UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

ANALYTICS-BOL: UM PROTÓTIPO PARA GERAÇÃO DE ESTATÍSTICA EM UM JOGO DE VOLEIBOL

PAOLA ADRIANO

BLUMENAU
2018

PAOLA ADRIANO

ANALYTICS-BOL: UM PROTÓTIPO PARA GERAÇÃO DE ESTATÍSTICA EM UM JOGO DE VOLEIBOL

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Aurélio Faustino Hoppe, Mestre - Orientador

BLUMENAU
2018

ANALYTICS-BOL: UM PROTÓTIPO PARA GERAÇÃO DE ESTATÍSTICA EM UM JOGO DE VOLEIBOL

Por

PAOLA ADRIANO

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Aurélio Faustino Hoppe, Mestre – Orientador, FURB
Membro:	Prof. Dalton Solano dos Reis, Mestre – FURB
Membro:	Prof. Ruy Fernando Marques Dornelles, Mestre – FURB

Blumenau, 09 de julho de 2018

Dedico este trabalho a minha mãe, por sempre ter me motivado a seguir na área de tecnologia.

AGRADECIMENTOS

À minha família, pelo amor, incentivo e apoio ao longo dos anos.

Aos meus amigos, que fizeram esses anos de estudos serem mais alegres e divertidos.

Aos meus professores, que sempre me transmitiram seus conhecimentos e orientações.

Ao meu orientador, Aurélio Faustino Hoppe, pela paciência e dedicação para conclusão desse trabalho.

O futuro pertence àqueles que acreditam na beleza de seus sonhos.

Eleanor Roosevelt

RESUMO

Este trabalho apresenta um protótipo para geração de estatística de pontuação em jogo de vôlei. As imagens foram capturadas de forma manual de vídeos feitos em um treino do time feminino do Blu Vôlei no ginásio da FURB. Foram selecionadas imagens que continham a bola sobre a quadra, demonstrando um ponto, para depois serem submetidas a várias etapas de processamento de imagens e por fim, gerar estatísticas de quantas vezes ela caiu em determinado quadrante da quadra. A primeira etapa do processamento de imagens tinha como objetivo determinar o que era a quadra. Primeiramente, a imagem foi convertida de RGB para o sistema de cores HSV. O canal V foi utilizado para delimitar a quadra, sendo feito a segmentação por limiariazação e a remoção de ruídos por meio dos operadores morfológicos de dilatação e erosão. A partir disto, no passo seguinte, as jogadoras são removidas da quadra para não existir a possibilidade delas serem confundidas com a bola. A terceira etapa teve como objetivo o descobrimento do centro da quadra, para processar as partes de forma isoladas, limitando a área de busca pela bola. Neste processo foi utilizada a transformada de Hough. A partir da parte esquerda da quadra, a imagem foi convertida novamente de RGB para HSV, onde utilizou-se o canal H para predizer a localização da bola dentro da quadra. Após a identificação do local, calculou-se o porcentual de quantas vezes a bola apareceu em um determinado quadrante. No total foram analisadas 45 imagens, onde em 32 a bola estava posicionada no lado esquerdo da quadra e em 13, ela estava posicionada na parte direita da quadra. Nas 32 imagens do lado esquerdo, somente em duas a bola não foi encontrada e consecutivamente computando as estatísticas de forma errada. Já nas 13 imagens do lado direito, o protótipo funcionou corretamente até a etapa de recorte da meia quadra, mas não conseguiu identificar em nenhum dos casos a bola corretamente.

Palavras-chave: Vôlei. Processamento de imagens. Geração de estatísticas.

ABSTRACT

This work presents a prototype for generation of score statistics in volleyball game. The images were captured manually from videos made in a training session of the women's team of Blu Volleyball at the FURB gymnasium. Images were selected that contained the ball on the court, showing a point, then being subjected to several stages of image processing and, finally, generating statistics of how many times it fell in a certain quadrant of the court. The first step of image processing was to determine what the court was. First, the image was converted from RGB to the HSV color system. The V channel was used to delimit the court, being made the segmentation by thresholding and the noise removal by the morphological operators of dilation and erosion. From this, in the next step, the players are removed from the court so there is no possibility of them being confused with the ball. The third step was to discover the center of the court, to process the parts in isolation, limiting the area of search for the ball. In this process the Hough transform was used. From the left side of the block, the image was converted again from RGB to HSV, where the channel H was used to predict the location of the ball inside the court. After identifying the location, the percentage of how many times the ball appeared in a given quadrant was calculated. In total, 45 images were analyzed, where in 32 the ball was positioned on the left side of the court and in 13, it was positioned on the right side of the court. In the 32 images on the left side, in only two the ball was not found and consecutively computing the statistics in the wrong way. Already in the 13 images on the right side, the prototype worked correctly until the step of cutting half a block, but could not identify the ball correctly in either case.

Key-words: Volleyball. Image processing. Generation of statistics.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2D – Duas Dimensões

EE – Elemento Estruturante

HSV – Hue, Saturation, Value

OpenCV – Open Source Computer Vision Library

RF – Requisitos Funcionais

RGB – Red, Green, Blue

RNF – Requisitos Não Funcionais

UML – Unified Modeling Language

XML – Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Criado no ano de 1895 pelo professor William Morgan, em Massachusetts, Estados Unidos, o Voleibol é um jogo desportivo que nasceu da necessidade de variar as atividades lúdicas destinadas a recreação da vida e do clube. Sua criação teve como objetivo criar um jogo que não fosse tão rude e fatigante como o basquetebol, e que não necessitasse de tantos instrumentos como o tênis (LOTUFO, 1976, p. 5).

O Voleibol é um esporte coletivo que para sua execução, é necessária uma quadra com medidas pré-definidas, uma rede que divide a quadra em duas partes e uma bola. Por parte dos participantes, é necessário ter o número mínimo e máximo de seis jogadores de uma equipe em uma partida. Como o jogo de vôlei não tem empates à equipe que fizer mais pontos vence a equipe adversária. A marcação de um ponto acontece de diversas maneiras, mas as mais conhecidas e praticadas estão relacionadas a fazer a bola aterrissar sobre a quadra adversária como resultado de um ataque ou de um bloqueio bem sucedido (LOTUFO, 1976, p. 12-32).

Atualmente, as equipes sentem a necessidade de estudo no jogo de vôlei, pois é um jogo complexo, de alto nível de exigência técnica onde é necessário tomar decisões rápidas. Devido as suas características e complexidade, é fundamental uma equipe estudar bem a sua equipe adversária, para que seja possível tomar decisões mais certeiras adquirindo o máximo de rendimento em quadra (PESSOA; BERTOLO; CARLAN, 2009, p. 24-25).

Segundo Okazaki et al. (2012), a partir de análises estatísticas é possível diagnosticar características do esporte, assim como, encontrar relações causa-efeito para possíveis predições do desempenho. Tal diagnóstico pode auxiliar na escolha de estratégias para tentar neutralizar a ação do adversário ou adquirir algum tipo de vantagem no jogo.

Em função da importância em quantificar os parâmetros e variáveis potencialmente úteis para a avaliação do desempenho desportivo, diversos sistemas para coleta e análise de dados têm sido utilizados. Esses sistemas são diferenciados em função da natureza de seus dados e da forma em que eles são analisados. Entre os métodos mais simples e mais utilizados em esportes amadores e profissionais encontra-se a utilização da análise estatística do desempenho, também chamado de *scout*. Basicamente, o *scout* é a quantificação da frequência (absoluta ou relativa) que ocorre em um determinado evento. Em razão de sua simplicidade, para realizar uma coleta de dados pelo método de *scout* bastaria apenas um papel, uma caneta e um observador experiente (OKAZAKI et al., 2012). Contudo, com o avanço tecnológico, várias empresas e pesquisadores estão utilizando análise por imagens para facilitar o levantamento de informações estatísticas a respeito do jogo. Através dela é possível

identificar características das equipes ou atletas sem a necessidade de outras tecnologias envolvidas, podendo obter resultados em tempo real e de maneira simplória (MARQUES; VIEIRA, 1999, p. 2-3).

Diante do exposto, este trabalho apresenta o desenvolvimento de um protótipo que utilizará técnicas de processamento de imagens para analisar e gerar de estatísticas em um jogo de voleibol.

1.1 OBJETIVOS

O objetivo deste trabalho é disponibilizar um protótipo para geração de estatísticas de ataques em um jogo de voleibol utilizando processamento de imagens.

Os objetivos específicos são:

a) identificar a área/linhas que delimitam a quadra de vôlei;
b) identificar a bola e o local onde ela está caindo na quadra;
c) gerar estatísticas das posições que os adversários normalmente atacam as bolas.

1.2 ESTRUTURA

Este trabalho está dividido em quatro capítulos. O primeiro capítulo apresenta a motivação para este trabalho e os objetivos do mesmo. O segundo capítulo apresenta a fundamentação teórica que embasa o desenvolvimento do trabalho. No terceiro capítulo é descrito o detalhamento da implementação do método e os resultados dos testes realizados para validar a implementação. O quarto capítulo contempla as conclusões e limitações, assim como sugestões para possíveis extensões deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

A seção 2.1 descreve a geração de estatística em jogo de vôlei. A seção 2.2 explica o conceito da morfologia matemática e suas principais operações. A seção 2.3 aborda as técnicas de segmentação de imagem. A seção 2.4 dedica-se a explicar a transformada de Hough. Por fim, a seção 2.5 apresenta os trabalhos correlatos.

2.1 ESTATÍSTICA EM JOGO DE VÔLEI

A geração de estatística da equipe adversária em um jogo de vôlei é fundamental para o melhor desempenho da equipe, já que por suas características, é um esporte que necessita de tomada de decisões rápidas (PESSOA; BERTOLO; CARLAN, 2009, p. 24). Segundo os autores, ter conhecimento das áreas da quadra e, onde a bola normalmente é lançada pelo time adversário, com objetivo de fazer um ponto, é necessário para se criar estratégias ofensivas e defensivas (PESSOA; BERTOLO; CARLAN, 2009, p. 25).

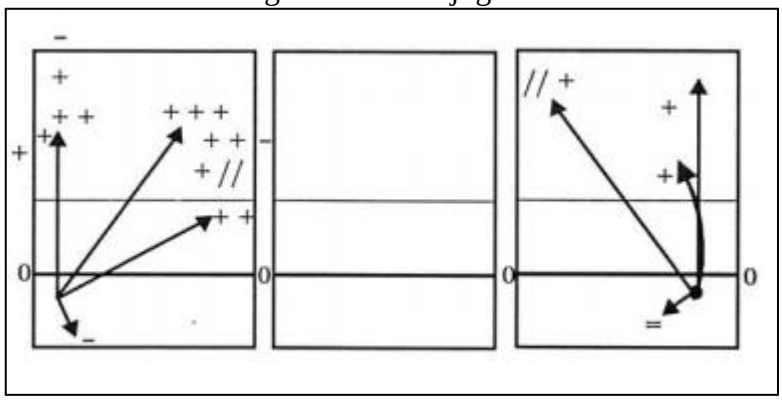
Segundo Shondell e Reynaud (2005), as estatísticas no voleibol são difíceis de registrar e ainda mais difíceis de analisar da forma correta. São muitas vezes enganosas, porque podem ser idealizadas e mal interpretadas. Quando o levantamento de dados vem a partir da visão do jogador, normalmente ficam equivocadas, porque na maioria das vezes o jogador só lembra das suas melhores jogadas e não das suas jogadas ruins, só conseguindo apontar o erro para os seus companheiros. Uma estatística para ser eficaz, necessita ser precisa, válida e abrangente (SHONDELL; REYNAUD, 2005, p. 315).

Rezende (2006) descreve a necessidade dos times de mensurar quanto realmente vale um jogador, e não o que às vezes ele nos faz pensar que vale por ter feito uma jogada sensacional. Por isso, ele substituiu os tradicionais olheiros, que muitas vezes julgam de forma subjetiva os jogadores a serem contratados por um processo estatístico desenvolvido, possuindo nas mãos informações amplas e exatas sobre os jogadores. Com isso, é possível fazer um estudo estatístico permitindo que se tenha o verdadeiro desempenho do jogador.

Segundo Rezende (2006), na seleção brasileira de vôlei existem duas etapas para geração de estatísticas. A primeira é a tática, que faz um mapeamento da quantidade, do percentual e dos tipos de jogadas do time adversário onde são analisadas as tendências de todos os jogadores, como as direções, preferencias e posicionamento. A partir desses dados são formuladas as estratégias para enfrentá-los. A segunda etapa, a técnica, é o estudo do seu próprio time, onde se analisa como o jogador se comportou em cada fundamento e qual foi o seu aproveitamento final.

Rezende (2006) destaca que esses controles são feitos através de um software onde o técnico informa os dados da partida e dos jogadores de forma manual. A partir disso, o software compara os resultados disponibilizados pelo técnico e os existentes em uma base de dados, gerando informações mais precisas sobre os eventos que acontecem em uma partida de vôlei. A análise também pode ser feita de forma gráfica, onde, entre outras informações, é possível ver o local de ataque preferido pelo jogador. Em cada set de um jogo, para cada jogador, é confeccionada uma ficha com três quadras, para que seja possível, de forma manual desenhar em quais locais da quadra as bolas foram atacadas pelo jogador. É marcado + se o ataque foi eficaz, / se a bola continuou em jogo e - se o ataque foi errado (RIBEIRO, 2004, p. 166). A Figura 1 demonstra a ficha de um jogador preenchida.

Figura 1 - Ficha jogador



Fonte: Ribeiro (2004, p. 166).

Rezende (2006) explica que devido à inserção manual, podem existir alguns equívocos ou algo que não foi percebido na partida, resultando em estatísticas incorretas. Rezende (2006) sugere que para melhorar a precisão das informações, é interessante utilizar um software que realiza a coleta dos dados de forma automática. Nesse cenário, a base para qualquer identificação de forma automática em um jogo de vôlei, é descobrir onde estão os limites da quadra, jogadores, a bola e o local ao qual ela está caindo ou sendo jogada (PESSOA; BERTOLO; CARLAN, 2009, p. 25).

2.2 MORFOLOGIA MATEMÁTICA

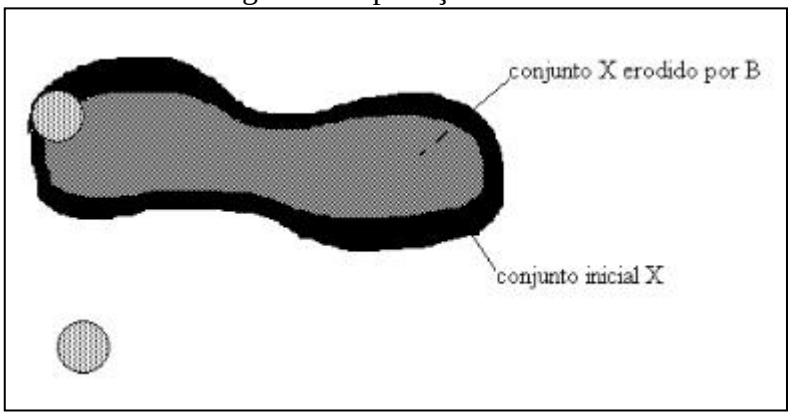
A morfologia matemática tem como objetivo estudar as estruturas geométricas presentes em uma imagem. Como uma imagem pode conter qualquer tipo de matéria, a morfologia matemática pode ajudar diversas áreas, como biologia, medicina, controle de qualidade, visão robótica, entre diversas outras (FACON, 1996, p. 2).

Facon (1996) explica que a morfologia matemática consiste em analisar estruturas geométricas a partir de um conjunto definido e conhecido chamado de Elemento Estruturante

(EE). Esse elemento vai ser comparado com entidades contidas na imagem, podendo assim ser extraídas algumas informações, que poderão ser diferentes com base no tipo do elemento e o tipo da imagem estudada. Para a análise da imagem é necessário aplicar algum tipo de transformação. Entre os tipos de transformação que existem, a erosão e a dilatação serão explicadas a seguir.

A erosão tem por principal objetivo no processamento de imagens, a eliminação de ruídos, facilitando a análise da imagem. O processo de erosão é feito com o elemento estruturante deslizando sobre a imagem X e para cada pixel é verificado a configuração de sua vizinhança em relação ao elemento estruturante. Cada pixel do elemento estruturante, tenta aparelhar-se com seu vizinho. Para ser considerado relevante, cada pixel do elemento estruturante deve estar contido na mesma posição do seu vizinho, onde será mantido na imagem. Caso contrário, o pixel é marcado como não relevante e será excluído (FACON, 1996, p. 16). A Figura 2 mostra a aplicação da erosão.

Figura 2 - Aplicação Erosão

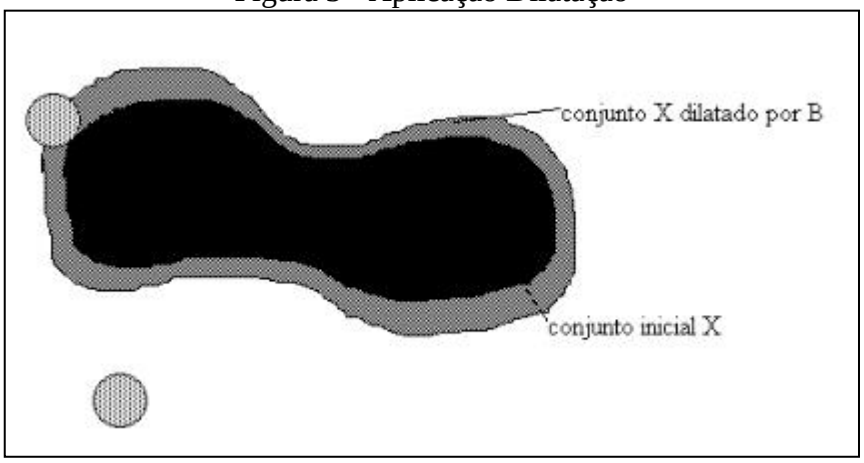


Fonte: Facon (1996, p.15).

Facon (1996) explica que é possível observar alguns resultados aplicando a erosão em uma imagem, como diminuição das partículas da imagem, eliminar partículas de tamanho inferior ao tamanho estruturante, aumentar buracos e permitir a separação de partículas próximas.

A dilatação é um entendimento complementar a erosão, mas com o resultado obtido diferente. Na erosão diminuímos alguns pixels, na dilatação o objetivo é aumentar. O processo de dilatação é feito com o elemento estruturante deslizando sobre a imagem X e para cada pixel é verificado uma possível interseção com seu vizinho. Caso exista interseção, o pixel será marcado como relevante. Caso contrário o pixel será marcado como não relevante e será excluído. Isso fará com que os pixels próximos se juntem (FACON, 1996, p. 22). Na Figura 3 demonstra a aplicação da dilatação.

Figura 3 - Aplicação Dilatação



Fonte: Facon (1996, p. 22).

Alguns resultados são obtidos aplicando a dilatação em uma imagem, como engordar partículas, preencher pequenos buracos e conectar partículas próximas (FACON, 1996, p. 26).

Tanto a erosão como a dilatação, tem por objetivo corrigir defeitos na imagem. A erosão é muito utilizada para a remoção de pixels e objetos indesejados na imagem e principalmente, na desconexão de elementos próximos. Já a dilatação é muito utilizada para preenchimento de cavidades brancas em objetos. Ela também é utilizada para aumentar um elemento, facilitando a sua identificação ou preenchimento do objeto (FACON, 1996, p. 33).

2.3 SEGMENTAÇÃO

Para análise de uma imagem existem diversas etapas, dentre elas encontra-se a segmentação. Ela tem por objetivo subdividir a imagem em partes ou em objetos. Como exemplo, Gonzales e Woods (2000) destacam a identificação de veículos na estrada. O primeiro passo é a segmentação da estrada e, por subsequência, a segmentação dos elementos que podem ser veículos comparados ao seu tamanho (GONZALES; WOODS, 2000, p. 295).

A segmentação pode ser uma das tarefas mais difíceis no processamento da imagem, porque o resultado dela implicará diretamente no resultado final da análise. O que torna ela complexa é que os elementos desejados normalmente estão misturados com informações ruidosas que podem distorcer as informações analisadas (FACON, 1996, p.261). Em alguns casos são utilizadas técnicas para realçar os objetos de interesse, como por exemplo, o uso de imageamento infravermelho para melhor detecção de objetos que se destacam no calor (GONZALES; WOODS, 2000, p. 295).

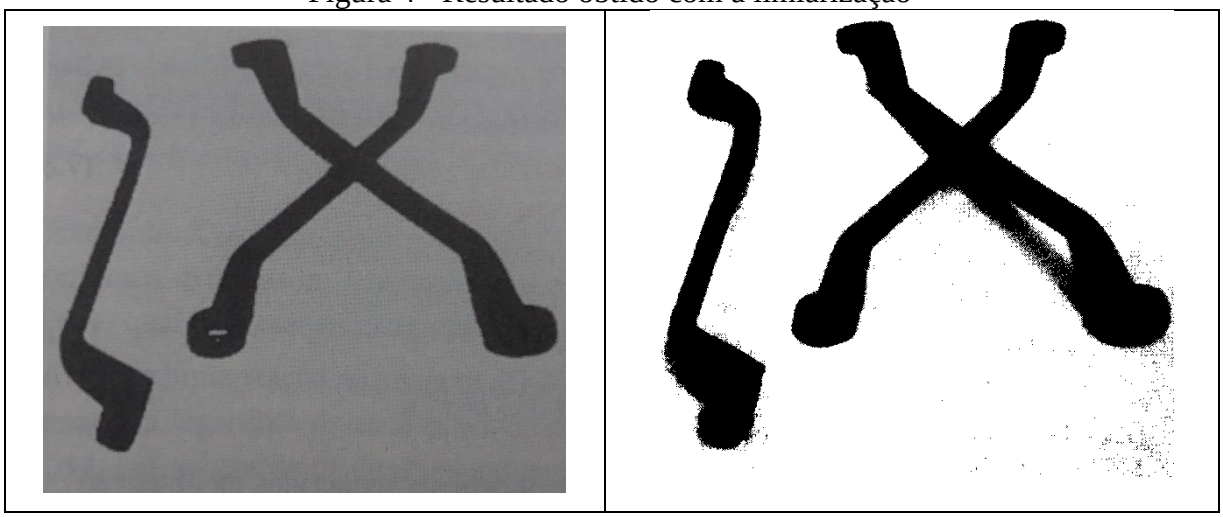
Existem duas abordagens utilizadas nos algoritmos de segmentação para imagens monocromáticas. A primeira é a descontinuidade, que tem objetivo de subdividir a imagem se baseando nas mudanças bruscas nos níveis de cinza. Os principais resultados dessa

abordagem são a detecção de pontos isolados e a detecção de bordas e linhas na imagem. A segunda abordagem se baseia em limiarização, crescimento, divisão e fusão das regiões da imagem (GONZALES; WOODS, 2000, p. 296).

2.3.1 LIMIARIZAÇÃO

A limiarização é um dos processos mais importantes para a segmentação da imagem. Ela se constitui na diferença dos níveis de cinza que o objeto e o fundo da imagem possuem. Para separar o objeto desejado do fundo da imagem é necessário estabelecer um limiar que separa os dois grupos. O grupo com níveis de cinza abaixo do limiar e o grupo com níveis de cinza acima do limiar, podendo assim, destacar o que é o fundo e o que é o objeto (GONZALES; WOODS, 2000, p. 316). A Figura 4 mostra o antes e depois da aplicação da limiarização.

Figura 4 - Resultado obtido com a limiarização



Fonte: Gonzales e Woods (2000, p. 319).

A imagem à esquerda é a imagem de entrada, e a imagem à direita é o resultado depois de se aplicar a limiarização, onde é possível ver nitidamente o que é o objeto e o que é o fundo. O objeto de interesse foi estabelecido pelo nível de cinza de cada pixel. Quando o nível de cinza for maior que um determinado valor, ele é marcado de preto, se não for menor, é marcado como branco, tendo como resultado uma imagem binarizada (GONZALES; WOODS, 2000, p. 318).

2.4 TRANSFORMADA DE HOUGH

A transformada de Hough é um método para detecção de formas em imagens binarizadas, como linhas, círculos e elipses. Ele foi desenvolvido pelo Paul Hough em 1962 e patenteada pela IBM (PIVETTA; MANTOVANI; ZOTTIS, 2018).

O método consiste em detectar grupos de pontos colineares ou quase colineares. Para saber se os pontos são colineares, calcula-se os coeficientes das retas para cada par de pontos efetuando uma permutação entre todos, dois a dois, comparando em seguida quais coeficientes são iguais.

Segundo Pivetta, Mantovani e Zottis (2018) em um dado ponto da imagem podem passar várias retas. E, uma reta que passa por esse ponto é definida por dois parâmetros (0, R), onde "R" indica a distância mínima entre a reta e a origem do plano cartesiano e, "0" é o ângulo que o segmento da reta "R" faz com o eixo "X" do plano cartesiano. Os pontos que estão sobre essa reta podem ser representados pela equação apresentada no Quadro 1.

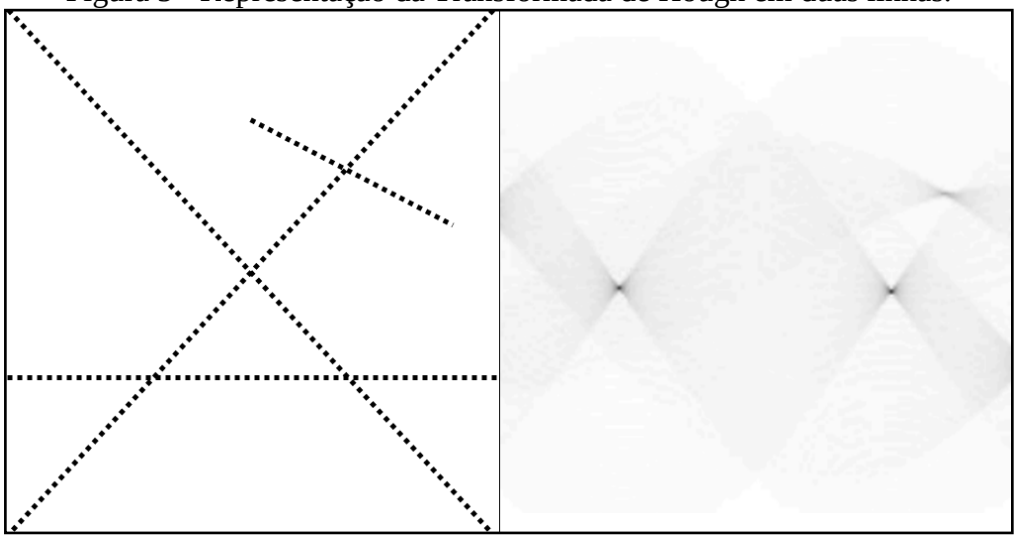
Quadro 1 - Equação da reta no espaço de Hough

$R = X . \cos 0 + Y . \text{sen } 0$

Fonte: adaptado de Pivetta, Mantovani e Zottis (2018).

Essa equação associa cada reta da imagem a um único ponto (0, R) no Espaço de Hough. Diversas retas passam por um ponto no Espaço de Hough, formando um senóide (TROFINO, 2014, p. 29). A Figura 5 demonstra a criação dos senóides em uma imagem com duas linhas, após a aplicação da transformada de Hough.

Figura 5 - Representação da Transformada de Hough em duas linhas.



Fonte: adaptado de Trofino (2014 p. 29).

Segundo Trofino (2014, p. 29), a Transformada de Hough para retas é a base para encontrar outras formas em imagens, como por exemplo, o círculo. Ele possui três parâmetros, que representam suas coordenadas do centro e de seu raio, transformando o espaço de Hough em tridimensional. O autor também indica que para encontrar objetos circulares no espaço de Hough, é necessário criar um espaço acumulador, que é feito por uma célula para cada pixel, conforme mostra o Quadro 2.

Quadro 2 - Equação do círculo no espaço de Hough

$$R^2 = (i - a)^2 + (j - b)^2$$

Fonte: adaptado de Trofino (2014 p. 29)..

Essas células são representadas na equação pela letra "a". Para cada valor encontrado, é feito uma busca de todos os possíveis valores de "b" que satisfaçam a equação do círculo. No final, é feito uma comparação para encontrar as células que possuem valores maiores que suas vizinhas (TROFINO, 2014, p. 29).

Trofino (2014) ressalta que a transformada de Hough só é eficiente para encontrar os círculos quando a célula correta receber um valor alto de votos, fazendo com que ela possa ser detectada facilmente entre os demais ruídos da imagem. Sendo assim, é necessário que a imagem possua uma boa qualidade e que seus círculos sejam bem definidos (TROFINO, 2014, p. 29).

2.5 TRABALHOS CORRELATOS

A seguir são apresentados trabalhos com objetivos semelhantes ao tema proposto. A seção 2.5.1 apresenta o trabalho de conclusão de mestrado de Alves (2012), que desenvolveu um sistema na linguagem Java para análise de esportes genéricos automaticamente. Na seção 2.5.2 é descrito o sistema Prozone (1995) que analisa jogos esportivos e gera analises referente ao jogador e do time como um todo. Na seção 2.5.3, é apresentado o sistema BBall Stats (1992) que fornece acesso instantâneo a estatísticas dos jogadores e da equipe de basquete. Por fim, na seção 2.5.4 é apresentado o sistema Data Volley que gera estatísticas em jogo de voleibol.

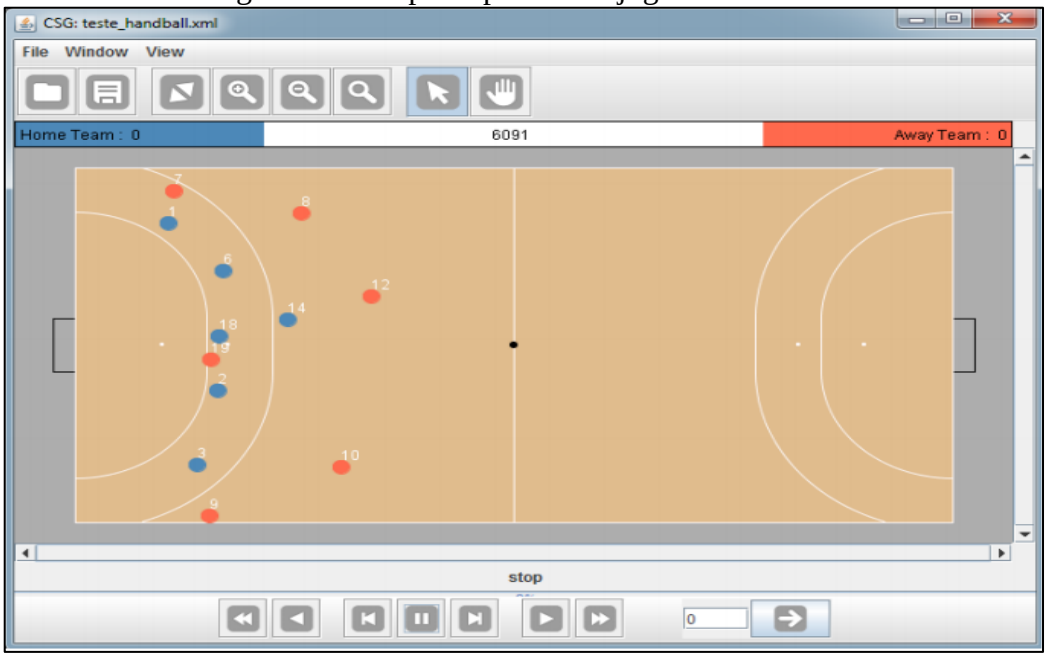
2.5.1 Sistema de observação e registo do desempenho táctico-técnico em jogos esportivos coletivos

Alves (2012) desenvolveu um sistema para esportes, que analisa automaticamente os eventos dos jogos de futsal, futebol, handebol e basquetebol. O sistema detecta as coordenadas cartesianas dos jogadores e da bola, passes bem-sucedidos, gols, chutes de baliza, chutes de canto e lançamentos da linha lateral. O sistema permite que sejam feitas representações de outros jogos não abordados, através da configuração manual das características do campo e dos acontecimentos do jogo.

Desenvolvido na linguagem de programação Java, com o recurso da biblioteca `java-xmlbuilder` para a leitura e escrita de arquivos XML, o sistema possui uma interface que permite o usuário criar um novo projeto. Nele é necessário escolher o esporte desejado e

realizar algumas configurações, como informar as dimensões do terreno do jogo, selecionar o tipo de área e suas dimensões, as marcas de pênalti/lançamento, os cantos, o tamanho do círculo central e as dimensões dos alvos. Essa interface também permite ao usuário carregar um projeto já criado e continuar seu trabalho a partir dele. A Figura 6 mostra a tela principal de um jogo de handebol.

Figura 6 - Tela principal de um jogo de handebol



Fonte: Alves (2012, p. 74).

Alves (2012) descreve que o sistema tem 3 módulos: módulo de animação, módulo de eventos e o módulo de estatísticas. O módulo de animação permite a visualização do jogo em 2D e permite o controle da animação, como definir a velocidade da animação ou saltar diretamente para o tempo desejado. O módulo de eventos permite que o usuário crie ações e situações para cada jogo, levando em consideração as configurações feitas para cada esporte. Por fim, o módulo de estatística possui uma tabela com a contagem dos eventos do jogo. Esse módulo permite a visualização dos eventos detectados automaticamente pelo sistema, como também os que forem manualmente inseridos pelo usuário. Como resultado, o sistema teve uma boa porcentagem de detecção de ações automáticas e criação visual das características de cada campo atenderam ao esperado.

Para avaliação do sistema, Alves (2012) realizou uma comparação entre o método de detecção automática do seu próprio sistema e os resultados de um jogo final da FIFA World Cup 2006 entre as seleções da Itália e da França. A Tabela 1 mostra os resultados obtidos.

Tabela 1 - Resultados da detecção dos eventos do jogo final do FIFA World Cup 2006

Evento	Observados no jogo	Detectados na ferramenta	Detectados incorretamente	Porcentagem de acertos	Porcentagem de erro
Passe com sucesso	641	661	41	96,7%	3,3%
Gol	2	2	0	100,0%	0,0%
Canto	10	7	0	70,0%	0,0%
Lançamento de lateral	44	33	1	72,7%	3,0%
Chute de baliza	21	21	0	100,0%	0,0%

Fonte: adaptado de Alves (2012, p. 68).

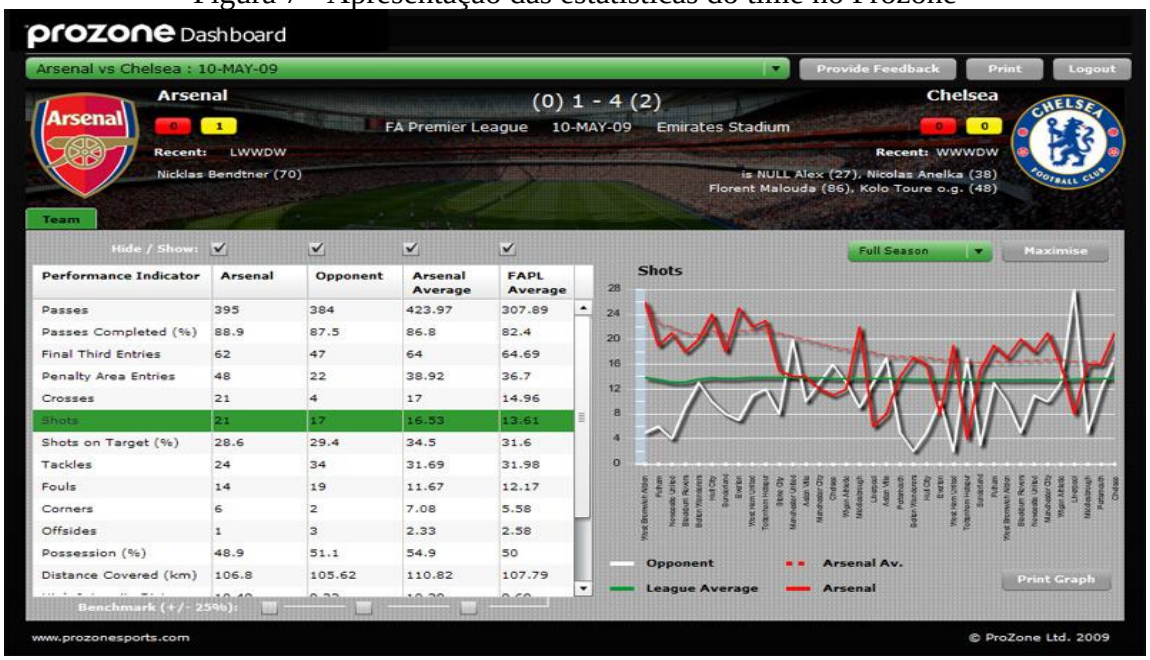
Dos passes detectados de forma errônea, a maioria são resultados de interrupções como faltas ou foras de jogo. Os erros identificados também são devido a má configuração da localização do sistema na zona do campo, fazendo com que os jogadores e a bola desaparecessem no meio da partida. Essa situação fez com que não fossem detectados 3 chutes de canto e de 8 lançamentos da linha lateral. Apesar do sistema não ter 100% de acertos, levando em conta a complexidade de um jogo de futebol, ele teve uma boa média na detecção automática das ações de um jogo, observando todos os gols da partida e todos os chutes de baliza.

2.5.2 PROZONE

Prozone (1995) é um sistema de análise de jogos esportivos, que mostra as informações estatísticas relevantes de cada jogador individualmente, da equipe e dos adversários por jogo. Ele utiliza dados colhidos para proporcionar estatísticas técnicas, táticas e de performance física completa. O sistema atende aos esportes: futebol, rugby, hóquei no gelo, basquetebol, basebol e futebol americano.

O sistema realiza a monitorização individual de atletas, faz a gestão de carga dos jogadores para evitar lesões, monitorando seu desempenho físico durante os jogos/treinamentos, realiza análise de desempenho de todos os elementos da equipe. Permite assim, uma análise técnica, tática e física detalhada. Ele também oferece dados de rastreamento avançados capturados por diversas câmeras dentro do estádio. Segundo Prozone (1995), com estas informações é possível implantar processos organizacionais para a equipe que sejam mais eficientes para determinadas situações. A Figura 7 apresenta o painel de estatísticas dos resultados obtidos de um jogo entre Arsenal x Chelsea.

Figura 7 - Apresentação das estatísticas do time no Prozone



Fonte: Prozone (1995).

O Prozone utiliza câmeras (entre 8 a 10 câmeras) posicionadas em locais específicos no campo de jogo. Essas câmeras capturam sem interrupções os eventos ocorridos durante a partida. Com as imagens obtidas do jogo, o sistema produz as informações sobre o desempenho da equipe e de cada jogador. O sistema também cria uma representação do jogo em 2D, para melhor visualização dos técnicos do que ocorreu no jogo.

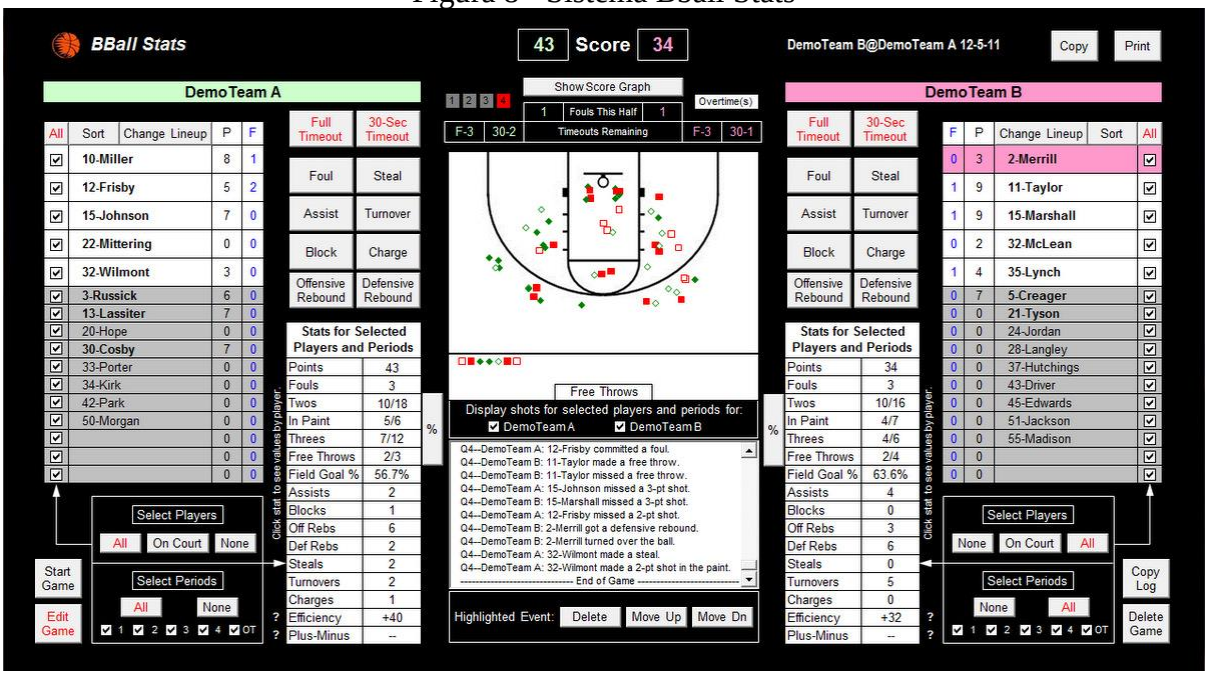
Com o sistema cada detalhe dos jogadores e adversários passou a ser percebido, como a perda de velocidade, regiões que erram mais passes, etc. Suas informações em forma de gráficos ajudam o usuário que está analisando as informações a tomar decisões mais rápidas. Mesmo sendo um sistema que impulsiona os resultados das equipes, no Brasil não existe nenhum time que o utiliza por causa do custo, em média R$ 400 mil por temporada. Fora o custo alto, outro problema do sistema é o fato de não ser portável, ou seja, a equipe que possui esse sistema, somente pode utilizar nos jogos em seu próprio campo.

2.5.3 BBall Stats

BBall Stats (1992) é um sistema para gravar eventos durante jogos de basquete, que fornece acesso as estatísticas da equipe e dos jogadores. Exibe os resultados sobre o desempenho atual da partida e também de partidas passadas em tempo real. O sistema fornece um único lugar para registrar todas as ações dos jogos. Para utilizar o sistema, é necessário cadastrar os jogadores. As informações dos jogadores vão ficar disponíveis para os futuros jogos.

O treinador possui em tempo real as estatísticas do desempenho individual de cada jogador e o desempenho da sua própria equipe como um todo ou da equipe adversária, para que ele consiga tomar decisões durante a partida. No final da partida, os resultados são mesclados com as informações já existentes em sua base de dados, para fornecer estatísticas. Na Figura 8 é apresentada a tela do sistema.

Figura 8 - Sistema Bball Stats



Fonte: BBall Stats (1992).

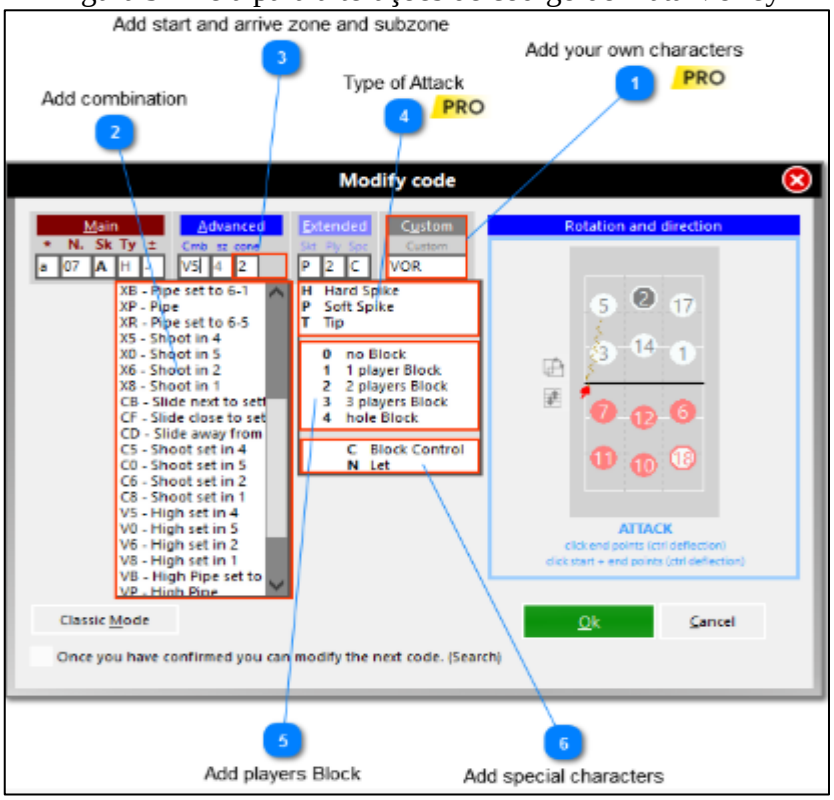
O sistema não necessita de internet para funcionar e para utilização do mesmo é necessário ter instalado o Microsoft Excel 2003-2010. Uma vantagem do BBall Stats é a sua mobilidade, ou seja, o treinador pode utilizar o sistema em qualquer lugar. Como desvantagem o sistema apresenta poucas informações ao usuário comparado com outros sistemas, trazendo poucas estatísticas para o usuário. Outra desvantagem é a falta de processamento automático das informações, sendo necessário colocar todas as informações manualmente no sistema, o transformando menos atrativo para o mercado. Sua falta de animação dos eventos do jogo também é um ponto negativo.

2.5.4 Data Volley

O Data Volley (2007) é um sistema criado para geração de estatística e analise em jogos de voleibol. O sistema permite criar personalizações de acordo com as necessidades do usuário e também permite explorar todas as perspectivas do jogo de um modo geral. Os dados de entrada são inseridos no sistema via códigos pré-definidos no software, que podem ser modificados pelos usuários. Via código é possível adicionar informações avançadas, como as

zonas, chamadas de sets, combinações de ataques, etc. Com o usuário não tendo que digitar informações completas, os códigos possibilitam inserir informações de forma mais rápida. A Figura 9 mostra a tela principal para alterações dos códigos respeitados pelo sistema.

Figura 9 - Tela para alterações de código do Data Volley



Fonte: Data Volley (2007).

O sistema também conta com analise de vídeos, onde as informações são processadas de forma automática a partir de câmeras posicionadas na quadra que captam os lances do jogo. O usuário tem possibilidade de assistir a partir de qualquer dispositivo o replay dos últimos 5 pontos durante o jogo. Os vídeos ficam disponíveis em um servidor para que seja possível realizar os estudos necessários no final da partida. O sistema também conta com uma área totalmente preparada para estudar o rendimento de cada jogador, mostrando sua eficiente e evolução ao longo do tempo, afim de preparar exercícios destinados a melhorar o desempenho individual.

O Data Volley funciona somente no sistema operacional Windows. Para uma boa operacionalidade, o fabricante recomenda que o sistema rode em um computador com no mínimo um processador Core i5 e pelo menos 2GB de RAM. O computador também deve possuir conexão com a internet para ativação da licença e download automático das atualizações. O sistema está disponível em vários idiomas, como italiano, inglês, espanhol, japonês, russo, alemão, francês, polonês, holandês, português e chinês.

3 DESENVOLVIMENTO

Nesse capítulo são demostradas as etapas do desenvolvimento do protótipo. Na seção 3.1 é mostrado os principais requisitos. A seção 3.2 apresenta a especificação e implementação do protótipo. Por fim, a seção 3.3 demostra os resultados obtidos.

3.1 REQUISITOS

O protótipo a ser desenvolvido deverá:

a) permitir o carregamento da imagem que representa o fim da disputa do ponto (Requisito Funcional - RF);
b) descobrir em qual região da quadra a bola está caindo (ataques), identificando a bola e a quadra (RF);
c) gerar estatísticas das regiões onde o jogador normalmente ataca (RF);
d) ser implementada na linguagem C++ no ambiente de desenvolvimento Visual Studio (Requisito Não Funcional - RNF);
e) desenvolver o protótipo para executar no sistema operacional Windows (RNF);
f) utilizar a biblioteca OpenCV (RNF).

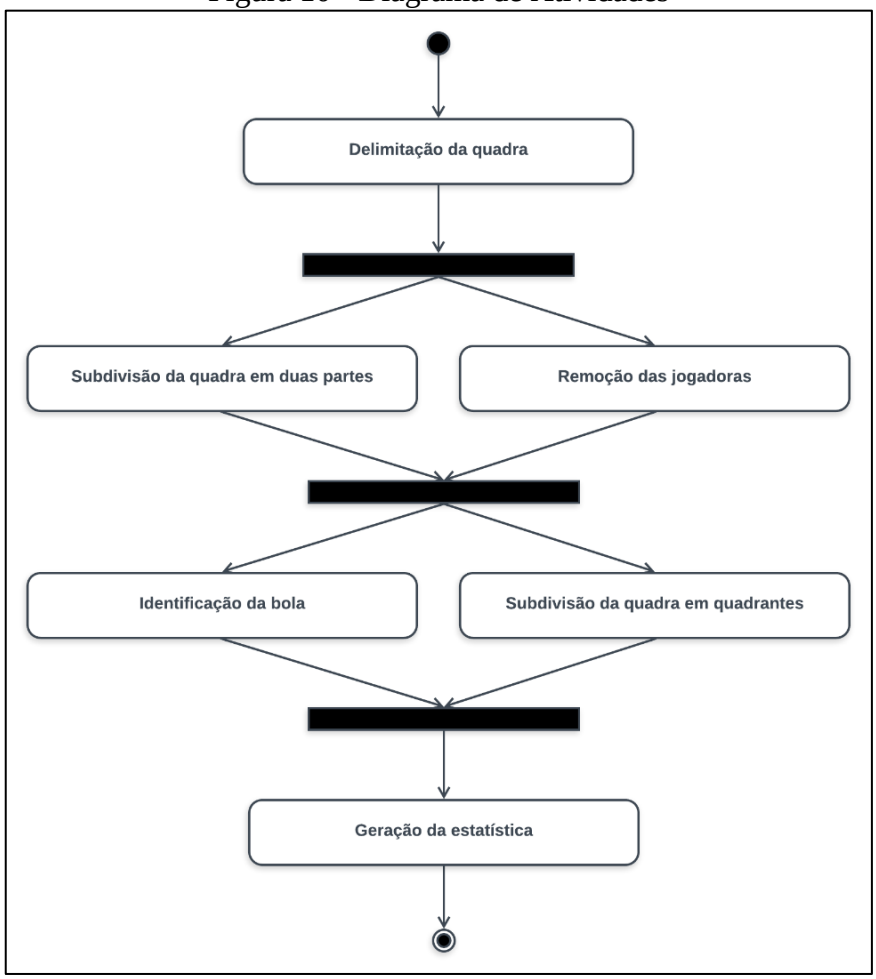
3.2 ESPECIFICAÇÃO

A especificação do protótipo foi representada através de um diagrama de atividades da Unified Modeling Language (UML), utilizando a ferramenta StarUML. A seguir são apresentadas o diagrama de atividades, as técnicas e ferramentas utilizadas e a implementação de forma detalhada.

3.2.1 Diagrama de Atividades

O diagrama de atividades tem por objetivo demonstrar o fluxo dos processos realizados pelo protótipo. A Figura 10 mostra as atividades realizadas no processo para geração de estatística em pontuação no jogo de voleibol.

Figura 10 - Diagrama de Atividades



Fonte: elaborado pela autora.

A partir do diagrama de atividades apresentado na Figura 10 é possível observar que existem diversas etapas antes de chegar ao objetivo final que é encontrar a bola e a sua localização na quadra para geração da estatística. Como as imagens estudadas não possuem apenas a quadra é necessário delimitar o que é a quadra do resto da imagem. Isso se faz necessário, pois ao redor da quadra podem existir objetos que sejam confundidos com a bola.

Depois da delimitação da quadra, faz-se necessário a remoção das jogadoras da imagem. Essa etapa tem por objetivo deixar a imagem com menos elementos possíveis, para facilitar a procura da bola. Juntamente com essa etapa, é possível realizar a divisão da quadra em duas partes. Isso se faz necessário para que as estatísticas sejam geradas somente do lado do time a ser estudado.

Tendo apenas meia quadra, ela é subdividida em nove quadrantes. Isso facilita a identificação da região onde a bola está localizada e consecutivamente, permite a geração de estatísticas do lugar ao qual normalmente acontecem os ataques adversários. Nas próximas seções serão apresentadas as técnicas e as ferramentas utilizadas e os passos para da implementação do protótipo.

3.2.2 Técnicas e ferramentas utilizadas

O protótipo foi desenvolvido utilizando a linguagem de programação C++ no ambiente de desenvolvimento Visual Studio 2015 no sistema operacional Windows 10. A tecnologia utilizada na implementação foi a biblioteca Open Source Computer Vision Library – OpenCV versão 3.2.0. É uma biblioteca multiplataforma para o desenvolvimento de aplicativos da área de visão computacional. Ela fornece diversas soluções como carregamento de imagens, conversões para outros sistemas de cores e separação de canais, segmentação, operações morfológicas, extração de contornos, cálculo de área, cálculo de bounding box, implementação de algoritmos como por exemplo, de bordas de Canny, entre outras funcionalidades.

3.2.3 Implementação

Nessa seção são detalhados os processos utilizados no desenvolvimento do protótipo, a estratégia utilizada para solução, código fonte e o resultado obtido de cada etapa.

3.2.3.1 Captura das imagens e delimitação da quadra

O primeiro passo realizado foi à captura das imagens que seriam utilizadas no processamento até o objetivo proposto. Para isso, contou-se com o apoio da equipe do Blu Vôlei Feminino para obter imagens reais do treino de vôlei. A câmera foi posicionada para baixo próxima a linha central da quadra do ginásio da FURB, com uma distância de 6,5 metros do chão, objetivando pegar toda a extensão da quadra. A câmera utilizada foi uma GOPRO HERO com resolução de 10 megapixels e com qualidade de 2K nas capturas de vídeos. Depois de adquirir os vídeos do treino, separou-se imagens de forma manual com a bola sobre a quadra, indicando a marcação de um ponto. Como não foi possível posicionar a câmera na melhor posição possível, a imagem ficou destacando mais um lado da quadra do que o outro, e não capturou uma parte da borda da quadra. É possível destacar também que pôr a quadra possuir um material que reflete, as luzes ficam bem marcantes na imagem. A Figura 11 mostra um exemplo de imagem capturada.

Figura 11 - Imagem quadra

Fonte: elaborado pela autora.

Depois da construção da base de dados, iniciou-se o processamento das imagens. O primeiro passo realizado foi o carregamento das imagens a serem analisadas, conforme mostra o Quadro 3.

Quadro 3 - Carregamento da imagem

```
int qtdImagens = 32;
for (int i = 1; i <= qtdImagens; i++) {
	std::stringstream numImagem;
	numImagem << i;
	string caminhoImagem = "C:/volei/imagem" + numImagem.str() + ".PNG";
	Mat src = imread(caminhoImagem, IMREAD_COLOR);
```

Fonte: elaborado pela autora.

Na variável `qtdImagens` é possível determinar quantas imagens serão processadas. O próximo passo, depois do carregamento da imagem, é a delimitação da quadra. Para isso, a imagem foi convertida para o modelo de cores HSV, que faz com que a quadra fique destacada. O Quadro 4 mostra a codificação da conversão de RGB para HSV.

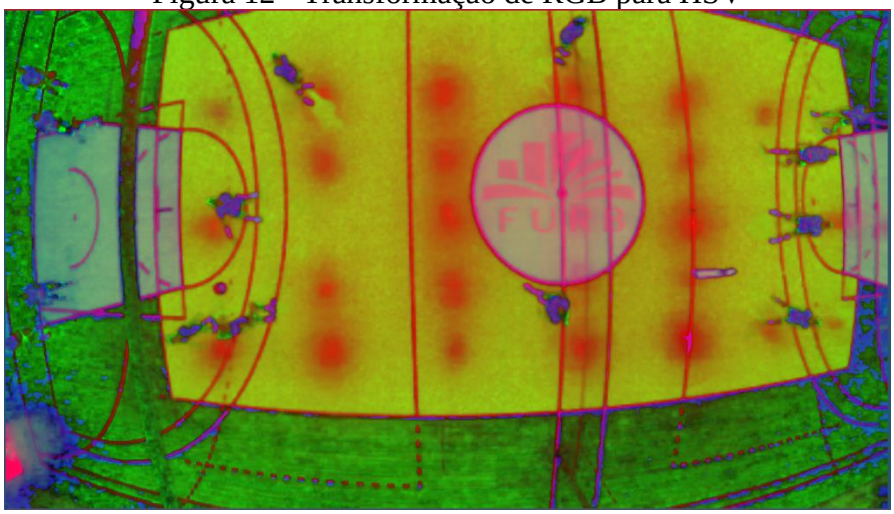
Quadro 4 - Conversão de RGB para HSV

```
Mat hsv;
cvtColor(src, hsv, CV_BGR2HSV);
```

Fonte: elaborado pela autora.

A partir da codificação acima é criada uma nova imagem que receberá a imagem da quadra convertida para HSV. O método `cvtColor` realiza essa conversão com base no parâmetro `CV_BGR2HSV`. O resultado desse processamento pode ser visualizado na Figura 12.

Figura 12 - Transformação de RGB para HSV



Fonte: elaborado pela autora.

O modelo HSV pode ser separado em canais, denominados: canal H (matriz), canal S (saturação) e canal V (valor). Para realçar a quadra, utilizou-se o canal V. O Quadro 5 mostra a codificação para realizar a separação dos canais.

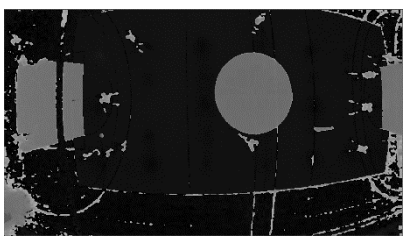
Quadro 5 - Separação dos canais

```
vector<Mat> canal;
split(hsv, canal);
```

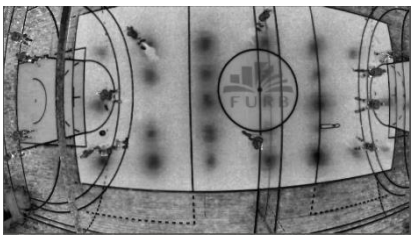
Fonte: elaborado pela autora.

Na codificação acima é criado um vetor que contém as imagens em seus respectivos canais. O método `split` tem por função converter a imagem e popular o vetor. A Figura 13 mostra o resultado da separação dos três canais.

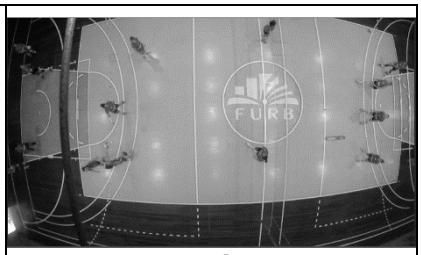
Figura 13 - Separação dos canais HSV



Canal H



Canal S



Canal V

Fonte: elaborado pela autora.

Mediante a separação em canais, pode-se perceber que o canal V realça a quadra de jogo. A partir disso, aplicou-se uma limiarização na imagem. O Quadro 6 mostra a codificação para realização da binarização.

Quadro 6 - Limiarização do canal V

```
Mat bin;
threshold(canal[2], bin, 100, 255, THRESH_BINARY);
```

Fonte: elaborado pela autora.

Para realizar a binarização, utilizou-se o método `threshold` com o valor de limiar de 100. O parâmetro 255 se refere a cor de destino e o parâmetro `THRESH_BINARY` é o tipo de limiarização. Na Figura 14 é mostrado o resultado a aplicação do método `threshold`.

Figura 14 - Limiarização do canal V

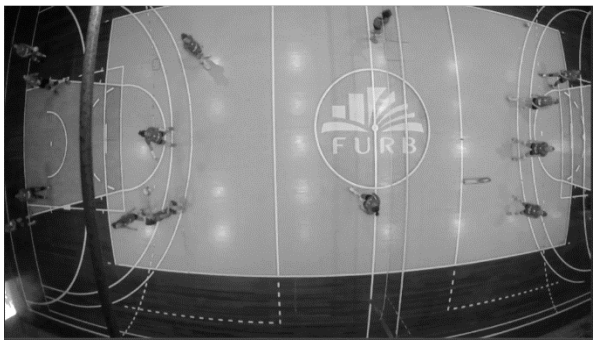


Imagem do canal V

Imagem limiarizada

Fonte: elaborado pela autora.

Após o processo de binarização, pode-se observar que a quadra está totalmente branca, se destacando das laterais da imagem. Antes de recortar a quadra, é necessário remover os ruídos laterais. Para isso, foram utilizados os operadores morfológicos de dilatação e erosão, conforme mostra o Quadro 7.

Quadro 7 - Utilização do erode e dilate

```
int erosion_size = 2;
Mat element = getStructuringElement(MORPH_RECT,
    Size(2 * erosion_size + 1, 2 * erosion_size + 1),
    Point(erosion_size, erosion_size));

Mat erode1;
erode(bin, erode1, element);

Mat dilate1;
dilate(erode1, dilate1, element, Point(-1, -1), 1);
```

Fonte: elaborado pela autora.

Conforme mostrado na codificação acima, a imagem binarizada passa pelo processo de erosão e em seguida por uma dilatação. A erosão faz com que os pixels brancos diminuam, ou seja, as linhas finas serão removidas da imagem. Porém, a delimitação da quadra terá seu tamanho reduzido. Por isso, aplica-se posteriormente uma dilatação para retornar ao tamanho original. Tanto o método `erode` quanto o método `dilate` recebem como parâmetro o elemento estruturante, que determinará a quantidade de vizinhos que serão levados em consideração no processo de remoção de ruídos. A Figura 15 mostra o resultado da remoção dos ruídos.

Figura 15 - Aplicação erode e dilate

Imagem limiarizada

Imagem erodida/dilatada

Fonte: elaborado pela autora.

Após a aplicação dos filtros morfológicos de erosão e dilatação, percebe-se a quadra está delimitada. A partir disso, ela foi utilizada como máscara binária para recortar a imagem original, destacando somente a quadra. O Quadro 8 mostra a codificação do recorte.

Quadro 8 - Operação de recorte

```
Mat imgQuadra;
src.copyTo(imgQuadra, dilate1);
```

Fonte: elaborado pela autora.

Na codificação acima é utilizada o método `copyTo`, que é responsável por criar uma imagem nova a partir do resultado da subtração da imagem original com a imagem resultante do erode/dilate. Essa subtração visa eliminar da imagem original elementos que podem atrapalhar a subdivisão da quadra ou a identificação da bola. A Figura 16 mostra o resultado do recorte da quadra.

Figura 16 - Resultado final da delimitação da quadra

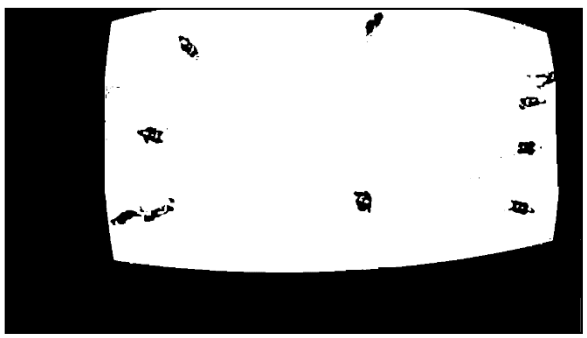


Imagem erodida/dilatada

Imagem recortada

Fonte: elaborado pela autora.

3.2.3.2 Remoção das jogadoras e Subdivisão da quadra em duas partes

Depois de delimitar a quadra, o próximo passo é a remoção das jogadoras. Isso se fez necessário, pois elas possuem características que podem ser confundidas com a bola. O Quadro 9 mostra o trecho de código responsável pela remoção das jogadoras.

Quadro 9 - Remoção das jogadoras

```
vector<Mat> canal1;
split(imgQuadra, canal1);
Mat bin2, imgSemJogadoras, erode2;
threshold(canal1[2], bin2, 100, 255, THRESH_BINARY);
erode(bin2, erode2, element);
imgQuadra.copyTo(imgSemJogadoras, erode2);
```

Fonte: elaborado pela autora.

O processo de remoção das jogadoras iniciou-se com a conversão da imagem para o sistema HSV, binarização do canal V e a aplicação do operador erode. A partir dessas etapas, realizou-se a subtração da imagem da quadra igualmente ao que foi realizado da etapa de delimitação da quadra. A Figura 17 mostra o resultado da remoção das jogadoras.

Figura 17 - Remoção jogadoras

Imagem quadra

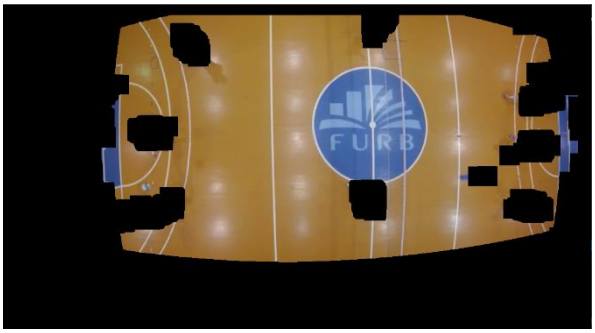


Imagem sem jogadoras

Fonte: elaborado pela autora.

Como o objetivo do protótipo é gerar estatística referente a um time, precisava-se dividir a quadra ao meio para que a análise fosse feita somente a partir de um dos lados da quadra. Como a parte esquerda da quadra é a mais reta, ela foi escolhida para realizar a divisão de forma automática.

Como a quadra não está centralizada na imagem é necessário descobrir o meio da quadra para realizar o recorte do lado desejado. Para isso, é necessário descobrir a linha que divide o meio da quadra.

Inicialmente realizou-se a conversão da imagem para a escala de cinza e posteriormente aplicou-se uma binarização para destacar as linhas que delimitam internamente as regiões da quadra (linha central e linha dos 3 metros), conforme mostra o trecho de código do Quadro 10.

Quadro 10 - Conversões na imagem

```
Mat img_bin, src_gray;
cvtColor(imgSemJogadoras, src_gray, CV_BGR2GRAY);
threshold(src_gray, img_bin, 190, 255, THRESH_BINARY);
```

Fonte: elaborado pela autora.

O método `cvtColor` foi utilizado para converter a imagem para escala de cinza. Posteriormente, utilizou-se o método `threshold` para binarizar a imagem. Os resultados dessas conversões estão apresentados na Figura 18.

Figura 18 - Imagem convertida

Imagens sem jogadoras em escala RGB

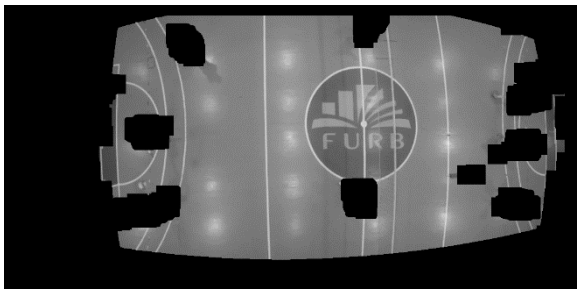


Imagem em escala de cinza

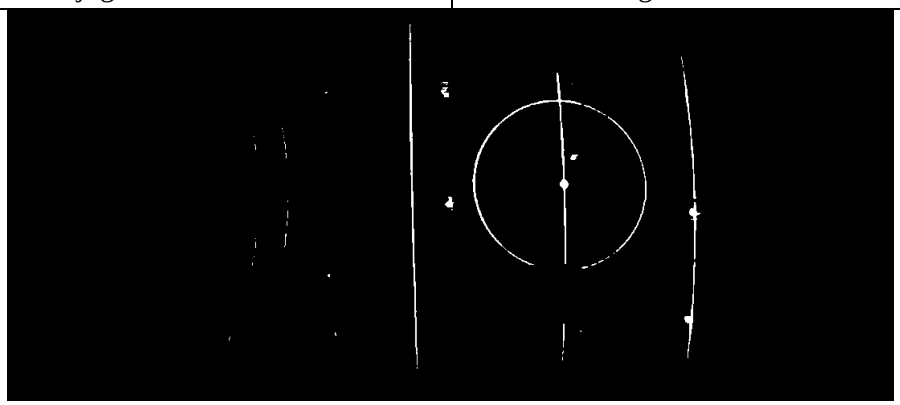


Imagem binarizada

Fonte: elaborado pela autora.

Com a imagem binarizada, realizou-se o procedimento de detecção de bordas. Essa etapa é necessária para identificar as linhas principais da quadra, conforme mostra o Quadro 11.

Quadro 11 - Detecção de bordas

```
Mat img_canny;
Canny(img_bin, img_canny, 50, 200, 3);
```

Fonte: elaborado pela autora.

Para detectar as bordas, utilizou-se o método `Canny`, sendo passado como parâmetro a imagem binarizada, a imagem de destino, os valores de limiarizações e tamanho da borda, o valor padrão é 3. Na Figura 19 é mostrado o resultado da aplicação do método `Canny`.

Figura 19 - Detecção de bordas

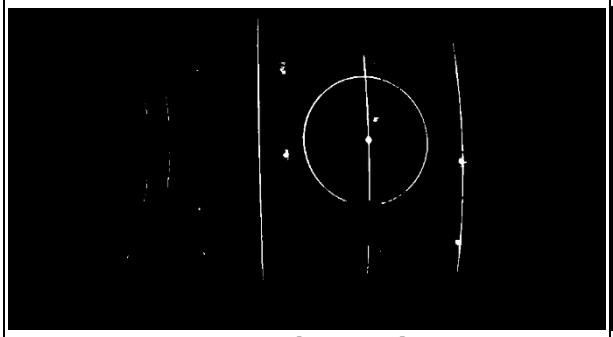


Imagem binarizada

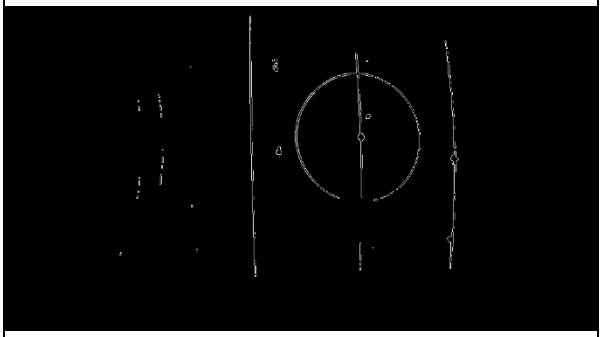


Imagem com bordas detectadas

Fonte: elaborado pela autora.

A partir da detecção das bordas é possível realizar o processamento para detecção das linhas da quadra. Para detectar as linhas foi utilizada a transformada de Hough, conforme mostra o trecho de código do Quadro 12.

Quadro 12 - Detecção das linhas

```
vector<Vec4i> lines;
HoughLinesP(img_canny, lines, 1, CV_PI / 180, 40, 10, 10);
for (size_t i = 0; i < lines.size(); i++){
   Vec4i l = lines[i];
   line(img_canny,Point(l[0], l[1]), Point(l[2], l[3]), Scalar(0,0,255), 8,LINE_AA);
}
```

Fonte: elaborado pela autora.

A partir da codificação acima, percebe-se que a imagem com as bordas detectadas é passada como parâmetro no método `HoughLinesP`, que armazena no vetor `lines`, todas as linhas encontradas na imagem. Logo em seguida é realizado um laço de repetição com o total de linhas encontradas para que sejam desenhas linhas na imagem de entrada através do método `line` com o tipo de linha `LINE_AA`. A Figura 20 mostra o resultado do processo de identificação de linhas via transformada de Hough.

Figura 20 - Detecção de linhas através da transforma de Hough

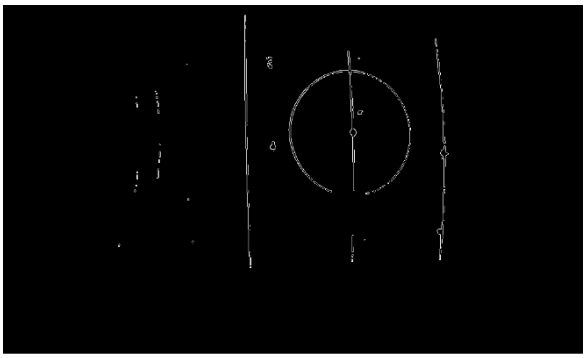


Imagem com bordas detectadas

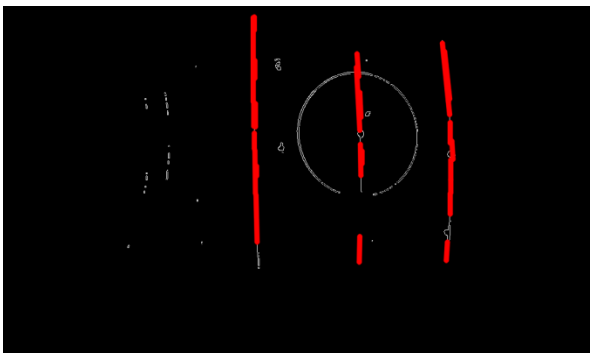


Imagem com linhas detectadas

Fonte: elaborado pela autora.

Depois de detectar as linhas, o próximo passo é a remoção de ruídos, deixando apenas as linhas principais da quadra. O Quadro 13 mostra o código que realiza a limpeza da imagem.

Quadro 13 - Remoção dos ruídos

```
erosion_size = 3;
Mat element1 = getStructuringElement(MORPH_RECT,
    Size(2 * erosion_size + 1, 2 * erosion_size + 1),
    Point(erosion_size, erosion_size));
Mat erode3;
erode(img_gray, erode3, element1);
```

Fonte: elaborado pela autora.

Para realizar a remoção dos ruídos foi utilizado o método `erode`. A partir de testes realizados, ficou estabelecido o valor 3 para a variável `erosion_size` que é utilizada como parâmetro para a função `getStructuringEmelent`. Ou seja, o operador morfológico de

erosão utilizou um elemento estruturante de tamanho 3x3. Na Figura 21 é possível ver o resultado dessa codificação.

Figura 21 - Remoção dos ruídos

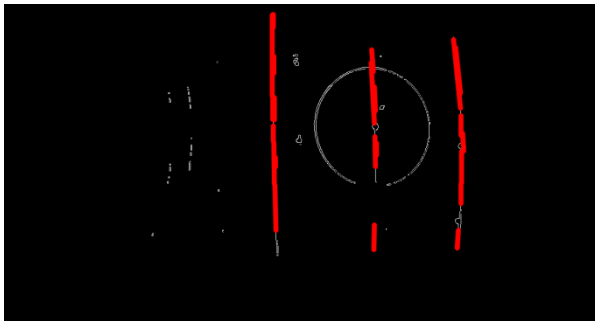


Imagem com linhas detectadas

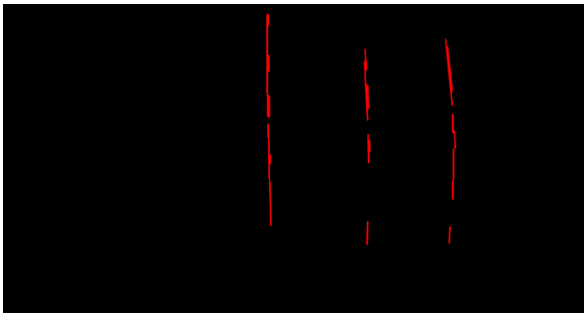


Imagem sem ruídos

Fonte: elaborado pela autora.

Tendo apenas as linhas principais, o próximo passo é convertê-las para branco. O Quadro 14 mostra a codificação da conversão.

Quadro 14 - Conversão para linhas brancas

```
Mat imgBin, imgGrey;
cvtColor(erode3, imgGrey, CV_BGR2GRAY);
threshold(imgGrey, imgBin, 75, 255, THRESH_BINARY);
```

Fonte: elaborado pela autora.

Para converter as linhas para branco, a imagem foi convertida para escada de cinza e posteriormente para binário através do processo de binarização. O resultado é mostrado na Figura 22.

Figura 22 - Imagem com linhas brancas

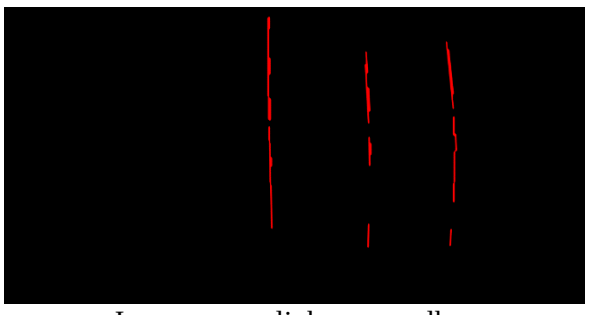


Imagem com linhas vermelhas

Imagem com linhas brancas

Fonte: elaborado pela autora.

A partir dos resultados obtidos e exemplificados na Figura 22, buscou-se desenvolver uma forma de descobrir qual era a linha do meio. Para isso, foi utilizado o cálculo da distância euclidiana onde o objetivo era encontrar a segunda linha a partir do lado esquerdo da imagem. O Quadro 15 mostra a codificação da estratégia utilizada para encontrar a linha central da quadra.

Quadro 15 - Detecção da linha do meio através da distância euclidiana

```
bool primeira = true, segunda = true;
int auxX, auxY, valorColunas = 21, contEuclidiana = -1 ;
float euclidina, matrizEuclidiana[21][2];
int linhas = (int)imgBin.rows / 20;
for (int a = 0; a < valorColunas; a++) {
    for (int b = 0; b < 2; b++) {
        matrizEuclidiana [a][b] = 0;
    }
}
for (int x = linhas; x < (imgBin.rows); x += linhas) {
    for (int y = 0; y < (imgBin.cols); y++) {
        if (imgBin.at<uchar>(x, y) == 255) {
            if (imgBin.at<uchar>(x, y - 1) != 255) {
                if (primeira) {
                    primeira = false;
                    auxX = x;
                    auxY = y;
                } else {
                  if (segunda) {
                      segunda = false;
                      euclidena = sqrt(pow(auxX - x, 2) + pow(auxY - y, 2));
                      contEuclidiana ++;
                      matrizEuclidiana [contEuclidiana][0] = y;
                      matrizEuclidiana [contEuclidiana][1] = euclidina;
                    }
                 }
              }
         }
    }
    primeira = true;
    segunda = true;
}
```

Fonte: elaborado pela autora.

Na codificação apresentada, a imagem é percorrida, mas para melhorar o desempenho, o laço de repetição não percorre todas as linhas, pulando em blocos. Basicamente o algoritmo percorre as colunas da imagem e, quando encontra um elemento com o valor 255, que significa a existência da cor branca, é verificado se o elemento anterior é branco, se for, não faz nada porque aquela coluna já foi processada. Se o elemento anterior não for branco, é verificado se é a primeira coluna, se for, é guardada a referência da linha e da coluna que ele se encontra. Se não for a primeira coluna, é verificado então se é a segunda, se for, é calculada a distância euclidiana com base na referência guardada linha e coluna da primeira coluna encontrada. Os valores da distância euclidiana e da referência da segunda coluna são armazenados em uma matriz que será utilizada na próxima etapa mostrada no Quadro 16.

Quadro 16 - Detecção da linha do meio

```
float auxA, auxB = 0;
for (int a = 0; a < valorColunas; a++) {
    for (int b = 0; b < valorColunas - 1; b++) {
        if ((matrizEuclidiana[b][1] > matrizEuclidiana[b + 1][1]) &&
                                (matrizEuclidiana[b + 1][1] != 0)) {
            auxA = matrizEuclidiana[b][0];
            auxB = matrizEuclidiana[b][1];
            matrizEuclidiana[b][0] = matrizEuclidiana[b + 1][0];
            matrizEuclidiana[b][1] = matrizEuclidiana[b + 1][1];
            matrizEuclidiana[b + 1][0] = auxA;
            matrizEuclidiana[b + 1][1] = auxB;
        }
    }
}
int coluna = matrizEuclidiana[2][0];
Mat metadeQuadra = imgSemJogadoras(Rect(coluna - 460, 0, 460, imgBin.rows)).clone();
```

Fonte: elaborado pela autora.

Nessa etapa, a matriz que armazena a distância euclidiana e a referência da sua coluna é ordenada, para que seja possível pré-determinar que o terceiro elemento da matriz é a coordenada da coluna do meio. Depois disso com o valor que representa a coluna do meio, é realizado o recorte da imagem utilizando o método `Rect`. Os parâmetros passados no `Rect` correspondem as coordenadas onde a imagem deve ser cortada. O resultado de procedimento pode ser visto na Figura 23.

Figura 23 - Imagem da quadra recortada ao meio

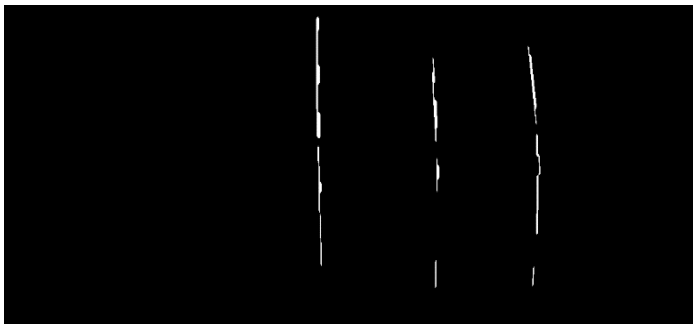


Imagem com as linhas da quadra

Imagem da quadra ao meio

Fonte: elaborado pela autora.

Para facilitar a etapa de subdivisão da quadra em quadrantes, as partes pretas da imagem, que não fazem parte da quadra serão removidas, deixando apenas a quadra na imagem. O trecho de código do Quadro 17 é responsável pela limpeza da imagem.

Quadro 17 - Remoção das partes pretas da imagem

```
int refLinha = 0;
for (int i = metadeQuadra.rows - 1; i >= 0; i--) {
    if (metadeQuadra.at<uchar>(i, metadeQuadra.cols) != 0) {
        refLinha = i;
        break;
    }
}

Mat quadraLimpa;
quadraLimpa = metadeQuadra(Rect(0, 0, metadeQuadra.cols, refLinha));
refLinha = 0;
for (int i = 0; i < quadraLimpa.rows; i++) {
    if (quadraLimpa.at<uchar>(i, metadeQuadra.cols) != 0) {
        refLinha = i;
        break;
    }
}
quadraLimpa=quadraLimpa(Rect(0,refLinha,quadraLimpa.cols,quadraLimpa.rows-refLinha));
```

Fonte: elaborado pela autora.

Para remoção das partes pretas, primeiramente é realizado um laço de repetição que lê a imagem de baixo a cima, ao encontrar um elemento diferente de preto, é guardado a linha como referência para aplicação do método `Rect`, que irá remover a parte preta debaixo da quadra. Depois disso, a imagem é lida de cima a baixo, quando encontrado um elemento diferente de zero, é guardada a linha como referência para aplicação de outro `Rect`, que irá remover a parte preta de cima da quadra. A Figura 24 mostra o resultado deste procedimento.

Figura 24 - Remoção das partes pretas da imagem

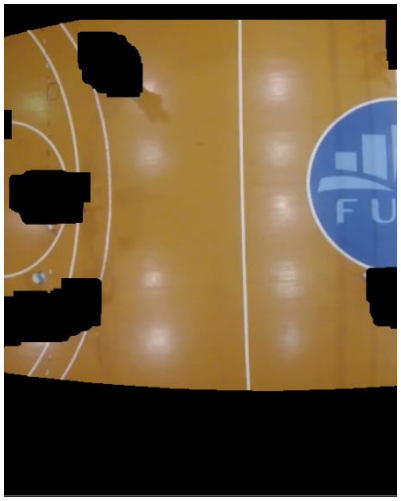


Imagem da quadra ao meio

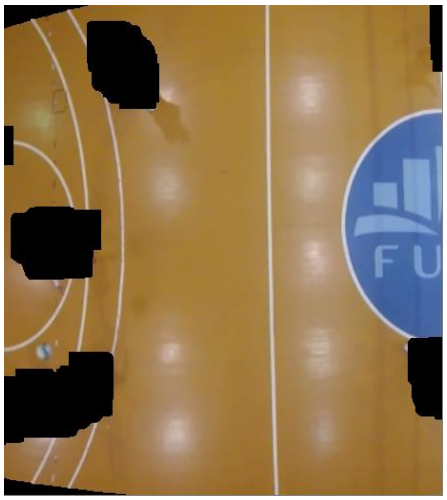


Imagem sem partes pretas

Fonte: elaborado pela autora.

3.2.3.3 Identificação da bola

Com a área da quadra estabelecida e sem as jogadoras, o próximo passo é realizar o processamento para encontrar a bola. Para isso, a imagem é convertida para HSV. A partir desta conversão, percebeu-se que a bola se destaca no canal H (matriz). O Quadro 18 mostra a codificação para realização da conversão da imagem para HSV.

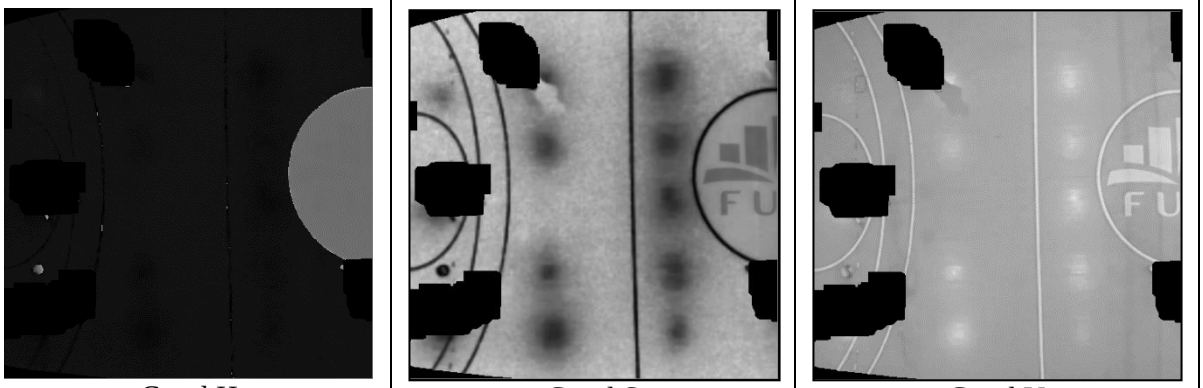
Quadro 18 - Conversão meia quadra para HSV

```
Mat hsv2;
cvtColor(quadraLimpa, hsv2, CV_BGR2HSV);
vector<Mat> canal2;
split(hsv2, canal2);
```

Fonte: elaborado pela autora.

Conforme já visto anteriormente, para converter a imagem para HSV é utilizado o método `split`. O vetor `canal2` contém todos os canais da imagem separados, sendo exibidos na Figura 25.

Figura 25 - Separação dos canais HSV (Meia quadra)

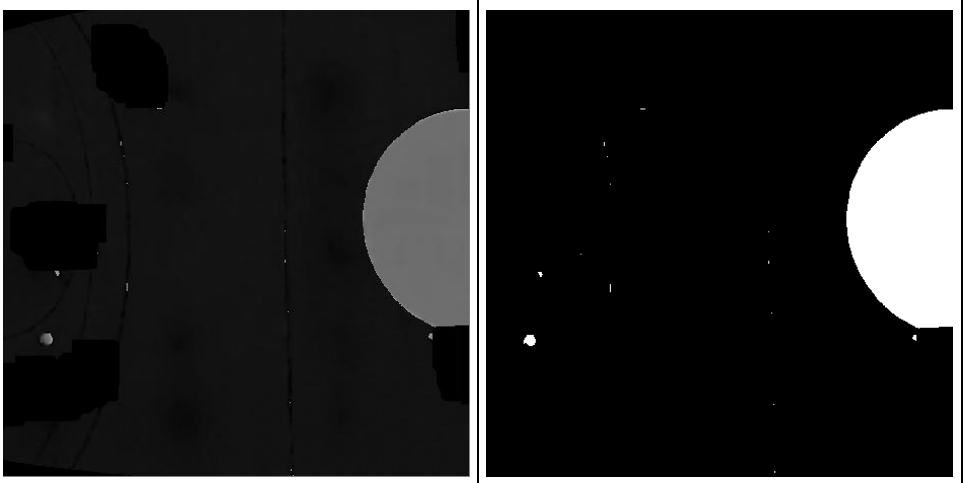


Canal H | Canal S | Canal V

Fonte: elaborado pela autora.

A partir da Figura 25 é possível observar que no canal S a bola também se destaca, mas não foi utilizado esse canal para o próximo passo porque os reflexos das luzes na quadra também ficam destacadas, podendo ser confundidos com a bola. Entretanto existem alguns elementos que precisam ser removidos, pois não representam uma bola. Sendo assim, inicialmente a imagem foi convertida para binário, conforme exemplifica a Figura 26.

Figura 26 - Resultado binarização (meia quadra)



Canal H | Imagem binarizada

Fonte: elaborado pela autora.

Como é possível observar na Figura 26, a imagem binarizada ficou com alguns ruídos que precisam ser retirados. Para isso, foram utilizados os métodos `erode` e `dilate`, conforme mostra o Quadro 19.

Quadro 19 - Remoção dos ruídos (meia quadra)

```
Mat erode4, dilate4;
int erosion_size2 = 2;
Mat elementX2 = getStructuringElement(MORPH_RECT,
                Size(2 * erosion_size2 + 1, 2 * erosion_size2 + 1),
                Point(erosion_size2, erosion_size2));

int dilate_size = 5;
Mat elementX3 = getStructuringElement(MORPH_RECT,
                Size(2 * dilate_size + 1, 2 * dilate_size + 1),
                Point(dilate_size, dilate_size));

erode(bin2, erode4, elementX2);
dilate(erode4, dilate4, elementX3, Point(-1, -1), 1);
```

Fonte: elaborado pela autora.

Com a aplicação do filtro morfológico de erosão, a bola tem seu tamanho reduzido. Para retorná-lo, é aplicada uma dilatação em seguida. Os resultados desse processo são mostrados na Figura 27.

Figura 27 - Aplicação erosão e dilatação (meia quadra)

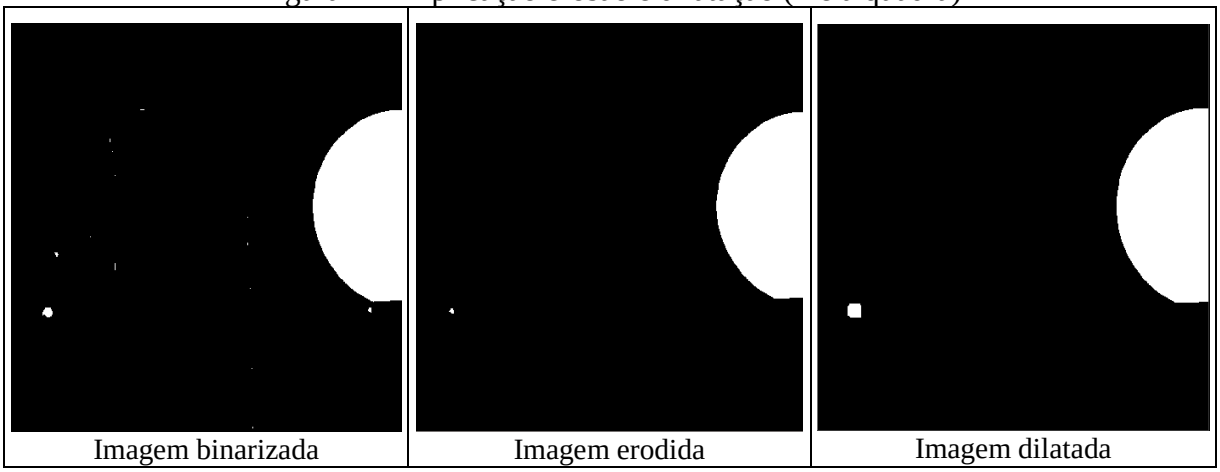


Imagem binarizada | Imagem erodida | Imagem dilatada

Fonte: elaborado pela autora.

Observando a imagem dilatada da Figura 27, percebe-se a existência de dois elementos brancos, um que é a bola efetivamente e o outro que é o centro da quadra. Para descobrir qual dos elementos é a bola e qual é o centro da quadra, identificou-se os contornos e a partir deles, têm-se os tamanhos dos objetos, conforme pode ser visto no Quadro 20.

Quadro 20 - Desenho do contornos

```
Mat canny_output;
vector<vector<Point> > contours;
vector<Vec4i> hierarchy;
RNG rng(12345);

Canny(dilate4, canny_output, 100, 100 * 2, 3);
findContours(canny_output, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE,
Point(0, 0));
Mat drawing = Mat::zeros(canny_output.size(), CV_8UC3);
Point2f center;
float radius = 0;
bool achouBola = false;
for (int i = 0; i < contours.size(); i++) {
    if (contourArea(contours[i]) > 110) {
        Scalar color = Scalar(rng.uniform(0, 255), rng.uniform(0, 255), rng.uniform(0,
255));
        drawContours(drawing, contours, i, color, 2, 8, hierarchy, 0, Point());
        minEnclosingCircle(contours[i], center, radius);
        circle(drawing, center, (int)radius, Scalar(0, 255, 0), 2, 8, 0);
        achouBola = true;
    }
}
```

Fonte: elaborado pela autora.

Inicialmente, determina-se as bordas dos objetos existentes na imagem através do filtro de `Canny` e posteriormente utiliza-se o método `findContours` para estabelecer os contornos. Mediante aos testes realizados, constatou-se que a bola sempre é o maior elemento na imagem. Isso acontece, pois, o centro da quadra encontra no limite da imagem e ao qual, esta parte não se caracteriza e não é considerada como parte do objeto. Sabendo que a bola é o maior objeto e tendo o tamanho de todos os elementos da imagem, realizou-se uma seleção deixando apenas os objetos maiores que 110, conforme mostra a Figura 28.

Figura 28 - Detecção bola

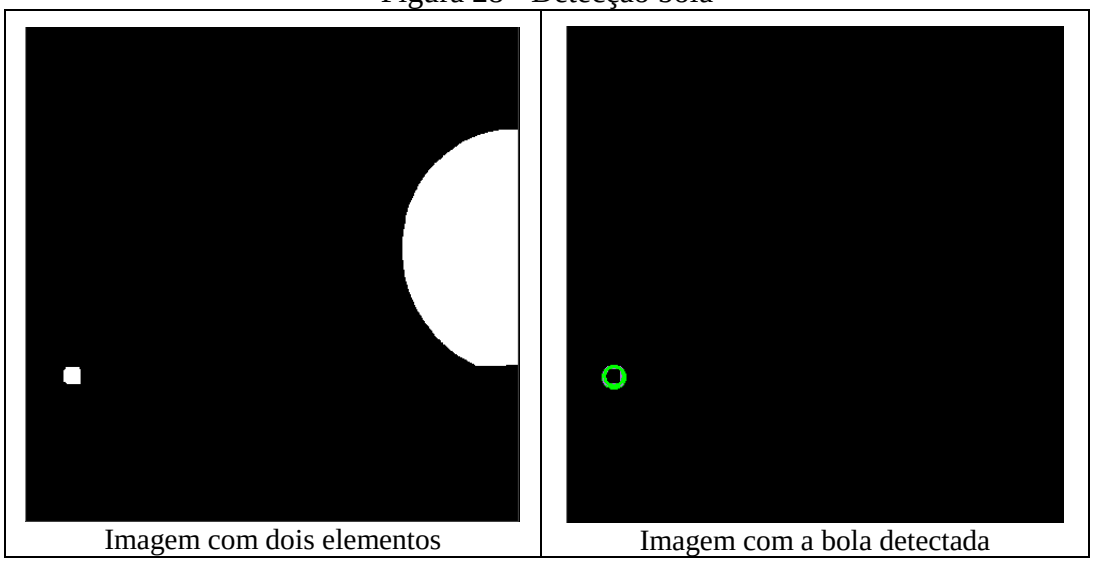


Imagem com dois elementos | Imagem com a bola detectada

Fonte: elaborado pela autora.

3.2.3.4 Subdivisão da quadra em quadrantes para geração das estatísticas

Sabendo qual objeto representa a bola, é necessário descobrir em qual parte da quadra ela está posicionada. Para isso, a quadra foi dividida em nove quadrantes do mesmo tamanho. E, utilizando as coordenadas da bola, é verificando se ela está posicionada entre as coordenadas que correspondem a um determinado quadrante. O Quadro 21 mostra a codificação para encontrar em qual quadrante a bola está posicionada.

Quadro 21 - Detecção de qual quadrante a bola está

```
float xColunas, xLinhas;
xColunas = drawing.cols / 3;
xLinhas = drawing.rows / 3;
float xColunasAux = xColunas;
float xLinhasAux = xLinhas;

if (achouBola) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if ((center.x >= (j*xColunas)) && (center.x <= (xColunasAux))) {
                if ((center.y >= (i*xLinhas)) && (center.y <= (xLinhasAux))) {
                    matrizPontuacao[j][i] = matrizPontuacao[j][i] + 1;
                    qtdBolasEncontradas++;
                    break;
                }
            }
            xColunasAux += xColunas;
        }
        xColunasAux = xColunas;
        xLinhasAux += xLinhas;
    }

}
```

Fonte: elaborado pela autora.

Tendo a localização da bola, a imagem é percorrida por blocos que são estabelecidos com o valor do total de linhas e colunas divididos por três, totalizando nove regiões. Para cada bloco percorrido é verificado se a bola está contida nele, caso esteja, é armazenado esse valor para geração da estatística. Para melhor visualização das regiões definidas na quadra, são desenhadas linhas que demonstram os nove quadrantes. O Quadro 22 mostra o trecho de código responsável pela subdivisão da quadra em nove quadrantes.

Quadro 22 - Desenho das linhas divisórias

```
line(quadraLimpa, Point(0, xLinhas), Point(quadraLimpa.cols, xLinhas), Scalar(0, 255, 0),
1, CV_AA);
line(quadraLimpa, Point(0, xLinhas + xLinhas), Point(quadraLimpa.cols, xLinhas + xLinhas),
Scalar(0, 255, 0), 1, CV_AA);
line(quadraLimpa, Point(xColunas, 0), Point(xColunas, quadraLimpa.rows), Scalar(0, 255,
0), 1, CV_AA);
line(quadraLimpa, Point(xColunas + xColunas, 0), Point(xColunas + xColunas,
quadraLimpa.rows), Scalar(0, 255, 0), 1, CV_AA);
```

Fonte: elaborado pela autora.

O método `line` desenha as linhas conforme a coordenadas indicadas nos parâmetros. Para desenhar as quatro linhas que dividem os quadrantes em tamanhos iguais, é chamado quatro vezes o método `line`. A Figura 29 mostra o desenho das linhas divisórias.

Figura 29 - Linhas divisórias

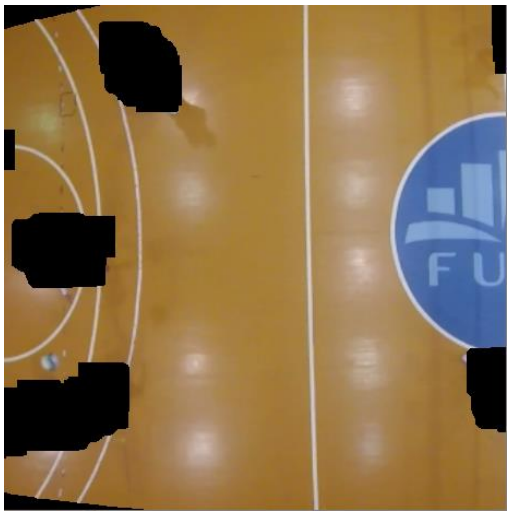


Imagem meia quadra

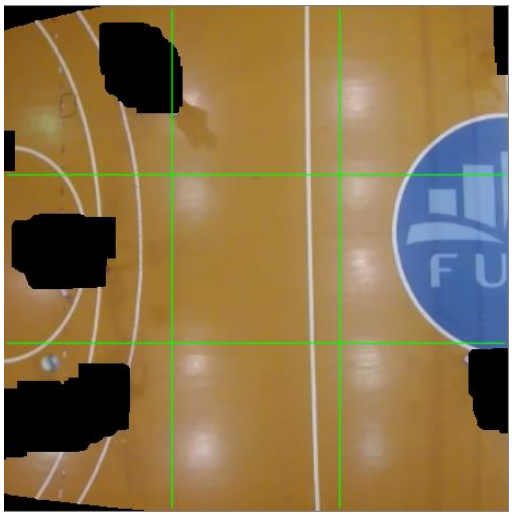


Imagem com linhas divisórias

Fonte: elaborado pela autora.

Com o resultado das outras etapas descritas acima, é possível estabelecer estatísticas de quantas vezes a bola cai em cada quadrante. Para geração dessas estatísticas, no processo que encontra em qual quadrante a bola está localizada, é somado um ponto para o quadrante que possui a bola. Ao final, tem-se o total de quantas bolas caíram em cada quadrante. O Quadro 23 mostra o trecho de código responsável por achar o quadrante que tem mais bolas atacadas.

Quadro 23 - Quadrante com mais bolas

```
int maiorValor = 0;
int xline, xcol = 0;
for (int x = 0; x < 3; x++) {
    for (int y = 0; y < 3; y++) {
        if (maiorValor < matrizPontuacao[x][y]) {
            maiorValor = matrizPontuacao[x][y];
            xline = x;
            xcol = y;
        }
    }
}
```

Fonte: elaborado pela autora.

Para descobrir qual é o quadrante com maior pontuação de bolas aterrissadas sobre ele, é percorrido quadrante por quadrante, comparando a pontuação de cada quadrante com os demais. Definindo assim, qual é o quadrante que teve mais bolas aterrissadas sobre ele.

Visualmente para mostrar ao usuário quanto cada bola aterrissa em um quadrante, é realizado o cálculo de porcentagem e posteriormente, ele é apresentado por quadrante da quadra, conforme mostra o trecho de código do Quadro 24.

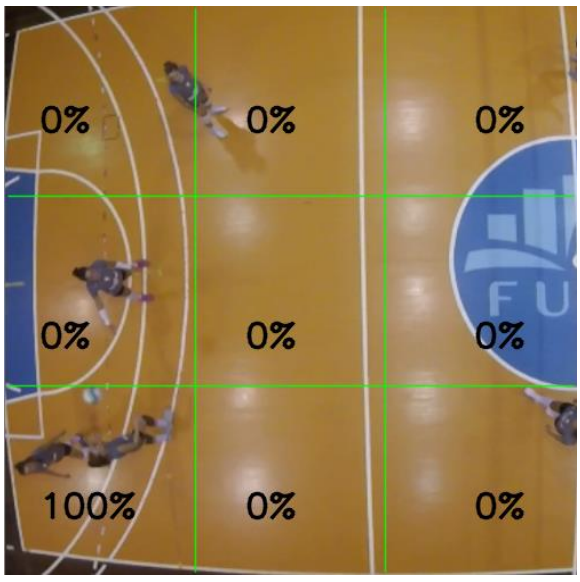
Quadro 24 - Calculo porcentagem

```
string matrizPorcentagem[3][3];
for (int x = 0; x < 3; x++) {
    for (int y = 0; y < 3; y++) {
        float porcentagem = ((float)matrizPontuacao[x][y] / (float)qtdImagens) * 100;
        std::stringstream sporcentagem;
        sporcentagem << porcentagem;
        matrizPorcentagem[x][y] = sporcentagem.str();
        cout << matrizPorcentagem[x][y] << endl;
    }

}

int font = FONT_HERSHEY_SIMPLEX;

putText(img_result, matrizPorcentagem[0][0] + "%", Point(30, 120), font, 1, (255, 0, 0),
2, LINE_AA);
putText(img_result, matrizPorcentagem[1][0] + "%", Point(220, 120), font, 1, (255, 0, 0),
2, LINE_AA);
putText(img_result, matrizPorcentagem[2][0] + "%", Point(420, 120), font, 1, (255, 0, 0),
2, LINE_AA);
putText(img_result, matrizPorcentagem[0][1] + "%", Point(30, 320), font, 1, (255, 0, 0),
2, LINE_AA);
putText(img_result, matrizPorcentagem[1][1] + "%", Point(220, 320), font, 1, (255, 0, 0),
2, LINE_AA);
putText(img_result, matrizPorcentagem[2][1] + "%", Point(420, 320), font, 1, (255, 0, 0),
2, LINE_AA);
putText(img_result, matrizPorcentagem[0][2] + "%", Point(30, 500), font, 1, (255, 0, 0),
2, LINE_AA);
putText(img_result, matrizPorcentagem[1][2] + "%", Point(220, 500), font, 1, (255, 0, 0),
2, LINE_AA);
putText(img_result, matrizPorcentagem[2][2] + "%", Point(420, 500), font, 1, (255, 0, 0),
2, LINE_AA);
```

Fonte: elaborado pela autora.

Para mostrar os resultados obtidos para o usuário, os valores são desenhados na imagem da quadra em seus respectivos quadrantes. O método `putText` é utilizado para exibir o percentual obtido, conforme mostra a Figura 30.

Figura 30 - Resultado geração estatística



Fonte: elaborado pela autora.

3.3 ANÁLISE DOS RESULTADOS

Este capítulo apresenta os resultados obtidos na execução do protótipo. Inicialmente, foi necessário a construção de uma base de imagens. Elas foram obtidas de vídeos feitos em um treino do time feminino do Blu Vôlei no ginásio da FURB. Para captura das filmagens foi necessário criar um mecanismo para pendurar uma câmera GOPRO no teto do ginásio, de forma que ela ficasse filmando a extensão da quadra. Para obter as imagens dos vídeos, foi utilizada a ferramenta Movavi Video Editor em sua versão 14. Como os vídeos foram feitos em um treino, era comum a existência de mais de uma bola na quadra. Por fim, foram selecionadas apenas imagens que continham uma bola na quadra. No total foram adquiridas 45 imagens, sendo 32 com a bola no lado esquerdo e as outras 13 posicionadas no lado direito da quadra. O banco de imagens pode ser visualizado no Apêndice A.

É importante ressaltar que aqui não serão demonstrados os resultados alcançados nas etapas de identificação e subdivisão da quadra, pois dentre as imagens que formam o banco de imagens ambas as etapas foram realizadas corretamente. Sendo assim, o primeiro teste teve como objetivo verificar a eficiência do protótipo em localizar a bola na quadra. A Tabela 2 descreve os resultados obtidos. Nela, tem-se a quantidade de imagens utilizadas, a quantidade bolas detectadas, quantidade bolas não detectadas e o percentual de acurácia do protótipo.

Tabela 2 - Resultados das detecções da bola no lado esquerdo da quadra

Quantidade de imagens processadas	**Quantidade de imagens com a bola detectada**	**Quantidade de imagens com a bola não detectada**	**Percentual de bolas encontradas**
32 imagens	30	2	93,75%

Fonte: elaborado pela autora.

A partir da Tabela 2, observa-se que em duas imagens a bola não foi detectada. Esta situação ocorre nas imagens 10 e 30 do Apêndice A pois a bola acaba tendo a mesma tonalidade do círculo azul existente no centro da quadra. A Figura 31 mostra um exemplo da imagem original e outro da imagem utilizada para identificar a bola.

Figura 31 - Bola sobre o círculo azul no centro da quadra

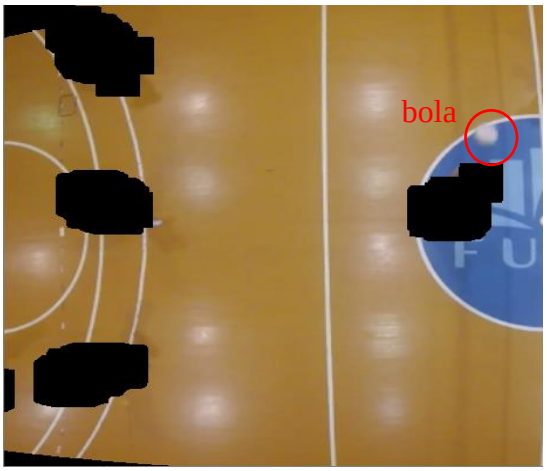


Imagem meia quadra | Imagem no canal H

Fonte: elaborado pela autora.

Depois da separação dos canais HSV é feito a binarização da imagem, como o círculo do centro da quadra e a bola ficam com cores muito parecidas no canal H, virando um elemento único, conforme mostra a Figura 32.

Figura 32 - Imagem com um único elemento

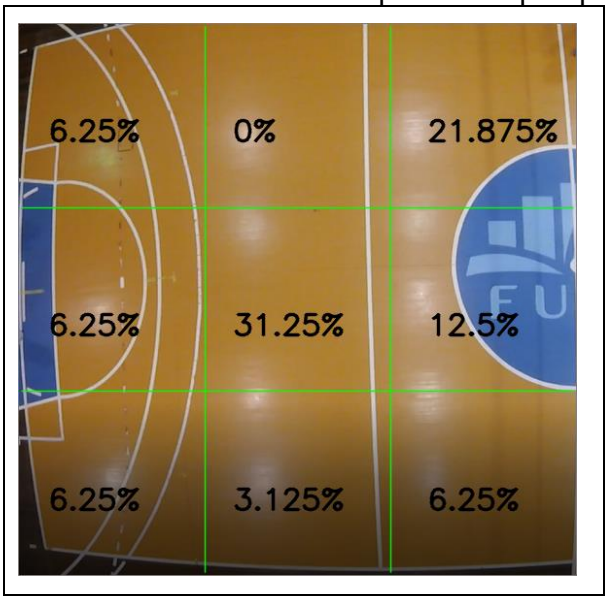
Imagem canal H | Imagem binarizada

Fonte: elaborado pela autora.

Como é possível ver na Figura 32, a imagem binarizada só possui um elemento, impossibilitando a separação da bola e do círculo do centro da quadra. Conclui-se que para esses casos seria necessário adotar outra estratégia de detecção da bola.

O processo de geração de estatísticas e a indicação do percentual de vezes que a bola caia em determinada região da quadra obteve bons resultados. Das 30 imagens em que a bola foi encontrada, o protótipo identificou corretamente todos os quadrantes em que as bolas estavam posicionadas. É importante destacar que para efeito de cálculo de porcentagem de cada quadrante, foram levadas em consideração as 32 imagens da base de dados. A Figura 33 mostra o resultado final da aplicação.

Figura 33 - Resultado do cálculo do percentual por quadrante



Fonte: elaborado pela autora.

O segundo teste teve como objetivo verificar a eficiência do protótipo em localizar a bola no lado direito da quadra. Nele, foram utilizadas 13 imagens. O protótipo funcionou corretamente até a parte de dividir a quadra ao meio. Porém, a etapa de detecção da bola não funcionou corretamente, identificando vários objetos como se fossem a bola, conforme mostra a Figura 34.

Figura 34 - Resultado obtido ao processar o lado direito da quadra

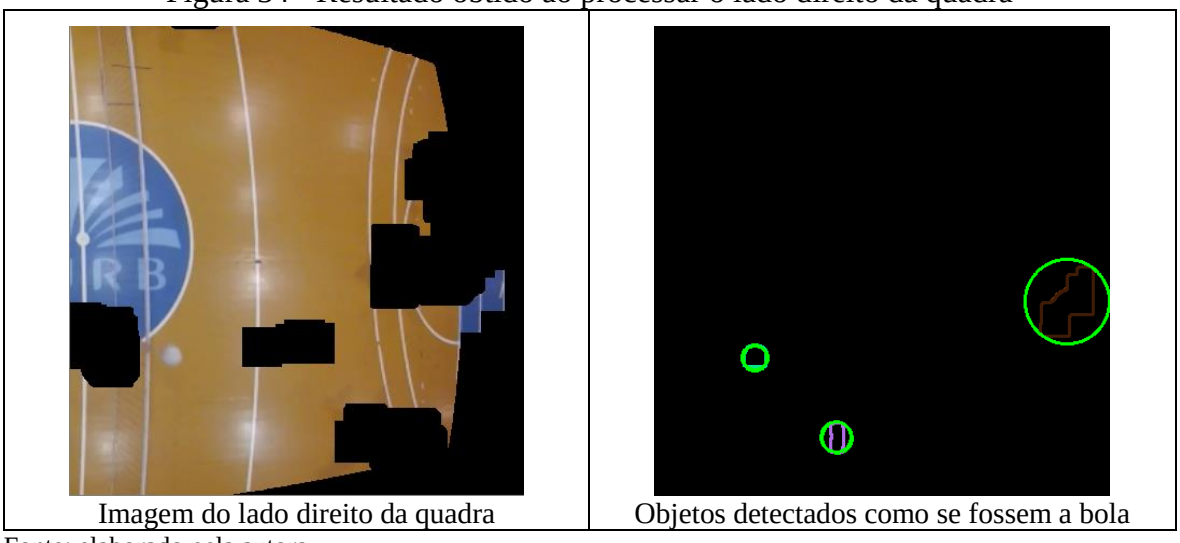


Imagem do lado direito da quadra | Objetos detectados como se fossem a bola

Fonte: elaborado pela autora.

Acredita-se que isso tenha acontecido devido a intensidade do reflexo das luzes na quadra ou por causa da angulação das imagens capturadas. Conclui-se que os procedimentos adotados precisam de ajustes ou fazem-se necessário a inclusão de novas etapas a fim de atingir resultados satisfatórios.

4 CONCLUSÕES

A proposta desse trabalho era desenvolver um protótipo para geração de estatística de pontuação em jogo de vôlei para ajudar os técnicos na análise da partida, tendo por objetivo saber qual região da quadra a bola mais aterrissa. Para criação do protótipo foi utilizada a linguagem C++ com apoio da biblioteca OpenCV para facilitar o trabalho de processamento das imagens.

As imagens foram obtidas de filmagens feitas de um treino do Blu Vôlei no ginásio da FURB. Para seleção das imagens, foi observado quando a bola estava sobre o chão, exemplificando uma pontuação. Como as filmagens foram feitas de um treino, foi possível retirar somente 45 imagens que realmente demonstrassem o resultado de um ponto sem interferências, como por exemplo, mais de uma bola em quadra. Observando as imagens, constatou-se que o lado esquerdo da quadra estava melhor posicionado. Isso aconteceu porque não foi possível colocar a câmera totalmente no centro da quadra para realizar as filmagens, favorecendo mais um lado da quadra. A partir disso, optou-se por utilizar o lado esquerdo da quadra no processamento de imagens, mas no final, também foram realizados testes considerando o lado direito da quadra.

O primeiro objetivo do protótipo era delimitar a quadra na imagem, sendo alcançado. Para isso, foram realizados diversos processamentos utilizando o canal V do modelo HSV pois a quadra se diferenciava em relação a área externa da quadra pela cor. Para que a delimitação da quadra acontecesse de forma perfeita, utilizou-se filtros de erosão e a dilatação para eliminar alguns ruídos.

Para contemplar o segundo objetivo do trabalho, foram necessárias realizar algumas etapas de processamentos antes de efetivamente encontrar a bola na quadra. Uma das etapas realizadas foi à remoção das jogadoras, para não existir possibilidade do protótipo confundir uma jogadora com a bola. Essa etapa foi fundamental, pois sem sujeira na quadra, o trabalho de encontrar a bola foi mais eficaz. A outra etapa era determinar o lado da quadra que seria utilizado nas demais etapas. Para isso, sabendo que o lado esquerdo da quadra estava melhor posicionado na imagem, foram realizados processamentos para identificar o centro da quadra e depois disso, foi realizado o recorte para ficar apenas o lado esquerdo da quadra na imagem. A partir dessas duas etapas, era possível iniciar o processamento para encontrar a bola.

Com as etapas de remoção das jogadoras e recorte da quadra concluídos, a etapa de encontrar a bola ficou facilitada. Para isso, a imagem foi convertida para o modelo HSV, sendo utilizado o canal H, pois ele destacou a bola sem a influência das luzes que refletiam na

quadra. Depois disso, a imagem foi convertida para binário e os algoritmos de erosão e dilatação aplicados para remover ruídos. Como resultado, sobrou apenas a bola e o círculo do centro da quadra, sendo diferenciados pelo seu tamanho. É importante ressaltar que a bola só não foi identificada quando ela estava posicionada em cima do círculo do meio da quadra pois caracterizava-se como um único elemento.

Para atingir o terceiro objetivo, o protótipo precisava saber em qual região da quadra a bola estava posicionada. Para isso, o lado esquerdo da quadra foi dívida em nove quadrantes de tamanhos iguais. Posteriormente, foram utilizadas as coordenadas da posição da bola para descobrir em qual quadrante ela se encontrava. Das 32 imagens processadas, em apenas duas não foi possível determinar em qual quadrante a bola caiu mais vezes. Os resultados são mostrados para o usuário dentro dos quadrantes estabelecidos na forma de percentagem.

Por fim, conclui-se que o protótipo teve resultados satisfatórios nos processamentos realizados para o lado esquerdo da quadra, onde dos três objetivos propostos, somente a detecção da bola não teve 100% de acerto. Já em relação aos resultados do lado direito, acredita-se que neste caso, o posicionamento da câmera para captura influenciou negativamente. Também é possível concluir que esse trabalho pode ser utilizado como base para desenvolvimento de vários protótipos que gerem estatísticas de outros problemas contidos em um jogo de vôlei.

4.1 EXTENSÕES

Abaixo estão listadas algumas sugestões de extensões para o protótipo:

a) realizar filmagens da quadra posicionando a câmera no centro da quadra;
b) realizar o processamento para lado direito da quadra pois o protótipo desenvolvido só realiza o processamento para o lado esquerdo da quadra;
c) encontrar a bola quando ela está posicionada no círculo do centro da quadra, porque não foi possível identificar a bola nesses casos;
d) realizar o processamento em tempo real para tomadas de ações imediatas;
e) criar uma interface para facilitar a interação com o usuário.

REFERÊNCIAS

ALVES, Bruno V. R. **Sistema de Observação e Registo do Desempenho Táctico-Técnico em Jogos Desportivos Colectivos**. 2012. 87 f. Dissertação (Mestrado Integrado em Engenharia Informática e Computação) - Faculdade de Engenharia da Computação, Universidade do Porto, Portugal, 2012.

BBALL STATS. **BBall Stats**. 1992. Disponível em: <http://www.bballstats.net/>. Acesso em: 12 mar. 2017.

DATA VOLLEY. Data Volley. 2007. Disponível em: < http://www.dataproject.com/Products/EN/en/Volleyball/DataVolley>. Acesso em: 10 jul. 2018.

FACON, Jacques. **Morfologia matemática:** teoria e exemplos. Curitiba: [Champagnat], 1996.XV, 304 p.

MARQUES, Ogê; VIEIRA, Hugo. **Processamento Digital de Imagens**. Rio de Janeiro: Brasport, 1999. 331 p.

GONZALEZ, Rafael C; WOODS, Richard E. (Richard Eugene). **Processamento de imagens digitais**. São Paulo: Edgard Blucher, 2000. 509 p.

LOTUFO, João Nogueira. **Voleibol**: história, técnica, prática, regras, seleção de estudos de vários autores, súmulas. São Paulo: Cia. Brasil, 1976. 134 p.

PESSOA, André Eduardo; BERTOLLO, Mauro; CARLAN, Paulo. **Voleibol**. Ijuí: Ed. Unijuí, 2009. 144 p.

OKAZAKI, Vitor H. A et al. Ciência e tecnologia aplicada à Melhoria do desempenho esportivo. Revista Mackenzie de Educação Física e Esporte, [S.l], v. 11, n. 1, 2012. Disponível em: <http://editorarevistas.mackenzie.br/index.php/remef/article/viewFile/3451/3471>. Acesso em: 29 jun. 2018.

PIVETTA, Cleber; MANTOVANI, Gustavo; ZOTTIS, Felipe. **Transformada de Hough**. Cascavel: Universidade Estadual do Oeste do Paraná – UNOESC, 2018. 24 slides, color. Disponível em: <http://www.inf.unioeste.br/~adair/PID/Notas%20Aula/Transformada%20de%20Hough.pdf> Acesso em: 19 jul. 2018.

PROZONE. **Prozone**. 1995. Disponível em: <http://prozonesports.stats.com/>. Acesso em: 13 mar. 2017.

REZENDE, Bernardo Rocha. **Transformando Suor em Ouro**. Rio de Janeiro: Sextante, 2006. 187 p.

RIBEIRO, Jorge L. S. **Conhecendo o voleibol**. Rio de Janeiro: Sprint, 2004. 173 p.

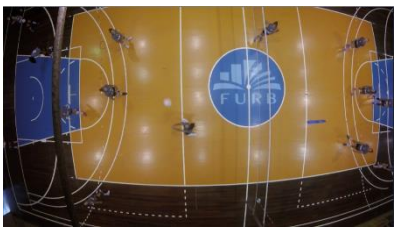
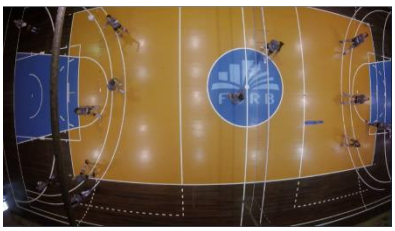
SHONDELL, Donald S; REYNAUD, Cecile. **A bíblia do treinador de voleibol**. Porto Alegre: Artmed, 2005. 352 p.

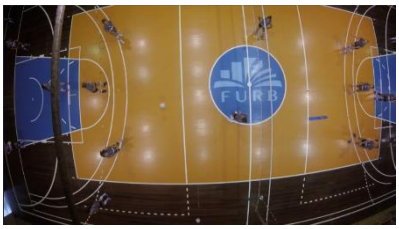
TROFINO, André F. N. **Implementação de sistema robótico autônomo movimentado de acordo com informações visuais**. 2014. 95 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia da Computação) - Faculdade de Engenharia de Computação, Escola de Engenharia de São Carlos da Universidade de São Paulo, São Paulo, 2014.

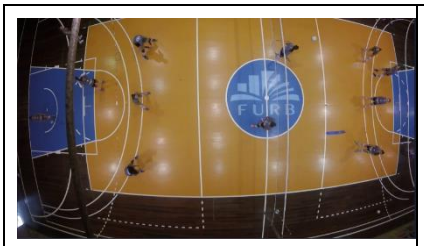
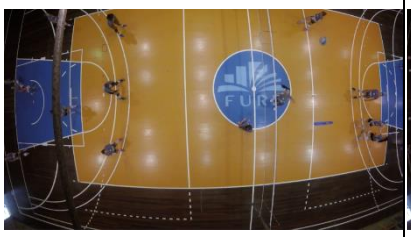
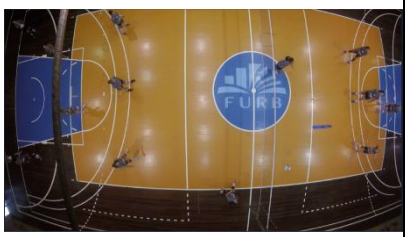
APÊNDICE A – Base de imagens

O Quadro 25 apresentada as imagens que compõem a base de dados utilizada nesse trabalho. São 32 imagens onde a bola encontra-se no lado esquerdo (E) e 13 no lado direito (D).

Quadro 25 - Base de imagens

 01E	 02E	 03E
 04E	 05E	 06E
 07E	 08E	 09E
 10E	 11E	 12E
 13E	 14E	 15E

 16E	 17E	 18E
 19E	 20E	 21E
 22E	 23E	 24E
 25E	 26E	 27E
 28E	 29E	 30E
 31E	 32E	 01D

 02D	 03D	 04D
 05D	 06D	 07D
 08D	 09D	 10D
 11D	 12D	 13D

Fonte: elaborado pela autora.