**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**

**FELIPE ARISI**

# RECONHECIMENTO AUTOMÁTICO DE POSE EMPREGADO NA ESTIMAÇÃO DA POSIÇÃO DA BOLA EM PARTIDAS DE FUTSAL

**PATO BRANCO**

**2022**

**FELIPE ARISI**

# RECONHECIMENTO AUTOMÁTICO DE POSE EMPREGADO NA ESTIMAÇÃO DA POSIÇÃO DA BOLA EM PARTIDAS DE FUTSAL

# AUTOMATIC POSE RECOGNITION USED IN ESTIMATING BALL POSITION IN FUTSAL GAMES

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Dalcimar Casanova

Coorientador: Prof. Dr. Pablo Guaterio Cavalcanti

**PATO BRANCO**

**2022**

**FELIPE ARISI**

# RECONHECIMENTO AUTOMÁTICO DE POSE EMPREGADO NA ESTIMAÇÃO DA POSIÇÃO DA BOLA EM PARTIDAS DE FUTSAL

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Engenharia da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 13/maio/2022

---

Prof. Dr. Dalcimar Casanova
Doutorando em Física Computacional
Universidade Tecnológica Federal do Paraná (UTFPR) - Pato Branco

---

Prof. Dr. Marco Antonio de Castro Barbosa
Doutorado em Informática
Universidade Tecnológica Federal do Paraná (UTFPR) - Pato Branco

---

Prof. Dra. Rubia Eliza de Oliveira Schultz Ascari
Doutorado em Informática
Universidade Tecnológica Federal do Paraná (UTFPR) - Pato Branco

**PATO BRANCO**
**2022**

Dedico este trabalho aos meus pais Silvionei
Paulo Arisi e Luciane Cristiane Arisi e também
ao meu querido irmão Gabriel Henrique Arisi.

## AGRADECIMENTOS

Gostaria de agradecer a todos que de alguma maneira estiveram comigo durante esse tempo.

Aos meus amigos que escutaram minhas reclamações, me deram forças para continuar e estão ao meu lado.

A toda minha família pelo suporte, carinho e compreensão, os quais foram fundamentais para que eu conseguisse iniciar e concluir esse ciclo.

A UTFPR que me proporcionou tantas oportunidades, alegrias e desafios. Também a todos os professores que contribuíram para meu crescimento pessoal e profissional.

E ao Prof. Dr. Dalcimar Casanova por ter me guiado na faculdade e neste projeto.

A todos, meu sincero, muito obrigado!

## RESUMO

Determinar a localização da bola em partidas de futsal é de grande importância para relatórios técnicos de times, para transmissões esportivas e quaisquer outras análises pertinentes. Entretanto, este reconhecimento não transcorre de uma maneira trivial devido ao seu pequeno tamanho, a grande velocidade que ela pode adquirir e também o fato dela ficar obstruída pelos jogadores. Isto acaba tornando-se uma tarefa difícil para métodos de reconhecimentos de visão computacional convencionais ou fazendo necessário o uso de inúmeros equipamentos no local. Um recurso plausível a ser utilizado, porém pouco aproveitado para esta demanda, é a pose dos jogadores durante as partidas. Esse projeto realiza a aquisição dessas poses como também avaliar uma relação entre elas e a posição original da bola, empregando assim algoritmos de regressão para prever a posição dela. Realizando essa estimativa de maneira genérica independente de câmera ou do local de gravação.



**Palavras-chave:** detecção de pose; overffiting; estimativa da localização da bola.

**ABSTRACT**

Determining the location of the ball in futsal matches is of great importance for team technical reports, sports broadcasts and any other pertinent analysis. However, this recognition does not take place in a trivial way, due to its small size, the great speed it can acquire and also the fact that it is obstructed by players, it becomes a difficult task for conventional computer vision recognition methods or doing the use of numerous equipment. A plausible resource to be used, but hardly used for this demand, is the pose of the players during the matches. This project performs the acquisition of these poses as well as evaluating a relation between them and the original position of the ball, thus employing regression algorithms to predict its position. Performing this estimation in a generic way independent of camera or recording location.

**Keywords:** pose detection; overffiting; estimation of ball location.

## LISTA DE FIGURAS

## LISTA DE TABELAS

**SUMÁRIO**

# 1 INTRODUÇÃO

Em esportes coletivos de alto desempenho, estatísticas servem como parâmetro para equipes desenvolverem treinos e aprimorarem táticas. Várias modalidades se beneficiam dessa análise, que até um tempo atrás era realizada de forma manual, trabalho esse bastante laborioso, pouco preciso e pouco detalhista.

No anseio de melhorar a qualidade das análises grandes esforços vem sendo realizados no desenvolvimento de sistemas automatizados, principalmente aqueles baseados em análise de imagens (CHEN *et al.*, 2021). Exemplos de sistemas automáticos em estatísticas esportivas podem ser vistos nos trabalhos de Link (2018) e Morgulev, Azar e Lidor (2018).

Uma abordagem recorrente, e que vem ganhando espaço na literatura especializada, são trabalhos que tentam estimar/reconhecer a posição da bola.

O reconhecimento eficaz da bola permite uma avaliação precisa e ágil de vários fatores esportivos, tais como, número de passes e finalizações, mapa de calor de um jogo, entre outras. Essas informações são de suma importância para emissoras de televisão, para clubes e também comissões técnicas (BORG, 2007).

Outro ponto há de ser beneficiado por este método é a tecnologia de validação de gols. A Fifa certificou o *goal-line technology* (GLT) em fevereiro de 2017 para mais 108 estádios, na maioria destes campos o sistema é baseado em um conjunto de sete câmeras posicionadas de maneira estratégica tornando assim o processo de validação muito complexo (THOMAS *et al.*, 2017).

Todavia o rastreamento da bola por análise de video é árduo em virtude de seu pequeno tamanho, velocidade obtida durante as partidas e, em vários momentos ela fica obstruída pelos jogadores tornando-se, por vezes, uma tarefa bastante complicada (MAKSAI; WANG; FUA, 2016).

Para contornar essa dificuldade e conseguir realizar um rastreamento eficiente modelos matemáticos tentam estimar o posicionamento da bola de forma indireta, via posicionamento dos jogadores. Dentre os trabalhos com essa abordagem (MAKSAI; WANG; FUA, 2016) emprega uma modelagem de grafo da interação jogador bola e (WANG *et al.*, 2014) estima a posição da bola indiretamente, via reconhecimento do jogador com posse da mesma, estimando posteriormente a possível trajetória durante passes e finalizações.

Em ambos os trabalhos um sistema multi-câmeras sincronizadas foi necessário, o que encarece e dificulta sua aplicabilidade prática por diversos motivos.

A hipótese desse trabalho é que seria possível estimar a posição da bola utilizando uma única câmera, dispensando assim a complexidade de aquisição das imagens, diminuindo custo financeiro e computacional, mantendo um erro aceitável.

método é elaborado com base nas interações entre a bola e os jogadores. Outra questão permitida pelo modelo é a generalização para vários tipos de modalidades (MAKSAI; WANG; FUA, 2016). No entanto, para uma elaboração eficiente foi necessário o uso de várias câmeras

sincronizadas o que resultou no aumento do custo e da complexidade para a resolução do problema.

O objetivo é estimar o posicionamento da bola de forma indireta, utilizando como mecanismo de inferência a pose e posição dos jogadores, com um aparato de aquisição único.

Para tal desenvolvimento é empregado o algoritmo AlphaPose (FANG *et al.*, 2017) para a detecção da pose dos jogadores em partidas de futsal e o XGBoost como o método de regressão para estimar a posição da bola, *3Way holdout e random subsampling* para a validação e os resultados do erro médio absoluto e quadrático como métricas de avaliação.

O desenvolvimento e os resultados podem ser visualizados no seguinte repositório do *GitHub*: https://github.com/FelipeArisi/estimativa_posicao_bola.

## 1.1 Objetivo Geral

Estimar a localização da bola em partidas de futsal a partir da detecção automática de pose dos jogadores em quadra com um único aparato de aquisição.

## 1.2 Objetivos específicos

- Representar visualmente a predição da bola;
- Segmentar automaticamente os jogadores na quadra;
- Verificar quais pontos de pose são mais relevantes para criação do modelo.

# 2 REVISÃO DE LITERATURA

Todo o desenvolvimento do projeto é baseado na pose dos jogadores em partidas de futsal. Por esse motivo, o objetivo deste capítulo é descrever como é realizada a aquisição desses dados. Tal como a fundamentação para a escolha do método de regressão e as etapas para aperfeiçoar os resultados.

## 2.1 Métodos de estimativa de pose

A escolha de um algoritmo de reconhecimento de pose eficaz é de extrema importância para a elaboração do método para estimativa da localização da bola. Alguns algoritmos que desempenham esta tarefa com uma boa acurácia na detecção são o AlphaPose e o OpenPose.

Segundo Fang *et al.* (2017), o AlphaPose consiste em uma estrutura de duas etapas, no primeiro estágio é utilizado um mecanismo para realizar o reconhecimento das pessoas na cena e delimitá-las em caixas. Na segunda etapa, cada delimitação do passo anterior é empregada em um sistema no qual se estima a pose, esse método é divido em vários fragmentos: pré-processamento em relação à imagem da pessoa, a detecção da pose e retroceder à imagem original.

Já, de acordo com Cao *et al.* (2018), o método OpenPose utiliza uma técnica diferente para realizar a estimativa. Aplica-se ferramenta não paramétrica chamada *Part Affinity Fields* (PAFs). Essa técnica consiste em aprender cada parte do corpo, ou seja, diferente do AlphaPose este método não processa cada pessoa individualmente, mas sim cada fragmento.

Utilizando como base de dados o COCO (VEIT *et al*., 2016) o algoritmo escolhido para uso neste projeto foi o AlphaPose.Tendo em vista que esse método apresenta um desempenho melhor que o OpenPose, pode-se observar tal comportamento na Tabela 1.

| | AP 1 | AP50 | AP75 | APM | APL |
|---|---|---|---|---|---|
| AlphaPose | 71.0 | 87.9 | 77.7 | 69.0 | 75.2 |
| OpenPose | 64.2 | 86.2 | 70.1 | 61.0 | 68.8 |

**Tabela 1 – Comparação de entre algoritmos detectores de pose. Onde AP é a relação entre a escala das pessoas e os resultados são a acurácia nesse modelo.**

**Fonte: Adaptada de Cao *et al.* (2018).**

Além dos resultados listados em *benchmarks* a escolha também foi fundamentada em testes iniciais, onde foram realizados recortes de vídeos em partidas de futsal e aplicados ambos os algoritmos. O exemplo da Figura 1 representa a execução do AlphaPose para um instante da execução, cada vértice das linhas coloridas sobre as pessoas, representa um ponto da pose. Na Figura 2 está presente a mesma representação e no mesmo momento, porém com o algoritmo OpenPose.

Neste instante percebe-se que o AlphaPose conseguiu detectar e construir a pose de mais jogadores comparado ao OpenPose. E tal fato ocorre para outros recortes da partida, tornando assim a utilização do OpenPose inviável para o projeto.

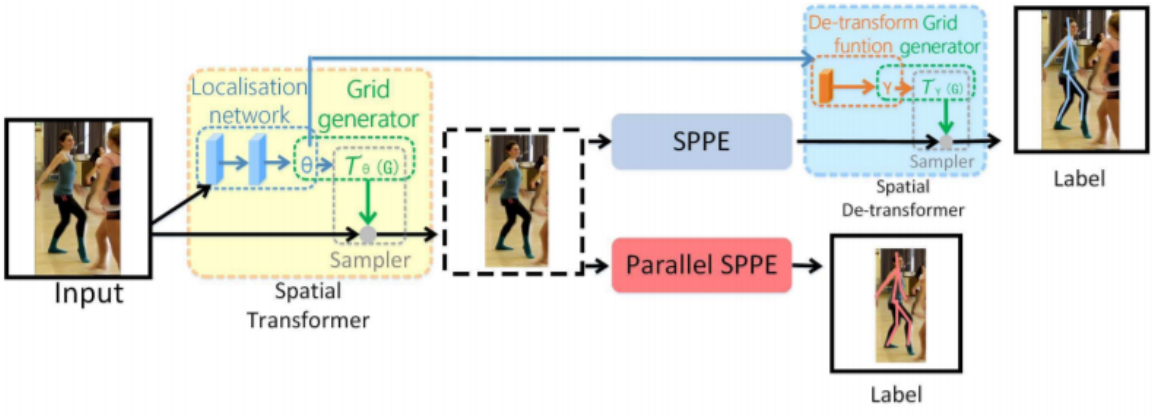

**Fonte: O autor.**

**Figura 1 – Estimativa utilizando método AlphaPose para *frame* de vídeo. Quando há sucesso na detecção das poses, estas são mapeadas pelas linhas coloridas nas pessoas da imagem.**


**Fonte: O autor.**

**Figura 2 – Estimativa utilizando método OpenPose para *frame* de vídeo. Processo similar a Figura 1. O algoritmo não obteve sucesso em realizar a construção da pose nos jogadores na parte superior da quadra.**

A estrutura do AlphaPose é dividida em três módulos *Spatial transformer networks (STN), Stacked hourglass networks for human pose estimation (SPPE) e Spatial de-transformer networks (SDTN)* que juntos formam o conjunto de reconhecimento da pose. A delimitação das pessoas em caixas é realizada através de uma rede neural chamada YOLO (REDMON; FARHADI, 2018). Após este processo, a estrutura é definida pela Figura 3. Cada uma das etapas serão descritas nos próximos tópicos, além de conceitos utilizados durante o processo.




**Fonte: (FANG *et al.*, 2017).**

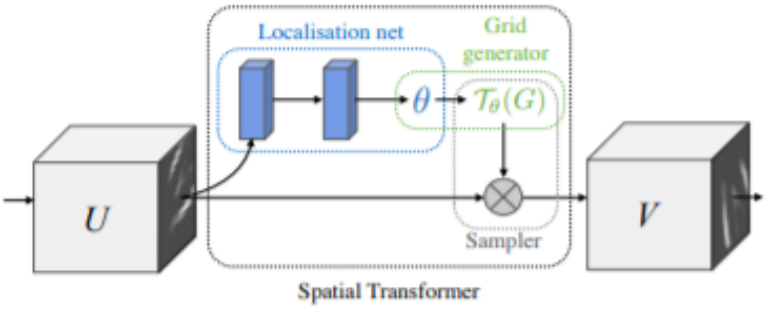
**Figura 3 – Estrutura do AlphaPose. Como entrada uma única pessoal, a qual transcorre pelos módulos Spatial Transformer, SPPE e Spatial De-transformer, resultando em pontos com a pose da pessoa.**

## 2.2 Módulo spatial transformer networks (STN)

De acordo com Jaderberg *et al.* (2015), o método *Spatial Tranformer Networks* consiste na manipulação inicial dos dados (escala, rotação e distorção), sem a necessidade de alteração dos parâmetros da rede neural.

Essa etapa do AlphaPose é necessária pois para a detecção da pose em si, é empregado outro algoritmo o qual é sensível aos parâmetros de entrada, ou seja, conforme a posição da pessoa na imagem, pode afetar seu resultado da construção de sua pose.

Por esse motivo, é utilizado o STN, no qual a estrutura está definida conforme a Figura 4.




**Fonte: (JADERBERG *et al.*, 2015).**

**Figura 4 – Estrutura do STN. Apresenta uma imagem de entrada U, onde é aplicada os módulos Localisation Net e Grid Generator e apresenta uma imagem de saída V**

### 2.2.1 Localisation net

Dado um mapa de características da imagem U definida pela Formula 1 com os parâmetros: W (largura), H (altura) e C (canais) e com a saída $\theta$, os parâmetros da transformação. A função de rede de localização pode assumir qualquer formato, como uma rede totalmente conectada ou uma rede convolucional, mas deve incluir uma camada de regressão final para produzir a transformação dos parâmetros $\theta$ (JADERBERG *et al.*, 2015).

$$U \epsilon R^{HxWxC} \tag{1}$$

### 2.2.2 Grid generator

O *Grid Generator* é uma etapa primordial do STN a qual é responsável por realizar a transformação pixel por pixel da imagem. Esta modificação na qual pode ser uma translação, mudança de escala, rotação ou distorção mais genérica, dependendo do ângulo $\theta$ definido anteriormente (JADERBERG *et al.*, 2015).

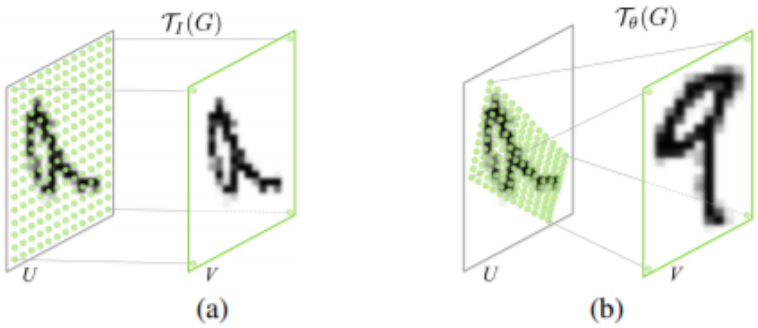
A transformação é definida pela equação 2;

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = [\theta_1 \ \theta_2 \ \theta_3] \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{2}$$

Com $x_i^s$ e $y_i^s$ referem-se a saída normalizada do pixel correspondente a nova imagem. O símbolo $\theta$ remete-se ao angulo definido pelo *Localisation Net* e $x_i^t$ e $y_i^t$ remetem-se à matriz de entrada, a qual também está normalizada (JADERBERG *et al.*, 2015).

### 2.2.3 Sampler

O *Sampler* é a etapa final do STN, a qual é responsável por recriar a imagem agora com os parâmetros das transformações. É possível verificar a conversão da imagem na Figura 5, onde esta foi alterada pelos parâmetros citados anteriormente, permanecendo assim no padrão pré-estabelecido (JADERBERG *et al.*, 2015). O caso em questão, é um exemplo de reconhecimento de números através de imagens, após a etapa do *Sampler* a imagem está de uma maneira tal que algum determinado algoritmo consegue reconhecê-la com uma menor taxa de erro.




**Fonte: (JADERBERG *et al.*, 2015).**

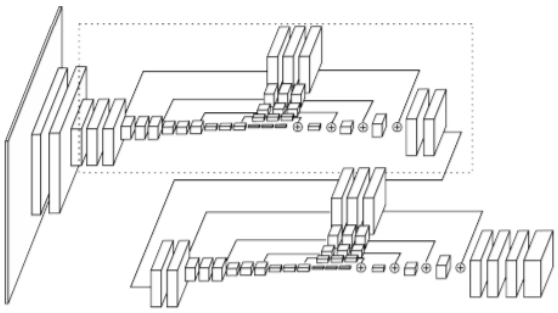
**Figura 5 – Resultado da transformação do STN, onde apresenta como entrada uma imagem distorcida e como resultado a imagem corrigida.**

## 2.3 Módulo stacked hourglass networks for human pose estimation (SPPE)

Conforme a Figura 3 O *Staked Hourglass Networks for Human Pose Estimation* é o estágio do Alphpresen¸ca deaPose responsável por reconhecer a pose. Essa é uma tarefa complexa, pois vários fatores precisam ser abstraídos para que não haja distorção no reconhecimento, tais como roupas, iluminação e até possíveis distorções na pose da pessoa em questão (FANG *et al.*, 2017).

O SPPE é um método baseado em redes totalmente convolucionais com uma arquitetura nomeada de *stacked hourglass*, a Figura 6 representa o formato desta arquitetura. Essa sessão é baseada nos conceitos de Newell, Yang e Deng (2016a), Newell, Yang e Deng (2016b).

As características são processadas em todas as escalas e consolidadas para melhor capturar as várias relações espaciais associadas ao corpo. O processamento é repetido de maneira *bottom-up* (altas resoluções para baixas) e *top-down* (baixas resoluções para altas) em conjunto com a supervisão intermediária na qual é fundamental para melhorar o desempenho da rede.



**Fonte: (NEWELL; YANG; DENG, 2016b).**

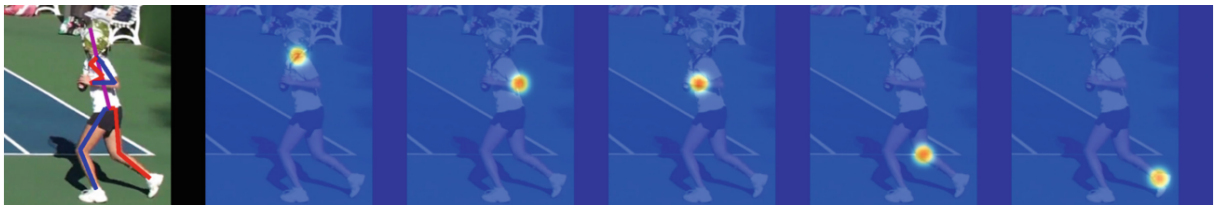
**Figura 6 – Arquitetura do SPPE, representando a repetição *bottom-up* e a *top-down*.**

A Figura 6 mostra a arquitetura geral de duas ampulhetas empilhadas juntas. O *design* da ampulheta é motivado pela necessidade de capturar informações em todas as escalas. Ela se ramifica em cada resolução e combina informações em várias resoluções pela *upsampling* de vizinhos mais próximos seguida pela adição elementar. Após atingir a resolução de saída,

três convoluções 1x1 consecutivas são aplicadas para produzir as previsões finais, que são um conjunto de mapas de calor que indicam probabilidades da presença de cada articulação da pose em cada pixel.

Enquanto é mantida a forma geral da ampulheta, explora-se várias opções na implementação específica de camadas e são os módulos de aprendizado residual. A primeira camada é uma convolução 7x7 padrão com passo 2 e em qualquer outro lugar em que a resolução diminua implica um *maxpool* com uma janela 2x2 e passo 2. Todos os módulos residuais produzem 256 recursos, exceto as camadas logo antes da ampliação de amostra onde existem 512.

Duas ampulhetas são empilhadas ponta a ponta com perda intermediária, fornecendo as repetições *bottom-up* e *top-down* e reavaliação de estimativas iniciais em toda a imagem. A primeira ampulheta prevê um conjunto inicial de mapas de calor, representado na Figura 7 nos quais aplica-se uma perda. Em seguida, a segunda ampulheta processa essas características de alto nível novamente em todas as escalas para capturar ainda mais as relações, o que é essencial para o desempenho final.



**Fonte: (NEWELL; YANG; DENG, 2016a).**

**Figura 7 – Representação dos mapas de calor gerados pela rede do SPPE, os quais utilizados para determinar cada posição da pose. O mapa de calor representa da esquerda para a direita: pescoço, cotovelo esquerdo, pulso esquerdo, joelho direito, tornozelo direito).**

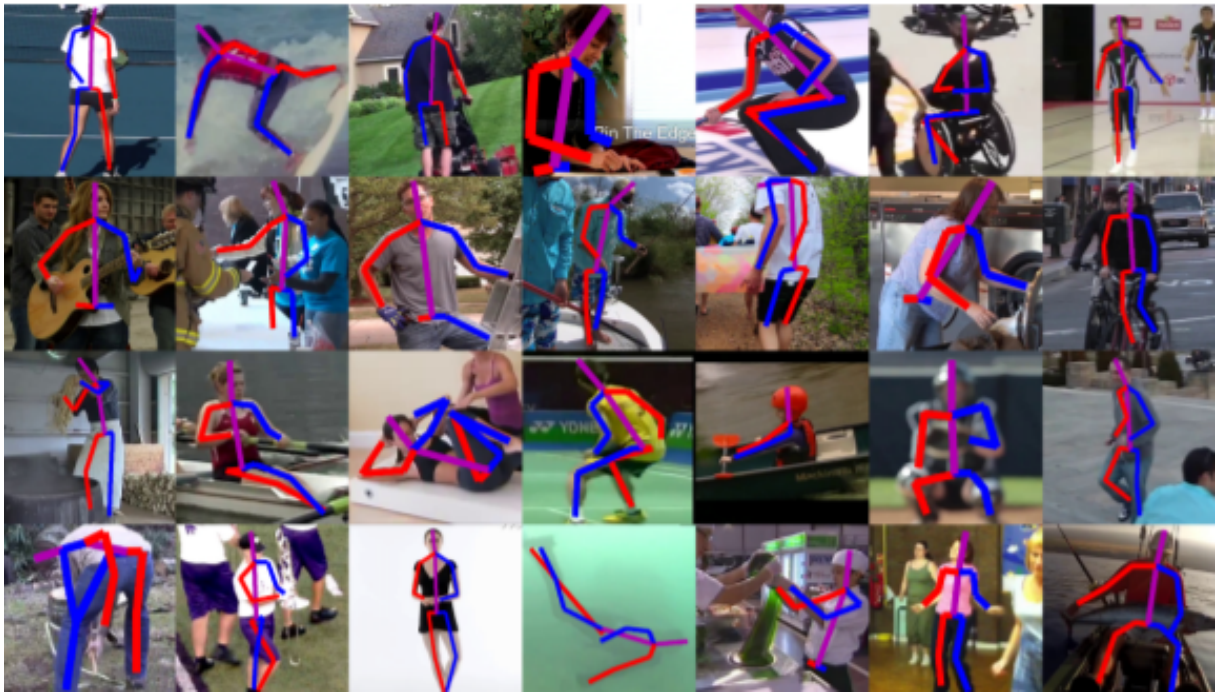
Após esse processo, o resultado da estimativa da pose pode ser visualizado na Figura 8, nela é mostrado alguns exemplos do comportamento do algoritmo para diversas situações.

Dado toda a construção da pose utilizando as técnicas do SPPE, percebe-se que a elaboração desses mapas de calor é facilitada se a posição da pessoa está apropriada em relação a câmera, sendo assim, são necessárias as etapas do AlphaPose de STN e SDNT conforme as seções 2.2 e 2.4 respectivamente.

## 2.4 Módulo spatial de-transformer networks (SDTN)

De acordo com a arquitetura do AlphaPose relatada na Figura 3 cada pessoa encontrada na imagem é submetida a um processo denominado STN o qual está descrito em 2.2 e posteriormente a pose da pessoa é construída pelo SPPE 2.3.

Com essas informações é necessário reverter as modificações realizadas anteriormente mas com a informação da pose, esse processo é denominado de Spatial De-Tranformer Networks, que consiste no inverso do STN.



**Fonte: (NEWELL; YANG; DENG, 2016a).**

**Figura 8 – Representação visual da construção das poses empregando somente o método SPPE.**

## 2.5 Non-maximum suppression

Os detectores, em geral, geram dados redundantes e com detectores de pessoas isso é inevitável. Com o algoritmo utilizado no AlphaPose, com o Yolo, isso ocorre também.

Para contornar esses problemas existe uma técnica denominada Non-maximum suppression. Segundo (HOSANG; BENENSON; SCHIELE, 2017) sua ideia basicamente é unir itens pertencentes a uma mesma classe tal como a Figura 9.

Segundo Fang *et al.* (2017), o AlphaPose seleciona as poses com maiores indicies de assertividade como referências e as poses detectadas próximas a ela são submetidas a um processo de eliminação e essa técnica é repetida até que não se apresente mais pontos redundantes.

O AlphaPose segue de maneira similar, em alguns casos utiliza a acurácia da detecção para fazer esta eliminação e se isso não for possível, utiliza um critério de eliminação baseado na distância.

## 2.6 Resultados do alphapose

O resultado do AlphaPose é composto por um arquivo *json* o qual é divido por cada *frame* ou imagem inserida no algoritmo. Cada uma destas divisões contém a pessoa detectada com as respetivas informações da pose, tais como: nome da imagem, id da categoria, *keypoints* da


**Fonte: (HOSANG; BENENSON; SCHIELE, 2017).**

**Figura 9 – Detecção de pessoas após aplicado o Non-maximum suprresion, onde é possível verificar que várias detecções redundantes foram eliminadas.**

pose e um *score*. Para a elaboração do método da estimativa da posição da bola, será utilizado este arquivo *json*.

O nome da imagem corresponde a identificação de qual *frame* a informação refere-se, e seu conteúdo é: 0.jpg, 1.jpg,..., n.jpg.

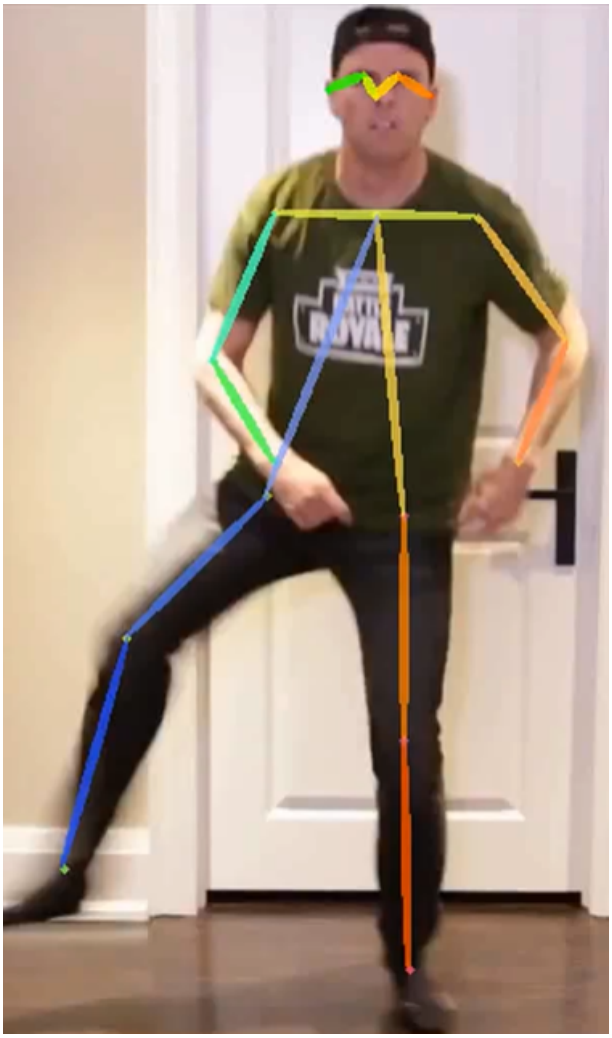
O id da categoria tem como valor padrão 1 (um). Esta variável representa que a demarcação da imagem representa uma pessoa.

A variável *keypoints* contém os locais das partes do corpo e a detecção de confiança apresentada no formato: x1, y1, c1, x2, y2, c2,...cn. Onde o grau de confiança encontra-se no intervalo [0, 1] sendo que quanto maior o valor melhor é a confiança da pose detectada.

A última variável de cada pessoa detectada, é *score*. Ele representa o grau de confiança total da pessoa, calculado através da pose paramétrica NMS.

Um exemplo visual de saída do AlphaPose está presente na Figura 10. O algoritmo consegue reconhecer a posição dos olhos das pessoas, o tórax, duas diferentes articulações para braços e pernas e também uma conexão do tórax com quadril representando o tronco.

A Figura 11 mostra a reconstrução através do arquivo *json* da pessoa na Figura 10. Cada fragmento do arquivo é responsável pelas coordenadas das poses, ao todo são 17 pontos diferentes, os quais apresentam suas determinadas ligações, formando assim uma parcela da pose.



**Fonte: O autor.**

**Figura 10 – Exemplo de saída do AlphaPose, onde as vértices das linhas coloridas representam a construção da pose da pessoa.**

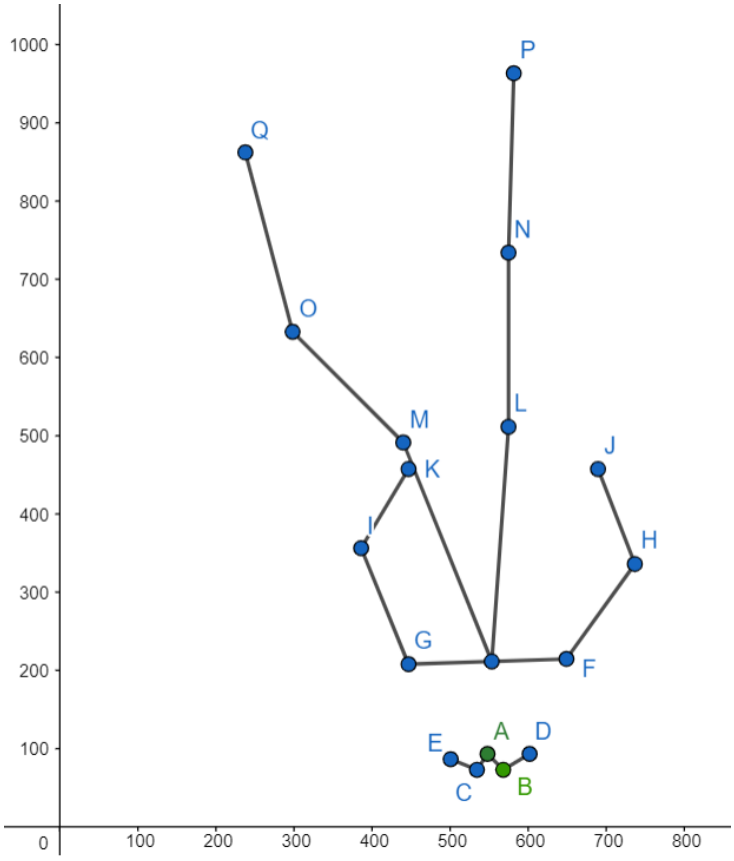
## 2.7 Seleção e avaliação de modelos

Previamente a discussão de técnicas para redução do *overffiting* é necessário definir técnicas a serem usadas para garantir a validação do modelo escolhido.

### 2.7.1 3way holdout

Dentre as técnicas uma delas é o *3Way Holdout*, segundo (RASCHKA, 2015) a divisão da base de dados é realizada em três partes: treino, validação e teste. O conjunto de treinamento é usado para ajustar os diferentes modelos e o desempenho no conjunto de validação sendo usado para a seleção do modelo.

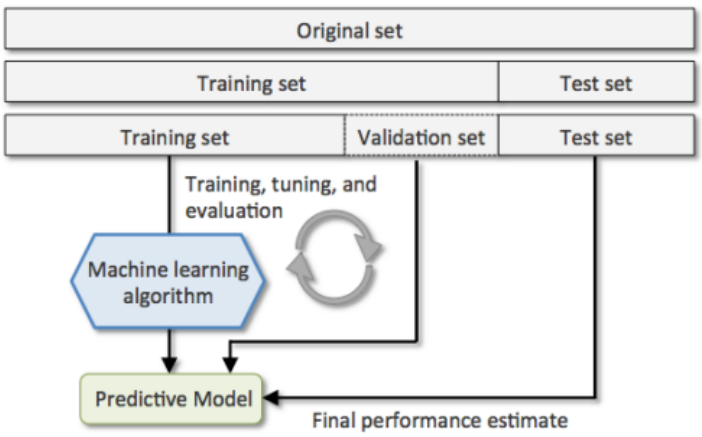
A vantagem de ter um conjunto de teste em que o modelo não tenha sido utilizado durante as etapas de treinamento e seleção do modelo, é que pode-se obter uma estimativa menos tendenciosa de sua capacidade de generalizar para novos dados.




**Fonte: O autor.**

**Figura 11 – Pose reconstituída através do arquivo `json`, onde cada letra retrata um diferente ponto disponível.**

Na Figura 12 está representado como esse processo é realizado. O conjunto de treino e validação são utilizados durante o processo para aperfeiçoamento relatado anteriormente. Enquanto os dados de testes são utilizados somente no momento que o modelo for definido.



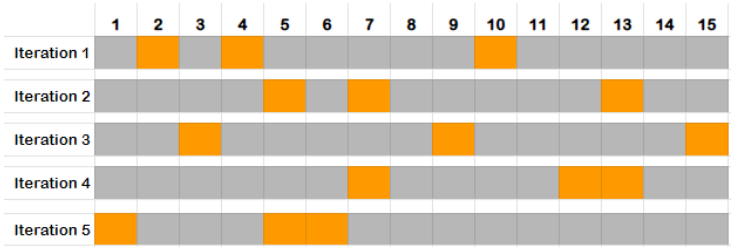

**Fonte: (RASCHKA, 2015).**

**Figura 12 – Arquitetura do método 3way Holdout, onde a base de dados é dividida em três partes, treino-validação e teste, na qual o teste é empregado somente na última etapa da elaboração do algoritmo de *machine learning***

### 2.7.2 Random subsampling

Em alguns casos onde não é exigido um custo computacional elevado, é possível realizar de maneira similar a uma combinação de execuções do *3Way Holdout* tal como definido na seção 2.7.1. Tal técnica denominada *Random Subsampling validation* [1].

A ideia desta validação é dividir o treinamento entre treino e teste de maneira aleatória com um numero pré-definido de interações. A Figura 13 exemplifica como esse processo de divisão atua.




**Fonte: (PEDREGOSA *et al.*, 2011).**

**Figura 13 – Exemplo do *Random Subsambpling*, onde cada linha representa uma execução do método com uma base aleatoriamente definida (Pontos cinzas representando amostras de treino e os amarelos a base de teste).**

A equação 3, representa esse processo da divisão da dados de dados utilizando o *Random Subsampling*. Onde *acc* representa o erro e *k* é o numero de iterações.

$$acc_{cv} = \sum_{i=1}^{k} \frac{acc_i}{k} \tag{3}$$

Entretanto, essa metodologia não garante que todas as amostras são selecionadas ao menos uma vez para treino ou para validação, sendo assim, não é recomendada para bases desbalanceadas.

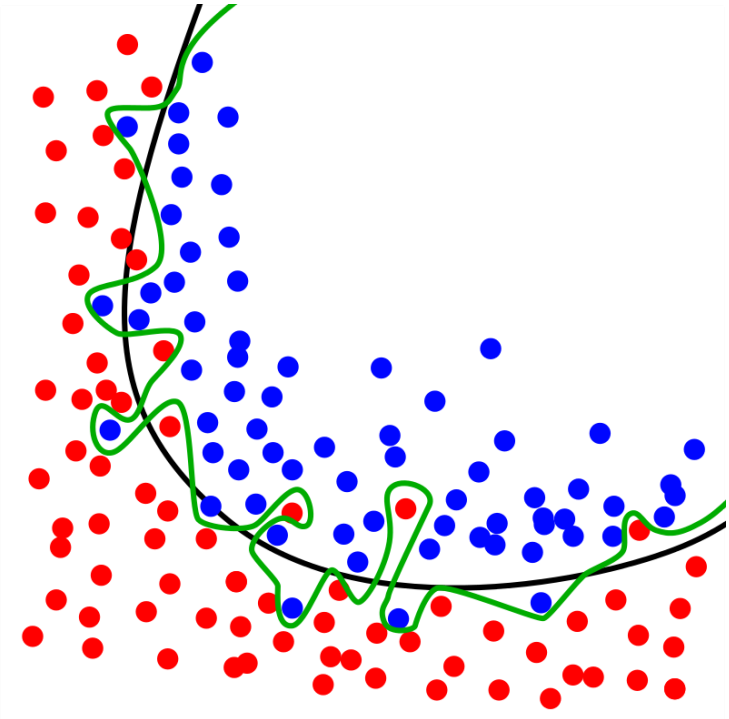
## 2.8 Overfitting

Segundo (LIN, 2020) *overfitting* ocorre quando um modelo tem um bom desempenho na base de treinamento, mas ruim para bases de teste e validação. É um problema recorrente na área de *Machine Learning*.

A Figura 14 representa como isso afeta uma amostra de dados. O modelo com *overffiting* apresenta uma precisão muito grande o que dificultaria a predição de novos dados.

Este problema é ocasionado por alguns fatores tais como um modelo complexo ou uma base de dados desbalanceada. Todavia, não existe uma maneira de realmente saber se o modelo terá um *overfitting* até realmente validá-lo (OVERFITTING. . . , 2020). Caso esse problema

[1] Representada como diferentes nomenclaturas na bibliografia, além de *Random Subsampling* também conhecida como `Monte Carlo Cross-Validation` e *Random permutations cross-validation* (PEDREGOSA *et al.*, 2011)



**Fonte: (CÁRDENAS-MONTES, 2006).**

**Figura 14 – Exemplo de Overfitting representado pela linha verde, onde o modelo apresenta um resultado com muitos detalhes comparada a linha preta a qual representa um modelo balanceado.**

seja detectado, existem algumas técnicas para solucionar ou diminuir esse problema, tais como *feature selection, data cleaning, data augumentation* e métodos com esembling.

### 2.8.1 Seleção de características

Ainda remetendo a problemas relacionados a base de dados, algo que pode influenciar não só no *overfitting* mas também na acurácia do modelo é a seleção dos atributos de entrada.

Em alguns casos a base de dados pode apresentar muitas características o que pode atrapalhar o modelo principal. Para isso é necessário selecionar as principais características.

Existem algumas opções para realizar isso. Dependendo o modelo escolhido ele já apresenta uma opção para verificar quais características apresentam maior importância para o treinamento (BROWNLEE, 2016).

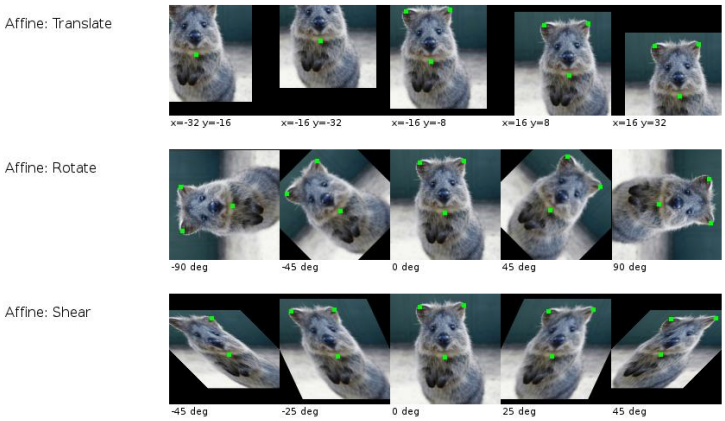
Outra opção é realizar uma analise do resultado e a combinação de vários atributos diferentes, entretanto, esse processo pode exigir um custo computacional muito grande. Uma outra possibilidade, é selecionar os atributos conforme a regra de negócio, dependendo do problema, nem todas as características disponíveis podem ser úteis para a resolução do mesmo.

### 2.8.2 Data augmentation

Uma grande base de dados pode reduzir o *overfitting*, porém, nem sempre é possível incrementar a base de treino e para esses casos existe a técnica de *data augmentation*.

Essa técnica consiste em adicionar novos dados a partir dos dados originais porém adicionando algum ruído à entrada.

Por exemplo, na Figura 15 onde a base dados consiste em imagens, o dado original pode ser recortado, rotacionado ou distorcido que ainda corresponde a mesma saída.




**Fonte: (LIN, 2020).**

**Figura 15 – Exemplo de data augumentation**

O mesmo pode valer para regressões, os dados de treino podem sofrer uma pequena perturbação, mas ainda poderá corresponder ao mesmo valor.

### 2.8.3 Ajuste de parâmetros

Finalizados os problemas relacionados à base de dados, será abordado nesse tópico sobre o modelo de aprendizado.

Em alguns casos o *overffiting*, pode ser ocasionado pelos parâmetros do algoritmo. Para otimizar esse processo de busca de parâmetros existem alguns algoritmos tais como o *GridSearch* e o *RandonSearch* (PEDREGOSA *et al.*, 2011).

A ideia de ambos é basicamente utilizar o *cross-validation* com um número de *folds* pré-definido e testar uma lista de parâmetros. A diferença de ambos os modelos citados é que o *GridSearch* realiza todos os testes com a lista de parâmetros, já o *RandonSearch* utiliza um método aleatório com parte da amostragem.

### 2.8.4 Ensembling

A escolha do algoritmo também deve ser analisada com cautela para reduzir o *overfitting*. Uma técnica utilizada em alguns algoritmos chamada *ensambling* pode apresentar melhores resultados.

*Ensembling* são métodos de aprendizado que combinam vários modelos separados. Existem algumas técnicas de *ensembling*, dentre elas está a de *boosting*.

A técnica de *boosting* consiste em utilizar vários métodos de aprendizado "fracos"e restrito (com uma árvore de decisão limitada, em sequência), cada um utiliza os erros do modelo anterior para os ajustes e assim no final essa junção resulta em um método robusto.

## 2.9 Métricas de avaliação

Existem inúmeras maneiras para avaliar o desempenho de uma regressão. Duas técnicas foram escolhidas, as quais serão descritas nesta seção, o erro médio absoluto (MSE) e o erro médio quadrático (MAE).

### 2.9.1 Erro médio quadrático

Para regressões, uma forma de mensurar os resultados é a média do erro quadrático (MSE) (PEDREGOSA *et al.*, 2011). Esta forma de análise consiste em subtrair o valor real com o valor previsto e elevar à potência de dois, este método potencializa maiores erros. Este processo pode ser definido pela Equação 4.

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y - \hat{y})^2 \tag{4}$$

### 2.9.2 Erro médio absoluto

Outra forma de análise para regressões é a média do erro absoluto (MAE) (PEDREGOSA *et al.*, 2011). Como o próprio nome remete, a técnica em questão realiza uma média com as somas dos erros pegando seus valores absolutos.

Como a média do erro quadrático pode penalizar de maneira mais incisiva grandes erros é importante trazer outra métrica para os resultados.

O MAE é definido pela Equação 5.

$$MAE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y - \hat{y}| \tag{5}$$

# 3 PROPOSTA PARA ESTIMATIVA DA POSIÇÃO DA BOLA

Para estimar a localização da bola durante partidas em esportes coletivos parte da hipótese de que o jogador que está com a posse da bola apresenta uma postura diferenciada dos demais atletas. Isso acontece não apenas no futsal, mas também em vários outros esportes coletivos.

A título de exemplos observa-se que na Figura 16 os dois jogadores que disputam a bola apresentam uma pose distinta dos demais, com os joelhos flexionados e a coluna levemente curvada. Também, na Figura 17, de um jogo de basquete, os jogadores que disputam a bola estão com os braços levantados e uma posição corporal diferente dos demais.

**Fonte: Autor.**

**Figura 16 – Exemplo da construção da pose em jogadores em uma partida futsal**


**Fonte: Autor.**

**Figura 17 – Exemplo de pose em jogo de basquete**

Baseado nessas características, o método proposto constitui-se em tentar estimar a posição da bola através da pose dos jogadores.

O processo consiste em 2 partes, sendo a primeira a obtenção da localização da pose dos jogadores e, com base nessas informações, ir para a segunda etapa onde é feito o treinamento de um método inteligente para realizar uma estimativa da posição da bola.

## 3.1 Materiais

Os dados utilizados nesse trabalho foram adquiridos por meio de gravações dos jogos do Pato Futsal. As imagens foram adquiridas por uma câmera GoPro Hero 4, no ginásio Dorival Lavarda, no município de Pato Branco.

Para a base de dados de treinamento foram utilizados 300 *frames*, escolhidos de maneira aleatória, e em momentos distintos da partida, de forma que fosse captado a bola e os jogadores em várias posições. Para esses registros, a posição da bola foi adicionada manualmente. Para fins de visualização e validação do modelo, 4 pequenos segmentos da partida foram separados, variando de 8 até 32 segundos com uma taxa de 5 *frames/seg*.

## 3.2 Método

O método proposto está representado na Figura 18, a qual tem como entrada os vídeos adquiridos.

Sobre as imagens adquiridas e pré-processadas o algoritmo AlphaPose foi empregado, retornando como resultado a pose de todos os indivíduos presentes na cena, frame a frame. Após esse procedimento é necessário executar uma segmentação, ou seja, selecionar apenas os indivíduos de interesse (i.e. os jogadores) e excluir outros indivíduos que não são objetos da análise (e.g. juiz, público, etc).

Realizada essa etapa uma correção da distorção radial da câmera, foi feito a conversão das posições dos jogadores para metros.

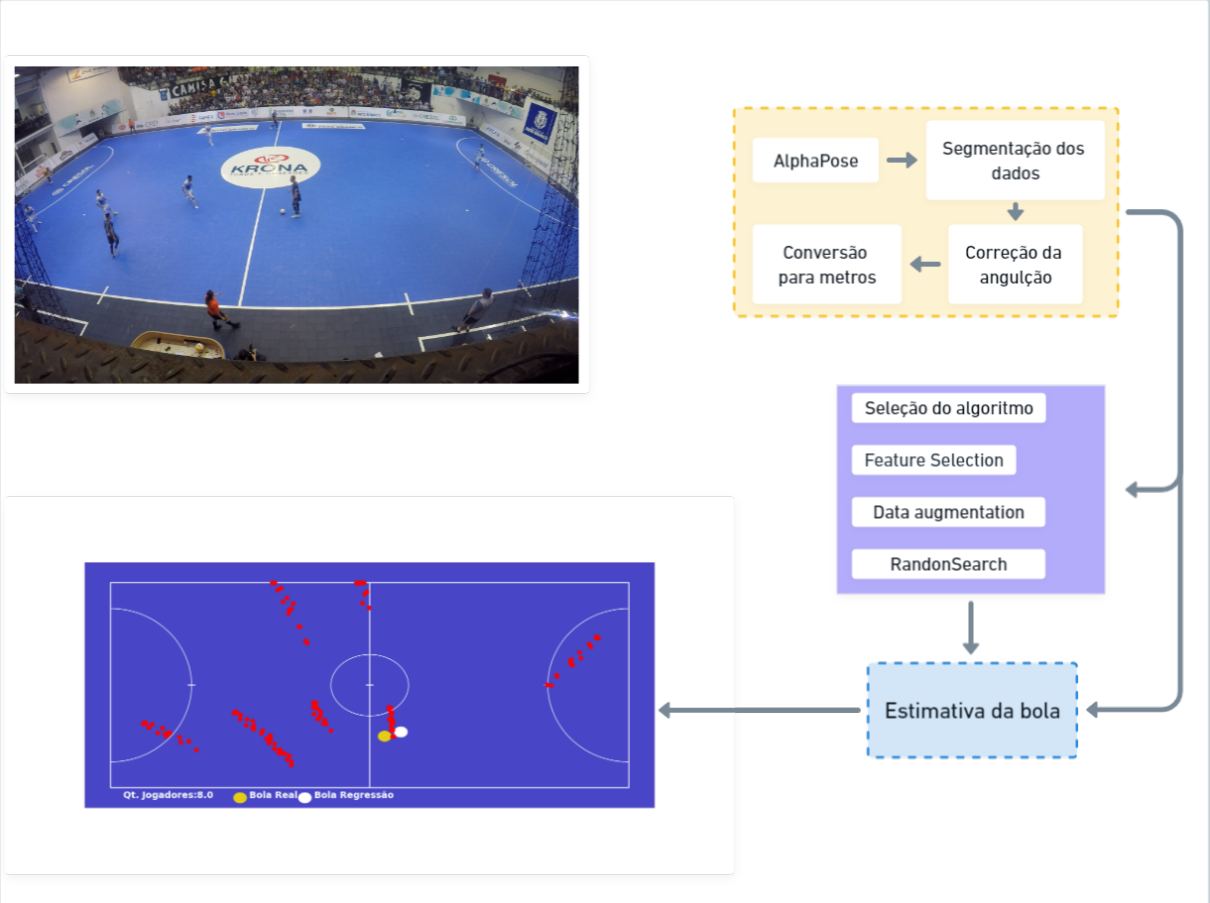
Uma vez obtidos os dados e realizada as devidas etapas de pré-processamento, um método inteligente é treinado para estimar a posição da bola.

A seleção do modelo utilizou o método de *randomSearch* sobre uma validação cruzada do tipo 3-way hold-out.

A avaliação será realizada utilizando as métricas de erro médio absoluto (MAE) e erro médio quadrático (MSE) como citados respectivamente nas seções 2.9.2 e 2.9.1

Referente a etapa de treinamento, algumas técnicas foram empregadas, sendo estas: escolha do algoritmo, *feature selection*, *data augmentation* e seleção de parâmetros.

E, finalizado o processo, os dados dos jogadores junto com a posição da bola estimada são visualizados de uma maneira gráfica.




**Fonte: Autor.**

**Figura 18 – Definição da arquitetura desenvolvida. Onde apresenta como entrada um vídeo de uma partida de futsal, seus vídeos passam pelo processo de aquisição de pose e conversão para metros. Para a estimativa da posição da bola, o treinamento passa por algumas técnicas discutidas em 2.8 e por último, uma representação gráfica do resultado da posição da bola é construída.**

# 4 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Esse capítulo apresenta os resultados conforme as etapas apresentadas na Figura 18.

## 4.1 Aquisição de dados

A primeira etapa do projeto consiste em, a partir da aquisição dos vídeos da partida de futsal, estimar a pose dos indivíduos presentes na cena utilizando o método AlphaPose.

Um exemplo de resultados pode ser observado na Figura 19.


**Fonte: Autor.**

**Figura 19 – Exemplo da aquisição das poses dos jogadores com o AlphaPose na imagens utilizadas no desenvolvimento. Poses de juízes, treinadores e pessoas da arquibancada também foram marcadas.**

Na imagem acima pode ser observado um problema, as poses das pessoas da arquibancada, comissão técnica e juízes são reconhecidas, além daquelas dos jogadores. Para eliminar esses objetos da análise é realizada uma segmentação, limitando o escopo das poses de interesse aos jogadores da quadra.

Para realizar a segmentação, foram selecionados os pontos mínimos e máximos da pose de cada jogador e confrontado com os pontos limites da quadra. Na Figura 20 pode-se visualizar o resultado desta segmentação, onde apenas os jogadores estão com suas poses delimitadas.

Outro ponto interessante a ser avaliado é que nem todos os jogadores tiveram suas poses reconhecidas logo, a base de dados, apresentará valores faltantes. Além disso, não haverá nenhuma diferenciação entre os times.


**Fonte: Autor.**

**Figura 20 – Resultado da segmentação dos jogadores, eliminando a pose das pessoas não envolvidas na partida como na Figura 19.**

## 4.2 Rotulação dos dados

Por se tratar de uma abordagem supervisionada é necessário rotular os dados, ou seja, é necessário marcar manualmente a posição da bola em todos os *frames* em análise.

Diferentemente da posição dos jogadores, não existe método automático totalmente efetivo para detecção por imagens da bola. A bola, além de ser um objeto muito pequeno, normalmente encontra-se oclusa pelo corpo dos jogados, tem uma velocidade bastante grande e confunde-se com os objetos da cena.

O objetivo de se estimar a posição da bola, via pose dos jogadores vem, justamente, dessas dificuldades. Nesse sentido optou-se por uma rotulagem manual a fim de não gerar erros nos dados de entrada.

## 4.3 Conversão dos dados

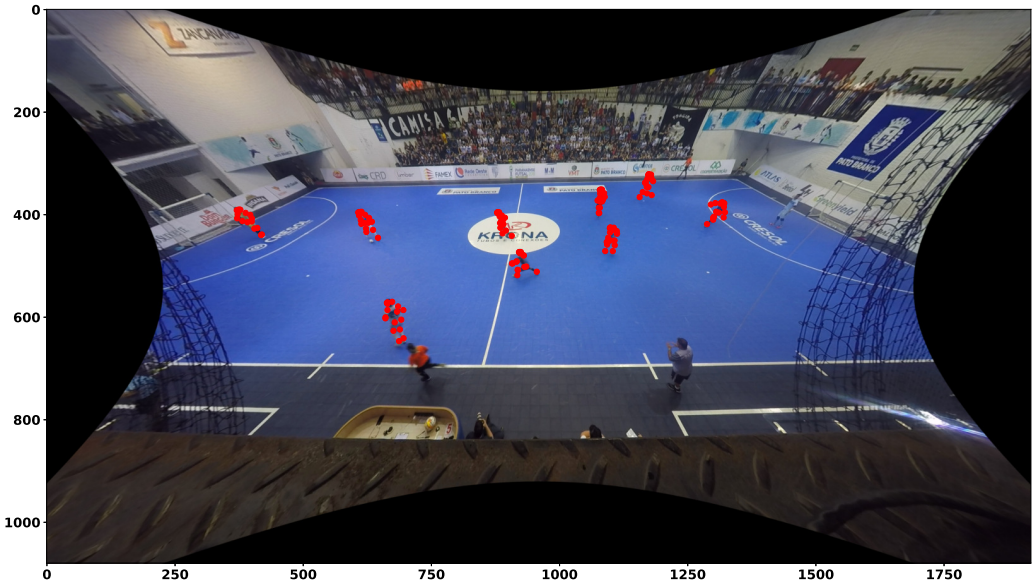
Os pontos da pose dos jogadores foram reconhecidos e fazem referência dos mesmos na imagem. Entretanto, existe distorções de perspectiva na imagem adquirida que precisam ser corrigidas.

Nesse sentido, ponto referente a pose são transformados para metros em relação a quadra, ou seja, a referência da posição das poses, leva em consideração somente as medidas da quadra e não mais da imagem.

Sendo assim o modelo para a detecção da posição da bola continuará atuando mesmo com alterações no local e na câmera.

Essa transformação é realizada em duas etapas (1) correção da distorção da lente angular da Go Pro HERO4 e (2) conversão para metros.

O resultado da correção da distorção radial é apresentado na Figura 21.




**Fonte: Autor.**

**Figura 21 – Exemplo da correção da distorção radial da lente, utilizando como base as Figuras 19 e 20. Onde cada ponto em vermelho representa as poses dos jogadores.**

Com a correção de perspectiva efetuada, a conversão para metros é realizada de acordo com (PAULICHEN, 2019). A Figura 22 apresenta os pontos das poses dos jogadores convertidos em metros da Figura 21, onde os pontos representam os jogadores em quadra, na imagem percebe-se que existem 9 conjuntos distintos, onde cada um deles é a representação de uma pessoa diferente e cada ponto constitui um fragmento da pose.

Também percebe-se que os pontos estão de acordo com a posição na quadra, os primeiros itens a esquerda representam um dos goleiros.

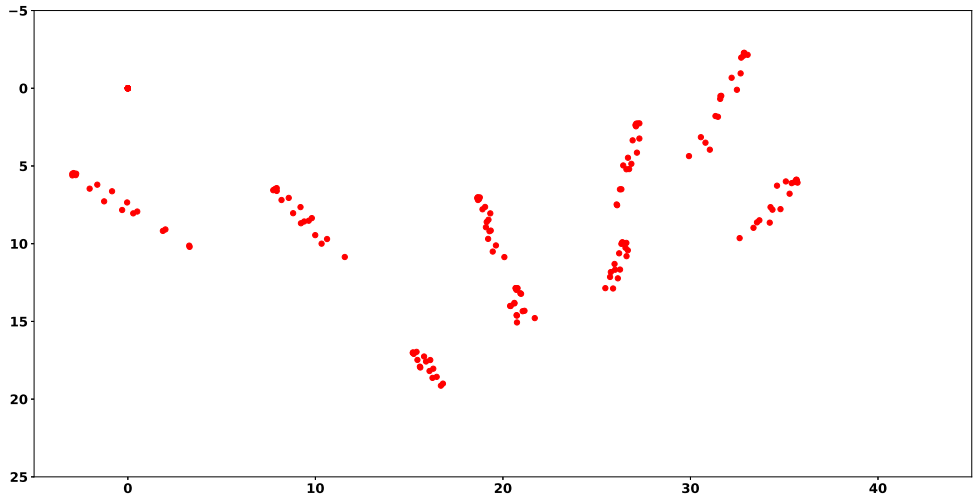
Existe uma diferença no tamanho dos jogadores nessa conversão, mas isso se dá pela distância de cada pessoa da câmera.

## 4.4 Escolha do algoritmo

Finalizada a etapa 1, referente a identificação e correção dos pontos de pose dos jogadores, a etapa 2 consiste no treinamento de um modelo inteligente para realizar uma estimativa da posição da bola.

Sendo um problema de regressão, é necessário realizar a seleção do melhor modelo. A principio foram escolhidos três métodos: *XGBoox, MPLRegressor e a Gausian Processes* que serão avaliados utilizando as métricas de erro absoluto 2.9.2 e quadrático 2.9.1.

Aplicando-se uma validação cruzada do tipo *Random SubSampling*, os resultados obtidos podem ser visualizados na Tabela 2 e na Figura 23, um comparativo com a distância eu-




**Fonte: Autor.**

**Figura 22 – Exemplo da transformação das poses para metros, onde cada ponto em vermelho, representa os jogadores em quadra. Mesmo *frame* em comparação com a Figura 21, percebe que os jogadores se mantem na mesma disposição.**

clidiana entre a posição real da bola e a posição estimada, onde no eixo horizontal representa esse resultado e no eixo vertical representa a quantidade de pontos naquele intervalo.

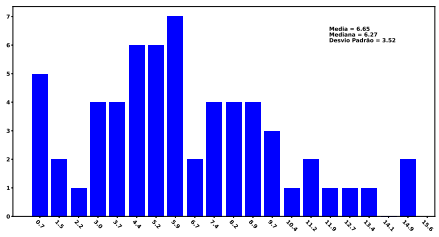
| | Treino | | Teste | | |
|---|---|---|---|---|---|
| Algoritmo | MAE | MSE | MAE | MSE | Qtd. *features* |
| XGBoost | 0,01 | 0,11 | 4,18 $\pm$ 0,18 | 5,47 $\pm$ 0,50 | 510 |
| MPLRegressor | 2,28 $\pm$ 0,52 | 3,09 $\pm$ 0,63 | 4,35 $\pm$ 0,35 | 5,43 $\pm$ 0,49 | 510 |
| Gaussian Process | 1,52 $\pm$ 0,07 | 1,95 $\pm$ 0,12 | 3,98 $\pm$ 0,28 | 4,23 $\pm$ 0,34 | 510 |

**Tabela 2 – Comparação entre algoritmos de regressão, utilizando como métricas de avaliação o erro médio quadrático (MSE) e o erro médio absoluto (MAE). Todos os valores representados em metros.**

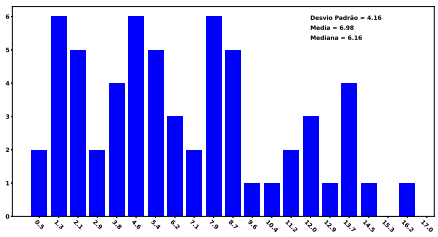
Os resultados são similares, com ligeira vantagem para o método *GaussianProcess*. Nesse resultado um erro absoluto de 4,18 significa que a bola foi estimada, em média, 4,18 metros distante de sua posição real.

Trata-se de um erro alto, que dificilmente poderia ser empregado em uma análise estatística real com precisão. Por outro lado é um erro bastante interessante, se considerar que a única entrada é a posição dos jogadores. A ideia é que essa estimação seja utilizada em conjunto com outros estimadores da posição da bola, auxiliando principalmente em casos onde o objeto bola esteja oculto ou confundido com elementos do jogo (e.g. bola oculta atrás de um jogador).

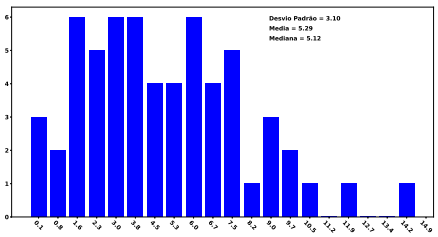
Todavia, independentemente do método testado, foi observado uma alta tendência de overffiting. A hipótese que se apresenta é que, diminuindo overffiting, talvez se consiga um erro de estimativa menor.




**(a) XGBoost. Média = 6,65m, Mediana = 6,27m. Desvio padrão = 3,52m**




**(b) MPLRepressor. Média = 6,98. Mediana = 6,16. Desvio Padrão = 4,16**




**(c) GuassianProcess. Média = 5,29. Mediana = 5,12. Desvio Padrão = 3,10**

**Figura 23 – Representação em metros da distância euclidiana para cada um dos algoritmos escolhidos.**

Várias estratégias podem ser empregadas na tentativa de diminuir overfitting. Dentre elas estão: (1) seleção de características (2) *data augumentation* e (3) ajuste de hiperparâmetros. Nos próximos experimentos essas estratégias são implementadas e seus resultados avaliados. Para tal, selecionou-se o método de *XGBoost* em virtude de 2 motivos: (1) foi o método que mostrou melhor resultado na base de treino e (2) é um método que já implementa a combinação de classificadores, fator que pode auxiliar na diminuição do overfitting, conforme descrito na seção 2.8.4.

## 4.5 Seleção de características

Segundo (KOTSIANTIS, 2011) a seleção de características consiste em selecionar um subconjunto das características originais que são mais representativas do problema em análise.

Conforme a seção 2.6, a pose de cada jogador é constituída de 17 coordenadas e em cada uma dessas contendo um índice de assertividade denominados de *score*. Todos esses pontos acabam gerando um base de dados com muitas características o que pode atrapalhar o treinamento.

Para tal 3 sub-conjuntos foram criados para compor a base de dados. São eles:

- Olhos: conjunto composto por 5 pontos, próximos uns dos outros na região dos olhos. Acredita-se que esses pontos não contribuam para a estimação visto que a resolução é baixa para uma boa precisão dessa detecção.

- Scores: composto por 17 pontos, representando índice de assertividade de cada coordenada da pose. Acredita-se que não seja uma informação tão relevante no contexto de estimação da posição da bola.
- Cabeça, tronco e braços: composto por 8 pontos.
- Pés e joelhos: composto por 4 pontos. Acredita-se que esses pontos possam ser os mais discriminativos em uma partida de futsal, visto que são os membros alvo do esporte.

Dados os sub-conjuntos acima, 4 bases de dados distintas foram criadas, são elas:

- DB1: base de dados composta por todos os pontos, 51 atributos para cada jogador, 510 características.
- DB2: base de dados composta por todos os pontos, exceto os pontos dos olhos, 39 atributos para cada jogador, totalizando 390 características.
- DB3: base de dados composta por todos os pontos, exceto os score. 34 atributos para cada jogador totalizando 340 características
- DB4: base de dados composta por todos os pontos, exceto olhos, score e cabeça, tronco e braços. Em outras palavras essa base possui apenas os pontos de pés e joelhos, 8 atributos para cada jogador, totalizando 80 características.

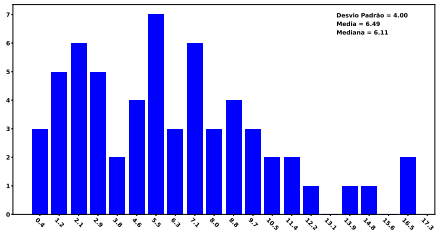
Levando em consideração essa hipóteses, pode-se observar o resultado para cada uma das bases de dados na Tabela 3 e a distância euclidiana na Figura 24.

| | Treino | | Teste | | |
|---|---|---|---|---|---|
| Base | MAE | MSE | MAE | MSE | Qtd. *features* |
| DB1 | 0,01 | 0,11 | 4,18 $\pm$ 0,18 | 5,47 $\pm$ 0,50 | 510 |
| DB2 | 0,01 | 0,01 | 4,03 $\pm$ 0,34 | 5,47 $\pm$ 0,51 | 390 |
| DB3 | 0,01 $\pm$ 0,01 | 0,11 $\pm$ 0,05 | 3,51 $\pm$ 0,41 | 4,89 $\pm$ 0,58 | 340 |
| DB4 | 0,01 | 0.01 | 3,85 $\pm$ 0,30 | 5,27 $\pm$ 0,49 | 80 |

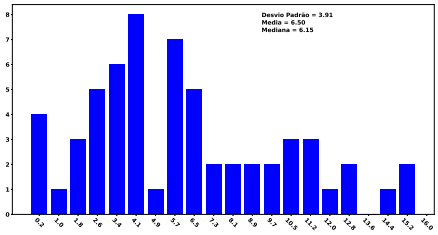
**Tabela 3 – Resultado de diferentes sub-conjuntos de características escolhidas como entradas.**

Pode-se observar que, somente realizando a seleção de características, não foi possível eliminar o *overffiting*, percebendo também que as segmentações DB1, DB2 e DB4 não chegaram a apresentar um valor de desvio padrão significativo. Entretanto, foi possível analisar que o score estava atrapalhando a análise, logo quando retirado obteve-se resultados com uma melhora significativa.

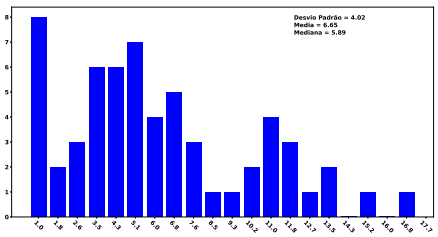
As segmentações DB4 e DB3 obtiveram os melhores resultados comparados a DB1 e DB2. Em ambos os casos houve uma redução na quantidade de características, logo uma simplificação no modelo.




**(a) Retirando pontos dos olhos. Média = 6,49m, Mediana = 6,11m. Desvio padrão = 4,00m**




**(b) Retirando os scores. Média = 6,50. Mediana = 6,15. Desvio Padrão = 3,91**




**(c) Somente pontos dos pés e joelhos. Média = 6,65. Mediana = 5,89. Desvio Padrão = 4.02**

**Figura 24 – Representação em metros da distância euclidiana para cada uma das escolhas para a base de dados.**

Todavia a DB4 apresentou o melhor resultado, e começa-se a concluir que toda a desenvoltura do jogador durante a partida, pode contribuir para a posição da bola e, por esse motivo, as próximas análises levaram em conta esse conjunto de pontos.

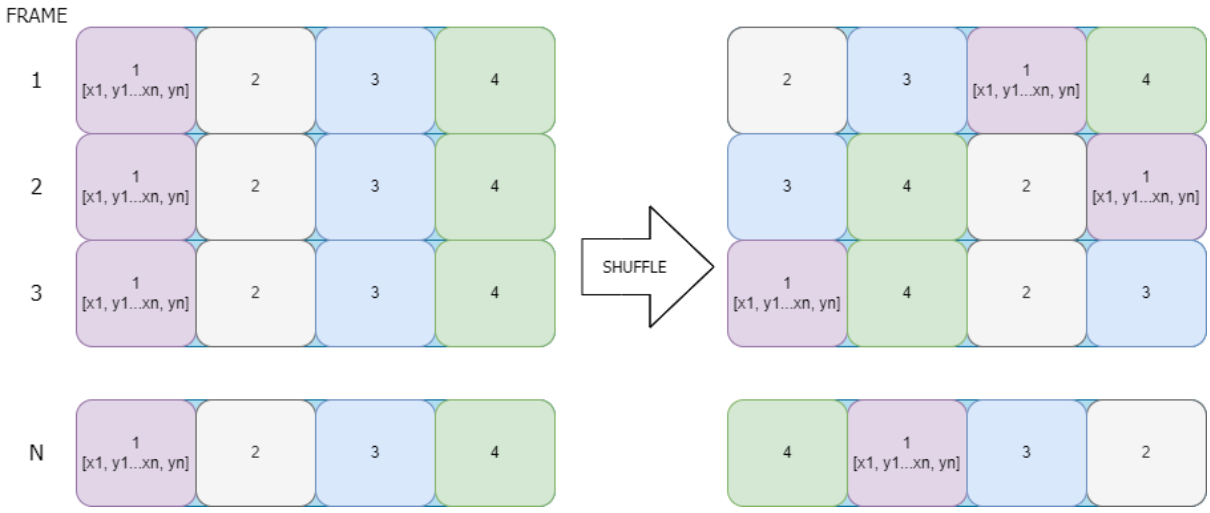
## 4.6 Data augmentation

Outro fator que pode ocasionar o *overffiting* é o tamanho da base de dados, esta que conta com apenas 298 itens que ainda são separados em treino em teste, para contornar esse problema é necessário utilizar das técnicas de *data augmentation* descrito na secção 2.8.2.

No caso do projeto, não foi realizado uma *data augmentation* padrão. Devido o resultado do AlphaPose, não é possível afirmar que cada jogador se manterá na mesma posição. Em um *frame* um dos goleiros pode ser o primeiro jogador a ser reconhecido, já no próximo *frame* ele pode ser o último jogador.

Então, por esse motivo, não é possível simplesmente adicionar novos pontos, é necessário fazer um embaralhamento dos jogadores na base de dados, processo o qual foi denominado de *shuffle*. A Figura 25 representa como esse processo de embaralhamento funciona, no final do processo, cada ponto da pose representa o mesmo valor, por exemplo, onde estava os pontos dos pés, continuará mantendo essas características, porém de um outro jogador.

Isso é necessário para que o algoritmo de regressão, não decore as posições dos jogadores com o *data augumentation*.




**Fonte: O autor.**

**Figura 25 – Exemplo do método Shuffle, Na esquerda representa a base de dados com os jogadores em sequência. Já no lado direito, mostra que os jogadores ficaram embaralhados, porém, sempre mantendo a sequência dos pontos de suas poses.**

Junto com o processo de *shuffle* novos dados vão sendo adicionados na base de dados e cada um deles com um leve ruído para que o método de regressão não tenha mais problemas com *overffiting*, assim finalizando o processo de *data augmentation*.

Para analisar a efetividade deste processo, foram realizados 5 testes com diversas quantidades de novos dados: 1.000, 10.000, 25.000, 35.000 e 50.000. Os resultados podem ser verificados na Tabela 4 e a distância euclidiana na Figura 26.

| | Treino | | Teste | | |
|---|---|---|---|---|---|
| Qtd. de novos dados | MAE | MSE | MAE | MSE | Qtd. *features* |
| 1.000 | 0,08 ± 0,01 | 0,12 ± 0,01 | 3,65 ± 0,35 | 4,78 ± 0,51 | 340 |
| 10.000 | 0,90 ± 0,04 | 1,21 ± 0,06 | 3,37 ± 0,21 | 4,38 ± 0,32 | 340 |
| 25.000 | 1,23 ± 0,03 | 1,64 ± 0,05 | 3,14 ± 0,17 | 4,17 ± 0,37 | 340 |
| 35.000 | 1,31 ± 0,03 | 1,75 ± 0,04 | 3,07 ± 0,15 | 4,12 ± 0,35 | 340 |
| 50.000 | 1,40 ± 0,03 | 1,86 ± 0,04 | 3,08 ± 0,28 | 4,14 ± 0,39 | 340 |

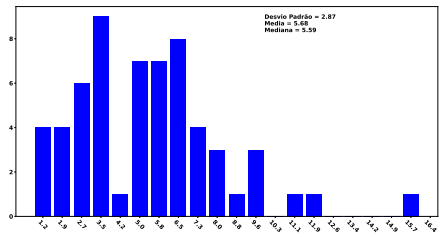
**Tabela 4 – Resultados utilizando diferentes quantidades de novos dados para a base com a técnica de *data augmentation*.**

O processo de *data augumentation* é realizado apenas na base de dados do treino para que não ocorra uma falsa impressão de melhora nos dados de teste.

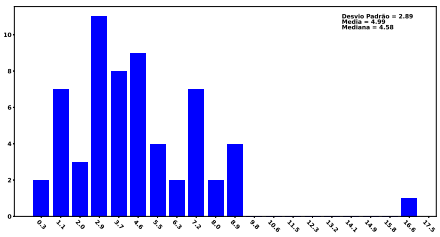
Porém é perceptível que existe uma melhora tanta na acurácia quanto na redução do *overffiting* conforme novos dados são adicionados. O erro médio absoluto quando acrescentado 1.000 novos dados é de 3,65 metros e quando são adicionados 35.000 dados o erro é de 3,07.

Mas essa melhora é limitada a uma determinada quantidade de novos dados. Quando adicionado 50.000 novos dados o erro começa a subir novamente.

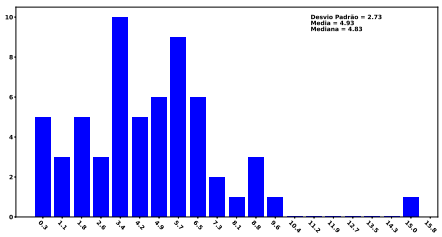
Devido os resultados apresentados na Tabela 4 e na Figura 26 que indicam que será utilizado uma base dados com a inserção de 35.000 novos dados para os próximos testes.



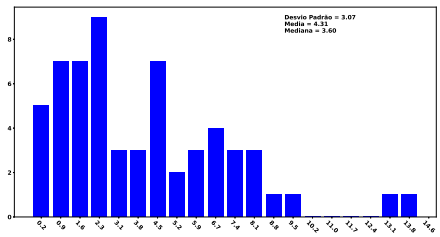

**(a) Adicionando 1.000 dados. Média = 5,68m, Mediana = 5,59m. Desvio padrão = 2,87m**



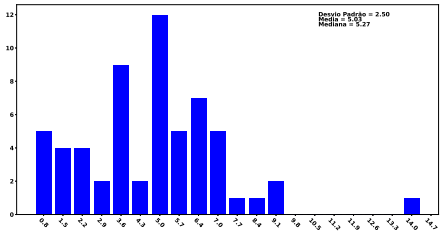

**(b) Adicionando 10.000 dados. Média = 4,99. Mediana = 4,58. Desvio Padrão = 2,89**




**(c) Adicionando 25.000 dados. Média = 4,93. Mediana = 4,83. Desvio Padrão = 2,73**




**(d) Adicionando 35.000 dados. Média = 4,31. Mediana = 3,60. Desvio Padrão = 3,07**




**(e) Adicionando 50.000 dados. Média = 5,03. Mediana = 5,27. Desvio Padrão = 2,50**

**Figura 26 – Representação em metros da distância euclidiana para a inclusão de novos dados utilizando a técnica de *data augmentation*.**

## 4.7 Escolha de parâmetros

A próxima etapa para a melhoria da acurácia é manipular os parâmetros do método de regressão. Até o momento todos os testes foram realizados com os parâmetros padrões.

Para a escolha dos parâmetros foi utilizado o método *RandomSearch* (PEDREGOSA *et al.*, 2011) que consiste em adicionar uma lista de parâmetros a serem testados e uma quantidade de iterações que será realizada onde em cada uma, os parâmetros são escolhidos de forma aleatória.

A base de dados foi previamente separada em treino e teste. O *RandomSearch* utiliza-se do processo de validação cruzada, ou seja, os dados posteriormente separados para treino, são novamente divididos para a avaliação do modelo. Já os dados de teste são utilizados somente no final deste processo, sendo assim a validação de todo esse processo é definida como *Holdout*.

Dada uma lista de atributos e definindo 100 iterações o *RandomSearch* obtém os resultados conforme a Tabela 5.

| Métrica de avaliação | Treino | Teste |
|---|---|---|
| MAE | 1,23 | 2,68 |
| MSE | 1,68 | 3,78 |

**Tabela 5 – Resultados em metros para o resultado do algoritmo *Randon Search*, resultados em metros e utilizando como validação o *3Way Holdout* 2.7.1 .**

E com isso pode verificar que houve a seguinte alteração no valores padrões do XGBoost na Tabela 6

| Parâmetros | Valor Padrão | Resultado do *Random Search* |
|---|---|---|
| *subsample* | 1 | 1 |
| *reg_lambda* | 1 | 0,7 |
| *reg_alpha* | 0 | 0 |
| *n_estimators* | 100 | 500 |
| *max_depth* | 3 | 5 |
| *learning_rate* | 0,1 | 0,1 |
| *colsample_bytree* | 1 | 0,7 |

**Tabela 6 – Resultado dos novos parâmetros do modelo *XGBoost* conforme execuções da Tabela 5**

Não foram todos os parâmetros que tiveram alguma alteração, porém percebe-se a alteração em alguns casos tais como *reg_lambda*, *n_estimators*, *max_depth* e *colsample_bytree*.

As mudanças foram esperadas especialmente pelo *max_depth* que diminuiu do valor padrão. Conforme o aumento dessa característica o modelo fica propenso a ter um overffiting (CHEN; GUESTRIN, 2016).

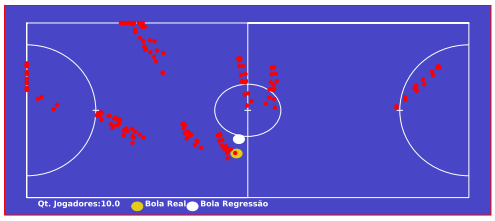
Os resultados foram satisfatórios também, pois já houve uma melhora no desempenho, apresentando um erro absoluto de 2,6864 metros e também reduzindo overffiting.

## 4.8 Resultados finais

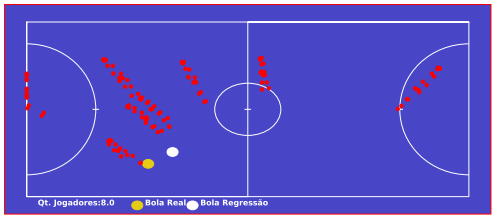
Utilizando algumas pequenas segmentações de vídeos da partida uma representação visual da predição da bola pode ser observada na Figura 27. Nessa figura os pontos estão inseridos em uma representação de uma quadra de futsal, os jogadores estão definidos pelos pontos vermelhos, a posição real da bola é representada pelo circulo amarelo e a posição estimada pelo circulo branco.

Observa-se por esses resultados que a estimativa foi bastante precisa, considerando-se que apenas a posição dos jogadores é utilizada como informação de entrada.

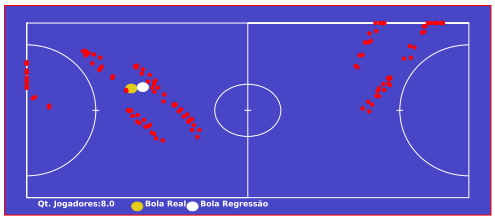
Em algumas casos, porém, o erro apresenta-se relativamente grande, podendo-se observar isso na Figura 28. Analisando os resultados percebe-se que na maioria destes casos uma quantidade significativa de jogadores não foi detectado pelo algoritmo de pose, o que poderia ser uma possível explicação pelo aumento do erro médio de estimação.



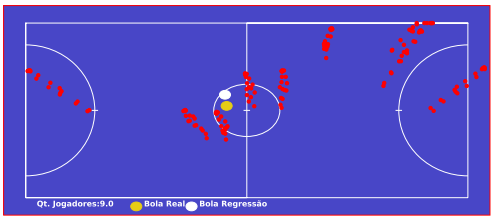

(a) Exemplo 1



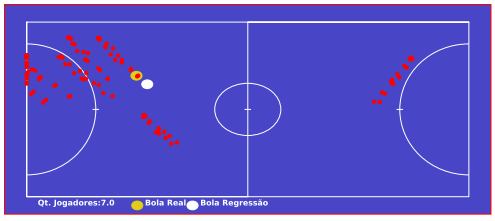

(b) Exemplo 2




(c) Exemplo 3

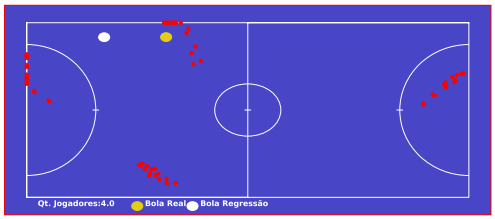



(d) Exemplo 4

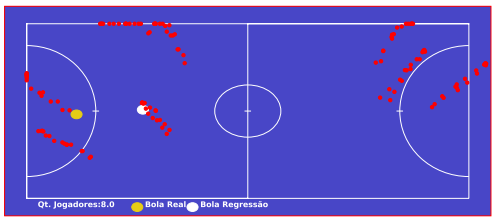



(e) Exemplo 5

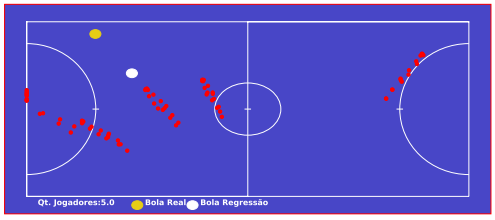
**Figura 27 – Resultados finais, cada imagem representa um momento da partida, os pontos em vermelho são as representações dos jogadores, o ponto amarelo representa a bola original e o ponto branco o resultado da regressão.**




**(a) Exemplo 1**




**(b) Exemplo 2**




**(c) Exemplo 3**

**Figura 28 – Exemplos nos quais a predição da bola não obteve resultados satisfatórios. Percebe-se que a bola predita (branca) ficou longe da bola real (amarela). Também vale ressaltar que nestes exemplos, o AlphaPose não consegue realizar a construção de todos os jogadores.**

# 5 CONCLUSÃO

Os resultados apresentados demonstram que existe uma relação entre a pose dos jogadores e a posição da bola em quadra, e que com esse tipo de dados, é possível construir um modelo que estima a posição da mesma.

O maior problema observado foi a não detecção de todos os 10 jogadores pelo AlphaPose, que impacta diretamente à precisão da estimativa.

Observou-se pelos experimentos que muitos pontos da pose (e.g. olhos) são irrelevantes ou não tem impacto significativo na precisão da estimativa, sendo assim é possível utilizar apenas o subconjunto mais significativo dos mesmos, a saber, pés e joelhos.

No resultado final é possível observar visualmente a posição real e estimada da bola. Tal metodologia, embora ainda apresente erro elevado, pode ser agregada a outros métodos de estimação, melhorando assim a estimativa final.

Finalmente demonstrou-se que uma única câmera é suficiente para se realizar tal estimativa. Obviamente que correções de ângulos e perspectivas se fazem necessárias, mas o resultado final é bastante satisfatório. O custo de operação e manutenção de um aparato de aquisição único é drasticamente reduzido, se comparado a alguns sistemas multi-câmeras que necessitam de sincronizações e outros procedimentos extras. Essa facilidade pode habilitar o uso desse tipo de tecnologia em times de menor expressão e com pouco poder financeiro, nivelando assim as competições esportivas.

## 5.1 Trabalhos futuros

Conseguindo a posição da bola, as vantagens para uma partida são ilimitadas, sendo possível elaborar estatísticas e mapas de calor de forma automática. Tais estatísticas podem ser posse de bola, precisão de passe e chute, distância percorrida, dentre outros.

Como definido a relação entre a pose e a posição da bola para o futsal é possível pensar em levar esta ideia para outros esportes, tais como basquete, handebol e vôlei.

# REFERÊNCIAS


BORG, J. **Detecting and tracking players in football using stereo vision**. [*S.l.*]: Institutionen för systemteknik, 2007.

BROWNLEE, J. **Feature Importance and Feature Selection With XGBoost in Python**. 2016. Disponível em: https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/.

CAO, Z. *et al.* Openpose: realtime multi-person 2d pose estimation using part affinity fields. **arXiv preprint arXiv:1812.08008**, 2018.

CÁRDENAS-MONTES, M. Sobreajuste–overfitting. **Ciemat (Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas)**, 2006.

CHEN, T.; GUESTRIN, C. XGBoost: A scalable tree boosting system. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2016. (KDD '16), p. 785–794. ISBN 978-1-4503-4232-2. Disponível em: http://doi.acm.org/10.1145/2939672.2939785.

CHEN, Z. *et al.* Augmenting sports videos with viscommentator. **IEEE Transactions on Visualization and Computer Graphics**, IEEE, v. 28, n. 1, p. 824–834, 2021.

FANG, H.-S. *et al.* Rmpe: Regional multi-person pose estimation. In: **Proceedings of the IEEE International Conference on Computer Vision**. [*S.l.*: *s.n.*], 2017. p. 2334–2343.

HOSANG, J.; BENENSON, R.; SCHIELE, B. Learning non-maximum suppression. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.*: *s.n.*], 2017. p. 4507–4515.

JADERBERG, M. *et al.* Spatial transformer networks. In: **Advances in neural information processing systems**. [*S.l.*: *s.n.*], 2015. p. 2017–2025.

KOTSIANTIS, S. Feature selection for machine learning classification problems: a recent overview. **Artificial Intelligence Review**, v. 42, n. 1, p. 157–176, 2011.

LIN, D. C.-E. **8 Simple Techniques to Prevent Overfitting**. 2020. Disponível em: https://towardsdatascience.com/8-simple-techniques-to-prevent-overfitting-4d443da2ef7d.

LINK, D. Sports analytics. **German Journal of Exercise and Sport Research**, Springer, v. 48, n. 1, p. 13–25, 2018.

MAKSAI, A.; WANG, X.; FUA, P. What players do with the ball: A physically constrained interaction modeling. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.*: *s.n.*], 2016. p. 972–981.

MORGULEV, E.; AZAR, O. H.; LIDOR, R. Sports analytics and the big-data era. **International Journal of Data Science and Analytics**, Springer, v. 5, n. 4, p. 213–222, 2018.

NEWELL, A.; YANG, K.; DENG, J. Stacked hourglass networks for human pose estimation. In: SPRINGER. **European conference on computer vision**. [*S.l.*], 2016. p. 483–499.

NEWELL, A.; YANG, K.; DENG, J. Stacked hourglass networks for human pose estimation. **arXiv preprint arXiv:1603.06937**, 2016.

OVERFITTING in Machine Learning: What It Is and How to Prevent It. 2020. Disponível em: https://elitedatascience.com/overfitting-in-machine-learning.

PAULICHEN, H. M. **Ferramentas para análise de partidas de futsal utilizando processamento de imagens e visão computacional**. 2019. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2019.

PEDREGOSA, F. *et al.* Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

RASCHKA, S. **Python machine learning**. [*S.l.*]: Packt publishing ltd, 2015.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. **arXiv preprint arXiv:1804.02767**, 2018.

THOMAS, G. *et al.* Computer vision for sports: Current applications and research topics. **Computer Vision and Image Understanding**, Elsevier, v. 159, p. 3–18, 2017.

VEIT, A. *et al.* Coco-text: Dataset and benchmark for text detection and recognition in natural images. **arXiv preprint arXiv:1601.07140**, 2016.

WANG, X. *et al.* Take your eyes off the ball: Improving ball-tracking by focusing on team play. **Computer Vision and Image Understanding**, Elsevier, v. 119, p. 102–115, 2014.