

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/343049662>

# Mapeamento e Controle de Robôs Móveis para a Categoria IEEE Very Small Size de Futebol de Robôs

Thesis · November 2019

DOI: 10.13140/RG.2.2.35392.35844

---

CITATIONS  
0

READS  
454

2 authors:



Júlio Resende  
Federal University of São João del-Rei

10 PUBLICATIONS 0 CITATIONS

[SEE PROFILE](#)



Marcos A. M. Laia  
Federal University of São João del-Rei

32 PUBLICATIONS 26 CITATIONS

[SEE PROFILE](#)

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Júlio César Mendes de Resende

**Mapeamento e Controle de Robôs Móveis para  
a Categoria IEEE *Very Small Size* de Futebol  
de Robôs**

São João del-Rei

2019

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Júlio César Mendes de Resende

**Mapeamento e Controle de Robôs Móveis para a  
Categoria IEEE *Very Small Size* de Futebol de Robôs**

Monografia apresentada como requisito parcial para obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal de São João del-Rei.

Orientador: Marcos Antonio de Matos Laia

Universidade Federal de São João del-Rei – UFSJ

Bacharelado em Ciência da Computação

São João del-Rei

2019

Júlio César Mendes de Resende

## **Mapeamento e Controle de Robôs Móveis para a Categoria IEEE *Very Small Size* de Futebol de Robôs**

Monografia apresentada como requisito parcial para obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal de São João del-Rei.

Trabalho aprovado. São João del Rei, 05 de novembro de 2019:

---

**Marcos Antonio de Matos Laia**  
Orientador

---

**Daniel Luiz Alves Madeira**  
Convidado 1

---

**Edimilson Batista dos Santos**  
Convidado 2

São João del Rei  
2019

*Este trabalho é dedicado à continuação da equipe UaiSoccer VSS, que tanto poderá colaborar para a formação de profissionais com conhecimentos multidisciplinares.*

# Agradecimentos

Primeiramente agradeço à Deus; meus pais Tiago e Zélia e a minha família por estarem sempre comigo, mesmo nos momentos mais difíceis da caminhada.

Agradeço a todos os meus professores pelos ensinamentos repassados, em especial ao Edimilson e ao Daniel Madeira, que prontamente aceitaram o convite para comporem a banca avaliadora deste trabalho.

Ao Marcos, meu orientador, obrigado por confiar em meu trabalho e por ter me acompanhado durante toda a graduação.

Aos meus amigos, gratidão pela cumplicidade e motivação.

À Universidade Federal de São João del Rei agradeço por oferecerem um ensino público, gratuito e de qualidade.

À Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), agradeço pela bolsa de iniciação científica concedida no ano de 2018, período em que grande parte do trabalho foi desenvolvido.

Agradecimentos especiais aos amigos André e Felipe e a toda equipe UaiSoccer VSS, responsáveis pela montagem e comunicação com os robôs utilizados.

Por fim, agradeço também a Rodrigo Chaves e Drauzio Oppenheimer pela ajuda no início da pesquisa e ao professor Eduardo Bento pela orientação dada no ano de 2018.

# Resumo

O futebol, um dos esportes mais praticados no mundo, ocupa também um papel de suma importância no meio acadêmico. A aplicação do mesmo na robótica vem sendo amplamente pesquisada devido à capacidade que o esporte possui de atrair estudantes, sendo o futebol de robôs da categoria IEEE Very Small Size (VSS) uma maneira de se praticar diferentes áreas de conhecimentos. Nesse sentido o trabalho aqui apresentado descreve uma combinação de métodos para mapeamento e controle de robôs da categoria VSS. Os robôs são reconhecidos através de um módulo de visão computacional que faz o uso de técnicas como filtragem de cores e detecção de contornos. Já a movimentação dos robôs ocorre através do método de campo de vetores unitários juntamente com técnicas de controle clássico. Ao final da pesquisa foi possível obter resultados satisfatórios, tornando os softwares produzidos parte considerável do acervo da recém criada equipe UaiSoccer VSS.

**Palavras-chaves:** Visão Computacional, Controle, Robótica.

# Abstract

Soccer, one of the most practiced sports in the world, also plays an important role in the academic environment. Its application in robotics has been widely researched due to the ability that the sport has to attract students, being the soccer of robots of the category IEEE Very Small Size (VSS) a way to practice different areas of knowledge. In this sense, this work describes a combination of mapping and control methods for VSS robots. They are recognized through a computer vision module that makes the use of techniques such as color filtering and contour detection. On the other hand, the movement of robots occurs through the univector field method along with classical control techniques. At the end of the research it was possible to obtain satisfactory results, making the software produced here considerable part of the collection of the newly created UaiSoccer VSS team.

**Key-words:** Computer Vision, Control, Robotics.

# Listas de ilustrações

Figura 1 – Representação da categoria RoboCup Soccer Simulation 2D. Extraída de RoboCup (2019).	16
Figura 2 – Representação da categoria KidSize da RoboCup Humanoid League. Extraída de RoboCup (2019).	16
Figura 3 – Representação do funcionamento de uma partida de futebol de robôs da categoria VSS. Extraída de Mehl et al. (2001)	17
Figura 4 – Processo de Identificação dos Objetos. Extraída de Chaves (2015).	19
Figura 5 – Aplicação do campo potencial de Khatib. Extraída de Faria (2006).	20
Figura 6 – Mínimos Locais gerados pelo campo potencial de Khatib. Extraída de Faria (2006)	20
Figura 7 – Aplicação do Campo Potencial Harmônico. Extraída de Faria (2006)	21
Figura 8 – Representação do funcionamento do sistema proposto.	23
Figura 9 – Câmera <i>PlayStation Eye</i>	24
Figura 10 – Bola de golf utilizada no trabalho	24
Figura 11 – Dimensões do campo de futebol utilizado. Extraída e editada de IEEE (2009).	25
Figura 12 – Robôs utilizados durante a pesquisa.	25
Figura 13 – Modelo de um robô diferencial. Extraída e editada de Porto (2007).	26
Figura 14 – Sequência dos passos aplicados no desenvolvimento do módulo de visão computacional.	27
Figura 15 – Comparação do modelo de cor RGB e HSV. Extraída de Roza et al. (2016)	29
Figura 16 – Representação das dependências criadas para cada cor de time.	32
Figura 17 – Campo vetorial resultante da Equação 4.19. Extraído de Lim et al. (2008)	36
Figura 18 – Aplicação da transformação projetiva na imagem capturada.	41
Figura 19 – Canais explícitos de uma imagem em RGB e HSV.	42
Figura 20 – Aplicação da operação de abertura em uma imagem com ruídos.	43
Figura 21 – Filtro de kalman discreto aplicado para filtragem e previsão da posição da bola.	43
Figura 22 – Filtro de kalman discreto aplicado para filtragem e previsão da posição de um robô.	44
Figura 23 – Interface de calibração de cores para o sistema de visão computacional.	44
Figura 24 – Janela principal do sistema de visão computacional.	45
Figura 25 – Avaliação do controle do robô através da curva Lemniscata de Bernoulli	47

Figura 26 – Condução da bola para o gol esquerdo sem obstáculos.	48
Figura 27 – Condução da bola para o gol direito com um obstáculo.	48
Figura 28 – Condução da bola para o gol direito com três obstáculos (a).	49
Figura 29 – Condução da bola para o gol direito com três obstáculos (b).	49
Figura 30 – Condução da bola para o gol direito com cinco obstáculos.	50
Figura 31 – Situação de falha na detecção dos pares de cores.	51
Figura 32 – Situação de falha do modelo de controle.	52

# **Lista de tabelas**

Tabela 1 – Taxa de identificação dos objetos . . . . .	45
Tabela 2 – Tempos de execução do módulo de visão computacional por etapas. . .	46

# **Lista de abreviaturas e siglas**

UFSJ	Universidade Federal de São João del Rei
RoboCup	Robot World Cup Initiative
FIRA	Federation of International Robot Soccer Association
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
VSS	IEEE <i>Very Small Size Soccer</i>
LARC	<i>Latin American Robotics Competition</i>
DCOMP	Departamento de Ciência da Computação
CyRoS	Núcleo de Robótica e Tecnologias Assistivas da UFSJ
UDP	<i>User Datagram Protocol</i>
RGB	<i>Red, green, blue</i>
fps	<i>frames per second</i>
DLT	Transformação Linear Direta
HSV	<i>Hue, saturation, value</i>
JSON	<i>JavaScript Object Notation</i>
PID	Controlador Proporcional Integral Derivativo
PI	Controlador Integral
PD	Controlador Derivativo

# Sumário

<b>1</b>	<b>Introdução . . . . .</b>	<b>13</b>
<b>2</b>	<b>O Futebol de Robôs . . . . .</b>	<b>15</b>
2.1	<i>RoboCup Soccer Simulation</i> . . . . .	15
2.2	<i>RoboCup Humanoid League</i> . . . . .	16
2.3	<i>IEEE Very Small Size Soccer</i> . . . . .	16
2.3.1	A Equipe UaiSoccer VSS . . . . .	17
<b>3</b>	<b>Revisão da Literatura . . . . .</b>	<b>18</b>
<b>4</b>	<b>Metodologia . . . . .</b>	<b>23</b>
4.1	Materiais . . . . .	24
4.1.1	Câmera . . . . .	24
4.1.2	Bola . . . . .	24
4.1.3	Campo . . . . .	25
4.1.4	Robôs . . . . .	25
4.1.4.1	Modelo Cinemático dos Robôs . . . . .	26
4.2	Módulo de Visão Computacional . . . . .	27
4.2.1	Transformação Projetiva . . . . .	28
4.2.2	Conversão para HSV . . . . .	29
4.2.3	Filtro por <i>Threshold</i> . . . . .	30
4.2.4	Remoção de Ruídos . . . . .	30
4.2.5	Identificação de Contornos . . . . .	31
4.2.6	Identificação dos Robôs . . . . .	31
4.2.6.1	Cálculo da Posição dos Robôs . . . . .	32
4.2.7	Identificação e Cálculo da Posição da Bola . . . . .	32
4.2.8	Filtragem dos dados . . . . .	32
4.2.9	Envio de dados . . . . .	34
4.3	Navegação . . . . .	35
4.4	Controle . . . . .	38
4.4.1	Ajuste e Sintonia dos Controladores . . . . .	40
<b>5</b>	<b>Resultados . . . . .</b>	<b>41</b>
5.1	Módulo de Visão Computacional . . . . .	41
5.1.1	Transformação Projetiva . . . . .	41
5.1.2	Conversão para HSV . . . . .	41

5.1.3	Filtros por <i>Threshold</i> e Remoção de Ruídos . . . . .	42
5.1.4	Filtragem de Dados . . . . .	42
5.1.5	Interface Gráfica . . . . .	44
5.1.6	Desempenho do Sistema . . . . .	45
5.2	Módulo de Navegação e Controle . . . . .	46
5.2.1	Avaliação do Controle . . . . .	46
5.2.2	Aplicação do Campo Vetorial . . . . .	47
5.3	Discussões e Propostas para Trabalhos Futuros . . . . .	50
<b>6</b>	<b>Conclusão</b> . . . . .	<b>54</b>
<b>Referências</b> . . . . .		<b>55</b>

# 1 Introdução

Recentemente, o desenvolvimento de tecnologias capazes de auxiliar humanos a desempenharem trabalhos tem sido alvo de variadas pesquisas, no qual o uso de robôs móveis tem demonstrado ser uma alternativa viável para ambientes insalubres ou que demandem trabalhos repetitivos e cansativos, como os praticados em laboratórios e na agricultura (ARESTEGUI, 2009).

Nesse contexto, surge a necessidade do estudo e prática da robótica, garantindo assim profissionais preparados para lidar com o avanço tecnológico da mesma. Percebendo-se essa realidade, a *Robocup Federation*, o IEEE e outras associações promovem anualmente e em diversos países competições de robótica como forma de motivar o estudo da mesma (BAZYLEV et al., 2014). Dentre as competições, a de futebol, um dos esportes mais falados e apreciados no mundo todo, tem o poder de aumentar o desempenho do pensamento crítico e analítico de estudantes (ZAINAL et al., 2012).

Atualmente existem várias modalidades de competições de futebol de robôs, as quais abrangem conhecimentos como visão computacional, navegação, controle dinâmico, redes de computadores, dentre outros. Tais conhecimentos são essenciais para o desenvolvimento de sistemas mais robustos e muitas vezes podem ser compreendidos através de comportamentos naturais dos seres vivos.

Humanos, por exemplo, ao realizarem qualquer tarefa utilizam de sentidos como a visão e a audição para se posicionarem em um determinado local e se locomoverem. No mesmo contexto, os robôs também precisam usufruir de algum tipo de informação como forma de entrada para os algoritmos executados. Tais informações podem ser obtidas através de tecnologias como sensores de distância, luminosidade e similares, ou então de informações advindas do processamento de imagens de uma câmera em posição capaz de coletar todas as informações de múltiplos robôs de uma só vez, sendo essa uma alternativa mais barata e eficiente do que o uso dos vários tipos de sensores presentes no mercado. Tal alternativa faz o uso de técnicas de visão computacional, sendo uma definição tradicional dessa área apresentada por Trivedi e Rosenfeld (1989): “visão computacional é a disciplina que investiga as questões computacionais e algorítmicas associadas à aquisição, ao processamento e à compreensão de imagens”.

Nesse contexto, a visão computacional aparece como parte essencial de um sistema colaborativo, processando as imagens capturadas e gerando as informações necessárias para cada robô. Essa colaboração demonstra ser de suma importância para a evolução da pesquisa em robótica, sendo hoje o futebol de robôs uma forma de se avaliar estratégias de interação dos agentes robóticos (WU et al., 2013).

Dentre tantas categorias de futebol de robôs que fazem o uso de visão computacional, a categoria IEEE *Very Small Size Soccer* (VSS) (IEEE, 2009) se destaca por utilizar robôs de pequeno porte com hardware de menor complexidade, sendo ideal para pesquisas focadas majoritariamente no desenvolvimento e aplicação de algoritmos em modelos reais. Dessa forma, o objetivo deste trabalho é relatar o desenvolvimento de um sistema para mapeamento e controle de robôs da categoria VSS, motivando assim a prática da mesma na UFSJ e, consequentemente, colaborando para o aperfeiçoamento de sistemas mais robustos.

Para a validação dos métodos aqui propostos, foram realizados testes utilizando robôs reais, considerando fatores como a velocidade de processamento e a precisão dos movimentos realizados por cada robô dados os respectivos comandos.

Este trabalho está organizado da seguinte forma. No Capítulo 2 é apresentado sobre o futebol de robôs e a categoria VSS. No Capítulo 3 está a revisão da literatura. A metodologia está descrita no Capítulo 4 e os resultados no Capítulo 5, juntamente com as propostas para trabalhos futuros. Por fim, no Capítulo 6, está presente a conclusão.

## 2 O Futebol de Robôs

A ideia de se usar pequenos robôs em uma partida de futebol foi proposta inicialmente na Coréia do Sul em 1996 pelo professor *Jong-Kwan Kin*, do KAIST (*Korean Advanced Institute of Science and Technology*) (MEHL et al., 2001). A proposta foi vista como uma atividade científica que integra uma grande variedade de tópicos ligados à educação e tecnologias. Com isso o Futebol de Robôs se tornou uma área de pesquisa popular e houve a necessidade da criação de regras de modo a garantir harmonia dentre as equipes (SCHWARTZ et al., 2003).

Por consequência, surgiram duas instituições: A RoboCup e a FIRA (*Federation of International Robot Soccer Association*), que estipularam regras para que fosse possível realizar torneios de robótica. Entre as diversas competições existentes no mundo, destaca-se a LARC (*Latin American and Brazilian Robotics Competition*), um evento de robótica que ocorre uma vez ao ano em algum país da América do Sul, habitualmente o Brasil.

A LARC abrange entusiastas da robótica que vão desde estudantes do ensino fundamental até doutores e pós doutores. Além do futebol, dentro do evento são praticadas categorias de robótica em que robôs autônomos realizam atividades como resgate de vítimas em ambientes desastrosos, auxílio doméstico, dança, corrida, dentre outras. Na LARC, as competições de futebol de robôs são divididas em categorias como simulação, (*RoboCup Soccer Simulation 2D / 3D*), robôs humanóides (*RoboCup Humanoid League / Standart Plataform League*) e robôs que utilizam rodas (*RoboCup Small Size / IEEE Very Small Size Soccer*). As seções a seguir detalham algumas categorias de futebol de robôs presentes na LARC.

### 2.1 *RoboCup Soccer Simulation*

A *RoboCup Soccer Simulation* é uma modalidade de futebol de robôs simulada. Nesta modalidade 11 jogadores virtuais competem contra outros 11 de uma equipe adversária de forma autônoma através de um servidor chamado *SoccerServer*. Esse servidor fornece dados para os robôs baseados em sensores virtuais e recebe as ações de cada time, podendo assim monitorar a partida e exibir o estado de jogo em um monitor para o público presente (LARC; CBR, 2019a).

A modalidade de futebol de robôs simulado se divide em duas categorias: 2D e 3D. Na categoria 3D o realismo é aumentado, e os robôs passam a ter que lidar com desafios de comportamentos básicos, como chutar a bola, caminhar, se levantar, dentre outros (LARC; CBR, 2019b). A Figura 1 exibe uma partida de futebol de robôs da categoria 2D.



Figura 1 – Representação da categoria RoboCup Soccer Simulation 2D. Extraída de [RoboCup \(2019\)](#).

## 2.2 *RoboCup Humanoid League*

A *RoboCup Humanoid League* é uma categoria em que os robôs possuem características próximas a de humanos. Nesta categoria cada time pode ter até 4 robôs que devem ser equipados com sensores que imitam os sentidos humanos. A partida ocorre em um campo com dimensões de 675 cm × 450 cm e utiliza uma bola similar a utilizada por humanos no futebol. A Figura 2 exibe uma partida de futebol de robôs da categoria *RoboCup Humanoid League*.

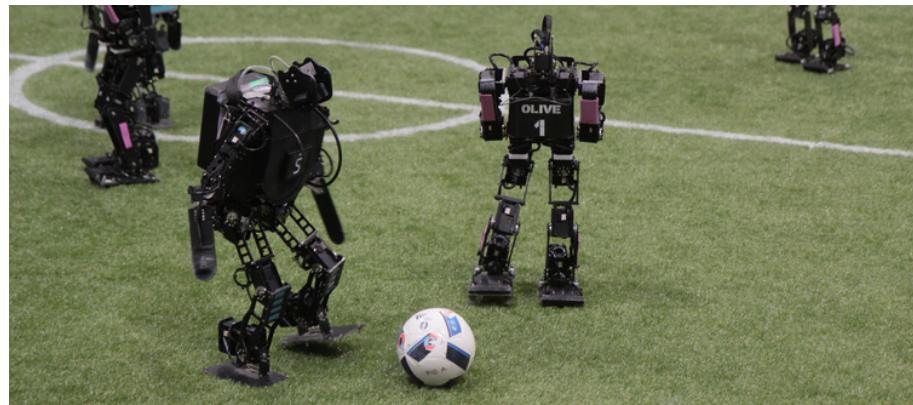


Figura 2 – Representação da categoria KidSize da RoboCup Humanoid League. Extraída de [RoboCup \(2019\)](#).

## 2.3 *IEEE Very Small Size Soccer*

A categoria de futebol de robôs VSS é realizada no Brasil desde o ano de 2003 e é praticada oficialmente na LARC. Basicamente, cada equipe participante deve possuir 3 robôs com tamanho delimitado a um cubo de 8 cm × 8 cm × 8 cm capazes de disputarem uma partida de futebol contra outra equipe. Os robôs devem ser controlados remotamente e de forma autônoma através de um computador, o qual executa um sistema de visão computacional que tem como entrada as imagens de uma câmera suspensa sobre o campo.

O sistema de visão deve ser produzido pela equipe e não deve haver intervenção humana no momento da execução, o período da partida ([IEEE, 2009](#)).

Para que o algoritmo possa identificar os robôs, cada um deles deve possuir um esquema de identificação por cor em sua parte superior. No início da partida irá haver um sorteio definindo a cor de cada time, que pode ser azul ou amarelo. Assim, cada robô deverá conter uma etiqueta contendo a cor do time mais outra etiqueta com uma cor que o identifique (robô 1, 2 ou 3). A cor de identificação do robô pode ser qualquer uma diferente de laranja - a cor da bola - azul ou amarelo. Tais etiquetas podem ser de qualquer formato, sendo exigido apenas que possuam no mínimo uma região equivalente a um quadrado de 3.5 cm de lado, ou um círculo com 4 cm de diâmetro. As etiquetas também não devem ultrapassar o tamanho do robô ([IEEE, 2009](#)). A Figura 3 ilustra a forma em que é realizado um jogo de futebol de robôs nessa categoria.

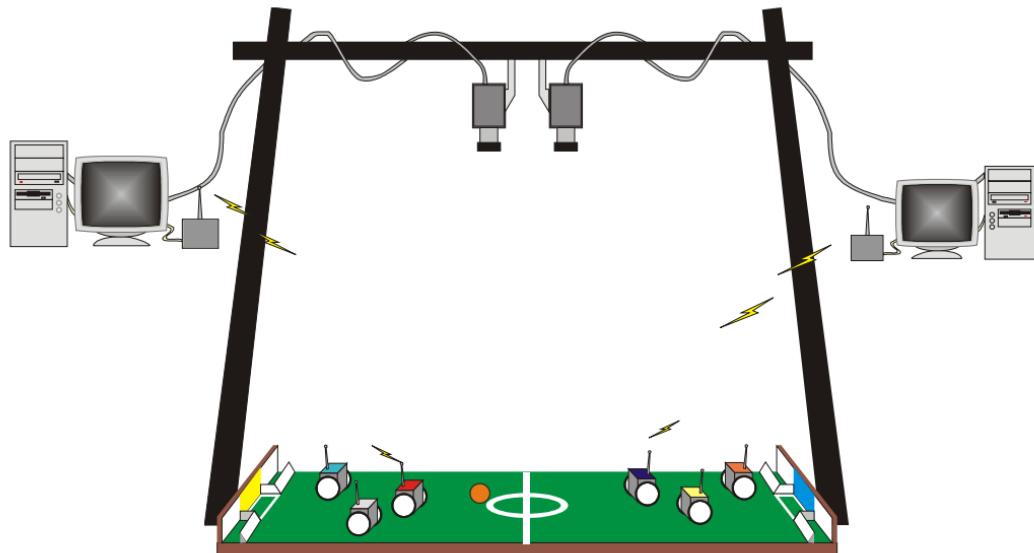


Figura 3 – Representação do funcionamento de uma partida de futebol de robôs da categoria VSS. Extraída de [Mehl et al. \(2001\)](#)

### 2.3.1 A Equipe UaiSoccer VSS

A UaiSoccer VSS é uma equipe de robótica que compete na categoria de futebol de robôs VSS. O projeto nos moldes atuais começou a ser planejado no ano de 2016 por uma iniciativa de membros do Departamento de Ciência da Computação (DCOMP) da UFSJ que inicialmente tinham como objetivo apenas o desenvolvimento de pesquisa e a aplicação de algoritmos existentes em robôs reais. Em 2017 o projeto se associou a equipe de competição da UFSJ UaiSoccer, no qual foi decidido que a nova equipe se chamaria UaiSoccer VSS, como uma referência a categoria em questão. No ano de 2018 a equipe participou da primeira competição de robótica: a LARC 2018. Atualmente a UaiSoccer VSS está situada no DCOMP e é composta por 6 alunos.

### 3 Revisão da Literatura

O futebol de robôs é amplamente usado em pesquisas como, por exemplo, as que envolvem comportamento de robôs em um ambiente controlado, o que torna o tema bastante popular. Os trabalhos presentes na literatura se dividem nas 3 principais etapas do desenvolvimento de um sistema para o futebol de robôs: Visão Computacional, Navegação e Controle.

Quanto aos trabalhos de visão computacional, são utilizadas técnicas que fazem reconhecimento de objetos por forma e/ou cor. [Martins et al. \(2007\)](#) propuseram o desenvolvimento de um sistema de visão computacional para o futebol de robôs da categoria VSS, no qual utiliza como base a subtração de imagens para extrair os elementos de interesse presentes em campo. Para isso os autores coletam um quadro de vídeo antes do início do jogo, sem robôs presentes em campo, e depois esse é subtraído de cada novo quadro processado. Os autores utilizaram o método de *Hough Circles* ([YIP et al., 1992](#)) para auxiliar na identificação das etiquetas dos próprios robôs, que são arredondadas, e a segmentação por cor para identificar as etiquetas dos robôs oponentes, que podem utilizar diferentes formatos de etiquetas. A combinação da técnica de *Hough Circles* com filtragens por cor também é adotada por [Araujo et al. \(2008\)](#).

Já [Chaves \(2015\)](#) utilizou em seu trabalho uma metodologia que faz o reconhecimento dos objetos por cor e área, o que torna o sistema capaz de reconhecer etiquetas de diferentes formas e ainda assim ser robusto, pois insere verificações de área nas etiquetas. O autor propôs a seguinte sequência de passos para o desenvolvimento do sistema: captura da imagem, transformação, conversão para HSV, identificação dos objetos (ilustrado na Figura 4), conversão de coordenadas, detecção de bolas, detecção dos robôs e envio dos dados. Para a parte de detecção dos robôs o autor utilizou a técnica proposta por [Martins et al. \(2007\)](#) que cria árvores de dependências através da distância euclidiana para cada etiqueta de time e depois encontra os pares de cores através da remoção de arestas, conforme ilustrado na Figura 16.

[Chaves \(2015\)](#) destacou a importância de se realizar otimizações na etapa de transformação da imagem tal como também a implementação do filtro de Kalman ([KALMAN, 1960](#)) para redução de ruídos e previsão das posições dos objetos em casos de falhas no reconhecimento.

O trabalho de [Rosa \(2015\)](#) engloba todas as partes do desenvolvimento de um time de futebol de robôs da categoria VSS. Quanto a etapa de visão computacional, foi utilizada uma metodologia próxima a de [Chaves \(2015\)](#), mas com a ausência de algumas etapas, como a transformação projetiva e a conversão para o espaço de cores HSV. Quanto

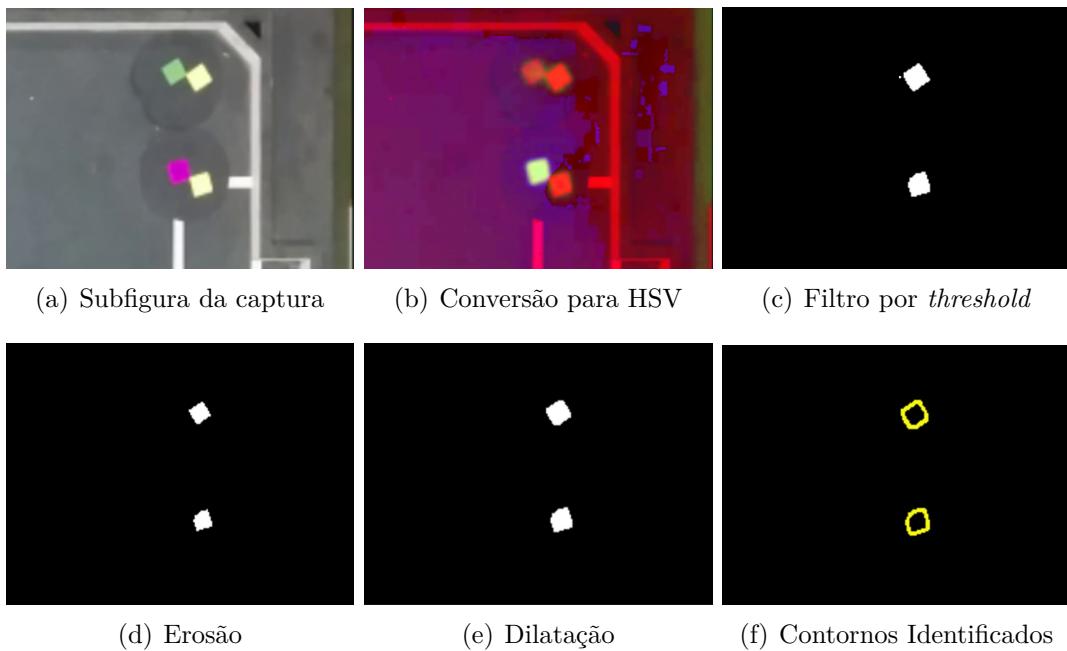


Figura 4 – Processo de Identificação dos Objetos. Extraída de [Chaves \(2015\)](#).

a navegação, o autor abordou dois métodos que fazem o uso de campos potenciais: o Khatib ([KHATIB, 1986](#)) e o Goodrich ([GOODRICH, 2002](#)).

[Khatib \(1986\)](#) foi o primeiro a propor a ideia de se utilizar campos elétricos atuando sobre um robô. A implementação do Khatib é simples e de baixo custo computacional, podendo ser dividida em três etapas: cálculo do campo repulsivo, cálculo do campo atrativo e cálculo da força resultante. O cálculo da força repulsiva  $\vec{F}_r$  entre um robô e um obstáculo é realizado com base na distância  $d$  entre ambos e na disposição angular dos mesmos. A Equação 3.1 exibe como o campo repulsivo pode ser obtido, dividindo-o em módulo e direção.  $Q$  representa uma constante escalar ajustável. No caso de mais de uma força repulsiva, basta realizar o somatório dos vetores.

$$|\vec{F}_r| = \frac{Q}{d^2} \quad \theta_{\vec{F}_r} = -\tan^{-1} \left( \frac{\Delta y}{\Delta x} \right) \quad (3.1)$$

Para o campo atrativo a força de atração deve ser constante para que o robô possa ser atraído de qualquer parte de campo, como pode ser visto na Equação 3.2, em que agora  $\Delta x$  e  $\Delta y$  e correspondem à relação entre o robô e o alvo.

$$|\vec{F}_a| = c \quad \theta_{\vec{F}_a} = \tan^{-1} \left( \frac{\Delta y}{\Delta x} \right) \quad (3.2)$$

A força resultante é definida pela soma vetorial de  $\vec{F}_r$  e  $\vec{F}_a$ . A Figura 5 exibe o resultado dos campos potenciais de Khatib.

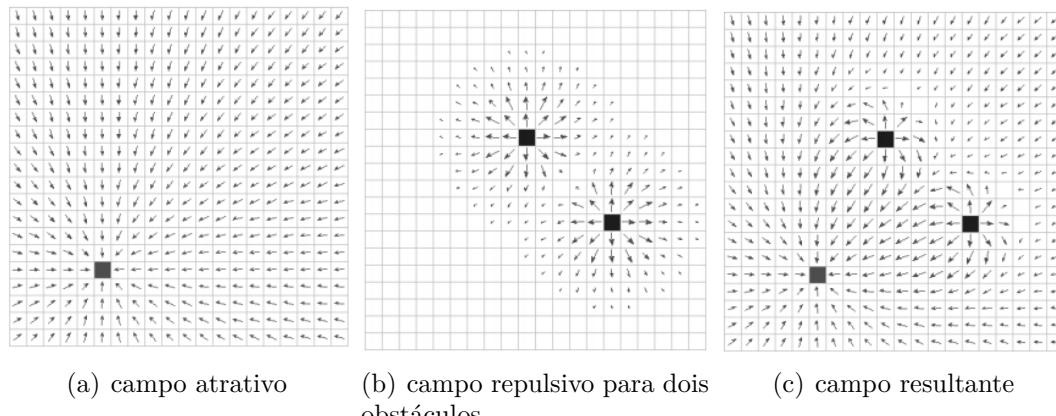


Figura 5 – Aplicação do campo potencial de Khatib. Extraída de [Faria \(2006\)](#).

Apesar dos campos potenciais de Khatib serem de fácil implementação e demandarem pouco poder computacional, os mesmos possuem problemas com mínimos locais, que ocorrem em situações em que as forças de atração e repulsão se anulam, conforme pode ser visto na Figura 6. No trabalho desenvolvido por [Connolly et al. \(1990\)](#) esse problema foi resolvido utilizando funções harmônicas para o cálculo do campo potencial em ambientes nos quais as posições dos objetos sejam conhecidas ([SILVA, 2011](#)). Tais funções são soluções para a equação de Laplace, e possuem propriedades importantes no desenvolvimento de sistemas de controle robótico, como a integridade e a robustez ([FARIA, 2006](#)). A integridade se diz respeito à garantia que o método oferece em encontrar um caminho até o objetivo, quando ele existe. Já a robustez é dada pela capacidade do método em lidar com obstáculos que não são conhecidos a priori. A Figura 7 exibe uma ilustração dos campos potenciais harmônicos para os mesmos ambientes da Figura 6.

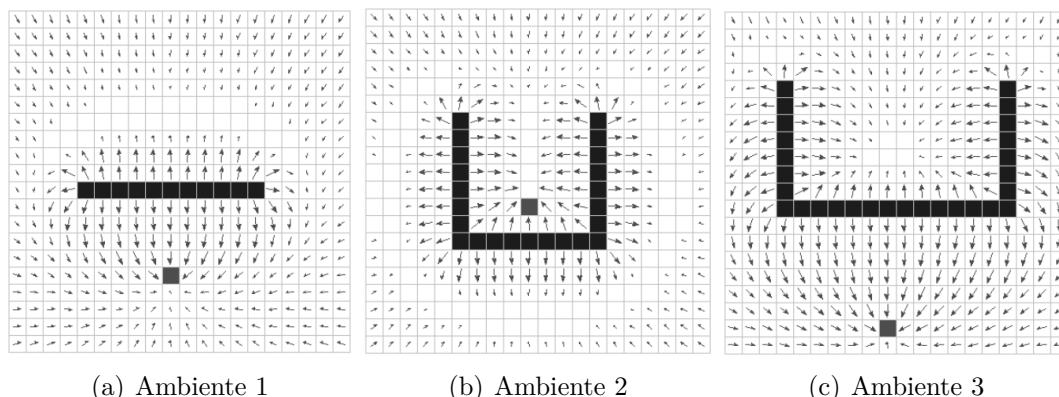


Figura 6 – Mínimos Locais gerados pelo campo potencial de Khatib. Extraída de [Faria \(2006\)](#)

Existem também outros métodos de navegação, como os Campos Potenciais Orientados ([JÚNIOR, 2003](#)), os Campos Potenciais Localmente Orientados ([FARIA, 2006](#)) e métodos que utilizam outras técnicas, como grafos ([REIS et al., 2013](#)). Os campos

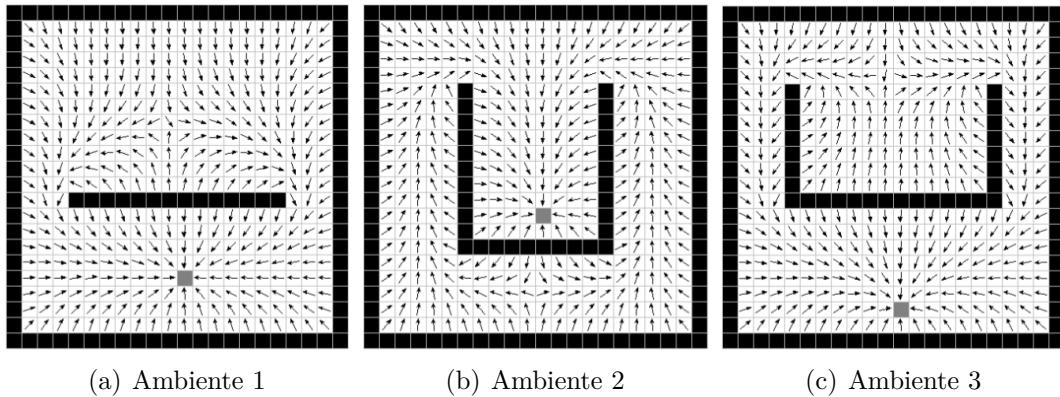


Figura 7 – Aplicação do Campo Potencial Harmônico. Extraída de [Faria \(2006\)](#)

potenciais orientados são extremamente úteis no futebol de robôs, pois permitem que os robôs possam chegar a um objetivo com determinada orientação, evitando assim que robôs possam chutar em gols contrários ao se direcionarem até a bola. Dentre os métodos de navegação que levam em consideração a orientação ao objetivo, está o campo de vetores unitários ([LIM et al., 2008](#)). Esse método, além de ser um campo orientado, foi desenvolvido especialmente para ambientes dinâmicos, sendo ideal para o futebol de robôs.

O estudo sobre métodos de Navegação para robôs móveis é extenso, e por isso existem trabalhos dedicados inteiramente a este tema ([SILVA, 2011](#)), ([FARIA, 2006](#)), ([PIRES, 2016](#)). Apesar da complexidade dos diversos métodos de navegação existentes, os mesmos se restringem apenas em definir uma trajetória a ser seguida, não abrangendo, por exemplo, a tarefa de decompor tal trajetória em velocidades para os robôs.

Esta tarefa é usualmente realizada baseada nos fundamentos da teoria clássica de controle para sistemas de malha fechada, em que faz o uso de um ou mais sensores para mensurarem e controlarem a saída de determinado atuador. Mas para isso é necessário também a entrada de um valor de referência, que no contexto deste trabalho, define qual a ação esperada de cada robô.

O valor de referência pode ser obtido de diferentes formas a variar de acordo com o modelo cinemático de cada robô e da arquitetura de cada sistema, conforme pode ser visto em [Santos et al. \(2018\)](#), [AI-Ammri e Ahmed \(2010\)](#), [Kamada et al. \(2018\)](#) e [Tavares et al. \(2015\)](#). Já o valor mensurado varia de acordo com o tipo de sensor utilizado. O modelo mais comum são os sensores de giro (*encoders*). No entanto também é possível utilizar os dados oriundos da câmera, como orientação e posição do robô, para retroalimentar o sistema de controle ([ROSA, 2015](#)).

Dado o valor de referência e o valor mensurado, o controlador é quem vai produzir o valor de saída, buscando assim corrigir possíveis imperfeições nos robôs e reagir a imprevistos que por ventura podem atrapalhar o atuador a cumprir com o valor de referência.

As teorias e técnicas de projeto de controle podem ser divididas em duas categorias: métodos de controle clássico, que são baseados no uso das transformadas de Fourier e Laplace, e métodos de controle moderno, que são baseados em equações diferenciais na forma de espaço de estados ([FRANKLIN et al., 2013](#)). Dentre os métodos de controle clássico pode ser destacado o controlador proporcional integral derivativo (PID) e dentre os métodos de controle moderno, o filtro de Kalman ([KALMAN, 1960](#)).

Os métodos de navegação em conjunto com as técnicas de controle são responsáveis por encaminharem cada robô a um objetivo, se atentando a restrições como a presença de obstáculos, possíveis imperfeições em um robô e a necessidade de uma orientação na chegada ao objetivo. No desenvolvimento de um sistema completo para o futebol de robôs existe ainda uma camada acima, chamada de estratégia. A estratégia é a parte do software que determina o comportamento e também o destino final para cada robô controlado, determinando se o mesmo deve atuar como um defensor, um atacante ou até mesmo um goleiro. No VSS esta etapa é de menor complexidade se comparada com outras categorias que possuem uma quantidade maior de robôs por equipe. Com apenas três robôs em campo, no qual um atua como goleiro, o desafio da estratégia pode ser resumido na tomada de decisão para o comportamento dos outros dois robôs.

O trabalho de desenvolver uma estratégia é bem próximo ao trabalho realizado pelo técnico de um time de futebol. Não existem estratégias perfeitas, pois o desempenho de determinada equipe, ao que se diz respeito ao posicionamento dos jogadores, vai sempre estar relacionado ao comportamento dos oponentes. Para isso, [Caputo \(2014\)](#) construiu um classificador utilizando redes bayesianas com o objetivo de escolher a melhor jogada em cada momento do jogo. O classificador foi construído para a categoria *Small Size League*, e a base de dados foi montada utilizando os logs de dados da final do torneio da RoboCup, que são disponibilizados para essa categoria específica.

Neste trabalho a etapa da estratégia não será abordada. A metodologia do trabalho é embasada nos trabalhos apresentados nessa seção e focada no mapeamento de todos os robôs dentro do campo e na movimentação de um único robô, que engloba as etapas de navegação e controle.

## 4 Metodologia

Antes de iniciar a programação dos robôs, foi necessário realizar um planejamento para a definição de uma arquitetura de software que permitisse fácil manutenção, alto desempenho e integração de outros utilitários. Dessa forma, foi optado pela criação de um sistema separado em três grandes módulos.

Cada módulo presente no sistema é responsável por um conjunto de tarefas específicas e todos se relacionam como "caixas pretas", no qual um módulo apenas recebe a saída de outro, sem ter conhecimento de como tal informação foi gerada. O diagrama da Figura 8 ilustra como ocorre a interação dos mesmos.

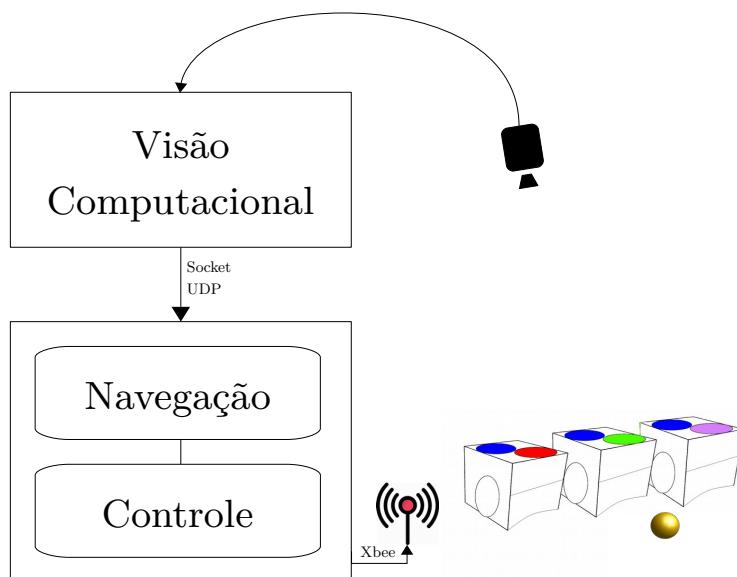


Figura 8 – Representação do funcionamento do sistema proposto.

O primeiro módulo é o de Visão Computacional, responsável por receber as imagens da câmera e extrair informações úteis para o módulo seguinte. Essa etapa é melhor descrita na Seção 4.2. O segundo módulo é o de Navegação e Controle, responsável por gerar trajetórias e enviar comandos para cada robô. Esse módulo está descrito nas Seções 4.3 (Navegação) e 4.4 (Controle). Já o último módulo representa os robôs que atuam em campo. O desenvolvimento dos mesmos não é abordado neste trabalho. No entanto, o funcionamento dos robôs e o modelo cinemático dos mesmos está descrito de forma breve na Seção 4.1, juntamente com os outros materiais utilizados nesta pesquisa.

A organização ilustrada na Figura 8 permite integrar outros softwares no sistema, como por exemplo os simuladores de partidas de futebol, muito úteis em casos em que os robôs reais estão indisponíveis, no qual um simulador substituiria o primeiro e terceiro módulo. Também permite que uma mesma câmera possa ser usada por dois times de forma simultânea, uma vez que o módulo de visão computacional pode transmitir os dados em

modo *multicast*.

Todos módulos foram implementados utilizando a linguagem C++, que possui alto desempenho, e se comunicam via rede. Os dois primeiros módulos, que executam em um mesmo computador, se comunicam através do protocolo UDP (*User Datagram Protocol*) e o segundo módulo se comunica com os robôs através de rádios XBee. O UDP é um protocolo da camada de transporte da pilha TCP/IP e opera baseado no protocolo IP. Já o XBee é um rádio que opera baseado no protocolo IEEE 802.15.4. Em Resende et al. (2019) é apresentado uma discussão sobre redes sem fio em competições de robótica, no qual o uso da comunicação 802.11 (Wi-Fi) é desencorajado, fator este que foi considerado para a escolha do XBee na comunicação com os robôs.

## 4.1 Materiais

### 4.1.1 Câmera

Para fazer a captura das imagens que atuam como entrada para o módulo de visão computacional, foi utilizada a câmera *PlayStation Eye* fabricada pela empresa *Sony*. A câmera, exibida na Figura 9, é capaz de fornecer 60 quadros RGB (*red, green, blue*) por segundo utilizando uma resolução de 640x480 pixels.



Figura 9 – Câmera *PlayStation Eye*<sup>1</sup>.

### 4.1.2 Bola

Na pesquisa foi utilizada uma bola de golf laranja, conforme exibido na Figura 10.



Figura 10 – Bola de golf utilizada no trabalho<sup>2</sup>

<sup>1</sup> Extraída de <[https://pt.wikipedia.org/wiki/PlayStation\\_Eye](https://pt.wikipedia.org/wiki/PlayStation_Eye)>

<sup>2</sup> Extraída de <<https://www.halfpricegolfballs.com/products/custom-new-orange-blank-balls>>

### 4.1.3 Campo

O ambiente de testes é composto por um mini campo de futebol com características conforme proposto pelas regras da categoria VSS. O campo é um tablado de madeira na cor preta, com dimensões de 150 x 130cm. A Figura 11 exibe as dimensões detalhadas do campo utilizado.

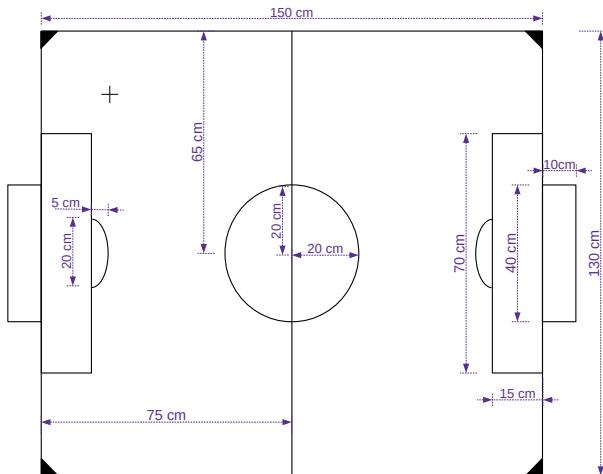


Figura 11 – Dimensões do campo de futebol utilizado. Extraída e editada de [IEEE \(2009\)](#).

### 4.1.4 Robôs

Como atuadores foi utilizado a versão 2019 dos robôs desenvolvidos pelos integrantes da equipe UaiSoccer VSS. Tais robôs foram desenhados computacionalmente, de modo que tiveram suas carcaças impressas por uma impressora 3D. A eletrônica dos robôs é composta pelo micro controlador ESP8266 e um rádio XBee S1. Também possuem uma ponte H para controle dos dois motores presentes em cada robô e outros componentes menores. Na Figura 12 é possível visualizar dois dos robôs utilizados, em que o da direita está com a tampa removida.

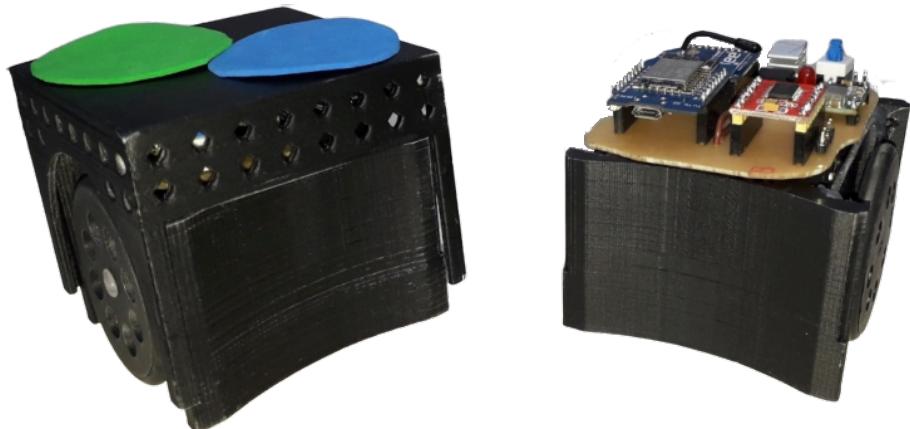


Figura 12 – Robôs utilizados durante a pesquisa.

#### 4.1.4.1 Modelo Cinemático dos Robôs

A cinemática é o estudo mais básico do comportamento de um sistema mecânico. Compreender o modelo cinemático de um robô é essencial para a construção de um controle eficiente (ARESTEGUI, 2009). Os robôs utilizados nesta pesquisa são robôs não holonômicos com tração diferencial, pois possuem restrições de movimentos e apenas duas rodas com tração.

A posição de um robô no instante  $t$  em relação aos eixos de coordenada inercial pode ser definida pelo vetor  $p(t)$  da Equação 4.1, no qual  $(x(t), y(t)) \in R^2$  representa a posição do centro do robô e  $\theta(t) \in [-\pi, \pi]$  representa a orientação do mesmo em relação ao eixo  $x$ , ambos no instante  $t$ .

$$p(t) = [x(t), y(t), \theta(t)] \quad (4.1)$$

Já o vetor velocidade  $\dot{p}(t)$  do robô no instante  $t$  pode ser definido em função da velocidade linear  $v$  e angular  $\omega$  do robô, conforme exibido na Equação 4.2.

$$\dot{p}(t)^T = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (4.2)$$

As velocidades rotacionais das rodas direita  $\omega_d$  e esquerda  $\omega_e$  podem ser obtidas através da Equação 4.3, em que  $r$  representa o raio da roda utilizada e  $b$  representa metade da largura do robô (PORTO, 2007). A Figura 13 ilustra o modelo de um robô diferencial.

$$\begin{bmatrix} \omega_d \\ \omega_e \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & \frac{b}{r} \\ \frac{1}{r} & -\frac{b}{r} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (4.3)$$

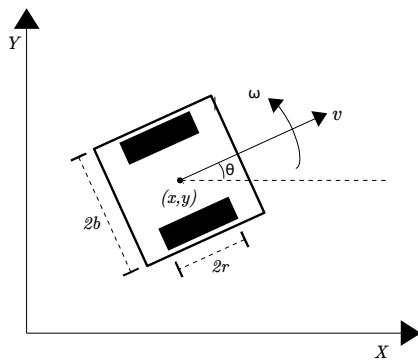


Figura 13 – Modelo de um robô diferencial. Extraída e editada de Porto (2007).

## 4.2 Módulo de Visão Computacional

Além dos materiais listados na seção anterior, também foi necessário realizar a escolha de tecnologias a serem utilizadas no desenvolvimento de cada módulo. No caso do software de visão computacional, além do desempenho, foi definido também como requisito a presença de uma interface gráfica para facilitar a calibração das diversas constantes descritas posteriormente. Sendo assim foi optado pela utilização do *framework Qt*, que permite a criação de interfaces gráficas de forma simples e é compatível com a linguagem C++, que possui alto desempenho.

Também foi utilizado como base do software o OpenCV<sup>3</sup>, que é uma biblioteca multiplataforma de código aberto. A biblioteca oferece suporte a diversas operações de visão computacional e processamento digital de imagens que podem ser paralelizadas automaticamente.

O software desenvolvido identifica cada objeto através de uma câmera posicionada em um local privilegiado a ponto de captar todo o ambiente desejado com boa qualidade. Todos os passos realizados são detalhados a seguir. A Figura 14 ilustra a ordem em que estes passos ocorrem.

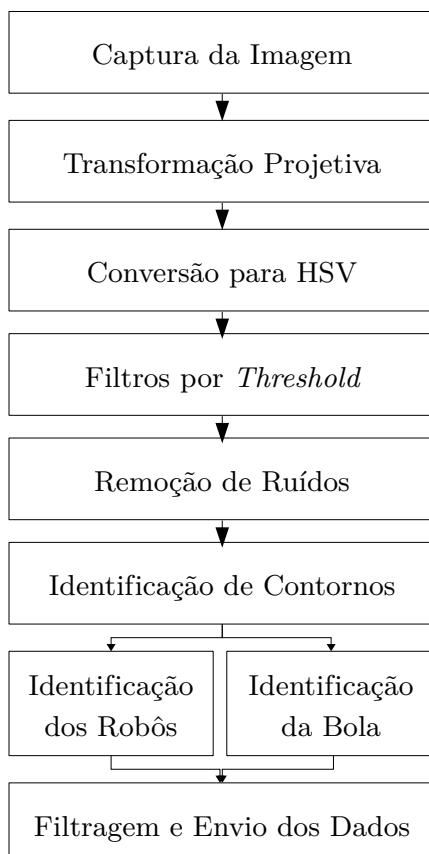


Figura 14 – Sequência dos passos aplicados no desenvolvimento do módulo de visão computacional.

<sup>3</sup> Disponível em: <<https://github.com/opencv/opencv>>

### 4.2.1 Transformação Projetiva

Mesmo posicionando a câmera do melhor modo possível, podem ocorrer distorções na perspectiva da imagem e também captura de áreas não desejadas. A solução para a correção desse problema vem das transformações projetivas da álgebra linear, no qual transforma uma imagem com uma perspectiva não desejada em outra que demonstra o ambiente de captura de forma plana, como se a câmera estivesse em posição totalmente ortogonal ao plano.

Este processo transforma um ponto  $(x, y)$  em outro  $(x', y')$  através da transformação linear. Para isso é preciso de uma matriz de transformação linear  $H$ , como representado na Equação 4.4, e o novo ponto pode ser obtido como na Equação 4.5.

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (4.4)$$

$$x' = \frac{\alpha}{\gamma} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \quad y' = \frac{\beta}{\gamma} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \quad (4.5)$$

Através do algoritmo de Transformação Linear Direta (DLT), é possível encontrar a matriz de transformação linear  $H$ . Sendo assim a Equação 4.5 deve ser rearranjada a fim de se obter a Equação 4.6, em que  $h = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33}]^T$ .

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y & -x' \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y & -y' \end{bmatrix} h = 0 \quad (4.6)$$

Mas para encontrar todos os valores da matriz  $H$  é necessário adicionar mais três pontos a serem mapeados, de forma que a Equação 4.6 se torna a Equação 4.7.

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y_2 & -x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y'_2x_2 & -y'_2y_2 & -y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x'_3x_3 & -x'_3y_3 & -x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y'_3x_3 & -y'_3y_3 & -y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y_4 & -x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4x_4 & -y'_4y_4 & -y'_4 \end{bmatrix} h = 0 \quad (4.7)$$

A Equação 4.7 está no formato  $Dh = 0$ , em que  $D$  é uma matriz de ordem  $8 \times 9$ . Dado a posição de origem e destino dos quatro pontos, será possível solucionar o sistema linear resultante da Equação 4.7 e então encontrar todos os valores de  $h$ , que correspondem aos valores de  $H$ , para finalmente aplicar as equações 4.4 e 4.5 para todo ponto da imagem.

[Chaves \(2015\)](#) propõe a realização dessa operação fornecendo os quatro pontos referentes as extremidades do campo e inserindo-os na função `getPerspectiveTransform()` da biblioteca OpenCV, que gera uma matriz de transformação linear para posteriormente ser aplicada para cada frame de vídeo capturado através da função `warpPerspective()`, também do OpenCV. Porém, como relatado pelo próprio autor, essa não é uma alternativa eficiente. A função `warpPerspective()` aplica a Equação 4.5 para todo ponto da imagem sempre que é chamada, mas como a câmera não se move durante o jogo, é possível realizar esse processo apenas uma vez e gerar um mapeamento de origem e destino para todos os pixels da imagem. Dessa forma, sempre que um novo quadro é capturado, o algoritmo deve apenas consultar esse mapa, reduzindo assim o custo computacional da transformação projetiva. A operação de consulta e aplicação da transformada pode ser feita através da função `remap()` do OpenCV e é o método utilizado por este trabalho.

#### 4.2.2 Conversão para HSV

Cada quadro de vídeo capturado pela câmera possui codificações de cores no formato RGB, e então o algoritmo o converte para o sistema de cores HSV (*hue, saturation, value*).

*Hue* (matiz) indica a tonalidade da cor, abrangendo todas as cores do espectro. Seus valores variam de 0 a 360, mas costumam ser normalizados para outras escalas de acordo com a linguagem e/ou biblioteca utilizada. *Saturation* (saturação) é referente à pureza da cor, quanto mais baixo seus valores, mais a imagem se assemelha a uma em escala de cinza, e *Value* (valor) indica o brilho da cor ([GARCIA, 2011](#)).

Uma das vantagens em utilizar o formato HSV se deve ao fato do sistema de cores possuir componentes explícitos de brilho e saturação, facilitando o processo de normalização da iluminação da imagem quando necessário.

Outra vantagem do uso do formato HSV para filtragens de cores é apresentado por [Neiva et al. \(2015\)](#), no qual com base nos estudos comparativos de [Ohta et al. \(1987\)](#) constata que “[...] o modelo HSV apresentou informações de cor e luminância mais descorrelacionadas que no modelo RGB”, fator que facilita a filtragem de cores. A Figura 15 exibe um comparativo entre os formatos de cores em RGB e HSV

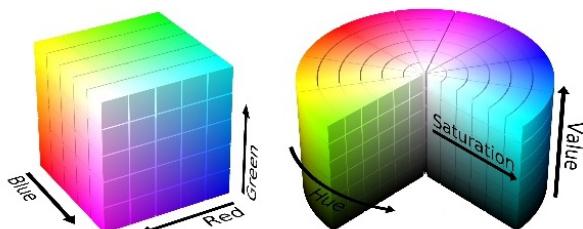


Figura 15 – Comparação do modelo de cor RGB e HSV. Extraída de [Roza et al. \(2016\)](#)

### 4.2.3 Filtro por *Threshold*

O filtro por *Threshold*, também conhecido como limiarização, é um processo utilizado para separar regiões de interesse em uma imagem, no qual a partir de valores mínimos e máximos dos componentes de uma cor, gera uma outra imagem com apenas valores 0 e 255, sendo os pixels representados por 255 a região de interesse.

O trecho de código presente na Listagem 4.1 representa a realização desse processo de forma bit a bit na linguagem C++. É importante ressaltar que antes da aplicação do filtro dessa forma é necessário separar os 3 canais da imagem, no qual no código `img` representa uma classe que os possui como atributos, sendo as matrizes `h`, `s` e `v` para matiz, saturação e valor, respectivamente. `h_min`, `h_max`, `s_min`, `s_max`, `v_min` e `v_max` são as constantes que determinam o intervalo da limiarização e `Bin` representa a imagem resultante.

```

1      H_bin = img.h > h_min & img.h < h_max;
2      S_bin = img.s > s_min & img.s < s_max;
3      V_bin = img.v > v_min & img.v < v_max;
4      Bin = H_bin & S_bin & V_bin;
```

Listagem 4.1 – Aplicação do filtro por threshold de forma bit a bit na linguagem C++.

Com a função `inRange()` da biblioteca OpenCV esse processo pode ser realizado de forma mais simples, não sendo necessário separar os canais da imagem. Devido a esse motivo e também às otimizações próprias da biblioteca a função `inRange()` obteve um tempo de execução melhor do que o método bit a bit, e portanto é a utilizada no algoritmo. No software desenvolvido foi incluído uma interface gráfica com barras de rolagens para que o usuário calibre de forma interativa os valores máximos e mínimos das componentes H, S e V para as cores utilizadas.

### 4.2.4 Remoção de Ruídos

Em uma imagem binária é classificado como ruído todo pixel que não representa uma região de interesse, mas está classificado como tal na mesma, ou seja, marcado pela cor branca. Para a remoção desses ruídos é comum a aplicação de duas operações morfológicas de forma sequencial: a erosão e a dilatação.

A erosão em imagens consiste em eliminar pixels marcados como pertencentes a uma região de interesse, mas que possuem nenhum ou poucos vizinhos. Ou seja, cada pixel branco com um número menor que  $n$  de vizinhos brancos sofrerá a inversão da própria cor. Já a dilatação em imagens pode ser definida como o processo inverso da erosão. Da mesma forma que a erosão causa uma impressão de diminuir os objetos em uma imagem, a dilatação os aumenta. Nesse processo cada pixel preto considerado como vizinho de um pixel branco sofrerá a inversão da própria cor, aumentando assim o tamanho do objeto.

Na aplicação dessas operações o recurso que determina quais pixels são considerados como vizinhos é chamado de elemento estruturante. Um elemento estruturante do tipo cruz e tamanho  $3 \times 3$ , por exemplo, vai considerar como vizinhos apenas o primeiro pixel diretamente acima, abaixo, a esquerda e a direita do pixel em questão. A escolha do tamanho e do formato do elemento estruturante deve considerar as características do objeto de interesse e é essencial para a remoção eficiente dos ruídos.

A aplicação da erosão e dilatação de forma sequencial forma uma nova operação morfológica, denominada de abertura (*opening*). Esse processo pode ser aplicado com as funções `erode()` e `dilate()` da biblioteca OpenCV.

#### 4.2.5 Identificação de Contornos

Uma forma de buscar informações dos objetos selecionados nos passos anteriores é com a detecção de contornos. Esse processo consiste em perseguir as bordas de cada objeto branco da imagem até que os pontos se encontrem. A função `findContours()` do OpenCv realiza esse processo e gera um vetor de contornos. Dessa forma é possível utilizar os momentos invariantes dos objetos contornados para encontrar a área e a localização de cada um deles. A informação sobre as dimensões dos objetos é útil para descobrir se os mesmos são válidos ou não, e a posição é o que possibilita a realização dos passos posteriores.

#### 4.2.6 Identificação dos Robôs

Com a posição central de todas as etiquetas de cores, chega o momento em que é necessário relacionar os pares de cores para coletar a posição central dos robôs. Essa etapa é baseada no modelo proposto por [Martins et al. \(2007\)](#), que cria árvores de dependências entre as cores. Primeiramente é procurado ao redor de cada objeto que representa a cor de uma equipe a presença de etiquetas referentes a robôs, armazenando todas as coordenadas das mesmas em um vetor.

Na melhor das hipóteses é obtida apenas uma correspondência para cada etiqueta referente a um time, ou seja, todos os robôs estão distribuídos no ambiente de captura de forma que a relação dos pares de cores é feita facilmente. No entanto, em casos em que os robôs estão bem próximos entre si essa relação pode demandar a realização de cálculos que determinam prioridades na identificação dos robôs. A Figura 16 ilustra uma situação em que três robôs de um mesmo time estão bem próximos um dos outros, sendo possível notar que, se não realizado tratamentos para esses casos no algoritmo, o sistema pode confundir a posição dos robôs devido a etiqueta 2 estar igualmente próxima a três etiquetas que diferenciam robôs de um mesmo time: etiquetas verde, rosa e vermelho.

A solução utilizada foi calcular primeiramente os robôs no qual a etiqueta referente

ao time possui apenas uma dependência, como no caso das etiquetas 1 e 3 da Figura 16, e depois excluir a cor do robô das dependências das outras etiquetas azuis, que nesse caso é a cor da equipe. Determinando azul e verde como um par de cores, azul e rosa como outro, e eliminando as cores rosa e verde das dependências da etiqueta 2, é possível encontrar o último par de cores, que é azul e vermelho.

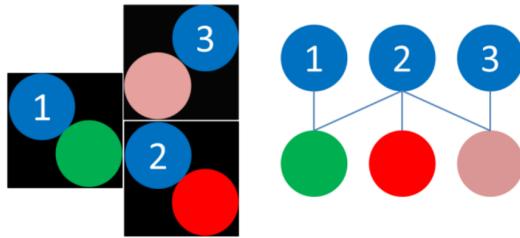


Figura 16 – Representação das dependências criadas para cada cor de time.

#### 4.2.6.1 Cálculo da Posição dos Robôs

Segundo Chaves (2015), considerando robôs com etiquetas dispostas como na Figura 16, com duas cores de centro de massa  $C_1$  e  $C_2$  dispostas a um ângulo  $\vartheta = 45^\circ$ , é possível obter a posição  $(x, y)$  de cada robô e a orientação  $\theta$  através da Equação 4.8.

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} \frac{C_{1x} + C_{2x}}{2} \\ \frac{C_{1y} + C_{2y}}{2} \\ \arctan\left(\frac{C_{2y} - C_{1y}}{C_{2x} - C_{1x}}\right) - \vartheta \end{bmatrix} \quad (4.8)$$

O par  $(x, y)$  corresponde à posição de um robô na imagem, então em seguida esses valores são convertidos para uma escala que independe do número de pixels de cada quadro de vídeo. Vale ressaltar também que a Equação 4.8 deve ser alterada de acordo com a disposição das etiquetas no topo do robô.

#### 4.2.7 Identificação e Cálculo da Posição da Bola

Por possuir apenas uma cor e ser o único objeto laranja em campo, a identificação da bola se torna bem simples se comparada com a dos robôs. Basta realizar os passos anteriores, que são o filtro por *Threshold*, a identificação de contornos e o cálculo do centro de massa para o único objeto laranja presente no campo, que será possível encontrar a localização da bola na imagem e depois converter os valores para a escala desejada.

#### 4.2.8 Filtragem dos dados

Como todo sensor, os dados extraídos da imagem de uma câmera naturalmente são acrescidos de ruídos. Sendo assim é necessário que exista uma filtragem final nos dados

para que as etapas posteriores do processo não sejam prejudicadas por tais ruídos. Para esse fim foi aplicado o filtro de Kalman Discreto de terceira ordem.

O filtro de Kalman foi inicialmente apresentado em [Kalman \(1960\)](#) como uma solução recursiva para o problema de filtragem linear de dados. O propósito do filtro em questão é eliminar ruídos e outras incertezas, sendo considerado um estimador ótimo para uma larga classe de problemas ([LAIA; CRUVINEL, 2008](#)).

O modelo do filtro de Kalman assume que um estado  $\chi$  no passo  $k$  ( $\chi_k$ ), é representado conforme a Equação 4.9, em que  $A$  é a matriz de transição de estados,  $B$  é o modelo das entradas de controle e  $u_k$  é o vetor das entradas de controle.  $q_k$  e  $r_k$  são o ruído do processo e da medição, respectivamente.

$$\chi_k = A\chi_{k-1} + Bu_k + q_k \quad (4.9)$$

O valor observado  $z_k$  é assumido como representado na Equação 4.10, em que  $C$  é a matriz do modelo de observação, que vai transformar o espaço de estados real para o espaço observado.

$$z_k = C\chi_k + r_k \quad (4.10)$$

Os ruídos são assumidos como independentes entre si e com probabilidade normal conforme exibido nas Equações 4.11 e 4.12.

$$q_k \sim N(0, Q_k) \quad (4.11)$$

$$r_k \sim N(0, R_k) \quad (4.12)$$

As matrizes de covariância  $Q_k$  e  $R_k$  dos ruídos são variáveis com o tempo. No entanto, aqui assumimos que elas são constantes e por isso são definidas daqui para frente apenas como  $Q$  e  $R$ . O ajuste inicial das mesmas é essencial para uma boa filtragem.

Em cada passo  $k$  no filtro de Kalman ocorre a atualização de duas variáveis:  $\hat{\chi}_k$ , que é a estimativa posterior do estado e  $P_k$ , a matriz de covariância do erro do estado a posteriori. As equações de Kalman podem ser divididas em atualização do tempo e atualização das medidas. As equações para a atualização do tempo são responsáveis por projetar uma estimativa do estado e da covariância  $P_k$  a priori:  $\hat{\chi}_k^-$  e  $P_k^-$  respectivamente. Já as equações de atualizações das medidas são responsáveis por atualizar as estimativas com base no valor mensurado  $z_k$ , gerando assim as estimativas a posteriori que serão utilizadas para realimentar o filtro no passo  $k+1$ . Dessa forma o algoritmo combina métodos preditores e corretores para a solução de problemas numéricos ([LAIA; CRUVINEL, 2008](#)).

As Equações 4.13 e 4.14 são as equações para a atualização do tempo:

$$\hat{\chi}_k^- = A\hat{\chi}_k + Bu_k \quad (4.13)$$

$$P_k^- = AP_{k-1}A^T + Q \quad (4.14)$$

Já as Equações 4.15, 4.16 e 4.17 são as equações para a atualização das medidas. A matriz  $K_k$  é conhecida como o ganho de Kalman. É ela quem vai ponderar a previsão do estado a priori com o valor mensurado  $z_k$ .

$$K_k = P_k^- C^T (CP_k^- C^T + R)^{-1} \quad (4.15)$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - C\hat{x}_k^-) \quad (4.16)$$

$$\hat{P}_k = (I - K_k C)P_k^- \quad (4.17)$$

O fato do filtro de Kalman utilizado ser de terceira ordem significa que para a filtragem de um objeto em um plano bidimensional, por exemplo, o estado  $\chi$  vai ser formado por 6 variáveis: a posição, a velocidade e a aceleração do objeto em cada uma das duas dimensões do plano. Considerando o objeto inicialmente em repouso, o estado apenas precisa ser inicializado com o primeiro valor observado transformado para o espaço de estados ( $C^T z_1$ ), que o modelo matemático do filtro de Kalman automaticamente é capaz de derivar a posição para obter a velocidade e a aceleração através da derivada segunda.

Na metodologia proposta, em que os módulos são tratados como "caixas pretas", o sistema de visão computacional não tem conhecimento do vetor das entradas de controle  $u_k$ . Sendo assim  $Bu_k$  não foi considerado na Equação 4.13. Nesse trabalho foram realizadas filtragens nas posições da bola e dos robôs. Quanto aos robôs, foi optado por realizar a filtragem na posição de cada etiqueta, ao invés de se aplicar apenas uma filtragem final na posição calculada através da Equação 4.8. A motivação para se aplicar a filtragem nas etiquetas é que dessa forma o resultado do cálculo da orientação do robô estará de forma indireta filtrado também.

Além de filtrar os dados para a remoção de ruídos, o filtro de Kalman também foi utilizado para prever a posição dos objetos quando por alguma situação inesperada o sistema não foi capaz de detectá-los. Sendo assim foi utilizado a predição do estado a priori (Equação 4.13) nessas situações. Essa é uma solução eficiente quando as falhas não ocorrem durante muitos quadros seguidos, pois a ausência da entrada de controle na previsão a priori faz com que a mesma seja realizada apenas baseada na velocidade e aceleração do robô, o que leva os estados previstos a priori a terem um comportamento estritamente linear.

#### 4.2.9 Envio de dados

Como relatado anteriormente, os módulos que compõem esse sistema são executados de forma independente, mas se comunicam através do protocolo UDP. Apesar do protocolo prover a transmissão dos dados, é necessário determinar uma codificação para

que os mesmos possam ser enviados. Sendo assim, as informações referentes aos robôs e a bola foram organizadas de acordo com o formato JSON (*JavaScript Object Notation*). O JSON é uma formatação leve utilizada para troca de dados, sendo fácil de interpretação tanto para humanos quanto para máquinas. A Figura 4.2 exibe um trecho do conteúdo de um pacote, no qual está detalhado a posição de dois robôs e da bola.

```

1  {
2      "B": {
3          "x": 220,
4          "y": 342
5      },
6      "R": [
7          {
8              "id": 0,
9              "w": 30.68,
10             "x": 50.00,
11             "y": 98.50
12         },
13         {
14             "id": 1,
15             "w": 60.75,
16             "x": 59.22,
17             "y": -65.52
18         }
19     ]
20 }
```

Listagem 4.2 – Formatação JSON utilizada para o envio dos dados

### 4.3 Navegação

Como detalhado no Capítulo 3, o estudo sobre métodos de navegação para robôs móveis é bem abrangente. No entanto, dentre tantos trabalhos, o método de campo de vetores unitários de Lim et al. (2008) se destaca por permitir que seja possível inserir um ângulo de chegada do robô na bola, o que é muito útil para evitar que o mesmo realize gols contras e que ele consiga direcionar a bola ao gol, por exemplo. Para isso esse método cria uma espiral hiperbólica próxima ao objetivo, gerando trajetórias que respeitam as restrições de movimento dos robôs não holonômicos com tração diferencial. Portanto, esse método foi o adotado por este trabalho e todas as equações utilizadas nesta seção foram extraídas de Lim et al. (2008) e adaptadas para que sejam apresentadas de uma forma mais compreensível.

Para a eficácia do método todas as variáveis referentes a ângulos devem assumir valores no intervalo  $[-\pi, \pi]$ . A modelagem da espiral em questão pode ser vista na Equação 4.18, no qual:

$d_e$  é uma constante que define o raio da espiral;  
 $\theta$  é o ângulo entre ponto  $p$  e o centro do plano cartesiano;  
 $\rho$  é a distância euclidiana entre o ponto  $p$  e a origem;  
 $K_r$  é um parâmetro ajustável;  
 $d_{ir}$  indica o sentido de giro do robô. Quando assume o valor 1, indica um giro no sentido horário. Quando assume o valor -1, sentido anti horário.

$$\phi_h(p, d_{ir}) = \begin{cases} \theta + d_{ir} \frac{\pi}{2} (2 - \frac{d_e + K_r}{\rho + K_r}), & \text{se } \rho > d_e \\ \theta + d_{ir} \frac{\pi}{2} \sqrt{\frac{\rho}{d_e}}, & \text{se } 0 \leq \rho \leq d_e \end{cases} \quad (4.18)$$

Assumindo-se que o desafio do robô é conduzir a bola para um objetivo a direita em um plano sem obstáculos, a Equação 4.19 é capaz de gerar o campo vetorial suficiente para que o robô conduza a bola ao objetivo, assim como representado na Figura 17.

$$\Phi_{TUF} = \begin{cases} \tan^{-1} \left( \frac{y_l \sin(\phi_h(p_l, 1)) + y_r \sin(\phi_h(p_r, -1))}{y_l \cos(\phi_h(p_l, 1)) + y_r \cos(\phi_h(p_r, -1))} \right) & \text{se } -d_e \leq y < d_e \\ \phi_h(p_l, 1) & \text{se } y < -d_e \\ \phi_h(p_r, -1) & \text{se } y \geq d_e \end{cases} \quad (4.19)$$

em que:

$$y = robô.y - bola.y$$

$$x = robô.x - bola.x$$

$$y_l = y + d_e, y_r = y - d_e$$

$$p_l = [x, y_l], p_r = [x, y_r]$$

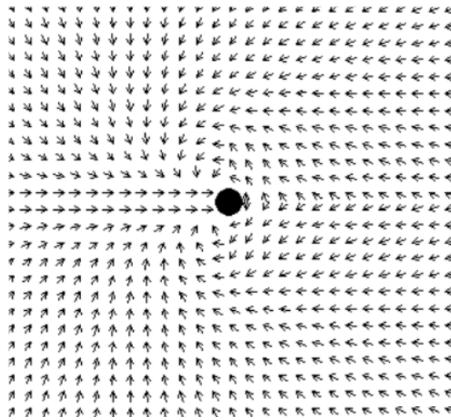


Figura 17 – Campo vetorial resultante da Equação 4.19. Extraído de Lim et al. (2008)

Apesar da Equação 4.19 ser capaz de conduzir o robô até o objetivo, ela apenas é eficaz quando não existem obstáculos no trajeto, sendo necessário a criação de campos repulsivos nos obstáculos, que são os outros robôs presentes no campo e as paredes que

limitam o mesmo. Para tal, o método de Lim et al. (2008) propõe a definição de uma distância mínima ( $d_{min}$ ) entre o robô e o obstáculo para que o campo repulsivo seja aplicado isoladamente de forma a evitar uma colisão imediata. Já nos casos em que a distância ( $d$ ) entre o robô e o obstáculo é maior que  $d_{min}$ , é necessário que seja feita uma combinação linear convexa entre o campo repulsivo ( $\phi_{AUF}$ ) e o atrativo ( $\phi_{TUF}$ ) através de uma função gaussiana com parâmetro ajustável  $\delta$ , conforme exibido na Equação 4.21. Nesse contexto o campo repulsivo  $\phi_{AUF}$ , representado na Equação 4.20, é calculado conforme a angulação entre o robô e o obstáculo e é ponderado em função da distância.

$$\phi_{AUF} = \tan^{-1} \left( \frac{robô.y - obst.y}{robô.x - obst.x} \right) \quad (4.20)$$

$$\phi_{composed} = \begin{cases} \phi_{AUF} & \text{se } d \leq d_{min} \\ \phi_{AUF}G(d-d_{min}, \delta) + \phi_{TUF}(1-G(d-d_{min}, \delta)) & \text{se } d > d_{min} \end{cases} \quad (4.21)$$

em que:

$$G(r, \delta) = e^{-\frac{r^2}{2\delta^2}}$$

O valor de  $\phi_{composed}$  corresponde à orientação para a qual o robô deve seguir em um determinado instante de tempo. Para adicionar mais de um obstáculo no cálculo do campo vetorial resultante, é necessário fazer o uso de uma estrutura de repetição com um número de iterações equivalente a quantidade de obstáculos a serem contabilizados, no qual, a partir da primeira iteração, o valor de  $\phi_{TUF}$  deve ser substituído na Equação 4.21 pelo valor anterior de  $\phi_{composed}$ .

Para a aplicação do futebol de robôs, em que nem sempre o robô deve se direcionar à bola em um sentido paralelo ao eixo  $x$ , é necessário substituir as posições referentes ao robô, a bola e aos obstáculos por posições virtuais. Tais posições são obtidas a partir do rotacionamento das coordenadas dos anteriores de acordo com a angulação da bola em relação ao destino (gol adversário), alinhando a coordenada referente ao eixo  $y$  da bola virtual com o gol adversário. Dessa forma será possível que o campo de vetores abordado forneça um direcionamento virtual para o robô, que depois deve sofrer a subtração do deslocamento angular utilizado na criação dos objetos virtuais para gerar o direcionamento real para o robô.

## 4.4 Controle

Na movimentação de um robô existem vários fatores que precisam ser corrigidos, como por exemplo imperfeições dos motores, rodas empenadas, atrito irregular e etc. A solução desse problema pode ser realizada através da implementação de um filtro de controle, que deve ser capaz de compensar toda imperfeição a partir de uma amostra do erro informada a cada iteração do algoritmo.

O PID é um método de controle utilizado para eliminar erros e suavizar trajetórias, sendo a técnica de controle mais utilizada na indústria devido à vasta aplicação e grande desempenho em diferentes casos (ROSA, 2015). O PID completo pode ser calculado a partir da soma dos três componentes que compõem o controlador, mas em algumas aplicações é comum utilizar apenas o controle integral (PI), controle derivativo (PD) ou em casos mais simples apenas o controle proporcional (P).

O controle proporcional (P) consiste em multiplicar o erro por uma constante  $k_p$  a fim de se obter uma correção que seja de acordo com o crescimento da imperfeição. O erro e a ação de controle  $u_P(t)$  podem ser calculados a partir das equações 4.22 e 4.23, sendo  $e(t)$  o erro no instante  $t$ ,  $V_r$  o valor de referência e  $V_m$  o valor observado.

$$e(t) = V_m - V_r \quad (4.22)$$

$$u_P(t) = k_p e(t) \quad (4.23)$$

A componente Integral (I) é responsável por dar significância a pequenos erros que se acumulam ao longo de um período de tempo e não são eliminados através do controle Proporcional. A parcela  $u_I(t)$  pode ser calculada através da Equação 4.24.

$$u_I(t) = k_i \int_0^t e(t) dt \quad (4.24)$$

Já a componente Derivativa (D) é responsável por acrescentar uma ação antecipatória, de modo a diminuir o sobressinal. Por ser a mais sensível a ruídos dos três componentes,  $u_D(t)$  é a parcela que geralmente possui menor ganho (ROSA, 2015). O cálculo é realizado assim como demonstra a Equação 4.25.

$$u_D(t) = k_d \frac{de(t)}{dt} \quad (4.25)$$

O PID completo é composto pela soma das equações 4.23, 4.24 e 4.25, mas em algumas abordagens é comum que ele seja apresentado no formato ideal, padrão ou ISA. Nesse formato a constante  $k_i$  é representada por  $k_p \frac{1}{T_i}$ , em que  $T_i$  representa o tempo

integral, e a constante  $k_d$  representada por  $k_p T_d$ , em que  $T_d$  é o tempo derivativo. Assim, o PID pode ser representado conforme a Equação 4.26.

$$PID(t) = k_p(e(t) + \frac{1}{T_i} \int_0^t e(t)dt + T_d \frac{de(t)}{dt}) \quad (4.26)$$

Nessa pesquisa foi aplicado dois controladores para transformar o vetor unitário resultante da etapa de navegação em velocidades para o robô. Para isso foi considerado a Equação 4.3 da Seção 4.1.4.1, que determina as velocidades de cada roda em função da velocidade linear e angular do robô.

O primeiro controlador é responsável por determinar a velocidade angular do robô. Sendo assim, o valor observado ( $V_m$ ) representa a angulação do robô no campo, que é obtida através dos dados fornecidos pela camada de Visão, e o valor de referência ( $V_r$ ) é obtido pelo algoritmo de navegação. Logo o erro do robô no instante  $t$  é dado de acordo com a diferença angular do robô ao ponto objetivo, conforme a Equação 4.22.

Como em uma partida de futebol o destino de um jogador é bastante variável, os acúmulos da componente de integral do PID podem prejudicar a reação do sistema de controle quando o ponto de objetivo sofrer mudanças bruscas. Por isso, para o controlador angular, será utilizado o PD.

Para a velocidade linear do robô é comumente utilizado como entrada para o sistema de controle o módulo do vetor resultante da etapa de navegação. No entanto, o algoritmo de navegação utilizado neste trabalho tem sempre como saída um vetor unitário, e utilizar uma velocidade linear constante traz perda de desempenho para a movimentação do robô. Com uma velocidade linear constante o robô poderá não conseguir realizar as curvas ou então ir bastante devagar em uma reta. A alternativa encontrada foi modelar o controlador também em função do erro angular, mas com uma ação reversa, fazendo com que o robô acelere mais nas retas, situação em que o erro angular está próximo de zero, e avance mais lentamente nas curvas, quando o erro angular é maior. Essa abordagem é ideal para situações em que o robô vai empurrar uma bola ao gol, por exemplo, pois permite que o mesmo chegue bem veloz ao alvo.

No controlador da velocidade linear, em específico, será considerado o módulo do erro, pois o robô deve ter a mesma velocidade linear para trajetórias de curvaturas iguais mas de direções diferentes. Nessa aplicação, além dos problemas já relatados em relação a componente integral do PID, a componente derivativa também não é indicada. Como a definição da derivada leva em consideração o valor amostrado anteriormente, o uso do módulo do erro pode gerar valores inadequados nos instantes em que o erro passa por uma inversão de sinais. Sendo assim, para a velocidade linear, será utilizado apenas o controlador proporcional.

Apesar de não ser abordado por este trabalho, nas situações em que o robô precisa

ir até um ponto e parar, o uso da distância euclidiana até o ponto final como parâmetro para um controlador linear é encorajado. Outro ponto importante está relacionado com a mecânica dos robôs utilizados. Como os mesmos possuem duas áreas de domínio para a bola é possível que ocorra uma inversão da área definida como frente do robô nas situações em que o módulo do erro angular é maior que  $90^\circ$ . Nesse caso é necessário somar  $180^\circ$  no valor observado do robô e recalcular o módulo de navegação.

#### 4.4.1 Ajuste e Sintonia dos Controladores

Um dos maiores desafios da implementação de controladores é a definição das constantes utilizadas, que precisam ser bem sintonizadas para a eficácia do controlador. Neste trabalho foi utilizada a seguinte metodologia para ajuste dos controladores: inicialmente a velocidade linear do robô foi definida como zero e foram ajustadas as constantes para o controlador angular. Depois, com as constantes do controlador angular já definidas, ocorreu o ajuste do controlador da velocidade linear.

O ajuste das constantes foi realizado de forma empírica. Um comparativo entre diferentes constantes para a velocidade linear foi realizado com base no desempenho do robô em seguir uma trajetória previamente conhecida. Essa trajetória foi a curva Lemniscata de Bernoulli, que é ideal para analisar o desempenho do robô pois é formada por segmentos retilíneos e cuvos para os dois sentidos. Para fazer com que o robô siga essa trajetória foram armazenados 400 pontos da curva e escolhido como destino em cada passo aquele que minimize a Equação 4.27, em que  $\phi$  é o erro angular do robô em relação ao ponto e  $d$  é a distância euclidiana entre ambos.

$$f(\phi, d) = 0.8d + 0.2\phi \quad (4.27)$$

# 5 Resultados

Neste capítulo são apresentados os resultados provenientes da aplicação da metodologia descrita anteriormente. Todos os algoritmos foram executados em uma máquina executando Ubuntu 19.04 com processador Intel Core i7-8550U 1.80GHz, 8Gb de memória ram e placa de vídeo AMD *radeon graphics* de 2Gb. Os resultados estão divididos entre o módulo de visão computacional, com resultados na Seção 5.1 e o módulo de navegação e controle, na Seção 5.2. Ao final, na Seção 5.3, são apresentadas discussões sobre o trabalho e propostas para trabalhos futuros.

## 5.1 Módulo de Visão Computacional

### 5.1.1 Transformação Projetiva

A Figura 18 exibe o resultado da aplicação de uma transformação projetiva. Na Figura 18 (a) é exibida a imagem original, em que os quatro pontos de corte foram destacados. Na Figura 18 (b) está a imagem pós transformação. É interessante destacar que surgiram duas faixas pretas na imagem - parte inferior e lateral direita. Tais faixas são frutos da correção na projeção. Na Figura 18 (c) ocorreu o corte na imagem, em que a mesma está agora pronta para os próximos passos do processamento.

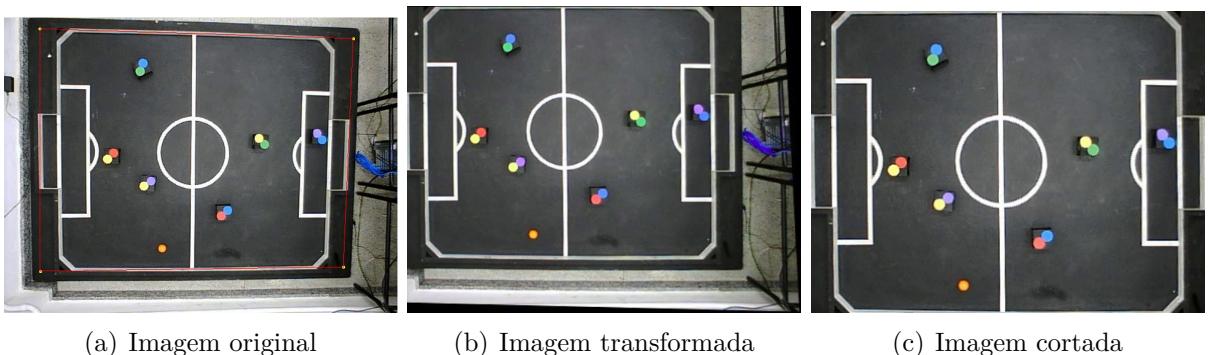


Figura 18 – Aplicação da transformação projetiva na imagem capturada.

### 5.1.2 Conversão para HSV

A Figura 19 ilustra o resultado da conversão de uma imagem RGB para HSV, em que cada canal da imagem está exibido de forma separada. Nas figuras percebe-se que apenas pelo componente de saturação (Figura 19 (e)) é possível filtrar todos os elementos de interesse da imagem, pois os mesmos aparecem bem realçados em relação ao fundo da imagem. Essa característica agiliza bastante a calibração dos espaços de cores, pois depois que ocorre a filtragem dos elementos de interesse, basta usar apenas o canal matiz

e/ou brilho para distinguir uma cor de outra, comportamento que não ocorre em imagens RGB.

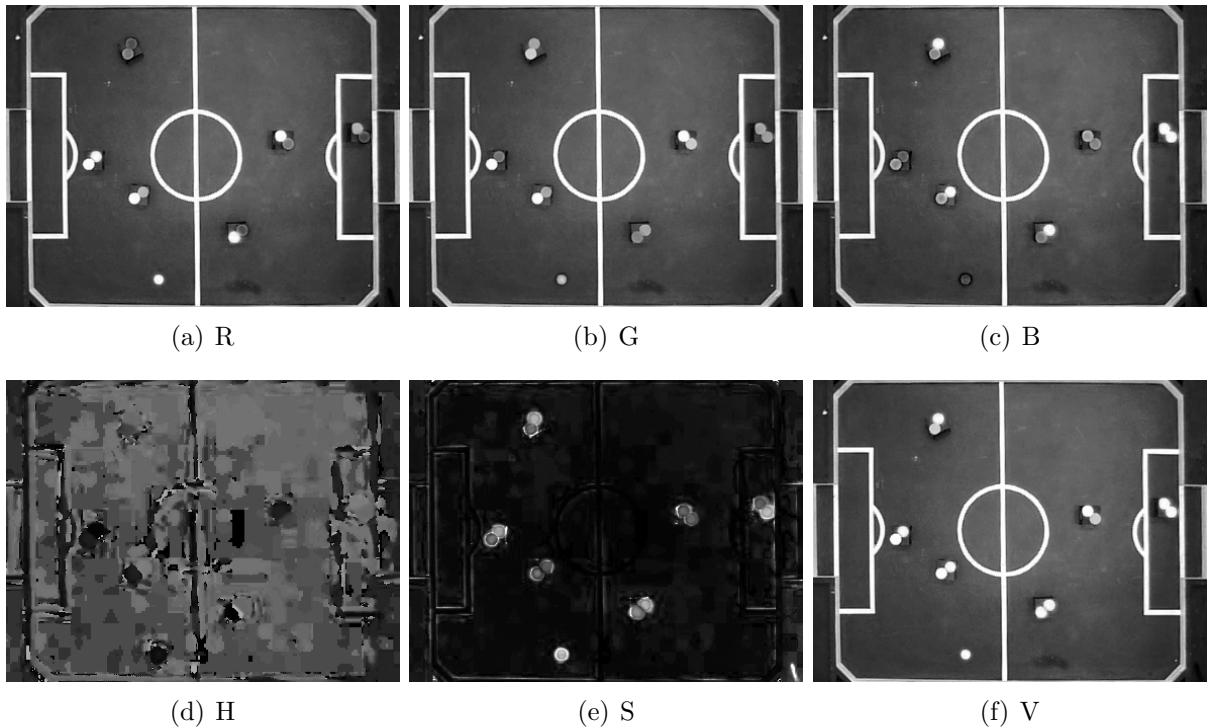


Figura 19 – Canais explícitos de uma imagem em RGB e HSV.

### 5.1.3 Filtros por *Threshold* e Remoção de Ruídos

Para exemplificação da etapa de remoção de ruídos foi aplicado o filtro por *threshold* na imagem pós conversão para HSV com o objetivo de filtrar as etiquetas de cor amarela. Propositalmente, as constantes que limitam o espaço de cores não foram bem ajustadas, tentando replicar assim situações em que a iluminação ambiente é muito alta e a cor amarela se torna bem similar com a cor branca, que delimita as linhas do campo. O resultado da aplicação do filtro pode ser visualizado na Figura 20 (a). Para uma melhor visualização apenas um recorte da imagem foi exibido.

Sequencialmente, foi aplicada a erosão na imagem resultante através de uma máscara de ordem  $7 \times 7$  em formato de cruz. Como pode ser observado, a operação de erosão removeu os ruídos da imagem, mas os objetos de interesse também sofreram um pouco de redução na área. Esse problema foi resolvido com a aplicação sequencial da dilatação, exibida na Figura 20 (c).

### 5.1.4 Filtragem de Dados

Para demonstrar a efetividade da filtragem de dados implementada foram utilizadas posições referentes a um percurso da bola utilizada, de modo a comparar as posições

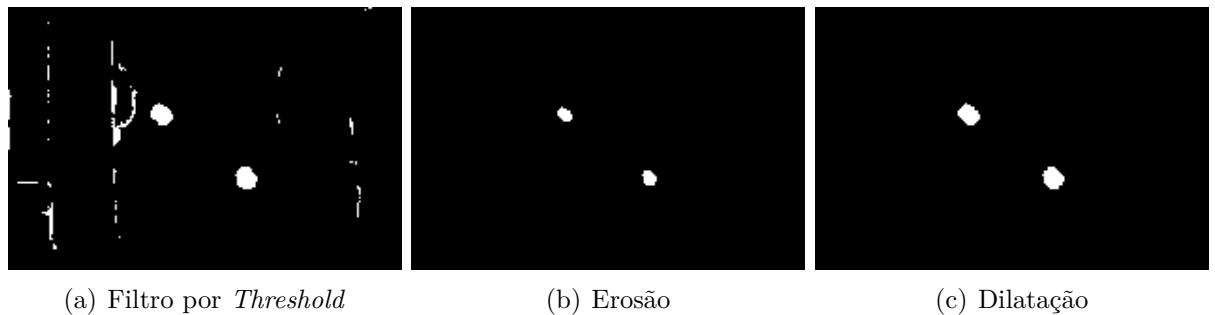


Figura 20 – Aplicação da operação de abertura em uma imagem com ruídos.

observadas com as filtradas. No mesmo teste foram omitidas algumas das posições observadas, simulando assim uma falha temporária do sistema de visão em reconhecer a bola. O resultado pode ser visualizado na Figura 21.

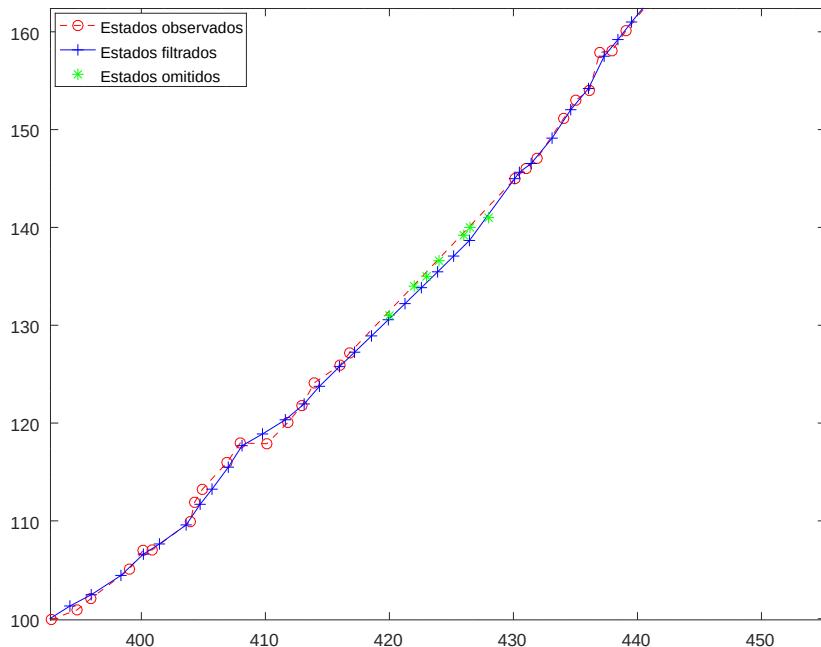


Figura 21 – Filtro de kalman discreto aplicado para filtragem e previsão da posição da bola.

Através da Figura 21 observa-se que o filtro obteve bom desempenho em filtrar os dados e conseguiu também aproximar a posição da bola no caso da ausência do valor observado. O mesmo experimento também foi realizado utilizando como entrada os estados observados de um robô. Nesse caso os resultados são provenientes da combinação de dois filtros, uma vez que esses foram aplicados em cada uma das duas etiquetas do robô e não na posição final do mesmo. Os resultados estão na Figura 22, no qual percebe-se que o filtro de Kalman também obteve bom desempenho na filtragem dos dados. No entanto, nos quadros em que foram ocultados os estados de observação, a previsão não foi capaz de acompanhar o percurso do robô. Isso acontece pois o robô não tem um comportamento

linear e com a ausência das entradas de controle no modelo de previsão o filtro não foi capaz de estimar com sucesso a posição do mesmo em falhas consecutivas.

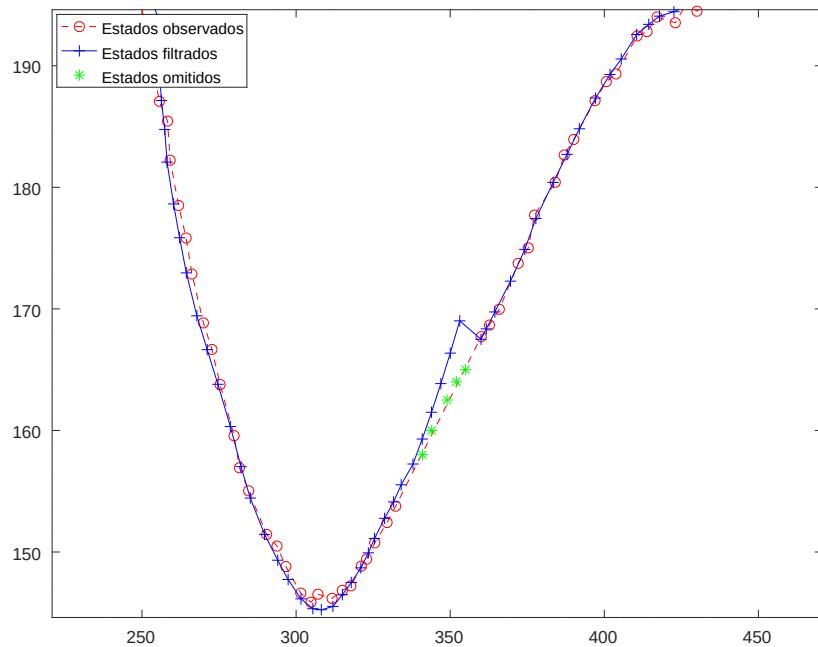


Figura 22 – Filtro de kalman discreto aplicado para filtragem e previsão da posição de um robô.

### 5.1.5 Interface Gráfica

As Figuras 23 e 24 exibem duas janelas do sistema de visão computacional desenvolvido. A primeira janela possibilita a calibração das cores de forma interativa através de barras de rolagem. Já a segunda representa a janela de execução do sistema, em que na parte direita é gerado uma animação que permite o usuário acompanhar o reconhecimento durante a partida

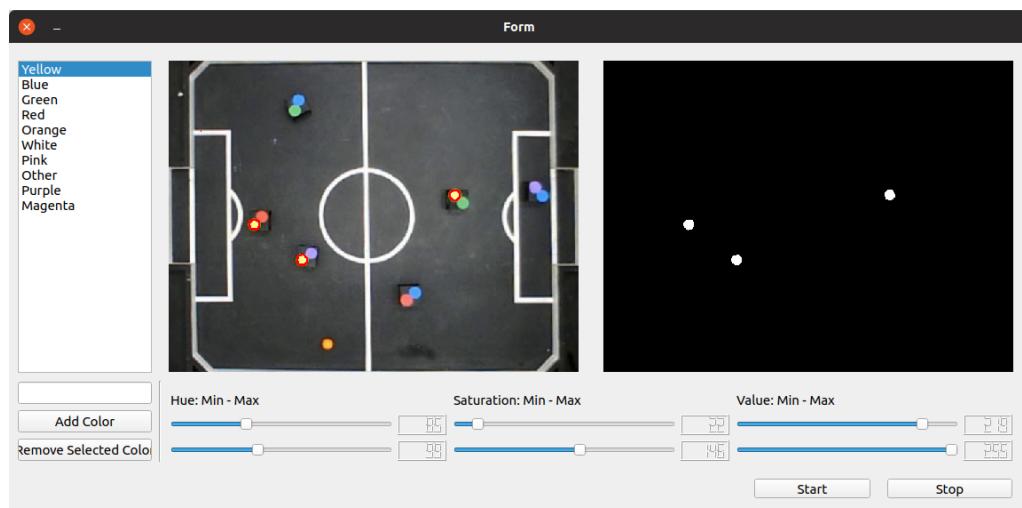


Figura 23 – Interface de calibração de cores para o sistema de visão computacional.



Figura 24 – Janela principal do sistema de visão computacional.

### 5.1.6 Desempenho do Sistema

Constatar a precisão da identificação da posição dos robôs é uma tarefa difícil de se realizar, uma vez que não existe uma forma de mensurar a posição exata dos mesmos no ambiente de captura. Assim, para mensurar o desempenho do sistema de visão, foi tomado como medida comparativa a quantidade de quadros em que o algoritmo conseguiu identificar os robôs e a bola. Para um ambiente com 6 robôs e uma bola, a Tabela 1 exibe a porcentagem de captura dos objetos para uma amostragem de 540 quadros advindos de um vídeo gravado pela câmera utilizada. Nesse caso, mesmo que as previsões com o filtro de Kalman tenham acertado a posição do objeto, o mesmo foi considerado como não identificado.

Objetos	Taxa de Captura
Robô 1	99,62%
Robô 2	99,81%
Robô 3	100%
Robô 4	99,81%
Robô 5	100%
Robô 6	99,25%
Bola	98,88%

Tabela 1 – Taxa de identificação dos objetos

No mesmo teste, foi mensurado também o tempo médio de execução das etapas do algoritmo, tal como o desvio padrão, em que o tempo do processo de identificação dos robôs e da bola foi avaliado juntamente com o reconhecimento dos contornos dos mesmos. A Tabela 2 exibe essas informações.

Com o tempo médio de 8.6090 ms de execução total é possível constatar que o sistema desenvolvido tem uma capacidade de processar aproximadamente 116 fps em

Etapa	Tempo Médio	Desvio Padrão
Transformação Projetiva	0,9014 ms	0,1378
Conversão para HSV	1,0538 ms	0,0395
Filtros por Threshold	2,7295 ms	0,2273
Remoção de Ruídos	1,2325 ms	0,0974
Identificação dos Robôs	0,3017 ms	0,0461
Identificação da Bola	0,1294 ms	0,0496
Filtragem dos Dados	0,0776 ms	0,0168
Envio dos Dados	0,0806 ms	0,0150
Atualização da Interface Gráfica	2,1025 ms	1,9480
Total	8,6090 ms	2,5775

Tabela 2 – Tempos de execução do módulo de visão computacional por etapas.

computadores com hardware similar ao utilizado. No mesmo contexto, pelos dados da Tabela 1 é possível constatar que o software desenvolvido reconheceu todos elementos em quase 100% dos quadros. A bola, apesar de também resultar em um alto índice de reconhecimento, foi o objeto que mais foi perdido. Isso acontece devido à mesma percorrer em alta velocidade em algumas partes do jogo, podendo assim gerar borões na imagem. Vale ressaltar que nos casos em que ocorre a perda de algum dos objetos, a previsão da posição dos mesmos é realizada através do filtro de Kalman implementado.

Nas execuções em tempo real o sistema pode ser limitado pela câmera utilizada, como acontece com a câmera *PlayStation Eye*, que é capaz de fornecer apenas 60 fps na resolução de 640 x 480 pixels. Sendo assim ocorre um acréscimo de tempo referente ao período em que a execução permanece ociosa esperando por um novo quadro da câmera.

Pela Tabela 2 percebe-se que o gargalo do sistema está na aplicação dos filtros por *threshold* seguido da atualização da interface gráfica, que também possui um alto desvio padrão. No entanto o tempo referente a atualização da interface gráfica pode ser reduzido drasticamente com o uso de uma *thread* dedicada inteiramente a este fim. Com a remoção do tempo para a atualização da interface gráfica o sistema passa a ser compatível com câmeras de 120 fps também, pois terá capacidade de processar aproximadamente 153 fps.

## 5.2 Módulo de Navegação e Controle

Esta seção apresenta os resultados provenientes dos módulos de navegação e controle. Nas figuras que apresentam movimentações do robô foi adotado a escala 1 : 0.25cm<sup>2</sup>.

### 5.2.1 Avaliação do Controle

Antes de realizar testes com o algoritmo de navegação utilizado foram realizados experimentos com o robô para seguir a curva Lemniscata de Bernoulli. Nesse caso foram considerados três valores para a constante *kp* da velocidade linear: 2.5, 3.0 e 3.5. As

contantes do controlador da velocidade angular foram:  $k_p = 1.1$  e  $T_d = 0.2818$  para todos os testes. Os resultados podem ser visualizados na Figura 25.

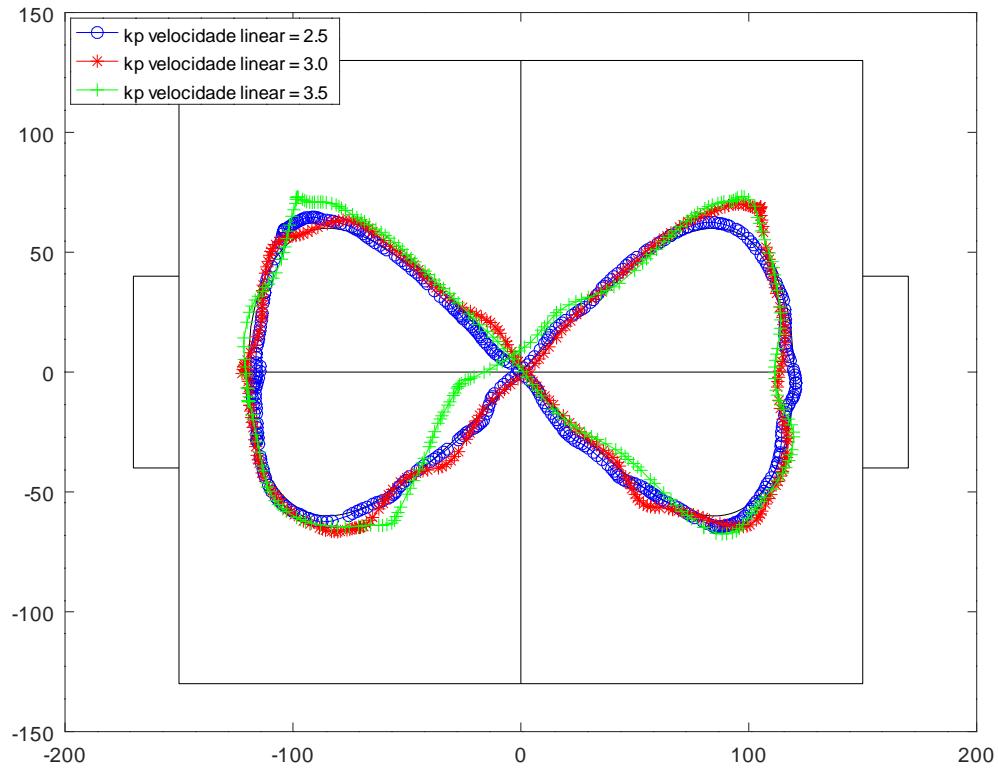


Figura 25 – Avaliação do controle do robô através da curva Lemniscata de Bernoulli

Os tempos que o robô demorou para completar uma volta no trajeto foram 8.4s, 7.50s e 7.18s respectivamente para os três valores de  $kp$ . No entanto observa-se na Figura 25 que as trajetórias geradas por  $kp = 3.5$  e  $kp = 3$  escaparam bastante da rota original, principalmente nas curvas. Em compensação, o trajeto com  $kp = 2.5$  foi capaz de seguir a curva com perfeição. Sendo assim foi definida a velocidade como 2.5 para os testes posteriores.

### 5.2.2 Aplicação do Campo Vetorial

As Figuras 26, 27, 28, 29 e 30 exibem situações em que foi aplicado o método de campo de vetores unitários, no qual o robô em movimento foi representado por dois traçados azuis das laterais traseiras do chassi; a bola representada por um traçado laranja e os obstáculos representados por círculos. Nos testes as constantes do campo vetorial foram definidas como:  $d_{min} = 7$ ,  $d_e = 10$ ,  $K_r = 50$  e  $\delta = 13$ .

Em todas as situações o objetivo do robô foi de levar a bola até o gol. Na Figura 26 o objetivo foi conduzir a bola até o centro do gol esquerdo. Na Figura 27 o gol de objetivo foi o direito com um obstáculo no trajeto. Nas Figuras 28, 29 e 30 o número de obstáculos foi 3, 3 e 5 respectivamente. Para uma melhor visualização, foi disponibilizado também

um vídeo no YouTube<sup>1</sup>.

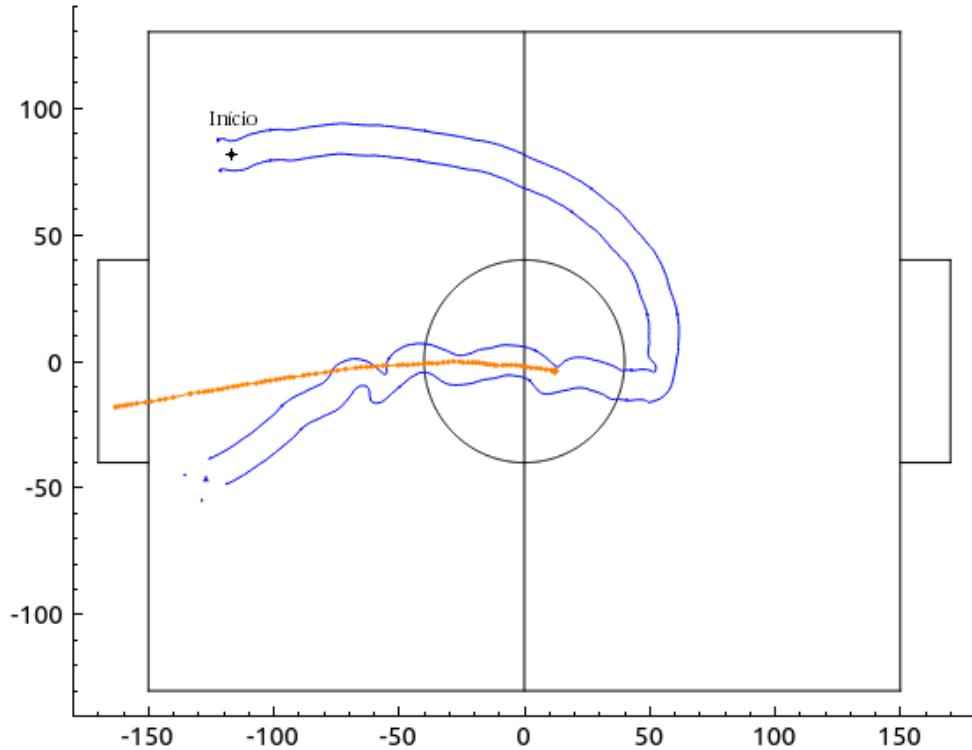


Figura 26 – Condução da bola para o gol esquerdo sem obstáculos.

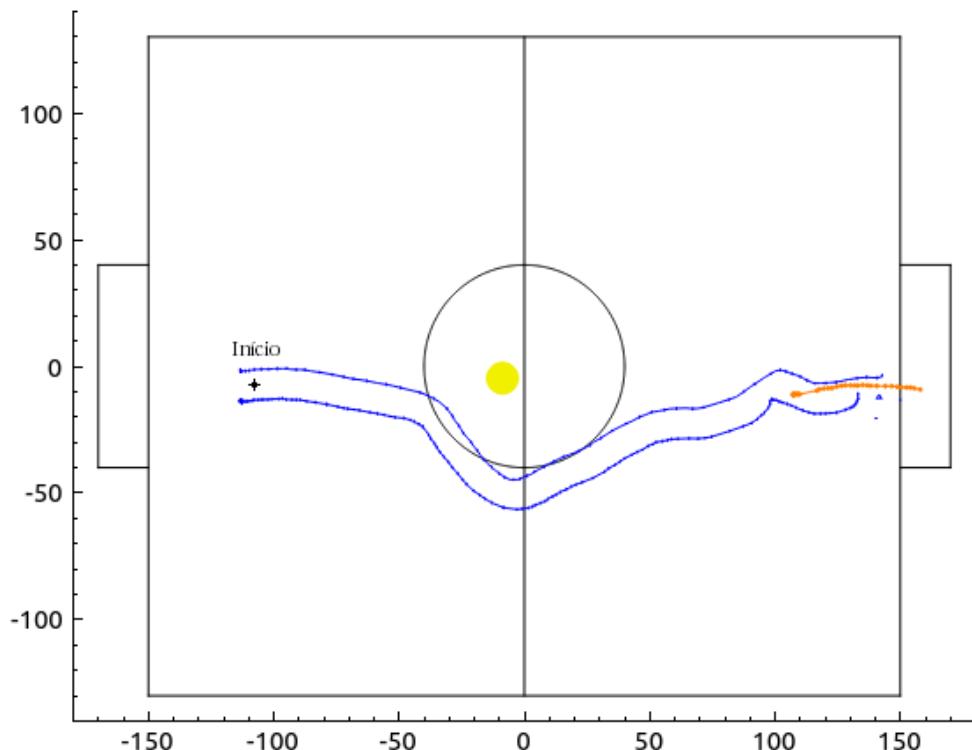


Figura 27 – Condução da bola para o gol direito com um obstáculo.

<sup>1</sup> <<https://www.youtube.com/watch?v=YU4e1UFWLeY>>

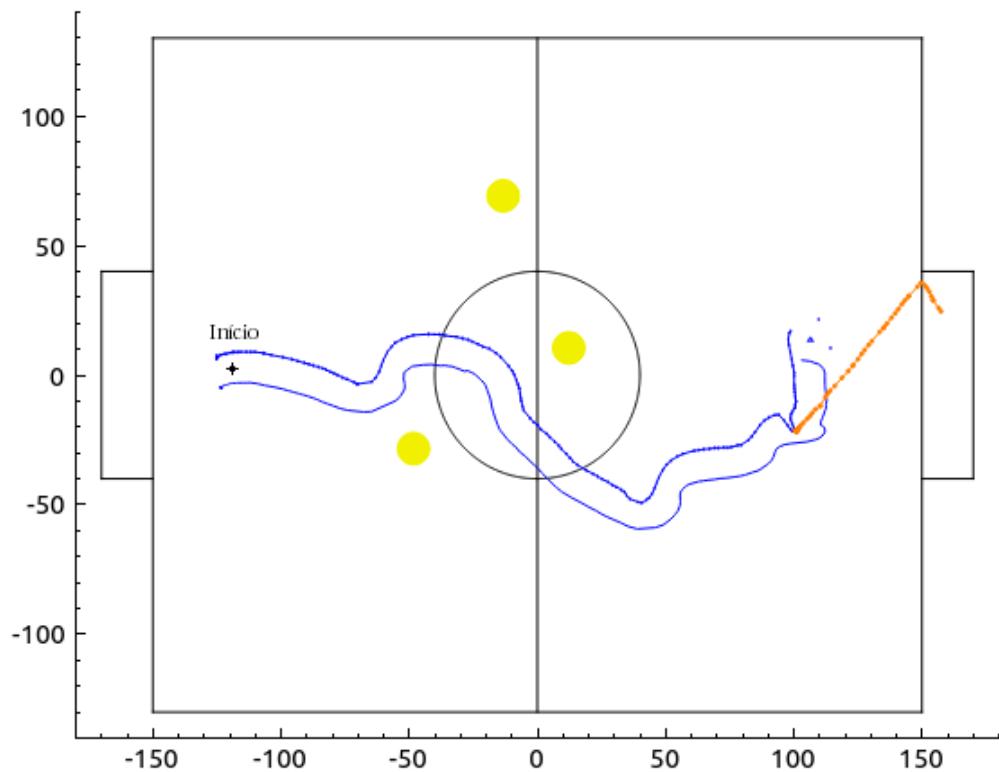


Figura 28 – Condução da bola para o gol direito com três obstáculos (a).

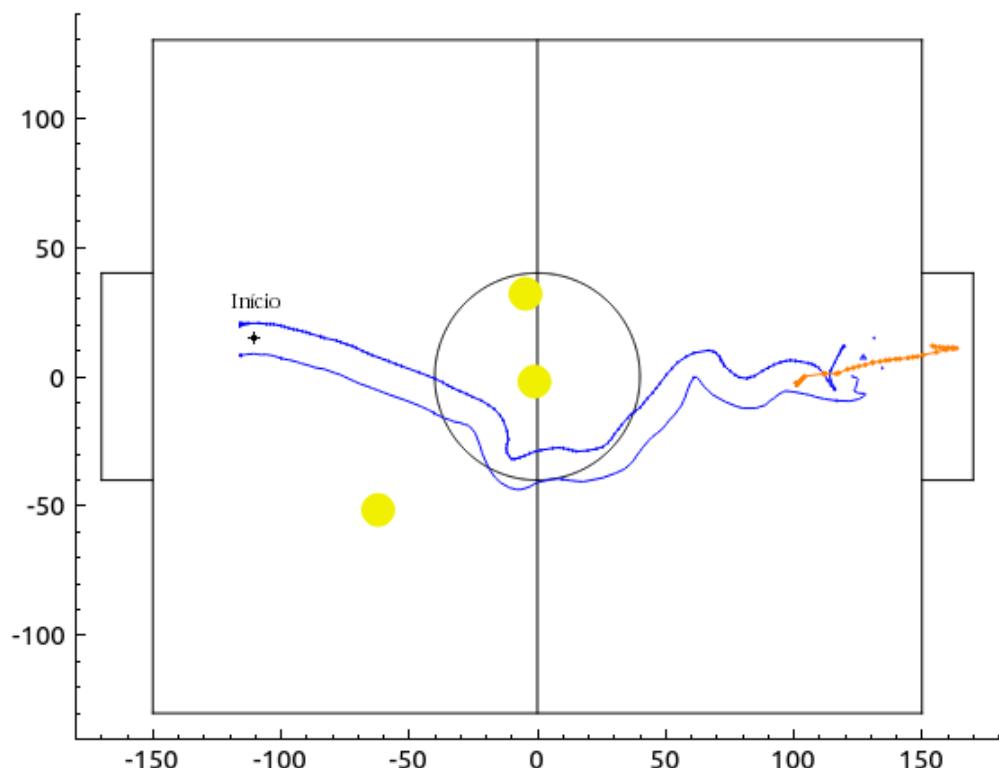


Figura 29 – Condução da bola para o gol direito com três obstáculos (b).

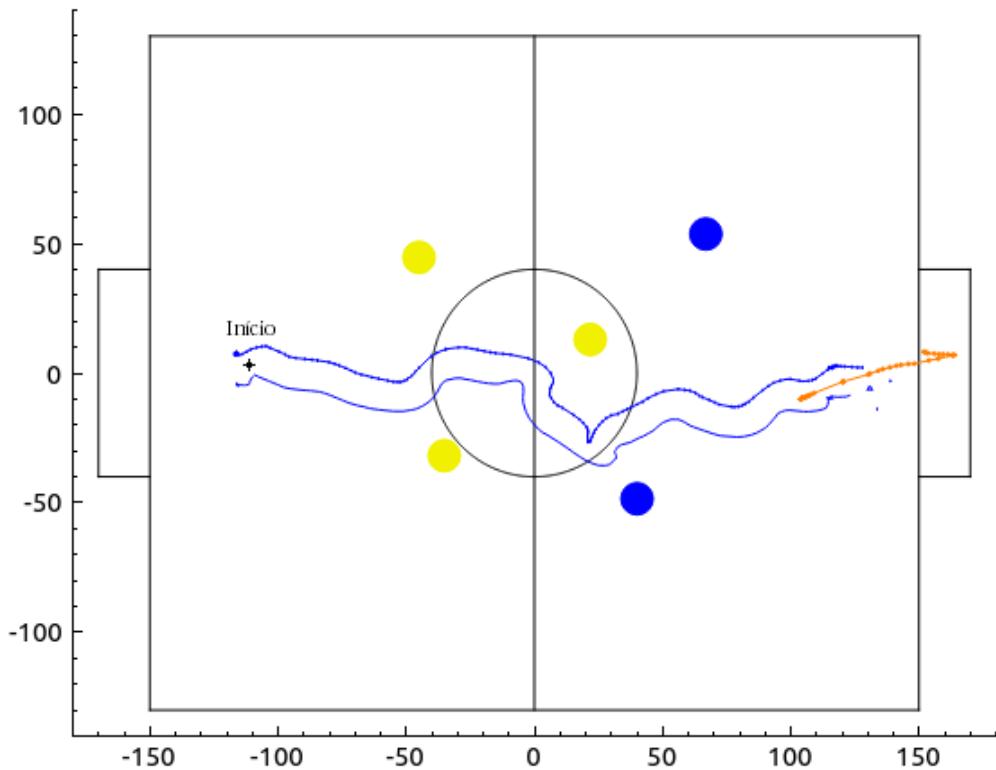


Figura 30 – Condução da bola para o gol direito com cinco obstáculos.

É possível observar que o robô obteve bom desempenho em realizar a tarefa, não colidindo com os obstáculos e conduzindo a bola até o gol. Pela Figura 26 fica claro uma oscilação do robô quando o mesmo chega até a bola. Isso acontece pois a bola desliza dentro da área de domínio do robô, o que exige que o mesmo tenha que realizar manobras para mantê-la sobre domínio.

### 5.3 Discussões e Propostas para Trabalhos Futuros

Analizando os resultados é possível perceber que os métodos aqui utilizados foram eficientes para o problema abordado. O sistema de visão computacional reconheceu os objetos em campo com alta frequência e com uma taxa de erros muito baixa, contando ainda com um recurso para previsão de posições nas possíveis falhas durante um jogo da categoria VSS. Quanto aos métodos de navegação e controle, os mesmos também se demonstraram eficientes, fazendo com que o robô consiga seguir trajetórias pré definidas e também navegar em um ambiente com trajetórias adaptativas.

Uma situação bem específica em que o sistema de visão computacional pode falhar é descrita em [Martins et al. \(2007\)](#), que propôs o método de árvores de dependências implementado. Esse caso está representado na Figura 31, em que é possível notar que todas as etiquetas de cores de time possuem duas dependências. Esse comportamento faz com que o padrão de cores de um robô esteja representado em mais de um lugar no

campo. Sendo assim uma proposta para trabalhos futuros é adicionar a verificação da proximidade dos padrões de cores com a posição anterior do robô, escolhendo como par de cores aquele que é mais próximo da posição anterior do robô.

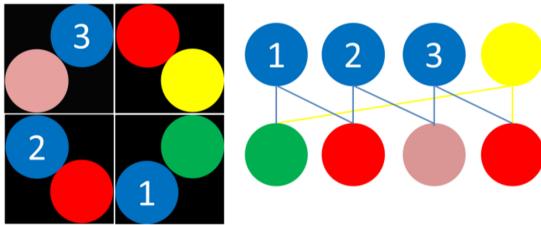


Figura 31 – Situação de falha na detecção dos pares de cores.

Outro ponto importante foi notado nos testes de controle, em que o robô começou a perder precisão dos movimentos com velocidades não muito altas. Considerando que em um jogo de futebol a velocidade dos mesmos é um dos importantes fatores para a vitória de um time, esse é um ponto que deve ser abordado em trabalhos futuros. Para amenizar esse problema podem ser utilizadas câmeras com capacidade de fornecer mais quadros por segundo, aumentando assim a taxa de correção dos movimentos do robô. No entanto para um aumento da taxa de processamento do sistema é necessário um estudo sobre algumas áreas não abordadas neste trabalho, como a latência da comunicação com os robôs e a capacidade dos mesmos em receber um grande número de pacotes e ainda acionar os motores.

O fato de uma câmera ser capaz de fornecer 60 fps significa que aproximadamente a cada 16.5 ms um novo quadro é fornecido. No entanto existe um atraso maior no fornecimento do primeiro quadro e que se arrasta durante todos os outros. Isso indica que a diferença temporal entre a captura de um quadro e o recebimento do mesmo no computador não é de apenas 16.5 ms. Na câmera utilizada essa diferença é de aproximadamente 65ms. Sendo assim as ações do robô estão sempre pautadas em um estado anterior, que varia não apenas com a frequência da câmera. Esse atraso se mostra cada vez mais influente a medida que o robô navega em uma velocidade maior no ambiente.

Neste trabalho foi aplicado o filtro de kalman discreto para a filtragem das posições dos objetos e para previsão das posições em situações de falhas no reconhecimento dos objetos. Uma forma de compensar o atraso da câmera é utilizar a previsão a priori do filtro de kalman não apenas em situações de falhas, mas em todos os momentos. No entanto, como constatado, a previsão das posições do robô sem a utilização das entradas de controle não é ideal devido a não linearidade dos movimentos do mesmo. Sendo assim, para trabalhos futuros, fica a necessidade do deslocamento da filtragem de dados presente no módulo de visão computacional para o módulo de navegação e controle. Essa alteração permitirá a inserção das entradas de controle na previsão da posição dos robôs e pode ser

realizada baseada na Equação 4.2, que transforma a velocidade linear e angular do robô em velocidades nos dois eixos do plano cartesiano.

Em relação à velocidade linear do robô, em algumas situações notadas em testes o robô não conseguiu convergir para o objetivo. Essa situação é ilustrada na Figura 32, no qual nota-se que o valor de referência ( $V_r$ ) exige que o robô realize um giro de 90° para ir até a bola que está bem próxima. Inicialmente o controlador angular do robô terá uma saída bem alta e o controlador linear terá como saída o valor zero, uma vez que o mesmo é de ação reversa em relação ao ângulo. Essa combinação de valores fará com que o robô gire em torno do próprio eixo. No entanto na medida que o erro angular diminui a velocidade linear também aumenta. Sendo assim o robô não conseguirá convergir para o objeto corretamente devido à proximidade entre ambos.

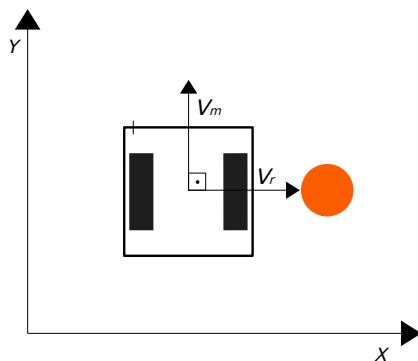


Figura 32 – Situação de falha do modelo de controle.

A Figura 32 foi uma situação notada nos testes finais e mostra a necessidade da inclusão da distância até o objetivo no cálculo da velocidade linear. Essa modificação pode ser realizada na função de erro do controlador através de uma equação que retorne um erro maior quando a distância é pequena e o erro angular é grande, mas que também retorne um erro menor quando a distância e o erro angular são pequenos, de forma a não prejudicar a capacidade do robô de impulsionar a bola até o gol. Um exemplo de função pode ser visualizado em Soares et al. (2019), que fez o uso de uma equação exponencial para combinar as duas variáveis. Uma outra alternativa é continuar usando apenas o erro angular, mas utilizar uma função de erro com crescimento não linear, fazendo com que a velocidade linear do robô apenas ganhe significância no modelo final quando o valor do erro angular é pequeno.

Outro ponto a ser melhorado em trabalhos futuros é em relação às diversas constantes utilizadas nesse trabalho, principalmente as pertencentes ao módulo de navegação e controle. Para facilitar o ajuste das mesmas e melhorar o desempenho do robô pode ser criado um modelo dinâmico do mesmo, permitindo assim que sejam realizadas simulações computacionais com o objetivo de encontrar a combinação das constantes que maximizem o desempenho do robô em seguir determinadas trajetórias. Só com um modelo dinâmico

(que leva em consideração particularidades do robô como torque dos motores, peso e etc) será possível encontrar a melhor combinação de valores para as constantes.

## 6 Conclusão

Esse trabalho teve como objetivo o desenvolvimento de um sistema para mapeamento e controle de robôs da categoria VSS. Foi apresentado inicialmente um panorama sobre o futebol de robôs, destacando algumas das categorias presentes na LARC. Em seguida a revisão da literatura foi focada na categoria VSS, destacando as principais etapas da construção de um sistema de futebol de robôs juntamente com os métodos aplicáveis em cada etapa.

Na metodologia foi descrito de forma detalhada uma combinação de técnicas para solucionar o problema proposto pela categoria VSS. Esses métodos são capazes de mapear todos os elementos presentes no campo de futebol e também controlar um robô através de um campo de vetores unitários. Apesar da metodologia de controle focar em um só robô, essa parte pode ser replicada para o controle simultâneo de múltiplos agentes.

Quanto aos resultados, os mesmos se demonstraram promissores, revelando um sistema de visão computacional com baixa taxa de falhas e capacidade de execução em alta frequência em computadores com hardware moderado. As técnicas abordadas para a navegação e controle do robô também foram capazes de apresentar bons resultados, conseguindo controlar o mesmo através de trajetórias adaptáveis e conduzir uma bola até o gol sem muitas dificuldades. Juntamente com os resultados foram apresentadas discussões sobre as fragilidades do sistema, listando para cada uma delas possíveis soluções.

O desenvolvimento desse trabalho resultou também na criação da primeira equipe de robótica associada ao DCOMP da UFSJ, atuando agora como uma importante documentação para futuros alunos que se interessarem pela continuação do projeto. A prática da robótica com robôs reais durante o curso de graduação é um valioso mecanismo para que o aluno veja de perto aplicações das teorias lecionados em sala de aula, podendo assim colaborar para a redução da evasão escolar nos cursos de ciência da computação e engenharias.

## Referências

- AI-AMMRI, A. S.; AHMED, I. Control of omni-directional mobile robot motion. *Al-Khwarizmi Engineering Journal*, v. 6, n. 4, 2010. Disponível em: <<https://pdfs.semanticscholar.org/e6f0/defb58919bad147eb7f9eab51820609b73e8.pdf>>. Citado na página 21.
- ARAUJO, G. M.; MENDONÇA, M. M.; FREIRE, E. Reconhecimento automático de objetos baseado em cor e forma para aplicações em robótica. *XVII Congresso Brasileiro de Automática*, 2008. Disponível em: <<http://www.gprufs.org/arquivos/reconhecimentoautomatico.pdf>>. Citado na página 18.
- ARESTEGUI, N. C. A. *Localização e mapeamento de robôs móveis utilizando inteligência e visão computacional*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, Dept. de Engenharia Mecânica, Rio de Janeiro, 9 2009. Citado 2 vezes nas páginas 13 e 26.
- BAZYLEV, D. et al. Participation in robotics competition as motivation for learning1. *Procedia - Social and Behavioral Sciences*, Elsevier BV, v. 152, p. 835–840, out. 2014. Disponível em: <<https://doi.org/10.1016/j.sbspro.2014.09.330>>. Citado na página 13.
- CAPUTO, R. R. *Aplicação de classificadores bayesianos para tomada de decisão no futebol de robôs*. 2014. Monografia (Bacharel em Ciência da Computação), UFSJ. Citado na página 22.
- CHAVES, R. A. *UaiVision: um Sistema de Visão para Futebol de Robôs*. 2015. Monografia (Bacharel em Ciência da Computação), UFSJ. Citado 5 vezes nas páginas 7, 18, 19, 29 e 32.
- CONNOLLY, C.; BURNS, J.; WEISS, R. Path planning using laplace's equation. In: *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, 1990. Disponível em: <<https://doi.org/10.1109/robot.1990.126315>>. Citado na página 20.
- FARIA, G. *Uma arquitetura de controle inteligente para múltiplos robôs*. Tese (Doutorado), 2006. Disponível em: <<https://doi.org/10.11606/t.55.2006.tde-28022007-100455>>. Citado 3 vezes nas páginas 7, 20 e 21.
- FRANKLIN, G. F.; POWELL, D. J.; EMAMI-NAEINI, A. *Sistemas de Controle Para Engenharia*. [S.l.]: Bookman, 2013. Citado na página 22.
- GARCIA, R. O. *Sistema de monitoração de bloqueios tipo porta de vidro*. 2011. Trabalho de conclusão de curso(bacharel em Engenharia Elétrica) - Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá. Disponível em: <<https://repositorio.unesp.br/handle/11449/119217>>. Citado na página 29.
- GOODRICH, M. A. Potential fields tutorial. In: . [S.l.: s.n.], 2002. Citado na página 19.
- IEEE. *IEEE Very Small Size*. 2009. Disponível em: <[http://www.cbrobotica.org/wp-content/uploads/2014/03/VerySmall2009\\_ptbr.pdf](http://www.cbrobotica.org/wp-content/uploads/2014/03/VerySmall2009_ptbr.pdf)>. Citado 4 vezes nas páginas 7, 14, 17 e 25.

- JÚNIOR, E. P. e S. *Navegação exploratória baseada em problemas de valores de contorno.* Tese (Doutorado), 2003. Doutor em Ciência da Computação, UFRGS. Disponível em: <<https://lume.ufrgs.br/handle/10183/3819>>. Citado na página 20.
- KALMAN, R. E. *A new approach to linear filtering and prediction problems.* [S.l.]: Journal of Fluids Engineering, American Society of Mechanical Engineers, v. 82, n. 1, p. 35–45, 1960. Citado 3 vezes nas páginas 18, 22 e 33.
- KAMADA, E. et al. Tdp da equipe de futebol de robôs red dragons ufscar categoria ieee very small size. *Mostra Virtual de Robótica*, 2018. Disponível em: <<http://www.cbrobotica.org/mostravirtual/interna.php?id=25621>>. Citado na página 21.
- KHATIB, O. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, SAGE Publications, v. 5, n. 1, p. 90–98, mar. 1986. Disponível em: <<https://doi.org/10.1177/027836498600500106>>. Citado na página 19.
- LAIA, M.; CRUVINEL, P. Soil science tomographic projections filtering using discrete kalman and neural networks. *IEEE Latin America Transactions*, Institute of Electrical and Electronics Engineers (IEEE), v. 6, n. 1, p. 114–121, mar. 2008. Disponível em: <<https://doi.org/10.1109/tla.2008.4461640>>. Citado na página 33.
- LARC; CBR. *Robocup Soccer Simulation League 2D Rules*. 2019. Disponível em: <[https://bitbucket.org/larc\\_cbr\\_2d\\_simulation/larc-cbr-2019-simulation-2d/src/master/](https://bitbucket.org/larc_cbr_2d_simulation/larc-cbr-2019-simulation-2d/src/master/)>. Citado na página 15.
- LARC; CBR. *Robocup Soccer Simulation League 3D Rules*. 2019. Disponível em: <[http://www.cbrobotica.org/?page\\_id=118](http://www.cbrobotica.org/?page_id=118)>. Citado na página 15.
- LIM, Y. et al. Evolutionary univector field-based navigation with collision avoidance for mobile robot. *IFAC Proceedings Volumes*, Elsevier BV, v. 41, n. 2, p. 12787–12792, 2008. Disponível em: <<https://doi.org/10.3182/20080706-5-kr-1001.02163>>. Citado 5 vezes nas páginas 7, 21, 35, 36 e 37.
- MARTINS, M. F.; TONIDANDEL, F.; BIANCHI, R. A. C. Towards model-based vision systems for robot soccer teams. In: *Robotic Soccer*. I-Tech Education and Publishing, 2007. Disponível em: <<https://doi.org/10.5772/5126>>. Citado 3 vezes nas páginas 18, 31 e 50.
- MEHL, E. M. et al. *O 'FUTEBOL DE ROBÔS' COMO FERRAMENTA TECNOLÓGICA PARA O ENSINO DE ENGENHARIA ELÉTRICA E CIÊNCIA DA COMPUTAÇÃO*. 2001. Disponível em: <<http://www.eletrica.ufpr.br/mehl/downloads/COBENGE-2001.pdf>>. Citado 3 vezes nas páginas 7, 15 e 17.
- NEIVA, O. et al. Descrição da equipe sirsoccervss-a para a categoria ieee very small size soccer. *Mostra Virtual de Robótica. LARC 2015*, 2015. Disponível em: <<http://sistemaolimpico.org/midias/uploads/99312e0307a4639dabc55fa656cce3f1.pdf>>. Citado na página 29.
- OHTA, H. et al. Computer classification of rosette-forming cells from microscope images. *Systems and Computers in Japan*, Wiley-Blackwell, v. 18, n. 4, p. 64–75, 1987. Disponível em: <<https://doi.org/10.1002/scj.4690180407>>. Citado na página 29.

- PIRES, P. A. A. *Navegação de Robôs Móveis através de Campos Potenciais baseados em Problemas de Valor de Contorno*. 2016. Monografia (Bacharel em Engenharia Elétrica com habilitação nas áreas de Robótica e Automação Industrial), UFJF. Citado na página 21.
- PORTO, L. E. S. *Controle de movimento de um robô não holonômico com tração diferencial*. 2007. Monografia (Engenharia de Controle a Automação), UNB. Citado 2 vezes nas páginas 7 e 26.
- REIS, G. A. et al. *TDP 2013 da Equipe UaiSoccer de Futebol de Robôs – Small Size*. 2013. Disponível em: <<http://sistemaolimpo.org/midias/uploads/b012a085f6ffba6fd835284c05945feb.pdf>>. Citado na página 20.
- RESENDE, J. C. M. de et al. O impacto das fragilidades da comunicação 802.11 em competições de robótica. *ANAIIS DO WSCDC 2019*, Sociedade Brasileira de Computação (SBC), v. 2, n. 1, p. 53–57, 2019. Disponível em: <<http://sbcrc2019.sbc.org.br/wp-content/uploads/2019/05/wscdc2019.pdf>>. Citado na página 24.
- ROBOCUP. *RoboCup Leagues*. 2019. Disponível em: <<https://www.robocup.org/leagues>>. Citado 2 vezes nas páginas 7 e 16.
- ROSA, J. F. da. *Construção de um time de futebol de robôs para a categoria IEEE Very Small Size Soccer*. 2015. Monografia (Tecnólogo em Tecnologia da Informação e da Comunicação), FAETERJ. Citado 3 vezes nas páginas 18, 21 e 38.
- ROZA, F. S. da et al. Modular robot used as a beach cleaner. *Ingeniare. Revista chilena de ingeniería*, SciELO Comision Nacional de Investigacion Cientifica Y Tecnologica (CONICYT), v. 24, n. 4, p. 643–653, oct 2016. Disponível em: <<https://doi.org/10.4067/s0718-33052016000400009>>. Citado 2 vezes nas páginas 7 e 29.
- SANTOS, M. C. et al. Caboclinhos b: Artigo de descrição do time de futebol de robôs da universidade federal de sergipe na categoria vsss. *Mostra Virtual de Robótica*, 2018. Disponível em: <<http://sistemaolimpo.org/midias/uploads/edbf0aa9676823eb7092d9237d907f4.pdf>>. Citado na página 21.
- SCHWARTZ, W. et al. Reconhecimento em tempo real de agentes autônomos em futebol de robôs. In: *Simpósio Brasileiro de Automação Inteligente*. [S.l.: s.n.], 2003. Citado na página 15.
- SILVA, M. O. da. Campos potenciais modificados aplicados ao controle de múltiplos robôs. In: . Universidade de São Paulo Sistema Integrado de Bibliotecas - SIBiUSP, 2011. Dissertação (Mestre em Ciências - Ciência da Computação e Matemática Computacional.), USP. Disponível em: <<https://doi.org/10.11606/d.55.2011.tde-23112011-090055>>. Citado 2 vezes nas páginas 20 e 21.
- SOARES, M. S. et al. *TDP da Equipe de Futebol de Robôs Red Dragons UFSCar Categoria IEEE Very Small Size*. 2019. Disponível em: <<http://sistemaolimpo.org/midias/uploads/ed093fa72d45fdec7402b7f9a9af495.pdf>>. Citado na página 52.
- TAVARES, D. F.; PAULUCIO, L. S.; VALENTIM, R. V. Improvement of a robotic soccer team using a multilayered control architecture. *Mostra Virtual*

- de Robótica, 2015. Disponível em: <<http://sistemaolimpo.org/midias/uploads/a816a014c3a7360f4c201e99ebadd438.pdf>>. Citado na página 21.
- TRIVEDI, M. M.; ROSENFIELD, A. On making computers “see”. *IEEE transactions on systems, man and cybernetics*, v. 19-6, p. 1333–1336, nov. 1989. Citado na página 13.
- WU, J. et al. Analysis of strategy in robot soccer game. *Neurocomputing*, Elsevier BV, v. 109, p. 66–75, jun. 2013. Disponível em: <<https://doi.org/10.1016/j.neucom.2012.03.021>>. Citado na página 13.
- YIP, R. K.; TAM, P. K.; LEUNG, D. N. Modification of hough transform for circles and ellipses detection using a 2-dimensional array. *Pattern Recognition*, Elsevier BV, v. 25, n. 9, p. 1007–1022, sep 1992. Disponível em: <[https://doi.org/10.1016/0031-3203\(92\)90064-p](https://doi.org/10.1016/0031-3203(92)90064-p)>. Citado na página 18.
- ZAINAL, N. F. A.; ABDULLAH, S. N. H. S.; PRABUWONO, A. S. Adapting robot soccer game in student self-centered learning. *Procedia - Social and Behavioral Sciences*, Elsevier BV, v. 59, p. 130–137, out. 2012. Disponível em: <<https://doi.org/10.1016/j.sbspro.2012.09.256>>. Citado na página 13.