Universidade Federal do Pampa

Anderson Fortes

Identificação e Rastreamento de robôs no Contexto de uma Partida de Futebol de Robôs

Alegrete

2013

Anderson Fortes

Identificação e Rastreamento de robôs no Contexto de uma Partida de Futebol de Robôs

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr José Carlos Bins Filho

Alegrete

2013

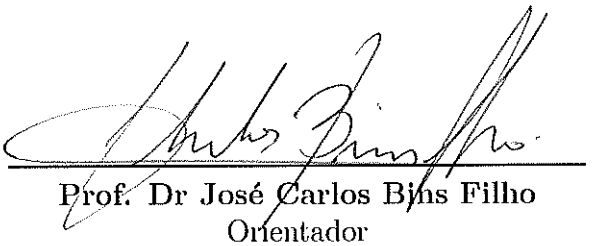
Anderson Fortes

Identificação e Rastreamento de robôs no Contexto de uma Partida de Futebol de Robôs

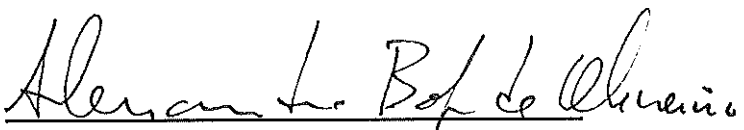
Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho de Conclusão de Curso defendido e aprovado em 4 de outubro de 2013

Banca examinadora:



Prof. Dr José Carlos Bins Filho
Orientador



Prof. Me. Alessandro Bof de Oliveira
UNIPAMPA

Prof. Dr. Ewerson Luiz de Souza Carvalho
UNIPAMPA

À minha família, com amor.

Agradecimentos

Em primeiro lugar a Deus pela força e por ter aberto os caminhos até aqui.

À minha família, principalmente aos meus pais Francisco e Iara por batalharem para me dar todo o apoio, sempre fazendo tudo o que estivesse ao alcance, me dando força, incentivo e confiança incondicional nesta caminhada. Agradeço também às minhas irmãs Margiani, Luciani e Carolini, por todo o carinho e alegria que me proporcionam nas horas em que estamos juntos.

Ao meu orientador Professor José Carlos Bins Filho pela ajuda, pelos ensinamentos, pela compreensão, e principalmente pela oportunidade de trabalhar ao seu lado. Agradeço também a todos os professores que tive durante a graduação, por todo o ensinamento passado e por trabalharem sempre para fazer dessa universidade uma instituição melhor.

Não posso deixar de agradecer também aos meus professores dos ensinos fundamental e médio, por todo o ensinamento, por alimentarem o sonho dessa conquista e por sempre lutarem para fazer da educação uma arma para construir cidadãos.

À gurizada da 303 pela companhia nas madrugadas adentro nas muitas noites de pesquisa, estudos e risadas.

Agradeço aos muitos amigos que fiz durante essa caminhada, pelos churrascos, pelas cervejas e principalmente pelo amparo nas horas difíceis e nos momentos de dúvidas. Enfim, agradeço a todos que de alguma forma ajudaram nesta conquista.

Muito obrigado a todos!

"Vinde a mim
todos que estais cansados e sobrecarregados,
e eu vos aliviarei.
Tomai sobre vós o meu jugo
e aprendei de mim,
porque sou manso e humilde de coração;
e achareis descanso para vossa alma.
Porque o meu jugo é suave,
E meu fardo é leve ."
(Jesus de Nazaré)

Resumo

A visão computacional é uma área de grande relevância no ponto de vista tecnológico. Ela está relacionada com diversas áreas de grande importância, como medicina, vigilância, robótica, entre outras. Uma dessas áreas, que apresenta grandes desafios não só para a visão computacional, mas para outros ramos como inteligência artificial, engenharia e matemática, é o futebol de robôs.Este trabalho apresenta um estudo sobre visão computacional voltada para o futebol de robôs. Nesse trabalho são apresentadas técnicas de identificação dos robôs e da bola usando como base as técnicas de subtração de background. São mostrados todos os passos necessários para o rastreamento dos objetos no campo de jogo. Também são demonstrados os resultados obtidos, como identificação da posição dos objetos e suas respectivas velocidades. Todos os algoritmos utilizados são abordagens que consideram apenas a utilização de câmera fixa. Ao final do trabalho é apresentada uma breve discussão sobre os resultados obtidos e possíveis trabalhos futuros.

Palavras-chave: Visão Computacional, Futebol de Robôs, Subtração de Background, Rastreamento.

Abstract

Computer vision is an area of great relevance in a technological standpoint. It is related to several areas of great importance, such as medicine, surveillance, robotics, among others. One such area that presents challenges not only for computer vision, but for other branches of artificial intelligence, engineering and mathematics is the soccer robots. This paper presents a study on computer vision toward the robot soccer. In this paper presents techniques for the identification of the robots and the ball using algorithms based on the background subtraction. Also shown are the results obtained, as identification of the position of objects and their speeds. All algorithms used are approaches that consider only the use of fixed camera. At the end of the work is a brief discussion of the results and possible future work.

Key-words: Computer Vision, Soccer Robots, Background Subtraction, Tracking.

Lista de ilustrações

Lista de siglas

GRU Grupo de Robótica da UNIPAMPA

HSV Hue Saturation Value

LAPIA Laboratório de Processamento de Imagens Aplicado

RGB Red Green Blue

RIA Robotic Institute of America

SSL RoboCup Small Size League

XML Xtensible Markup Language

Sumário

1 Introdução

Nos dias atuais, tem sido cada vez mais comum a utilização de equipamentos que visam facilitar as tarefas do dia a dia das pessoas. Em busca disso, tornou-se indispensável o estudo de novas tecnologias e novos equipamentos. Uma área em grande evidência que retrata bem esse crescimento é a visão computacional, a qual tornou-se um assunto bastante relevante, tanto do ponto de vista financeiro, quanto pelo olhar tecnológico. Tamanho atrativo se deve ao fato de que o estudo da visão computacional está ligado diretamente à áreas de grande importância como medicina, astronomia, vigilância, entre outras.

Um fato que também contribuiu para o avanço desta área foi a evolução dos computadores e câmeras de captura, a qual permite assim um grande avanço no que se diz respeito ao processamento de imagens e vídeos.

Como atualmente há grande interesse por parte de empresas e organizações em automatizar tarefas para diminuir custos de mão de obra, eliminar riscos de trabalho para as pessoas, ou até mesmo agilizar algumas atividades, uma outra área que tem ganhado grande importância é a robótica.

A robótica e a visão computacional estão relacionadas em um grande número de aplicações, uma vez que a visão computacional é bastante utilizada dentro da robótica. Como exemplo, podemos citar um sistema de reconhecimento automático de peças defeituosas em uma esteira, ou então o sistema de visão de um robô humanoide.

Dessa forma, com o tempo foram surgindo propostas para incentivar o estudo tanto da visão computacional quanto da robótica. Uma dessas propostas, que se mostrou bastante efetiva e atraente, é o futebol de robôs, uma vez que ele engloba também outras áreas de grande relevância como inteligência artificial, engenharia e microeletrônica.

Dentre os principais benefícios do estudo do futebol de robôs, está o fato de que ele é considerado uma potencial abordagem para tratar de desafios como ambientes dinâmicos, localização e navegação, colaboração entre robôs e aprendizagem.

Nesse contexto, o presente trabalho busca além de introduzir a importância da robótica e da visão computacional, desenvolver um sistema de visão computacional com aplicação para o futebol de robôs. A idéia básica do sistema consiste em realizar identificação e rastreamento dos robôs, para fornecer coordenadas dos mesmos para posteriores tarefas no contexto do jogo de futebol.

Nas seguintes seções serão apresentados tópicos como uma contextualização do futebol de robôs nas áreas de pesquisa, o que motiva o estudo e a pesquisa em sua área,

os objetivos gerais do trabalho e a metodologia utilizada na realização do trabalho.

1.1 Contexto e Motivação

Como já citado, assim como a visão computacional, a robótica também tem recebido bastante atenção por parte de pesquisadores. Ela tornou-se um ramo bastante aceito em países desenvolvidos. Segundo KIM Sung Ho; CHOI (1997), a pesquisa sobre robótica está relacionado às necessidades da comunidade científica, das políticas industriais e governamentais.

Uma grande área de pesquisa em robótica que atualmente recebe bastante atenção é a de sistemas com múltiplos robôs. Neste caso, os robôs podem agir de modo cooperativo, o que torna possível realizar tarefas que um único robô não seria capaz, otimizando a capacidade de trabalho.

Segundo RUIZ M.A.; URESTI (2008), um caso específico de sistemas com múltiplos robôs é o futebol de robôs. O futebol, um dos esportes mais praticados no mundo, é um típico jogo em equipe, onde cada jogador deve atuar de forma cooperativa. Outra característica é o fato do jogo mudar dinamicamente. Por causa destas características, o futebol tem sido considerado um problema padrão de sistemas com múltiplos agentes e algoritmos cooperativos.

Apesar do aspecto à primeira vista de um mero brinquedo, a proposta de uma partida de futebol jogada por robôs autônomos envolve muitas características de extrema complexidade do ponto de vista científico. Além de serem necessários robôs especialistas, o duelo principal no futebol de robôs é entre softwares, ou seja, a programação dos robôs, o que está relacionado diretamente à inteligência artificial, onde o time que conseguir um algoritmo de melhor desempenho provavelmente vencerá a partida. SOBREIRA Rodolfo Marengo; SILVA (2004)

Dessa forma, a escolha do futebol de robôs como aplicação para sistema de visão computacional do presente trabalho foi motivada pelo fato de permitir a possibilidade de abordar também tópicos desta área tão relevante que é a robótica.

1.2 Objetivos Gerais do Trabalho

O presente trabalho tem como objetivo geral desenvolver um sistema de rastreamento de robôs no contexto de uma partida de futebol de robôs, utilizando técnicas de rastreamento (*tracking*) baseadas em subtração de *background*. Essa tarefa pode ser resumida na identificação de cada um dos robôs, e detecção da posição (coordenadas) do mesmo no campo do jogo, detecção da posição da bola e as dimenções do campo.

Os objetivos gerais podem ser resumidos nos seguintes objetivos específicos:

- Capturar a imagem do campo onde ocorre a partida entre os robôs.

- Identificar a posição de cada um dor robôs e da bola (coordenadas)

- Detectar a movimentação dos robôs e da bola dentro do campo.

Todos esses objetivos técnicos, estão ligados a um objetivo maior, que é incentivar o início da busca por desenvolvimento e soluções para o futebol de robôs da Universidade Federal do Pampa, que conta com dois grupos de pesquisa nessa linha, o Laboratório de Processamento de Imagens Aplicado (LAPIA) e o Grupo de Robótica da UNIPAMPA (GRU).

O escopo desta pesquisa trás um embasamento teórico sobre as etapas do futebol de robôs, porém, a proposta de problema a ser solucionado limita-se à parte de visão computacional do sistema.

1.3 Organização do Documento

O restante do trabalho está organizado na seguinte sequência:

- Capítulo 2: Apresenta uma introdução à aplicação do problema e a descrição geral de um sistema de futebol de robôs. Este capítulo está focado em descrever a aplicação do problema proposto. Ele está dividido em duas partes. Na primeira parte é feita uma revisão de alguns conceitos básicos sobre robôs e a robótica, importantes para o melhor entendimento e contextualização do restante do trabalho. Na segunda parte é apresentada uma introdução e descrição mais detalhada sobre o funcionamento do futebol de robôs relacionado a este trabalho.

- Capítulo 3: Apresenta uma coleção de publicações pesquisadas em bases específicas da área que, de alguma forma, estão relacionadas com a visão de solução para o problema levantado. Neste capítulo é apresentada uma revisão da literatura, abordando trabalhos relacionados a este tema. São apresentadas algumas das diferentes técnicas usadas por autores para rastrear robôs e detectar objetos em movimento. Além disso, alguns dos trabalhos expostos trazem o conceito e técnicas utilizadas focadas ao futebol de robôs.

- Capítulo 4: Apresenta a descrição do problema e a metodologia a ser utilizada para chegar à solução. São apresentadas todas as etapas consideradas na sua resolução. São descritos os passos vistos de um nível mais alto, para introduzir as etapas que serão apresentadas de forma mais detalhada a partir do capítulo 5.

- Capítulo 5: Apresenta os algoritmos utilizados para a segmentação dos objetos de interesse e os resultados parciais obtidos pelos mesmos. Neste capítulo são apresen-

tadas as técnicas utilizadas para a resolução do problema proposto e os algoritmos de segmentação dos objetos do interesse do fundo da cena. Além disso, são apresentados resultados de testes realizados em diferentes técnicas propostas para o tema do trabalho. Nesse capitulo também é apresentado um estudo e avaliação de alguns algoritmos relevantes propostos para este tema. O objetivo principal deste capítulo é descrever técnicas utilizadas.

- Capítulo 6: Apresenta os resultados obtidos após a realização das etapas intermediárias na resolução do problema. Além disso, também são apresentadas as ferramentas utilizadas para a elaboração dos algoritmos, como suas características e informações a respeito de suas licenças e formas de obtenção dos mesmos. do presente trabalho.

- Capítulo 7: São descritas as considerações finais sobre o trabalho e as conclusões obtidas durante o desenvolvimento do mesmo. Além disso, é apresentada uma breve discussão sobre possíveis trabalhos futuros que podem complementar ou serem desenvolvido a partir deste.

2 Descrição da Aplicação

Neste capítulo será apresentada a descrição da aplicação proposta no presente trabalho. Para fins de um melhor relacionamento entre a proposta do trabalho com uma aplicação prática, este capítulo visa introduzir e apresentar o "alvo" deste trabalho, ou seja, o futebol de robôs como uma ferramenta que faz o uso da visão computacional descrita nas seguintes seções apresentadas. O principal objetivo deste capítulo é retratar o funcionamento do sistema de futebol de robôs como um todo, para isso, as seguintes seções apresentam a aplicação de forma mais detalhada, como a história da robótica, história do robô e do futebol de robôs.

2.1 Introdução

Um robô é um manipulador reprogramável, multifuncional, projetado principalmente para mover peças, materiais, ferramentas, e diversos outros equipamentos em movimentos variáveis programados para a realização de uma variedade de tarefas. (Robotic Institute of America (RIA) *apud* PEREZ (2005)). Segundo BEKEY (2005) *apud* PEREZ (2005), um robô é uma máquina que percebe, planeja e atua sobre um ambiente. O precursor do termo foi Karel Capek, novelista e escritor de uma peça teatral da Tchecoslováquia, que usou pela a palavra "robota" pela primeira vez, em 1920. Robota significa serviço compulsório, atividade forçada, e esse fato deu origem à palavra "*robot*" em inglês e traduzido para o português como "robô".(GROOVER M. P.; WEISS, 1989)

Durante anos foram imaginados robôs que pensavam e agiam como seres humanos, porém até então a maioria dos robôs criados tem como tarefas executar atividades repetitivas para substituir a mão de obra humana em tarefas consideradas perigosas, ou então simplesmente para aumentar a produtividade. Assim sendo, é notável que a maior beneficiada pelo desenvolvimento da robótica é a indústria.

Atualmente os investimentos realizados nas áreas de pesquisas sobre robótica aumentam a cada dia. Isso tem ocorrido pelo fato de empresas terem interesse em automatizar tarefas, que ao serem feitas por robôs, se tornam mais baratas. Diversas grandes empresas estão realizando altos investimentos para desenvolver novas tecnologias.

Com a intenção de deixar mais clara a área de aplicação deste trabalho, na seguinte seção serão apresentados tópicos como definição de um robô e uma breve história do futebol de robôs.

2.2 O Futebol de Robôs

O futebol de robôs foi criado em 1996 pelo professor Jong-Hwan Kim, do departamento de engenharia elétrica do KAIST (Korean Advanced Institute of Science and Technology), da República da Coreia. Segundo BIANCHI R. A. C; COSTA (2000), dispositivos mecatrônicos, hardware especializado para controle de sensores e atuadores, teoria de controle, interpretação e fusão sensorial, redes neurais, computação evolutiva, visão computacional, e sistemas de multi-agentes são exemplos de áreas de pesquisa que estão relacionadas a este desafio.

Com a difusão da idéia do futebol de robôs foi realizado o primeiro Campeonato Mundial de Futebol de Robôs no ano de 1996. Em poucos anos a plataforma já apresenta amadurecimento suficiente e novas tecnologias começam a aparecer.

No Brasil, o futebol de robôs foi introduzido em 1997 através do Centro Tecnológico de Informática do Instituto de Automação de Campinas-SP. O Primeiro Campeonato Brasileiro de Futebol de Robôs foi realizado na Universidade de São Paulo em 1998. Hoje em dia existem diversas outras instituições no Brasil que desenvolvem tecnologias nesta plataforma, dentre elas: CTI, USP, UFRS e UNESP. (GOMES MARCELO M.; KAWAMURA, 2001)

Um sistema que abrange todas as etapas envolvidas no futebol de robôs como um todo pode ser resumido em 4 etapas:

- Sistema Visão Computacional - Determina basicamente a posição, velocidade e direção dos objetos dentro do campo de jogo. Na grande maioria dos casos faz uso de uma câmera situada acima do campo, geralmente à uma distância mínima de dois metros. A imagem captada pela câmera é digitalizada e processada por um computador.
- Estratégia - A partir das informações obtidas pelo sistema de visão computacional são analisadas as situações do jogo, como posição de cada robô, posição da bola, e então são definidas estratégias para determinar como o time deve proceder.
- Sistema de Transmissão - Faz uso de um meio de transmissão para passar as orientações para os robôs.
- Robôs - Os robôs devem ser autônomos, isto é, não devem ser ligados através de nenhum meio físico, devendo todas as informações serem enviadas pelo ar.

As principais etapas, descritas acima, podem ser observadas na Figura 1.

Figura 1 – Principais etapas do sistema para o futebol de robôs.

Os componentes físicos utilizados na categoria de futebol de robôs utilizada para a aplicação deste trabalho podem ser resumidos como:

- Um campo de jogo, com dimensões aproximadas de 350 x 200 cm.
- Uma câmera.
- Um sistema de aquisição e processamento de imagens para cada time.
- Um computador.
- Um sistema de transmissão de dados.
- Um número n de robôs.

Podemos observar na Figura 2 um exemplo que retrata os recursos utilizados em boa parte dos sistemas de futebol de robôs.

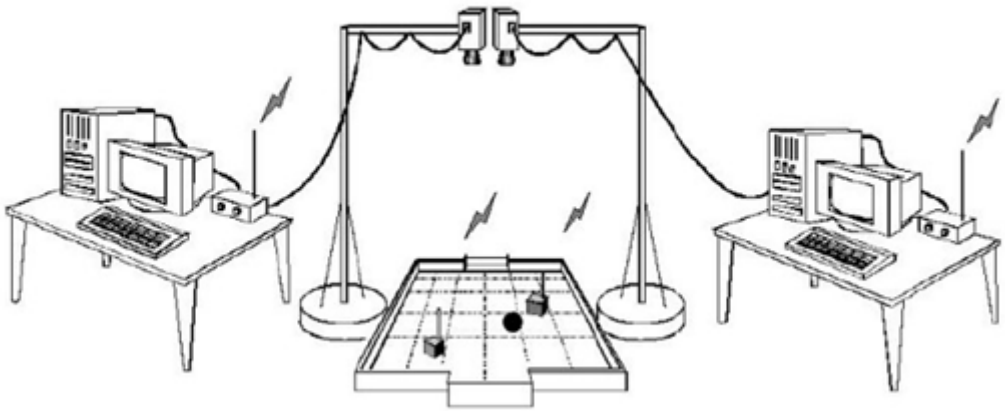


Figura 2 – Recursos utilizados na categoria de futebol de robôs relacionada a este trabalho.

3 Trabalhos Relacionados

Neste capitulo são apresentados alguns dos trabalhos relacionados encontrados na literatura. Na seção 3.1 são apresentados alguns trabalhos sobre o futebol de robôs, enquanto na seção 3.2 são demonstrados trabalhos que descrevem métodos e algoritmos voltados à subtração de *background*.

3.1 Trabalhos Relacionados sobre Futebol de Robôs

Na literatura, existem diversos trabalhos recentes acerca de métodos e técnicas que podem ser utilizadas como base ou complemento para a detecção e rastreamento dos objetos em movimento, assim como propostas para a implementação de um sistema de futebol de robôs completo.Segundo CUCCHIARA R.; GRANA (2003), a detecção de objetos em movimento é a primeira informação relevante para a maioria destas aplicações. Muitos trabalhos desenvolvidos nesta área tem como foco métodos para reconhecimento e rastreamento (*tracking*) de objetos em movimento.

Grande parte das técnicas propostas nos trabalhos encontrados utilizam uma câmera fixa para monitorar o ambiente de interesse e então o primeiro passo do método normalmente é realizar a subtração de background. Segundo JUNG C. R. ; MUSSE (2005), a remoção de *background* consiste em obter um modelo matemático do fundo da cena, e então cada quadro novo de imagem é comparado com esse modelo. As diferenças entre os dois modelos geralmente são associadas a objetos em movimento. Entretanto, existem diversos fatores que tornam essa tarefa difícil, como por exemplo a variação de luz no ambiente, ou então sombras de objetos, que podem também ser consideradas como objetos em movimento. Na literatura existem vários trabalhos que tratam de subtração de *background*, alguns inclusive propõem técnicas que levam em consideração as variações da iluminação no ambiente.

CUCCHIARA R.; GRANA (2003) propõe um método que considera 3 tipos de *background*:

1. O background é sempre estático;
2. O background varia de acordo com variações de luz;
3. O background varia quando objetos que estavam estáticos ganham movimentos, ou vice-versa.

O modelo apresentado pelo autor, lida com detecção errônea de "fantasmas", que acontece quando objetos que pertenciam ao modelo de background entram em movimento, o que fará com que haja mudanças em duas áreas da imagem: Onde o objeto está localizado agora, e a área que ele se encontrava originalmente. Para detectar os objetos em movimento, é usado um filtro temporal da mediana no espaço RGB. Além disso, o modelo também realiza a remoção de sombras, já que sombras detectadas como o objeto afeta muitas tarefas subsequentes como classificação e avaliação da posição dos objetos em movimento.

No trabalho de MCKENNA S.; JABRI (2000) é assumido que a câmera está parada e que as grandes mudanças de fundo são em relação aos movimentos das pessoas na cena. O modelo combina informação de cor e do gradiente para detectar objetos em movimento. O método consiste em três fases e produz uma máscara de segmentação de *foreground*. Também é assumido que alterações na ilumirnação ocorrem raramente em comparação com o movimento do objeto. O modelo de fundo é adaptado através de simples atualizações que ocorrem a cada período de tempo. De acordo com o autor, a idéia do método também é não rotular sombras como objetos. Segundo ele, qualquer mudança significativa de intensidade, sem mudança significativa de cromaticidade pode ter sido causado pela sombra, então deve ser rotulado como *background*.

Em JUNG et al. (2006), é proposto um modelo de rastreamento que incorpora remoção de sombras para detectar objetos em primeiro plano (bolhas) em tons de cinza. O modelo consiste em analisar a mediana temporal dos pixels, com tratamento de sombras e adaptação à variação de iluminação. Para acompanhamento da bolha, é calculada a soma das diferenças ao quadrado entre o modelo de fundo e o novo quadro (*frame*), e então é recuperada a posição do centro dessa bolha. Esse procedimento é repetido para todos os frames seguintes.

Após a identificação dos objetos em movimento, a grande maioria dos métodos tem como idéia central reconhecer a trajetória de cada objeto, ou seja, analisar a evolução deles no cenário ao decorrer do tempo. Segundo SOLDERA (2007), podem ser utilizadas informações de forma, cor, textura e consistência de movimento para identificar uma mesma bolha em quadros adjacentes de uma sequência de vídeo.

Alguns dos trabalhos, assim como este, apresentam propostas focadas na etapa de visão computacional com aplicação para o futebol de robôs. Dentre esses trabalhos, tem-se o trabalho de BORSATO F.H.; FLORES (2004), o qual propõe um método para detecção e acompanhamento em tempo real de objetos aplicados ao futebol de robôs. O sistema consiste em um módulo de calibração, que consiste em definir um modelo de campo vazio e um valor limiar mínimo de intensidade, seguido por um módulo de visão computacional aplicado para detectar e rastrear os objetos no campo de jogo. O módulo de visão recebe como entrada um quadro e tem como saída a posição, a direção e a velocidade dos robôs

e da bola.

No trabalho de COSTA A.H.L; PEGORARO (2000), o reconhecimento dos objetos de interesse (robôs e bola) no campo é baseado nas cores da imagem capturada pela câmera, sendo que o conjunto de pixels adjacentes conectados com cores similares são considerados componentes de um mesmo elemento, desde que o número de pixels conectados esteja na faixa de limiares previamente estabelecido na fase de calibração do sistema. Pixels conectados de cor laranja representam a bola, de cor azul os robôs de um time, e de cor amarela, os robôs do outro time. O modulo de visão computacional, possui uma fase inicial de calibração, executada previamente ao início da partida, que também define um modelo de campo vazio. A visão identifica, localiza e rastreia a bola e os jogadores no campo.

Como os robôs de um mesmo time possuem as mesmas cores, é necessário o uso de um operador humano no início da partida para identificação de cada robô individualmente. A partir deste instante, a identificação se dá por intermédio de um algoritmo de rastreamento, que usa um método de otimização exaustiva para encontrar a solução de mínimo custo.

No trabalho ZICKLER S.; LAUE (2009), é proposto um sistema de visão computacional padrão para todas as equipes participantes do campeonato mundial de futebol de robôs categoria small - RoboCup Small Size League (SSL). Segundo Zickler, o fato de a maioria das equipes, preferir utilizar duas câmeras, uma cobrindo cada metade do campo, faz com que seja necessário um tempo muito grande de preparação e configuração do sistema de visão de cada equipe antes e até mesmo durante a competição. Assim sendo, a comissão da SSL decidiu migrar para um sistema de visão compartilhada, o SSL-Vision, em que câmeras são conectadas a um servidor de processamento de imagem que transmite a saída da visão para as equipes participantes.

Zickler divide a descrição do sistema em várias etapas, na primeira é descrito o *loop* de captura de quadros. Esse plugin de captura produz uma imagem de saída que é associada a um "compartimento". Em cada iteração de captura, a pilha de câmera-única é associada a um desses compartimentos, onde é guardada a imagem capturada e alguma informação adicional resultante do processamento desta imagem.

A segunda etapa descreve a configuração dos parâmetros do sistema. Para que a configuração seja fácil, todos os parâmetros de configuração do sistema são representados de uma forma unificada através de um sistema de gestão variável. Esse sistema de gestão permite a organização de parâmetros de tipos arbitrariamente complexos, utilizando armazenamento de dados baseado em Xtensible Markup Language (XML).

A terceira etapa abordada no trabalho, descreve o sistema de processamento da imagem. Nessa etapa, os robôs devem ser identificados unicamente.

Zickler propõe que cada robô será identificado pelas cores de cima. Cada robô transporta um marcador no centro área a identificação de qual equipe ele pertence, bem como um arranjo adicional de marcadores coloridos de modo a proporcionar o robô um identificador único, como podemos notar na Figura 3.

Figura 3 – Robôs com identificadores por cores em sua parte superior.

Fonte: ZICKLER S.; LAUE (2009).

Para a segmentação de cores, o algoritmo assume a existência de uma tabela de pesquisa que mapeia a cor de entrada para uma etiqueta de cor única, que é capaz de representar qualquer uma das cores capturadas. O algoritmo percorre cada pixel da imagem e então converte cada pixel de cor original para sua etiqueta de cor correspondente.

Após a limiarização da imagem, é calculada as caixas delimitadoras e centróides de todas as regiões mescladas e então elas são classificadas por cor e tamanho. Na seqüência, é realizada a calibração do sistema, que utiliza a imagem do campo e as dimensões reais do campo, que são definidas em normas da liga. A seguir, é realizada uma projeção de um ponto M tridimensional do campo em um ponto m bidimensional no plano de imagem. Os parâmetros do modelo para esta função são orientação q e a posição t, transformando pontos a partir do sistema de coordenadas do campo para o sistema de coordenadas da câmera. Então, um detector de borda é aplicado para localizar as linhas usando a sua posição prevista calculada a partir da estimativa e das dimensões do campo.

Depois que todas as regiões foram extraídas a partir da imagem de entrada e todas as suas coordenadas do mundo real terem sido calculado, o passo seguinte é o reconhecimento de padrões. O objetivo deste passo é extrair as identidades, locais, e orientações de todos os robôs, bem como a localização da bola.

3.2 Trabalhos Relacionados sobre Subtração de Background

Como uma etapa considerada chave em nosso sistema é a de subtração de *background*, nessa sessão é descrita de forma detalhada e exemplificada essa técnica.

Na literatura existe uma grande variedade de trabalhos que abordam modelos e técnicas de subtração de background. Abordagens que vão de algoritmos com técnicas triviais até algoritmos que tratam de remoção de sombras e problemas com a variação de iluminação. Alguns algoritmos exploram a diferença estatística e ou probabilística de cor entre a imagem atual e a imagem de modelo de background, a qual é treinada durante um período de tempo ou um número determinado de imagens (TRUYENQUE, 2005).

Um dos grandes problemas, enfrentado pelos diferentes tipos de abordagem, é a mudança de iluminação e presença de sombras. A mudança na iluminação acarretará em valores de pixels diferentes aos do modelo de fundo, o que irá prejudicar o desempenho do algoritmo. A movimentação do objeto gera sombras do mesmo na cena, e este problema, leva ao fato de que algoritmos que não tratam a remoção de sombras reconheçam-nas como parte dos objetos em movimento, o que afeta na forma geométrica do objeto detectado. Podemos encontrar diversos algoritmos que consideram esses fatores. Alguns deles lidam parcialmente com sombras, enquanto outros dedicam-se fortemente ao tratamento ou remoção das mesmas.

ELGAMMAL Ahmed; HARWOOD (2000) descrevem uma técnica que desconsidera o brilho dos pixels, uma vez que, em teoria, a sombra possui as mesmas coordenadas cromáticas, com uma mudança apenas no brilho do pixel. Como podemos notar na Figura 5, os pixels próximos ao robô sofrem uma diminuição na iluminação, causada pela diminuição da luminosidade. Estes pixels não mudam de cor, portanto, algoritmos que identifiquem que houve apenas mudança na iluminação e não na cor são uma boa alternativa para alguns tipos de aplicações.

JACQUES J.C.S.; JUNG (2005) propõem um método de correlação cruzada normalizada, em que é verificado na comparação entre as duas imagens se houve uma mudança apenas na intensidade da iluminação em determinada área, e não uma grande variação nos valores de cor dessa área. Na ocorrência desse critério, é considerado que esses pixels representam a sombra de um objeto, e não o próprio objeto.

Para observar a diferença de resultados entre algoritmos que não tratam a presença de sombras para os que lidam com essas características, podemos observar a Figura 4, que mostra o resultado da subtração de background que desconsidera as informações de luminosidade, ou seja, são consideradas apenas as informações de valor das cores de uma imagem.

Na Figura 4, podemos notar facilmente uma grande diferença no resultado da definição do objeto de interesse. Muitos pontos da sombra do objeto foram considerados

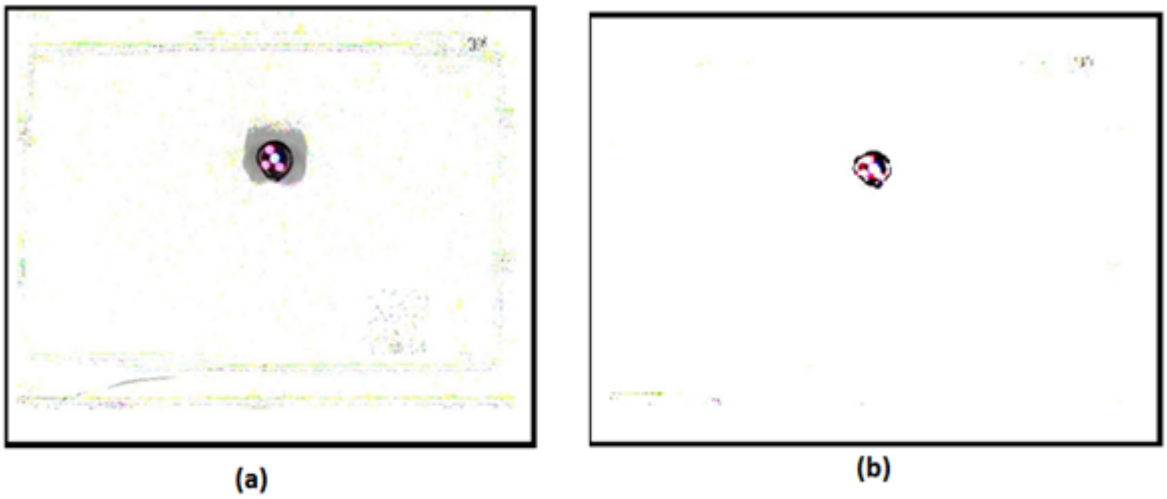


Figura 4 – Subtração de background utilizando remoção de sombras.

Figura (a): resultado da subtração sem remoção. Figura (b): Resultado da subtração aplicando a remoção de sombras.

como sendo o objeto na parte esquerda da figura. Estas características muitas vezes limitam a eficácia de diversos algoritmos, pois geralmente deformam a morfologia real do objeto durante a segmentação. Uma alternativa a esse problema é ajustar o limiar, ou seja, alterar o valor de *threshold*, porém, é difícil estabelecer um valor de limiar que exclua apenas os pixels de sombra sem afetar no restante da subtração. Na parte direita, podemos notar que não existe sombra ao redor do robô, o que pode tornar o sistema muito mais eficaz.

No trabalho CHEUNG (2000), é apresentada uma técnica robusta de modelagem de fundo e segmentação de objetos que lida parcialmente com a remoção de sombras. Para a modelagem de fundo, no processo de treinamento do algoritmo apresentado em CHEUNG (2000), é criada uma imagem média, que é calculada a partir de um determinado número de imagens de background. Essa média é calculada utilizando o espaço RGB, pois os valores de cor não são decompostos nem transformados neste algoritmo. Portanto, neste algoritmo é necessário um tempo de treinamento para a modelagem da imagem de referência. Para a etapa de subtração do algoritmo, é utilizado um método baseado no cálculo do ângulo entre cada uma das três cores para cada ponto a ser analisado da imagem, tanto de referência quanto da atual. Então cada um desses pixels são classificados conforme o ângulo formado entre as cores.

O cálculo do ângulo não é aplicado a todos os pontos da imagem, pois alguns pixels são excluídos do teste através de uma verificação de cores, ou seja, quando o resultado da

subtração é uma distância muito grande entre os valores de cores e maior que um limiar estabelecido T_U, não é necessário o teste do ângulo, e então esse pixel já é considerado como objeto de interesse (*foreground*). Se não houver uma diferença muito grande entre as cores dos pontos então quer dizer que este ponto da imagem atual possui cores bem parecidas com o mesmo ponto da imagem de referência. E se o resultado dessa subtração entre as duas imagens for um valor menor que um segundo limiar T_L, ele é considerado como parte do background, e assim também não é necessário que seja aplicado o teste do ângulo. Então, os pixels que não foram ainda classificados como background ou foreground nos casos acima, ou seja, se os valores resultantes da subtração se encontram entre os dois limiares T_U e T_L, eles são submetidos ao teste do ângulo.

Para classificar os pixels sujeitos ao teste do ângulo, é utilizado um terceiro limiar T_C. O resultado do cálculo do ângulo é comparado com esse valor limiar estabelecido. Se o ângulo $\Theta < T_C$, o pixel é considerado como foreground, caso contrário é classificado como background.

Apesar de muitos algoritmos utilizarem muitas vezes um aprendizado estatístico para uma seleção automática de limiares, o que é feito para um melhor resultado na subtração de background, o trabalho apresentado por CHEUNG (2000) não apresenta um procedimento específico para a escolha dos valores de limiar. Como a parte principal do algoritmo possui apenas um limiar, a definição desse valor pode ser trivial por um método de tentativa e erro, até que se alcancem resultados satisfatórios.

No algoritmo proposto por CHEUNG (2000) um eventual movimento de objetos do fundo da cena após o período de treinamento seria considerado como objeto de interesse. Além disso, a abordagem não pode lidar com as mudanças de iluminação na cena. Para a solução desses problemas, muitos métodos que tratam da adaptação do modelo de fundo tem sido propostos. KOLLER (1994) utilizam um filtro de Kalman para acompanhar as mudanças na iluminação de fundo para cada pixel. FRIEDMAN e RUSSELL (1997) modelam cada pixel com um modelo de mistura paramétrica adaptativa de três distribuições gaussianas. Ambos os métodos, conseguem lidar bem com mudança na iluminação, porém não tratam do problema de objetos serem introduzidos ou removidos do fundo da cena.

O processo de segmentação de objetos em movimento a partir de uma sequência de imagens é uma das principais etapas da visão computacional. Essa fase muitas vezes representa um problema crucial a ser tratado na maioria dos sistemas de visão computacional.

Uma das principais abordagens usadas na segmentação de objetos em movimento é a subtração de fundo. A subtração de fundo consiste em subtrair uma imagem atual de uma imagem de referencia, a qual é adquirida e modelada a partir de um fundo estático durante um período de tempo, chamado também como período de treinamento.

A qualidade dos resultados obtidos na segmentação por subtração de background depende em grande parte da forma com que é definido o modelo de fundo usado como referência para a subtração posterior.

Na próxima seção são apresentados alguns métodos de elaboração do modelo de fundo.

3.2.1 Modelo de Fundo

Uma etapa considerada chave na subtração de background é a definição de um modelo de fundo, como representá-lo para que haja um melhor desempenho no processo de segmentação. Existem diversas abordagens para a elaboração desse modelo, algumas muito simples e outras mais robustas e "espertas".

Uma forma trivial e intuitiva de elaboração de um modelo de fundo é simplesmente calcular uma imagem com base na média obtida entre diversas imagens, dado um conjunto de imagens de background. Segundo TRUYENQUE (2005), nessa abordagem, quando um objeto de interesse entra na cena, ele poderá ser identificado facilmente, uma vez que haverá uma mudança significativa no nível de pixel entre a imagem de fundo obtida previamente e o novo frame a ser analisado. Apesar dessa abordagem ser teoricamente funcional e aceitável, ela requer um período de ausência de objetos em primeiro plano, os de nosso interesse (KAEWTRAKULPONG P.; BOWDEN, 2001). Além disso, existe um conjunto de problemas que a tornam uma tarefa mais complicada, como a incapacidade de lidar com as mudanças da iluminação da cena.

Na Figura 5 podemos observar um exemplo de subtração de *background*. Para deixar claro a importância das diversas técnicas de definição do modelo de background (campo vazio), na primeira parte da figura é demonstrado um background que não utiliza técnica alguma. Simplesmente foi obtida uma única imagem de background, sem os objetos de interesse, para então ser realizada a segmentação dos objetos através da subtração de background. A segunda figura retrata um frame com um objeto de interesse. O resultado desse processo pode ser visualizado na terceira parte da figura.

Como podemos observar, existe muito ruído na imagem resultante, pois nesse processo não é utilizado nenhuma técnica inteligente, tanto na obtenção do modelo de fundo, quanto na subtração. Obviamente, a olho humano, pode parecer um resultado aceitável, uma vez que podemos observar que o objeto mantido na imagem realmente foi o que esperávamos. Porém, para fins computacionais, os resultados são muito ruins, já que existem diversos pontos de "sujeira" na imagem resultante, além de que uma grande região de sombra do robô também foi detectada como objeto de interesse, o que deve atrapalhar e muito nos objetivos destes sistemas.

Tanto para a elaboração do modelo de fundo, quanto para a subtração entre o

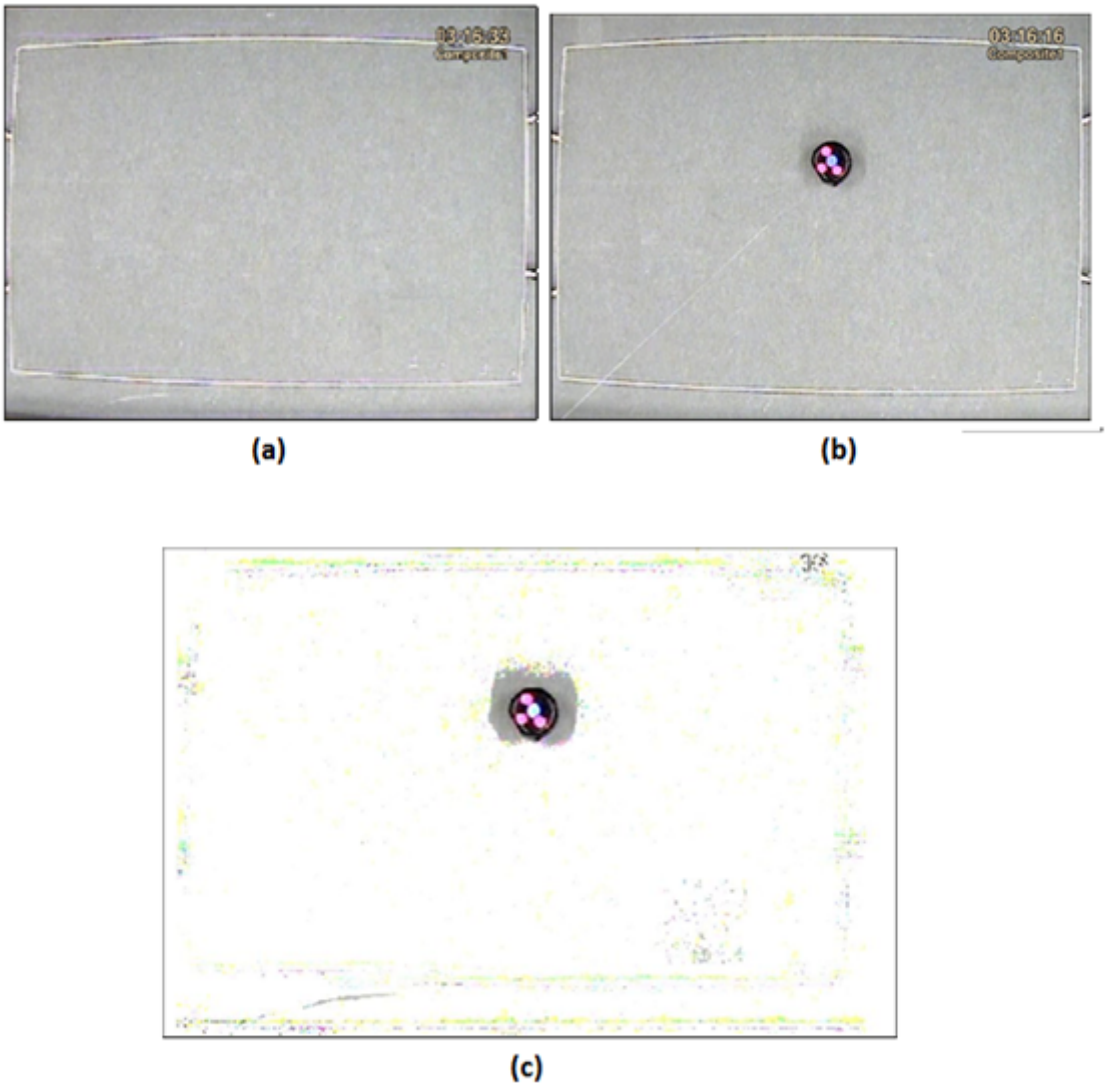


Figura 5 – Demonstração da subtração de background sem a utilização de técnicas robustas. É apresentado um modelo de fundo (a), um segundo frame com um robô (b), e o resultado da subtração entre essas duas imagens (c).

frame corrente e esse modelo, podemos aplicar até mesmo conceitos básicos e triviais a fim de melhorar os resultados, como o uso da média dos pixels na elaboração da imagem de referência ou utilização de um valor limiar para a subtração entre o frame corrente e o modelo de fundo. Logo, podemos concluir que a abordagem apresentada na Figura 5 é considerada inaceitável, considerando que aplicação de algumas técnicas simples já podem melhorar muito os resultados. Entretanto, o objetivo deste trabalho é utilizar métodos mais robustos e eficientes. Alguns desses métodos são apresentados na próxima seção.

4 Metodologia

Neste capítulo é apresentada a metodologia utilizada para a resolução do problema proposto. Na primeira seção é apresentado o problema de forma geral. Além disso, na segunda seção é dada uma introdução ao processamento de imagens, e então é apresentada a metodologia utilizada na seção 4.3.

4.1 Problema

O problema a ser resolvido consiste na primeira fase de um sistema para futebol de robôs. Nessa fase, a câmera, situada acima do campo de jogo, ligada a uma placa de captura de imagens, deve capturar as imagens da partida, que serão processadas pelo sistema de computador capaz de detectar e rastrear os robôs e a bola, assim como fornecer outros dados que serão repassados para a faze de estratégia do sistema.

Como já apresentado na seção 1.3, o problema pode ser dividido basicamente nas seguintes etapas:

- Capturar a imagem do campo onde ocorre a partida entre os robôs.

- Fazer a segmentação dos objetos de interesse (robô e bola).

- Identificar a posição e velocidade de cada um dos objetos (coordenadas).

Neste capítulo serão apresentadas de forma mais detalhada as etapas realizadas para a resolução do problema, como características da segmentação de objetos, principalmente da abordagem baseada na subtração de background. Para isso será mostrado um estudo feito sobre alguns dos principais algoritmos para detectar objetos em um fundo estático.

O objetivo deste capítulo é estabelecer um modelo de algoritmo capaz de detectar e segmentar os objetos a partir da sequência de frames obtidos pela câmera. Neste capítulo serão descritas todas essas etapas. Para isso, foram estudados diversos algoritmos, com o intuito de escolher a abordagem que mais se adapta às necessidades do problema descrito no presente trabalho. Nas seguintes seções são apresentados os métodos escolhidos para a utilização no desenvolvimento do mesmo.

4.2 Conceitos Fundamentais de Processamento de Imagens

O primeiro passo para o começo do processamento do sistema de visão computacional é a captura de imagem, onde são obtidas as imagens do campo. Estas imagens são

obtidas através de um loop de captura de quadros. Esse sistema de captura produz uma imagem de saída para posterior processamento. Como o foco principal do nosso trabalho é desenvolver um sistema que identificará os objetos de interesse a partir de uma sequência de imagens, faremos uso extensivo de técnicas de processamento de imagens. O principal objetivo de um sistema de processamento de imagens é extrair as informações que são consideradas relevantes para determinada aplicação. Segundo CHACON G. GASTARDELLI (2011) um sistema deste tipo é composto de uma série de etapas, que são ilustradas na Figura 6.

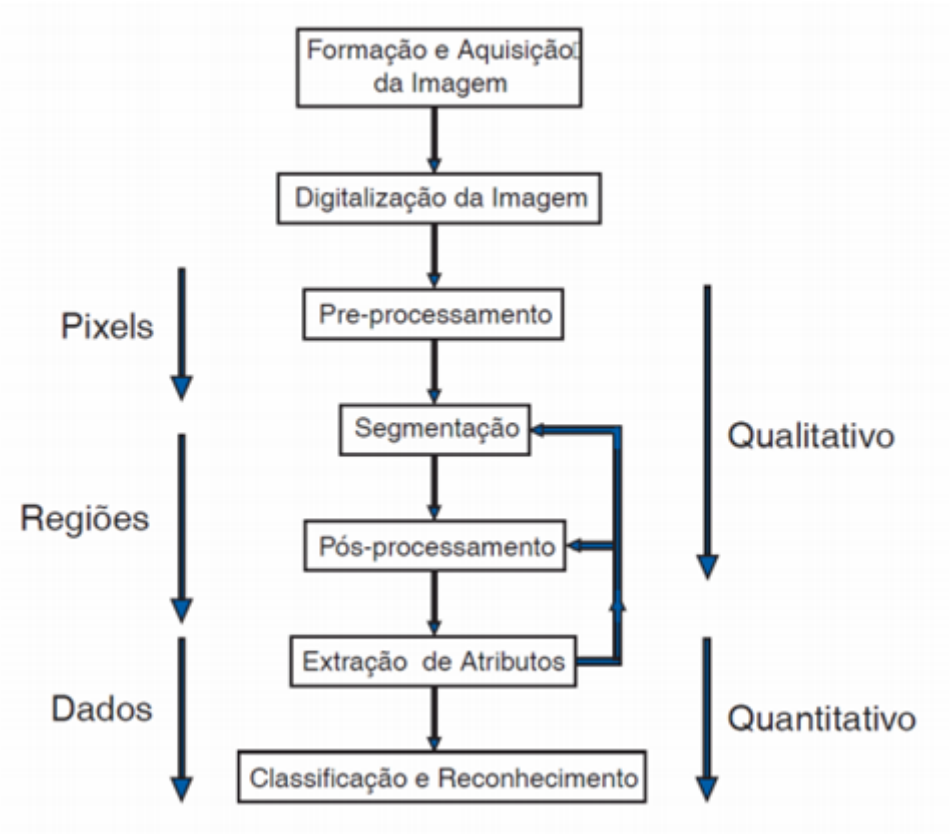


Figura 6 – Fases de um sistema de processamento de imagens.

Fonte: (CHACON G. GASTARDELLI, 2011).

A imagem é geralmente organizada como uma matriz de pixels em uma simetria quadrada, pelo fato de isso facilitar a manipulação e visualização dessas imagens de forma eletrônica.

As duas primeiras etapas da Figura 6 representam a forma de captura das imagens, as quais são obtidas através de uma câmera e digitalizadas pelo computador. A seguir são apresentados detalhes das 3 etapas seguintes.

Pré-processamento:

As técnicas de pré-processamento tem como objetivo de melhorar a qualidade da imagem, de acordo com as necessidades do sistema, como realçar ou melhorar a qualidade de alguma parte específica da imagem. No nosso sistema, uma técnica de pré-processamento é a obtenção de um modelo de background, pois em alguns algoritmos é necessário definir um modelo de referência previamente, para então poder realizar a segmentação dos objetos.

Segmentação:

A segmentação da uma imagem basicamente se resume em separar as partes de interesse. Essa etapa, dentre todas do processo, é considerada a mais crítica, uma vez que é nessa etapa que são definidas as regiões de interesse para as seguintes tarefas no processamento da imagem, ou seja, se essa etapa não for realizada com sucesso, e os resultados não forem satisfatórios, isso afetará todo o restante do sistema.

Existem diversas técnicas de segmentação, em que cada uma se adéqua a alguns tipos específicos de imagens e aos resultados que se deseja obter. No nosso trabalho, utilizamos duas técnicas de segmentação que parte da mesma abordagem: a similaridade entre os pixels, pois essa técnica é utilizada na subtração de *background*. Ambas as técnicas utilizadas em nossa proposta aplicam a técnica de binarização, e serão apresentadas nas próximas seções deste capítulo.

Pós-processamento:

O objetivo da fase de pós-processamento consiste em melhorar os resultados obtidos na fase de segmentação. Na nossa abordagem utilizamos técnicas de morfologia matemática para eliminar alguns ruídos e destacar os objetos de interesse. A morfologia matemática e os filtros morfológicos que utilizamos são descritos na subseção 4.3.4.1.

4.3 Metodologia

Nesta seção é apresentada a metodologia utilizada para a resolução do problema, como descrito na seção 4.1. Nas proximas seções são apresentados de forma detalhada todos os algoritmos e passos escolhidos para elaboração da solução e obtenção dos resultados esperados.

4.3.1 Calibração

De um modo geral, a etapa de calibração serve para preparar o sistema de acordo com as características do ambiente de jogo.

A fim de deduzir informações sobre os objetos no campo a partir das medições da câmera, definimos uma relação entre a geometria do campo e a imagem. Isso serve para definir uma relação Tamanho de imagem x Tamanho do campo. Esta etapa é usada para normalizar o robô e posições de bola a partir de coordenadas de imagem (em pixels) para reais coordenadas espaciais (em metros).

4.3.2 Segmentação por subtração de Background

Para a realização da segmentação por subtração de *background* no nosso algoritmos, utilizamos como base o método apresentado no trabalho de KAEWTRAKULPONG P.; BOWDEN (2001), pois o mesmo trata dos problemas de iluminação e de objetos serem introduzidos ou retirados do modelo de fundo, já que esse modelo é atualizado a cada iteração.

Apesar da aplicação do nosso trabalho não sofrer dos problemas de variação de iluminação e de a cena de fundo não apresentar mudanças com relação a novos objetos ou a retirada dos que já se encontravam na cena, nosso trabalho aplica a abordagem de atualização do modelo de fundo proposta por KAEWTRAKULPONG P.; BOWDEN (2001), uma vez que a forma de captura das imagens resulta em constantes variações consideráveis dos valores de pixels. Nesta abordagem, os autores se baseiam no trabalho de STAUFFER (1999), em que cada pixel é modelado com uma mistura de distribuição Gaussiana.

Nessa abordagem, a cada iteração são adicionados novos exemplos ao conjunto de treinamento, e descartadas as antigas. Diferentes valores gaussianos são assumidos para representar cores diferentes. Cada pixel do fundo é modelado por uma mistura de k distribuições Gaussianas ($k = 3$ a 5). Os pesos da mistura representam as proporções de tempo que as cores ficam na cena. As prováveis cores de fundo são as que permanecem mais tempo de forma mais estática.

Para a estimativa do fundo, um limiar mínimo T é usado que é a mínima probabilidade anterior é de o background estar na cena. Qualquer pixel que tem um desvio padrão maior que 2,5 em relação a qualquer uma das N distribuições é considerado como foreground.

O algoritmo proposto por KAEWTRAKULPONG P.; BOWDEN (2001) estima o modelo de mistura gaussiana por estatísticas de equações de atualização. Quando as L primeiras amostras são processadas, as equações são realizadas com base nas L recentes amostras, ou seja, são dadas prioridades aos dados recentes para que haja adaptação às recentes mudanças da cena.

Para que o modelo se adapte às mudanças na iluminação, um sistema de atualização foi aplicado. Este baseia-se na atualização seletiva. Cada novo valor de pixel

é comparado com os componentes existentes no modelo, e então se não é encontrado nenhuma correspondência, um novo componente de Gauss será adicionado.

4.3.2.1 Detecção de Sombra

Para que o algoritmo trate dos problemas de reconhecimento de sombras, utilizamos o conceito de separação dos valores de cromaticidade e de brilho. Isto é feito através da comparação de um pixel considerado objeto em movimento com os pixels do modelo de fundo. Se a diferença entre as componentes cromáticas e brilho estão dentro de certos limites, o pixel é considerado como uma sombra.

A remoção de sombra no nosso trabalho é baseada no algoritmo proposto por KAEWTRAKULPONG P.; BOWDEN (2001), que calcula a diferença entre as componentes cromáticas e de brilho entre o fundo e os objetos considerados como foreground. Se essa diferença está entre certos limites, o pixel é considerado como uma sombra. O algoritmo utiliza um modelo computacional que é constituído de um vetor que guarda as médias dos pixels de fundo do quadro no formato RGB, E, uma linha de cromaticidade $||E||$, uma distorção cromática d, e um limiar de brilho T. Para um dado valor do pixel observado, I, uma distorção brilho, a, e uma distorção da cor, c, a partir do modelo de *background*, podem ser calculados como

$$a = argmin(I - zE)^2$$

e

$$c = ||I - aE||$$

Com o pressuposto de distribuição de Gaussiana esférica em cada componente da mistura, o desvio padrão do K^{th} componente σ_k k pode ser definida igual a d. O cálculo de a e c são triviais usando produto escalar do vetor. (KAEWTRAKULPONG P.; BOWDEN, 2001) Um objeto de foreground é considerado sombra se movendo se a está dentro de um desvio padrão de 2,5 e $\tau < c <$1.

4.3.3 Segmentação por Cor

Outro método bastante utilizado de segmentação de objetos é o que utiliza informações de faixas de cores dos objetos a serem identificados. Esses algoritmos são uma opção atraente por causa de sua simplicidade e robustez sob oclusão, profundidade e escala mudanças parciais (KRAVTCHENKO, 1999). Porém, existem alguns problemas nesse tipo de abordagem. O maior desses problemas está relacionado a constância de cor, a qual afeta o desvio de cor ao efeito de iluminação. Questões como sombras, mudanças

nas características de iluminação e câmera de afetar o fenômeno da constância de cores (SATTAR; DUDEK, 2006).

Como necessitamos que a identificação de cor seja realizada de uma forma robusta, já que deve será aplicada em tempo real, necessitamos que as cores sejam representadas de uma forma simples e eficiente. Segundo KLINKER, SHAFER e KANADE (1990), a representação do espaço de cor representa o ponto que mais influencia na robustez de algoritmos que consideram o espaço de cores. Dois tipos de representação muito usados em visão computacional são Red Green Blue (RGB) e Hue Saturation Value (HSV).

RGB significa vermelho, verde e azul, respectivamente, e é um sistema de cores aditivas. As três cores primárias vermelho, verde e azul são combinadas para formar a cor desejada (JAYASHREE, 2013).

Figura 7 – Representação gráfica do espaço de cores RGB.

Para a identificação de cor em nosso algoritmo, primeiramente transformamos a imagem capturada, que é representada no espaço de cor RGB, para uma representação no espaço HSV, que representa *Hue* (matiz), *Saturation* (saturação) e *Value* (valor). Essa transformação é realizada pelo fato de que a representação HSV separa as informações de intensidade da imagem das informações de valor, que representam as cores. Isso facilita atividades que tratam de mudanças de iluminação e de remoção de sombras.

Existem diversas outras formas de representação de cores que separam valor da intensidade, porém, nós utilizamos HSV pela facilidade de conversão entre RGB e HSV, como por exemplo, no nosso trabalho, utilizando a função *cvCvtColor* (*BGR2HSV*), da bilioteca *OpenCv*, que será apresentada no próximo capítulo.

Após a transformação da imagem para o formato HSV, realizamos a segmentação com base na faixa de cor desejada. Esse processo é realizado através da função *inRange*(), que basicamente verifica se os elementos de uma determinada matriz (ou *frame*, no nosso caso) ficam entre os elementos de dois intervalos, ou outras duas matrizes. A Figura 8 mostra o algoritmo do nosso trabalho que executa esses passos.

```
1   //Esta função irá ter uma imagem, e retornar uma imagem binária
2
3   IplImage* GetThresholdedImage(IplImage* img)
4   {
5       IplImage* imgHSV = cvCreateImage(cvGetSize(img), 8, 3);
6       cvCvtColor(img, imgHSV, CV_BGR2HSV);
7       //cvShowImage("Imagem HSV",imgHSV);
8       IplImage* imgThreshed = cvCreateImage(cvGetSize(img), 8, 1);
9
10      cvInRangeS(imgHSV, Scalar(H_MIN,S_MIN,V_MIN), Scalar(H_MAX,S_MAX,V_MAX), imgThreshed);
11      // os dois 'cvScalar' representam o limite inferior e superior de valores que são de cor alaranjada.
12      cvReleaseImage(&imgHSV);
13
14       return imgThreshed;
15  }
```

Figura 8 – Algoritmo que realiza a conversão de uma imagem RGB para uma imagem no formato HSV.

A função realiza a conversão (linha 6), seleciona os pixels que estão entre o intervalo de cor especificado (linha 10) e retorna a imagem resultante (linha 14).

Como resultado dessa função, é retornada uma imagem binária em que os pixels equivalentes aos da imagem original que se encontram entre o intervalo estipulado, são retornados com valor zero (branco), enquanto os outros são retornados com valor 1 (pretos).

A imagem retornada pela função a partir de um frame de entrada, em que consideramos apenas os valores de pixel entre os intervalos (H= 0, S=110, V=200) e (H=13 , S=256, V=256) pode ser vista na Figura 9.

4.3.4 Refinamento dos resultados da Segmentação

Para muitos algoritmos, o resultado tanto da subtração de fundo quanto da segmentação por cor pode apresentar muito ruído. Isso pode acontecer por uma série de motivos, como pequenos movimentos no fundo da cena, ou, no nosso caso, principalmente por ruídos causados pelos componentes físicos utilizados para captura das imagens, como os cabos de transmissão das imagens analógicas até o computador.

Para muitas aplicações de visão computacional este tipo de detecção falsa pode não ser eliminado facilmente por filtros de ruído ou morfologia matemática, uma vez que essas operações podem afetar na detecção de pequenos objetos de interesse (ELGAMMAL AHMED; HARWOOD, 2000). No entanto, como na aplicação da nossa proposta não temos pequenos objetos que devem ser detectados, não existe a possibilidade de variações em pequenas áreas serem objetos de interesse, então essas técnicas se mostram bastante úteis.

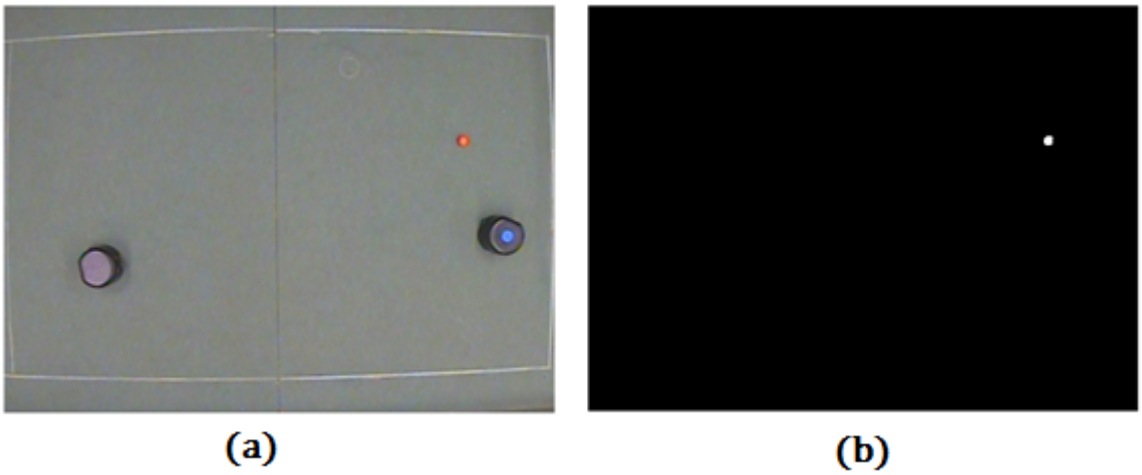


Figura 9 – Resultado da segmentação por cor.

É demonstrado um quadro de entrada com dois robôs e a bola (a), e o resultado da segmentação de cores alaranjadas (b).Utiliza-se valores que limitam à tonalidades alaranjadas, pois os pixels da bola se encontram nessa faixa de cores.

Com o intuito de corrigir ou diminuir possíveis ruídos resultantes do processo de segmentação, utilizamos uma abordagem de processamento de imagens baseada em filtros morfológicos, pois além de ser uma técnica simples, ela é capaz de produzir bons resultados. Essa abordagem resulta em eliminação de pequenas regiões isoladas e no preenchimento de pequenos buracos. A seguinte seção apresenta de forma resumida a morfologia matemática.

4.3.4.1 Morfologia Matemática

A morfologia matemática é baseada na teoria de conjunto. Foi introduzida em 1964 por MATHERON (1967) *apud* BLOCH Isabelle; SA (2002). A morfologia digital, no entanto, é uma área mais recente, uma vez que seu uso foi possível apenas com o surgimento dos computadores digitais. Ela pode ser utilizada em diversas áreas, como realce de cores, filtragem, segmentação, entre outras. A ideia principal da morfologia matemática é realizar operações em conjuntos de pixels de uma imagem. Segundo FACON (1996) *apud* WELFER (2011), sua estrutura básica é o elemento estruturante. O elemento estruturante é um conjunto de forma e estrutura definida, que é comparado com o conjunto de uma imagem que está sendo processada com o objetivo de extrair informações relativas à geometria e à topologia dessa imagem.

As operações básicas da morfologia digital são a erosão e a dilatação. Segundo WELFER (2011), a dilatação de uma imagem f por um elemento estruturante B, em um dado ponto x, é o valor máximo da imagem cuja vizinhança foi definida por B e está

cercada em x, e que pode ser definido como

$$(\delta^{(B)}(f))(x) = max_{b \epsilon B} f(x + b).$$

E a erosão em uma imagem f por um elemento B, em um dado ponto x, como o valor mínimo da imagem cuja vizinhança é compreendida por B, entrada em x, e que pode ser formulada como

$$(\varepsilon^{(B)}(f))(x) = min_{b \epsilon B} f(x + b).$$

Em nosso trabalho, o primeiro filtro aplicado é o filtro de erosão. A Erosão, em geral, faz com que os objetos diminuam de tamanho. Na erosão, se um ponto faz parte do objeto e tem um vizinho que faz parte do fundo, então ele também faz parte do fundo. O elemento estruturante utilizado foi um elemento de vizinhança 4, também chamado de N4, como pode ser observado na Figura 10.

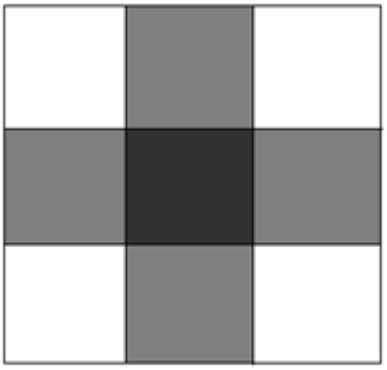


Figura 10 – Ilustração do elemento estruturante de vizinhança 4.

A Dilatação, por sua vez, faz com que os objetos se dilatem, ou aumentem de tamanho. Na dilatação é seguido o seguinte critério: se um ponto faz parte do fundo e tem um vizinho que faz parte de um objeto, então ele também faz parte do objeto. Ambos os filtros atuam nas bordas internas e externas dos objetos. A Figura 11, abaixo, apresenta o resultado da aplicação destes dois filtros na imagem binária resultante da segmentação de um robô no campo de jogo.

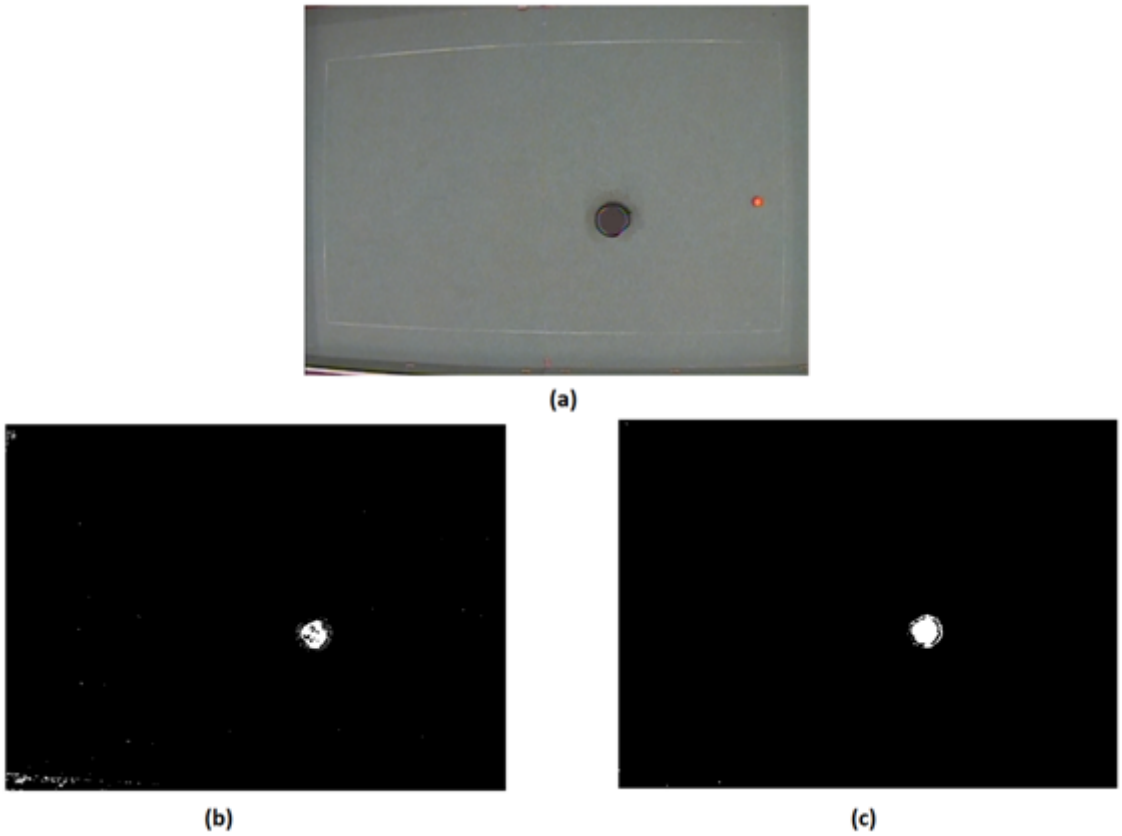


Figura 11 – Resultado da aplicação dos filtros morfológicos de erosão e dilatação.

Figura (a): frame de entrada. Figura (b): resultado da subtração de background. Figura (c): resultado após a aplicação dos filtros.

4.3.5 Cálculo da Posição

Após a segmentação dos objetos de interesse, o próximo passo é identificar as coordenadas dos mesmos. Para calcular as coordenadas de um determinado objeto de interesse, fazemos o uso do conceito de Momentos. Um momento é uma característica do contorno calculado. A técnica de momentos, também chamados momentos estatísticos, é um dos vários métodos utilizados para extração de características de uma imagem. Estes momentos e as funções derivadas deles se caracterizam por valores numéricos calculados a partir da imagem previamente segmentada e que descrevem a distribuição espacial dos pontos contidos na imagem ou em uma região. (CHACON G. GASTARDELLI, 2011)

Os momentos de imagem mais utilizados são os momentos regulares definidos a partir da seguinte fórmula:

$$m_{p,q} = \sum_{k=1}^{nx} \sum_{k=1}^{ny} x^p y^q f(x,y)$$

Nessa fórmula $m_{p,q}$ é momento de ordem $(p+q)$ da função de intensidade $f(x,y)$ onde nx e ny representam respectivamente a largura e a altura do objeto segmentado. Uma imagem binária terá valores da função $f(x,y)$ iguais a 0 ou 1. (CHACON G. GASTARDELLI, 2011)

O somatório é realizado sobre todos os pixels do limite de contorno (denotado por n na equação). Em seguida, segue-se imediatamente que se p e q são ambos iguais a 0, então o momento $m00$ é o comprimento em pixels do contorno. Como estamos olhando um contorno e não um polígono cheio, o comprimento e a área são a mesma em um espaço de pixels discretos (BRADSKI GARY; KAEHLER, 2008) Os momentos $m10$ e $m01$ representam as projeções nos eixos x e y respectivamente.

A partir dos momentos regulares podemos definir algumas medidas importantes sobre os objetos de interesse, e que são úteis na identificação de diferentes formas, por exemplo, os momentos regulares de ordem 0 e 1 são usados para o cálculo do baricentro ou centro de massa do objeto, através das seguintes fórmulas:

$$x_c = \frac{m_{10}}{m_{00}}$$

$$y_c = \frac{m_{01}}{m_{00}}$$

Na Figura 12, demonstra-se a implementação da função que realiza o cálculo da posição, ou centro de massa de um determinado objeto dentro da imagem. As coordenadas de um determinado objeto são copiadas para um vetor de duas posições, nas quais são gravados os valores de x ($vetor[0]$) e y ($vetor[1]$), considerando um plano cartesiano.

Após obtidas as coordenadas dos centros dos objetos, para fins de observação do rastreamento, contornamos os objetos identificados e imprimimos na tela suas coordenadas dentro do quadro. A implementação da função que realiza essa tarefa no nosso trabalho pode ser vista na Figura 13.

```
218 void getPosition(IplImage * thresh, int * vet){ // recebe uma imagem threshold (binária)
219     //e calcula as posicoes dos objetos
220     CvMoments *moments = (CvMoments*)malloc(sizeof(CvMoments));
221     cvMoments(thresh, moments, 1);
222 
223     // valor atual dos momentos
224     double moment10 = cvGetSpatialMoment(moments, 1, 0);
225     double moment01 = cvGetSpatialMoment(moments, 0, 1);
226     double area = cvGetCentralMoment(moments, 0, 0);
227 
228 
229     static int posX = 0;
230     static int posY = 0;
231 
232     //int lastX = posX; // armazenamos a posição atual em uma variável.
233     //int lastY = posY;
234 
235     posX = moment10/area;
236     posY = moment01/area;
237 
238     vet[0]= posX;
239     vet[1]=posY;
240     //printf("position (%d,%d)\n", posX, posY);
241     delete moments;
242 }
```

Figura 12 – Função que realiza o cálculo da posição de um objeto (em coordenadas (x,y)) e grava em um vetor.

```
143 void drawObject(int x, int y,Mat &frame, bool a){
144     // Desenha o circulo e que contorna os objetos trackeados
145     if(a==true){
146         circle(frame,Point(x,y),20,Scalar(0,255,255),2);
147         //escrever linhas com 25 pixels para cada lado do ponto y
148         if(y-25>0)
149             line(frame,Point(x,y),Point(x,y-25),Scalar(0,255,0),2);
150         else line(frame,Point(x,y),Point(x,0),Scalar(0,255,0),2);
151 
152         if(y+25<FRAME_HEIGHT)
153             line(frame,Point(x,y),Point(x,y+25),Scalar(0,255,0),2);
154         else line(frame,Point(x,y),Point(x,FRAME_HEIGHT),Scalar(0,255,0),2);
155         //escrever linhas com 25 pixels para cima e para baixo do ponto x
156         if(x-25>0)
157             line(frame,Point(x,y),Point(x-25,y),Scalar(0,255,0),2);
158         else line(frame,Point(x,y),Point(0,y),Scalar(0,255,0),2);
159 
160         if(x+25<FRAME_WIDTH)
161             line(frame,Point(x,y),Point(x+25,y),Scalar(0,255,0),2);
162         else line(frame,Point(x,y),Point(FRAME_WIDTH,y),Scalar(0,255,0),2);
163         // escrevemos as coordenadas do objeto ao seu lado
164         putText(frame,intToString(x)+","+intToString(y),Point(x,y+30),1,1,Scalar(0,255,0),2);
165 
166     }
167     else{
168 
169         circle(frame,Point(x,y),9,Scalar(0,0,255),2);
170         putText(frame,intToString(x)+","+intToString(y),Point(x,y+30),1,1,Scalar(0,0,0),2);
171     }
172 }
```

Figura 13 – Função que desenha o contorno nos objetos para observação do rastreamento.

Recebe um booleano que define se a imagem recebida é o resultado da segmentação do robô ou da bola.

4.3.6 Cálculo da Velocidade

Após determinarmos as posições dos objetos de interesse em coordenadas (x, y) determinamos então a velocidade do objeto.Para o cálculo da velocidade, precisamos, obviamente, determinar a distância percorrida por esse objeto em um período de tempo. Primeiramente definimos a distância, em pixels, percorrida pelo objeto. Como possuímos a informação do ponto em que o objeto se encontra em cada frame, basta calcularmos a distância entre os dois pontos de dois frames de tempos diferentes. Para esse cálculo utilizamos um conceito da geometria analítica, que diz que a distância entre dois pontos $A(x_A, y_A)$ e $B(x_B, y_B)$ no plano é calculada como segue:

$$(d_{A,B})^2 = (x_b - x_a)^2 + (y_b - y_a)^2$$

$$d_{A,B} = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$$

O tempo percorrido entre um determinado *frame* F_{t1} e outro *frame* F_{t2}, de tempos diferentes, é calculado com base no conhecimento da taxa de frames capturados em um segundo (FPS) pelo método de captura. Como utilizamos um dispositivo de captura que opera a uma taxa igual a 30 FPS, podemos utilizar uma simples regra de 3 para descobrirmos o tempo percorrido entre qualquer número de frames.

Sabendo a distância percorrida e o periodo de tempo, então calculamos a velocidade dos objetos com base na fórmula

$$v = \Delta s / \Delta t$$

O onde Δs representa a distância percorrida, e Δt representa o intervalo de tempo.

Como o deslocamento da bola é realizado de uma forma muito mais rápida do que a dos robôs, utilizamos unidades diferentes para o cálculo da velocidade de cada um. Isso se deve ao fato de que a velocidade dos robôs se calculada entre dois quadros proximos num periodo de tempo muitas vezes será retornada como zero. Para calcular a velocidade da bola, definimos que o tempo utilizado é igual a 0,1 segundo, ou seja, a velocidade será calculada com base na distância percorrida pela bola entre 3 frames. Já para descobrirmos a velocidade dos robôs, utilizamos como tempo o período de 0,5 segundo, o que define que o cálculo será realizado a cada 15 frames. Ao final desse cálculo, teremos a velocidade dos objetos em pixels/segundo. Então transformamos a distância para metros, com base

numa transformação que segue a escala 216p x 1m, determinada com base na comparação entre tamanho da imagem e do campo. Assim sendo, cada pixel da imagem representa 0,00462963 metros.

5 Resultados Finais

Ao decorrer do capítulo anterior, a fim de tornar o entendimento das técnicas descritas mais fácil e exemplificar as características de cada técnica, foram apresentados alguns dos resultados intermediários. Pudemos observar passo a passo alguns dos resultados das atividades intermediárias necessárias para que se chegasse ao resultado final.

Neste capítulo são demonstrados os resultados finais do desenvolvimento do trabalho, aqueles obtidos após todas as tarefas descritas no capítulo anterior. Nas seguintes seções serão apresentados os resultados finais do trabalho. Além disso, são descritas as vantagens e desvantagens das principais técnicas e algoritmos utilizados para obter os resultados finais. Ao final do capítulo, são descritas as ferramentas utilizadas para a elaboração dos algoritmos.

5.1 Destacamento dos Objetos Identificados

Após a segmentação dos objetos e o cálculo da posição, realizamos o destacamento dos objetos identificados, a fim de podermos observar os dados obtidos durante as etapas descritas no capítulo anterior de uma forma mais fácil.

Na Figura 14 é demonstrado o resultado do desenho de contorno dos objetos com suas respectivas posições.

Como podemos observar na Figura 14, o resultado da segmentação é satisfatório, pois apresenta claramente os objetos de *foreground* , tanto na segmentação por subtração de *background* quanto na segmentação por cor. O desenho do círculo sobre os objetos identificados permite observar que foram alcançados bons resultados dentro do escopo da nossa proposta.

5.2 Velocidade

Como descrito na subseção 4.3.6 , após determinarmos as posições dos objetos de interesse em coordenadas (x, y) determinamos então a velocidade do objeto. O resultado é impresso pelo comando de saída padrão do c++, como podemos observar em Figura 15, que demonstram o resultado do cálculo da velocidade da bola.

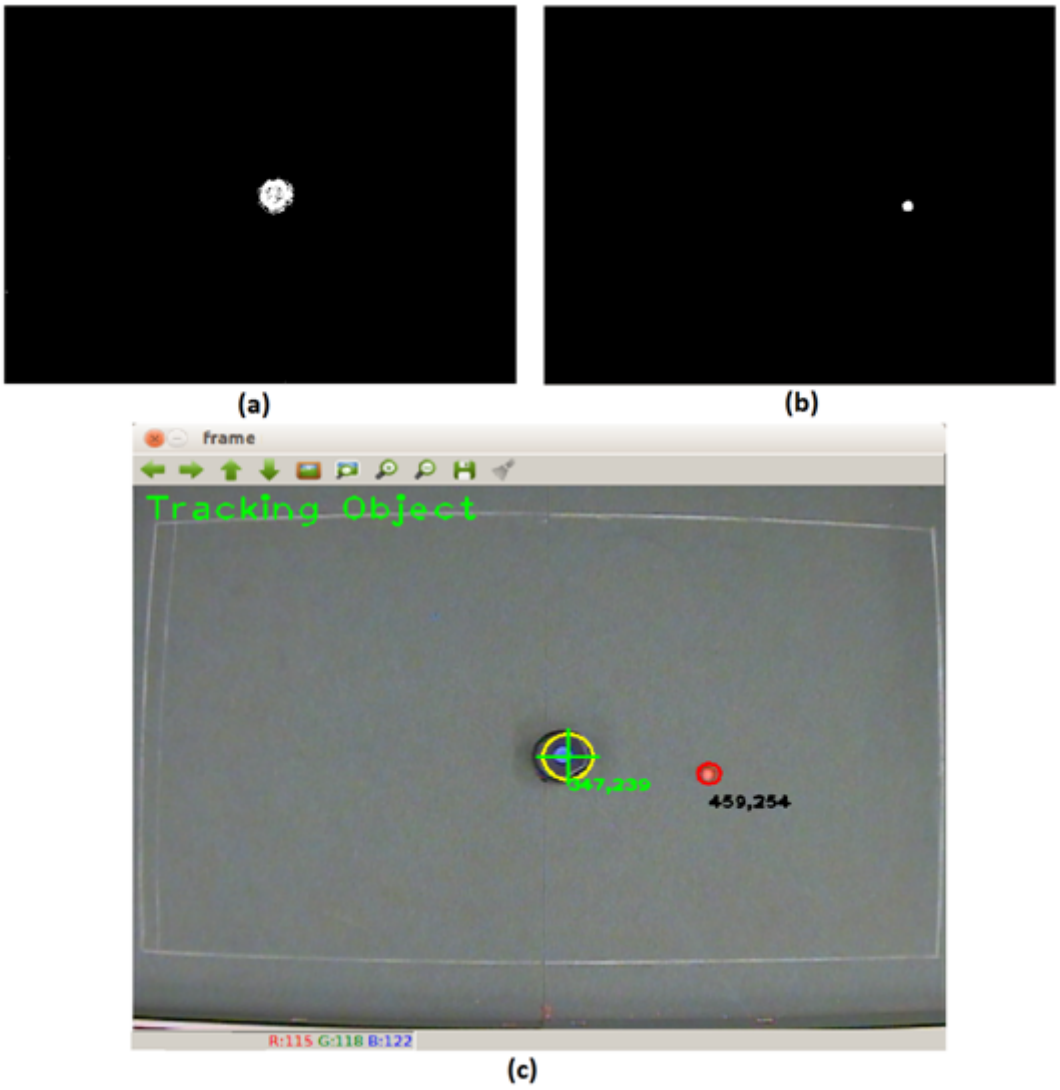


Figura 14 – Resultado do desenho do contorno e das posições nos objetos identificados.

A figura (a) representa a segmentação do robô por subtração de background após a aplicação dos filtros morfológicos para a eliminação de ruído, enquanto a figura (b) representa a imagem resultante da segmentação da bola através do método de segmentação por cores, demonstrado na subseção 4.3.3

5.3 Vantagens e Desvantagens das Técnicas Utilizadas

Durante o estudo das técnicas encontradas na literatura e da implementação de alguns métodos propostos a fim de chegarmos à nossa solução final, pudemos identificar problemas em algumas dessas técnicas quando aplicadas ao nosso trabalho, assim como também observamos vantagens de alguns dos diferentes tipos de abordagens.

Métodos para a criação do modelo de fundo que utilizam a média dos valores de pixels de n quadros são de simples implementação e proporcionam resultados razoáveis

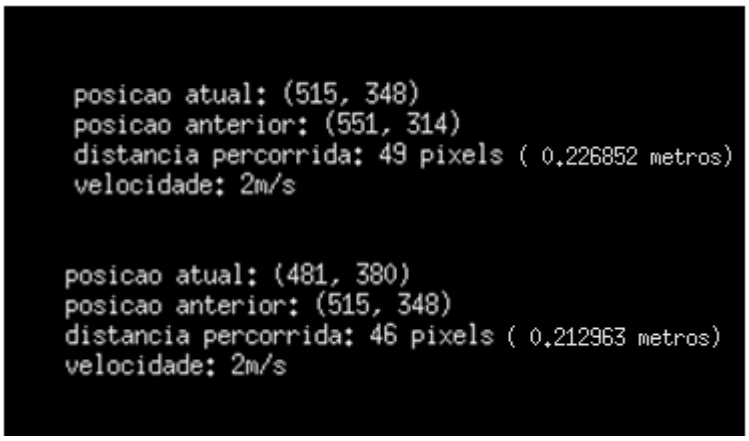


Figura 15 – Coordenadas atuais (tempo T) e anteriores (tempo T-3), distância percorrida em pixels e velocidade em metros/segundo são jogadas na saída padrão para fins de observação.

quando aplicados à ambientes fechados e com pouca variação de iluminação, como ocorre no nosso caso. No entanto, a técnica que utilizamos, que cria um modelo de *background* com misturas gaussianas e o atualiza com o passar do tempo garantiu um resultado muito mais satisfatório ao nosso trabalho. A técnica de subtração de *background* que utilizamos garante que o resultado da segmentação apresente pouco ruído, entretanto, podemos observar que a técnica de segmentação por cor que utilizamos apresenta um resultado com maior exatidão e muito menos ruído, apesar de, em muitos casos, não ser aplicável aos mesmos tipos de problemas.

Um fato que podemos observar no desempenho da nossa aplicação, é que para que a função de cálculo das coordenadas do centro de massa retorne resultados aceitáveis, deve ser garantido que o resultado da segmentação apresente objetos conectados bem definidos e com poucas falhas, caso contrário, a função poderá retornar falsar posições e coordenadas de objetos inexistentes.

5.4 Ferramentas Utilizadas

Para resolver os diversos tipos de desafios de visão computacional que encontramos atualmente, incentivar e disseminar o conhecimento da visão computacional, era necessário a criação de uma biblioteca de funções de programação de código livre. Com essa finalidade, em 1999 a Intel fez o lançamento de um biblioteca de código aberto chamada *OpenCV*.

Como os algoritmos de processamento de imagens apresentados neste trabalho fazem uso do *OpenCV*, na seguinte seção será feita uma abordagem sucinta sobre esta biblioteca, apresentando os seus objetivos e sua capacidade de prover soluções para pro-

blemas relacionados à visão computacional.

5.4.1 OpenCV

OpenCV, que significa *Open Source Computer Vision Library*, é uma biblioteca de código aberto, que é amplamente utilizado em aplicações de processamento de imagem. Como os algoritmos de processamento de imagens são projetados para aplicações de tempo real, sua implementação é original foi feita em C, atualmente também em C++, especificamente para o aumento da eficiência computacional. (CHACZKO ZENON; YEOH, 2010)

A biblioteca *OpenCV*, além de fornecer tarefas elementares de processamento de imagens, como as operações lógicas e aritméticas, também é composta de operações complexas como detecção e rastreamento de objetos. Um dos objetivos dessa biblioteca é fornecer uma infra-estrutura de visão computacional simples e de fácil utilização, que auxilie os usuários a construir aplicações de visão computacional sofisticadas, de forma bastante simples e rápida (BRADSKI GARY; KAEHLER, 2008).

Exemplos de aplicação da biblioteca *OpenCV* incluem Interação Humano-Computador (IHC), identificação de objetos, segmentação e reconhecimento, reconhecimento de face, reconhecimento de gestos, Compreensão Movimento, entre diversas outras aplicações de visão computacional (CULJAK, 2012). Como a visão computacional e aprendizado de máquinas muitas vezes andam de mãos dadas, o *OpenCV* também possui muitas funções para uso em aprendizado de máquinas.

A última grande mudança ocorreu em 2009 (OpenCV 2), quando foi necessário a inclusão das principais alterações para adaptação à linguagem C++. Segundo a homepage da biblioteca, hoje em dia, a biblioteca possui mais de 2500 algoritmos otimizados. É amplamente utilizado em todo o mundo, com 2,5 milhões de downloads mais de 40 mil usuários. Isso se deve ao fato de a licença de código permitir que o usuário possa construir um produto comercial usando todo ou parte do *OpenCV*. Dessa forma, o *OpenCV* se tornou bastante usado em ambientes acadêmicos e em aplicações comerciais, pois está sob licença BSD (CULJAK, 2012).

Em parte por causa dos termos de licenciamento liberais, há uma grande comunidade de usuários que estão incluídos em grandes empresas (IBM, Microsoft, Intel, Sony, Siemens, e Google, para citar apenas alguns) e centros de pesquisa (tais como Stanford, MIT , CMU, Cambridge, e INRIA) (BRADSKI GARY; KAEHLER, 2008). Dessa forma, um grande número de programadores têm contribuído para os desenvolvimentos mais recentes da biblioteca. A versão mais recente da biblioteca pode ser encontrada no site oficial do *OpenCV* (INTEL,).

5.4.2 QtCreator

Para desenvolvimento, testes e execuções dos algoritmos mostrados no trabalho, foi utilizada a ferramenta QtCreator (QT-PROJECT,). Qt Creator é uma IDE voltada para programação nas linguagens C++ e Java. Ela é amplamente utilizada para o desenvolvimento de aplicações de software com uma interface gráfica e também para o desenvolvimento de aplicações não gráficas como manipulação de arquivos, acesso a banco de dados, análise de XML, gerenciamento de threads e suporte de rede. (QT-PROJECT,). Uma de suas principais vantagens é a portabilidade, ou seja, ele pode ser usado em diversos sistemas operacionais. Outra vantagem, que motivou a sua utilização neste trabalho, é a fácil integração com as bibliotecas OpenCV. Outra vantagem dessa ferramenta, é o fato de contar com a opção *"Debug"*, que permite acompanhar passo a passo a execução do programa.

O *download* das ultimas versões dessa IDE pode ser feito através da homepage do projeto (QT-PROJECT,).

6 Conclusão

O presente trabalho, além de procurar introduzir e contextualizar a robótica e a visão computacional, tem como principal objetivo a criação de um sistema de visão computacional para futebol de robôs.

Com a leitura da bibliografia abordada por ele, nota-se a importância de estudar métodos para os sistemas de visão computacional, além de aprender os diversos tipos de aplicações para estes sistemas.

O foco principal do trabalho foi desenvolver um algoritmo capaz de identificar e rastrear os objetos de interesse no contexto de uma partida de futebol de robôs para fornecer coordenadas e informações destes objetos para posteriores tarefas em um sistema de futebol de robôs.

Este trabalho aborda a definição da metodologia adotada para a solução final, assim como trabalhos relacionados que serviram como base para conhecimento dos métodos e algoritmos que foram utilizados para a implementação do sistema. Estudando os métodos propostos nos trabalhos relacionados, fica claro que a segmentação de objetos é uma tarefa complexa, que requer a aplicação de bons métodos para a modelagem de um modelo de fundo da cena, para a subtração entre este frame e o frame atual assim como para a atualização do modelo de referência. Além disso, essa tarefa possui algumas peculiaridades que em muitos casos precisam ser tratadas, como ruídos resultantes da segmentação devido aos vários fatores que interferem nessa fase, assim como tratamento de sombras que podem prejudicar muito nos resultados obtidos nesses sistemas.

Embora o presente trabalho obtenha resultados razoáveis, ele poderá ser bastante aprimorado no futuro. Ele tem como uma das finalidades procurar incentivar e quem sabe auxiliar em trabalhos futuros mais robustos e complexos.

Outra fato que podemos concluir a partir da elaboração deste trabalho, é que torna-se necessário o prosseguimento no estudo das funções da biblioteca OpenCV a fim de adquirir um conhecimento maior sobre a funcionalidade do grande número de funções e da capacidade dessa biblioteca. Portanto, a fim de construir um sistema mais completo a partir deste, torna-se imprescidível o estudo mais a fundo das funções OpenCV. Algumas técnicas mais eficientes e robustas, serão uma opção para a continuidade do trabalho no futuro. Uma das principais técnicas que podem ser criadas/aprimoradas são previsão da trajetória de cada robô.

Referências

BEKEY, G. Autonomous robots: From biological inspiration to implementation and control. *The MIT Press*, 2005. Citado na página 25.

BIANCHI R. A. C; COSTA, A. H. R. O sistema de visão computacional do time futepoli de futebol de robôs. *FProceedings of XIII Congresso Brasileiro de Automática*, 2000. Citado na página 26.

BLOCH ISABELLE; SA, A. W. R. s. u. F. Mathematical morphology. *Congress On Neuro-fuzzy Technologies*, 2002. Citado na página 46.

BORSATO F.H.; FLORES, F. A real time method to object detection and tracking applied to robot-soccer. *Cybernetics and Intelligent Systems, 2004 IEEE Conference*, 2004. Citado na página 30.

BRADSKI GARY; KAEHLER, A. Learning opencv. *USA: O'reilly*, 2008. Citado 2 vezes nas páginas 49 e 56.

CHACON G. GASTARDELLI, E. M. F. O. G. A. M. A. P. Aplicação da técnica de momentos invariantes no reconhecimento de padrões em imagens digitais. *Centro Brasileiro de Pesquisas Físicas*, 2011. Citado 3 vezes nas páginas 40, 48 e 49.

CHACZKO ZENON; YEOH, L. A. M. V. Apreliminary investigation on computer vision for telemedicine systems using opencv. *IEEE, Conference On Open Systems*, 2010. Citado na página 56.

CHEUNG, G. K. A real time system for robust 3d voxel reconstruction of human motions. *Computer Vision And Pattern Recognition, 2000. Proceedings. IEEE Conference*, 2000. Citado 2 vezes nas páginas 34 e 35.

COSTA A.H.L; PEGORARO, R. Construindo robôs autônomos para partidas de futebol: O time guaraná. controle e autoamação. 2000. Citado na página 31.

CUCCHIARA R.; GRANA, C. P. M. P. A. Detecting moving objects, ghosts, and shadows in video streams. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, 2003. Citado na página 29.

CULJAK, I. A brief introduction to opencv. *MIPRO, IEEE*, 2012. Citado na página 56.

ELGAMMAL AHMED; HARWOOD, D. D. L. Non-parametric model for background subtraction. *Computer Vision Laboratory University Of Maryland, College Park*, 2000. Citado 2 vezes nas páginas 33 e 45.

FACON, J. *Morfologia Matemática: Teorias e. Exemplos*. [S.l.]: Editora Universitária Champagnat da. Pontifícia Universidade Católica do Paraná, 1996. Citado na página 46.

FRIEDMAN, N.; RUSSELL, S. Image segmentation in video sequences: A probabilistic approach. *The Thirteenth Conference on Uncertainty in Artificial Intelligence*, 1997. Citado na página 35.

GOMES MARCELO M.; KAWAMURA, J. L. F. Aplicação do sistema de futebol de robôs como ferramenta didática. *Escola de Engenharia Mauá do Centro Universitário do Instituto Mauá de Tecnologia*, 2001. Citado na página 26.

GROOVER M. P.; WEISS, M. N. R. N. O. N. G. Robótica. tecnologia e programação. *McGraw-Hill*, 1989. Citado na página 25.

INTEL. Opencv. Disponível em: <http://opencv.org/>. Acesso em: 02.07.2013. Citado na página 56.

JACQUES J.C.S.; JUNG, C. M. S. Background subtraction and shadow detection in grayscale video sequences. *Computer Graphics and Image Processing*, 2005. Citado na página 33.

JAYASHREE, R. Rgb to hsi color space conversion via mact algorithm. *International conference on Communication and Signal Processing*, 2013. Citado na página 44.

JUNG, C. R. et al. Detection of unusual motion using computer vision. In: IEEE. *Computer Graphics and Image Processing, 2006. SIBGRAPI'06. 19th Brazilian Symposium on*. [S.l.], 2006. p. 349–356. Citado na página 30.

JUNG C. R. ; MUSSE, S. R. Background subtraction and shadow detection in grayscale video sequences. *Proceedings of the XVIII Brazilian Symposium on Computer Graphics and Image Processing, IEEE Computer Society*, 2005. Citado na página 29.

KAEWTRAKULPONG P.; BOWDEN, R. An improved adaptive background mixture model for realtime tracking with shadow detection. *2nd European Workshop On Advanced Video Based Surveillance Systems*, 2001. Citado 3 vezes nas páginas 36, 42 e 43.

KIM SUNG HO; CHOI, J. S. K. J. K. K. B. K. A cooperative micro robot system playing soccer: Design and implementation. *Robotics and Autonomous systems*, 1997. Citado na página 22.

KLINKER, G.; SHAFER, S.; KANADE, T. A physical approach to color image understanding. *International Journal of Computer Vision*, 1990. Citado na página 44.

KOLLER. Towards robust automatic traffic scene analysis in realtime. *Proceedings of the 33rd IEEE Conference on Decision and Control*, 1994. Citado na página 35.

KRAVTCHENKO, V. Tracking color objects in real time. *Master's thesis, University of British Columbia*, 1999. Citado na página 43.

MATHERON, G. *Elements pour une theorie des milieuxporeux*. [S.l.]: Masson, Paris, 1967. Citado na página 46.

MCKENNA S.; JABRI, S. D. C. Z. R. A. W. H. Tracking groups of people. *Computer Vision and Image Unders tanding*, 2000. Citado na página 30.

PEREZ, A. L. F. Robótica inteligente: Tecnologias e aplicações. 2005. Disponível em: <http://www.reitoria.ufsc.br/~anderson.perez/palestras/RITA.PDF>. Acesso em: 12.05.2013. Citado na página 25.

QT-PROJECT. Qtcreator. Disponível em: <http://qt-project.org/>. Acesso em: 03.07.2013. Citado na página 57.

RUIZ M.A.; URESTI, J. Team agent behavior architecture in robot soccer. *Robotic Symposium, 2008. IEEE Latin American*, 2008. Citado na página 22.

SATTAR, J.; DUDEK, G. On the performance of color tracking algorithms for underwater robots under varying lighting and visibility. *IEEE Int. Conf. Robot. Autom*, 2006. Citado na página 44.

SOBREIRA RODOLFO MARENGO; SILVA, F. A. D. R. R. L. Futebol de robôs, uma aplicação de robótica. *Faculdade de Informática de Presidente Prudente*, 2004. Citado na página 22.

SOLDERA, J. *Detecção de Movimentos suspeitos em sequencias de vídeos*. Dissertação (Mestrado) — Unisinos, São Leopoldo, 2007. Citado na página 30.

STAUFFER, G. W. E. L. Adaptive background mixture models for real-time tracking. in proceedings. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999. Citado na página 42.

TRUYENQUE, M. A. Q. *Uma Aplicação de Visão Computacional que Utiliza Gestos da Mão para Interagir com o Computador*. Dissertação (Mestrado) — PUC-RIO, Rio de Janeiro, 2005. Citado 2 vezes nas páginas 33 e 36.

WELFER, D. *Métodos Computacionais Para Identificar Automaticamente Estruturas da Retina e Quantificar a Severidades do Edema Macular Diabético em Imagens de Fundo de Olho*. Dissertação (Mestrado) — UFRGS, Porto Alegre, 2011. Citado na página 46.

ZICKLER S.; LAUE, T. B. O. W. M. V. M. Ssl-vision: Theshared vision system for the robocup small size league. *RoboCup 2009: RobotSoccer World Cup XIII*, 2009. Citado 2 vezes nas páginas 31 e 32.

Apêndices

APÊNDICE A – Funções Implementadas

A.1 Função main()

```
int main()
{
   //subtracaoMetodo();
   int vet[2];
   //cv::VideoCapture cap(0);
   // CvCapture* cap =cvCaptureFromAVI("movie_01_01.avi");
   CvCapture* capture = cvCaptureFromCAM( 1 );

   cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH, 648.0 );
   cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT, 488.0 );
   cvSetCaptureProperty(capture, CV_CAP_PROP_BRIGHTNESS,0.6);
   cvSetCaptureProperty(capture, CV_CAP_PROP_CONTRAST,0.2);

   float frameRate = cvGetCaptureProperty(capture,CV_CAP_PROP_FPS);

   //cout<<"FRAMES POR SEGUNDO: "<<frameRate <<endl;

    Mat frame, frame2;

    //createTrackbars();
    IplImage* imagem_binaria_cor;
    IplImage *frame_img = NULL;
    IplImage *frame_imgBG;
    IplImage *img_fore;
    IplImage *imgAcumulada;

    cv::SimpleBlobDetector::Params params;
    params.minDistBetweenBlobs = 10.0;  // minimum 10 pixels between blobs
    params.filterByArea = true;         // filter my blobs by area of blob
    params.minArea = 20.0;              // min 20 pixels squared
    params.maxArea = 500.0;             // max 500 pixels squared
```

```
 SimpleBlobDetector myBlobDetector(params);
 std::vector<cv::KeyPoint> myBlobs;



 cv::Mat back;
 cv::Mat fore;

 cv::Mat blobImg;



 cv::BackgroundSubtractorMOG2 bg;


 bg.nmixtures = 3; //O nmero mximo permitido de componentes da mistura. O nmero real
 bg.bShadowDetection = true;

 std::vector<std::vector<cv::Point> > contours; // Vetor de vetores de pontos

 //cv::namedWindow("Frame");
 //cv::namedWindow("Background");
 int x,y,xb,yb;

int count =1;
float veloc_robo,veloc_bola;

for(;;)
{

    frame = cvQueryFrame(capture);
    frame.copyTo(frame2);

   Mat medianBackground (frame.rows,frame.cols,frame.type());

   // medianBackground = simpleModelBackground(frame,medianBackground);


        frame_img = new IplImage(frame);
```

```
//simpleModelBackground(frame_img,imgAcumulada);


bg.operator ()(frame,fore); // frame = modelo de fundo, fore = mscara binria
bg.getBackgroundImage(back); //retorna imagem de fundo

frame_imgBG = new IplImage(back);
morphOps(fore);
cv::findContours(fore,contours,CV_RETR_EXTERNAL,CV_CHAIN_APPROX_NONE);
//simpleBackgroundSubtraction(back,frame);
img_fore = new IplImage(fore);
//imshow("fore",fore);
//getPosition(img_fore, vet);
//x=vet[0];
//y=vet[1];
//drawObject(x,y,frame);
//imshow("fore mira",frame);

myBlobDetector.detect(frame, myBlobs);
cv::drawKeypoints(fore, myBlobs, blobImg);
cv::imshow("Blobs", blobImg);
trackFilteredObject(x,y,fore,frame);

for (int i=0; i<myBlobs.size(); i++){
    float X=myBlobs[i].pt.x;
    float Y=myBlobs[i].pt.y;
}


//fore = imagem binaria, retrieve only external contours , Detectar todos os
//cv::drawContours(frame,contours,-1,cv::Scalar(0,0,255),2);


//cv::morphologyEx(frame,frame,MORPH_OPEN,cv::Mat());



cvSmooth(frame_imgBG, frame_imgBG, CV_GAUSSIAN,3,3);
```

```
        cvSmooth(frame_img, frame_img, CV_GAUSSIAN,3,3);

        imagem_binaria_cor = TrackerCor(frame_img,frame,xb,yb);
//------cvSmooth(imagem_binaria_cor,imagem_binaria_cor,CV_GAUSSIAN,3,3);
        //Mat mat_binary_color (imagem_binaria_cor);
        //morphOps(mat_binary_color);
        //imshow("binaria cor",mat_binary_color);
        trackFilteredObject(x,y,fore,frame);

        //imshow("frame mira",frame);

        imshow("frame",frame);

        cvShowImage("threshold cor",imagem_binaria_cor);

        if(count%3==0){
            veloc_bola= getVelocidadeBola(xb,yb);
            cout<<"velocidade: "<< veloc_bola<<"m/s \n\n"<<endl;
        }else
            if(count%25==0){
                veloc_robo= getVelocidadeRobo(x,y);
            }
        count++;


        if(cv::waitKey(30) >= 0) break;
    }

    return 0;
}
```

A.2 Função trackFilteredObject

```
void trackFilteredObject(int &x, int &y, Mat threshold, Mat &cameraFeed){
    int id;
    Mat temp;
    threshold.copyTo(temp);
    //vetores para a saida do findCountors
    vector< vector<Point> > contours;
```

```
    vector<Vec4i> hierarchy;

    findContours(temp,contours,hierarchy,CV_RETR_CCOMP,CV_CHAIN_APPROX_SIMPLE );

    double refArea = 0;
    bool objectFound = false;
    if (hierarchy.size() > 0) {
        int numObjects = hierarchy.size();
        //cout<<"objetos: "<< numObjects << endl;
        if(numObjects<MAX_NUM_OBJECTS){
            for (int index = 0; index >= 0; index = hierarchy[index][0]) {

                Moments moment = moments((cv::Mat)contours[index]);
                double area = moment.m00;

                if(area>MIN_OBJECT_AREA && area<MAX_OBJECT_AREA && area>refArea){
                    x = moment.m10/area;
                    y = moment.m01/area;
                    objectFound = true;

                    // Aqui some
                    putText(cameraFeed,"Tracking Object",Point(10,30),1,2,
                    Scalar(0,255,0),2);
                    drawObject(x,y,cameraFeed,true);

                }else objectFound = false;
            }
            if(objectFound ==true){
            //    putText(cameraFeed,"Tracking Object",Point(10,30),1,2,
            Scalar(0,255,0),2);
              //  drawObject(x,y,cameraFeed,true);
            }

        }else putText(cameraFeed,"Muito ruido",Point(10,30),2,1,Scalar(0,0,255),2);
    }
}
```

A.3 Demais Funções

```
void captureforAvi(){


   CvCapture* capture = cvCaptureFromCAM( 1 );
   cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH, 648.0 );
   cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT, 488.0 );
   cvSetCaptureProperty(capture, CV_CAP_PROP_BRIGHTNESS,0.6);
   cvSetCaptureProperty(capture, CV_CAP_PROP_CONTRAST,0.2);
  // capture = cvCreateFileCapture();


   IplImage *bgr_frame=cvQueryFrame(capture);//Init the video read
   double fps = cvGetCaptureProperty (capture,CV_CAP_PROP_FPS);


   CvSize size = cvSize((int)cvGetCaptureProperty( capture, CV_CAP_PROP_FRAME_WIDTH),(in
   CvVideoWriter *writer = cvCreateVideoWriter("/home/unipampa/área de Trabalho/saida.av
   IplImage* logpolar_frame = cvCreateImage(size,IPL_DEPTH_8U, 3 );
   while( (bgr_frame=cvQueryFrame(capture)) != NULL ) {
       cvLogPolar( bgr_frame, logpolar_frame, cvPoint2D32f(bgr_frame->width/2, bgr_frame
       cvWriteFrame( writer, logpolar_frame );
   }
   cvReleaseVideoWriter( &writer );
   cvReleaseImage( &logpolar_frame );
   cvReleaseCapture( &capture );

}



void subtracaoMetodo(){
   IplImage *imgAcumulada, *frame_img;
   cv::Mat frame;

   CvCapture* cap = cvCaptureFromCAM( 1 );
   cvSetCaptureProperty(cap, CV_CAP_PROP_FRAME_WIDTH, 648.0 );
   cvSetCaptureProperty(cap, CV_CAP_PROP_FRAME_HEIGHT, 488.0 );
```

```
   frame = cvQueryFrame(cap);
   frame_img = new IplImage(frame);



   int i;


   for(i=0; i<100;i++)
   {
       frame = cvQueryFrame(cap);
       CvSize sz = cvGetSize( frame_img );
       imgAcumulada = cvCreateImage(sz,IPL_DEPTH_32F,3);
       cvAcc(frame_img,imgAcumulada);
       if(cv::waitKey(30) >= 0) break;
   }
}
//--------------------------------------------------------------------------------
void on_trackbar( int, void* )
{//função chamada sempre que uma posicao no trackbar é alterada

}
//--------------------------------------------------------------------------------

void createTrackbars(){

   namedWindow(trackbarWindowName,0);

   char TrackbarName[50];

   sprintf( TrackbarName, "H_MIN", H_MIN);
   sprintf( TrackbarName, "H_MAX", H_MAX);
   sprintf( TrackbarName, "S_MIN", S_MIN);
   sprintf( TrackbarName, "S_MAX", S_MAX);
   sprintf( TrackbarName, "V_MIN", V_MIN);
   sprintf( TrackbarName, "V_MAX", V_MAX);


   //primeiro parâmetro: variável a ser alterada quando o trackbar é movido
```

```
   // segundo: valor máximo q a trackbar pode receber
   //terceiro: função chamada toda a vez que a trackbar se mover
   //                                  ---->    ---->     ---->

   createTrackbar( "H_MIN", trackbarWindowName, &H_MIN, H_MAX, on_trackbar );
   createTrackbar( "H_MAX", trackbarWindowName, &H_MAX, H_MAX, on_trackbar );
   createTrackbar( "S_MIN", trackbarWindowName, &S_MIN, S_MAX, on_trackbar );
   createTrackbar( "S_MAX", trackbarWindowName, &S_MAX, S_MAX, on_trackbar );
   createTrackbar( "V_MIN", trackbarWindowName, &V_MIN, V_MAX, on_trackbar );
   createTrackbar( "V_MAX", trackbarWindowName, &V_MAX, V_MAX, on_trackbar );


}
//---------------------------------------------------------------------------------------
string intToString(int number){ // função para transformar um inteiro em uma string


   std::stringstream ss;
   ss << number;
   return ss.str();
}
//---------------------------------------------------------------------------------------

void drawObject(int x, int y,Mat &frame, bool a){
   // Desenha o circulo e que contorna os objetos trackeados
   if(a==true){
       circle(frame,Point(x,y),20,Scalar(0,255,255),2);
       //escrever linhas com 25 pixels para cada lado do ponto y
       if(y-25>0)
           line(frame,Point(x,y),Point(x,y-25),Scalar(0,255,0),2);
       else line(frame,Point(x,y),Point(x,0),Scalar(0,255,0),2);

       if(y+25<FRAME_HEIGHT)
           line(frame,Point(x,y),Point(x,y+25),Scalar(0,255,0),2);
       else line(frame,Point(x,y),Point(x,FRAME_HEIGHT),Scalar(0,255,0),2);
       //escrever linhas com 25 pixels para cima e para baixo do ponto x
       if(x-25>0)
           line(frame,Point(x,y),Point(x-25,y),Scalar(0,255,0),2);
       else line(frame,Point(x,y),Point(0,y),Scalar(0,255,0),2);
```

```
        if(x+25<FRAME_WIDTH)
            line(frame,Point(x,y),Point(x+25,y),Scalar(0,255,0),2);
        else line(frame,Point(x,y),Point(FRAME_WIDTH,y),Scalar(0,255,0),2);
        // escrevemos as coordenadas do objeto ao seu lado
        putText(frame,intToString(x)+","+intToString(y),Point(x,y+30),1,1,
        Scalar(0,255,0),2);

    }
    else{

        circle(frame,Point(x,y),9,Scalar(0,0,255),2);
        putText(frame,intToString(x)+","+intToString(y),Point(x,y+30),1,1,
        Scalar(0,0,0),2);
    }
}
//---------------------------------------------------------------------------

void morphOps(Mat &thresh){ // função que aplica filtros morfológicos

    cv::erode(thresh,thresh, cv::Mat());

    cv::dilate(thresh,thresh,cv::Mat());



    //cv::imshow("frame",);
    //cv::imshow("abertura",abertura);
    //cv::imshow("fechamento",fechamento);


}
//---------------------------------------------------------------------------

//Esta função irá ter uma imagem, e retornar uma imagem binária (onde determinada
cor será branca e o resto será preto).

IplImage* GetThresholdedImage(IplImage* img)
{
```

```
   IplImage* imgHSV = cvCreateImage(cvGetSize(img), 8, 3);
   cvCvtColor(img, imgHSV, CV_BGR2HSV);
   //cvShowImage("Imagem HSV",imgHSV);
   IplImage* imgThreshed = cvCreateImage(cvGetSize(img), 8, 1);

   cvInRangeS(imgHSV, Scalar(0,110,200), Scalar(13,256,256), imgThreshed);
   // os dois 'cvScalar' representam o limite inferior e superior de valores que
   serãoo de cor alaranjada.
   cvReleaseImage(&imgHSV);
   //Mat matThreshed (imgThreshed);
   //morphOps(matThreshed);
   //imgThreshed = new IplImage (matThreshed);
   return imgThreshed;
}


//-------------------------------------------------------------------------------
//-------------------------------------------------------------------------------

void getPosition(IplImage * thresh, int * vet){ // recebe uma imagem threshold (binária)
   //e calcula as posicoes dos objetos
   CvMoments *moments = (CvMoments*)malloc(sizeof(CvMoments));
   cvMoments(thresh, moments, 1);

   // valor atual dos momentos
   double moment10 = cvGetSpatialMoment(moments, 1, 0);
   double moment01 = cvGetSpatialMoment(moments, 0, 1);
   double area = cvGetCentralMoment(moments, 0, 0);


   static int posX = 0;
   static int posY = 0;

   //int lastX = posX; // armazenamos a posição atual em uma variávelvel.
   //int lastY = posY;

   posX = moment10/area;
   posY = moment01/area;
```

```
   vet[0]= posX;
   vet[1]=posY;
   //printf("position (%d,%d)\n", posX, posY);
   delete moments;
}


//-------------------------------------------------------------------------
//-------------------------------------------------------------------------

int getVelocidadeBola(int x,int y){
   cout<<"posicao atual: ("<<x<<", "<<y<<")"<<endl;
   cout<<"posicao anterior: ("<<lastxB<< ", " << lastyB<< ")"<<endl;
   float velocidade = -1;
   if(lastxB != -1 && lastyB != -1){
       int distancia = sqrt(pow((x-lastxB),2)+pow((y-lastyB),2));
       cout<<"distancia percorrida: "<<distancia<< " pixels"<<endl;
       velocidade = distancia / 0.1;
       velocidade = velocidade/216; //transforma de pixel para metro;
   }
   lastxB=x;
   lastyB=y;
   //cout<<"lastX saindo :  "<<lastxB<<"  lastY saindo: "<<lastyB<<endl;

   return velocidade;
}

int getVelocidadeRobo(int x,int y){
   float velocidade =-1;
   if(lastxR != -1 && lastyR != -1){
       int distancia = sqrt(pow(2,(x-lastxR))+pow(2,(y-lastyR)));
       velocidade = distancia / 0.5;
       velocidade = velocidade/216; //transforma de pixel para metro;
   }
   lastxR=x;
   lastyR=y;
   return velocidade;
}
```

```
//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------

IplImage*  TrackerCor (IplImage* img, Mat &frame, int &xb, int &yb){
   int vet[2];
          imgYellowThresh = GetThresholdedImage(img);

          if(imgScribble == NULL)// Se este é o primeiro quadro, é preciso inicializá-lo
                {
                    imgScribble = cvCreateImage(cvGetSize(img), 8, 3);
                }

          // Calcula os momentos para estimar a posicao da bola
          getPosition(imgYellowThresh,vet);
          xb = vet[0];
          yb = vet[1];


          // Queremos desenhar uma linha apenas se a sua posição válida
          //if(lastX>0 && lastY>0 && posX>0 && posY>0)
         // {
              // Desenha uma linha amarela a partir do ponto anterior ao ponto atual, a
             // cvLine(imgScribble, cvPoint(posX, posY), cvPoint(lastX, lastY), cvScalar
         // }


          // junta a imagem do rabisco e o quadro...
          // cvAdd(img, imgScribble, img);
          //cvShowImage("thresh", imgYellowThresh);
          //cvShowImage("videoooo", img);


          Mat imagem (img);
          //desenhar a posicao do objeto na tela
          drawObject(xb,yb,frame,false);


          //imshow("draw",imagem);
```

```
          //trackFilteredObject(posX,posY,imgYellowThresh,imagem);

           //imshow("agora nova",imagem);
          return imgYellowThresh;
          cvReleaseImage(&imgYellowThresh);


}

//-------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------------


void simpleBackgroundSubtraction(  Mat back, Mat frame){
IplImage *img_fore;

   cv::absdiff(back,frame,frame);// diferença absoluta entre duas imagens
   cv::threshold(frame,frame,15,255,CV_THRESH_BINARY); // seta o threshold para ign
   cv::imshow("Image output",frame); // display image
}


//-------------------------------------------------------------------------------------
```