

Lucas Mateus Fernandes

**Ferramenta de visão computacional para a  
análise do movimento de barra fixa voltada para  
testes de aptidão física**

Formiga - MG

2023

Lucas Mateus Fernandes

**Ferramenta de visão computacional para a análise do  
movimento de barra fixa voltada para testes de aptidão  
física**

Monografia do trabalho de conclusão de curso apresentado ao Instituto Federal Minas Gerais - Campus Formiga, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais

Campus Formiga

Ciência da Computação

Orientador: Prof. Me. Fernando Paim Lima

Formiga - MG

2023

Lucas Mateus Fernandes

Ferramenta de visão computacional para a análise do movimento de barra fixa voltada para testes de aptidão física/ Lucas Mateus Fernandes.  
– Formiga - MG, 2023.

92 p. : il.

Orientador: Prof. Me. Fernando Paim Lima

Monografia para trabalho de conclusão de curso (graduação) – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Ciência da Computação, Formiga, 2023.

1. Visão computacional. 2. Barra fixa. 3. TAF. I. Prof. Me. Fernando Paim Lima. II. Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais. III. Ciência da Computação. IV. Ferramenta de visão computacional para a análise do movimento de barra fixa voltada para testes de aptidão física

*Este projeto é uma homenagem a duas entidades: aquelas que o leram e as que um dia poderão apreciá-lo.*

# Agradecimentos

Gostaria de expressar minha profunda gratidão aos meus professores pela paciência inestimável que demonstraram ao longo do meu percurso acadêmico. Sua dedicação em compartilhar conhecimento e orientação, mesmo diante das minhas dúvidas constantes, foi fundamental para o meu crescimento e aprendizado. Agradeço por investirem seu tempo e esforço em me auxiliar a compreender os desafios que encontrei, tornando minha jornada educacional mais rica e gratificante.

A todos os meus colegas pela paciência e compreensão que demonstraram quando, em uma sexta-feira à noite, me envolvi em questionamentos sobre o conceito do terceiro quartil. E é graças à nossa colaboração que pude aprofundar meu entendimento. O apoio e a camaradagem que compartilhamos durante nossas discussões foram essenciais para nossa jornada acadêmica conjunta.

Aos meus queridos pais por tudo que fizeram ao longo da minha vida. Seu amor incondicional, apoio constante e sacrifícios incansáveis moldaram a pessoa que me tornei hoje. Desde o início estiveram ao meu lado, guiando-me com sabedoria, proporcionando conforto nos momentos difíceis e celebrando minhas conquistas com alegria. Cada oportunidade que tive e cada passo que dei foram iluminados pelo exemplo inspirador que vocês sempre foram.

“Dattebayo” - (*Naruto Uzumaki*)

# Resumo

A flexão de braços na barra fixa, conhecida como movimento de barra fixa, é uma atividade física comum nos testes de aptidão física de concursos públicos para cargos em carreiras relacionadas à segurança pública. No entanto, um desafio na avaliação do Teste de Aptidão Física ([TAF](#)) é a possibilidade de fadiga de decisão por parte dos avaliadores no dia do exame. Este trabalho visa desenvolver uma ferramenta de visão computacional em nível de prova de conceito para analisar a corretude do movimento de barra fixa, considerando parâmetros presentes nos editais de concursos públicos na esfera militar. Além disso, pretende oferecer um feedback sobre o movimento de barra fixa, permitindo o aprimoramento da consciência corporal do indivíduo. Na elaboração deste trabalho, foi adotada uma abordagem que combina o processo usual de elaboração de um trabalho acadêmico com a estruturação das etapas de um sistema de visão computacional, pré processamento, segmentação, representação e descrição, reconhecimento e interpretação. Como resultado, foi desenvolvida uma ferramenta com a capacidade de identificar o movimento correto na barra fixa, com base nos editais dos últimos cinco anos de instituições como a Polícia Militar de Minas Gerais ([PMMG](#)), o Corpo de Bombeiros Militar de Minas Gerais ([CBMMG](#)), a Polícia Rodoviária Federal ([PRF](#)) e o Departamento Penitenciário de Minas Gerais ([DEPEN-MG](#)). A maioria dos critérios estabelecidos nos editais referentes ao movimento correto na barra fixa foram atendidos, com exceção da pegada pronada e da extensão da cervical. Apesar do planejamento para alcançar um tempo de processamento próximo ao tempo real de execução, a ferramenta acabou demandando aproximadamente 5,96 vezes mais tempo do que o previsto, impossibilitando sua utilização em tempo real.

**Palavras-chave:** visão computacional, Barra fixa, TAF.

# Abstract

The pull-up exercise is a prevalent physical activity in the physical fitness tests of public service exams for careers in law enforcement. However, a challenge in the evaluation of the Physical Aptitude Test is the potential for decision fatigue among assessors on the exam day. This study aims to develop a computer vision tool at the proof-of-concept level to analyze the correctness of pull-up movements, considering parameters outlined in military public service exam guidelines. Additionally, it aims to provide feedback on the pull-up form, facilitating the improvement of individuals' body awareness. In the development of this tool, an approach was adopted that combines the standard academic work creation process with the structuring of phases from a computer vision system: pre-processing, segmentation, representation and description, recognition, and interpretation. As a result, a tool was created with the ability to identify the correct pull-up motion, based on the guidelines from the last five years of institutions such as [PMMG](#), [CBMMG](#), [PRF](#), and [DEPEN-MG](#). Most criteria outlined in the guidelines regarding the correct pull-up movement were met, with the exception of the pronated grip and cervical extension. Despite the initial plan to achieve real-time processing, the tool ended up requiring approximately 5.96 times more time than anticipated, hindering its use in real-time applications.

**Keywords:** Pull-up, computer vision.

# Listas de ilustrações

Figura 1 – Movimento de barra fixa . . . . .	18
Figura 2 – Planos anatômicos . . . . .	19
Figura 3 – Composição de um imagem RGB . . . . .	22
Figura 4 – Representação do Pixel de uma imagem com 3 canais . . . . .	23
Figura 5 – Fluxograma das etapas do processamento de imagens digitais . . . . .	24
Figura 6 – Captura de pontos de referência para estimativa de pose humana . . . . .	24
Figura 7 – Representação da reta pelos parametros $\rho$ e $\theta$ . . . . .	25
Figura 8 – Representação de um ponto no espaço parametrizavel $\rho$ e $\theta$ . . . . .	26
Figura 9 – Representação da reta por meio do ponto de intersecção . . . . .	27
Figura 10 – Imagem em preto e branco com filtro de gaus . . . . .	29
Figura 11 – Intensidade da mudança de gradiente usando operador de Sobel . . . . .	30
Figura 12 – Supressao não maxima . . . . .	31
Figura 13 – Imagem pixelizada com blocos de 32x32 . . . . .	33
Figura 14 – Organização de um reconhecedor genérico . . . . .	35
Figura 15 – Diagrama de estados de $M_1$ . . . . .	37
Figura 16 – Pontos de referência . . . . .	39
Figura 17 – Diagrama de Estados do AFD desenvolvido . . . . .	49
Figura 18 – Ambiente de gravação e Mascara Preto e Branca criada apartir do mesmo	51
Figura 19 – Aplicação do filtro de Canny. . . . .	52
Figura 20 – Diagrama do processo de detecção da barra . . . . .	53
Figura 21 – Imagem rotacionada em relação a barra . . . . .	54
Figura 22 – Pontos de referência usados na aplicação . . . . .	56
Figura 23 – Filtragem pela cor de pele . . . . .	56
Figura 24 – Conversão em escala de cinza . . . . .	57
Figura 25 – Binarização da Imagem . . . . .	58
Figura 26 – Redução de ruidos com a suavização e binarização . . . . .	58
Figura 27 – Imagem Pixelizada e binarizada . . . . .	59
Figura 28 – Mascara da região de interesse . . . . .	60
Figura 29 – Do lado esquerdo a imagem segmentada e do lado direto sua representação	60
Figura 30 – Diagrama do processo de detecção de mão na barra . . . . .	61
Figura 31 – Segmento de reta representante dos membros superiores . . . . .	62
Figura 32 – Diagrama do fluxo da informação para verificar extensão do cotovelo .	63
Figura 33 – Diagrama Ultrapassagem do queixo a barra . . . . .	64
Figura 34 – Preprocessamento até a binarização da imagem . . . . .	65
Figura 35 – Representação da mascara para região de interesse . . . . .	66
Figura 36 – Captura da região de interesse . . . . .	67

Figura 37 – Estrutura do diretório baseado em 5 grande blocos . . . . .	68
Figura 38 – Estrutura do diretório “/model” . . . . .	68
Figura 39 – Estrutura do diretório “/view” . . . . .	69
Figura 40 – Estrutura do diretório “/midia” . . . . .	69
Figura 41 – Estrutura do diretório “/util” . . . . .	69
Figura 42 – Estrutura do diretório “/controller” . . . . .	70
Figura 43 – Fluxograma do processo executada pela ferramenta. . . . .	72
Figura 44 – Tela “player” . . . . .	73
Figura 45 – Tela “painel de controle” . . . . .	73
Figura 46 – Resultado final com todas as Flags ativas “Barra”,“Dados” e “EPH” .	74
Figura 47 – Tempo de execução da função “barra” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos .	77
Figura 48 – Tempo de execução da função “verify_inclination” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos . . . . .	77
Figura 49 – Tempo de execução da função “verify_eph” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos . . . . .	78
Figura 50 – Tempo de execução da função “verify_angle_member” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos . . . . .	78
Figura 51 – Tempo de execução da função “verify_char_AFD” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos . . . . .	79
Figura 52 – Tempo de execução da função “verify_maoBarra” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos . . . . .	79
Figura 53 – Tempo de execução da função “verify_extensaoCotovelo” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos . . . . .	80
Figura 54 – Tempo de execução da função “verify_ultrapassarBarra” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos . . . . .	80
Figura 55 – Tempo de execução da função “verify_movimentoQuadrilPerna” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos . . . . .	81
Figura 56 – Tempo de execução da função “verify_AFD” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos . . . . .	81
Figura 57 – Tempo de execução da função “process_cell” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos . . . . .	82

Figura 58 – Média de 10 iterações do percentual de tempo gasto por cada função que compõe “verify_char_AFD” . . . . .	83
Figura 59 – Tempo médio de execução das funções que compõe “process_cell” . . . . .	84
Figura 60 – Média de 10 iterações do percentual de tempo gasto por cada função associado ao processamento de um frame . . . . .	85

## **Lista de Siglas**

**AF** Autômato Finito

**AFD** Autômato Finito Determinístico

**CBMMG** Corpo de Bombeiros Militar de Minas Gerais

**DEPEN-MG** Departamento Penitenciário de Minas Gerais

**EPH** Estimativa de Pose Humana

**IA** Inteligência Artificial

**ML** Machine learning

**openCV** Open Source Computer Vision Library

**PMMG** Polícia Militar de Minas Gerais

**PRF** Polícia Rodoviária Federal

**RANSAC** Random Sample Consensus

**RGB** Red Green Blue

**SIFT** Transformações de Características Invariantes à Escala

**TAF** Teste de Aptidão Física

## **Lista de Símbolos**

$\delta$  Função de transição de um autômato finito.

$Q$  Conjunto de estados de um autômato finito.

$\Sigma$  Alfabeto de entrada de um autômato finito.

$\theta$  Ângulo entre a reta analisada e o eixo x do sistema de coordenadas.

$\rho$  Distância perpendicular da reta à origem do sistema de coordenadas.

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>14</b>
<b>1.1</b>	<b>Justificativa</b>	<b>15</b>
<b>1.2</b>	<b>Objetivos</b>	<b>16</b>
1.2.1	Objetivos Gerais	16
1.2.2	Objetivos Específicos	16
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA</b>	<b>17</b>
<b>2.1</b>	<b>Treinamento de barra fixa para o TAF</b>	<b>17</b>
2.1.1	Teste de Aptidão Física	17
2.1.2	Barra Fixa	17
2.1.3	Feedback	18
2.1.4	Planos anatômicos e eixo de movimento	19
2.1.5	Articulação	19
2.1.6	Flexão e Extensão	19
2.1.7	Inserção e Origem Muscular	20
2.1.8	Parte Distal	20
2.1.9	Cadeia Cinemática	20
2.1.10	Ativação Muscular	20
2.1.11	Velocidade de Contração	21
<b>2.2</b>	<b>Visão computacional</b>	<b>21</b>
2.2.1	Imagen Digital	21
2.2.2	Processamento de Imagem	23
2.2.3	Estimativa de pose humana	24
2.2.4	Transformada de Hough	25
2.2.5	Detector de bordas de Canny	27
2.2.5.1	Redução de ruido	28
2.2.5.2	Calculo do Gradiente	28
2.2.5.3	Supressão não máxima	30
2.2.5.4	Limite duplo	30
2.2.5.5	Limite de histerese	31
2.2.6	Algoritmo de rastreamento ou Tracking	32
2.2.7	Pixelização	33
2.2.8	Limiarização	33
<b>2.3</b>	<b>Machine learning</b>	<b>34</b>
<b>2.4</b>	<b>Automatos Finitos</b>	<b>34</b>

<b>3</b>	<b>TECNOLOGIAS E MÉTODOS . . . . .</b>	<b>38</b>
3.1	Materiais . . . . .	38
3.2	Método . . . . .	41
<b>4</b>	<b>PROJETO E DESENVOLVIMENTO . . . . .</b>	<b>43</b>
4.1	Uso de visão computacional associado ao treinamento físico . . . . .	43
4.2	Escolha da linguagem de programação para desenvolvimento da ferramenta . . . . .	44
4.3	Escolha da biblioteca para realização do processamento de imagem . . . . .	45
4.3.1	Open Source Computer Vision Library . . . . .	45
4.3.2	MediaPipe . . . . .	45
4.4	Estratégia para detecção do movimento correto de barra fixa . . . . .	46
4.5	Extração de informação frame a frame . . . . .	50
4.5.1	Detecção da barra . . . . .	50
4.5.2	Inclinação da barra . . . . .	53
4.5.3	Reconhecimento de pose humana . . . . .	54
4.5.4	Mão na barra . . . . .	56
4.5.5	Braço esticado . . . . .	61
4.5.6	Ultrapassagem do queixo a barra . . . . .	64
4.5.7	Movimentação do quadril . . . . .	67
4.6	Código fonte . . . . .	67
4.6.1	Hierarquia de diretório . . . . .	68
4.7	Código fonte . . . . .	70
<b>5</b>	<b>RESULTADOS . . . . .</b>	<b>72</b>
5.1	Ferramenta . . . . .	72
5.2	Parte Gráfica . . . . .	72
5.3	Tempo de execução . . . . .	75
<b>6</b>	<b>CONCLUSÃO . . . . .</b>	<b>87</b>
6.1	Pontos fracos e possíveis melhorias . . . . .	87
6.2	Trabalhos futuros . . . . .	87
	<b>REFERÊNCIAS . . . . .</b>	<b>88</b>

# 1 Introdução

O Teste de Aptidão Física ([TAF](#)) é uma das etapas comuns em concursos públicos para provimento de cargos em carreiras relacionadas à área de segurança pública, sua finalidade é atestar a capacidade do indivíduo em desempenhar funções específicas do cargo, por meio de parâmetros preestabelecidos em edital. Os exercícios avaliados comumente são: barra fixa; salto de impulso horizontal; corrida de 12km ou 2.400m; natação; flexão abdominal. Entretanto, esta é uma etapa muito negligenciada e acaba sendo responsável por um alto índice de reprovação ([NOBRE, 2018](#)).

Embora seja possível encontrar ferramentas computacionais que utilizam o processamento de imagens para auxiliar no aprimoramento de movimentos em diversas atividades físicas, como mostrado no trabalho de Franke ([2016](#)), Pádua ([2014](#)) e Paulichen ([2019](#)), é incomum encontrar ferramentas analíticas tecnológicas que auxiliem na melhoria ou indiquem a execução correta do movimento de barra fixa. No entanto, ainda existem poucas ferramentas computacionais analíticas que visam aprimorar exercícios exigidos durante o [TAF](#), possivelmente devido a questões de investimento/retorno financeiro ou outras justificativas.

A flexão de braços na barra fixa, também conhecida como movimento de barra fixa, é uma atividade física que tem como objetivo avaliar a resistência e a força dos músculos do tronco e dos membros superiores. Esse exercício é amplamente utilizado em diversos contextos, principalmente por sua facilidade de aplicação, baixo custo e alta reproduzibilidade ([MATTOS, 2015](#)). De acordo com o Governo da Paraíba, a realização do exercício de barra fixa foi identificada como o principal motivo de reprovação durante o [TAF](#) nos concursos de 2018 para a Polícia Militar da Paraíba (PMPB) e o Corpo de Bombeiros Militar da Paraíba (PMPB) ([PARAIBA, 2018](#)).

Portanto, este trabalho visa à criação de uma ferramenta que contribua para o aperfeiçoamento do movimento de barra fixa, proporcionando um feedback visual com informações sobre a angulação dos cotovelos e dos quadris, quantidade de barras executadas e o atendimento aos critérios estabelecidos nos [TAFs](#).

## 1.1 Justificativa

O uso da visão computacional associado ao movimento de barra fixa pode proporcionar uma análise mais objetiva da execução correta do exercício, minimizando a subjetividade na avaliação. Embora os critérios de avaliação do **TAF** sejam preestabelecidos nos editais, as decisões tomadas pelos avaliadores no dia do exame podem estar sujeitas à fadiga de decisão (LINDER et al., 2014). Portanto, a aplicação dessa tecnologia pode ser uma alternativa para reduzir os efeitos da subjetividade na avaliação, aumentando a confiabilidade e a justiça do processo seletivo.

A consciência corporal é construída por meio da percepção do indivíduo durante o exercício a fim de poder aprimorar o movimento e o domínio do corpo (ROCHA, 2009), portanto a percepção do posicionamento do corpo em relação ao espaço e as partes ou segmentos do corpo entre si são essenciais para o processo de formação da consciência corporal. Assim o uso dessa ferramenta pode ser útil para auxiliar na construção da consciência corporal, uma vez que seu foco é extrair informações biomecânicas a partir de uma série de imagens e fornecer um retorno ao indivíduo sobre a execução correta do movimento. Com esses dados em mãos, o indivíduo poderá aprimorar seu movimento e dominar seu corpo, tomando consciência do próprio corpo em relação ao movimento de barra fixa, aprimorando, dessa forma, a execução consciente do movimento de barra fixa.

Deste modo, a ferramenta proposta não apenas se apresenta como uma opção para o treinamento de barra fixa associado ao **TAF**, mas também tem o potencial de contribuir para a evolução do indivíduo na maturidade da consciência corporal.

## 1.2 Objetivos

Nessa seção serão relatados o objetivo geral e os específicos para o trabalho de conclusão de curso

### 1.2.1 Objetivos Gerais

Este trabalho tem como objetivo projetar e desenvolver uma ferramenta de visão computacional para análise do movimento de barra fixa associada ao [TAF](#).

### 1.2.2 Objetivos Específicos

- Desenvolver uma ferramenta de visão computacional (em nível de prova de conceito) que analise o movimento de barra fixa.
  - Identificar a corretude do movimento de barra fixa.
  - Definir um procedimento para estabelecer a corretude do movimento de barra fixa, tendo como parâmetros alguns editais de concursos públicos da esfera militar dos últimos cinco anos de instituições como a [PMMG](#), o [CBMMG](#), a [PRF](#) e o [DEPEN-MG](#)
  - Analisar o movimento de barra fixa de modo a auxiliar na formação da consciência corporal do indivíduo por meio de feedback.

## 2 Fundamentação Teórica

Neste capítulo são mostrados os fundamentos necessários para entender os conceitos abordados no presente trabalho.

### 2.1 Treinamento de barra fixa para o TAF

#### 2.1.1 Teste de Aptidão Física

O **TAF** é uma etapa comum em concursos públicos para provimento de cargos em carreiras relacionadas à área de segurança pública, seu objetivo é atestar a capacidade do indivíduo em desempenhar funções específicas do cargo por meio da avaliação de determinados exercícios. A avaliação é embasada em exercícios e parâmetros preestabelecidos em edital como por exemplo: quantidade de repetições de um determinado movimento para o exercício de barra fixa ou flexão abdominal, tempo gasto para translocação em uma determinada distância nas corridas de 12km, 2.400m ou natação, ou até mesmo a distância percorrida durante o salto de impulso horizontal([PAES, 2023](#)).

#### 2.1.2 Barra Fixa

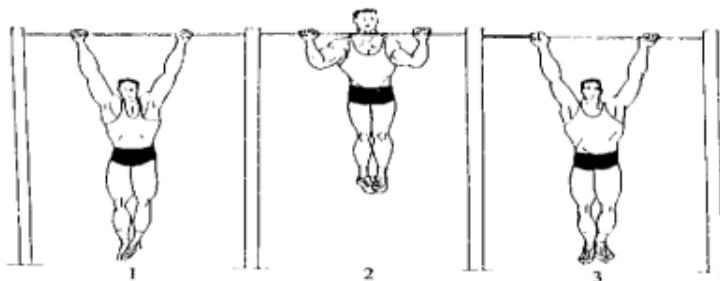
O teste de flexão em barra fixa, ou teste dinâmico de barra fixa, apresenta variações de acordo com os seguintes editais: Concurso público para admissão ao curso de formação de soldados Bombeiros Militar do quadro de praças (QP-BM) e do quadro de praças especialistas (QPE-BM) do Corpo de Bombeiros Militar de Minas Gerais para o ano de 2020 ([EDITAL..., 2018](#)); Concurso público para admissão ao curso de formação de soldados do quadro de praças especialistas da Polícia Militar de Minas Gerais ([EDITAL..., 2021a](#)); Concurso público para o provimento de vagas nos cargos de delegado de Polícia Federal, agente de Polícia Federal, escrivão de Polícia Federal e papiloscopista Policial Federal ([EDITAL..., 2021b](#)); Concurso público para o provimento de cargos da carreira de agente de segurança penitenciário/policial penal do quadro de pessoal da Secretaria de Estado de Justiça e Segurança ([EDITAL..., 2021c](#)).

No entanto, os princípios do movimento permanecem semelhantes:

- A barra fixa é instalada a uma altura tal, que o avaliado, mantendo-se pendurado, com os cotovelos em extensão, não tenha contato dos pés com o solo;
- A posição da pegada é pronada (dorso da mão voltado para o rosto) e a abertura das mãos corresponde à distância biacromial (largura dos ombros);

- Após assumir essa posição, o avaliado deverá elevar o corpo até que o queixo ultrapasse o nível da barra, após o que retornará à posição inicial;
- O movimento é repetido tantas vezes quanto possível, sem limite de tempo.
- Os cotovelos deverão estar em extensão total para o início de flexão;
- É permitido repouso entre um movimento e outro, contudo, o avaliado não poderá tocar os pés no solo;
- Não são permitidos movimentos de quadris ou pernas e extensão da coluna cervical como formas de auxiliar na execução da prova.
- A não extensão total dos cotovelos antes do início de nova execução é considerado um movimento incorreto, o qual não será computado no desempenho do candidato.
- Somente é contado o número de movimentos completados corretamente.

Figura 1 – Movimento de barra fixa



Fonte:(GOIÁS, 2017).

### 2.1.3 Feedback

Feedback é um processo de comunicação que envolve a transmissão de informações sobre o desempenho ou comportamento de uma pessoa ou grupo, com o objetivo de fornecer orientação e incentivo para a melhoria contínua.

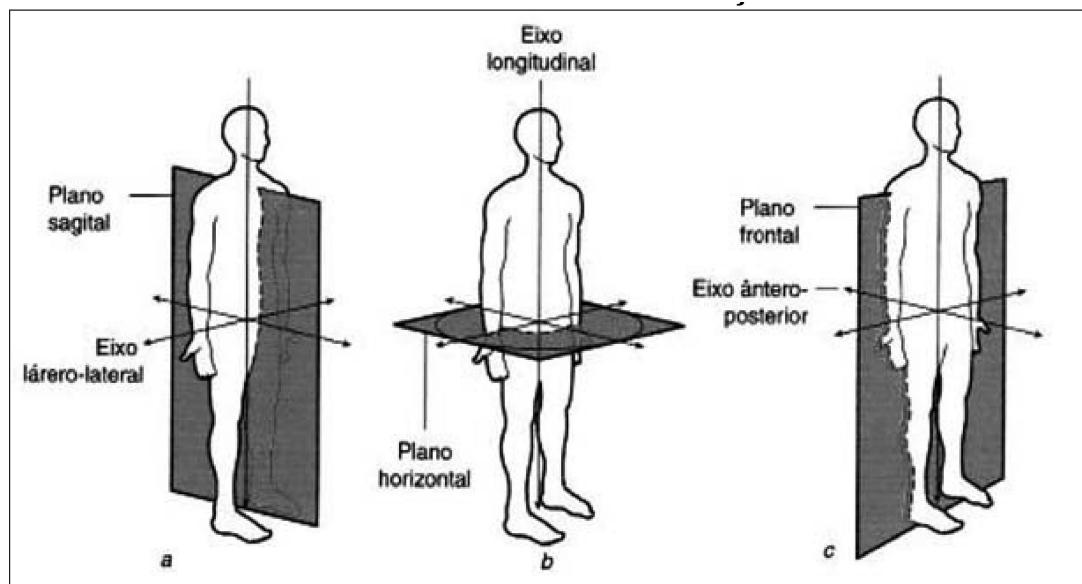
No contexto do treinamento físico, o feedback pode ser fornecido por um treinador ou por uma ferramenta tecnológica, como um aplicativo de treino ou um dispositivo de monitoramento de desempenho. Esse tipo de feedback permite que o indivíduo ajuste sua técnica ou intensidade do exercício durante a atividade, melhorando assim o resultado final (MARIANO et al., 2011).

### 2.1.4 Planos anatômicos e eixo de movimento

Os planos cardinais, também chamados de planos anatômicos, são três planos imaginários que dividem o corpo humano em seções para estudo e referência anatômica. São eles: o plano sagital, que divide o corpo em metades esquerda e direita; o plano frontal (ou coronal), que divide o corpo em partes anterior (frente) e posterior (costas); e o plano transversal (ou axial), que divide o corpo em partes superior e inferior (BRUNNSTROM, 1986).

Cada plano tem um eixo perpendicular utilizado para caracterizar os movimentos referentes a esse plano. O eixo sagital ou medial-lateral é responsável pelos movimentos de flexão e extensão, o eixo frontal ou anterior-posterior é responsável pelos movimentos de abdução e adução, e o eixo transversal, também conhecido como eixo látero-lateral, é responsável pelos movimentos de rotação medial e lateral(BRUNNSTROM, 1986).

Figura 2 – Planos anatômicos



Fonte:(CAZETTA; OLIVEIRA; TAVARES, 2019).

### 2.1.5 Articulação

A articulação é a conexão de duas ou mais superfícies ósseas que promovem movimento (DIAS, 2021)

### 2.1.6 Flexão e Extensão

Anatomicamente, o movimento de flexão refere-se à redução do ângulo entre dois ossos ou partes do corpo no plano sagital ao redor do eixo transverso, o que resulta em

um movimento que diminui a distância entre as duas estruturas. Em contraste, a extensão envolve aumentar o respectivo ângulo. ([KENHUB, 2021](#)).

### 2.1.7 Inserção e Origem Muscular

Inserção é a extremidade do músculo que está fixada a um ponto móvel ou a uma peça óssea que se desloca. Em contraste, a origem é a extremidade do músculo que está fixada a um ponto fixo ou a uma peça óssea que não se desloca ([SANTOS,](#)  ).

### 2.1.8 Parte Distal

A parte distal é o ponto mais afastado do tronco ou do ponto de origem ([THERRIE, 2021](#)).

### 2.1.9 Cadeia Cinemática

Os termos cadeia cinemática aberta ou fechada, são usados para demonstrar se o segmento mais distal da cadeia está fixado a algum objeto imóvel ou até mesmo o solo.

Sendo que cadeia cinemática aberta representa uma situação em que a extremidade do membro não está fixada ao solo ou a algum objeto imóvel e assim, o segmento está livre para se mover e cadeia cinemática fechada representa uma situação em que o segmento mais distal da cadeia está fixo ao solo ou a um objeto imóvel ([SILVA, 2015](#)).

### 2.1.10 Ativação Muscular: Concêntrica, Excêntrica e Isométrica

Existem diversas formas de ativação muscular, cada uma com características próprias. Uma delas é a ativação isométrica, na qual um músculo produz força sem que haja uma mudança aparente no ângulo articular. Também conhecida como contração estática ou sustentação, a ativação isométrica é importante para estabilizar as articulações durante atividades funcionais, ajudando a prevenir lesões ([BRUNNSTROM, 1986](#)).

A ativação concêntrica é outra forma de ativação muscular em que o músculo produz força encurtando-se, ou seja, os pontos de inserção proximal e distal do músculo se aproximam, o que resulta em movimento visível na direção da força. Essa atividade concêntrica produz aceleração dos segmentos do corpo ([BRUNNSTROM, 1986](#)).

Por fim, a ativação excêntrica ocorre quando o músculo produz força alongando-se, distanciando seus pontos de inserção proximal e distal e resultando em movimento visível, mas em direção oposta à da força produzida pelo músculo. Essa atividade excêntrica desacelera os segmentos corporais e oferece amortecimento ([BRUNNSTROM, 1986](#)).

Portanto o movimento concêntrico é também conhecido como trabalho positivo, em que o músculo exerce força para produzir movimento de uma articulação. Já o movimento

excêntrico é chamado de trabalho negativo, ocorrendo quando uma força externa produz o movimento articular enquanto o músculo controla seu nível de ocorrência.

### 2.1.11 Velocidade de contração

De acordo com Bunnstrom (1986) velocidade é uma medida da taxa de deslocamento em uma direção específica, é importante destacar que a velocidade de encurtamento ou alongamento do músculo é um fator crítico que afeta a força que pode ser desenvolvida durante a ativação muscular.

Conforme a velocidade da contração concêntrica se torna mais lenta, o desenvolvimento da força muscular aumenta pois a capacidade de produzir força no movimento concentrico está relacionada ao número de conexões entre filamentos de actina e miosina que podem ser formadas por unidade de tempo.

Por outro lado, quando o músculo se alonga durante a atividade, a relação entre velocidade de contração e produção de força é diferente da que ocorre com o encurtamento muscular, pois a força muscular aumenta com o aumento da velocidade durante a contração excêntrica até que a velocidade atinja um ponto no qual o músculo é incapaz de controlar a sobrecarga.

## 2.2 Visão computacional

Visão computacional é a ciência que estuda e desenvolve tecnologias que permitem extrair características de imagens capturadas por diferentes tipos de sensores e então permitem reconhecer, manipular e processar dados sobre os objetos que compõem a imagem capturada (BALLARD; BROWN, 1982)

### 2.2.1 Imagem Digital

Uma imagem monocromática ou simplesmente imagem, pode ser conceituada como uma função bidimensional que descreve a intensidade da luz através da notação  $f(x, y)$ , na qual  $x$  e  $y$  representam as coordenadas espaciais e o valor  $f$  representa o brilho naquele ponto, por sua vez, imagem digital é uma representação de uma imagem com um conjunto finito de valores, ou seja é a discretização da função  $f(x, y)$  tanto em cordenada quanto em brilho(GONZALEZ; WOODS, 2000).

Uma imagem digital pode ser representada como uma matriz com um numero fixo de linhas e colunas onde em cada celula é representado o nível de cinza naquele ponto específico, essas celulas são chamadas de elementos da imagem, em ingles *picture elements* abreviado como pixels. Portanto a discretização implica que uma imagem digital é uma

representação aproximada de uma cena real onde o pixel é a representação da menor unidade de uma imagem digital ([GONZALEZ; WOODS, 2000](#)).

A representação de uma imagem digital com  $N$  linhas e  $M$  colunas pode ser expressada da seguinte forma onde ( $0 < y < N - 1$ ) e ( $0 < x < M - 1$ ) e cada pixel corresponde ao valor de intensidade em uma posição (x,y) da imagem digital.

$$f(x, y) = \begin{bmatrix} f(0, 0) & f(0, 1) & \cdots & f(0, M - 1) \\ f(1, 0) & f(1, 1) & \cdots & f(1, M - 1) \\ \vdots & \vdots & \ddots & \vdots \\ f(N - 1, 0) & f(N - 1, 1) & \cdots & f(N - 1, M - 1) \end{bmatrix}$$

Embora Imagens Monocromáticas representem tons variados de uma única cor, elas podem representar qualquer cor, analogamente a isso, as imagens coloridas são a incorporação de diferentes canais monocromáticos, cada um dedicado a representar uma cor específica. Imagens coloridas são normalmente representadas por 3 canais de imagens monocromáticas sendo elas o vermelho, verde e azul em inglês Red Green Blue ([RGB](#)) ([ALTUNAY, 2019](#)).

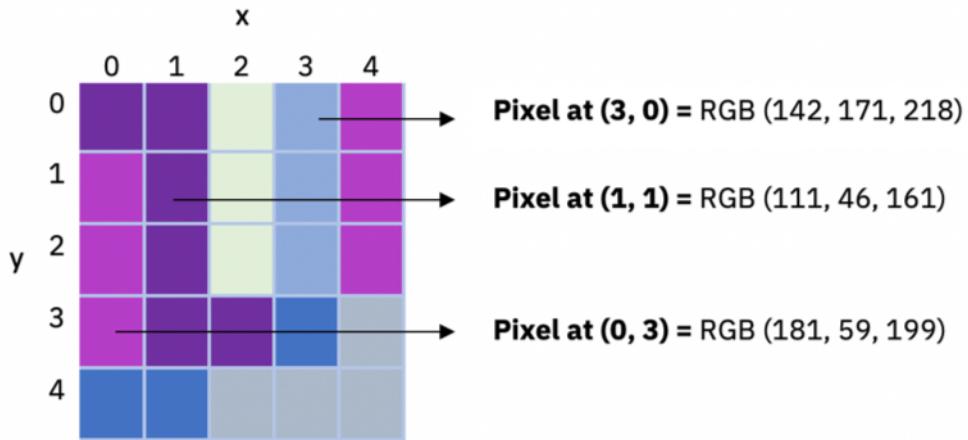
Figura 3 – Composição de um imagem RGB



Fonte: Elaborado pelo autor (2023).

Em uma imagem digital, a descrição da intensidade da luz pode ocorrer de maneiras distintas, relacionada à natureza da imagem. A descrição da intensidade pode adotar valores binários representados por branco e preto e são frequentemente associados a conceitos como a presença ou a ausência de um objeto. Para uma representação mais complexa de nuances de intensidade, texturas e detalhes é comum que a intensidade da luz seja descrita em 8 bits ou seja 256 tonalidades diferentes de cores, variando de 0 a 255, onde 0 é preto e 255 é branco. Por fim para uma riqueza maior de detalhes é comum que imagens coloridas sejam descritas por uma tripla de canais [RGB](#) onde cada canal representa uma cor, adotando valores de 8 bits por canal, portanto uma representação de 24 bits por pixel([ALTUNAY, 2019](#)).

Figura 4 – Representação do Pixel de uma imagem com 3 canais



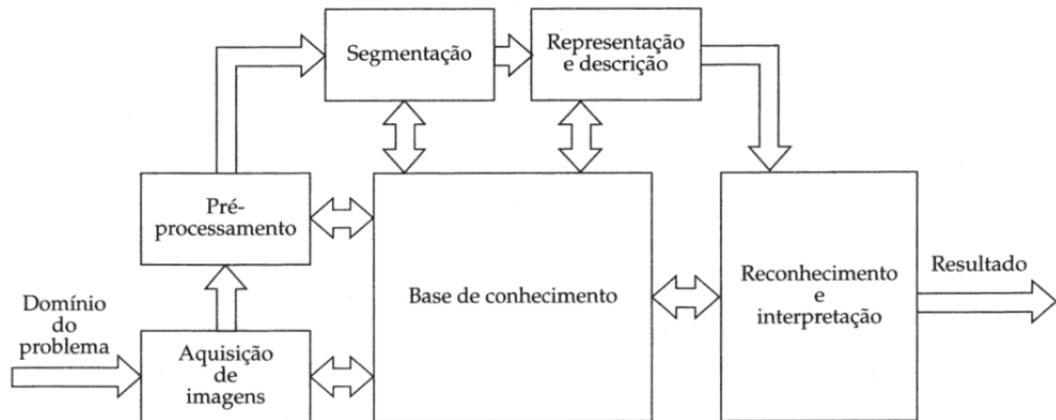
Fonte: ([ALTUNAY, 2019](#)).

## 2.2.2 Processamento de Imagem

O processamento de imagem refere-se à manipulação de imagens digitais para obter informações relevantes ou aprimorar a qualidade da imagem original por meio da aplicação de operações específicas. Em essência, o processamento de imagem atua como um tipo de processamento de sinais, onde a entrada é uma imagem digital e a saída resulta em uma nova representação da imagem ou informações derivadas dela. O objetivo central é extrair significado ou realçar características ao empregar uma série de técnicas e algoritmos, visando melhorar a utilidade ou interpretação das imagens ([ALTUNAY, 2019](#)).

O processamento de imagens abrange um conjunto de etapas interligadas, cada uma contribuindo para transformar uma imagem digital em informações comprehensíveis e relevantes. Inicialmente, ocorre a "aquisição de imagens", onde os dados visuais são capturados por meio de câmeras, scanners ou sensores. Em seguida, passa-se pelo "pré-processamento", onde a imagem é preparada para análise, eliminando ruídos e melhorando o contraste. A "segmentação" divide a imagem em regiões distintas, possibilitando focar em áreas específicas. A etapa de "representação e descrição" extraí características únicas de cada região, como texturas, formas e cores. Posteriormente, o "reconhecimento e interpretação" entra em cena, empregando algoritmos para identificar padrões, objetos ou informações relevantes com base nas características extraídas anteriormente. Esse processo cíclico de aquisição, pré-processamento, segmentação, representação e descrição, bem como reconhecimento e interpretação, forma o núcleo do processamento de imagens, permitindo a compreensão e análise de informações visuais de maneira mais eficaz e detalhada([GONZALEZ; WOODS, 2000](#)).

Figura 5 – Fluxograma das etapas do processamento de imagens digitais



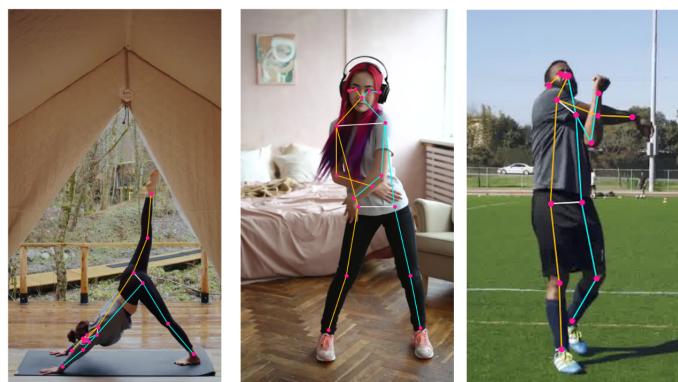
Fonte: (GONZALEZ; WOODS, 2000).

### 2.2.3 Estimativa de pose humana

Identificação de Pose ou Estimativa de Pose Humana (EPH) é um problema geral em Visão Computacional, que tem o objetivo de detectar a posição e a orientação de uma pessoa em imagens ou vídeos prevendo a localização de alguns pontos-chaves ,conhecidos tambem como *landmarks*, tais como cabeça, ombros, cotovelos, mãos, quadril, joelhos e pés.(VISÃO..., )

EPH pode ser bidimensional ou tridimensional, sendo o primeiro responsavel por estimar as cordenadas X Y e o segundo as cordenadas X Y Z de um espaço virtual que geralmente é inferido a partir de uma imagem bidimensional. A EPH pode tambem ser classificada de acordo com a quantidade de pessoas a serem estimadas podendo ser uma unica pessoa ou de várias pessoas o que interfere nas resoluções possiveis pois a estimativa de postura de várias pessoas é ligeiramente mais difícil do que o caso de uma única pessoa.(RAJ, 2019)

Figura 6 – Captura de pontos de referência para estimativa de pose humana



Fonte: (TENSORFLOW, 2021).

### 2.2.4 Transformada de Hough

A Transformada de Hough é um algoritmo amplamente empregado na detecção de linhas retas em imagens. Sua utilidade é especialmente evidente ao buscar a extração de informações relacionadas a linhas presentes na imagem independentemente da orientação, localização ou continuidade. Em suma a Transformada de Hough opera transformando coordenadas cartesianas  $(x, y)$  da imagem em uma representação polar  $(\rho, \theta)$  no espaço de parâmetros, também conhecido como plano de Hough, esse plano é usado como um espaço de votação, onde é verificado quantos pontos compartilham uma mesma reta assim as retas que acumulam mais votos no espaço de votação são consideradas as retas detectadas na imagem ([JAMUNDÁ, 2000](#)).

Uma reta pode ser descrita como:

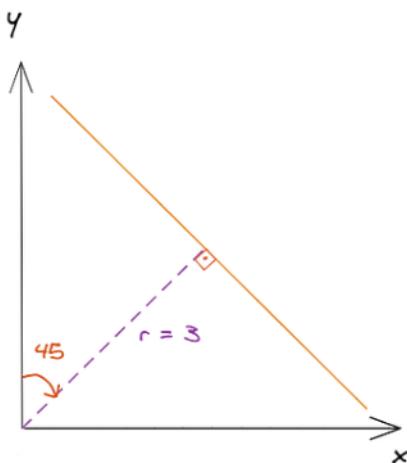
$$y = mx + b$$

As características desta reta são a inclinação  $m$  e a intersecção  $b$  assim, uma reta  $y = mx + b$  pode ser representada como um par ordenado  $(b, m)$ , porém em tal representação à medida em que a reta torna-se vertical, as magnitudes de  $b$  e  $m$  tendem ao infinito, contudo a mesma reta pode ser representada na forma:

$$\rho = x * \cos(\theta) + y * \sin(\theta)$$

Onde o parâmetro  $\rho$  é a mínima distância da reta a origem, centro do plano, e  $\theta$  é o ângulo de inclinação da normal à reta, sendo a normal uma linha perpendicular a direção da reta ([USP, 2023](#)).

Figura 7 – Representação da reta pelos parâmetros  $\rho$  e  $\theta$

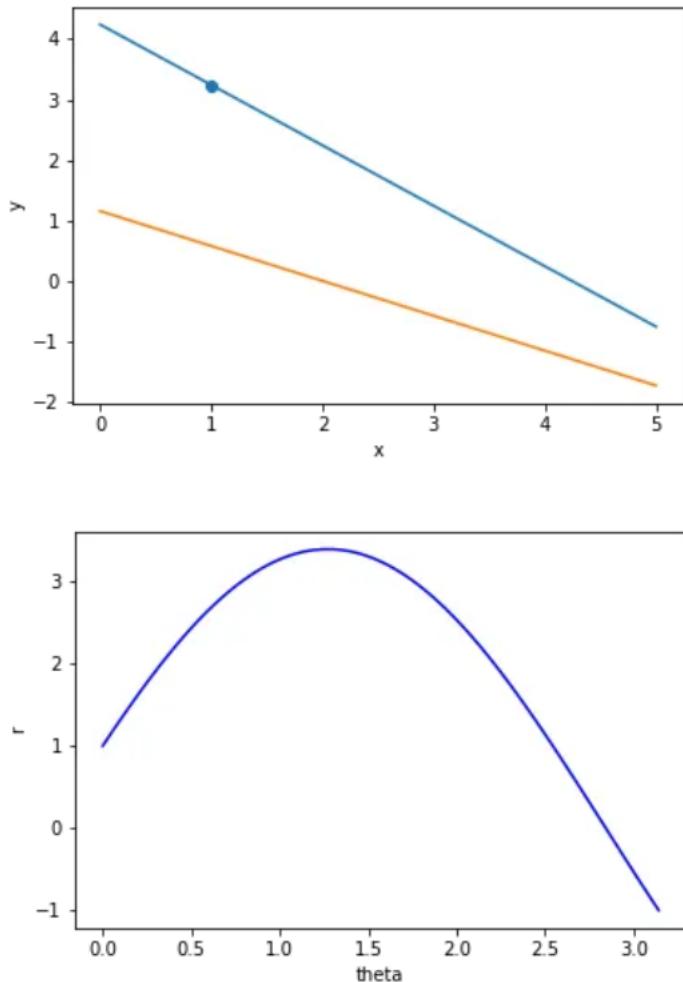


Fonte: ([ESTEVAM, 2021](#)).

O plano de Hough é um plano bidimensional onde os eixos representam os parâmetros, sendo eixo  $X$  representante do ângulo  $\theta$  e o eixo  $Y$  a representação da distância

$\rho$  (JAMUNDÁ, 2000). Nesse contexto, um ponto no plano da imagem é atravessado por inúmeras retas, e a combinação de todas essas retas resulta na formação de uma curva senoidal no espaço de Hough(USP, 2023).

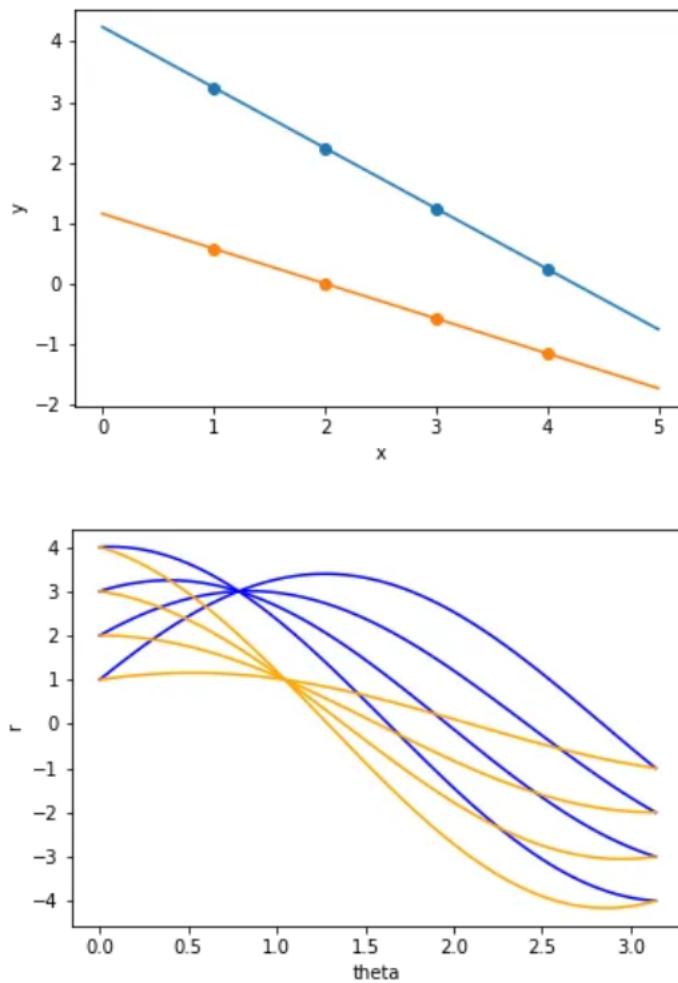
Figura 8 – Representação de um ponto no espaço parametrizável  $\rho$  e  $\theta$



Fonte: (ESTEVAM, 2021).

Dois pontos  $p$  e  $q$  no plano da imagem definem uma reta  $pq$  e correspondem a dois senóides no plano de Hough, a intersecção dos dois senóides representa a reta  $pq$  que passa pelos dois pontos no plano da imagem, analogamente 4 retas no plano da imagem correspondem a 4 senóides no plano de Hough que se acumulam em 4 pontos, portanto, para realizar a detecção de uma reta, é necessário examinar as interseções das curvas senoidais. Isso é alcançado por meio da utilização de um acumulador, que consiste em uma lista destinada a armazenar todos os cruzamentos identificados. O processo de detecção de retas, consiste em selecionar os cruzamentos que apresentam maior frequência na lista do acumulador (DUARTE, 2003).

Figura 9 – Representação da reta por meio do ponto de intersecção



Fonte: ([ESTEVAM, 2021](#)).

### 2.2.5 Detector de bordas de Canny

Um filtro de detecção de borda é um método utilizado no processamento de imagens para identificar mudanças abruptas na intensidade luminosa. Essas mudanças geralmente correspondem a limites entre diferentes objetos ou regiões na imagem. O filtro de detecção de borda realça essas mudanças, destacando os contornos e limites dos objetos presentes na imagem. Ele funciona calculando gradientes de intensidade em várias direções e destacando as regiões onde esses gradientes são mais acentuados. Em específico o filtro de Canny possui cinco etapas sendo elas: Redução de ruído; Cálculo de gradiente; Supressão não máxima; Limite duplo; Rastreamento de borda por histerese([SAHIR, 2019](#)).

### 2.2.5.1 Redução de ruido

A redução de ruido ocorre com a aplicação de um filtro Gaussiano, uma técnica de convolução de imagens que de acordo com um filtro substitue o valor de cada pixel da imagem pela média da vizinhança, tendo como efeito a suavização da imagem, uma vez que as altas frequências que correspondem às transições abruptas são atenuadas([CATARINA, 2023](#)).

A convolução é uma operação matemática que combina duas funções para criar uma terceira. No contexto de processamento de imagens, a imagem original atua como uma das funções e o filtro (ou kernel) atua como a segunda função. O filtro é uma matriz de números que define como a vizinhança de cada pixel na imagem será ponderada e combinada para gerar o novo valor desse pixel na imagem resultante. O processo de convolução envolve a superposição do filtro sobre cada região da imagem e a multiplicação elemento a elemento dos valores do filtro com os valores correspondentes da imagem. Os resultados dessas multiplicações são somados para produzir o novo valor do pixel na imagem de saída([ZHANG et al., 2020](#)).

Kernel 5x5 utilizado no filtro de Gaussiano ([CABRAL, n.d.](#)):

$$\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

O tamanho do kernel está relacionado com o efeito de desfoque, quanto menor o kernel, menos visível é o desfoque, no filtro de canny é usado um kernel 5x5.

### 2.2.5.2 Calculo do Gradiente

As bordas correspondem a uma mudança na intensidade dos pixels portanto o cálculo do gradiente em processamento de imagem refere-se à taxa de variação da intensidade luminosa em uma determinada região da imagem essa variação pode ser calculada usando operadores de detecção de borda na horizontal e vertical, ou seja convolucionando a imagem com os respectivos kernels:

Filtro Sobel Horizontal:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Filtro Sobel Vertical:

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Aplicando o filtro de sobel na horizontal tem se  $G_x$  que é o gradiente na direção horizontal e aplicando o filtro de sobel na vertical tem se  $G_y$  que é gradiente na direção vertical, fazendo uso dessas informações é possível chegar a  $G_r$  com a formula:

$$G_R = \sqrt{G_x^2 + G_y^2}$$

$G_r$  se refere ao valor resultante da combinação dos gradientes em duas direções (horizontal e vertical) pontos onde  $G_r$  é alto indicam mudanças bruscas de intensidade, muitas vezes correspondendo a bordas ou detalhes significativos na imagem.

A inclinação  $\theta$  que informa a direção do gradiente é calculado pela formula:

$$\theta(x, y) = \tan^{-1} \left( \frac{G_y}{G_x} \right)$$

Essa inclinação é perpendicular às arestas e arredondado para um dos quatro ângulos que representam as direções vertical, horizontal e duas diagonais.

Figura 10 – Imagem em preto e branco com filtro de gaus



Fonte: Elaborado pelo autor (2023)

Figura 11 – Intensidade da mudança de gradiente usando operador de Sobel



Fonte: Elaborado pelo autor (2023)

#### 2.2.5.3 Supressão não máxima

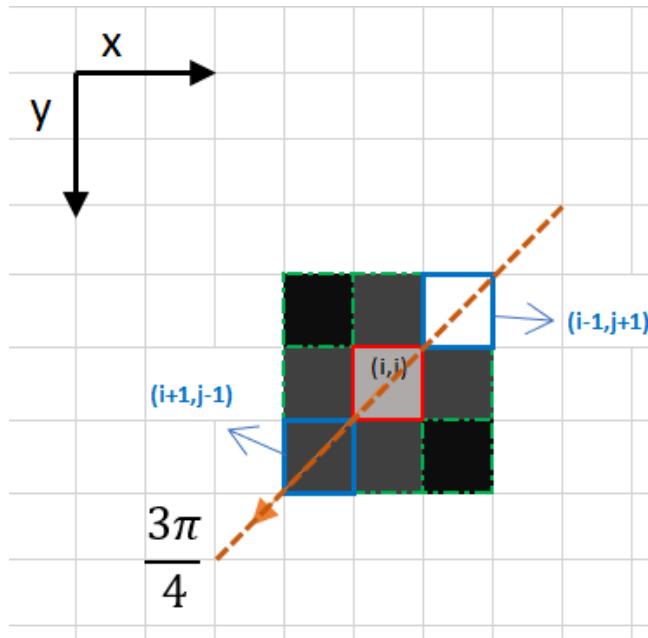
É desejável que a imagem resultante apresente bordas precisas e nítidas, para alcançar esse objetivo, aplicasse uma supressão não-máxima, a fim de refinar as bordas detectadas. Nesse processo, o algoritmo percorre todos os pontos da matriz de intensidade de gradiente, nos locais onde as bordas são encontradas, o algoritmo assegura que o pixel que é um máximo local dentro de sua vizinhança na orientação do gradiente seja mantido e os que não são, sejam suprimidos(SAHIR, 2019).

De maneira mais ilustrativa, o procedimento do algoritmo pode ser descrito da seguinte forma: o pixel que está destacado em vermelho é aquele que está sendo processado. A linha em laranja representa a orientação do gradiente ou seja é uma linha perpendicular à borda, os pixels de interesse presentes na vizinhança são aqueles que estão destacados em azul, uma vez que estão na mesma orientação do gradiente. Se o valor do pixel realçado em vermelho for maior do que o valor dos pixels vizinhos destacados em azul, então esse pixel será mantido. Caso contrário, ele será suprimido, assegurando que somente os pontos mais significativos sejam preservados na representação das bordas(SAHIR, 2019).

#### 2.2.5.4 Limite duplo

O "limiar duplo"é uma técnica usada no processamento de imagens para segmentar uma imagem com base nos valores de intensidade dos pixels. Essa técnica envolve a definição de dois limiares diferentes, um limiar inferior  $minVal$  e um limiar superior  $maxVal$ . Esses limiares dividem a escala de intensidade dos pixels em três intervalos:

Figura 12 – Supressão não maxima



Fonte: (SAHIR, 2019).

pixels abaixo de  $minVal$ , pixels entre  $minVal$  e  $maxVal$  e pixels acima de  $maxVal$ .

Com base nos limiares definidos, os pixels são definidos em três tipos: Pixels abaixo de  $minVal$  são desconsiderados pois não possuem uma intensidade mínima para ser considerado relevante. Pixels entre  $minVal$  e  $maxVal$  são atribuídos à região de transição no qual é necessário uma outra análise. Pixels acima de  $maxVal$  são atribuídos à região de interesse pois certamente contribuem para a borda final.(SAHIR, 2019)

#### 2.2.5.5 Limite de histerese

Baseado no limite duplo, o limite de histerese atua sobre os pixels que estão entre  $minVal$  e  $maxVal$ , transformando pixels possivelmente relevantes em certamente relevantes se e somente se pelo menos um dos pixels ao redor daquele que está sendo processado for relevante, conforme descrito abaixo:

Os pixels que estão entre  $minVal$  e  $maxVal$  são classificados como bordas ou não-bordas com base em sua conectividade. Se eles estiverem conectados a pixels superiores a  $maxVal$ , eles serão considerados parte das bordas, caso contrário serão descartados. Além das vizinhanças imediatas, verificações recursivas ou iterativas podem ser realizadas para explorar os pixels vizinhos dos pixels já marcados como relevantes. Isso permite estender a borda para pixels que não são diretamente vizinhos, mas que são alcançáveis por meio de pixels relevantes interligados.(OPENCV, 2023)

### 2.2.6 Algoritmo de rastreamento ou Tracking

O rastreamento, também conhecido como *tracking*, pode ser conceituado como um conjunto de procedimentos computacionais elaborados com o propósito de monitorar e seguir o movimento ou a posição de objetos em uma sequência de imagens ao longo do tempo. Um algoritmo de rastreamento eficiente utiliza informações acumuladas sobre o objeto até o momento baseado no modelo de movimento do objeto que engloba sua posição, sua velocidade e direção do movimento e baseado também no modelo de aparência, que proporciona informações sobre a aparência específica do objeto baseada nos quadros anteriores, ou seja, enquanto a detecção de objetos ocorre em quadros de imagem isolados, o algoritmo de rastreamento é aplicado com base em uma sequência temporal, permitindo uma análise contínua do movimento do objeto([VARGAS, 2020](#)).

O modelo de aparência desempenha um papel crucial no processo, permitindo a realização de uma busca precisa em uma pequena área próxima à localização prevista pelo modelo de movimento o que otimiza o processo de rastreamento, pois ocorre uma pesquisa local em vez de uma pesquisa global garantindo a identidade do objeto, sendo assim mesmo com uma oclusão do objeto é possível rastrear o objeto, porque o rastreamento leva em consideração a localização e a aparência do objeto no quadro anterior. Entretanto, esse processo pode enfrentar desafios consideráveis, como a perda do objeto quando ele fica obstruído por um período prolongado ou quando sua velocidade é tão elevada que o algoritmo de rastreamento não consegue acompanhá-lo, por isso é comum que os algoritmos de rastreamento acumulem erros ao longo do tempo, comprometendo a precisão da localização do objeto. Para mitigar essas questões inerentes aos algoritmos de rastreamento, uma abordagem utilizada é executar periodicamente um algoritmo de detecção para identificação da nova posição do objeto e, em seguida retransmitir essas informações para o rastreador, afim de contribuir para a correção dos erros acumulados e manter um acompanhamento mais preciso do objeto ao longo do tempo([VARGAS, 2020](#)).

Diversos algoritmos de rastreamento de objetos podem ser identificados, cada um com suas características particulares. Alguns exemplos notáveis incluem a Subtração de Plano de Fundo, o Casamento de Blocos, o Transformações de Características Invariantes à Escala ([SIFT](#)), e o Random Sample Consensus ([RANSAC](#)).

A Subtração de Plano de Fundo é capaz de identificar objetos destacando discrepâncias entre o plano de fundo e os objetos em movimento na cena. Por sua vez, o Casamento de Blocos realiza a comparação de blocos de pixels em diferentes quadros, o que lhe permite rastrear o movimento de objetos com eficácia. O algoritmo [SIFT](#) desempenha um papel crucial ao extrair características distintivas de uma imagem, possibilitando o rastreamento robusto de objetos, mesmo diante de variações na escala, rotação ou iluminação. Finalmente, o [RANSAC](#) é empregado para a estimativa de parâmetros de um modelo matemático previamente definido, de forma a se ajustar ao maior número possível

de dados, ou seja, aqueles que são comumente chamados de "inliers". Esses algoritmos oferecem soluções diversas e adaptáveis para a tarefa de rastreamento de objetos, cada um adequado a diferentes cenários e requisitos de aplicação (SILVA; FARIAS, 2017).

### 2.2.7 Pixelização

Quando uma imagem é ampliada ou exibida em uma resolução maior do que sua resolução original, os pixels individuais se tornam visíveis, resultando em uma aparência granulada ou “pixelizada” pois os detalhes da imagem original não podem ser representados com precisão em uma resolução mais baixa. Portanto o filtro de pixelização se resume em reduzir o número de valores de pixel distintos em uma imagem substituindo um bloco quadrado de valores de pixel por seu valor médio(NEWTON; SWEENEY; MALIN, 2005)

Figura 13 – Imagem pixelizada com blocos de 32x32



Fonte: Elaborado pelo autor (2023)

### 2.2.8 Limiarização

A limiarização, também conhecida como *thresholding*, é uma técnica de processamento de imagem usada para classificar os pixels em uma imagem com base em um valor específico, denominado limiar. Isso resulta na categorização dos pixels em dois grupos distintos, geralmente sendo aqueles que têm valores acima do limiar e aqueles que têm valores abaixo ou iguais a ele. Esse método é amplamente utilizado em processamento de imagem para destacar ou segmentar objetos de interesse em relação ao restante da imagem (GAZZONI et al., 2006).

A limiarização pode ser empregada de duas maneiras distintas: ela pode transformar uma imagem em escala de cinza em uma imagem binarizada, tornando os pixels acima do limiar em branco e os pixels abaixo do limiar em preto. Alternativamente, ela pode ser utilizada para enfatizar uma faixa específica de frequência, convertendo apenas os

pixels abaixo do limiar em preto ou apenas os pixels acima do limiar em branco([OPENCV, 2023a](#)).

## 2.3 Machine learning

Aprendizado de Máquina também conhecido como Machine learning ([ML](#)) é um ramo da Inteligência Artificial ([IA](#)) focado na análise de dados, que por meio de métodos estatísticos de aprendizado e otimização permite computadores analisarem conjuntos de dados e identificar padrões e tendências históricas para prever modelos futuros.([EDUCATION, 2020](#))

Comumente o algoritmo de aprendizado de máquina supervisionado consiste em três componentes: decisão, erro e otimização.

Existem três componentes no algoritmo típico de aprendizado de máquina supervisionado: decisão, erro e otimização.

O primeiro componente, decisão, é responsável por tomar decisões com base nas entradas fornecidas, que podem ser rotuladas ou não rotuladas. Esse componente é geralmente implementado como uma função matemática que mapeia as entradas para uma saída, seja uma classificação ou uma predição. Ele é responsável por transformar as entradas em informações úteis para o modelo, permitindo que o algoritmo aprenda a partir delas e tome decisões com base em novas entradas no futuro.

O segundo componente, erro, é responsável por avaliar a precisão da classificação ou da saída predita em relação à saída verdadeira, ou seja, os exemplos conhecidos. Esse componente é crucial porque permite uma análise do modelo.

O terceiro componente, otimização, é responsável por ajustar o modelo de modo que minimize o erro calculado pelo segundo componente e consequentemente seja possível melhorar a precisão da saída predita.

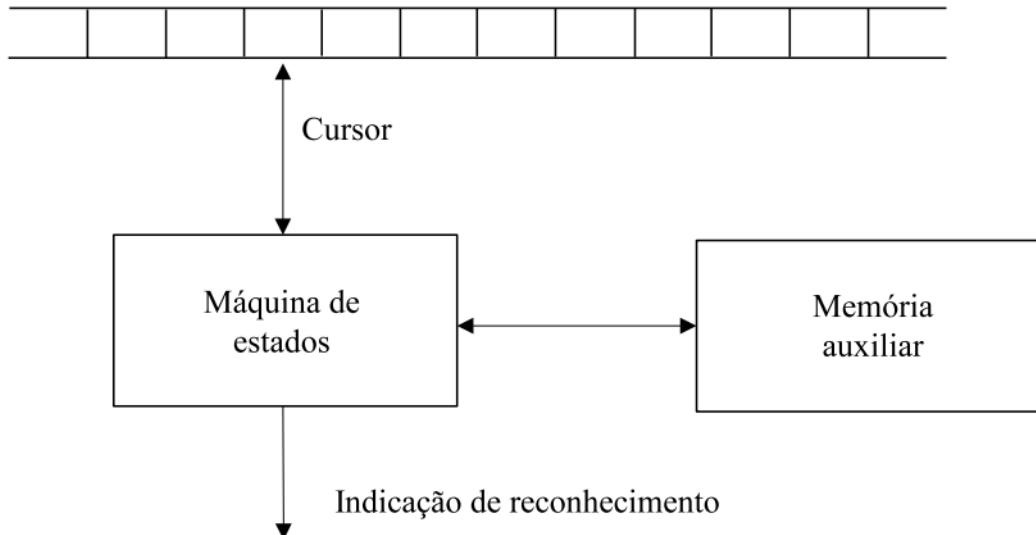
## 2.4 Automatos Finitos

Os reconhecedores são sistemas formais conhecidos também como autômatos capazes de aceitar todas as sentenças que pertençam a uma determinada linguagem, rejeitando todas as demais. Por esse motivo, constituem uma forma alternativa às gramáticas para a representação finita de linguagens.([RAMOS; UNIVASF, 2008](#))

Os reconhecedores apresentam quatro componentes fundamentais: uma memória (fita) contendo o texto de entrada do reconhecedor, um cursor, que indica o próximo elemento da fita a ser processado, uma máquina de estados finitos, sem memória, e uma memória auxiliar opcional.

Figura 14 – Organização de um reconhecedor genérico

Fita contendo a cadeia de entrada



Fonte: ([OLIVETE, 2020](#)).

A fita de entrada é composta pela cadeia que o reconhecedor precisa analisar. Ela é dividida em células, cada uma contendo um único símbolo da sequência, que faz parte do alfabeto de entrada escolhido. A sequência é organizada da esquerda para a direita, começando pelo símbolo mais à esquerda na fita. O cursor, chamado normalmente de cabeçote de acesso, é utilizado para ler os símbolos da fita de entrada. Ele aponta sempre para o próximo símbolo que deve ser processado. Os movimentos do cursor são controlados pela máquina de estados, podendo ser unidirecionais (em geral, movendo-se apenas para a direita) ou bidirecionais (movendo-se tanto para a esquerda quanto para a direita), dependendo do tipo de reconhecedor([RAMOS; UNIVASF, 2008](#)).

Alguns tipos de reconhecedores não apenas leem os símbolos da fita de entrada, mas também escrevem sobre a fita, substituindo os símbolos existentes por outros, de acordo com comandos definidos pela máquina de estados. A máquina de estados atua como o cérebro do reconhecedor, contendo um conjunto finito de estados que registram as informações coletadas e guiam o funcionamento do processo([RAMOS; UNIVASF, 2008](#)).

Um Autômato Finito ([AF](#)) é um modelo abstrato e simplificado de um sistema computacional que pode ser usado para reconhecer padrões em sequências de símbolos ([SIPSER, 2010](#)). Formalmente é representado como uma lista de cinco elementos com : um conjunto finito de estados representado por  $Q$ , um alfabeto de entrada representado por  $\Sigma$ , Funções de transição representada por  $\delta : Q \times \Sigma \rightarrow Q$ , estado inicial  $q_0 \in Q$  e um conjunto de estados finais  $F \subseteq Q$  portanto um automato finito é representado como uma

5-upla  $(Q, \Sigma, \delta, q_0, F)$

Sendo que o estado é uma representação de uma condição específica em que o autômato pode se encontrar durante a sua execução, o alfabeto é o conjunto finito de símbolos que o autômato pode receber como entrada, e as funções de transição definem como o autômato muda de um estado para outro em resposta a um símbolo de entrada.

As funções de transição são denotadas como  $\delta(e_0, 1) = e_1$ , em outras palavras, essa regra específica indica que um autômato no estado  $e_0$ , ao ler o caractere 1, se move para o estado  $e_1$ .

Também é possível representar um AF de modo mais visual por meio de um diagrama de estados onde é ilustrado os estados do autômato, suas transições e os símbolos de entrada. Cada estado é representado por um círculo e as transições são indicadas por setas entre esses círculos, rotuladas com os símbolos de entrada correspondentes.(SIPSER, 2010)

Em um diagrama de estados, cada estado é representado por um círculo, enquanto as transições são marcadas por setas que conectam esses círculos. Essas setas são rotuladas com os símbolos de entrada correspondentes. O estado inicial é indicado com uma seta apontando para ele, originada de um ponto vazio. Os estados de aceitação são ilustrados por círculos duplos.

Portanto um AF chamado  $M_1$  que aceita qualquer número binário ímpar pode ser representado de 2 maneiras, por uma 5-upla ou um diagrama de estados:

$M_1 = (Q, \Sigma, \delta, q_1, F)$ , onde:

$$Q = \{q_1, q_2, q_3\}$$

$$\Sigma = \{0,1\}$$

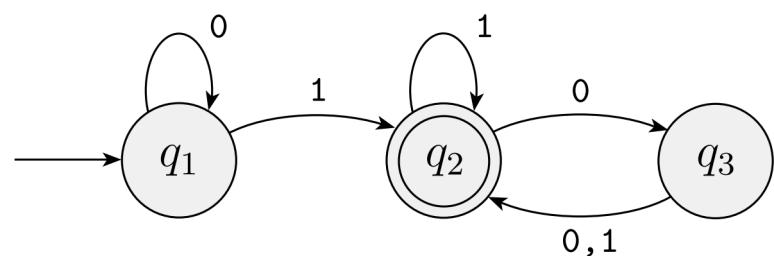
$\delta$  é descrito como

	0	1
$q_1$	$q_1$	$q_2$
$q_2$	$q_3$	$q_2$
$q_3$	$q_2$	$q_2$

$q_1$  é o estado inicial

$$F = \{q_2\}$$

Um reconhecedor e um AF são conceitos relacionados, mas cada um tem um papel específico sendo esse um modelo abstrato e aquele um sistema ou algoritmo que é capaz de identificar ou detectar algo em um conjunto de dados.

Figura 15 – Diagrama de estados de  $M_1$ 

Fonte: ([SALVIANO, 2022](#)).

# 3 Tecnologias e Métodos

Neste capítulo são apresentadas as metodologias adotadas, juntamente com as ferramentas, bibliotecas e bases de dados que foram utilizadas no desenvolvimento desse trabalho.

Em Materiais [3.1](#) são apresentados as ferramentas utilizadas, como: linguagem de programação, bibliotecas utilizadas e as especificações técnicas do sistema computacional utilizado. Na seção de Método [3.2](#) é descrito os passos do desenvolvimento a serem tomados para a reprodução deste trabalho, para que ao final seja possível executar os experimentos e verificar os resultados obtidos.

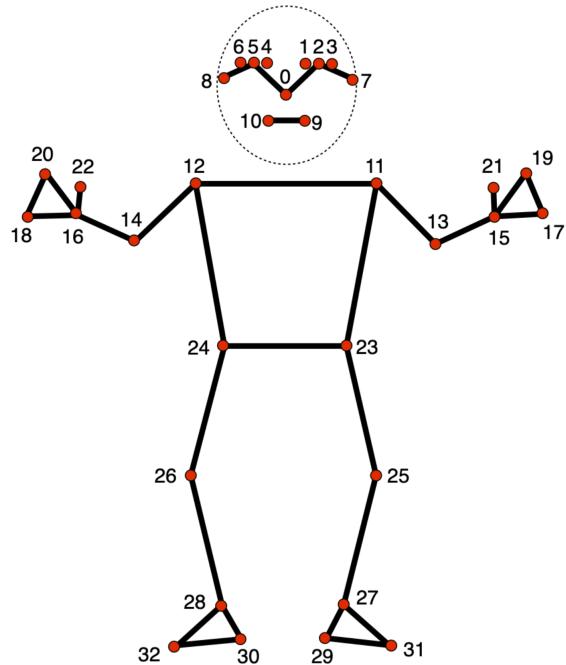
## 3.1 Materiais

Para o desenvolvimento deste trabalho foi necessário o uso de linguagens de programação e bibliotecas, abaixo segue uma lista com as ferramentas utilizadas:

- Python: Idealizada e desenvolvida no início dos anos 90, com o objetivo de otimizar a leitura de códigos, Python, é uma linguagem de programação de alto nível, dinâmica, interpretada, modular, multiplataforma e orientada a objetos com uma sintaxe simples e facilmente intercambiável com outras linguagens e com uma grande coleção de bibliotecas ([KRIGER, 2022](#)). Foi utilizado o python na versão 3.8.10.
- Open Source Computer Vision Library: Originalmente, desenvolvida pela Intel no ano de 2000, o openCV como tambem é conhecido é uma biblioteca multiplataforma, totalmente livre ao uso acadêmico e comercial, para o desenvolvimento de aplicativos na área de Visão computacional com interfaces em C++, Python, Java e MATLAB ([OPENCV, 2023](#)) Foi utilizado OpenCV na versão v4.7.0.
- MediaPipe: A MediaPipe é um projeto de código aberto desenvolvido pela Google, apresentando um conjunto de recursos e módulos pré-construídos, desenvolvidos com base no TensorFlow Lite. Esse framework destina-se a facilitar a implementação de técnicas de [IA](#) e [ML](#) em diversos ambientes, com foco na análise de mídia em tempo real e streaming contínuo. A característica distintiva da MediaPipe reside na execução integral do processamento dos dados de entrada diretamente no dispositivo, permitindo sua aplicação prática em uma ampla gama de plataformas, incluindo dispositivos móveis (Android, iOS), ambientes web, sistemas de desktop, dispositivos de borda e Internet das Coisas (IoT)([GOOGLE, Acesso em 2023](#)) Foi utilizada a versão v0.9.2.1 que é baseado em modelos ja treinados e usa uma cadeia de operações

para prever pontos de referência de pose. O primeiro modelo detecta a presença de corpos humanos dentro de um quadro de imagem, e o segundo modelo localiza até 33 pontos de referência nos corpos (GOOGLE, 2023a) que representam a localização aproximada das seguintes partes do corpo:

Figura 16 – Pontos de referência



Fonte: (GOOGLE, 2023a)

0 - nariz

1 - olho esquerdo (interno)

2 - olho esquerdo

3 - olho esquerdo (externo)

4 - olho direito (interno)

5 - olho direito

6 - olho direito (externo)

7 - orelha esquerda

8 - orelha direita

9 - boca (esquerda)

10 - boca (direita)

11 - ombro esquerdo

12 - ombro direito

- 13 - cotovelo esquerdo
- 14 - cotovelo direito
- 15 - pulso esquerdo
- 16 - pulso direito
- 17 - mindinho esquerdo
- 18 - mindinho direito
- 19 - indicador esquerdo
- 20 - indicador direito
- 21 - polegar esquerdo
- 22 - polegar direito
- 23 - quadril esquerdo
- 24 - quadril direito
- 25 - joelho esquerdo
- 26 - joelho direito
- 27 - tornozelo esquerdo
- 28 - tornozelo direito
- 29 - calcanhar esquerdo
- 30 - calcanhar direito
- 31 - dedo do pé esquerdo (indicador)
- 32 - dedo do pé direito (indicador)

- GitHub: O GitHub é uma plataforma de hospedagem em nuvem, lançada em 10 de abril de 2008 originalmente desenvolvida pela GitHub Inc e atualmente mantida pela Microsoft Corporation. Essa plataforma faz uso do Git um sistemas de controle de versão que mantém um registro abrangente e detalhado de todas as alterações realizadas no código-base, permitindo um rastreamento preciso do histórico de mudanças e do progresso do desenvolvimento ao longo do tempo. Sua capacidade de facilitar a colaboração eficaz e de manter um histórico completo. Além de desempenhar um papel essencial na colaboração entre desenvolvedores, proporcionando um ambiente colaborativo para projetos compartilhados([LONGEN, 2023](#)).

Na busca por uma abordagem robusta e fundamentada para a análise quantitativa do uso de memória, e tempo de execução, é imperativo descrever os materiais empregados na realização deste estudo e detalhar as características dos principais componentes relevantes para o desempenho durante o processamento da imagem, abaixo segue uma lista com as características:

- Notebook IdeaPad 3 82MF0004BR AMD Ryzen 7 5700U 15,6"8GB
- Pocessador
  - CPU: AMD Ryzen 7 5700U with Radeon Graphics
  - Frequência de clock: 1400 MHz
  - Número de núcleos: 16
  - Thread(s) per núcleo: 2
- Memória (RAM)
  - Tipo: DDR4
  - Capacidade: 8 GB
  - Frequencia: 1600 Mhz
- Placa de video
  - Nome: AMD Radeon RX Vega 8 Graphics
  - Driver em Uso: amdgpu
  - Capacidade de memória VRAM: 2.09 GB
- Memoria (SSD)
  - Capacidade de armazenamento: 256 GB
  - Destinado a swap: 50 GB
  - Interface: M2
  - Protocolo: NVMe
- Sistema operacional GNU/Linux
  - Distribuição: Linux Mint 20.2
  - Kernel: 5.15.0-76-generic
  - LSB Version: core-11.1.0ubuntu2-noarch:printing-11.1.0ubuntu2-noarch:security-11.1.0ubuntu2-noarch

## 3.2 Método

No desenvolvimento deste trabalho, adotou-se uma abordagem que combina processo usual de elaboração de um trabalho acadêmico com a estruturação das etapas de um sistema de visão computacional, fazendo uso do pré processamento, segmentação, representação e descrição, reconhecimento e interpretação conforme documentado na literatura ([GONZALEZ; WOODS, 2000](#)). Essa abordagem foi escolhida para o desenvolvimento de

um artefato em nível de prova de conceito, envolvendo os conteúdos abordados, o que torna este trabalho uma contribuição original e prática para a área em questão. Portanto, foram realizadas as seguintes etapas:

- Realização de revisão da literatura abordando o uso de visão computacional associado ao treinamento físico.
- Escolha da linguagem de programação para desenvolvimento da ferramenta.
- Escolha das tecnologias para realização do processamento de imagem.
- Escolha da ferramenta utilizada para a detecção de pose humana.
- Realização de revisão da literatura abordando a importância do movimento excentrico para o ganho de força
- Criação de uma estratégia para detecção do movimento correto de barra fixa
- Desenvolvimento de uma Ferramenta computacional para detecção da execução correta da barra fixa.
- Avaliação dos resultados.

# 4 Projeto e Desenvolvimento

Nesta seção são detalhados os passos e decisões tomadas ao decorrer do desenvolvimento deste trabalho. São apresentadas as etapas de implementação e integração com outras soluções utilizadas, assim como a execução e comparação dos resultados obtidos.

## 4.1 Uso de visão computacional associado ao treinamento físico

Foi feita uma revisão bibliográfica com objetivo de encontrar ferramentas ou trabalhos disponíveis que associam a visão computacional a algum tipo de atividade física, portanto foram formuladas algumas questões de pesquisa para identificar trabalhos pertinentes que possam agregar a este trabalho de alguma forma, são elas:

- Q1: Em quais atividades físicas foi utilizada a visão computacional como ferramenta?
- Q2: Como a visão computacional foi empregada em determinada atividade física?
- Q3: Quais métricas ou parâmetros são frequentemente extraídos das imagens

No decorrer desta revisão, foram identificados diversos estudos relevantes na área, com destaque especial para o trabalho intitulado “Análise postural digital para ciclistas” ([FRANKE, 2016](#)). Um dos objetivos desse estudo foi o desenvolvimento de um sistema digital denominado BFit4All. Esse sistema tem como objetivo identificar e propor ajustes na bicicleta com base nas análises posturais de ciclistas durante a atividade de pedalar por meio de fotografias. É importante ressaltar que a abordagem adotada nesse trabalho para a segmentação da imagem digital faz uso de marcadores fixados ao corpo do ciclista para destacar os pontos de referência e assim calcular as angulações entre eles. Embora essa metodologia revele-se eficaz em diversos aspectos, é válido destacar que esta implementação requer uma intervenção prévia à captura do vídeo.

Também foram identificados outros trabalhos relevantes, como os de Pádua ([2014](#)) e Paulichen ([2019](#)). Esses estudos abordam o desenvolvimento de sistemas de visão computacional que utilizam algoritmos de rastreamento de objetos, tais sistemas extraem informações detalhadas sobre o posicionamento, trajetória, velocidades e distâncias percorridas pelos jogadores de futsal. As informações resultantes desse processo proporcionam indicadores de natureza tática e fisiológica, que se mostram valiosos para a equipe técnica. Uma abordagem interessante que foi adotada no trabalho de Paulichen para o rastreamento de uma área de interesse foi a aplicação do filtro de Kalman, uma técnica de estimativa e previsão que combina informações medidas e modelos matemáticos para melhorar a precisão de uma estimativa reduzindo o ruído e as incertezas nas medições.

Em relação ferramentas computacionais para análise e aprimoramento do treinamento, é interessante destacar a referência ao Dartfish, feita no trabalho de Franke (2016). O Dartfish é uma plataforma amplamente adotada por treinadores, atletas e equipes esportivas, e seu destaque é reforçado pelo fato de ter sido utilizado por 73 % dos medalhistas olímpicos em Pequim. Essa plataforma de software comercial oferece uma abordagem abrangente para a otimização do desempenho esportivo por meio da análise de vídeo. Sua capacidade de rastrear objetos e movimentos em vídeos permite uma análise minuciosa e direcionada a elementos específicos. Ao segmentar e examinar detalhadamente o posicionamento, trajetória e dinâmica dos movimentos dos atletas, o Dartfish proporciona insights valiosos facilitando a identificação de padrões, embasamento de decisões e implementação de ajustes táticos e técnicos com base em informações visuais concretas (DARTFISH..., ). Embora essa plataforma seja uma das mais conceituadas soluções de análise de vídeo para desempenho esportivo vale destacar que ela é intimamente dependente de um profissional qualificado para analisar e aconselhar o treinamento esportivo.

## 4.2 Escolha da linguagem de programação para desenvolvimento da ferramenta

A escolha da linguagem de programação para a implementação deste projeto foi baseada nos requisitos do sistema proposto. Dentre as opções consideradas, JavaScript e Python, optou-se pelo uso da linguagem Python em sua versão 3.8.10 devido a uma série de fatores que se alinham de forma ideal com os objetivos deste trabalho sendo eles:

A facilidade de manutenção do código ao longo do ciclo de vida do sistema pois uma linguagem de programação com uma sintaxe clara e legível é particularmente vantajosa para a produção de um código facilmente comprehensível facilitando significativamente as atividades de manutenção contínua e de depuração, um aspecto que se torna crucial na busca pela robustez e estabilidade do sistema ao longo do tempo (MARTIN, 2008).

Agregou à escolha o fato da linguagem escolhida ter uma ampla biblioteca padrão e um vasto ecossistema de pacotes de terceiros que permitem a manipulação de imagens.

Dessa forma, a escolha do Python reflete diretamente na minimização dos desafios associados à evolução e manutenção do código-fonte, contribuindo para um desenvolvimento mais fluido e eficaz do projeto focado na produtividade.

## 4.3 Escolha da biblioteca para realização do processamento de imagem

Nesta seção do trabalho, será abordado o processo de escolha das bibliotecas, levando em consideração os critérios relevantes para assegurar a eficácia e o desempenho na manipulação de imagens.

### 4.3.1 Open Source Computer Vision Library

Para a realização do processamento de imagem, a escolha recaiu sobre a biblioteca Open Source Computer Vision Library ([openCV](#)). A decisão de adotá-la foi primordialmente embasada na sua perfeita integração com a linguagem selecionada, Python.

A sólida reputação do [openCV](#) na indústria, usado por empresas de renome como a Microsoft, Google e Intel, entre outras, torna-o uma escolha altamente relevante. Além disso, sua proeminente utilização no meio acadêmico oferece uma perspectiva valiosa de compartilhamento de conhecimento e habilidades para futuros projetos e oportunidades ([OPENCV, 2023b](#)).

A extensa documentação disponível, juntamente com uma extensa comunidade engajada e ativa, simplificam tanto a compreensão quanto a implementação. Assim, a escolha do [openCV](#) como biblioteca para o processamento de imagem é respaldada pelo potencial de eficiência e facilidade de implementação que essa escolha oferece ([OPENCV, 2023b](#)).

Por fim e um dos mais importantes aspectos levados em consideração foi a eficácia e o desempenho do [openCV](#) que é escrito em C++ e otimizado para aproveitar recursos de hardware, o que resulta em alto desempenho e eficiência no processamento de imagens.

### 4.3.2 MediaPipe

Dentro do contexto do processamento de imagem, optou-se por empregar a biblioteca MediaPipe para a detecção de poses. A decisão de incorporá-la foi fortemente motivada pela sua perfeita integração com a linguagem Python, destacando-se assim como uma alternativa coesa.

Vale ressaltar que a biblioteca MediaPipe é voltada para a simplificação da implementação de técnicas de [IA](#) e [ML](#) em diversos cenários, com um enfoque específico na análise de mídia. Seu diferencial reside na disponibilidade de modelos pré-treinados e customizáveis, permitindo uma integração eficaz e adaptável ao projeto em questão ([GOOGLE, 2023b](#)).

Por atuar com uma aplicação de [IA](#) e [ML](#) que trabalha com modelos já treinados,

o processamento dos dados de entrada ocorre completamente no dispositivo o que foi um ponto relevante, garantindo a persistência e funcionalidade contínua da ferramenta desenvolvida, independentemente da ação do tempo em relação a disponibilidade de servidores externos.

Um outro ponto relevante para a escolha dessa ferramenta foi o fato de que ela se destaca pela capacidade de fornecer resultados precisos e real na detecção de poses o que possibilita a emissão de um feedback por parte da ferramenta desenvolvida.

#### 4.4 Estratégia para detecção do movimento correto de barra fixa

Esta seção delineia a estratégia utilizada para detectar o movimento correto da barra fixa na ferramenta desenvolvida. O movimento de barra fixa se desdobra em um padrão que segue etapas distintas, as quais podem ser divididas em três fases bem definidas.

A primeira etapa engloba a posição inicial, onde o praticante se encontra pendurado na barra em pegada pronada, com os cotovelos estendidos e sem que os pés façam contato com o solo. A segunda etapa engloba a fase de contração concêntrica, onde o indivíduo eleva seu corpo até que seu queixo ultrapasse o nível da barra, sem fazer movimentos de quadris ou pernas e com extensão da coluna cervical como forma de auxílio na execução do movimento. Por fim, a terceira etapa engloba a fase de contração excêntrica, que envolve o retorno do corpo à posição inicial após a ultrapassagem da barra. Cabe salientar que a meta é fazer com que o queixo ultrapasse a barra independentemente de se manter assim por muito ou pouco tempo. Portanto, é possível adicionar mais uma fase, totalizando quatro, mas a essência do movimento se dá em três fases.

Dessa forma, optou-se por adotar a ideia de um reconhecedor para analisar o movimento de barra fixa. Nesse contexto, a fita é representada por um vídeo, no qual cada quadro do vídeo, comumente chamado de *frame*, representa uma célula da fita. Os *frames* são submetidos a um processamento afim de extrair informações cruciais que serão usadas na máquina como um caractere presente na fita. Para conduzir essa análise, é usada a ideia de uma máquina com cursor fixo que desloca a fita de maneira linear para a esquerda, lendo sequencialmente os caracteres presentes à direita. Além disso, essa abordagem incorpora uma memória auxiliar, responsável por registrar as transições entre dois estados específicos. Tal estratégia é implementada visando a precisão na contabilização dos movimentos corretos realizados durante o teste, conferindo, assim, uma avaliação meticulosa e sistemática do movimento de barra fixa.

Em suma, foi elaborado um Autômato Finito Determinístico (**AFD**) que inicia em um estado de preparação e avança para o estado de início somente quando as mãos estão na barra e os braços estão esticados.

O movimento concêntrico ocorre quando, partindo do estado inicial, os braços deixam de estar esticados e se mantêm nessa posição até que a meta seja alcançada ou ocorra um movimento ilegal dos quadris.

O movimento de barra é contabilizado no momento em que o **AFD** transita do estado concêntrico para o estado de meta. O estado de meta permanece enquanto o queixo estiver sobre a barra.

O movimento excêntrico começa quando, a partir do estado de meta, o queixo deixa de estar sobre a barra. É importante salientar que, como a meta foi atingida, os movimentos das pernas nesse momento não influenciam na contabilização dos movimentos de barra fixa. No entanto, mesmo não sendo contados, não deixa de ser um movimento indesejado. O movimento excêntrico se mantém até que o executor retorne à posição inicial.

A qualquer momento em que o executor retirar a mão da barra, o teste é automaticamente encerrado, levando o **AFD** ao estado Fim.

O **AFD** construído opera em relação a um alfabeto específico. Esse alfabeto é constituído por características extraídas de cada *frame*. Cada símbolo do alfabeto é representado por uma 4-upla de características (A,B,C,D). Sendo as características explicadas a seguir:

- A Presença das mãos na barra.
- B Ocorrência da extensão dos cotovelos.
- C Ocorrência da ultrapassagem do queixo sobre a barra.
- D Existência de movimento nos quadris.

Cada Elemento da 4-upla pode receber um valor binário, sendo 0 para representar a ausencia ou 1 para representar a existência de tal característica em um determinado *frame*, portanto o alfabeto do **AFD** pode ser representado da seguinte forma:

$$\Sigma = \{(0, 0, 0, 0), (0, 0, 0, 1), (0, 0, 1, 0), (0, 0, 1, 1), \\ (0, 1, 0, 0), (0, 1, 0, 1), (0, 1, 1, 0), (0, 1, 1, 1), \\ (1, 0, 0, 0), (1, 0, 0, 1), (1, 0, 1, 0), (1, 0, 1, 1), \\ (1, 1, 0, 0), (1, 1, 0, 1), (1, 1, 1, 0), (1, 1, 1, 1)\}$$

O **AFD** desenvolvido consiste em um conjunto de sete estados:

$$\{\text{Preparação, Início, Concêntrica, Meta, Excêntrica, Erro, Fim}\}$$

O estado “Preparação” é o estado inicial do **AFD** representado qualquer momento que antecede o começo do teste de barra fixa como um todo. Um exemplo disso são os momentos que o executor se aproxima da barra fixa e se prepara para assumir a posição inicial.

O estado “Início” simboliza o começo do movimento da barra fixa. Nesse ponto, o executor deve estar pendurado na barra com a pegada pronada, cotovelos estendidos e sem contato dos pés com o solo.

O estado “Concêntrica” simboliza a transição da posição inicial até a realização da meta que é a ultrapassagem do queixo sobre a barra, esse estado tem o nome de concêntrica pois para o alcance da meta é necessário que o executor recrute as fibras musculares dos membros superiores realizando uma contração concêntrica afim de alcançar a meta.

O estado “Meta” simboliza a realização da meta ou seja houve a ultrapassagem do queixo sobre a barra.

O estado “Excentrica” simboliza a transição de volta a posição inicial ou seja pendurado na barra com a pegada pronada, cotovelos estendidos e sem contato dos pés com o solo. Esse estado tem o nome de excêntrica pois para voltar a posição inicial é necessário que o executor realize uma contração excentrica. até o momento em que os cotovelos fiquem totalmente esticados

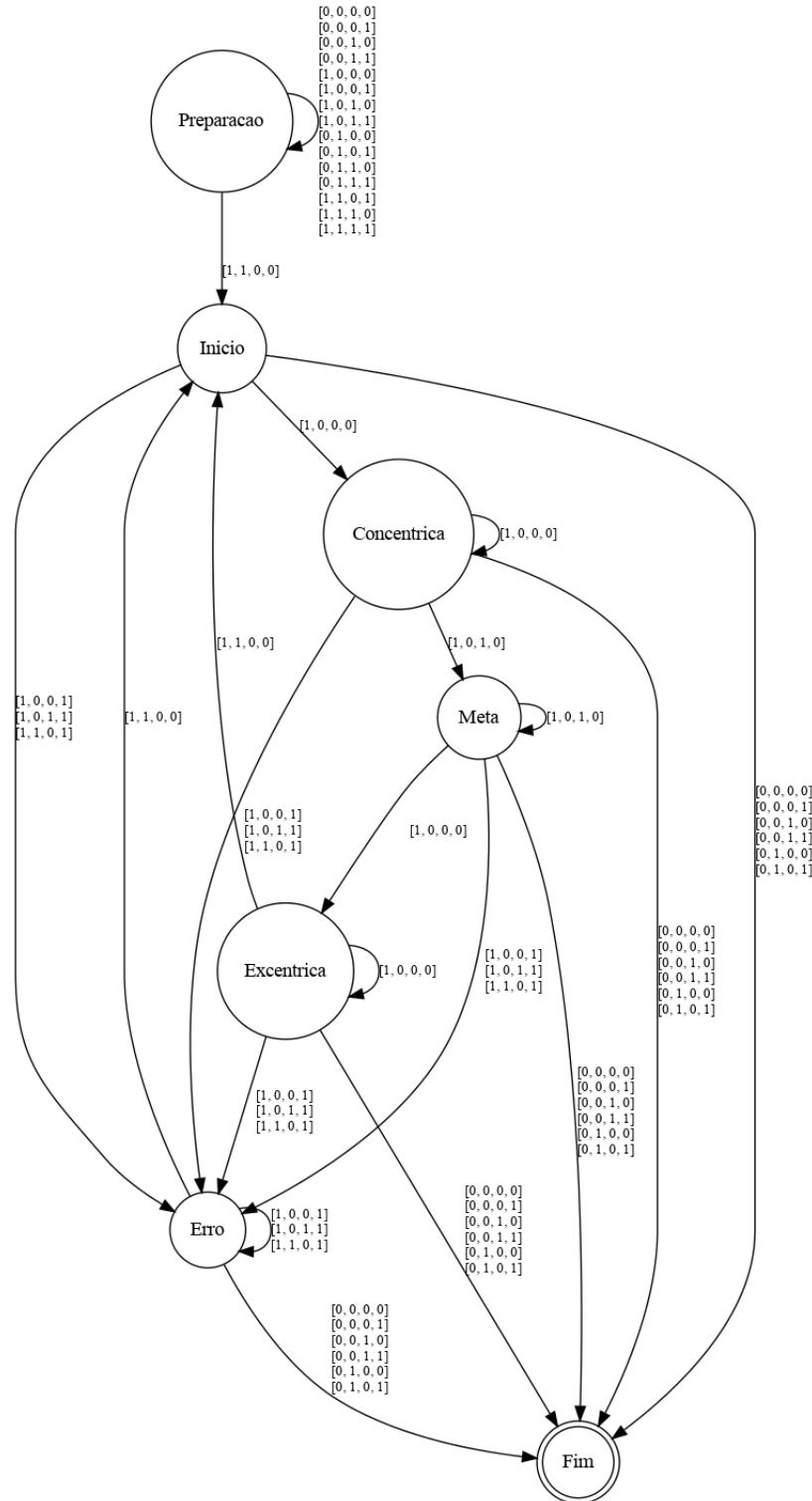
O estado “Erro” representa todas as instâncias em que o teste está em andamento, mas devido a uma falha do executor, parâmetros são identificados que exigem um retorno à posição inicial para que o movimento seja considerado correto. Por exemplo, isso pode ocorrer quando o executor realiza um movimento de balanço com o quadril para facilitar a execução do movimento. Em outras palavras, se o executor alcança a meta de ultrapassar o queixo acima da barra, mas utilizou o impulso das pernas para isso, o movimento não será considerado válido. Portanto, é necessário retornar à posição inicial e reiniciar o movimento.

Por ultimo o estado “Fim” representa o momento em que o teste é encerrado e os movimentos não são mais contabilizados.

Portanto o conjunto de estados finais do AFD pode ser representado da seguinte forma:  $F = \{Fim\}$

As funções de transições estão representadas no diagrama de estados a seguir:

Figura 17 – Diagrama de Estados do AFD desenvolvido



Fonte: Elaborado pelo autor (2023)

## 4.5 Extração de informação frame a frame

Dentro do processamento de imagem, a segmentação é a etapa onde é dividido a imagem em regiões ou componentes significantes que possam ser analisados e processados de forma independente. Em outras palavras, a segmentação visa identificar áreas distintas na imagem que correspondem a objetos, padrões, bordas ou características de interesse (GONZALEZ; WOODS, 2000).

O objetivo da ferramenta é operar em um ambiente controlado. Entretanto, devido às características específicas da infraestrutura do local de teste, tornou-se essencial incorporar uma máscara para uma manipulação adequada de cada quadro. Com esse propósito, uma máscara previamente elaborada a partir do primeiro quadro do vídeo foi aplicada, utilizando a função `bitwise_and`, disponibilizada na biblioteca `openCV`. A operação `bitwise_and` consiste em um processo de AND bit a bit entre os valores dos pixels do quadro e os valores correspondentes da máscara. O resultado é uma nova imagem em que os pixels que não fazem parte das regiões brancas da máscara são convertidos para preto, enquanto os pixels que pertencem a essas regiões permanecem intactos. A finalidade dessa máscara é realçar a região de interesse dentro do quadro, permitindo um foco preciso durante a etapa de segmentação da imagem.

### 4.5.1 Detecção da barra

Para a detecção da barra fixa, foi empregado o filtro de Canny 2.2.5 com o intuito de mitigar ruídos e enfatizar as bordas proeminentes. Para essa finalidade, foi adotada a função incorporada à biblioteca `openCV`, sendo parametrizada da seguinte forma: o limite inferior do processo de histerese foi estabelecido em 50, o limite superior definido em 100, e o tamanho do kernel aplicado ao operador Sobel foi estipulado como 3.

Após a etapa de pré-processamento da imagem, foi empregada a Transformada de Hough 2.2.4, também disponibilizada na biblioteca `openCV`. Essa aplicação foi configurada com os seguintes parâmetros: a resolução da distância no acumulador, em pixels, foi definida como 1; a resolução angular no acumulador, em radianos, foi estabelecida em  $\pi/180$ , equivalente a 1 grau; e o limiar do acumulador foi fixado em 150. Essas configurações asseguram a avaliação de cada pixel para sua inclusão ou não como parte de uma reta, além de garantir uma inclinação mínima de  $1^\circ$  de uma reta a outra e que sejam consideradas retas apenas aquelas que tiverem 150 votos ou mais contabilizados no acumulador.

O resultado da Transformada de Hough 2.2.4 foi filtrado, selecionando apenas retas horizontais com uma tolerância de  $10^\circ$ , ou seja, somente retas cujo valor absoluto do ângulo é menor que o valor da tolerância, ou ainda, cujo valor absoluto da diferença entre 180 e a tolerância é menor que o valor absoluto do ângulo.

Após a etapa inicial de filtragem, uma segunda filtragem foi aplicada para selecionar

Figura 18 – Ambiente de gravação e Mascara Preto e Branca criada apartir do mesmo



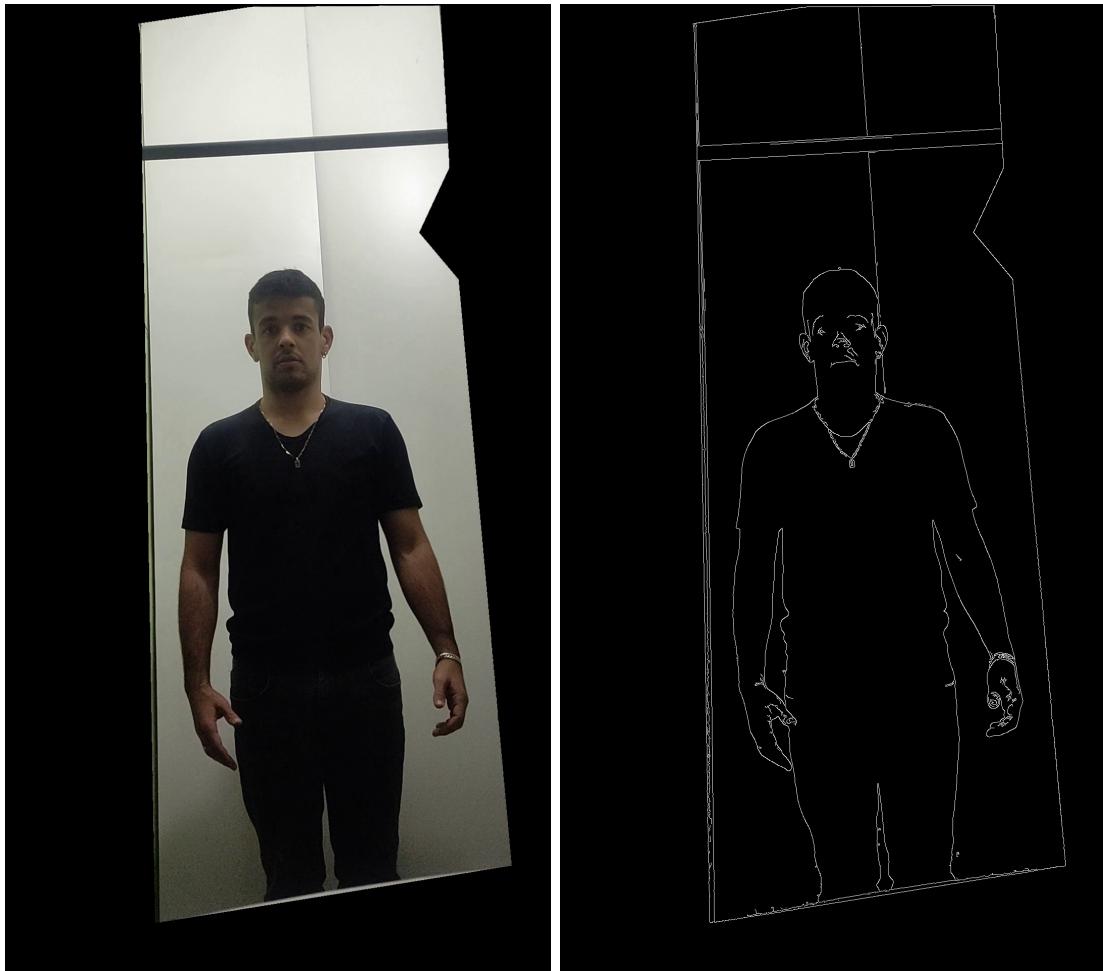
Fonte: Elaborado pelo autor (2023)

exclusivamente as retas localizadas em uma região específica do *frame*. Essa região é demarcada como a porção superior, correspondendo a 30% da área total. Essa abordagem foi adotada com base na expectativa de que a barra esteja consistentemente posicionada na parte superior do vídeo, uma vez que é crucial capturar o corpo estendido na barra e a parte dele que ultrapassa a barra.

Por meio dessa abordagem, a expectativa era identificar duas retas paralelas que representassem a seção superior e inferior da barra. Como desenvolvimento, as duas retas com maior número de votos foram selecionadas e procedeu-se à determinação do ponto médio das coordenadas  $y$  entre elas no sistema cartesiano  $(x, y)$ . Nesse contexto, a barra foi concebida como uma reta paralela às outras duas, passando pelo ponto médio das coordenadas  $y$  correspondentes a essas retas.

A cada quadro, o procedimento de detecção da barra é repetido. Para lidar com *frames* em que a barra não é detectada e minimizar a variação na posição da barra devido a flutuações na iluminação ou tremores no vídeo, os valores representativos da barra são acumulados. Em outras palavras, a cada nova detecção, os valores  $(x, y)$  que representam o início e o fim da barra são acumulados em quatro variáveis, aqui denominadas como  $(x_{\text{start}}, y_{\text{start}})$  e  $(x_{\text{end}}, y_{\text{end}})$ . O valor de  $x$  que representa a posição  $x$  correspondente ao

Figura 19 – Aplicação do filtro de Canny.



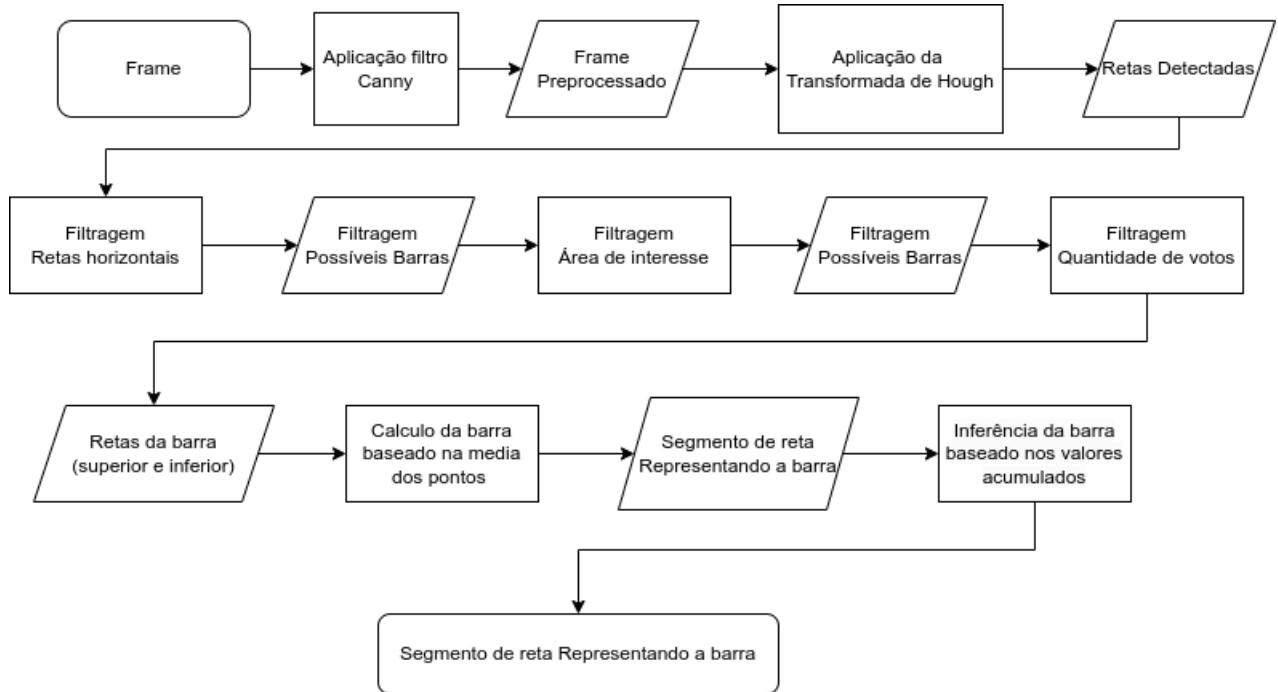
Fonte: Elaborado pelo autor (2023)

início da barra atual é definido como o valor acumulado na variável  $x_{start}$ , dividido pela quantidade de *frames* processados. O mesmo princípio é aplicado às outras coordenadas da barra. Esse método ajuda a suavizar a posição da barra ao longo do tempo, tornando-a menos suscetível a flutuações abruptas e permitindo uma estimativa mais estável da sua posição real.

É esperado que não haja muita variação na detecção da barra, mas mesmo assim, ao longo do tempo, o valor da barra tende a convergir a um ponto central. Quanto mais tempo passa, melhor fica a inferência da barra quando ela não consegue ser detectada.

O processo pode ser visualizado no diagrama a seguir:

Figura 20 – Diagrama do processo de detecção da barra

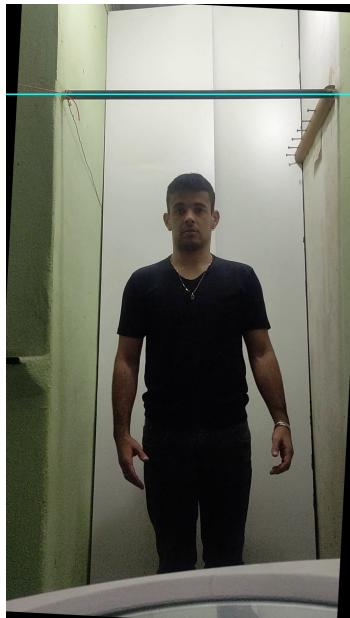


Fonte: Elaborado pelo autor (2023)

#### 4.5.2 Inclinação da barra

Durante o exame da barra física, é necessário que a barra esteja alinhada de forma paralela ao solo. Contudo, devido a um possível posicionamento inadequado da câmera, o vídeo pode demonstrar uma inclinação na barra. Para corrigir isso, o ângulo de inclinação da barra é calculado e, posteriormente, o frame é rotacionado de acordo com essa inclinação. Essa ação tem como objetivo assegurar que, na imagem, a barra se torne paralela à base da imagem, alinhando-a de forma adequada em relação ao plano horizontal.

Figura 21 – Imagem rotacionada em relação a barra



Fonte: Elaborado pelo autor (2023)

#### 4.5.3 Estimativa de pose humana

Foi adotado a biblioteca MediaPipe como uma ferramenta crucial para a predição dos pontos de referência da pose humana a partir de frame. Durante a configuração da ferramenta, foram considerados quatro de sete parâmetros para otimizar o desempenho do sistema: *static\_image\_mode*, *model\_complexity*, *min\_detection\_confidence* e *min\_tracking\_confidence*. Cada um desses parâmetros desempenha um papel fundamental na precisão e eficiência da EPH, contribuindo para a robustez e a qualidade dos resultados obtidos em nosso estudo.

O parâmetro *static\_image\_mode* pode ser configurado como *False* ou *True*. Esse parâmetro determina se as imagens de entrada serão tratadas como um fluxo contínuo de vídeo ou como imagens independentes. Quando *static\_image\_mode* é definido como *False*, o sistema realiza uma detecção inicial para identificar a pessoa mais proeminente, após essa detecção, ocorre o *tracking* dos pontos de referência, evitando a necessidade de iniciar uma nova detecção a cada quadro subsequente, a menos que o ponto de referência seja perdido. Por outro lado, quando *static\_image\_mode* é configurado como *True*, o sistema processa um lote de imagens como entidades independentes e não relacionadas, a cada nova detecção, os valores anteriores são descartados, e todo o processo é reiniciado. No contexto do nosso projeto, optamos por configurar o *static\_image\_mode* como *False*, aproveitando a eficiência proporcionada pelo rastreamento contínuo de pontos de referência, o que contribui para melhorar a computação geral e reduzir a latência do processo. Essa escolha foi feita com base nas necessidades específicas de nossa aplicação.

O parâmetro *model\_complexity* pode ser configurado com os valores predefinidos 0, 1 e 2 e está ligado a precisão do ponto de referência e quanto maior o numero, maior a complexidade influênciando diretamente com a latência de inferência. Foi utilizado o valor 2 pois os demais valores estavam gerando inferências errôneas em frames específicos de modo que influênciava diretamente na extração de características dos frames.

O parâmetro *min\_detection\_confidence* pode ser configurado como um valor entre 0.0 a 1.0 e define o valor de confiança mínimo para que o modelo de detecção de pessoa seja considerado como bem sucedido. O valor adotado foi 0.3

O parâmetro *min\_tracking\_confidence* pode ser configurado como um valor entre 0.0 a 1.0 e define o valor de confiança mínimo para que o modelo de rastreamento de pontos sejam considerados rastreados com sucesso. O valor adotado foi 0.75, configurá-lo com um valor mais alto pode aumentar a robustez da solução, às custas de uma latência mais alta.

A biblioteca midia MediaPipe fornece 33 pontos de referência como mostrado na Figura 16 porem foi utilizado apenas 16 pontos sendo eles:

11 - ombro esquerdo

12 - ombro direito

13 - cotovelo esquerdo

14 - cotovelo direito

15 - pulso esquerdo

16 - pulso direito

17 - mindinho esquerdo

18 - mindinho direito

19 - indicador esquerdo

20 - indicador direito

23 - quadril esquerdo

24 - quadril direito

25 - joelho esquerdo

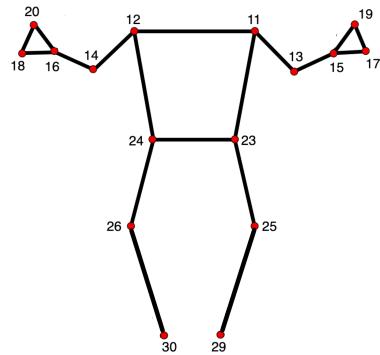
26 - joelho direito

29 - calcanhar esquerdo

30 - calcanhar direito

Podendo ser representado na figura a seguir:

Figura 22 – Pontos de referência usados na aplicação



Fonte: Elaborado pelo autor (2023) baseado na Figura 16

#### 4.5.4 Mão na barra

Para realizar a detecção das mãos na barra, foi seguida uma sequência de etapas bem definidas. Inicialmente, foi aplicada uma máscara que identificou os pixels na imagem que estavam dentro do intervalo de cor (0, 25, 0) a (20, 255, 255), enquanto os outros pixels foram definidos como pretos, destacando assim a cor da pele do executor.

Figura 23 – Filtragem pela cor de pele



Fonte: Elaborado pelo autor (2023)

Posteriormente, a imagem que estava em [RGB](#) foi unificada em um unico canal por meio da função [cvtColor](#) presente na biblioteca [openCV](#), tendo como parametro a constante [COLOR\\_BGR2GRAY](#) da propria biblioteca, convertendo os três canais (Blue, Gren e Red) para uma escala de cinza.

Figura 24 – Conversão em escala de cinza



Fonte: Elaborado pelo autor (2023)

A partir desse ponto, a imagem passa a conter diferentes níveis de brilho, representados em uma escala de cinza. Foi aplicado um filtro de limiarização da biblioteca [openCV](#) utilizando a função [threshold](#) tendo como parametro a constante [THRESH\\_BINARY](#) e o valor de limiar 50. Nesse processo, todos os pixels com valores iguais ou superiores a 50 foram convertidos para o valor 255, enquanto os pixels com valores menores foram convertidos para o valor 0. Isso resultou em uma imagem binarizada que tem o propósito de representar a presença ou ausência do executor.

Figura 25 – Binarização da Imagem



Fonte: Elaborado pelo autor (2023)

Com a imagem binarizada, foi aplicado um filtro de suavização usando um kernel quadrado de lado igual à metade do tamanho da barra ( $tamanhodabarreira/2, tamanhodabarreira/2$ ). O objetivo dessa etapa é reduzir o ruído na imagem. No entanto, como o filtro de suavização é uma operação de convolução, foi necessário aplicar a limiarização novamente para obter uma nova imagem binarizada.

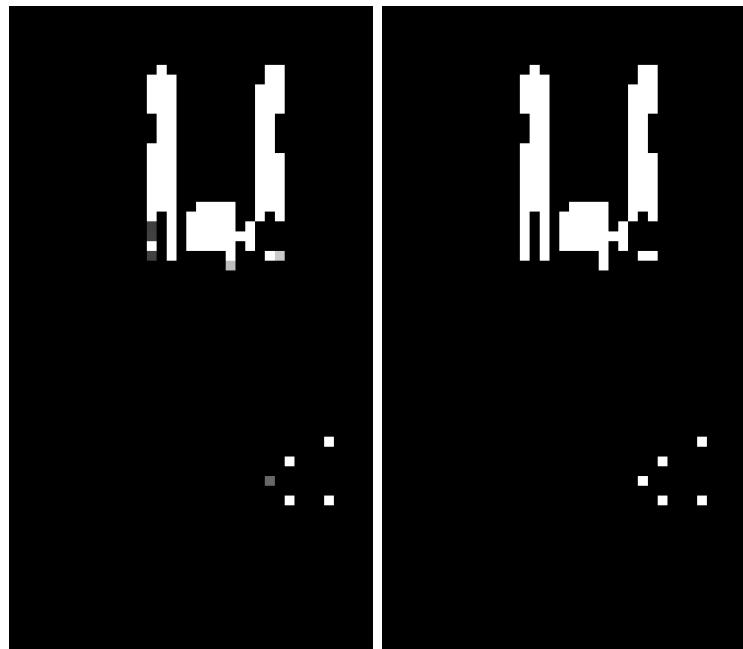
Figura 26 – Redução de ruidos com a suavização e binarização



Fonte: Elaborado pelo autor (2023)

Neste ponto, a imagem que passou por uma segunda binarização para melhorar a precisão e eliminar ruídos de descontinuidade, foi aplicado um filtro de pixelização com um kernel quadrado de lado igual ao tamanho da barra (*tamanhodabarreira*, *tamanhodabarreira*). Esse filtro foi utilizado para destacar claramente a forma do corpo do executor, representando a presença do corpo por blocos de tamanho fixo. Após a pixelização, a limiarização foi aplicada novamente, resultando em uma nova imagem binarizada.

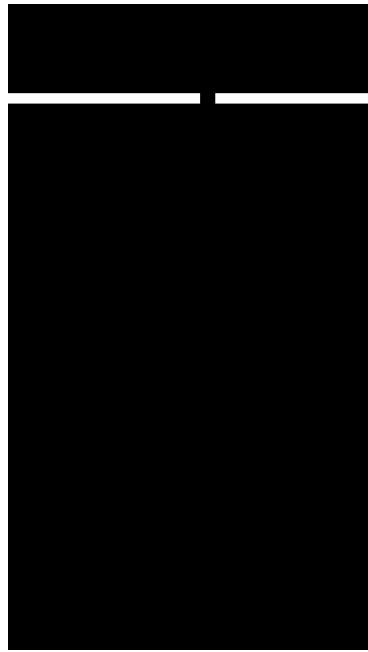
Figura 27 – Imagem Pixelizada e binarizada



Fonte: Elaborado pelo autor (2023)

A área de interesse é representado por uma linha no rumo da barra mesclada com uma mascara que dividida a barra em dois polos, esquerdo e direito, a divisão desses polos foi feita por meio da identificação do ponto médio entre os ombros esquerdo e direito em relação ao eixo  $x$ , com base nesse ponto, foi traçada uma linha vertical paralela a barra de modo que a imagem fosse dividida em dois polos forçando uma descontinuidade da região de interesse.

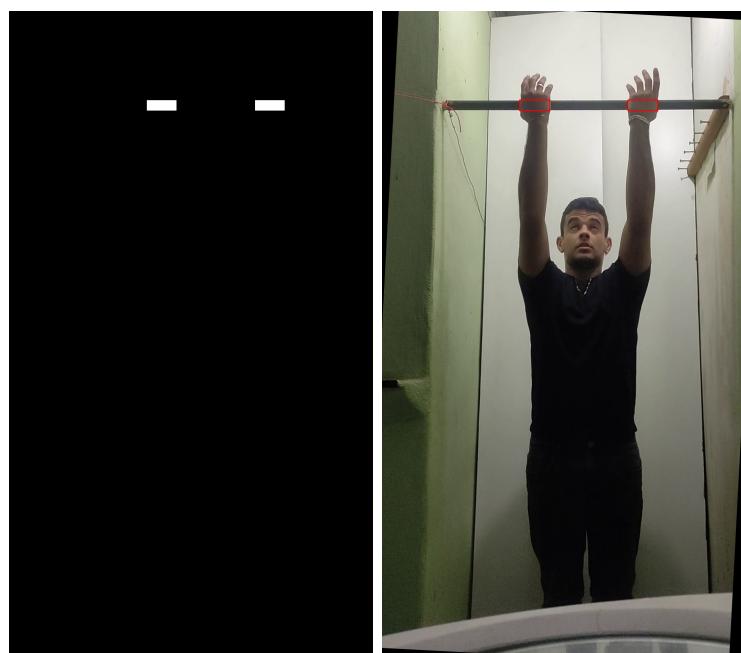
Figura 28 – Mascara da região de interesse



Fonte: Elaborado pelo autor (2023)

Por fim com a imagem filtrada e binarizada foi destacada a região de interesse e identificado os contornos presente nessa região, cada contorno representa algo obstruindo a barra em específico uma parte do corpo ou algo que tenha o mesmo intervalo de cor pre estabelecido. Portanto com a identificação de 1 ou mais contornos em cada lado da imagem possivelmente são as mãos tocando a barra.

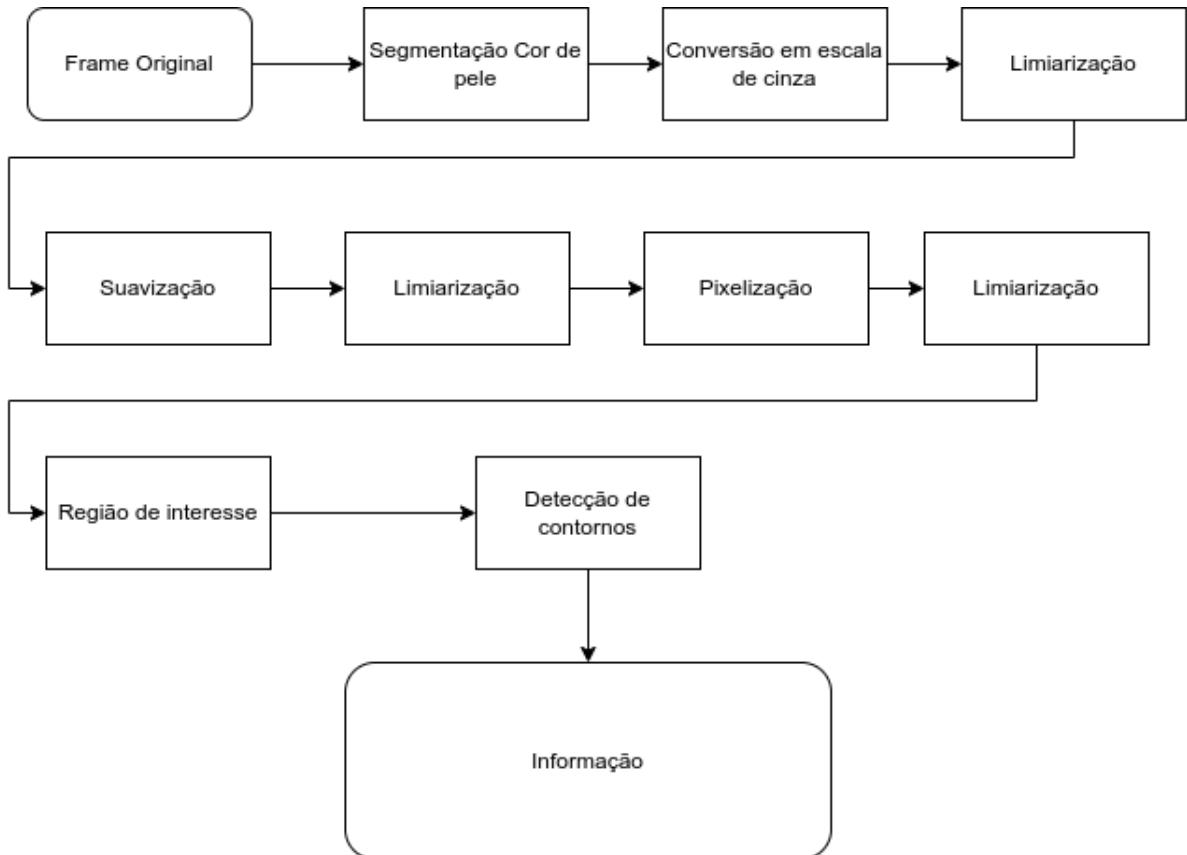
Figura 29 – Do lado esquerdo a imagem segmentada e do lado direto sua representação



Fonte: Elaborado pelo autor (2023)

O processo pode ser visualizado no diagrama a seguir:

Figura 30 – Diagrama do processo de detecção de mão na barra



Fonte: Elaborado pelo autor (2023)

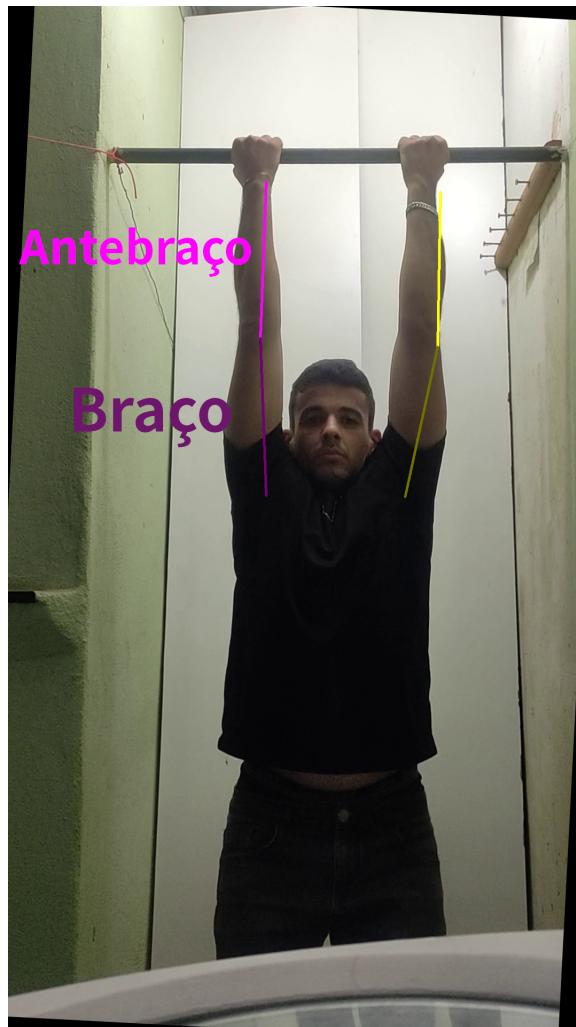
#### 4.5.5 Braço esticado

A verificação da extensão do cotovelo foi realizada estritamente quando o executor estava com as mãos na barra de maneira continua quadro a quadro, verificando duas condições denominadas aqui de *check\_1* e *check\_2*, a extensão total do braço só foi considerada quando ambas as condições foram atendidas. Esse processo de verificação ajudou a garantir que somente certas condições fossem identificadas como verdadeiras, reduzindo ao máximo a ocorrência de falsos positivos e melhorando a confiabilidade da verificação da extensão do cotovelo, tornando-a mais robusta em relação a possíveis fontes de erro ou variação nos dados de entrada.

*check\_1* foi estabelecido com base na comparação dos ângulos formados entre o antebraço e o braço. Nessa avaliação, o antebraço era representado pelo segmento de reta definido pelos pontos do punho e cotovelo, enquanto o braço era representado pelo segmento de reta definido pelos pontos do ombro e cotovelo. Para que a condição *check\_1* fosse considerada verdadeira, era necessário que tanto o ângulo formado pelo antebraço e

braço esquerdo quanto o ângulo formado pelo antebraço e braço direito fossem menores que o limite previamente estabelecido em 13º (valor do limiar).

Figura 31 – Segmento de reta representante dos membros superiores



Fonte: Elaborado pelo autor (2023)

*check\_2* foi estabelecido com base na medição da distância entre o peito do executor e a barra. Devido a rotação previa do frame em relação a barra e a garantia de que a barra é paralela ao solo a medida utilizou exclusivamente da coordenada *y*.

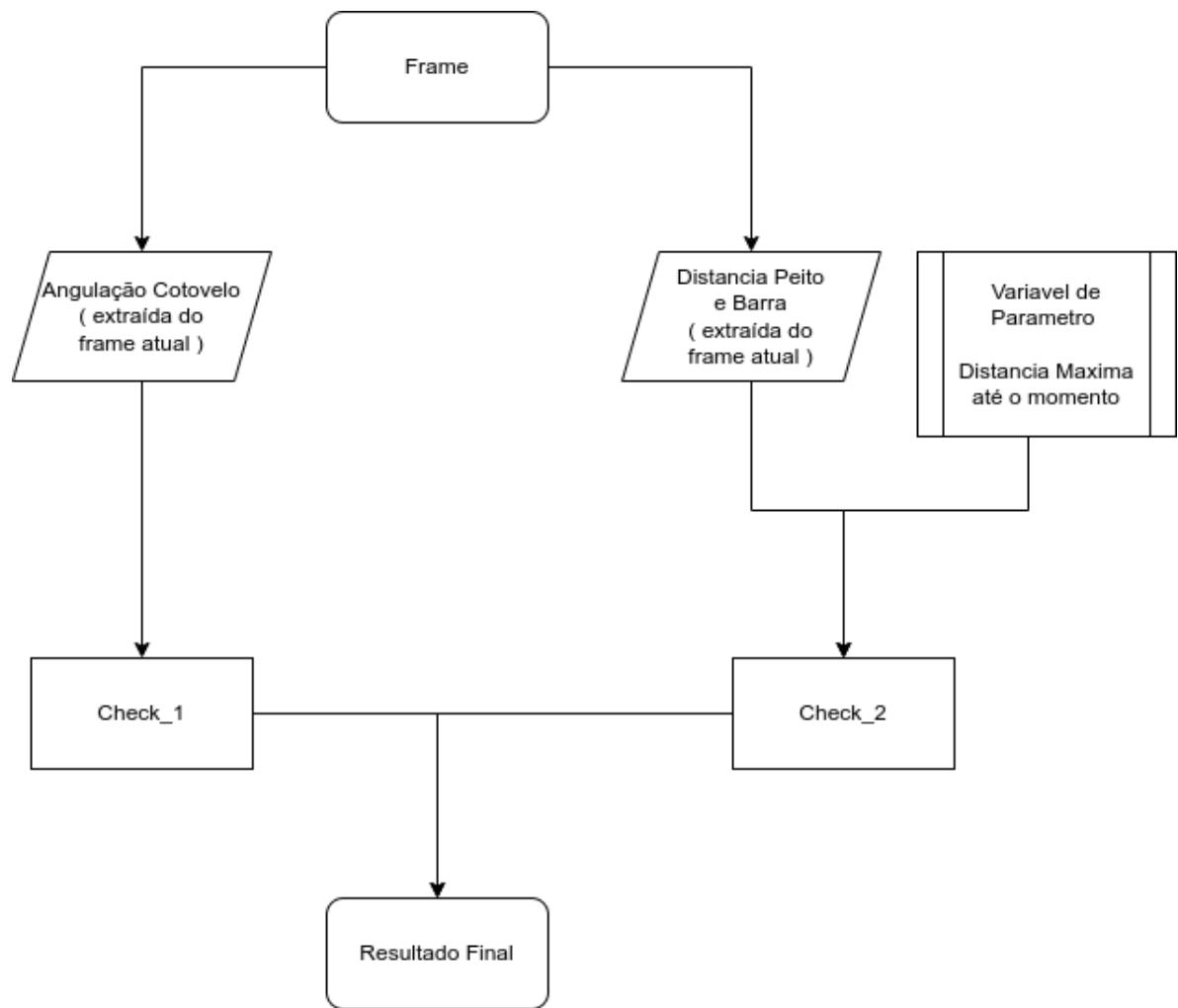
Foi feito o uso de uma variável como parâmetro, sendo atualizada frame a frame ao longo do vídeo, variavel essa que registra a posição do peito no momento em que a maior distância entre o peito e a barra é identificada.

Em termos práticos, *check\_2* avaliava se a diferença absoluta entre o ombro esquerdo e o valor armazenado do ombro esquerdo quando idendificada a maior distância entre peito e a barra era menor que o limite predefinido, 100 pixels, o mesmo critério era aplicado tanto ao ombro direito quanto ao ombro esquerdo.

Quando essa condição era atendida em um dos lados, a verificação era considerada verdadeira. A verificação unilateral ser suficiente se justificava devido à imprecisão na determinação exata da posição do ombro, a qual poderia estar dentro do intervalo aceitável em um lado e fora do intervalo no outro por um valor infimo de 1 pix.

Ao exigir que ambas as mãos estivessem tocando a barra, o alinhamento do peito tendia a ocorrer de forma paralela à barra. Isso implicava que, ao considerar um lado como verdadeiro, o outro também é automaticamente considerado verdadeiro. salvo em casos onde um cotovelo esta extendido e outro esta flexionado, porem nestes exemplos são cobertos pelo *check\_1* onde é verificado a angulação dos dois cotovelos, tanto esquerdo quanto direito.

Figura 32 – Diagrama do fluxo da informação para verificar extensão do cotovelo

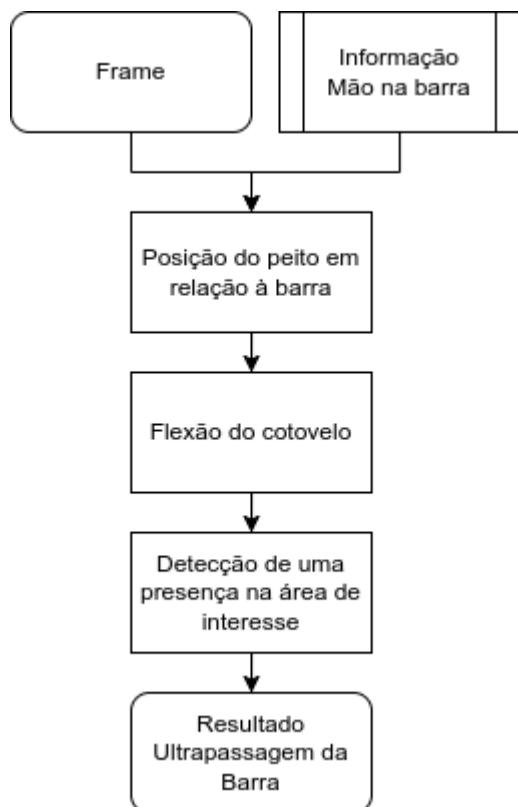


Fonte: Elaborado pelo autor (2023)

#### 4.5.6 Ultrapassagem do queixo a barra

Para verificar se ocorreu a ultrapassagem do queixo à barra, foi avaliando três condições essenciais: a posição do peito em relação à barra, a flexão do cotovelo e a detecção de uma presença na área de interesse, presumivelmente correspondente à cabeça do executor. As condições são feitas na ordem apresentada e para evitar o processamento desnecessário as condições são analisadas apenas se a mão estiver encostando na barra e a cada condição Falsa o processamento é interrompido.

Figura 33 – Diagrama Ultrapassagem do queixo a barra



Fonte: Elaborado pelo autor (2023)

A verificação da proximidade do peito em relação à barra é determinada pela medida da distância da coordenada entre os ombros esquerdo ou direito até a barra, com foco na coordenada  $y$ . Considera-se que o peito está em contato com a barra quando a distância entre o peito e a barra é inferior a 150% do comprimento da própria barra.

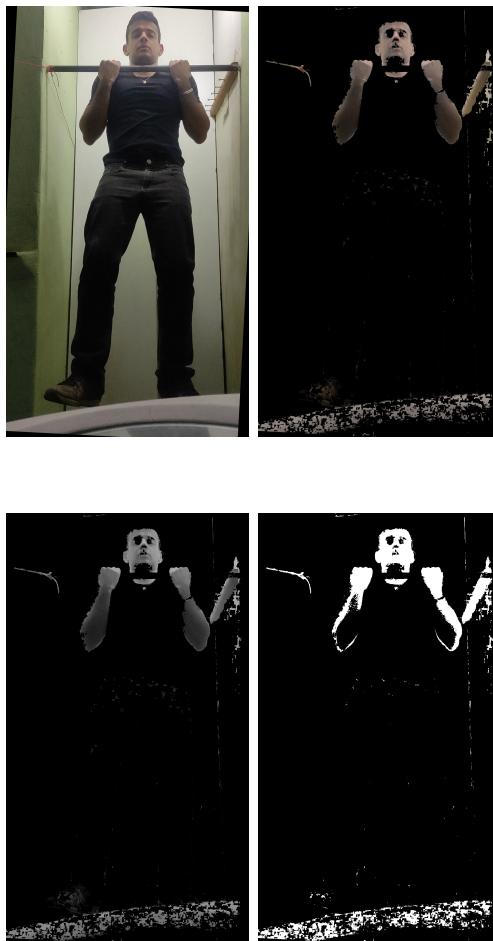
Portanto, a ultrapassagem bem-sucedida do queixo à barra é confirmada apenas quando todas as três condições, a proximidade do peito à barra, a flexão do cotovelo e a detecção de elementos na região de interesse, são avaliadas como verdadeiras.

O processamento para verificar a flexão de cotovelo segue uma abordagem semelhante ao processo de verificação de extensão de cotovelo utilizada na detecção de mão na barra. A diferença é que nesse caso, examinamos se o ângulo formado entre os segmentos

que representam o antebraço e o braço esquerdo ou direito é superior a 175, incorporando uma margem de imprecisão de 5 graus.

Por fim, o processo de detecção de presença na área de interesse segue uma abordagem semelhante àquela utilizada na detecção da mão na barra. Inicialmente, a imagem passa por um processo de realce da cor da pele. Em seguida, é convertida para escala de cinza e, posteriormente, submetida à limiarização, resultando em uma imagem binarizada na qual o executor se destaca em relação ao plano de fundo.

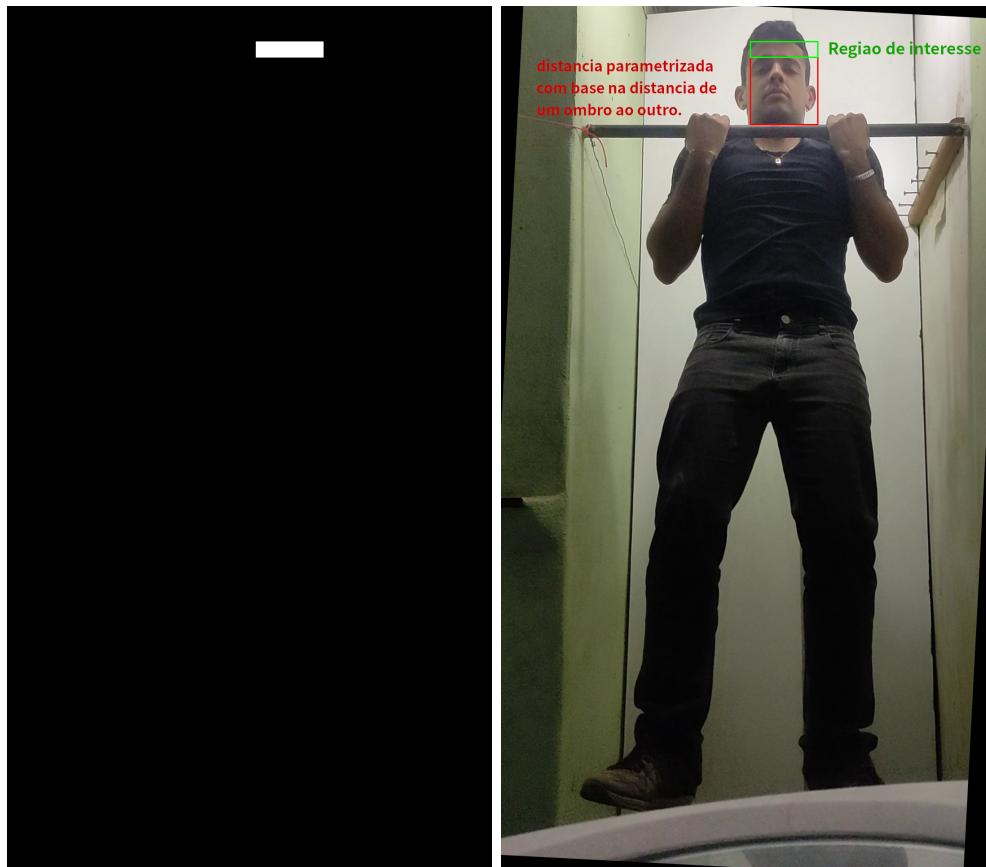
Figura 34 – Preprocessamento até a binarização da imagem



Fonte: Elaborado pelo autor (2023)

A partir deste ponto, o processo se destaca, pois é aplicada uma máscara para realçar a região de interesse. As dimensões da máscara é baseado em 1 terço da diferença da cordenada  $x$  do ombro esquerdo e o ombro direito, com o valor arredondado para baixo, nomeado aqui como  $size\_block$ . A máscara é definida como um retângulo com a base correspondentes a  $1,5 * (size\_block)$  e altura igual a  $(size\_block)/2$ , posicionado a uma distância de  $1,5 * (size\_block)$  da barra, com seu centro alinhado à coordenada  $x$  do ponto médio entre o ombro esquerdo e o ombro direito.

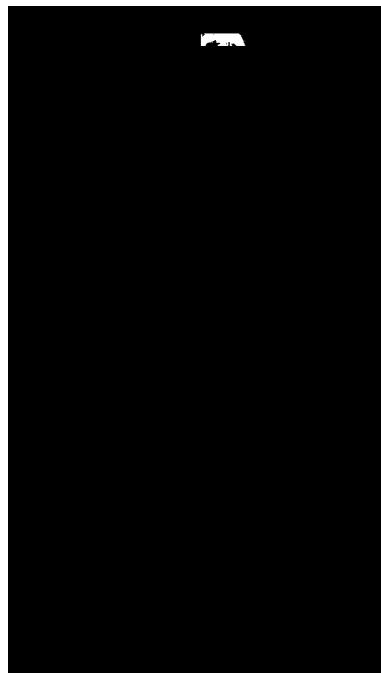
Figura 35 – Representação da mascara para região de interesse



Fonte: Elaborado pelo autor (2023)

Por fim a detecção de presença na área de interesse é considerada positiva se houver presença de pixels brancos na imagem resultante.

Figura 36 – Captura da região de interesse



Fonte: Elaborado pelo autor (2023)

#### 4.5.7 Movimentação do quadril

O movimento do quadril foi examinado de maneira semelhante à flexão e extensão do cotovelo, no entanto, os pontos de referência utilizados foram o quadril, o joelho e o tornozelo. A coxa foi considerada como o segmento de linha que se estende do quadril ao joelho, enquanto a perna foi definida como o segmento de linha que se estende do joelho até o tornozelo. Para avaliar a movimentação dos membros inferiores, foi analisado se o ângulo entre os membros ou direito ou esquerdo excedia 15 graus.

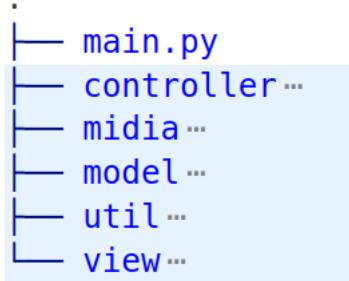
### 4.6 Código fonte

O código-fonte podem ser encontrado em um repositório específico hospedado no GitHub. Para acessar e explorar o conjunto completo de arquivos, você pode visitar o seguinte sitio eletronico: <<https://github.com/le314u/TCC/tree/main/Tecnologico/Source/Programa/python>>. Neste repositório, os arquivos estão organizados de forma a facilitar a navegação e a análise individual de cada componente do programa. Este repositório oferece uma visão abrangente do presente trabalho e é uma valiosa fonte de informações para qualquer pessoa interessada em entender melhor a implementação da ferramenta e queira ter este trabalho como base para trabalhos futuros.

#### 4.6.1 Hierarquia de diretório

O projeto foi estruturado de forma que os componentes do código conseguissem ser divididos em 5 grandes blocos, sendo eles controller, model, view, util e midia.

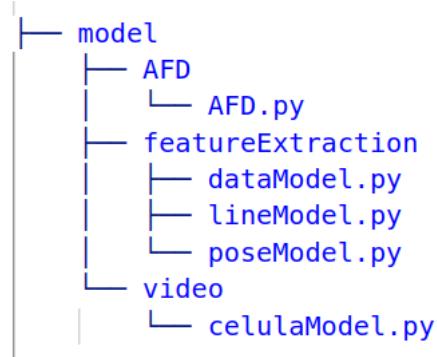
Figura 37 – Estrutura do diretório baseado em 5 grande blocos



Fonte: Elaborado pelo autor (2023)

No contexto do meu projeto, a estrutura de diretórios é organizada da seguinte forma o diretório “model” contém arquivos que desempenham o papel fundamental na representação dos dados e a lógica de negócios da ferramenta.

Figura 38 – Estrutura do diretório “/model”



Fonte: Elaborado pelo autor (2023)

A pasta “view” é responsável por apresentar esses dados de maneira atraente e interativa aos usuários, assegurando que a interface do usuário reflita com precisão o estado atual do Modelo.

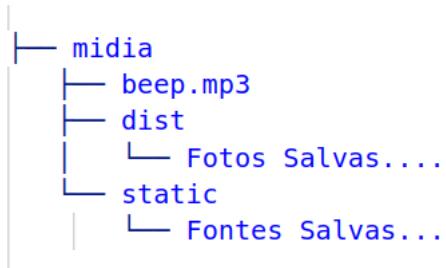
Figura 39 – Estrutura do diretório “/view”



Fonte: Elaborado pelo autor (2023)

O diretório “midia” abriga arquivos relacionados a imagens, áudio, vídeo e a fontes do sistema.

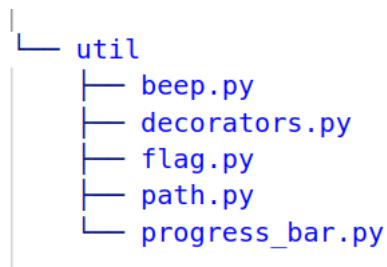
Figura 40 – Estrutura do diretório “/midia”



Fonte: Elaborado pelo autor (2023)

Por fim, o diretório “util” contém arquivos genéricos que não se encaixam especificamente em nenhum desses diretórios e podem ser utilizados em diferentes partes do programa conforme necessário.

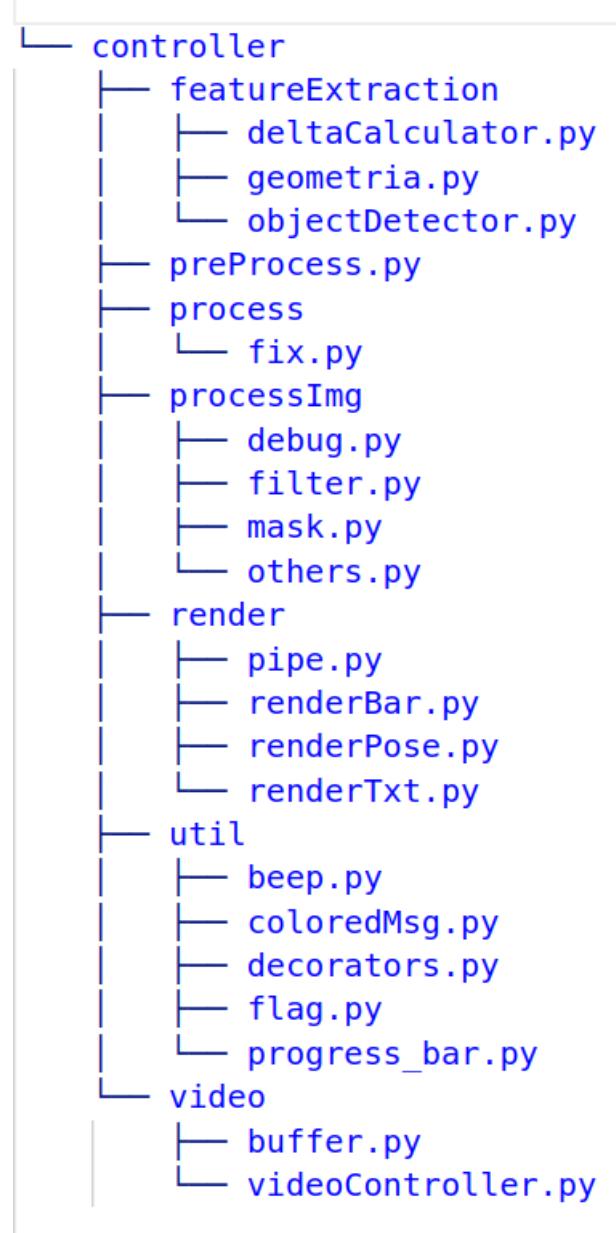
Figura 41 – Estrutura do diretório “/util”



Fonte: Elaborado pelo autor (2023)

O “controller” atua como um mediador entre arquivos da view e do model, interpretando as ações do usuário na interface e coordenando as operações necessárias do model para atender às solicitações, mantendo, assim, uma separação eficaz entre a lógica de apresentação e a lógica de negócios.

Figura 42 – Estrutura do diretório “/controller”



Fonte: Elaborado pelo autor (2023)

## 4.7 Código fonte

O código-fonte podem ser encontrado em um repositório específico hospedado no GitHub. Para acessar e explorar o conjunto completo de arquivos, você pode visitar

o seguinte sitio eletronico: <<https://github.com/le314u/TCC/tree/main/Tecnologico/Source/Programa/python>>. Neste repositório, os arquivos estão organizados de forma a facilitar a navegação e a análise individual de cada componente do programa. Este repositório oferece uma visão abrangente do presente trabalho e é uma valiosa fonte de informações para qualquer pessoa interessada em entender melhor a implementação da ferramenta e queira ter este trabalho como base para trabalhos futuros.

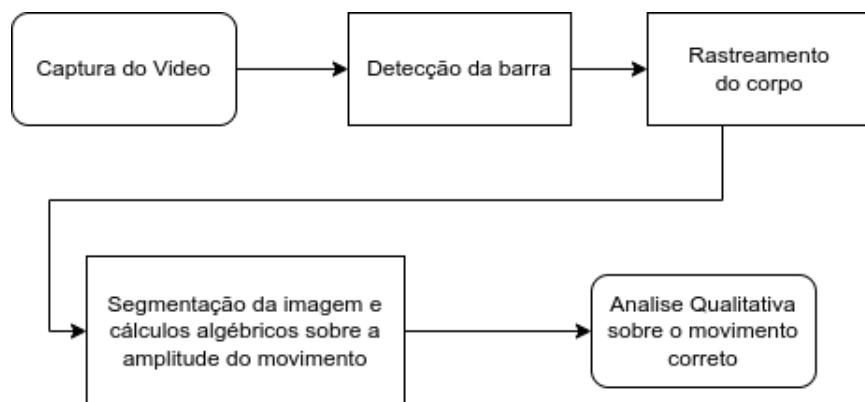
# 5 Resultados

Neste capítulo é apresenta a ferramenta obtida assim como detalhes sobre o tempo de execução.

## 5.1 Ferramenta

A ferramenta desenvolvida atua a partir de uma gravação no plano coronal da execução do movimento de barra fixa processando a imagem de modo a detectar a barra e os pontos de referência do corpo humano e logo em seguida ocorre a segmentação da imagem e cálculos algébricos retornando o estado atual do movimento de barra fixa e dados sobre o movimento como forma de feedback sobre o movimento. Todo este processo segue o fluxograma abaixo.

Figura 43 – Fluxograma do processo executada pela ferramenta.

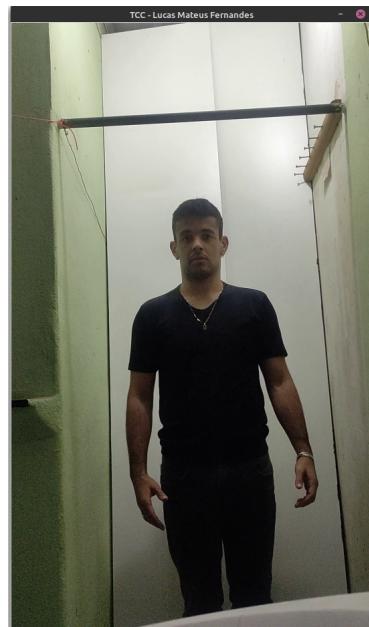


Fonte: Elaborado pelo autor (2023)

## 5.2 Parte Gráfica

A parte gráfica da ferramenta é baseada em duas telas, a tela de exibição que se assemelha a um player e é responsável exclusivamente para exibir o vídeo e as informações referentes ao feedback e o painel de controle que é destacado do player porém sua única função é controlar o que será exibido no player.

Figura 44 – Tela “player”



Fonte: Elaborado pelo autor (2023)

A tela responsável por ser um painel de controle do player possui nove campos de entrada sendo 2 caixas de texto “Speed” e “Frame” e cinco botões “PLAY”, “SaveF”, “SaveV”, “Barra”, “Dados” e “EPH” e uma barra deslizante.

Figura 45 – Tela “painel de controle”

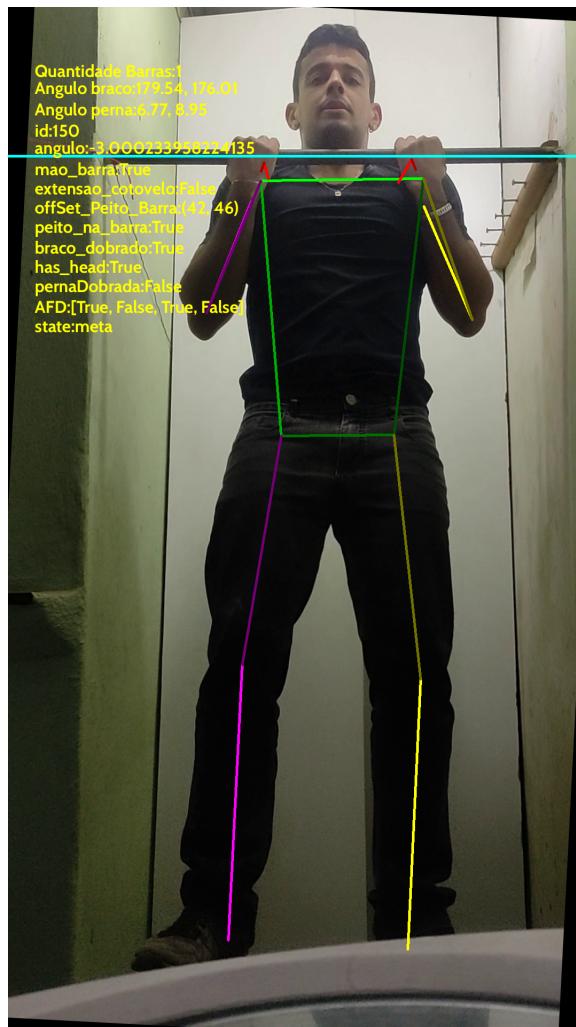


Fonte: Elaborado pelo autor (2023)

1. A caixa de texto “Speed” recebe um número decimal como entrada, representando um escalar pelo qual a velocidade de reprodução será alterada.
2. A caixa de texto “Frame” recebe um número inteiro como entrada, movendo o vídeo para o *frame* semelhante à barra deslizante que altera o *frame* em exibição.
3. O botão “PLAY” é responsável por reproduzir e pausar o vídeo.
4. O botão “SaveF” salva o *frame* visualizado no diretório “./midia/dist” do projeto.
5. O botão “SaveV” realiza a mesma ação do “SaveF”, mas para um vídeo em vez de uma imagem.

6. O botão “Barra” destaca a posição da barra no vídeo.
7. O botão “EPH” destaca a pose do executor traçando segmentos de reta sobre os membros e o tronco.
8. O botão “Dados” exibe informações no canto superior esquerdo, como o ID do *frame*, o estado do AFD, a quantidade de barras realizadas, o ângulo entre o braço e antebraço, o ângulo formado pela perna e coxa, o ângulo pelo qual o vídeo foi rotacionado para a barra ficar paralela ao solo, condições como se a mão está tocando a barra, se o cotovelo está estendido ou flexionado, se as pernas estão dobradas, se a cabeça ultrapassou a barra, a distância do peito até a barra, se o peito tocou a barra e o caractere do AFD que representa o *frame* atual.

Figura 46 – Resultado final com todas as Flags ativas “Barra”,“Dados” e “EPH”



Fonte: Elaborado pelo autor (2023)

### 5.3 Tempo de execução

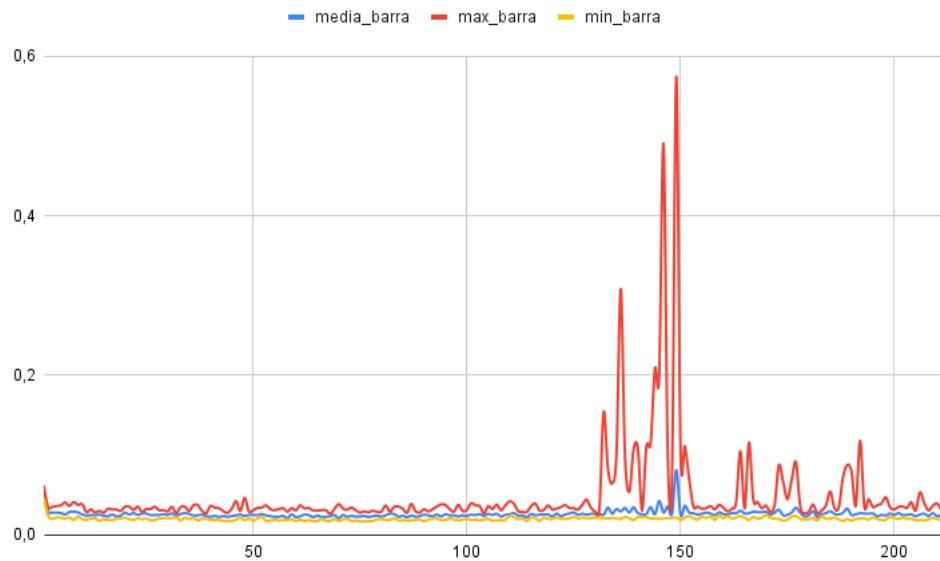
O tempo de execução desempenha um papel crucial pois representa o intervalo de tempo necessário para que um programa ou algoritmo conclua uma tarefa específica e é um fator fundamental na avaliação e otimização de sistemas computacionais utilizado como métrica para comparar a eficiência de diferentes abordagens e soluções, buscando identificar a estratégia mais eficaz para realizar uma tarefa determinada. Portanto para os testes foi avaliado um vídeo de 211 frames a uma taxa de 24 frames por segundos realizando um execução correta do movimento de barra fixa, para a amostragem foram realizadas 10 iterações da execução do programa.

Para um melhor entendimento dos resultados o processamento de um frame foi denominado de “process\_cell” e é subdividido em funções sendo elas:

- Função barra responsável pela Detecção da barra, conforme demonstrado na seção [4.5.1](#).
- Função verify\_inclination responsável pela rotação da imagem de acordo com a inclinação da barra, conforme demonstrado na seção [4.5.2](#).
- Função verify\_eph responsável pela identificação dos pontos chaves do corpo humano, conforme demonstrado na seção [4.5.3](#).
- Função verify\_angle\_member responsável pela identificação dos angulos entre os segmentos, conforme demonstrado na seção [4.5.5](#) e [4.5.7](#).
- Função verify\_char\_AFD responsável pela extração das informações presente no caracter do **AFD** conforme explicado em [4.4](#) Sendo um compilado de 4 funções
  - Função verify\_maoBarra responsável pela verificação do contato da mão com a barra demonstrado na seção [4.5.4](#)
  - Função verify\_extensaoCotovelo responsável pela verificação da extenção do braço demonstrado na seção [4.5.5](#)
  - Função verify\_ultrapassarBarra responsável pela verificação da ultrapassagem do queixo a barra demonstrado na seção [4.5.6](#)
  - Função verify\_movimentoQuadrilPerna responsável pela verificação da flexão dos membros inferiores demonstrado na seção [4.5.7](#)
- Função verify\_AFD responsável pela computação do **AFD** demonstrado na seção [17](#).

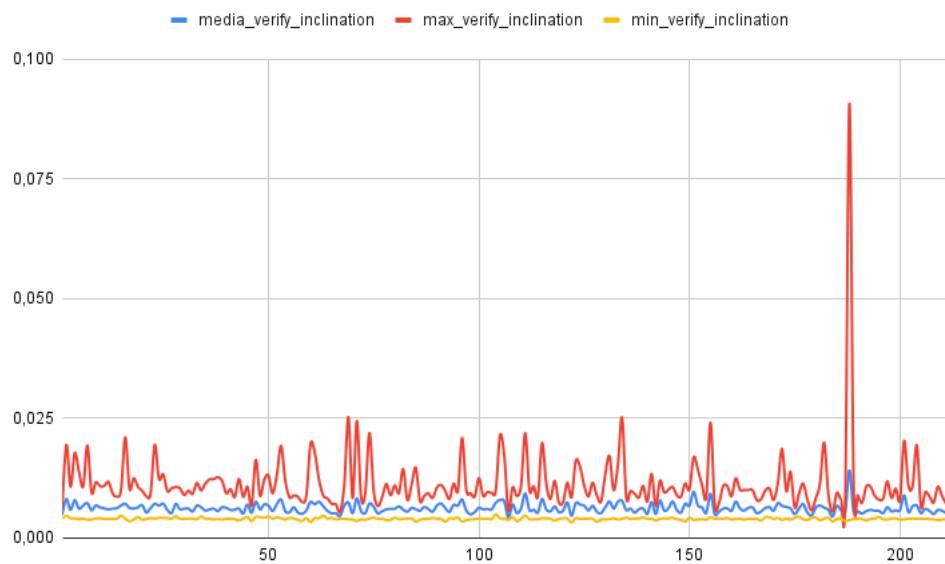
Os gráficos abaixo mostram no eixo vertical o tempo gasto em segundos para o processamento da função, associado aos identificadores dos frames representado no eixo horizontal, com base dados proveniente de 10 iterações.

Figura 47 – Tempo de execução da função “barra” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos



Fonte: Elaborado pelo autor (2023)

Figura 48 – Tempo de execução da função “verify\_inclination” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos



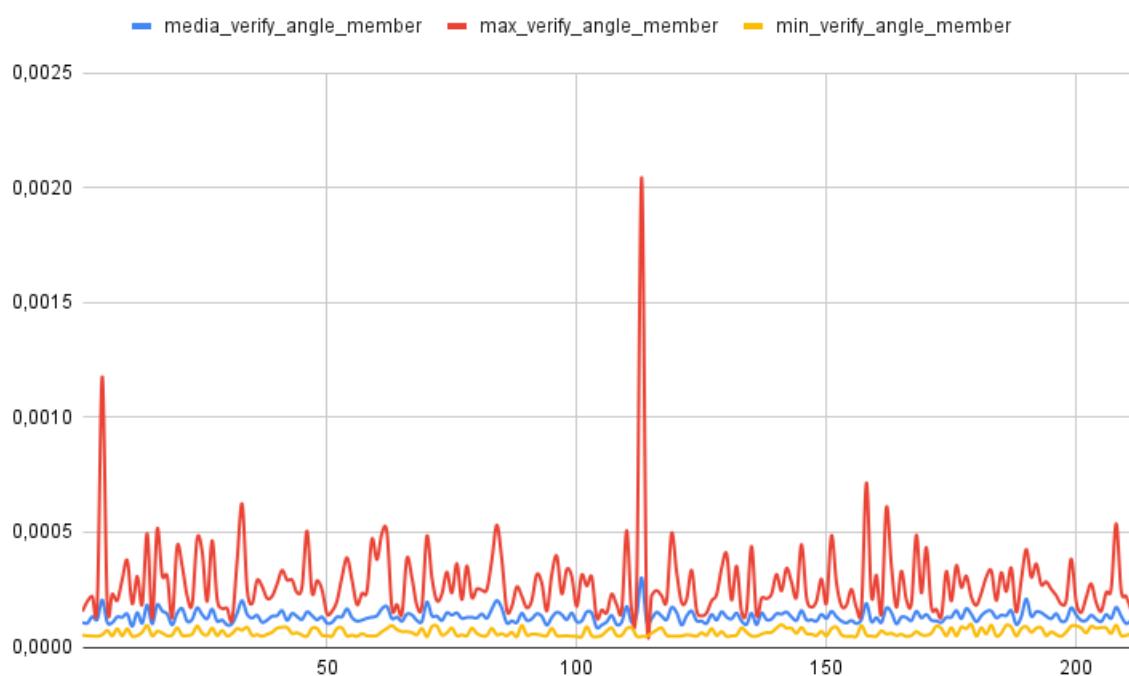
Fonte: Elaborado pelo autor (2023)

Figura 49 – Tempo de execução da função “verify\_eph” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos



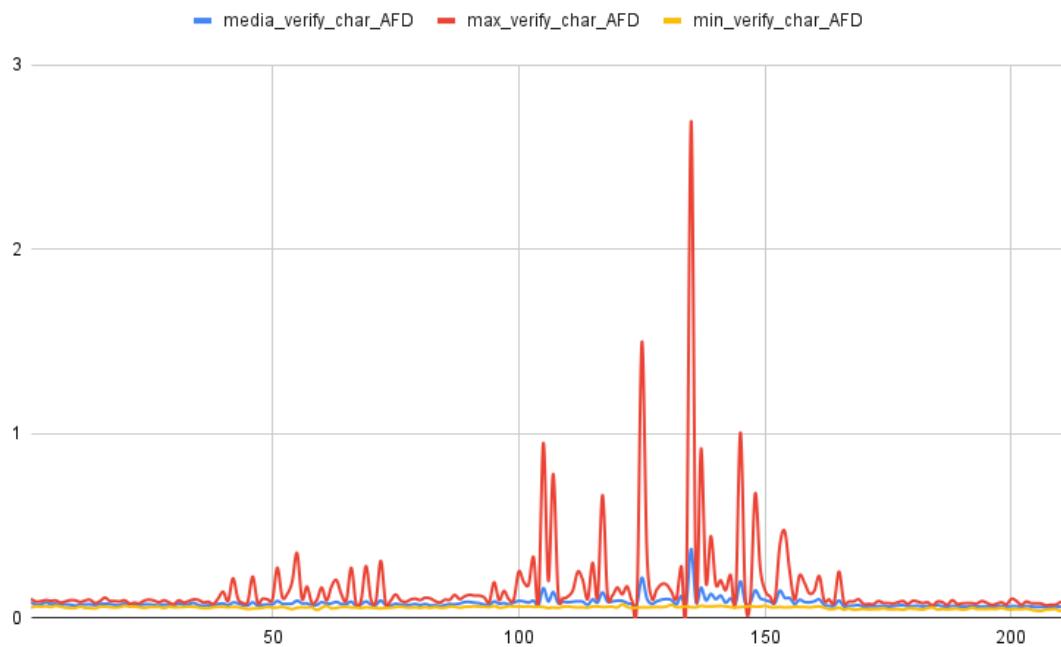
Fonte: Elaborado pelo autor (2023)

Figura 50 – Tempo de execução da função “verify\_angle\_member” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos



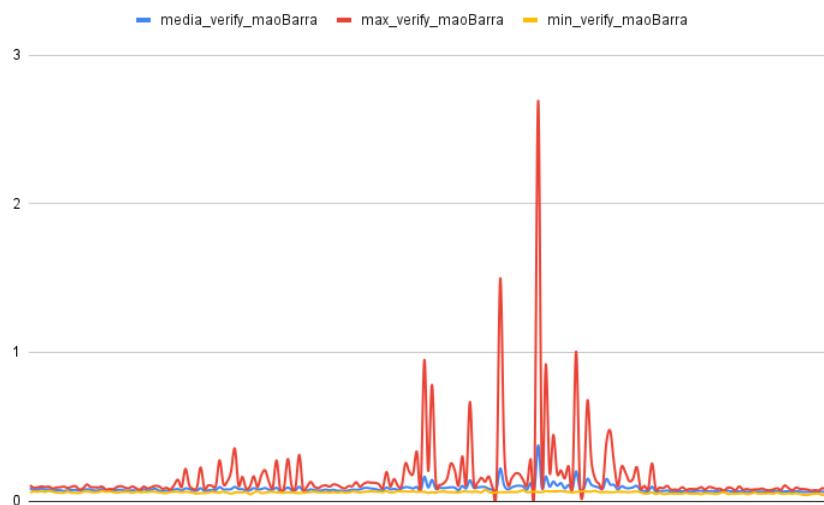
Fonte: Elaborado pelo autor (2023)

Figura 51 – Tempo de execução da função “verify\_char\_AFD” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos



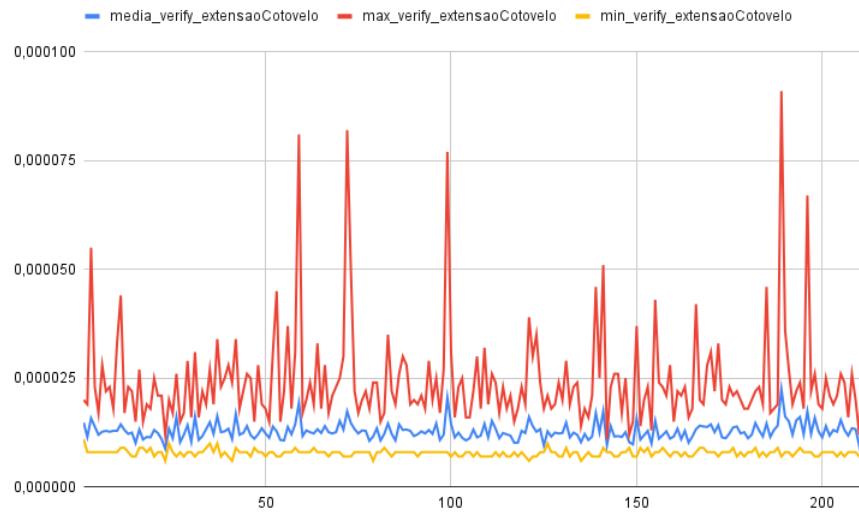
Fonte: Elaborado pelo autor (2023)

Figura 52 – Tempo de execução da função “verify\_maoBarra” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos



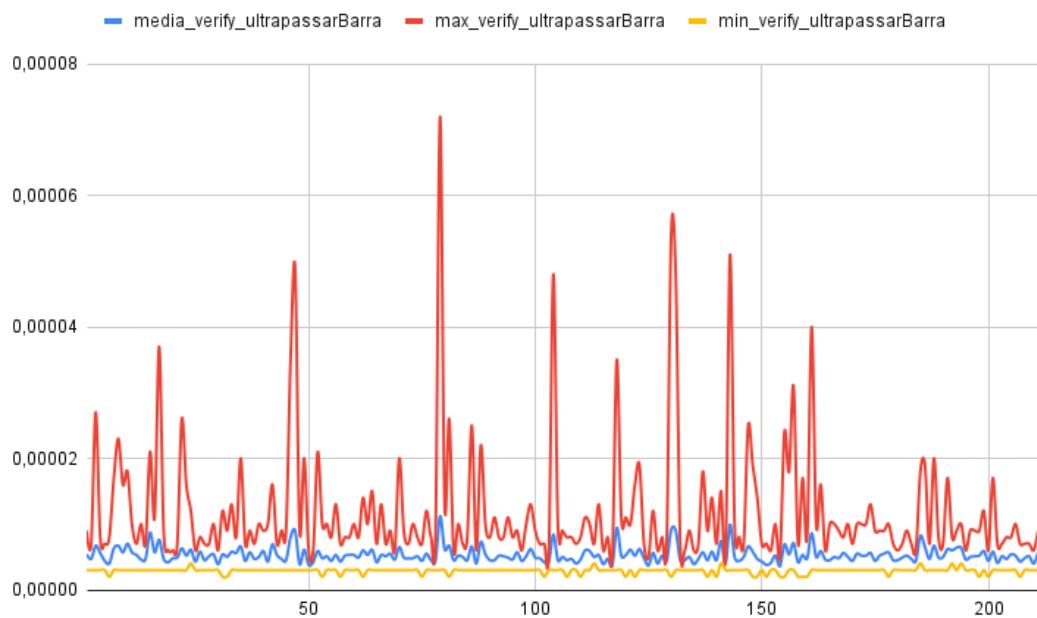
Fonte: Elaborado pelo autor (2023)

Figura 53 – Tempo de execução da função “verify\_extensaoCotovelo” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos



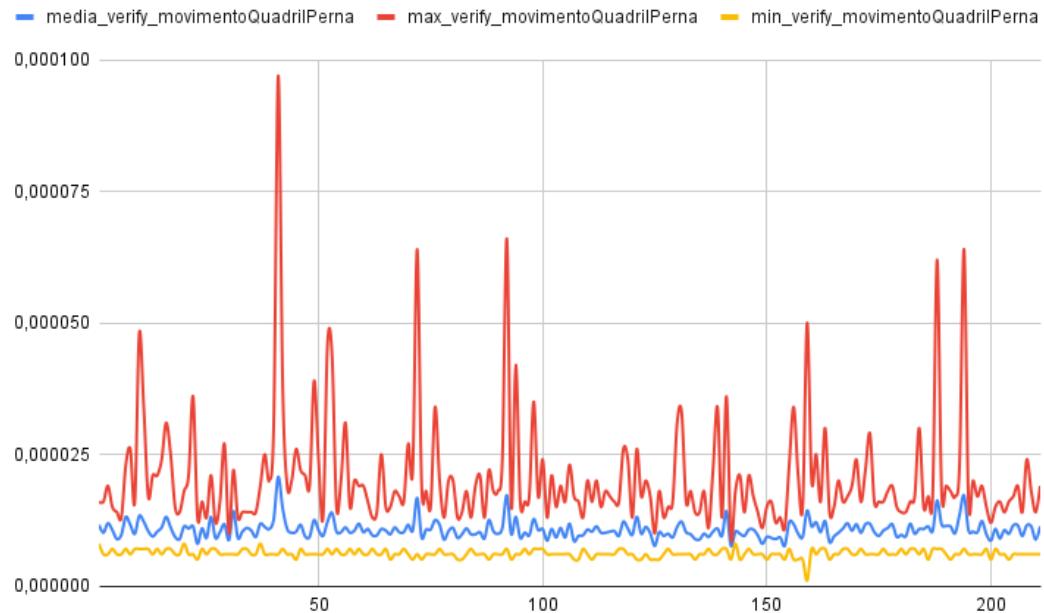
Fonte: Elaborado pelo autor (2023)

Figura 54 – Tempo de execução da função “verify\_ultrapassarBarra” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos



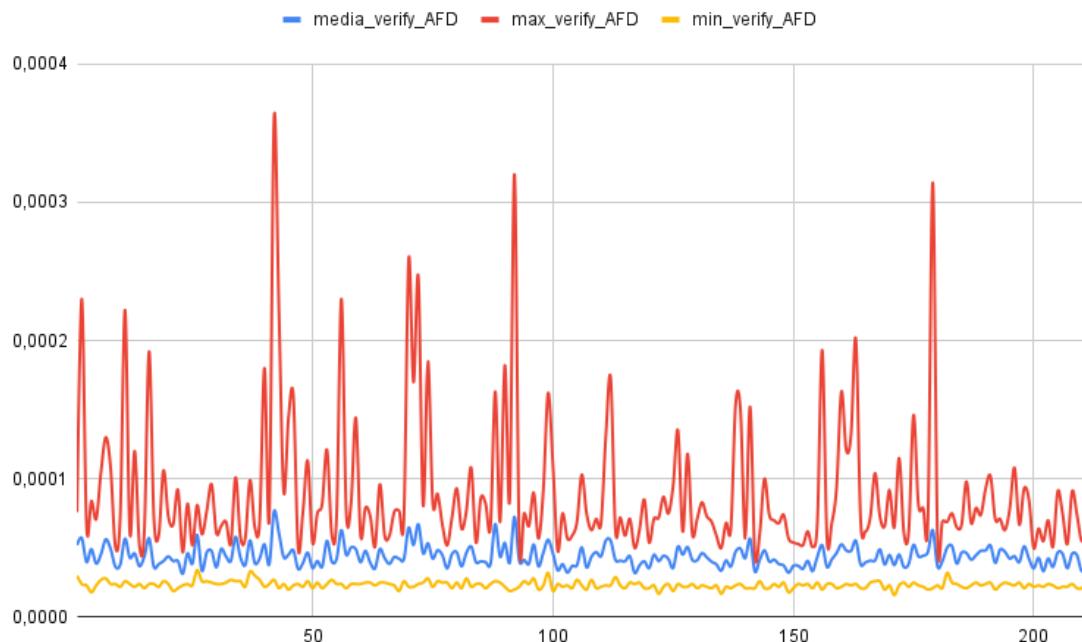
Fonte: Elaborado pelo autor (2023)

Figura 55 – Tempo de execução da função “verify\_movimentoQuadrilPerna” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos



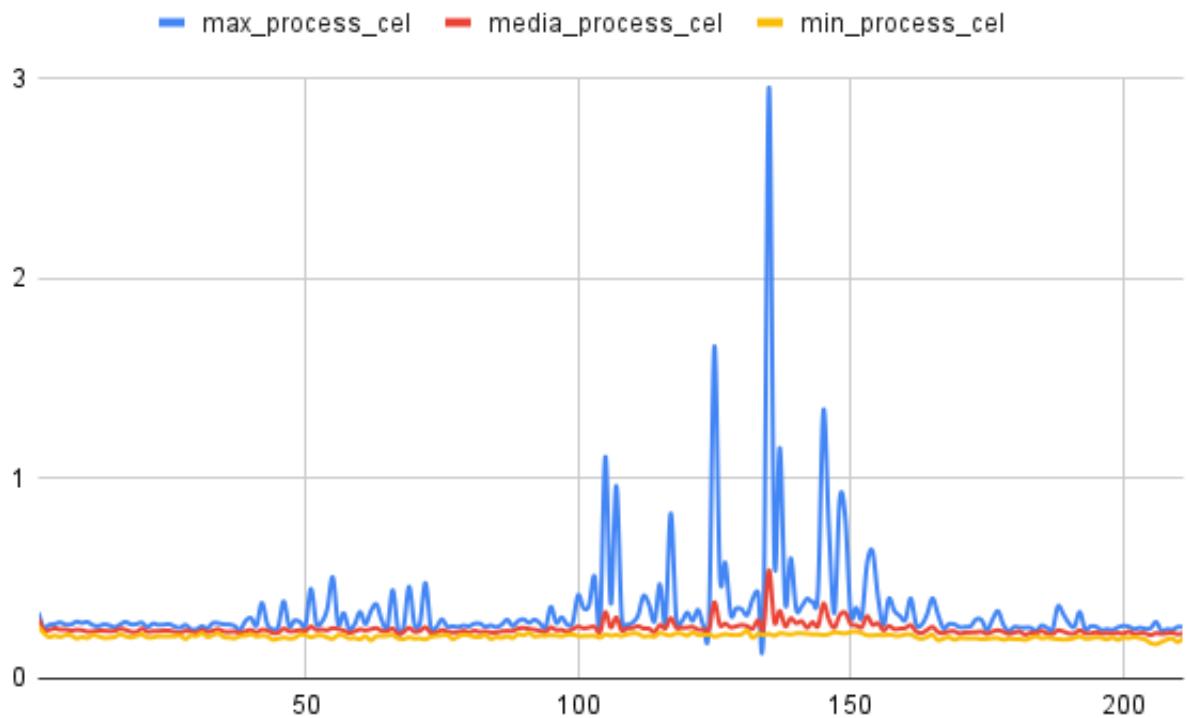
Fonte: Elaborado pelo autor (2023)

Figura 56 – Tempo de execução da função “verify\_AFD” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos



Fonte: Elaborado pelo autor (2023)

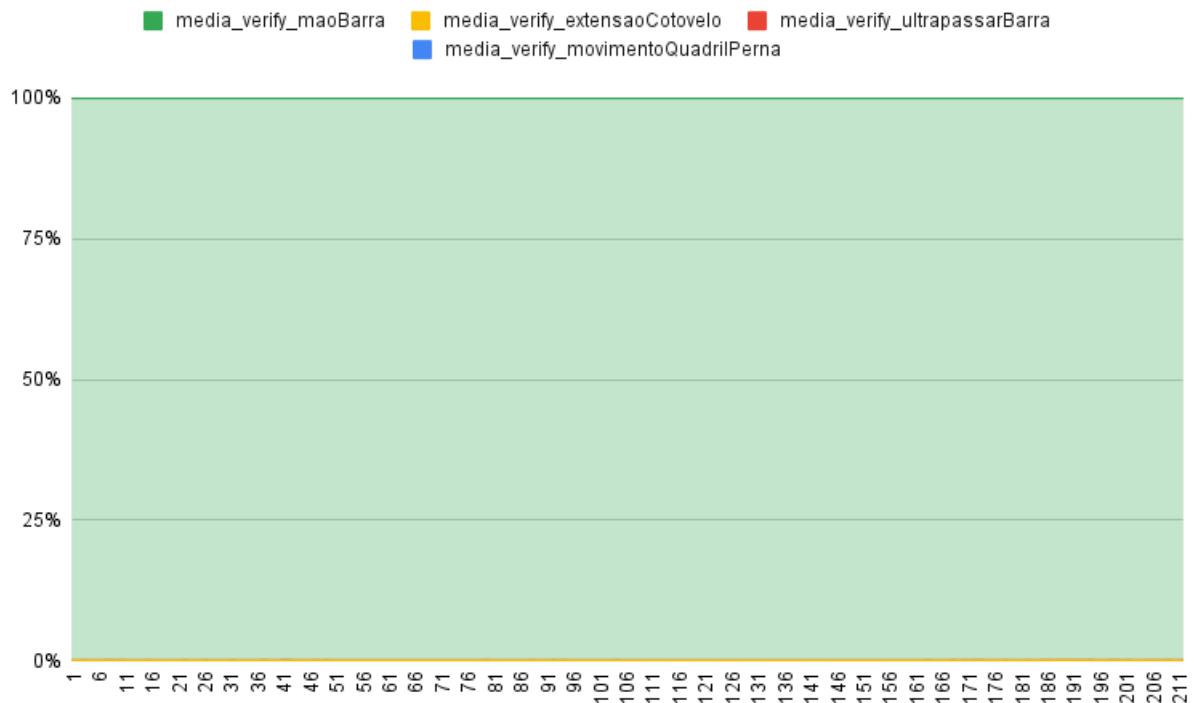
Figura 57 – Tempo de execução da função “process\_cell” em 10 iterações mostrando a média, mínimo e máximo do tempo gasto por quadro em segundos



Fonte: Elaborado pelo autor (2023)

O gráfico a seguir mostra o desempenho relativo de cada função que compõe “verify\_char\_AFD” em relação a propria função “verify\_char\_AFD”

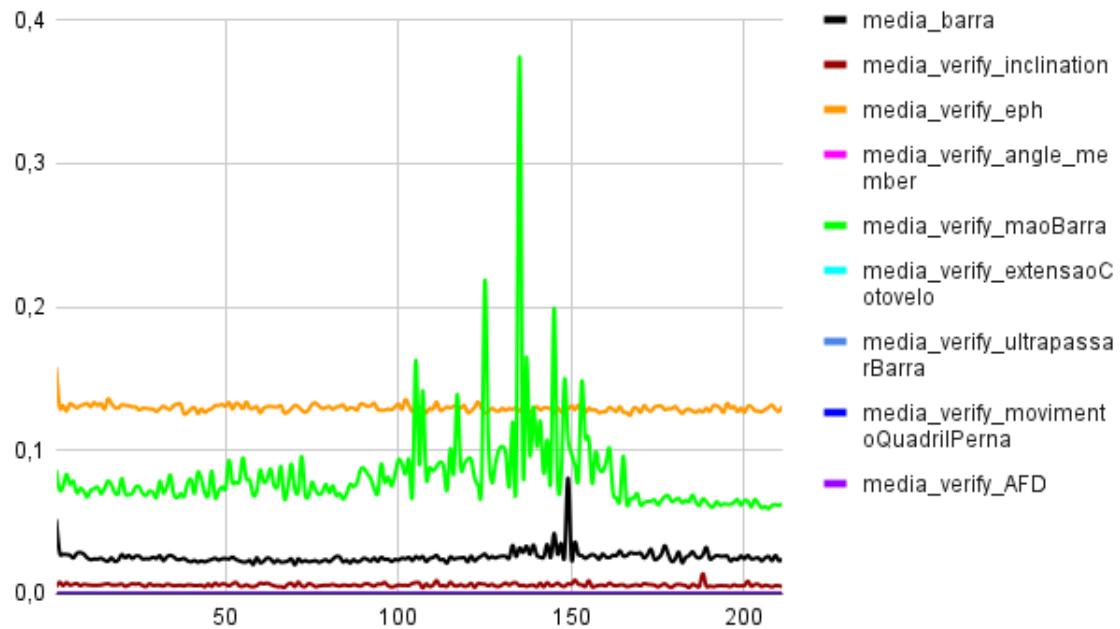
Figura 58 – Média de 10 iterações do percentual de tempo gasto por cada função que compõe “verify\_char\_AFD”



Fonte: Elaborado pelo autor (2023)

O gráfico a seguir apresenta, com base em 10 iterações, a média do tempo gasto em segundos por cada função que compõe “process\_cell”.

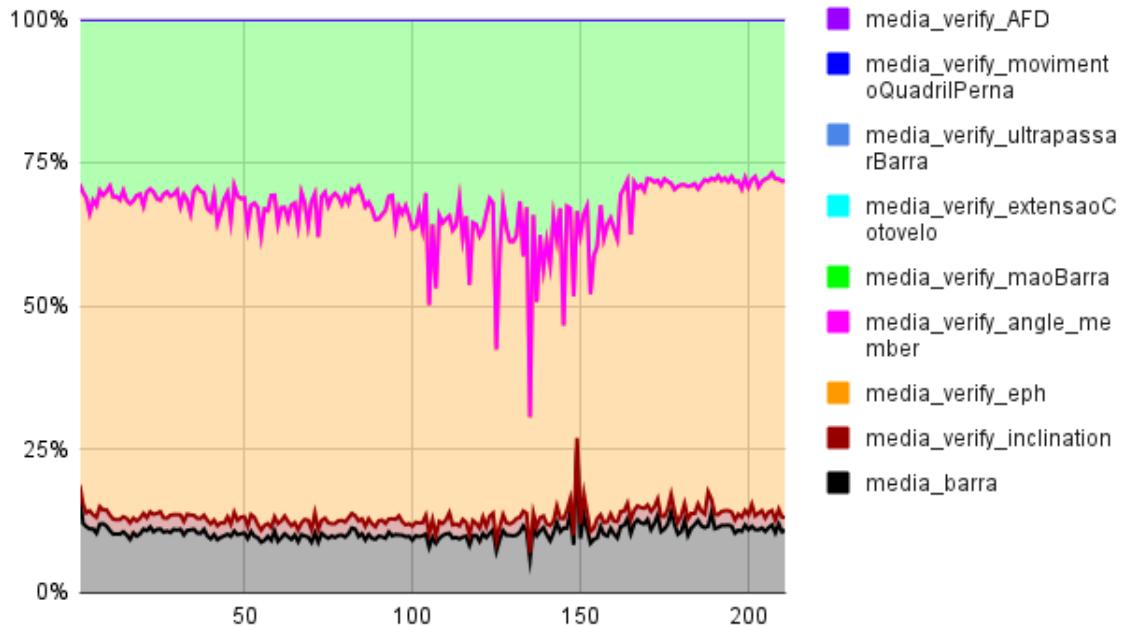
Figura 59 – Tempo médio de execução das funções que compõe “process\_cell”



Fonte: Elaborado pelo autor (2023)

O gráfico a seguir mostra o desempenho relativo de cada função que compõe “process\_cell” em relação a propria função “process\_cell”

Figura 60 – Média de 10 iterações do percentual de tempo gasto por cada função associado ao processamento de um frame



Fonte: Elaborado pelo autor (2023)

As 2 funções mais relevantes para a alta taxa de latência do processamento do frame são “verify\_eph” e “verify\_maoBarra” sendo responsável por cerca de 85.14% do tempo gasto por “process\_cell”

Função	Tempo médio em segundos	Menor tempo registrado em segundos
barra	0,03422	0,016335
verify_inclination	0,00603	0,00317
verify_eph	0,1293617	0,109623
verify_angle_member	0,00013	0,000045
verify_char_AFD	0,0753571	0,039386
verify_AFD	0,0000427	0,000016

Tabela 1 – Tempos médios e menores tempos registrados para cada função que compõe process\_cell

Por fim o tempo médio para a execução da função “process\_cell” foi de 0,2481895403 segundos, enquanto o menor tempo registrado foi de 0,171841 segundos assumindo um desvio padrão de 0,03161229806 e uma variância de 0,0009993373889.

<b>Função</b>	<b>Tempo médio em segundos</b>	<b>Menor tempo registrado em segundos</b>
verify_maoBarra	0,0747025	0,03924
verify_extensaoCotovelo	0,0000126	0,000006
verify_ultrapassarBarra	0,0000051	0,000002
verify_movimentoQuadrilPerna	0,0000104	0,000001

Tabela 2 – Tempos médios e menores tempos registrados para cada função que compõe verify\_char\_AFD

# 6 Conclusão

Neste trabalho, foi desenvolvido e apresentado um protótipo de um sistema de monitoramento do movimento de barra fixa baseado em testes de aptidão física com base em visão computacional, que visa fornecer ao indivíduo um feedback sobre o movimento a fim de auxiliar na formação da consciência corporal quanto a corretude do movimento.

## 6.1 Pontos fracos e possíveis melhorias

O protótipo apresentado demonstra uma precisão satisfatória na avaliação do movimento de barra fixa, de acordo com a maioria dos parâmetros estabelecidos nos editais citados em [2.1.2](#). No entanto, uma exceção notável diz respeito à análise da pegada da mão na posição pronada. A tentativa de verificar se a componente  $x$  da posição do mindinho da mão direita é menor do que a do indicador da mesma mão, e se a componente  $x$  da posição do mindinho da mão esquerda é maior do que a do indicador da mesma mão, revelou-se ineficiente, isso ocorreu devido a uma imprecisão da ferramenta Midia Pipe na detecção dos pontos de referência 17, 18 , 19 e 20 referênciados em [22](#), fazendo com que em um determinado momento do vídeo, houvesse uma inversão nas referências, o que levou a uma interpretação errônea da pegada.

O tempo médio de processamento de um frame foi de 0,2481895403 segundos. No entanto, para o processamento em tempo real de um vídeo com uma taxa de 24 frames por segundo, o tempo limite de processamento é de 0,0416 segundos por frame ou seja gastou 5,96 vezes mais tempo do que o esperado.

## 6.2 Trabalhos futuros

Alem da melhoria dos pontos citados em [6.1](#) é desejável para trabalhos futuros um aprimoramento do relatório de modo que contabilize a quantidade de tempo gasto em cada tipo de contração muscular separado por cada iteração do movimento, assim como o tempo medio gasto em cada execução completa. Elaborar estrategias para avançar com o grau de maturidade da ferramenta.

## Referências

- ALTUNAY, D. *Noções básicas de processamento de imagem e OpenCV*. 2019. Publicado em 25 de outubro de 2019. Disponível em: <<https://developer.ibm.com/articles/learn-the-basics-of-computer-vision-and-object-detection/>>. Citado 2 vezes nas páginas 22 e 23.
- BALLARD, D. H.; BROWN, C. M. *Computer Vision*. [S.l.: s.n.], 1982. ISBN 9780131653160. Citado na página 21.
- BRUNNSTROM, K. *Cinesiologia Clínica*. 6<sup>a</sup> ed.. ed. Rio de Janeiro: Guanabara Koogan, 1986. Citado 3 vezes nas páginas 19, 20 e 21.
- CABRAL, E. L. L. *Filtragem e Suavização*. n.d. Material da disciplina PMR2560 – Visão Computacional, Universidade de São Paulo. Disponível em: <[https://edisciplinas.usp.br/pluginfile.php/4112481/mod\\_resource/content/0/V9%20-Filtragem%20e%20suavizacao.pdf](https://edisciplinas.usp.br/pluginfile.php/4112481/mod_resource/content/0/V9%20-Filtragem%20e%20suavizacao.pdf)>. Citado na página 28.
- CATARINA, A. S. *Processamento de Imagens Digitais - Filtragem Espacial*. 2023. Curso de Ciência da Computação, Unioeste – Campus de Cascavel – PR. Material de aula. Citado na página 28.
- CAZETTA, V.; OLIVEIRA, R. C.; TAVARES, J. M. Os atlas anatômicos como pedagogia cultural e o pós-vida das imagens. *Educação & Realidade*, SciELO Brasil, v. 44, 2019. Citado na página 19.
- DARTFISH - Análise de Vídeo e Coaching Esportivo. <<https://www.dartfish.com/>>. Acessado em 09/08/2023. Citado na página 44.
- DIAS, T. Z. G. Cinesiologia, biomecânica e robótica. 2021. Citado na página 19.
- DUARTE, G. D. Uso da transformada de hough na detecção de círculos em imagens digitais. *Thema-Revista Científica do Centro Federal de Educação Tecnológica*, v. 4, n. 1, p. 51–58, 2003. Citado na página 26.
- EDITAL CBMMG Nº 13, DE 30 DE JULHO DE 2018. 2018. Disponível em: <<https://bombeiros.mg.gov.br/images/concursos/CFSdBM2020/edital.pdf>>. Citado na página 17.
- EDITAL DRH/CRS Nº 07/2021, DE 24 DE JUNHO DE 2021. 2021. Disponível em: <<https://www.policiamilitar.mg.gov.br/conteudoportal/sites/concurso/250620211349486800.pdf>>. Citado na página 17.
- EDITAL Nº 1 – DGP/PF, DE 15 DE JANEIRO DE 2021. 2021. Disponível em: <[https://cdn.cebraspe.org.br/concursos/PF\\_21/arquivos/ED\\_1\\_DPF\\_2021\\_ABТ.PDF](https://cdn.cebraspe.org.br/concursos/PF_21/arquivos/ED_1_DPF_2021_ABТ.PDF)>. Citado na página 17.
- EDITAL SEJUSP Nº. 02/2021 DE 17 DE AGOSTO DE 2021. 2021. Disponível em: <[http://www.seguranca.mg.gov.br/images/2021/Agosto/Documentos/EDITAL\\_CONCURSO\\_PP.pdf](http://www.seguranca.mg.gov.br/images/2021/Agosto/Documentos/EDITAL_CONCURSO_PP.pdf)>. Citado na página 17.

- EDUCATION, I. C. *Machine Learning*. 2020. Disponível em: <<https://www.ibm.com/br-pt/cloud/learn/machine-learning>>. Citado na página 34.
- ESTEVAM, R. *Transformada Hough - Detectando formas geométricas em imagens*. 2021. Medium. Artigo online. Disponível em: <<https://medium.com/turing-talks/transformada-hough-detectando-formas-geom%C3%A9tricas-em-imagens-6d831be61ee8>>. Citado 3 vezes nas páginas 25, 26 e 27.
- FRANKE, L. Análise postural digital para ciclistas. 2016. Citado 3 vezes nas páginas 14, 43 e 44.
- GAZZONI, J. C. et al. Limiarização e binarização na análise de objetos em imagens digitais. *Synergismus scyentifica UTFPR*, v. 1, n. 1, p. 685–695, 2006. Citado na página 33.
- GOIÁS, C. de Bombeiros Militar do Estado de. *Treinamento Físico-Militar e do Teste de Aptidão Física*. 2017. Norma Administrativa n. 02. Secretaria de Segurança Pública e Administração Penitenciária. Disponível em: <<https://www.bombeiros.go.gov.br/wp-content/uploads/2017/05/NA-02-TFM-e-TAF-3.pdf>>. Citado na página 18.
- GONZALEZ, R. C.; WOODS, R. E. *Processamento de imagens digitais*. [S.l.]: Editora Blucher, 2000. Citado 6 vezes nas páginas 21, 22, 23, 24, 41 e 50.
- GOOGLE. *MediaPipe Pose Landmarker*. 2023. Disponível em: <[https://developers.google.com/mediapipe/solutions/vision/pose\\_landmarker](https://developers.google.com/mediapipe/solutions/vision/pose_landmarker)>. Citado na página 39.
- GOOGLE. *MediaPipe Solutions Guide*. 2023. Disponível em: <<https://developers.google.com/mediapipe/solutions/guide.md>>. Citado na página 45.
- GOOGLE. *MediaPipe Solutions*. Acesso em 2023. Disponível em: <<https://developers.google.com/mediapipe/solutions>>. Citado na página 38.
- JAMUNDÁ, T. *Reconhecimento de Formas: A Transformada de Hough*. 2000. Seminário Visão Computacional - CPGCC/UFSC. <<http://www.inf.ufsc.br/~aldo.vw/visao/2000/Hough/index.html>>. Citado 2 vezes nas páginas 25 e 26.
- KENHUB. *Tipos de Movimentos do Corpo Humano*. 2021. Disponível em: <<https://www.kenhub.com/pt/library/anatomia/tipos-de-movimentos-do-corpo-humano>>. Citado na página 20.
- KRIGER, D. *O QUE É PYTHON, PARA QUE SERVE E POR QUE APRENDER?* 2022. Disponível em: <<https://kenzie.com.br/blog/o-que-e-python/>>. Citado na página 38.
- LINDER, J. A. et al. Time of day and the decision to prescribe antibiotics. *JAMA internal medicine*, American Medical Association, v. 174, n. 12, p. 2029–2031, 2014. Citado na página 15.
- LONGEN, A. S. *O Que é GitHub, Para Que Serve e Como Usar*. 2023. <<https://www.hostinger.com.br/tutoriais/o-que-github>>. Acessado em 21 de julho de 2023. Citado na página 40.

- MARIANO, G. et al. O feedback e ensino esportivo. *EFD Deportes*, v. 16, n. 164, 2011. Disponível em: <<https://www.efdeportes.com/efd164/o-feedback-e-ensino-esportivo.htm>>. Citado na página 18.
- MARTIN, R. C. *Clean Code: A Handbook of Agile Software Craftsmanship*. Upper Saddle River, NJ: Prentice Hall, 2008. ISBN 978-0-13-235088-4. Citado na página 44.
- MATTOS, R. S. L. de. *CORRELAÇÃO ENTRE AS CARACTERÍSTICAS ESTRUTURAIS DA FORÇA DE TRAÇÃO E O DESEMPENHO NO EXERCÍCIO DE PUXADA NA BARRA FIXA*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2015. Citado na página 14.
- NEWTON, E. M.; SWEENEY, L.; MALIN, B. Preserving privacy by de-identifying face images. *IEEE transactions on Knowledge and Data Engineering*, IEEE, v. 17, n. 2, p. 232–243, 2005. Citado na página 33.
- NOBRE, L. *TAF: etapa negligenciada por concursaços reprova até 60%*. 2018. Disponível em: <<https://www.metropoles.com/vaga-garantida/taf-etapa-negligenciada-por-concursaços-reprova-ate-60>>. Citado na página 14.
- OLIVETE, C. J. *Autômatos Finitos - LFA: Aula 04*. 2020. Material didático. Apresentado em Linguagens Formais e Autômatos. Disponível em: <<http://www2.fct.unesp.br/docentes/dmec/olivete/lfa/arquivos/Aula04.pdf>>. Citado na página 35.
- OPENCV. *About*. 2023. Disponível em: <<https://opencv.org/about/>>. Citado na página 38.
- OPENCV. *Canny Edge Detection*. 2023. Acessado em 30/08/2023. Disponível em: <[https://docs.opencv.org/3.4/da/d22/tutorial\\_py\\_canny.html](https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html)>. Citado na página 31.
- OPENCV. *Image Thresholding*. 2023. Acessado em 09/09/2023. Disponível em: <[https://docs.opencv.org/4.x/d7/d4d/tutorial\\_py\\_thresholding.html](https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html)>. Citado na página 34.
- OPENCV. *OpenCV Documentation*. 2023. Citado na página 45.
- PÁDUA, F. L. C.; PÁDUA, P. H. C. de. Sistema de apoio às análises tática e física no futsal baseado em visão computacional. In: *Seminário de Discentes dos Programas de Pós-Graduação Stricto Sensu*. [S.l.: s.n.], 2014. Citado 2 vezes nas páginas 14 e 43.
- PAES, G. *TAF: Guia Completo do Teste de Aptidão Física*. 2023. Disponível em: <<https://paesadvogados.com.br/guia-completo-do-taf-teste-de-aptidao-fisica>>. Citado na página 17.
- PARAIBA, G. da. *Exercício na barra fixa é o que mais reprova candidatos no concurso da Polícia Militar e do Corpo de Bombeiros*. 2018. Disponível em: <<https://paraiba.pb.gov.br/noticias/exercicio-na-barra-fix-a-e-o-que-mais-reprova-candidatos-no-concurso-da-policia-militar-e-do-corpo-de>>. Citado na página 14.
- PAULICHEN, H. M. *Ferramentas para análise de partidas de futsal utilizando processamento de imagens e visão computacional*. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2019. Citado 2 vezes nas páginas 14 e 43.

- RAJ, B. *An Overview of Human Pose Estimation with Deep Learning*. 2019. Disponível em: <<https://medium.com/beyondminds/an-overview-of-human-pose-estimation-with-deep-learning-d49eb656739b>>. Citado na página 24.
- RAMOS, M. V. M.; UNIVASF, E. de C. Linguagens formais e autômatos. *Apostila do Curso de Engenharia de*, 2008. Citado 2 vezes nas páginas 34 e 35.
- ROCHA, I. P. Consciência corporal, esquema corporal e imagem do corpo. *Corpus et Scientia*, v. 5, n. 2, 2009. Citado na página 15.
- SAHIR, S. Canny edge detection step by step in python - computer vision. *Towards Data Science*, 2019. Acessado em 30/08/2023. Disponível em: <<https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123>>. Citado 3 vezes nas páginas 27, 30 e 31.
- SALVIANO, M. *Autômatos Finitos e Não-determinismo*. 2022. Material didático. Área de Conhecimento em Algoritmos e Teoria - DCC/UFMG. Disponível em: <[https://homepages.dcc.ufmg.br/~msalvim/courses/ftc/Aula1.1\\_AFDs-AFNs%5Bstill%5D.pdf](https://homepages.dcc.ufmg.br/~msalvim/courses/ftc/Aula1.1_AFDs-AFNs%5Bstill%5D.pdf)>. Citado na página 37.
- SANTOS, H. S. dos. *SISTEMA MUSCULAR*. Disponível em: <<https://www.biologianet.com/anatomia-fisiologia-animal/sistema-muscular.htm>>. Citado na página 20.
- SILVA, F. A. A.; FARIA, G. S. Estudo de algoritmos de rastreamento de objetos em sequências de imagens. 2017. Citado na página 33.
- SILVA, V. R. Cinesiologia e biomecânica. *Rio de Janeiro: SESES*, v. 88, 2015. Citado na página 20.
- SIPSER, M. *INTRODUÇÃO À TEORIA DA COMPUTAÇÃO-2® EDIÇÃO NORTE-AMERICANA*. [S.l.]: Cengage Learning Edições Ltda., 2010. Citado 2 vezes nas páginas 35 e 36.
- TENSORFLOW, C. *Next-Generation Pose Detection with MoveNet and TensorFlow.js*. 2021. Disponível em: <<https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html>>. Citado na página 24.
- THERRIE, B. *Exercícios de cadeia cinética fechada e aberta: o que são e quem pode fazer*. 2021. Disponível em: <<https://www.uol.com.br/vivabem/noticias/redacao/2021/04/07/exercicios-de-cadeia-cinetica-aberta-e-fechada-o-que-sao-e-quais-vantagens.htm>>. Citado na página 20.
- USP, I. de Ciências Matemáticas e de C. I. *Transformada de Hough: Fundamentos e Aplicações*. 2023. Online document. Acessado em 20/08/2023. Disponível em: <<http://www.lps.usp.br/hae/apostila/hough.pdf>>. Citado 2 vezes nas páginas 25 e 26.
- VARGAS, D. *Rastreamento de Objetos x Detecção de Objetos*. 2020. Disponível em: <<https://iaexpert.academy/2020/06/24/rastreamento-de-objetos-x-deteccao-de-objetos/>>. Citado na página 32.

VISÃO Computacional::Deep Learnings. Disponível em: <<https://lapix.ufsc.br/ensino/vision/vision-computacionaldeep-learning/deep-learningreconhecimento-de-poses/>>. Citado na página 24.

ZHANG, A. et al. *Dive into Deep Learning*. [S.l.: s.n.], 2020. <<https://d2l.ai>>. Citado na página 28.