Sistema de Visão Computacional para Análise de Jogadas em Goalball

Adriel dos Santos Araujo Cabral

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2024

Adriel dos Santos Araujo Cabral

Sistema de Visão Computacional para Análise de Jogadas em Goalball

Monografia apresentada ao curso Ciência da Computação
do Centro de Informática, da Universidade Federal da Paraíba,
como requisito para a obtenção do grau de Bacharel em título

Orientador: Prof. Dr. Leonardo Vidal Batista

Novembro de 2024

Catalogação na publicação
Seção de Catalogação e Classificação

C117s Cabral, Adriel Dos Santos Araujo.
Sistema de visão computacional para análise de jogadas em goalball / Adriel Dos Santos Araujo Cabral. - João Pessoa, 2024.
51 f. : il.

Orientação: Leonardo Vidal Batista Batista.
TCC (Graduação) - UFPB/CI.

1. Visão computacional. 2. Goalball. 3. Aprendizagem de máquina. 4. Reconhecimento de ações. 5. Inteligência artificial. I. Batista, Leonardo Vidal Batista. II. Título.

UFPB/CI CDU 004.8

Elaborado por Michelle de Kássia Fonseca Barbosa - CRB-738

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Ciência da Computação intitulado ***Sistema de Visão Computacional para Análise de Jogadas em Goalball*** de autoria de Adriel dos Santos Araujo Cabral, aprovado pela banca examinadora constituída pelos seguintes professores:

Prof. Dr. Leonardo Vidal Batista
Universidade Federal da Paraíba - UFPB

Prof. Dra. Thaís Gaudencio do Rêgo
Universidade Federal da Paraíba - UFPB

Prof. Dr. Yuri De Almeida Malheiros Barbosa
Universidade Federal da Paraíba - UFPB

João Pessoa, 8 de novembro de 2024

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

AGRADECIMENTOS

Gostaria de expressar minha gratidão à minha família, cujo apoio e encorajamento foram fundamentais para que eu seguisse adiante, em cada etapa da minha jornada no curso. Agradeço também à TAIL, pois foi através da liga acadêmica que tive a oportunidade de conhecer professores dedicados que me apoiaram e orientaram durante a minha trajetória acadêmica. Em especial, agradeço aos professores Leonardo, Thais, Yuri e Marcelo pelo comprometimento e pelas valiosas orientações, que ampliaram minha visão e enriqueceram meu aprendizado.

Meus sinceros agradecimentos vão também aos meus amigos Ryan e Raphael, que estiveram ao meu lado desde o ensino médio, sempre incentivando e compartilhando momentos importantes. Também agradeço a Daniel, Cassiano e Carol pela companhia constante e pelas ajudas indispensáveis ao longo do curso.

Por fim, devo agradecer a todas as pessoas que, de alguma forma, contribuíram para o desenvolvimento desse trabalho. Seja por meio de palavras de incentivo, ensinamentos compartilhados ou pelo simples fato de estarem presentes ao longo dessa jornada, cada um deixou uma marca especial que me ajudou a chegar até aqui.

RESUMO

As tecnologias de visão computacional e aprendizado de máquina vêm transformando a forma como os esportes são analisados. Com essas ferramentas, é possível obter insights mais precisos sobre as ações dos atletas, otimizando estratégias e aprimorando o entendimento das dinâmicas de jogo. Dessa forma, este trabalho propõe um sistema para análise de jogadas em partidas de *goalball*, utilizando técnicas de visão computacional e aprendizado de máquina para reconhecer ações dos jogadores, detectar quiques da bola e calcular sua velocidade média. A abordagem metodológica incluiu a criação de um conjunto de dados próprio, uso do modelo Yolov8 para detectar os jogadores e a bola, modelos de redes neurais convolucionais (RNC) e *Long Short-Term Memory* (LSTM) para reconhecimento de ações, além de algoritmos de aprendizado de máquina para detectar quiques na trajetória da bola. Os resultados indicaram que o modelo de RNC 3D teve uma acurácia média de 0,8585 e *f1-score* de 0,8569, superando a combinação LSTM + RNC 2D. Na detecção de quiques, o algoritmo KNN obteve desempenho superior a outros algoritmos testados, com acurácia média de 0,9298 e *f1-score* médio de 0,8365. Na estimativa da velocidade média da bola, o sistema identificou 66 das 76 jogadas que ocorreram. Dentre essas, 36% apresentaram um erro abaixo de 10%, 38% tiveram um erro entre 10% e 40%, e 26% mostraram um erro superior a 40%. Esses resultados são promissores e, com melhorias nos modelos, o sistema tem o potencial de ser utilizado para a análise em tempo real das partidas.

Palavras-chave: <Visão computacional>, <Goalball>, <Aprendizagem de máquina>, <Reconhecimento de ações>.

ABSTRACT

The technologies of computer vision and machine learning have been transforming the way sports are analyzed. With these tools, it is possible to obtain more accurate insights into athletes' actions, optimizing strategies and improving the understanding of game dynamics. This work proposes a system for analyzing plays in goalball matches, using computer vision and machine learning techniques to recognize player actions, detect ball bounces, and calculate its average speed. The methodological approach included the creation of a custom dataset, the use of the Yolov8 model to detect players and the ball, convolutional neural network (CNN) models and Long Short-Term Memory (LSTM) networks for action recognition, and machine learning algorithms to detect bounces in the ball's trajectory. The results indicated that the 3D CNN model achieved an average accuracy of 0.8585 and an F1-score of 0.8569, outperforming the LSTM + 2D CNN combination. In bounce detection, the KNN algorithm outperformed other tested algorithms, achieving an average accuracy of 0.9298 and an average F1-score of 0.8365. In estimating the ball's average speed, the system identified 66 of the 76 plays that occurred. Of these, 36% showed an error below 10%, 38% had an error between 10% and 40%, and 26% showed an error above 40%. These results are promising, and with improvements to the models, the system has the potential to be used for real-time match analysis.

Key-words: <Computer Vision>, <Goalball>, <Machine Learninga>, <Action Recognition>.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

DP - Desvio padrão

DT - *Decision Tree* (Árvore de Decisão)

FPS - *Frames Per Second* (Quadros Por Segundo)

KNN - *K-Nearest Neighbors* (K-Vizinhos Mais Próximos)

LSTM - *Long Short-Term Memory* (Memória de Longo e Curto Prazo)

NB - *Naive Bayes* (Bayes Ingênuo)

NLP - *Natural Language Processing* (Processamento de Linguagem Natural)

RAH - Reconhecimento de Ação Humana

RGB - *Red, Green, Blue* (Vermelho, Verde, Azul)

RNC - Rede Neural Convolucional

RNN - *Recurrent Neural Network* (Rede Neural Recorrente)

SVM - *Support Vector Machine* (Máquina de Vetores de Suporte)

SW - *Sliding Window* (Janela Deslizante)

Sumário

1 INTRODUÇÃO

A visão computacional, uma subárea da inteligência artificial, capacita sistemas computacionais a interpretar e analisar imagens. Essa área tem demonstrado grande impacto em diversos setores, como a medicina, com a detecção precoce de câncer de mama [1]; a agricultura, ajudando na identificação de doenças em plantas [2]; e a segurança, com o uso do reconhecimento facial [3]. No esporte, essa tecnologia tem sido cada vez mais empregada para melhorar o desempenho dos atletas [4] e apoiar decisões estratégicas, com impacto direto na performance em campo.

Dentro desse contexto, o *goalball*, um esporte desenvolvido para atletas com deficiência visual, apresenta características que o tornam um campo interessante para a aplicação de visão computacional. Esse esporte ocorre em uma quadra com 9 metros de largura e 18 metros de comprimento, com movimentos dos jogadores caracterizados principalmente por padrões defensivos e ofensivos que se repetem ao longo das partidas. Diferente de esportes como futebol ou basquete, onde as movimentações são mais dispersas e complexas. Na defesa os jogadores realizam deslocamentos laterais rápidos para defender o gol e posicionam-se para bloquear a bola guiando-se pelo som. No ataque, utilizam técnicas como o movimento rotacional para lançar a bola com mais força e simulações para confundir a defesa. A identificação do movimento dos jogadores e o rastreamento da bola durante as jogadas podem ser utilizados para gerar informações que possibilitem ajustes estratégicos em tempo real.

Dessa forma, este trabalho propõe o desenvolvimento de um sistema para a análise de jogadas no *goalball*, utilizando vídeos das partidas. Por meio de modelos de aprendizado de máquina e visão computacional, o objetivo é identificar movimentos de ataque, os momentos em que a bola quica no chão e calcular a velocidade média dos arremessos. Essa pesquisa faz parte do doutorado de um aluno de Educação Física da UPE, cujo objetivo é usar inteligência artificial para criar um sistema que forneça *scouts* de forma automática em tempo real.

1.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver um sistema capaz de extrair informações referentes as jogadas durante uma partida de *goalball* usando vídeos.

1.2 Objetivos específicos

- Preparar os dados necessários para o treinamento dos modelos..
- Identificar o início e o fim de um movimento de ataque, que corresponde ao gesto realizado para lançar a bola para o lado adversário.

- Desenvolver um modelo que consiga reconhecer quando a bola quica no chão.
- Identificar a trajetória da bola.
- Estimar a velocidade média da bola em uma jogada.

1.3 Estrutura da monografia

- **Capítulo 1 - Introdução:** Apresenta o contexto e a motivação do trabalho, destacando a importância da visão computacional no esporte e introduzindo o objetivo geral da pesquisa, que é o desenvolvimento de um sistema para analisar partidas de *goalball*.
- **Capítulo 2- Conceitos Gerais e Revisão da Literatura:** Este capítulo discute os principais conceitos teóricos utilizados no trabalho, além de abordar estudos relacionados.
- **Capítulo 3 - Metodologia:** Descreve o processo de aquisição e processamento dos dados, os métodos empregados para o reconhecimento de ações, detecção de quiques e o cálculo da velocidade média da bola.
- **Capítulo 4 - Apresentação e Análise dos Resultados:** Neste capítulo, são discutidos os problemas para criar a base de dados, os resultados obtidos pelos modelos de reconhecimento de ação e detecção de quiques, assim como a precisão da estimativa de velocidade média da bola e alguns problemas encontrados.
- **Capítulo 5 - Conclusões e Trabalhos Futuros:** Apresenta as conclusões do trabalho, destacando as principais contribuições, limitações e sugestões para pesquisas futuras, incluindo possíveis melhorias nos modelos desenvolvidos.

2 CONCEITOS GERAIS E REVISÃO DA LITERATURA

Esta Seção apresenta os conceitos fundamentais para o trabalho, abordando a área de reconhecimento de ação humana e técnicas, como RNC, YOLO para detecção de objetos, LSTM para análise sequencial, *Sliding Window* para segmentação temporal e transformação de perspectiva para ajuste da visão da quadra. Por fim, uma revisão da literatura relevante.

2.1 Reconhecimento de ação humana

O reconhecimento de ação humana é uma área de pesquisa cuja finalidade é observar as pessoas a partir da análise de dados, tentando inferir quais comportamentos estão realizando. Além de identificar ações individuais, o reconhecimento de ações humanas também se estende à análise de ações em grupo, onde a interação entre várias pessoas é examinada [5].

Um dos desafios dessa área é a necessidade de lidar com variáveis como: oclusão, ângulos de câmeras e diferentes velocidades de movimento, as quais podem tornar essa tarefa complexa [5]. Além disso, existem diversos tipos de dados que podem ser úteis para essa observação, como abordado em [6].

No contexto desse trabalho, ela permite que o sistema interprete os movimentos dos jogadores utilizando as suas imagens. Para isso, será utilizada uma combinação de técnicas e modelos de *deep learning*, os quais serão detalhados nos tópicos seguintes.

2.2 Redes neurais convolucionais (RNC)

As redes neurais convolucionais (RNC) são um tipo de rede neural amplamente utilizado em tarefas de análise de imagens, como segmentação, detecção e classificação. Embora sejam especialmente eficazes em problemas relacionados à análise visual, também podem ser aplicadas em análises temporais de dados [7].

O princípio de funcionamento das RNC é semelhante ao dos filtros usados em processamento digital de imagens, como os de mediana, Gaussiano e de média. No entanto, nas RNC são aplicadas apenas operações lineares, que consistem em multiplicações e somas, correspondendo à correlação cruzada entre o *kernel* e os dados de entrada [7] (Figura 1. Além disso, os filtros podem ser aplicados de maneira independente em cada canal, em todos os canais simultaneamente ou em grupos, dependendo da configuração da rede [8].

No trabalho, elas são essenciais para a detecção de elementos visuais, como jogadores e a bola, usando o modelo YOLOv8, o qual será explica na próxima Seção. Além diso,

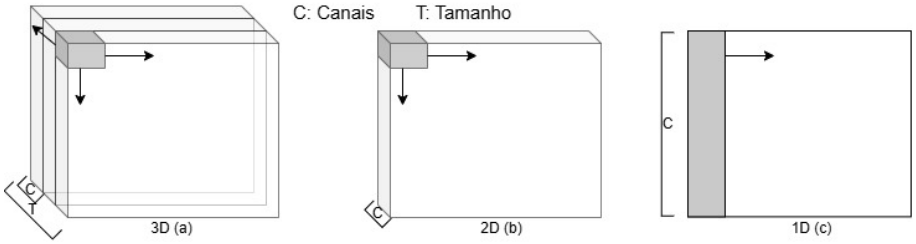

Figura 1: Em (a) temos o funcionamento de uma RNC 3D, na qual o kernel percorre 3 dimensões, (b) RNC 2D, com o kernel percorrendo 2 dimensões e (c) RNC 1D, em que o kernel percorre 1 dimensão (Fonte: O autor).

também foi usado para extrair as caracterísitcas das imagens dos jogadores para analisar os movimentos realizados.

2.3 YOLO (*You Only Look Once*)

O YOLO (*You Only Look Once*) [9] é uma rede neural convolucional voltada para a detecção de objetos em tempo real. Diferente de algumas abordagens, que analisam a imagem em múltiplas etapas [10], esse modelo realiza a detecção em uma única passada, dividindo a imagem em uma grade e identificando objetos em cada célula. A sua estrutura é formada por três partes [11]: a espinha dorsal (*backbone*), que extrai características da imagem; o pescoço (*neck*), que ajusta essas características para múltiplas escalas; e a cabeça (*head*), que gera as caixas delimitadoras e pontuações para identificar objetos.

Desde seu lançamento, o YOLO passou por várias modificações arquiteturais, resultando em novas versões, como YOLOv3, YOLOv4, YOLOv5 etc [12]. Neste trabalho, utilizou-se a versão YOLOv8 para localizar os jogadores e a bola (Figura 2), com o objetivo de analisar, de maneira mais detalhada, o movimento dos objetos.

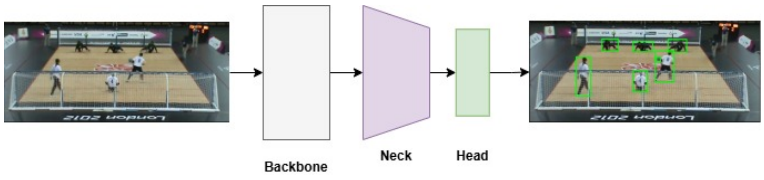

Figura 2: Imagem contendo as partes gerais da arquitetura YOLO e a detecção gerada (Fonte: O autor).

2.4 *Long Short Term Memory* (LSTM)

As *Long Short Term Memory*y (LSTM)[13] são um tipo específico de rede neural recorrente (RNN) projetadas para lidar com dados sequenciais [14]. Sua estrutura é formada por componentes chamados portões (*gates*) (Figura 3), e cada um é responsável

por ajustar, de forma dinâmica, as informações temporais dos dados [15]. O portão de esquecimento decide quais informações das células anteriores devem ser descartadas. O portão de entrada controla quais novas informações devem ser adicionadas ao estado da célula. Por fim, o portão de saída determina quais partes das informações internas da célula devem ser enviadas para a próxima camada da rede, ou para a próxima célula.

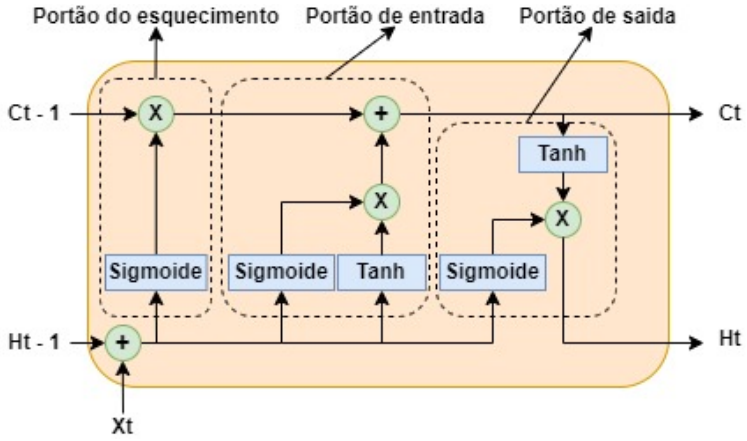

Figura 3: Arquitetura LSTM (Fonte: O autor).

No processamento de linguagem natural (NLP), as LSTMs são utilizadas em tarefas como, por exemplo, tradução automática, geração de texto, análise de sentimentos e *chatbots* [16]. Em previsão de séries temporais, ajudam a prever tendências futuras, por exemplo, o preço de ações [17]. Também são usadas no reconhecimento de fala [18] e na detecção de anomalias [19], onde aprendem padrões normais e identificam desvios que apontam possíveis problemas.

As LSTMs foram utilizadas para analisar a sequência das características extraídas pela RNC. Essa análise permiti uma compreensão do contexto e da continuidade dos gestos ao longo do movimento do atleta.

2.5 *Sliding Window* (SW)

A técnica de *Sliding Window* é usada em várias áreas da computação, sendo particularmente útil em situações que requerem a análise de fluxos de dados contínuos. Seu funcionamento consiste em utilizar uma janela de tamanho fixo, que se movimenta através dos dados de forma sequencial, examinando apenas uma subseção de dados por vez [20]. Dessa forma, cada "janela"captura uma pequena porção do conjunto de dados, permitindo que o sistema processe e extraia características.

No contexto de processamento de sinais, essa técnica é usada para capturar informações dentro de dados sequenciais, como na análise de eletroencefalogramas (EEG) ou eletrocardiogramas (ECG), facilitando a detecção de padrões [21]. Outro exemplo é

no uso em séries temporais, nas quais é escolhida uma quantidade N de dados, que serão analisados do passado com a finalidade de prever dados do futuro [22].

Esse método adiciona uma camada temporal ao sistema, analisando o intervalo e a duração dos movimentos capturados. Essa estratégia, em conjunto com os modelos de *deep learning* , permite segmentar cada movimento em períodos de tempo definidos, essencial para a análise precisa do início e do término das ações dos jogadores.

2.6 Transformação de perspectiva

A transformação de perspectiva é uma técnica matemática que aplica uma projeção linear para mapear pontos de um plano 2D em um novo sistema de coordenadas. Formalmente, é representada por uma multiplicação de matrizes, que descreve a relação entre as coordenadas originais de um plano e suas novas coordenadas projetadas [23]. Na visão computacional, esse método é utilizado para diversos fins, incluindo:

- Aumento de dados: É possível gerar diversas outras imagens a partir de uma, apenas aplicando mudanças de perspectiva [24];
- Vista aérea: Algumas aplicações que possuem detecção de objetos utilizam esse método para ter uma perspectiva aérea, *bird view*, dos elementos detectados;
- Corrigir distorção: Às vezes é necessário aplicar uma correção em imagens que estão distorcidas, por exemplo, para melhorar o reconhecimento de dígitos em placas de veículos [25] ou documentos [26].

Para que o sistema compreenda a posição aproximada dos jogadores e da bola em relação à quadra, a transformação de perspectiva é aplicada. Esse mecanismo ajusta a visualização do campo, permitindo um alinhamento que facilita o cálculo de distâncias e direções, além de fornecer uma interpretação espacial mais fiel do desfecho das jogadas.

2.7 Trabalhos relacionados

O artigo [27] aborda o desenvolvimento de um sistema que utiliza processamento de imagens para estimar a posição e a trajetória da bola durante o jogo. A metodologia inclui a detecção de objetos em movimento por meio da diferença temporal entre quadros do vídeo (*frame*), a identificação de formas circulares com a Transformada de Hough para detectar a bola, e a transformação de coordenadas por homografia para estimar a trajetória da bola na quadra. Além disso, o artigo apresenta um dispositivo tátil, que converte informações analíticas em *feedback* sensorial para os jogadores, facilitando a compreensão das táticas de jogo. O sistema alcançou uma precisão de 91% na detecção da

bola, embora ainda enfrente desafios quando a bola é parcialmente oculta por jogadores, ou postes do gol.

Em [28], os autores apresentam um trabalho em que o principal objetivo é fazer a análise e classificação dos movimentos dos jogadores e criar uma segmentação temporal dessas ações. Os autores criaram o próprio conjunto de dados, UNIRI-HBD_v2, que são imagens do recorte dos jogadores detectados, usando um sistema semi-supervisionado. No reconhecimento de ação, aplicaram um modelo com a arquitetura denominada *Inflated 3D Networks* (I3D) e a entrada é composta por uma sequência N de *frames rgb* e *optical flow*. Além disso, fizeram *fine-tuning* em um modelo que foi pré-treinado na base de dados *Kinetics* [29]. Para a detecção, executaram testes com Yolov3, Yolov3-PB, Mask R-CNN, Yolov7 e Yolov7-e6e.

O estudo feito em [30] faz uma análise de jogos, com o objetivo de identificar ações individuais e feitas em grupo. Para fazer o reconhecimento das ações, primeiramente é executada a extração de características de cada um dos jogadores, a partir de um sistema de detecção múltipla de objetos. Em seguida, esses atributos são passados para um modelo denominado *dual attention*, o qual é composto pelos módulos *Spatial attention* e *Mixture channel attention*. Por fim, a saída é passada para uma rede chamada *Individual spatial-temporal inference*, que produz uma saída usada para fazer o reconhecimento de ação individual dos jogadores e também como entrada para uma RNN, que fará o reconhecimento da ação em grupo. Esse método foi validado nas bases de dados *Volleyball* e *Collective Activity*. Apresentou resultados superiores, variando entre 0,002 e 0,02 na acurácia, e acima da média se comparando a outros estudos da literatura.

No trabalho [31], os autores propõem a *Energy-Motion Features Aggregation Network* (EMA-Net) para a análise de ações de jogadores de futebol, usando vídeos de partidas. Para isso, criaram um conjunto de dados próprio contendo vídeos anotados com tipos de dribles: *Stepover*, Elástico e *Chop*; tipos de chutes: pênalti, chute livre e chute para o gol; ocorrência de gols e defesas. O EMA-Net combina características da energia do movimento e o esqueleto de pontos dos jogadores, utilizando o Mask R-CNN para detecção de jogadores e bola, além do *OpenPose*, para mapear pontos-chave do corpo. Além disso, a rede inclui módulos que utilizam sistemas *self-attention* nas características de energia e movimento, que no final são agregadas. O modelo foi validado em vídeos de destaques de futebol e demonstrou superioridade na classificação de ações, em relação a outros trabalhos.

3 METODOLOGIA

Esta Seção detalha o processo metodológico aplicado no projeto. Inclui a seleção dos vídeos usados para o treinamento dos modelos, preparação dos dados para o reconhecimento de ação e detecção de quique, como foi feito o treinamento dos modelos e o método utilizado para calcular a velocidade média da bola.

3.1 Seleção e Recorte dos vídeos

Como não existem dados públicos para o reconhecimento de ações de jogadores no *goalball*, foi necessário criar uma base própria. Utilizando vídeos do YouTube, realizaram-se capturas de partidas com perspectiva traseira, conforme o protocolo do projeto. Além disso, foram selecionados jogos com variabilidade em termos de iluminação, tanto em relação à luz natural, quanto à artificial, à proporção que a quadra ocupa na imagem, ao ângulo e à cor da quadra, da vestimenta dos jogadores e dos juízes.

Foram catalogados 16 vídeos (Figura 4) com duração entre 7 e 53 minutos. Para os vídeos que ultrapassavam 10 minutos, selecionaram-se intervalos que respeitassem esse limite, priorizando aqueles com variedade e quantidade de ações. Após a seleção dos intervalos, os vídeos foram recortados. Contudo, um dos vídeos apresentava uma perspectiva de altura significativamente diferente dos demais, levando à sua exclusão desta versão da base de dados.

Os movimentos capturados foram: ataque (como entre as pernas, horizontal ou com giro) e normais (como abaixar para pegar a bola, levantar-se com a bola, se mover dentro da quadra, segurando a bola na mão e pós-arremeso).

3.2 Dados para o modelo de reconhecimento de ação

Como o objetivo é identificar quando uma jogada será realizada, foi escolhido analisar somente os jogadores, pois isso garante que o modelo se concentre exclusivamente no movimento corporal do atleta, sem ser influenciado por elementos irrelevantes na imagem completa. Tendo isso em mente, o modelo usará apenas a imagem do jogador, que será obtida a partir de um recorte utilizando a detecção gerada por um modelo Yolov8. Esse modelo foi treinado com imagens obtidas no YouTube, e o treinamento foi realizado por integrantes que fazem parte dessa pesquisa. No entanto, abordar o resultado e como foi feito o treinamento não faz parte do escopo deste trabalho.

Antes de utilizar o modelo de detecção, foi necessário fazer uma análise prévia dos vídeos e gerar um arquivo CSV, contendo informações a respeito de todos os movimentos que seriam capturados. Esse arquivo é necessário para indicar quais os *frames* utilizados

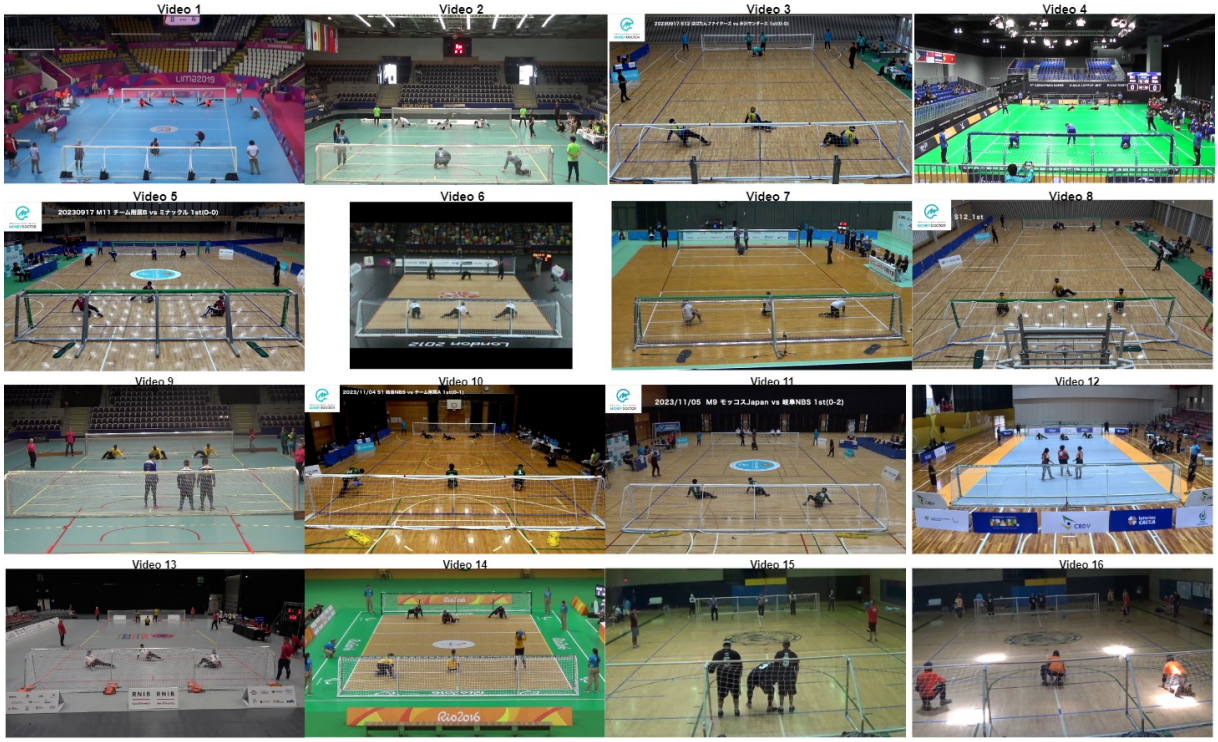

Figura 4: Todos os vídeos catalogados, incluindo o vídeo que não foi adicionado ao conjunto de dados, vídeo 8 (Fonte: O autor).

no recorte e possui informações sobre os movimentos. A Tabela 1 lista as colunas presentes no arquivo e seus respectivos significados.

Tabela 1: Significado de cada uma das colunas do csv

Coluna	Significado
video_name	nome do vídeo
first_frame	frame em que se inicia o movimento
last_frame	frame em que termina o movimento
play	valor booleano indicando se é um movimento de ataque ou não
side	valor booleano indicando o lado da quadra onde o jogador está (1 superior ou 0 inferior, na perspectiva da câmera)

O procedimento final envolve gerar imagens das áreas onde os jogadores foram identificados pelo modelo Yolo. Para isso, foi desenvolvido um código (Figura 6), que processa cada *frame* dos vídeos, seguindo algumas etapas fundamentais.

1. **Aplicação de máscaras em bolas fora da quadra:** As máscaras impedem que o modelo detecte as bolas fora do limite da quadra, diminuindo o risco de que bolas irrelevantes sejam utilizadas. Essa etapa se fez necessária em apenas dois vídeos.

2. **Filtro de confiança:** Se forem detectadas mais de uma bola, apenas a que tem maior confiança é utilizada no processo.

3. **Verificação da ausência de detecção de bola:** Se a bola não for detectada em algum *frame*, as coordenadas da última bola detectada serão utilizadas (Figura 5). Essa checa permite não perder quadros do vídeo caso a bola sofra oclusão pelo jogador, ou não seja detectada.

4. **Filtragem por proporção da caixa:** O filtro de proporção serve para ignorar caixas de jogadores com a altura menor que a largura (Figura 5), o que geralmente indica que o jogador está em uma posição não relevante, por exemplo, deitado.

5. **Verificação da interseção:** Verifica se a caixa delimitadora da bola possui interseção com a caixa de algum dos jogadores em quadra. Essa etapa ajuda a garantir que apenas os jogadores que estão perto, ou com a posse da bola, sejam adicionados ao conjunto de dados.

6. **Adição de folga:** Antes de salvar a imagem recortada do jogador, foi adicionado 15 pixels de altura e largura além da caixa delimitadora. Essa margem é importante, porque, em alguns casos, o modelo não detecta a totalidade do corpo do jogador. Portanto, esse espaço visa capturar partes que poderiam ser perdidas durante a detecção.

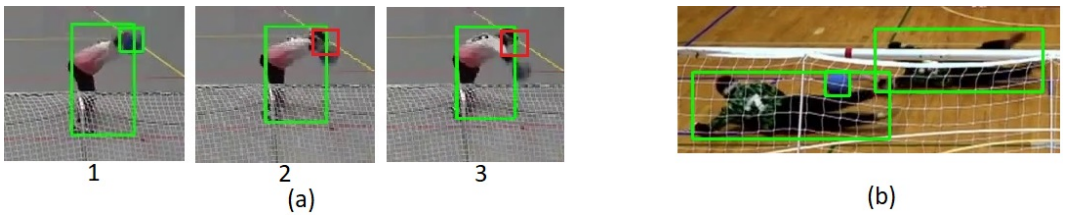

Figura 5: Em (a) temos um exemplo em que a bola não foi detectada no *frame* 2 e 3, porém é utilizado as coordenadas do *frame* 1 (quadrado vermelho) em (b) A caixa da bola possui interseção com a caixa de um dos jogadores, mas não passa pelo filtro de proporção, item 4 (Fonte: O autor).

3.3 Dados para a detecção de quique

O objetivo foi criar um conjunto de dados que poderia ser utilizado por um algoritmo de aprendizagem de máquina para classificar pontos da trajetória da bola, que ocorreram como quiques, classe 1, ou não, classe 0 (Figura 7).

O modelo YOLOv8 utilizado para o reconhecimento de movimento, apresentado na Seção anterior, foi usado para gerar um arquivo CSV, com as coordenadas centrais X e Y da caixa delimitadora da bola em cada *frame*.

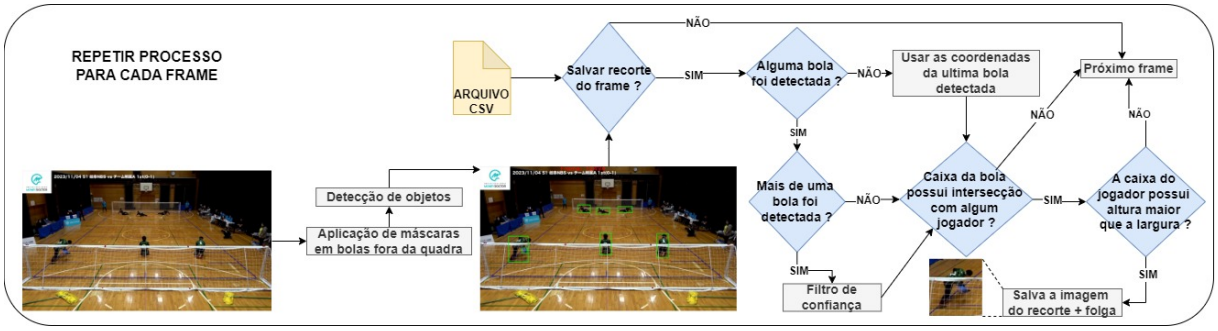

Figura 6: Diagrama contendo o fluxo lógico para extrair as imagens que foram usadas para treinar o modelo de reconhecimento de ação humana (Fonte: O autor).

Durante o processamento dos vídeos, aplicaram-se máscaras em bolas fora da quadra e o filtro de confiança. Caso nenhuma bola tenha sido detectada, os valores X e Y da última detecção foram repetidos.

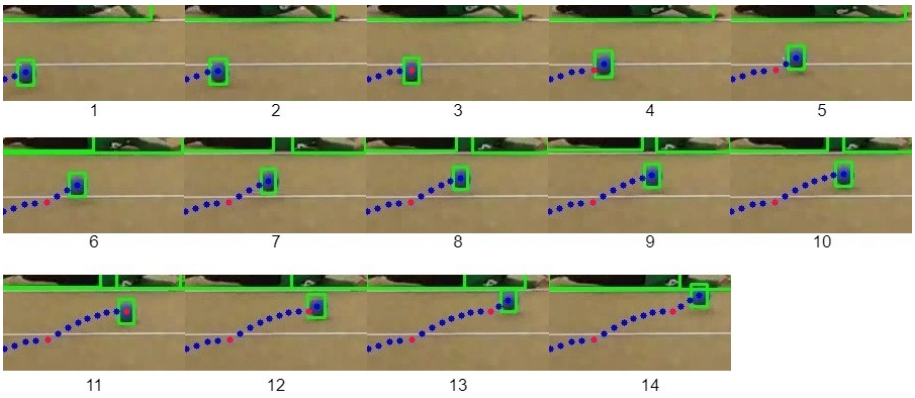

Figura 7: Exemplos de pontos em que ocorrem quiques, bolinhas vermelhas, e pontos em que não ocorrem quiques, bolinhas azuis, (Fonte: O autor).

Além disso, foi necessário criar outro arquivo CSV contendo o número dos *frames* em que os quiques ocorrem, classe 1. Todos os vídeos foram analisados; no entanto, nem todos os quiques que ocorreram após o arremesso foram adicionados, pois o processo de análise e anotação manual é demorado. Além disso, em alguns momentos, houve dúvida em alguns *frames*, devido à perspectiva do vídeo.

Esse arquivo também foi usado para selecionar os exemplos negativos, classe 0, que correspondem a momentos onde não ocorreu o quique. Para isso, adotou-se uma estratégia de amostragem temporal, escolhendo imagens específicas antes e depois dos quiques: o 2^{o}, 4^{o}, 6^{o} e 10^{o} pontos, garantindo tanto exemplos mais próximos (2^{o} e 4^{o}) quanto mais distantes (6^{o} e 10^{o}) do evento.

O diagrama da Figura 8 apresenta um resumo do fluxo de todas as etapas descritas anteriormente.

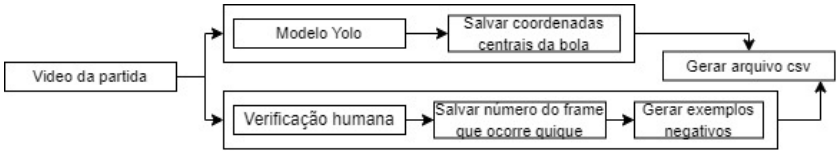

Figura 8: Diagrama ilustrando o fluxo de criação dos dados para a detecção de quique (Fonte: Autor).

3.4 Treinamento dos modelos

Esta subseção explica como foram realizados os treinamentos para os modelos de reconhecimento de ação e detecção de quiques.

3.4.1 Reconhecimento de ação

Devido à variabilidade na duração dos movimentos, o modelo foi treinado com pequenos trechos que fazem parte de toda a ação, denominados de sub-ações (Figura 10), como feito em [32] e [33]. Esses trechos são compostos por 4 imagens sequenciais com sobreposições, usando um tamanho de passe igual a 1. Todas as imagens foram redimensionadas para o tamanho 108x68 (altura x largura), visto que era necessário ter um tamanho padronizado para treinar o modelo. A escolha desses valores se deu a partir da análise da distribuição das medidas de altura, largura e a proporção das imagens. A Figura 9 contém gráficos a respeito.

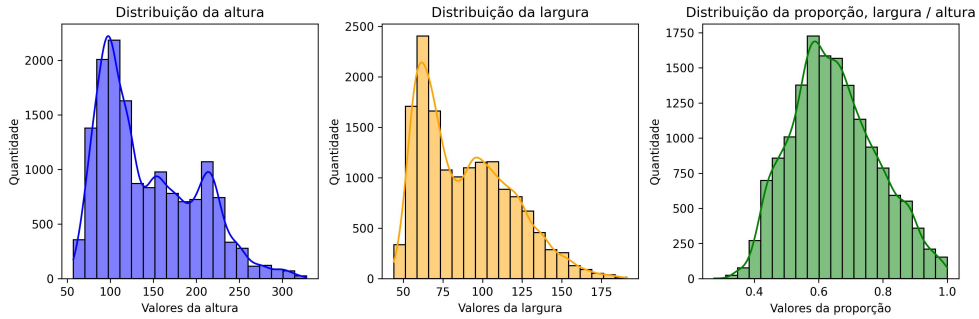

Figura 9: Histogramas contendo informações a respeito do tamanho das imagens (Fonte: O autor).

De acordo com a revisão apresentada em [34], o principal método para reconhecimento de ação em esportes usando *deep learning*, envolve o uso de RNC 2D combinadas com LSTM. Nesse cenário, a RNC é responsável por extrair as características das imagens, enquanto o LSTM realiza a análise temporal dessas informações. No entanto, arquiteturas que utilizam RNC 3D e *transformers* também foram discutidas em [34] e [35]. Apesar do potencial dos *transformers*, seu uso requer uma grande quantidade de dados para treinamento [36], o que inviabilizou sua aplicação neste trabalho, dado o tamanho da base de dados criada.

Figura 10: Figura contendo exemplos de movimentos completos e as sub-ações, em (a) um movimento de ataque, classe 1, e em (b) movimento normal, classe 0 (Fonte: O autor).

Portanto, realizou-se um teste preliminar comparando as arquiteturas RNC 2D + LSTM e RNC 3D para determinar qual apresentava os melhores resultados nos dados disponíveis. Conforme mostrado na Tabela 6, a RNC 3D demonstrou um desempenho superior em relação à RNC 2D + LSTM, sendo escolhida para prosseguir com os experimentos.

A arquitetura do modelo usada para o reconhecimento de ação é composta por convoluções 3D e 2D, *Squeeze-and-Excitation* [37], *GlobalAveragePooling*, camadas lineares e *dropout* para ajudar na regularização [38]. A estrutura específica da arquitetura está na Figura 11 e foi obtida a partir de mudanças até conseguir resultados, que sejam suficientes para testar o sistema de medição de velocidade média, que será explicado na Seção 3.5. Além de usar imagens coloridas, também foi feito um teste com o uso das imagens na escala de cinza, a fim de verificar se a cor possui impacto nos resultados.

Para aumentar a variabilidade das imagens, foram aplicadas técnicas de aumento de dados durante o treinamento. Para isso, utilizou-se a biblioteca *Albumentations* [39], que oferece uma ampla variedade de métodos e rápida execução. Além disso, o aumento de dados foi aplicado em tempo real, isto é, a cada época o modelo recebe as imagens originais com alterações. Essa abordagem também foi usada em [40] e [41]. Desse modo, não é necessário expandir a base de dados, o que ajuda a não prolongar o tempo de treinamento. Contudo, como cada exemplo é composto por 4 imagens sequenciais e não apenas 1, o aumento de dados é aplicado de forma igual para todas elas.

Como mencionado na Seção de coleta dos vídeos, foram escolhidos exemplos com

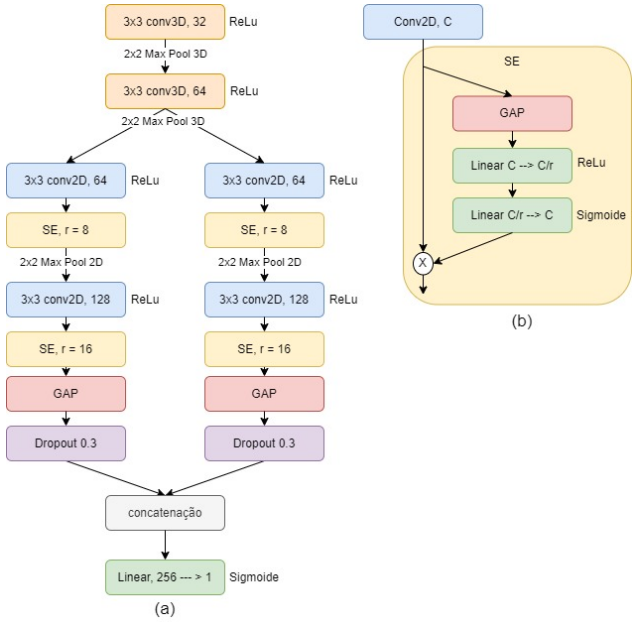

Figura 11: Arquitetura do modelo: em (a) está a representação completa e em (b) a estrutura do bloco SE, Squeeze-and-Excitation (Fonte: O autor).

aspectos visuais diferentes para ajudar na generalização do modelo. Atentando a isso, foram escolhidos aumentos de dados que aplicassem essas mudanças visuais e também outros que são mais comuns, como mostrado em [42] e [43].

Tabela 2: Aumentos de dados usados no treinamento

Aumento de dados	Parâmetros
HorizontalFlip	Default
CropAndPad	percent=0,13
Affine	rotate=(-5, 5)
CoarseDropout	max_height=10, max_width=10, max_holes=5
Perspective	keep_size=1
PixelDropout	dropout_prob=0,05
RandomBrightnessContrast	Default
HueSaturationValue	Default
MotionBlur	Default

Os hiperparâmetros utilizados foram: taxa de aprendizagem de 0,001, otimizador AdamW, ambos obtidos com testes empíricos, *batch size* de 64 [44] e a função de perda *Binary Cross Entropy* [45], recomendada em problemas de classificação binária. Todos os experimentos foram feitos no Google Colab, plataforma em que é possível criar *jupyters notebooks* e utilizar GPUs para treinar modelos de *deep learning*, porém os seus recursos são limitados. Tendo isso mente, foram feitos testes iniciais com 30 épocas e foi notado que a partir da época 15 alguns treinamentos apresentavam *overfitting*, então foi escolhido 20 épocas para fazer os demais experimentos.

Para avaliar a capacidade de generalização do modelo, foram criados 10 conjuntos aleatórios, cada um contendo 3 dos 15 vídeos selecionados. Ao formar esses conjuntos, garantiu-se que nenhum vídeo aparecesse mais de uma vez em um mesmo grupo e que cada um fosse incluído em pelo menos um deles, assegurando a utilização de todos. Os vídeos que pertencem ao conjunto são usados como dados de validação, enquanto os demais servem como dados de treinamento. Esse processo permite testar o desempenho do modelo em diferentes contextos, já que um mesmo vídeo não é utilizado simultaneamente para treinamento e validação. A Tabela 3 mostra os vídeos que fazem parte de cada um dos grupos criados.

Tabela 3: Grupos criados

Nº grupo	**Vídeos**
1	6, 11 e 13
2	5, 7 e 13
3	4, 10 e 11
4	7, 11 e 12
5	10, 12 e 15
6	1, 2 e 12
7	4, 5 e 13
8	3, 6 e 11
9	7, 14 e 16
10	9, 10 e 15

3.4.2 Detecção de quique

Cada ponto na trajetória da bola é representado pelos valores das coordenadas X e Y na imagem, porém apenas essa informação não é suficiente para diferenciar se um ponto corresponde a um quique ou não. Dessa forma, a partir das coordenadas, foram criados dois novos tipos de características para representar cada um dos pontos (Figura 12). Após a elaboração dos novos dados, as coordenadas X e Y foram descartadas.

A primeira característica envolve a diferença entre os valores Y do ponto analisado e dos três pontos seguintes e anteriores. A segunda refere-se ao ângulo formado entre o ponto central e os pontos anterior e posterior. Esses atributos foram desenvolvidos com base no estudo [46] e em testes empíricos.

Para classificar um ponto sendo quique ou não, foram testados modelos de aprendizagem de máquina utilizados no estudo [47], os quais são: *Naive Bayes*, *Decision Tree*, *K-Nearest Neighbour* e *Support Vector Machine*. Os parâmetros padrão foram aplicados a todos; no entanto, para os algoritmos que possuem a configuração *random_state*, foi atribuído o valor 2 para fins de reprodutibilidade.

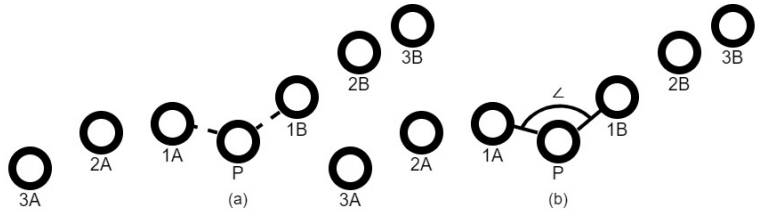

Tabela								
Py - 1By	Py - 2By	Py - 3By	Py - 1Ay	Py - 2Ay	Py - 3Ay	∠(1A, P, 1B)	∠(2A, P, 2B)	∠(3A, P, 3B)

(c)

Figura 12: Em (a) temos a primeira característica, a qual seria a diferença do valor Y entre os pontos posteriores e anteriores (b) é o ângulo formado entre os pontos, por exemplo, os pontos 1A, P e 1B (c) todas as features que seriam criadas para representar o ponto P (Fonte: O autor).

O método de validação cruzada utilizado foi o mesmo que o de reconhecimento de ação.

3.5 Medição de velocidade média

Para estimar a velocidade média da bola em uma jogada, o sistema utiliza a técnica de *sliding window*, juntamente com o modelo de reconhecimento de ação e detecção de quiques. O sistema possui 5 principais partes, as quais são:

1. **Máscara dinâmica:** Como as máscaras aplicadas para evitar detecções de bola fora da quadra são introduzidas usando *hard code*, não é possível utilizá-las no sistema em tempo real. Levando isso em consideração, foi criado um método de máscara dinâmica, que funciona em tempo real, ou não, e consegue bloquear algumas bolas fora da quadra. O método consiste em utilizar as detecções dos jogadores e os 4 pontos da quadra para criar uma área de filtro, que se modifica de acordo com a altura das caixas dos jogadores que estão dentro da área da quadra. A Figura 13 possui um exemplo de como essa máscara funciona.

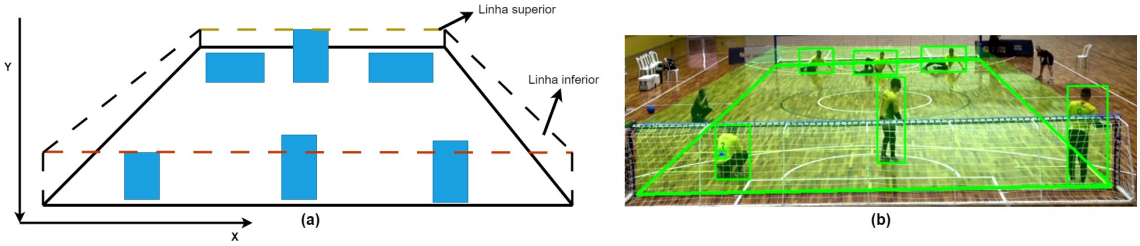

Figura 13: Em (a) temos um exemplo de como a linha superior e inferior se adequam à altura de alguns dos jogadores, caixas azuis, em (b) um frame real mostrando a máscara, parte verde com opacidade (Fonte: O autor).

A altura da linha superior se ajusta de acordo com o menor valor de Y das caixas dos jogadores de cima, enquanto a linha inferior se ajusta de acordo com o maior

valor de Y das caixas dos jogadores de baixo. Se algum jogador for detectado fora da quadra, a máscara não se ajustará a seu tamanho. Após as bolas passarem pela máscara dinâmico, elas são submetidas ao filtro de confiança, o qual já foi explicado nas seções anteriores.

2. **Detecção do início e fim de um movimento de ataque:** Como os movimentos de ataque variam em sua duração, uma *sliding window* é utilizada para capturar a sequência temporal desses movimentos (Figura 15). Sempre que ocorre uma interseção entre a bola e um jogador, o modelo de reconhecimento de ação humana (RAH) retorna a probabilidade do movimento ser um ataque. Essa probabilidade é então adicionada à *sliding window*. O início e o fim de um movimento de ataque são controlados por uma flag (variável binária usada para sinalizar estados), a qual é atualizada com base na média dos valores presentes na janela deslizante. Quando a média indica que não há movimento de ataque, a *flag* assume o valor 0, significando que o ataque terminou ou não está ocorrendo. Por outro lado, se a média indica a presença de um ataque, a *flag* assume o valor 1, indicando que o ataque começou ou está em andamento. Esse processo pode ser comparado ao funcionamento de uma máquina de estados, conforme ilustrado na Figura 14.

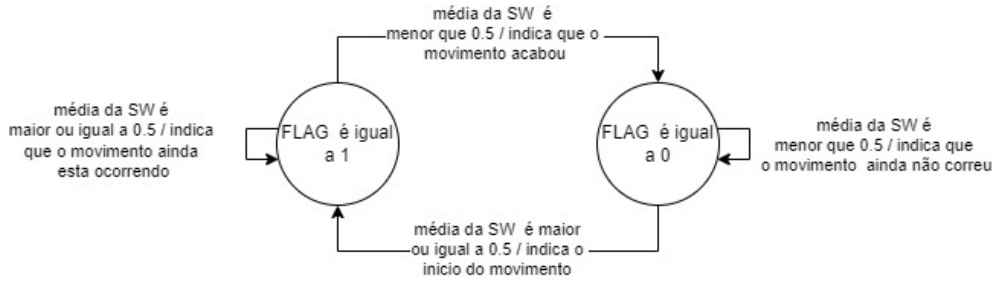

Figura 14: Essa figura ilustra o funcionamento da flag que controla o incio e fim de um movimento de ataque (Fonte: O autor).

3. **Intervalo de uma jogada:** Após a identificação do fim de um movimento de ataque, quando a *flag* muda de 1 para 0, o sistema armazena as coordenadas da caixa da bola nos próximos 32 *frames*. Essa escolha de intervalo se justifica pela dificuldade em determinar, com precisão, o momento exato em que a bola é defendida, ou sai de campo. Assim, esse período serve para simular o tempo, após o arremesso, e alguns momentos antes da bola ser defendida ou sair de jogo. O valor 32 foi selecionado com base na análise da quantidade de *frames* entre o arremesso e a saída da área de ataque do adversário (Figura 16). Para essa análise, foram escolhidos apenas quatro vídeos dos quinze, pois esse processo é demorado, visto que é um processo manual de contagem de *frames*.

4. **Detecção dos Quiques:** Após o intervalo, são geradas as novas características

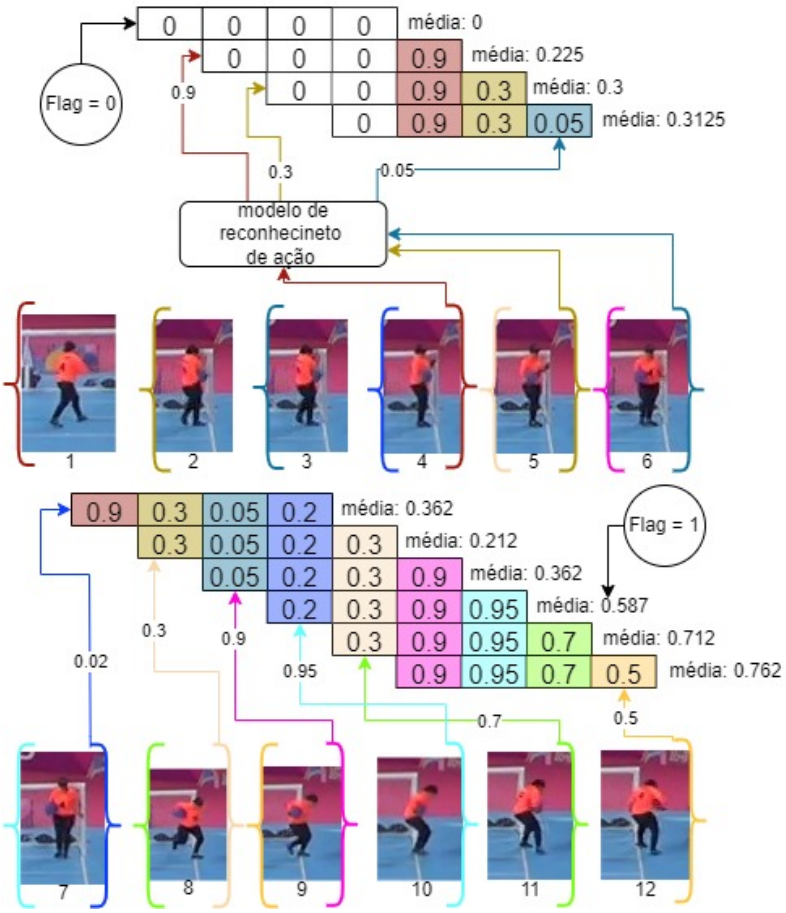

Figura 15: Esa imagem ilustra o funcionamento da SW em conjunto com a flag e o modelo de reconhecimento de ação. A flag se inicia com o valor 0 e continua assim até o frame 9. A partir do frame 10 a flag recebe o valor 1, indicando o início de um movimento de ataque, todavia o movimento se inicia no frame 8 (Fonte: O autor).

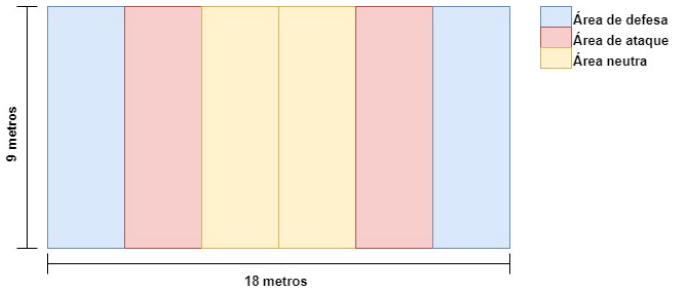

Figura 16: Divisões de uma quadra de goalball e seu tamanho (Fonte: O autor).

para cada um dos pontos e então é feita a classificação. Para a próxima etapa, é utilizado o último ponto em que o modelo classificou como um quique.

5. **Cálculo da Velocidade:** A velocidade média da bola é calculada utilizando as coordenadas do jogador que lançou a bola e as coordenadas do quique da bola (Figura 18). O método segue os seguintes passos:

(a) Fazer a transformação de perspectiva do ponto de quique e posição do jogador, para um novo mapa de coordenadas.

(b) Calcular o tempo de deslocamente(Δt) do ponto de arremeso até o último ponto de quique, usando a fórmula $\Delta t = \text{nº quique /FPS}$.

(c) Usar os pontos pós transformação para calcular o deslocamento da bola em metros (Δs).

(d) Usar o resultado dos pontos b e c para aplicar a fórmula da velocidade média e depois transformar para km/h.

$$v_m = \frac{\Delta s}{\Delta t}$$

O diagrama da Figura 17 contém o fluxo lógico das partes explicadas anteriormente e outros componentes necessários.

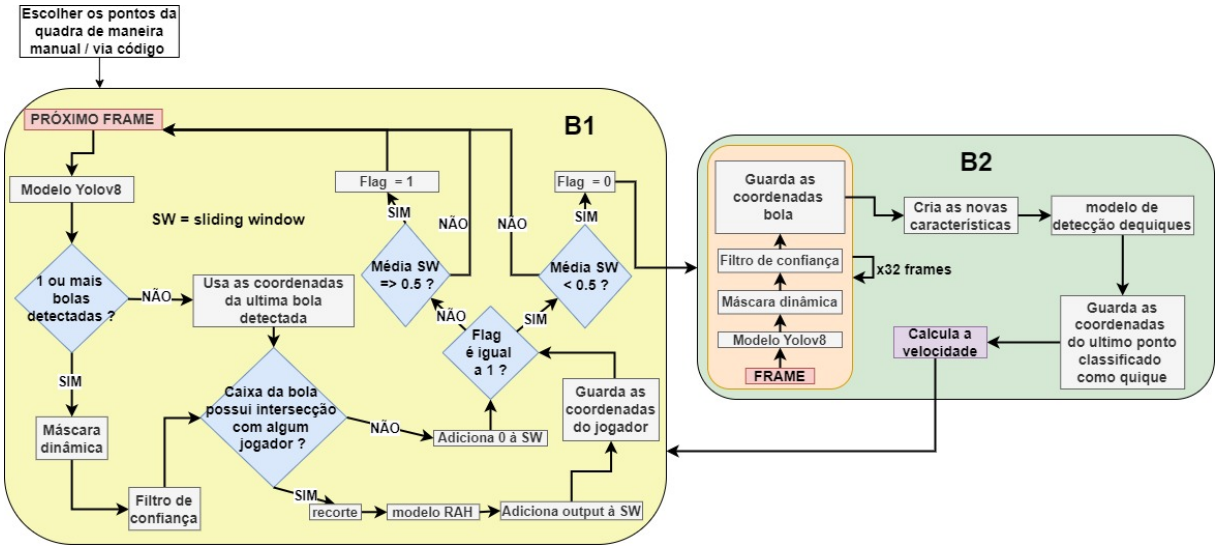

Figura 17: O bloco B1 contém a lógica responsável por identificar o início , fim de um movimento de ataque, enquanto o bloco B2 contém a parte responsável pela detecção de quiques e o cálculo a velocidade média (Fonte: O autor).

Como mostrado na Figura 17, se o sistema for utilizado em tempo real, será necessário fazer a seleção dos pontos da quadra de forma manual, pois não foi desenvolvido um meio para fazer a detecção automática. Caso seja uma gravação, pode ser feito de ambas as formas. Para esse experimento, os pontos foram adicionados via código.

Para efetuar a transformação de perspectiva, foi utilizada a biblioteca OpenCV [48], a qual é amplamente aplicada em projetos de visão computacional, pois possui diversas ferramentas para o processamento de imagens. A mudança de perspectiva é feita utilizando duas funções: getPerspectiveTransform() e perspectiveTransform(), a primeira é responsável por criar a matriz de transformação e a segunda por mapear os pontos da imagem original para o novo mapa de coordenadas.

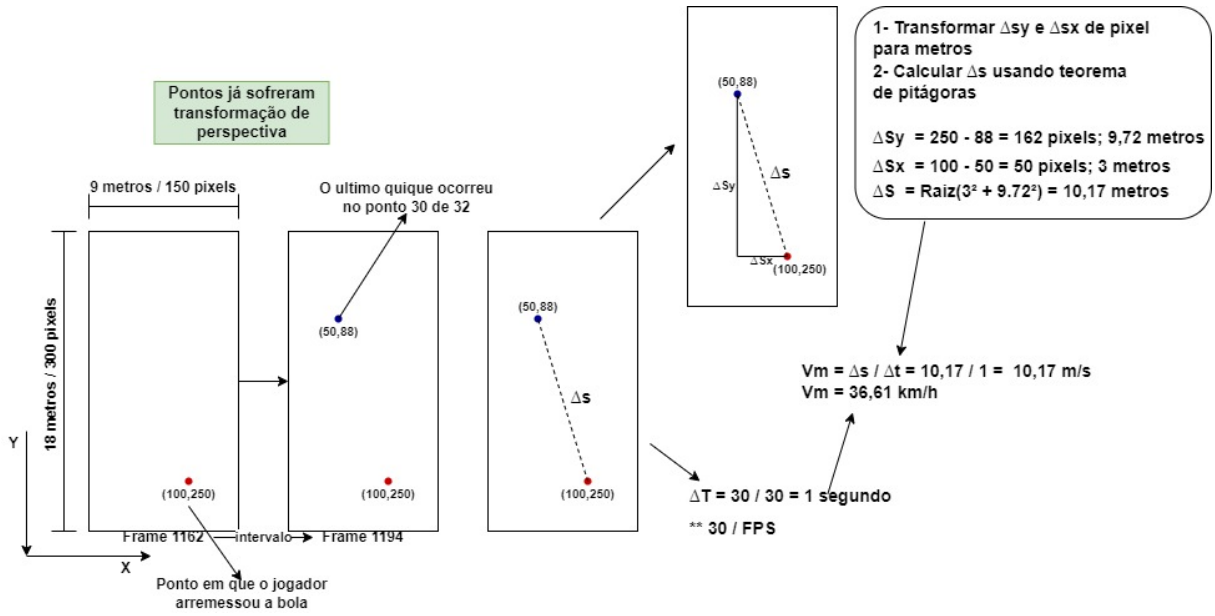

Figura 18: Lógica para calcular a velocidade média (Fonte: O autor).

Para testar o sistema, utilizou-se a gravação de uma partida de *goalball*, onde um radar tipo pistola mediu a velocidade da bola após o arremesso, e os dados foram comparados com as velocidades estimadas pelo sistema. Além disso, foram inseridas informações importantes em cada *frame* (Figura 19), como o estado das variáveis, entrada do modelo de RAH, transformação de perspectiva e valores de ΔSx e ΔSy, visíveis apenas após o intervalo de uma jogada, explicado anteriormente.

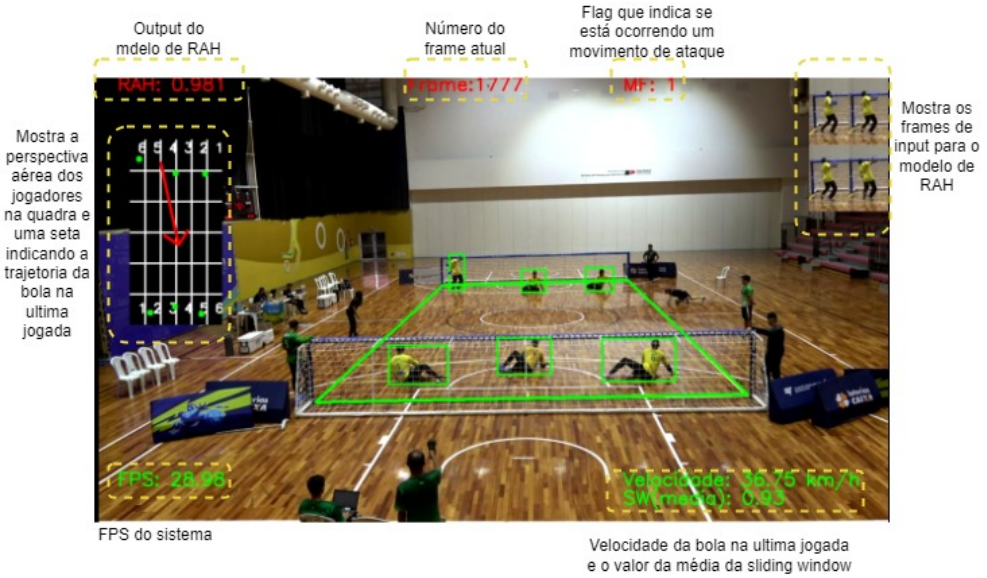

Figura 19: Informações mostradas no frame (Fonte: O autor).

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Esta Seção apresenta as dificuldades em preparar os dados para adicionar à base de dados e os resultados experimentais obtidos, incluindo métricas de desempenho como acurácia e *f1-score* dos modelos de reconhecimento de ação e detecção de quiques. Também é feito uma análise da saída do modelo de reconhecimento de ação em alguns movimentos de ambas as classes e são discutidos os desafios encontrados para calcular a velocidade média da bola.

4.1 Reconhecimento de ação - Recorte dos jogadores

Durante o processo de recorte de imagens para criar os dados, surgiram alguns problemas relacionados à detecção de objetos. Em certas situações, a caixa delimitadora da bola apresentava interseção com mais de um jogador no mesmo *frame* (Figura 20). Isso resultava no recorte de um jogador que não estava diretamente envolvido no movimento.

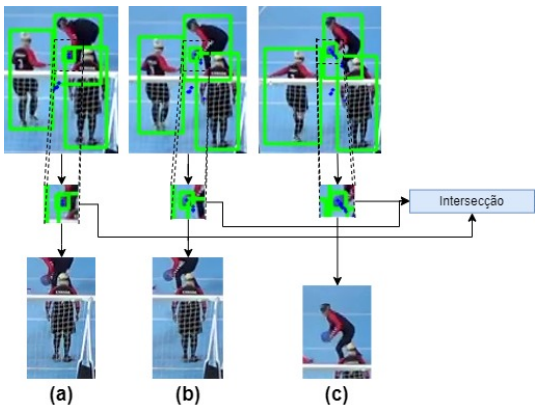

Figura 20: Em (a) e (b), a caixa da bola teve interseção com 2 jogadores devido à perspectiva, consequentemente, salvou o jogador que não estava com a bola. Em (c), o processo ocorreu corretamente (Fonte: O autor).

Além disso, a detecção de objetos falhou em capturar completamente o corpo dos jogadores em alguns *frames* [49]. Isso pode ocasionar na perda de informações cruciais sobre os movimentos, pois a posição dos braços e pernas são indicativos sobre o tipo de movimento (Figura 21).

Outro problema foi a detecção incorreta da bola. Quando ela não era detectada, ou era detectada em um local errado, o *frame* correspondente era ignorado, gerando lacunas nos dados.

Por último, foram gerados dois gráficos que mostram a porcentagem de imagens obtidas, em relação a quantidade total de imagens do movimento, considerando tanto as classes 0 e 1, quanto o lado em que ocorreu o movimento (Figura 22).

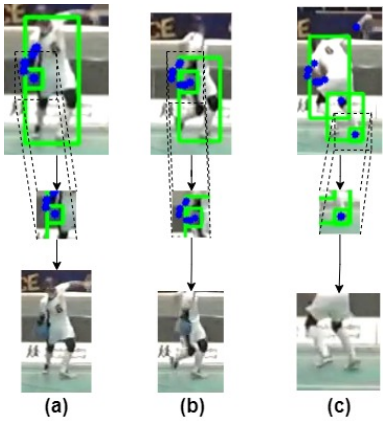

Figura 21: Em (a), a detecção do jogador gerou um recorte correto; em (b), cortou da metade do abdômen para cima; e em (c), o modelo detectou uma parte do pé e o corpo como jogadores diferentes. No entanto, uma bola foi identificada na parte do pé, resultando em um recorte da parte inferior (Fonte: O autor).

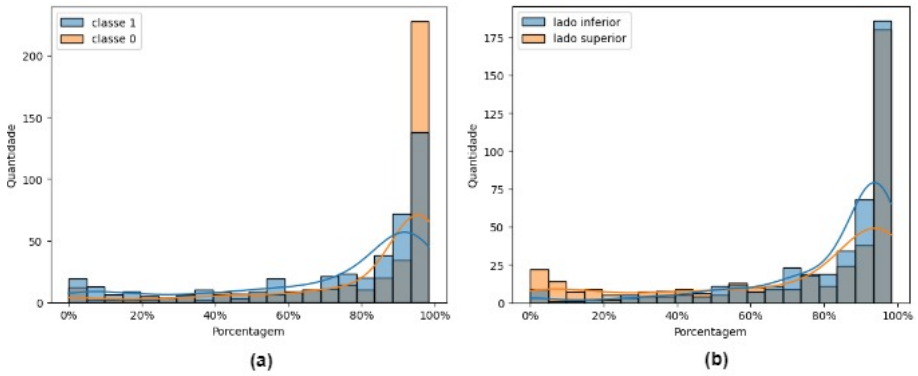

Figura 22: Em (a) temos a porcentagem em relação às classes e em (b) em relação aos lados (Fonte: o autor).

Conforme mostrado nos gráficos, em alguns casos não foi possível obter imagens do movimento, 0%. No entanto, de modo geral, o sistema ainda conseguiu gerar uma quantidade satisfatória de dados, pois a maioria dos valores estão próximos a 100%. O gráfico também revela que houve mais dificuldade para capturar as imagens dos recortes quando o movimento pertence à classe 1 e ocorre no lado superior da quadra, possivelmente devido a falhas do modelo Yolo nesses cenários.

4.2 Reconhecimento de ação - Modelo

Como mencionado, occoreu um teste preliminar comparando a RNC 3D com a combinação LSTM + RNC 2D. Os resultados mostraram que a RNC 3D superou significativamente o modelo LSTM + RNC 2D, com diferenças de até 0,2 na acurácia e chegando a até 0,32 no *f1-score*, no Grupo 6. A RNC 3D alcançou uma média de acurácia de 0,8340 e *f1-score* de 0,8234, enquanto o modelo LSTM + RNC 2D obteve 0,7026 e

0,6676, respectivamente. Além disso, a RNC 3D apresentou uma menor variação nos resultados, conforme indicado pelos desvios padrão menores. O resultado está na Tabela 4.

Tabela 4: Resultado dos grupos para LSTM + RNC 2D E RNC 3D

Grupo	LSTM + RNC 2D		RNC 3D	
	Acurácia	f1-Score	Acurácia	f1-Score
1	0,7997	0,7993	0,8465	0,8417
2	0,6610	0,6604	0,8435	0,8380
3	0,6740	0,6736	0,8137	0,8028
4	0,7042	0,6966	0,8134	0,8120
5	0,6485	0,6129	0,8142	0,8089
6	0,6258	0,4880	0,8250	0,8135
7	0,6993	0,6736	0,8621	0,8553
8	0,8000	0,7779	0,8612	0,8514
9	0,7635	0,6829	0,8339	0,8058
10	0,6497	0,6109	0,8286	0,8046
Média	0,7026	0,6676	0,8340	0,8234
DP	0,0640	0,0877	0,0186	0,0208

A Tabela 5 mostra os resultados de acurácia e *f1-score* para cada um dos grupos usando o modelo que possui a arquitetura da Figura 11. Para cada grupo, foram realizados testes utilizando imagens em escala de cinza e imagens coloridas (RGB).

Tabela 5: Resultado dos grupos para RGB e escala de cinza

Grupo	RGB		GRAY	
	Acurácia	f1-Score	Acurácia	f1-Score
1	0,8882	0,8849	0,9078	0,9068
2	0,8612	0,8563	0,8853	0,8783
3	0,8105	0,7997	0,8460	0,8415
4	0,8444	0,8417	0,8278	0,8248
5	0,8561	0,8469	0,8509	0,8474
6	0,8084	0,7912	0,8648	0,8565
7	0,8724	0,8689	0,8814	0,8773
8	0,8456	0,8360	0,8764	0,8694
9	0,8291	0,8019	0,8497	0,8156
10	0,8547	0,8411	0,8598	0,8460
Média	0,8501	0,8368	0,8585	0,8569
DP	0,0241	0,0292	0,0223	0,0304

De acordo com a Tabela 5, o uso de imagens na escala de cinza trouxe melhores resultados se comparado com imagens coloridas. Os grupos com resultados mais discrepantes entre o uso de escala de cinza e RGB foi o 6, contendo uma diferença de acurácia

de 0,0564, quase 6% e 0,0653, quase 7%, para *f1-score*. Existem duas possibilidade de justificativa para esses resultados: a primeira é que 12 vídeos ainda é uma quantidade baixa para trazer uma variabilidade em relação a cores, mesmo com *data augmentation*, e a segunda é que, para esse problema, a informação das cores não ajuda a diferenciar entre as classes 1 e 0.

Durante o treinamento, os dados de validação apresentaram uma mudança inicial nos valores de perda nas primeiras cinco épocas para a maioria dos grupos, enquanto a acurácia continuava a aumentar. Por volta da 15ª época, alguns grupos não apresentavam mais melhorias na acurácia, e os valores de perda começaram a variar significativamente, com alguns casos apresentando apenas aumento, indicando sinais de *overfitting* (Figura 23). Uma possível explicação para esse comportamento é a baixa quantidade de dados e variabilidade, isto é, é necessário ter mais exemplos de vídeos diferentes para ter um treinamento mais estável.

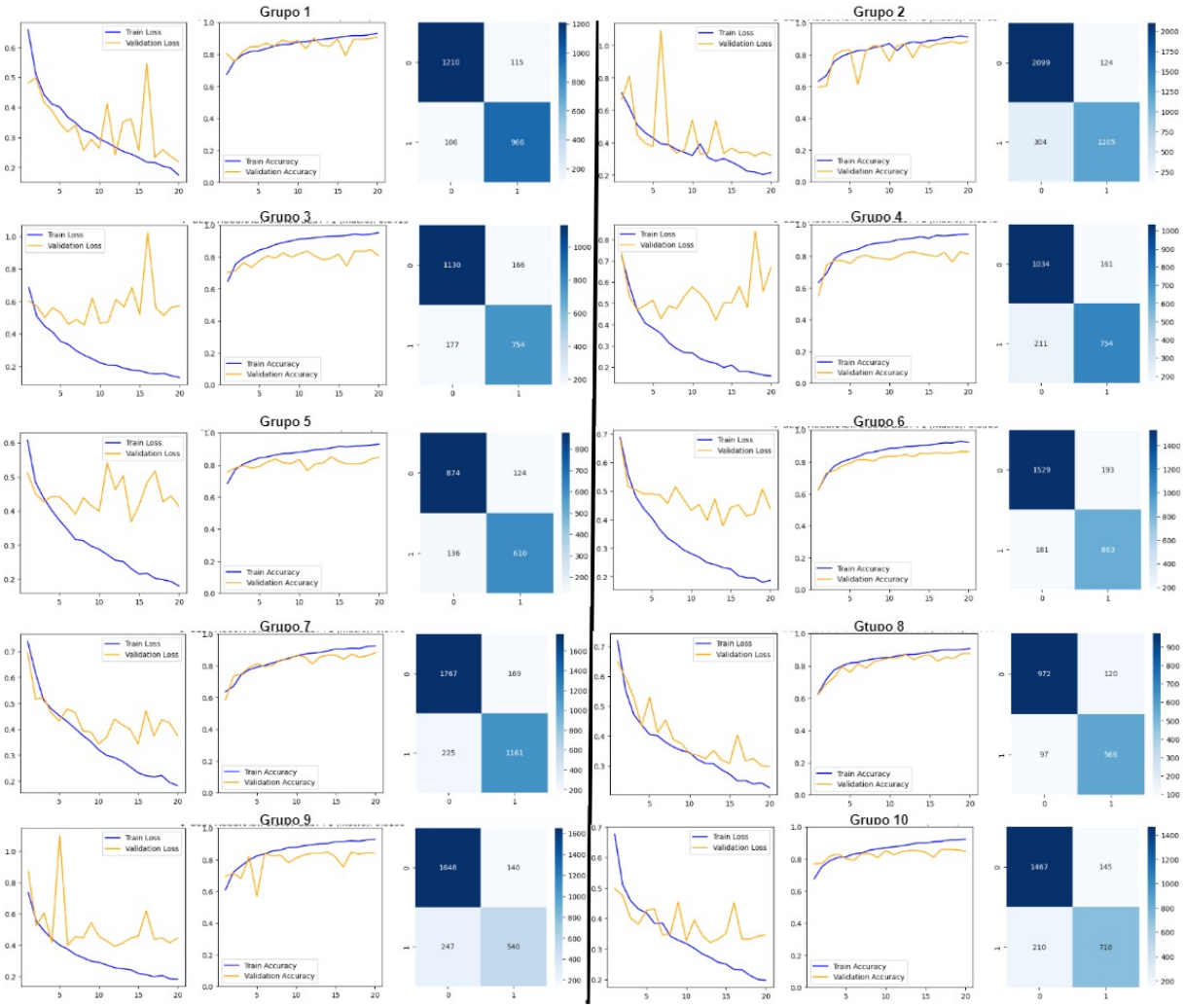

Figura 23: Essa figura contém os gráficos da perda, acurácia e uma matriz de confusão gerada pela melhor época em cada um dos grupos durante o treinamento, usando escala de cinza (Fonte: O autor).

Como foi pontuado na Figura 22, houve problemas na obtenção das imagens dos recortes, quando o movimento era da classe 1 e ocorria no lado superior. Tendo isso em vista, foram criados gráficos que mostram a distribuição da probabilidade para as sub-ações das classes 1 e 0, quando ocorrem tanto no lado superior, quanto inferior. O objetivo desses gráficos é analisar se o modelo possui maior dificuldade para diferenciar movimentos que ocorrem especificamente em um lado.

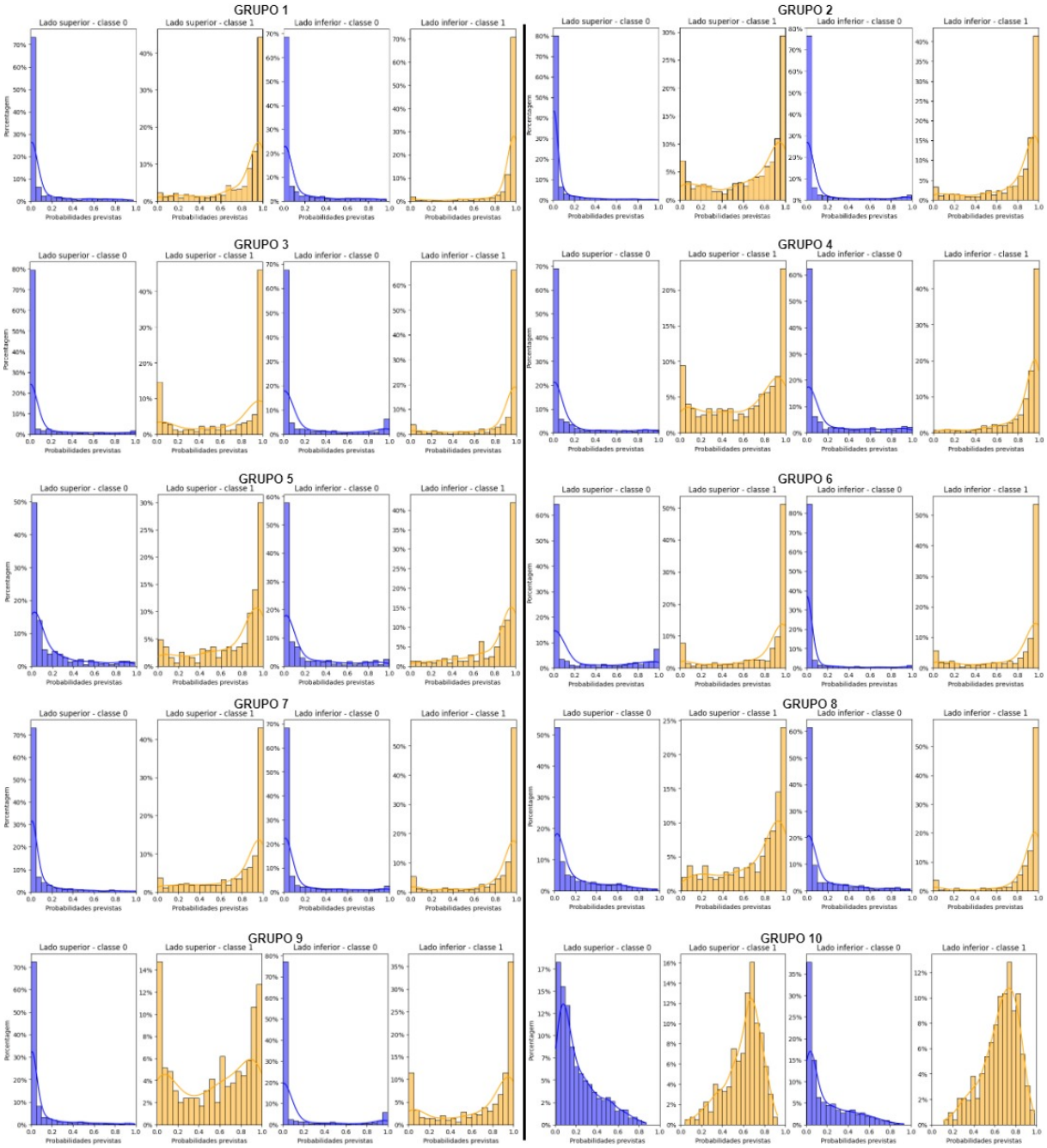

Figura 24: Essa figura ilustra a distribuição da saída do modelo para as sub-ações das classes 1 e 0, ocorrendo tanto no lado superior quanto no inferior (Fonte: O autor).

Como mostrado na Figura 24, o modelo possui maior dificuldade em identificar

movimentos da classe 1, que ocorrem no lado superior da quadra. Isso é possível notar, pois a distribuição da probabilidade de ser um movimento de ataque está mais deslocado para a esquerda, se comparado com os gráficos dos movimentos que ocorrem no lado inferior. Por exemplo, no Grupo 9, mais de 14% das sub-ações da classe 1 tiveram uma probabilidade perto de 0.

Por fim, foi feita uma análise dos movimentos que obtiveram o maior erro em relação a classe para o Grupo 9. As Figuras 25, 26 e 27 mostram a saída do modelo para cada sub-ação, composta por uma sequência de quatro imagens, de movimentos das classes 0 e 1. Cada linha representa uma sequência de frames, com valores numéricos abaixo indicando a probabilidade de ser um movimento de ataque.

Figura 25: Essa figura contém o saída do modelo para cada sub-ação de alguns movimentos que ocorreram no vídeo 7 do Grupo 9 (Fonte: O autor)..

No video 7, todas as jogadas do time azul foram feitas usando o movimento ataque "entre as pernas", enquanto o time branco usou "lançamento com giro". Olhando para a Figura 25, é possível notar que os maiores erros da classe 1 ocorreram no time azul, enquanto os da classe 0 aconteceram no outro. Contudo, existem problemas nos dados do movimento da primeira linha, visto que contém recortes de imagens de outra pessoa e não o jogador, como comentado na Seção 4.1. Esse impasse pode ter ocasionado confusão no modelo, pois não existe um padrão nos números ao longo de todo o movimento, mesmo que estejam longe de 1. Na segunda linha, todas as imagens foram recortadas de maneira correta, porém o modelo teve dificuldade em identificar os primeiros momentos do movimento, já que gerou valores longe de 1, diferente das ultimas imagens.

Os movimentos da classe 0, das duas ultimas linhas, ocorrem após o fim da classe

1. É notório que esses gestos são muito similares a um movimento de ataque, uma vez que eles ocorrem logo após o arremeso. Dessa forma, é compreensível que o modelo tenha uma maior dificuldade em diferençar esses momentos e acabe gerando valores próximos a classe 1, o que ocorre para a maioria das sub-ações dessa característica.

Figura 26: Essa figura contém o saída do modelo para cada sub-ação de alguns movimentos que ocorreram no vídeo 14 do Grupo 9 (Fonte: O autor).

Nos exemplos do vídeo 14, o modelo, de modo geral, classificou corretamente as probabilidades de acordo com a classe esperada. No entanto, em alguns momentos, ocorreram desvios. Por exemplo, no início do movimento da segunda linha (classe 1) e da última linha (classe 0), as probabilidades ficaram próximas de zero e um, respectivamente, indicando confusão do modelo. Além disso, um frame isolado, com um jogador que não faz parte do movimento, parece ter introduzido ruído, contribuindo para valores próximos da classe incorreta.

Na primeira linha (Figura 27), a maioria dos valores estão próximos da classe 1, exceto pela segunda sub-ação, que apresenta um valor de 0,101632, próximo de zero. Na segunda linha, grande parte dos valores estão próximos da classe 0, porém não é possível notar uma caracterísitca no movimento do jogador que indique essa dificuldade. No primeiro movimento da classe 0, os valores iniciais estão corretos, mas, nas sub-ações finais, as probabilidades aumentam. Esse comportamento pode ter sido causado por uma diferença no recorte dos frames em relação aos anteriores. Por fim, todas as sub-ações da última linha foram confundidas com a classe 1, provavelmente devido à semelhança com um ataque, como mencionado na análise da Figura 25, o que justifica a confusão do modelo.

De maneira geral, o modelo apresentou resultados acima de 0,8 tanto na acurácia, quanto no *f1-score*, no método de validação, mostrando que é posível utilizar esse método

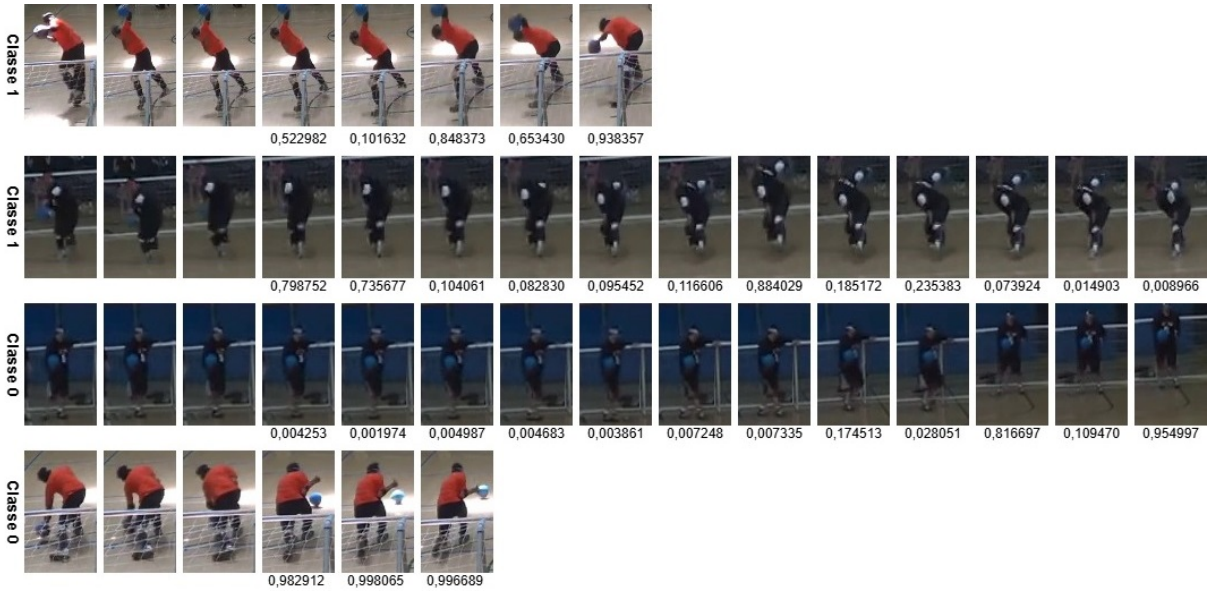

Figura 27: Essa figura contém o saída do modelo para cada sub-ação de alguns movimentos que ocorreram no vídeo 16 do Grupo 9 (Fonte: O autor).

para identificar tipos de movimentos. No futuro, esses resultados podem ser aprimorados com o aumento da base de dados e melhorias no modelo.

4.3 Detecção dos quiques

A tabela 6 contém os resultados de acurácia e *f1-score*, para cada um dos grupos e modelos testados.

Tabela 6: Resultados de Acurácia e *f1-score* para KNN, DT, SVM, e NB

Grupo	KNN		DT		SVM		NB	
	Acurácia	f1-Score	Acurácia	f1-Score	Acurácia	f1-Score	Acurácia	f1-Score
1	0,9257	0,8306	0,9125	0,7647	0,9268	0,8101	0,5260	0,4600
2	0,9376	0,8526	0,9175	0,7965	0,9264	0,8051	0,8325	0,6479
3	0,9360	0,8406	0,9161	0,7979	0,8934	0,5740	0,4666	0,4268
4	0,9278	0,8358	0,9268	0,8271	0,9275	0,8178	0,5896	0,5035
5	0,9321	0,8392	0,9287	0,8324	0,9250	0,7922	0,7906	0,6408
6	0,9215	0,8231	0,9338	0,8430	0,9212	0,8000	0,7612	0,6201
7	0,9327	0,8325	0,9190	0,8018	0,9246	0,7926	0,8680	0,6978
8	0,9403	0,8598	0,9174	0,8050	0,8809	0,4684	0,3096	0,3023
9	0,9156	0,8172	0,9222	0,8308	0,9236	0,8278	0,8543	0,6948
10	0,9291	0,8236	0,9317	0,8298	0,9254	0,7930	0,6521	0,5320

Em geral, o KNN apresentou os melhores resultados tanto em termos de acurácia, quanto de *f1-score*, variando de 0,9156, a 0,9403, na acurácia e de 0,8172, a 0,8598, no f1-score. É interessante pontuar que os resultados do *f1-score* foram consistentes entre os diferentes grupos, indicando que foi capaz de se sair melhor no desbalanceamento natural dos dados, ou seja, possui um equilíbrio entre a identificação da classe positiva e negativa.

Por outro lado, o Decision Tree (DT) e o SVM apresentaram acurácias comparáveis ou, em alguns casos, ligeiramente superiores ao KNN. No entanto, o desempenho desses algoritmos não se manteve equilibrado entre os grupos, especialmente no caso do SVM.

Por exemplo, ele obteve *f1-scores* muito baixos, como 0,5740 no Grupo 3 e 0,4684 no Grupo 8, o que indica uma dificuldade maior em lidar com o desbalanceamento.

O Naive Bayes (NB) foi o algoritmo com o pior desempenho, chegando a ter valores próximos a 0,3 para ambas as métricas. Isso pode ser atribuído à suposição simplificada de independência das características.

4.4 Medição de velocidade

Os resultados do cálculo da velocidade média apresentaram variações significativas, em relação às medidas obtidas pelo radar (Tabela 7). Embora algumas jogadas, como a 10 e a 72, tenham mostrado valores próximos entre o sistema e o radar, a maioria das velocidades calculadas apresentou uma diferença superior a 20%. As jogadas marcadas com o valor "X"correspondem àquelas em que o sistema não conseguiu identificar o início do movimento, ou não detectou os pontos de quique. Em particular, as Jogadas 31 e 69 apresentaram falhas bruscas na detecção precisa do fim do movimento de ataque.

Tabela 7: Velocidade média obtida para cada uma das jogadas.

Nº Jogada	Radar km/h	Resultado km/h	Diferença %	Nº Jogada	Radar km/h	Resultado km/h	Diferença %
1	54	90,5	67,59%	39	39	X	-
2	35	36,8	5,14%	40	39	X	-
3	54	X	-	41	49	X	-
4	42	44,2	5,24%	42	56	X	-
5	31	42,8	38,06	43	52	59,2	13,85
6	46	42,9	6,74	44	41	43	4,88
7	46	41,6	9,57	45	28	51,7	84,64
8	41	39,5	3,66	46	45	45,5	1,11
9	60	X	-	47	60	39,1	34,83
10	45	45,7	1,56	48	33	35,4	7,27
11	57	X	-	49	32	61,3	91,56
12	40	42,4	6,00	50	58	26	55,17
13	36	50,2	39,44	51	65	58,7	9,69
14	57	44,7	21,58	52	56	77,3	37,05
15	48	70,5	46,88	53	59	X	-
16	45	47,5	5,56	54	35	43,4	24,00
17	53	76,5	44,34	55	52	62,5	20,19
18	54	48,9	9,44	56	38	41,1	8,16
19	40	37,4	6,50	57	52	49	5,77
20	41	28,9	29,51	58	33	34,3	3,94
21	39	38,3	1,79	59	42	49,7	18,33
22	38	29,8	21,58	60	63	77,3	22,70
23	58	42,7	26,38	61	48	46,8	2,50
24	59	48,8	17,29	62	66	26,1	60,45
25	54	55	1,85	63	63	47,9	23,97
26	54	47	12,96	64	67	80	19,40
27	58	33	43,10	65	64	51,5	19,53
28	36	X	-	66	31	60,6	95,48
29	31	54,6	76,13	67	33	39,2	18,79
30	36	56	55,56	68	48	59,1	23,13
31	53	210,1	296,42	69	8	70,1	776,25
32	53	67,5	27,36	70	47	24,9	47,02
33	63	90,7	44,13	71	62	38	38,71
34	51	45	11,76	72	33	33,4	1,21
35	41	X	-	73	56	39	30,36
36	48	49,3	2,71	74	48	42,2	12,08
37	32	45,6	42,50	75	51	54,2	6,27
38	36	43	19,44	76	63	5	92,06

(i) As células em verde indicam uma diferença percentual abaixo de 10%. (ii) As células em laranja indicam uma diferença entre 10% e 40%. (iii) As células em vermelho indicam uma diferença acima de 40%.

Na Jogada 3 o modelo de RAH não conseguiu fornecer valores suficientes para

identificar o início do movimento de ataque a partir da SW. A Figura 28 mostra algumas sequências de frames do ocorrido.

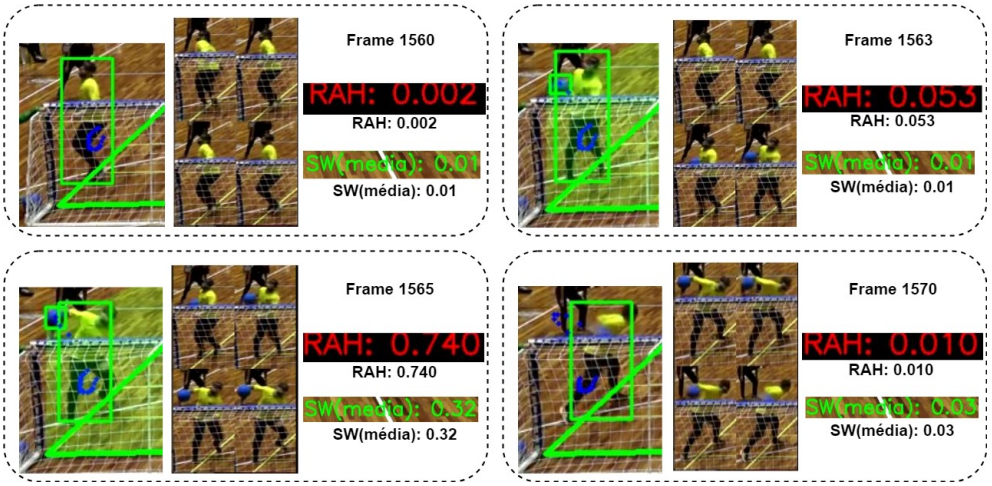

Figura 28: Cada bloco na imagem contém o jogador que está com a bola, a entrada para o modelo de RAH, a saída gerado pelo modelo e o valor da média da SW (Fonte: O autor).

Na maioria das vezes, o modelo de RAH não conseguiu produzir valores acima de 0,5, como mostrado nos *frames* 1560, 1563 e 1570. No entanto, mesmo gerando um valor de 0,74 no *frame* 1565, não foi o suficiente para chegar a uma média acima de 0,5 na SW.

Na Jogada 11, a bola que estava fora acabou interferindo, visto que ficou dentro da máscara e passou pelo filtro de confiança (Figura 29).

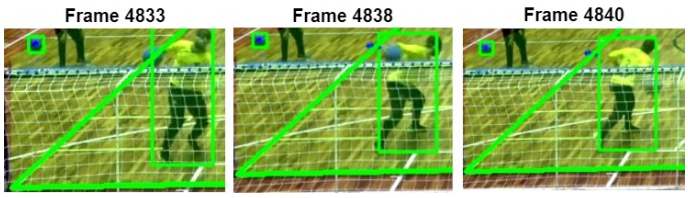

Figura 29: Imagem contendo os frames em que a bola fora da quadra atrapalhou a identificação do movimento (Fonte: O autor).

Na Jogada 31 (Figura 30), houve um atraso de 7 *frames* na identificação do fim do movimento de ataque. Além disso, entre o último ponto classificado como quique e o fim do movimento, havia um intervalo de 3 *frames*. Assim, a quantidade total de *frames* entre o fim do movimento e o quique deveria ser 10, somando os 7 do atraso e os 3 do intervalo. Utilizando os valores do comprimento, ΔSy, e largura, ΔSx, obtidos pelo sistema, é possível calcular qual seria a velocidade, considerando que o momento exato do arremesso tenha sido identificado corretamente.

Para ilustrar o impacto desse atraso, calculou-se a velocidade da bola em duas situações: a primeira, utilizando os 3 *frames* registrados pelo sistema, e a segunda, simu-

lando a identificação correta do arremesso, com os 10 *frames*. A forma de como é feito o cálculo, pode ser visualizado na Figura 18.

$$\Delta S = \sqrt{1,5^2 + 5,64^2} => \Delta S = 5,8360$$

Para o caso de 3 *frames*:

$$\Delta t = \frac{3}{30} = 0,1 \,\text{segundos} => v = \frac{5,8360}{0,1} = 58,36 \,\text{m/s} => 58,36 \times 3,6 = 210,09 \,\text{km/h}$$

Para o caso de 10 *frames*:

$$\Delta t = \frac{10}{30} = 0,33 \,\text{segundos} => v = \frac{5,8360}{0,33} = 17,68 \,\text{m/s} => 17,68 \times 3,6 = 63,65 \,\text{km/h}$$

Dessa forma, ao utilizar os 3 *frames*, a velocidade média estimada foi de 210,09 km/h, valor próximo ao obtido pelo sistema, enquanto, ao utilizar os 10 *frames*, simulando a identificação correta do arremesso, a velocidade foi de 63,65 km/h, mais próxima do valor registrado pelo radar. Esses resultados demonstram a importância de identificar com precisão o momento exato em que o jogador arremessa a bola. Uma diferença de apenas 7 *frames* resultou em uma diminuição de 70% na velocidade.

Figura 30: Imagem contendo algumas informações a respeito da Jogada 31 (Fonte: O autor).

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou uma abordagem para a análise automática de partidas de *goalball* utilizando visão computacional e aprendizado de máquina. Foi desenvolvido um sistema capaz estimar a velocidade média da bola, utilizando reconhecimento de ação e detecção de quiques, em cojunto com a técnica de *sliding window*.

Os experimentos demonstraram que o modelo de RAH obteve bons valores de acurácia e *f1-score*. A utilização de imagens em escala de cinza proporcionou melhoria nos resultados, em comparação com imagens coloridas, sugerindo que a cor não desempenha um papel significativo nesse contexto. Na detecção de quiques, o algoritmo KNN apresentou o melhor desempenho, superando outras abordagens como DT e SVM, porém ainda é possível explorar como o ajuste dos parâmetros pode melhorar o resultado. Além disso, a estimativa da velocidade média da bola mostrou-se confiável em diversas jogadas, isso mostra que o sistema pode ser utilizado para fazer análises mais precisas no futuro, com as devidas melhorias .

No entanto, algumas limitações foram identificadas. O sistema enfrenta dificuldades para reconhecer com precisão o exato momento em que o jogador arremessa a bola. Além disso, o modelo atual só funciona em vídeos capturados com a câmera posicionada na parte traseira da quadra, o que limita sua aplicabilidade em outras perspectivas. Outro ponto crítico é a dependência de um modelo robusto de detecção de objetos, uma vez que o sucesso na identificação dos jogadores e da bola influencia diretamente o desempenho do reconhecimento de ações e dos quiques.

Como trabalhos futuros, recomenda-se explorar a utilização de modelos de menor complexidade para o reconhecimento de ações, visando um equilíbrio entre precisão e eficiência computacional, fator crucial para aplicações em tempo real. Além disso, testar diferentes representações de entrada, como o uso de esqueletos de pontos obtidos por estimativa de pose, poderia oferecer comparações em relação ao método baseado em imagens utilizado neste trabalho.

Em resumo, o estudo conseguiu demonstrar que é possível utilizar visão computacional em partidas de *goalball* com a finalidade de obter informações a respeito das jogadas.

REFERÊNCIAS

[1] M. Rana and M. Bhushan, "Classifying breast cancer using transfer learning models based on histopathological images," *Neural Computing and Applications*, vol. 35, no. 19, pp. 14243–14257, 2023.

[2] J. Kotwal, D. Kashyap, and D. Pathan, "Agricultural plant diseases identification: From traditional approach to deep learning," *Materials Today: Proceedings*, vol. 80, pp. 344–356, 2023. 3rd International Congress on Mechanical and Systems Engineering (CAMSE 2022).

[3] S. Hangaragi, T. Singh, and N. N, "Face detection and recognition using face mesh and deep neural network," *Procedia Computer Science*, vol. 218, pp. 741–749, 2023. International Conference on Machine Learning and Data Engineering.

[4] N. Jin and X. Zhan, "Big data analytics for image processing and computer vision technologies in sports health management," *Technology and Health Care*, vol. 32, no. 5, pp. 3167–3187, 2024.

[5] R. Kumar and S. Kumar, "A survey on intelligent human action recognition techniques," *Multimedia Tools and Applications*, vol. 83, no. 17, pp. 52653–52709, 2024.

[6] Z. Sun, Q. Ke, H. Rahmani, M. Bennamoun, G. Wang, and J. Liu, "Human action recognition from various data modalities: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3200–3225, 2023.

[7] X. Ren, F. Zhang, H. Zhu, and Y. Liu, "Quad-kernel deep convolutional neural network for intra-hour photovoltaic power forecasting," *Applied Energy*, vol. 323, p. 119682, 2022.

[8] G. Huang, S. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2752–2761, 2018.

[9] J. Redmon, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[11] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using yolo: Challenges, architectural successors, datasets and applications," *multimedia Tools and Applications*, vol. 82, no. 6, pp. 9243–9275, 2023.

[12] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, 2023.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, p. 1735–1780, Nov. 1997.

[14] I. D. Mienye, T. G. Swart, and G. Obaido, "Recurrent neural networks: A comprehensive review of architectures, variants, and applications," *Information*, vol. 15, no. 9, 2024.

[15] J. Li and S. J. Qin, "Applying and dissecting lstm neural networks and regularized learning for dynamic inferential modeling," *Computers Chemical Engineering*, vol. 175, p. 108264, 2023.

[16] G. Bilquise, S. Ibrahim, and K. Shaalan, "Emotionally intelligent chatbots: a systematic literature review," *Human Behavior and Emerging Technologies*, vol. 2022, no. 1, p. 9601630, 2022.

[17] J. Shah, D. Vaidya, and M. Shah, "A comprehensive review on multiple hybrid deep learning approaches for stock prediction," *Intelligent Systems with Applications*, vol. 16, p. 200111, 2022.

[18] J. Oruh, S. Viriri, and A. Adegun, "Long short-term memory recurrent neural network for automatic speech recognition," *IEEE Access*, vol. 10, pp. 30069–30079, 2022.

[19] G. Lee, Y. Yoon, and K. Lee, "Anomaly detection using an ensemble of multi-point lstms," *Entropy*, vol. 25, no. 11, 2023.

[20] R. Dey, R. C. Balabantaray, and S. Mohanty, "Sliding window based off-line handwritten text recognition using edit distance," *Multimedia Tools and Applications*, vol. 81, no. 16, pp. 22761–22788, 2022.

[21] J. Jing, X. Pang, Z. Pan, F. Fan, and Z. Meng, "Classification and identification of epileptic eeg signals based on signal enhancement," *Biomedical Signal Processing and Control*, vol. 71, p. 103248, 2022.

[22] M. LIU, A. Zeng, M. Chen, Z. Xu, Q. LAI, L. Ma, and Q. Xu, "Scinet: Time series modeling and forecasting with sample convolution and interaction," in *Advances in Neural Information Processing Systems* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), vol. 35, pp. 5816–5828, Curran Associates, Inc., 2022.

[23] W. Lao, C. Cui, D. Zhang, Q. Zhang, and Y. Bao, "Computer vision-based autonomous method for quantitative detection of loose bolts in bolted connections of steel structures," *Structural Control and Health Monitoring*, vol. 2023, no. 1, p. 8817058, 2023.

[24] K. Wang, B. Fang, J. Qian, S. Yang, X. Zhou, and J. Zhou, "Perspective transformation data augmentation for object detection," *IEEE Access*, vol. 8, pp. 4935–4943, 2020.

[25] H. Nguyen, "A high-performance approach for irregular license plate recognition in unconstrained scenarios," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 3, 2023.

[26] T. Araújo, P. Chagas, J. Alves, C. Santos, B. Sousa Santos, and B. Serique Meiguins, "A real-world approach on the problem of chart recognition using classification, detection and perspective correction," *Sensors*, vol. 20, no. 16, 2020.

[27] T. Ikeda, T. Araie, A. Kakimoto, K. Ninomiya, and K. Ikeda, "Development of multimodal strategy board for improving competitiveness in goalball," *International Journal of Modeling and Optimization*, vol. 8, no. 4, pp. 212–216, 2018.

[28] K. Host, M. Pobar, and M. Ivasic-Kos, "Analysis of movement and activities of handball players using deep neural networks," *Journal of Imaging*, vol. 9, no. 4, 2023.

[29] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, A. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," *ArXiv*, vol. abs/1705.06950, 2017.

[30] Y. Li, Y. Liu, R. Yu, H. Zong, and W. Xie, "Dual attention based spatial-temporal inference network for volleyball group activity recognition," *Multimedia Tools and Applications*, vol. 82, no. 10, pp. 15515–15533, 2023.

[31] R. Li and B. Bhanu, "Energy-motion features aggregation network for players' fine-grained action analysis in soccer videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 34, no. 2, pp. 955–972, 2024.

[32] C. Li, J. Zhang, S. Wu, X. Jin, and S. Shan, "Hierarchical compositional representations for few-shot action recognition," *Computer Vision and Image Understanding*, vol. 240, p. 103911, 2024.

[33] R. M and R. M. R. Guddeti, "Human action recognition using multi-stream attention-based deep networks with heterogeneous data from overlapping sub-actions," *Neural Comput. Appl.*, vol. 36, p. 10681–10697, Mar. 2024.

[34] K. Host and M. Ivašić-Kos, "An overview of human action recognition in sports based on computer vision," *Heliyon*, vol. 8, no. 6, p. e09633, 2022.

[35] F. Wu, Q. Wang, J. Bian, N. Ding, F. Lu, J. Cheng, D. Dou, and H. Xiong, "A survey on video action recognition in sports: Datasets, methods and applications," *IEEE Transactions on Multimedia*, vol. 25, pp. 7943–7966, 2023.

[36] S. Oehri, N. Ebert, A. Abdullah, D. Stricker, and O. Wasenmüller, "Genformer–generated images are all you need to improve robustness of transformers on small datasets," *arXiv preprint arXiv:2408.14131*, 2024.

[37] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018.

[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[39] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, 2020.

[40] K. A. M. T. Villarim, "Detecção de ruas de barro a partir de imagens aéreas," 2021. Acessado em: 05/10/2024.

[41] R. Rai and D. S. Sisodia, "Real-time data augmentation based transfer learning model for breast cancer diagnosis using histopathological images," in *Advances in Biomedical Engineering and Technology* (A. A. Rizvanov, B. K. Singh, and P. Ganasala, eds.), (Singapore), pp. 473–488, Springer Singapore, 2021.

[42] A. Mumuni and F. Mumuni, "Data augmentation: A comprehensive survey of modern approaches," *Array*, vol. 16, p. 100258, 2022.

[43] S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, and F. Shen, "Image data augmentation for deep learning: A survey," *arXiv preprint arXiv:2204.08610*, 2022.

[44] R. Lin, "Analysis on the selection of the appropriate batch size in cnn neural network," in *2022 International Conference on Machine Learning and Knowledge Engineering (MLKE)*, pp. 106–109, IEEE, 2022.

[45] P. Contributors., "Bcewithlogitsloss," 2021. Acessado em: 05/10/2024.

[46] K. Sergey, "Tennis analysis using deep learning and machine learning.," 2023. Acessado em: 10/05/2024.

[47] V. Sheth, U. Tripathi, and A. Sharma, "A comparative analysis of machine learning algorithms for classification purpose," *Procedia Computer Science*, vol. 215, pp. 422–431, 2022. 4th International Conference on Innovative Data Communication Technology and Application.

[48] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[49] M. Ivasic-Kos and M. Pobar, "Building a labeled dataset for recognition of handball actions using mask r-cnn and stips," in *2018 7th European Workshop on Visual Information Processing (EUVIP)*, pp. 1–6, 2018.