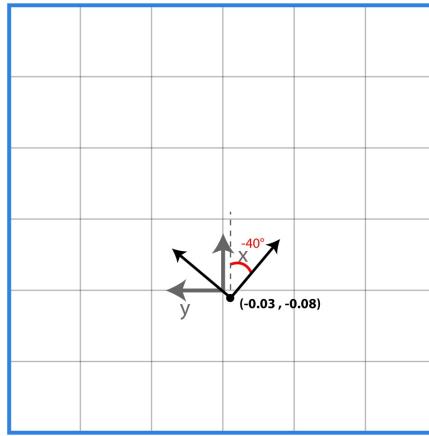


Figura 4.6: Posição final do robô após comando de movimentação para aferição da predição de orientação. Fonte: Autora

Como o modelo do sistema contava com uma matriz de rotação para estimativa da orientação do robô, o desempenho da estimativa da orientação foi avaliado a partir dos valores de projeção dos eixos x e y locais sobre os eixos do sistema global. Tais resultados são apresentados na Figura 4.7. Pode-se perceber que o filtro fornece uma predição acurada da orientação do robô, condizente com o movimento realmente realizado. Como verificado anteriormente, a calibração para o giroscópio forneceu uma boa estimativa do *bias* permitindo sua compensação na leitura dos sensores. A diminuição dos erros associados à leitura do sensor se reflete no gráfico pelo valor baixo da variância, representada pela região preenchida de azul.

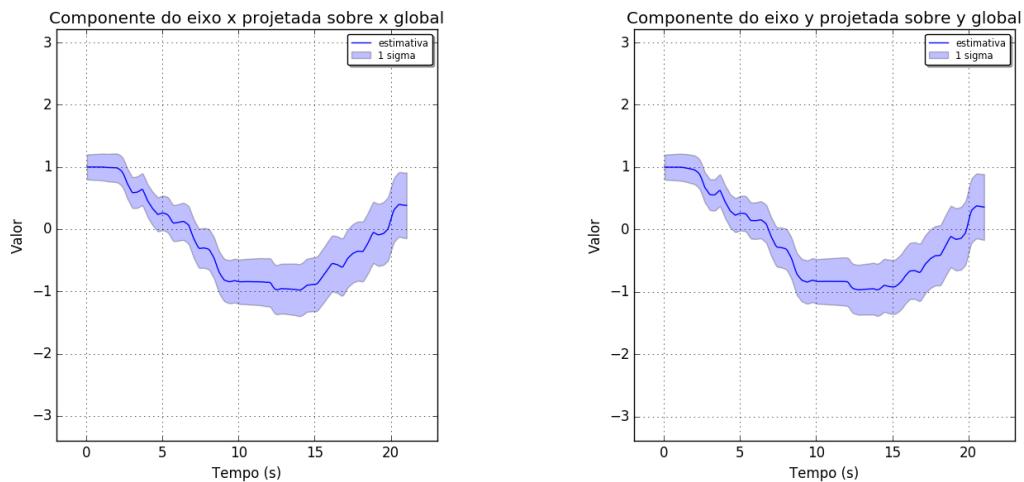


Figura 4.7: Resultados do filtro para as projeções dos eixos x e y após teste de rotação em torno do eixo z para aferição da predição de orientação. Fonte: Autora

4.4 Avaliação da correção

Para avaliar a etapa de correção, foi realizado um teste similar ao primeiro teste desta Seção para fins de comparação. Porém, neste ensaio a *landmark* foi posicionada no campo de visão do robô a 2,5 m, no eixo x , da origem do sistema global. Ainda, para melhor avaliação dos eventos determinantes, foi utilizado um comando de menor extensão dado por $\text{mov} = \begin{pmatrix} 1,5 & 0 & 0 \end{pmatrix}$. A representação da posição final do robô é apresentada na Figura 4.8.

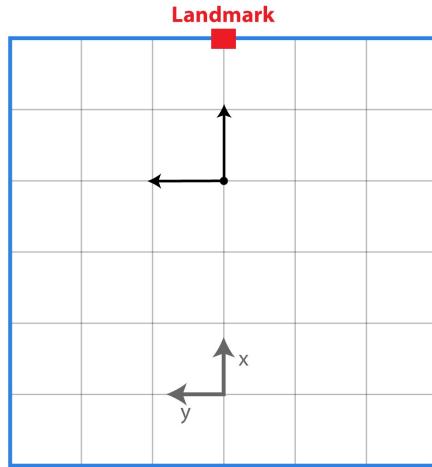


Figura 4.8: Posição final do robô após comando de movimentação para aferição da correção.

Fonte: Autora

Os resultados do filtro obtidos para essa movimentação são apresentados na Figura 4.9. Pode-se notar na figura a tendência do filtro de acompanhar o comportamento do robô, apesar dos pontos de divergência. No eixo x , o valor estimado aumenta à medida que o robô se movimenta para a frente. Já no eixo y , o filtro apresenta mais oscilações detectando deslocamentos laterais. Os intervalos em que a estimativa difere do comportamento real observado e em que a variância tem crescimento acelerado coincidem com os períodos em que o robô não foi capaz de identificar a Naomark. O crescimento dos erros se deve, portanto, ao acúmulo de erros provenientes da odometria.

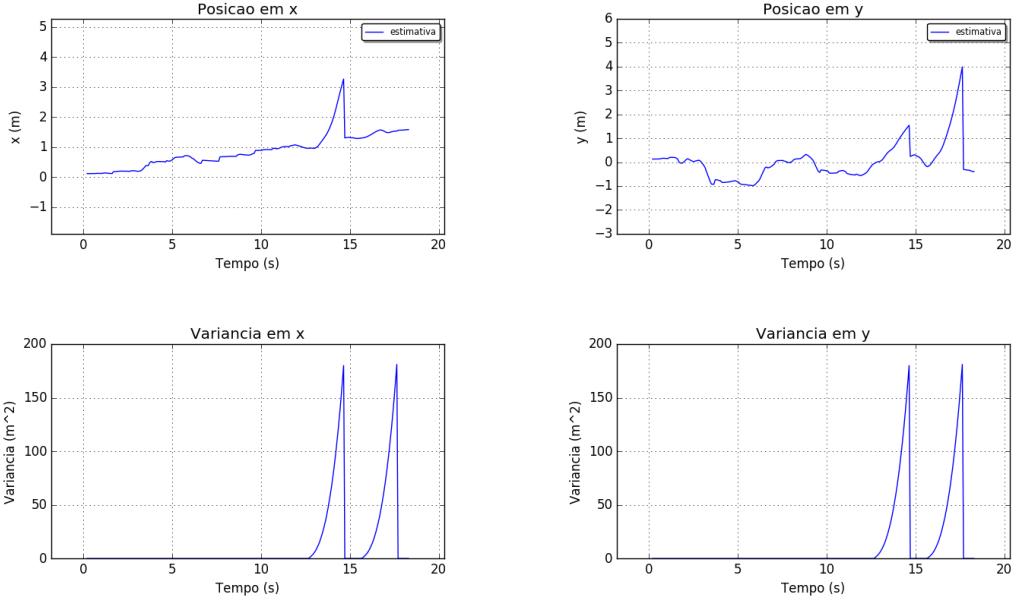


Figura 4.9: Resultados do filtro para x e y após teste de movimentação no eixo x para aferição da correção. Fonte: Autora

Para melhorar o resultado com relação à perda da detecção da *landmark*, poderia ainda ser implementado um armazenamento temporário dos dados em uma estrutura apropriada com função similar à de um *buffer* de dados. Assim, as informações relativas ao posicionamento do robô com relação a *landmark* podem ser utilizadas ainda pelos próximos instantes de tempo. Isso melhoraria o desempenho do filtro nesses períodos de visão prejudicada. Tais momentos podem ocorrer em uma situação de jogo por variação da iluminação, embora menos provável, e também por oclusão devido à movimentação de outros robôs, caso bastante comum.

Apesar do resultado oscilatório para a estimativa da posição, as estimativas da orientação se mantiveram muito próximas do valor esperado. O resultado para a orientação é apresentado na Figura 4.10. A região preenchida em azul representa a confiança do filtro com relação a essa estimativa. Embora o valor médio da estimativa de orientação apresente resultados muito bons, a variância da orientação está aumentando. Isso indica que o filtro não está convergindo.

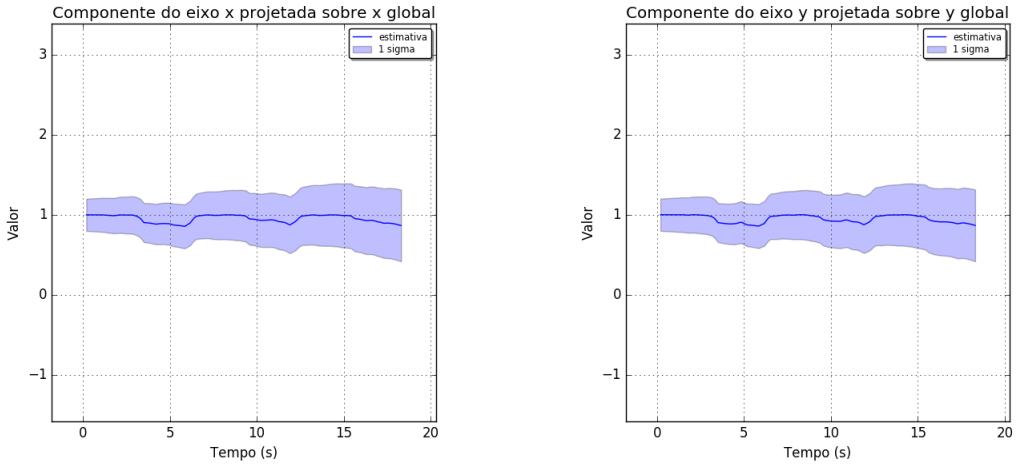


Figura 4.10: Resultados do filtro para as projeções dos eixos x e y após teste de movimentação no eixo x para aferição da correção. Fonte: Autora

4.5 Resultados do filtro em ambiente com múltiplas *landmarks*

A configuração do ambiente para o uso de múltiplas *landmarks* foi realizada considerando meio campo em uma escala de redução 1:2. Foram incluídos três tipos de *Naomarks* para representar as interseções entre meio de campo e lateral; entre linha de fundo e área do gol; e entre a lateral e o fundo de campo, no escanteio. A Figura 4.11 representa tanto a configuração do ambiente quanto o movimento comandado para o robô. O movimento incluiu translação e rotação. Uma representação tridimensional desse ambiente pode ser observada na Figura 4.12.

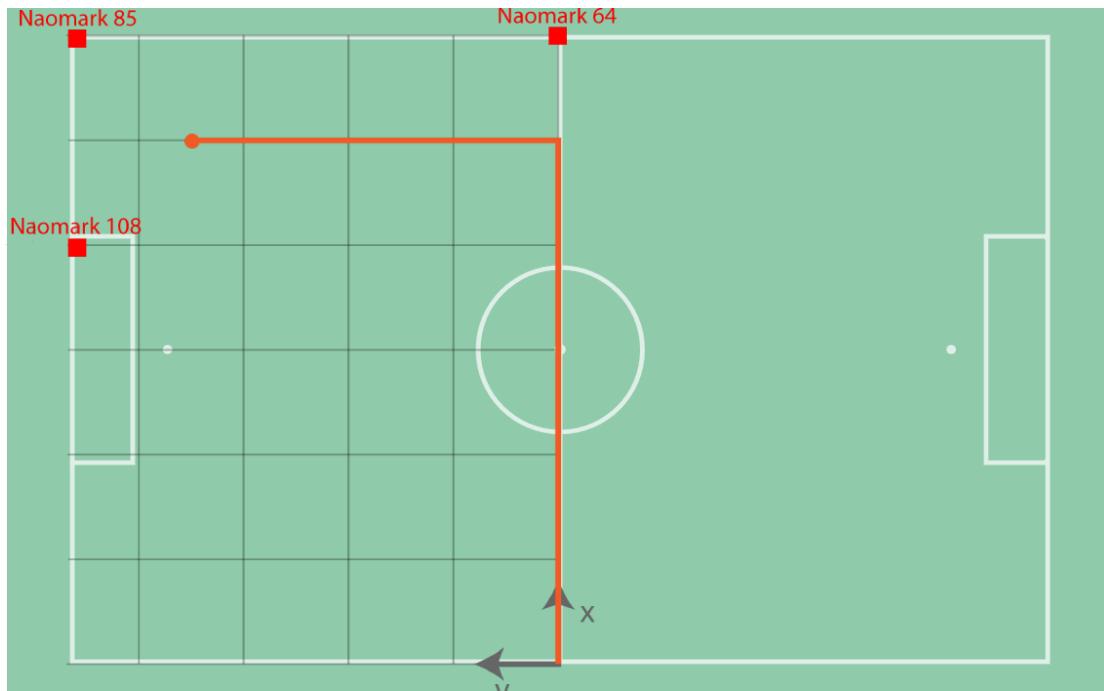


Figura 4.11: Representação do ambiente de teste com múltiplas *landmarks*. Fonte: Autora

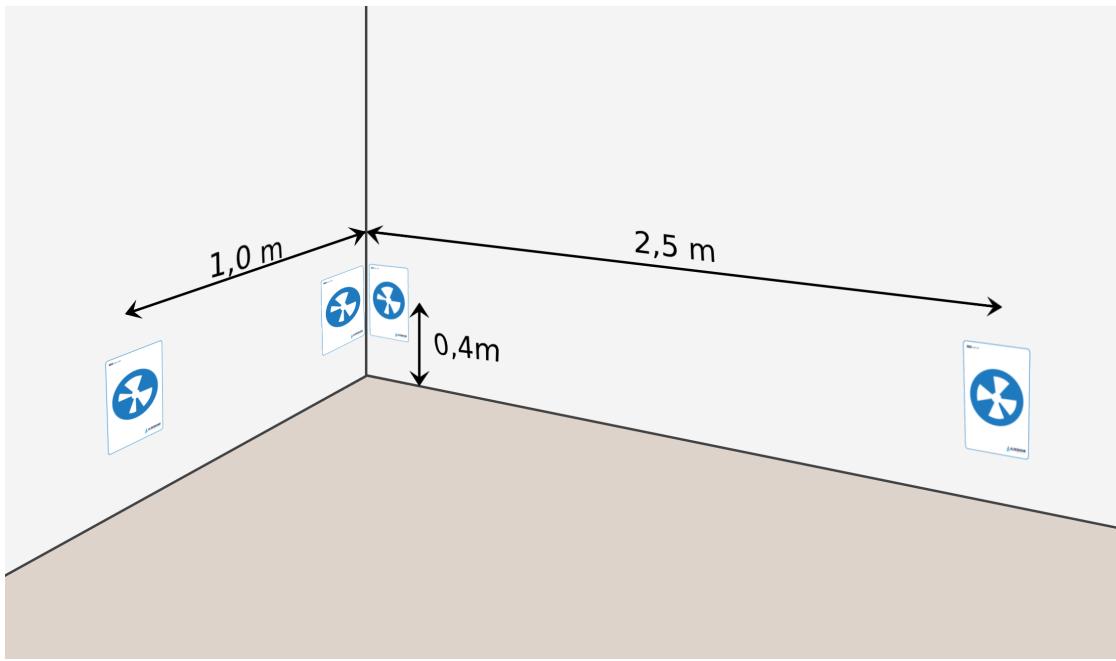


Figura 4.12: Representação tridimensional do ambiente de validação construído com múltiplas Naomarks afixadas nos planos verticais. Fonte: Autora

Os resultados do filtro são apresentados na Figura 4.13. Mais uma vez, pode-se perceber que o filtro apresenta oscilações devido a perda da *Naomark* no módulo de detecção. Nesses intervalos em que a câmera não foi capaz de identificar a *landmark*, os erros provenientes da odometria se acumulam rapidamente.

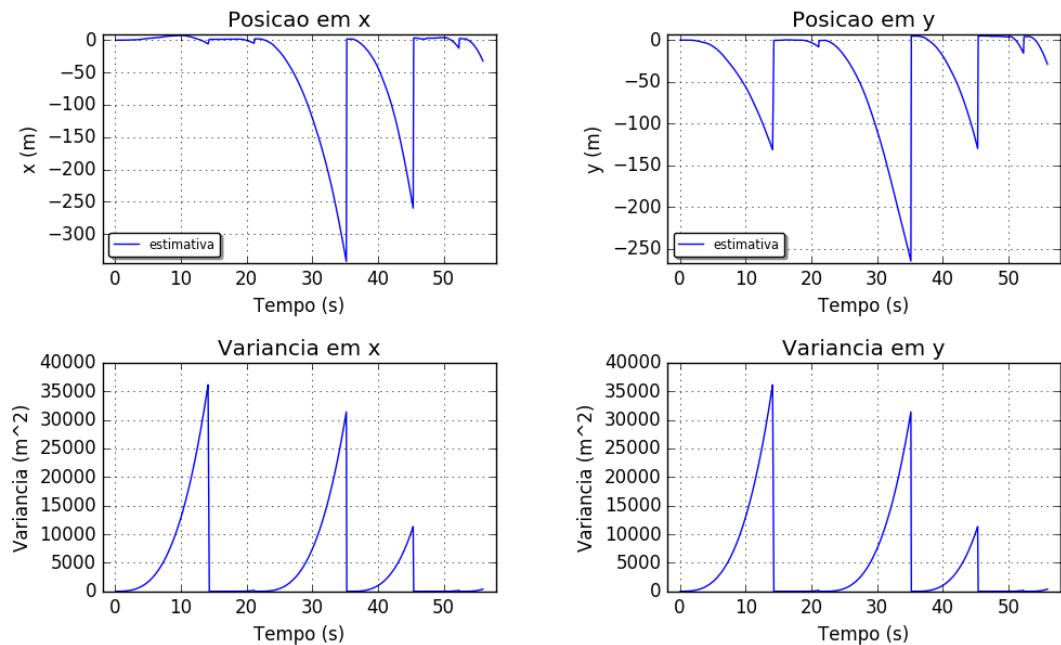
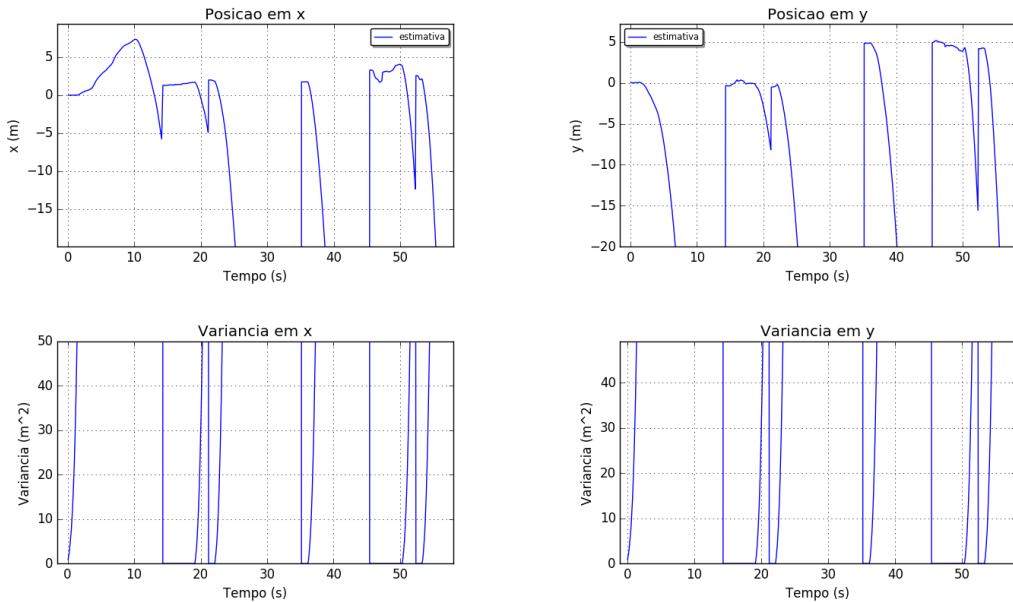


Figura 4.13: Resultados do filtro para x e y após teste de movimentação em ambiente com múltiplas *landmarks*. Fonte: Autora

A fim de melhor avaliar o resultado da fase de correção do filtro, os limites de visualização do eixo y dos gráficos apresentados anteriormente foram reduzidos. O gráfico resultante é apresentado na Figura 4.14. Os intervalos em que a *landmark* não foi detectada ainda são visíveis. Porém, percebe-se um bom resultado para a estimativa da posição. Na primeira metade do tempo, quando o robô se locomovia no eixo x , é possível perceber a tendência da curva de acompanhar esse movimento no eixo x e permanecer em zero para o eixo y . Avaliando a segunda metade do teste, quando o robô seguia pelo eixo y depois de haver girado em torno de seu eixo z , nota-se que os valores em x permanecem relativamente estáveis. No entanto, para y o resultado não foi satisfatório, indicando que a atualização da orientação não foi bem sucedida.



A Figura 4.15 apresenta as estimativas para a orientação durante esse teste. Percebe-se que na primeira metade do tempo, o filtro foi capaz de acompanhar bem a orientação do NAO. A faixa em azul apresenta a variância da medida e nota-se que a faixa se mantém relativamente larga, porém com crescimento estabilizado. Isso indica que o filtro está operando em situação subótima, mas ainda apresentando bons resultados.

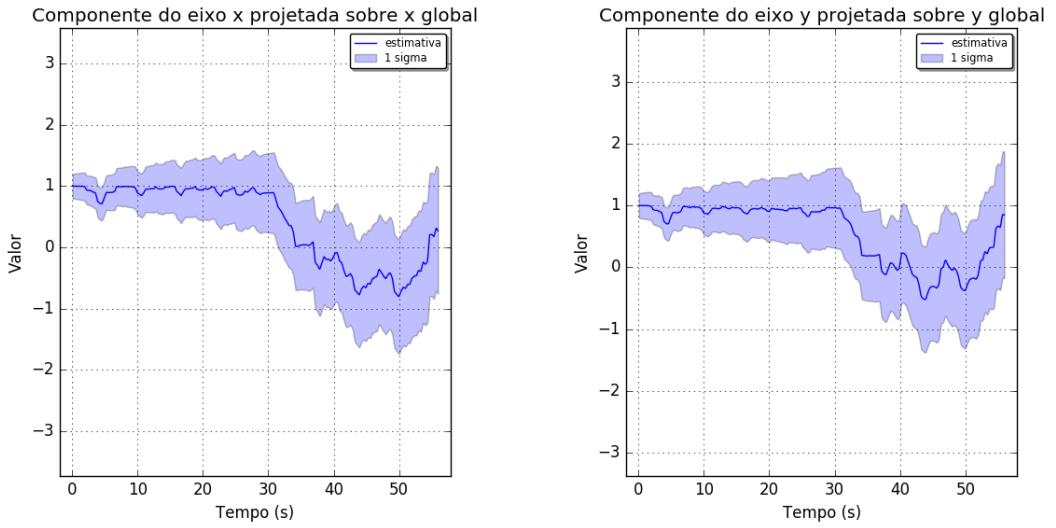


Figura 4.15: Resultados do filtro para as projeções dos eixos x e y após teste de movimentação em ambiente com múltiplas *landmarks*. Fonte: Autora

Comparando os testes realizados com e sem uso de *landmarks*, percebe-se a melhoria que o uso de dados da câmera traz para o sistema. O robô, que anteriormente não teria qualquer conhecimento de sua posição, passa a obter, nos momentos de detecção de feature, uma boa noção espacial. Isso permite que estratégias de jogo sejam exploradas considerando seu posicionamento levando a um melhor desempenho do time. Para alcançar esse objetivo, no entanto, os parâmetros do filtro ainda necessitam passar por ajustes para que o filtro opere de maneira ótima.

Capítulo 5

Conclusões

Este trabalho propôs um sistema de localização implementado no robô NAO visando sua localização no contexto de uma partida da SPL. Para desenvolver tal sistema, diversas estratégias de adaptação do problema foram estabelecidas a fim de validar o trabalho em um ambiente correspondente.

Primeiramente foi desenvolvido um módulo de detecção de Naomarks, as *landmarks* nativas da plataforma NAO. A partir do processamento das imagens da câmera, foi possível estimar com sucesso a distância entre a *landmark* e o robô. Em seguida, as leituras dos sensores iniciais foram aplicadas no cálculo de posição e orientação do robô por meio de odometria. Como esperado, devido ao acúmulo de erros do acelerômetro, a predição da posição divergiu rapidamente da posição real. No entanto, foi possível realizar a predição da orientação satisfatoriamente.

As informações obtidas nessas duas grandes etapas do desenvolvimento foram aplicadas a um Filtro de Kalman Estendido. Dentre as técnicas de filtragem discutidas neste trabalho, o Filtro de Kalman Estendido é a opção de menor custo computacional. De fato, o filtro foi executado na plataforma NAO, de processamento limitado, sem prejudicar a execução de demais tarefas imprescindíveis para a partida de futebol.

O filtro foi capaz de realizar a predição da orientação do robô satisfatoriamente bem como a correção de sua posição a partir da detecção de *landmarks*. A etapa de predição de posicionamento ainda apresenta grande acúmulo de erros. Apesar de parte desse acúmulo se dever ao próprio comportamento dos acelerômetros, ainda é possível melhorar o desempenho do filtro com o correto ajuste dos erros associados. Ainda, o sistema proposto se baseou em uma composição de matrizes alternativas. A construção das matrizes jacobianas conforme proposto no Filtro de Kalman Estendido original pode levar a um melhor cálculo da confiança das estimativas. O filtro ainda não pode ser confiavelmente aplicado em uma situação de jogo real, porém este trabalho apresenta avanços substanciais no desenvolvimento do sistema.

Para poder permitir a paralelização da caminhada, obtenção de informações do robô e localização, o código foi desenvolvido utilizando *threads* e memória compartilhada entre os módulos. Aspectos de mais baixo nível no gerenciamento do robô já foram cobertos por este trabalho e, portanto, próximas pesquisas poderão partir deste *framework*. Ainda, a modularidade do código

permite que futuros trabalhos se especializem em um único tópico a fim de alcançar o objetivo final em partidas de futebol. No que diz respeito a UnBeatables, a estrutura do código desenvolvido se assemelha àquela utilizada pela equipe, o que permitirá maior praticidade para a futura aplicação deste trabalho junto ao código do time.

5.1 Perspectivas futuras

Os próximos trabalhos inicialmente poderão focar em realizar melhorias no sistema desenvolvido neste trabalho. Ajustes no código poderão ser realizados a fim de permitir a correta execução dentro da plataforma NAO. Ainda a otimização do filtro pode ser alcançada pelo refinamento dos parâmetros utilizados. Outra possibilidade é a total reformulação do sistema para utilização de outro tipo de Filtro de Kalman, como o *Unscented Kalman Filter* e comparação do desempenho nos distintos filtros.

Ainda com o objetivo de melhorar o desempenho do filtro, em qualquer uma de suas variações que seja adotada, uma odometria baseada na caminhada do robô poderá ser desenvolvida. Nesse caso, a estimativa da translação do robô é calculada a partir dos estados das juntas do robô. Essas informações poderão ser incorporadas ao filtro levando a resultados mais acurados.

Também poderá ser incorporado a este trabalho um sistema de detecção de *features* do campo de futebol. Esse módulo substituiria o sistema de *landmarks* utilizado como alternativa para simulação dos resultados desse tipo de detecção. Mais outra melhoria pode ser realizada no que diz respeito à identificação das *features*. O sistema de localização foi testado com *landmarks* únicas. No entanto, o campo de futebol é simétrico e, logo, as *features* de mesmo aspecto precisam ser diferenciadas entre si.

Como último passo dessa cadeia de desenvolvimento, os módulos produzidos poderão ser incorporados ao *framework* da equipe UnBeatables para utilização do sistema de localização durante as partidas de futebol.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] SICILIANO, B. et al. *Robotics*: Modelling, planning and control. [S.l.: s.n.], 2009.
- [2] CORKE, P. *Robotics, Vision and Control*: Fundamental algorithms in matlab®. [S.l.]: Springer, 2011.
- [3] MATARIC, M. J. *The Robotics Primer*. [S.l.: s.n.], 2011.
- [4] ARKIN, R. C. Motor schema—based mobile robot navigation. *The International journal of robotics research*, Sage Publications Sage CA: Thousand Oaks, CA, v. 8, n. 4, p. 92–112, 1989.
- [5] KITANO, H. et al. Grand challenge ai applications. In: *IJCAI*. [S.l.: s.n.], 1993. p. 1677–1683.
- [6] FEDERATION, R. *A Brief History of RoboCup*. Disponível em: <http://www.robocup.org/a_brief_history_of_robocup>.
- [7] FEDERATION, R. *Objective*. Disponível em: <<http://www.robocup.org/objective>>.
- [8] MACKWORTH, A. K. On seeing robots. *Computer Vision: Systems, Theory and Applications*, p. 1–13, 1993.
- [9] CARVALHO, M. P. d. Controle de movimentação de humanoide em tempo real por teleoperação. 2014.
- [10] BALBINO, H. d. S. e. S. Desenvolvimento de sistema de teleoperação para robô humanoide. 2016.
- [11] COMMITTEE, R. T. *RoboCup Standard Platform League (NAO) Rule Book*. Disponível em: <<https://spl.robocup.org/wp-content/uploads/downloads/Challenges2016.pdf>>.
- [12] FARIA, C. M. d.; ROCHA, Y. G. Métodos de comunicação visual e controle cooperativo entre robôs humanoides. 2016.
- [13] RESENDE, R. A. B. Localização de robô humanoide aplicado a futebol de robôs. 2016.
- [14] COMMITTEE, R. T. *RoboCup Standard Platform League (NAO) Rule Book*. Disponível em: <<http://spl.robocup.org/wp-content/uploads/downloads/Rules2019.pdf>>.
- [15] ROBOTICS, S. *NAO Documentation*. Disponível em: <http://doc.aldebaran.com/2-1/home_nao.html>.

- [16] ROBOTICS, S. *NAO Documentation*. Disponível em: <http://doc.aldebaran.com/2-8/home_nao.html>.
- [17] RATH, C. Self-localization of a biped robot in the robocup standard platform league domain. 2010.
- [18] NÚNCIO, A. et al. Unbeatables team description.
- [19] BARRAQUAND, J.; LATOMBE, J.-C. Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research*, Sage Publications Sage CA: Thousand Oaks, CA, v. 10, n. 6, p. 628–649, 1991.
- [20] SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. *Introduction to autonomous mobile robots*. [S.l.]: MIT press, 2011.
- [21] FOX, D. et al. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, v. 1999, n. 343-349, p. 2–2, 1999.
- [22] GUTMANN, J.-S.; FOX, D. An experimental comparison of localization methods continued. In: IEEE. *IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.], 2002. v. 1, p. 454–459.
- [23] GUTMANN, J.-S. et al. An experimental comparison of localization methods. In: IEEE. *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)*. [S.l.], 1998. v. 2, p. 736–743.
- [24] BURCHARDT, A.; LAUE, T.; RÖFER, T. Optimizing particle filter parameters for self-localization. In: SPRINGER. *Robot Soccer World Cup*. [S.l.], 2010. p. 145–156.
- [25] MOLEN, H. van der; VISSER, A. Self-localization in the robocup soccer standerd platform league with the use of a dynamic tree. *Bachelor Thesis, University Of Amsterdam*, 2011.
- [26] MURRAY, R. M. *A mathematical introduction to robotic manipulation*. [S.l.]: CRC press, 2017.
- [27] WOODMAN, O. J. *An introduction to inertial navigation*. [S.l.], 2007.
- [28] BERNARDES, M. C. Controle servo-visual para aproximação de portas por robôs móveis equipados com duas câmeras. 2009.
- [29] FORSYTH, D. A.; PONCE, J. *Computer vision: a modern approach*. [S.l.]: Prentice Hall Professional Technical Reference, 2002.
- [30] ROMERO, R. A. et al. Robótica móvel. *São Paulo: LTC*, p. 21, 2014.
- [31] DUDEK, G.; JENKIN, M. *Computational principles of mobile robotics*. [S.l.]: Cambridge university press, 2010.

- [32] THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic robotics*. [S.l.]: MIT press Cambridge, 2000.
- [33] KOK, M.; HOL, J. D.; SCHÖN, T. B. Using inertial sensors for position and orientation estimation. *Foundations and Trends in Signal Processing*, Now Publishers Inc. Hanover, MA, USA, v. 11, n. 1-2, p. 1–153, 2017.
- [34] LABBE, R. Kalman and bayesian filters in python. *Chap*, v. 7, p. 246, 2014.

ANEXOS

I. PROGRAMAS UTILIZADOS

Foram implementados neste trabalho módulos de gerenciamento de *threads*, movimentação do robô NAO, detecção de *Naomarks* e localização utilizando o Filtro de Kalman Estendido. Os códigos e as instruções de execução estão disponíveis no repositório do projeto ¹.

¹<https://github.com/dfsboralocalization>