

# Projeto de Graduação



6 de julho de 2023

## **Estimativa e rastreamento de pose 3D em tempo real a partir de múltiplas câmeras**

Matheus Mello de Souza Mendes

Yuri Velasquez Rivas da Silva Moreira



[www.ele.puc-rio.br](http://www.ele.puc-rio.br)

## **Estimativa e rastreamento de pose 3D em tempo real a partir de múltiplas câmeras**

**Alunos: Matheus Mello de Souza  
Mendes**

**Yuri Velasquez Rivas da  
Silva Moreira**

**Orientador: Wouter Caarls**



## **Agradecimentos**

Agradecimento aos nossos pais, amigos e professores por sempre nos apoiarem ao longo de nossa jornada. A Leonardo Magalhães te agradecemos por acreditar neste projeto e ao BCo Space Makers por nos ceder o lugar para fazer nossos testes acontecerem.

## Resumo

Este trabalho tem como foco o desenvolvimento de um sistema de estimativa de poses 3D em tempo real para rastreamento de movimentos humanos. A estimativa de pose é vital na visão computacional e tem aplicações em entretenimento, esportes, saúde e robótica. O desafio de estimar poses 3D envolve identificar as principais articulações do corpo e suas relações espaciais, criando uma representação “esquelética” de figuras humanas, que é complicada por fatores como oclusões, variações do ponto de vista da câmera e interações próximas entre os indivíduos.

O projeto propõe uma solução de baixo custo mesmo que isso possa resultar em uma diminuição na precisão geral do sistema, ele fornece uma alternativa aos sistemas padrão da indústria usados na captura de movimento. Utilizamos um número arbitrário de câmeras ( $n \geq 4$ ) para minimizar erros de projeção e rastreamento causados por oclusões do alvo. A abordagem adotada neste projeto envolve uma lógica de reconstrução, como visto na fig 1-a. A linguagem de programação Python é usada para processamento de imagens, com a ajuda da biblioteca OpenCV para reconstrução de poses 3D. A câmera escolhida para o projeto é Intelbras VIP1430 B G2, e a implementação de código aberto MediaPipe do Google é usada para estimativa de pose 2D.

As limitações desta abordagem e possíveis melhorias também são discutidas, fornecendo uma base sólida para futuras pesquisas e aplicações no campo.

**Palavras-chave:** Estimativa de pose humana 3D, Visão computacional

## Abstract

3D pose estimation and tracking in real time from multiple cameras

This work focuses on the development of a real-time 3D pose estimation system for tracking human movements. Pose estimation is vital in computer vision and has applications in entertainment, sports, healthcare and robotics. The challenge of estimating 3D poses involves identifying the body's major joints and their spatial relationships, creating a "skeletal" representation of human figures, which is complicated by factors such as occlusions, variations in camera point of view, and close interactions between individuals.

The project proposes a low cost solution even though this may result in a decrease in overall system accuracy, it provides an alternative to industry standard systems used in motion capture. We used an arbitrary number of cameras ( $n \geq 4$ ) to minimize projection and tracking errors caused by target occlusions. The approach adopted in this project involves a reconstruction logic, as seen in fig 1-a. The Python programming language is used for image processing, with the help of the OpenCV library for 3D pose reconstruction. The camera chosen for the project is Intelbras VIP1430 B G2, and Google's MediaPipe open source implementation is used for 2D pose estimation.

The limitations of this approach and possible improvements are also discussed, providing a solid basis for future research and applications in the field.

**Key-words: 3D Human Pose Estimation, Computer Vision**

## Sumário

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introdução</b>  | <b>1</b>  |
| a        | Objetivos do Trabalho . . . . .                                | 1         |
| b        | Trabalhos Relacionados . . . . .                               | 1         |
| <b>2</b> | <b>Referencial Teórico</b>                                     | <b>3</b>  |
| a        | Calibração de Câmeras . . . . .                                | 3         |
| 1        | Intro . . . . .  | 3         |
| 2        | Modelo <i>Pin hole</i> . . . . .                               | 3         |
| 3        | Distorção da lente . . . . .                                   | 4         |
| b        | Estimativa de Pose . . . . .                                   | 5         |
| c        | Reconstrução de Cena 3D . . . . .                              | 6         |
| <b>3</b> | <b>Metodologia</b>   | <b>8</b>  |
| a        | Descrição do <i>Dataset</i> Utilizado . . . . .                | 8         |
| b        | Escolha do Hardware e Software Utilizados . . . . .            | 9         |
| 1        | Hardware - Câmeras . . . . .                                   | 9         |
| 2        | Hardware - Infraestrutura . . . . .                            | 11        |
| 3        | Software . . . . .   | 12        |
| c        | Procedimentos para Calibração de Câmeras . . . . .             | 13        |
| d        | Implementação do Modelo de Estimativa de Pose Humana . . . . . | 15        |
| e        | Processo de Reconstrução da Pose em 3D . . . . .               | 15        |
| <b>4</b> | <b>Experimentos</b>  | <b>17</b> |
| a        | Calibração extrínseca . . . . .                                | 17        |
| b        | Resultado Qualitativo (Experimento) . . . . .                  | 18        |
| 1        | Resultado Temporal - FPS . . . . .                             | 20        |
| c        | <i>Dataset</i> e Métricas de Avaliação . . . . .               | 21        |
| d        | Comparação com o Estado da Arte . . . . .                      | 22        |
| <b>5</b> | <b>Discussão</b>   | <b>23</b> |
| a        | Trabalhos relacionados . . . . .                               | 23        |
| b        | Limitações e Possíveis Melhorias . . . . .                     | 23        |
| 1        | fps . . . . .  | 23        |
| 2        | Obturador / <i>Shutter</i> . . . . .                           | 23        |
| 3        | Processamento . . . . .  | 23        |
| 4        | Sincronização de Hardware . . . . .                            | 24        |
| <b>6</b> | <b>Conclusão</b>   | <b>25</b> |
|          | <b>Referências</b>   | <b>25</b> |
| <b>A</b> | <b>Apêndice</b>  | <b>28</b> |
| a        | Camera Calibrador . . . . .                                    | 28        |
| b        | Código de cada <i>thread</i> . . . . .                         | 29        |
| c        | Código de reconstrução do esqueleto . . . . .                  | 30        |

## Lista de Figuras

|    |  |    |
|----|--|----|
| 1  | Pipelines existentes baseados em (a) métodos de reconstrução ou (b) representações volumétricas ou (c) problemas de regressão direta em [T. Wang et al] em [1] | 2  |
| 2  | Modelo de câmera <i>Pinhole</i> [2]  | 3  |
| 3  | Correspondências das operações matemáticas com seus análogos visuais [2]   | 4  |
| 4  | Exemplos de distorção  | 4  |
| 5  | Modelo de esqueleto humano utilizado no MediaPipe [3]  | 5  |
| 6  | Exemplo de utilização do modelo de inferência [4]  | 6  |
| 7  | Exemplo de projeções epipolares com e sem erros  | 6  |
| 8  | Erro RMS dos métodos de triangulação [5]   | 7  |
| 9  | Tempo de execução médio dos métodos de triangulação, reconstruindo 1310720 pontos 10 vezes [5]   | 7  |
| 10 | Imagem exemplo do dataset [6]  | 8  |
| 11 | Imagem do sistema de captura dos dados [6]   | 8  |
| 12 | Optitrack PrimeX 13 [7]  | 9  |
| 13 | Comparativo de mercado, modelos da Optitrack [7]   | 9  |
| 14 | Luxions [8] OAK D S2 - PoE   | 10 |
| 15 | Câmera Intelbras VIP 1430 B  | 10 |
| 16 | Calculadora de distância focal da lente da câmera, densidade de <i>pixels</i> e zonas da câmera em 3D  | 11 |
| 17 | Infraestrutura de hardware / rede  | 11 |
| 18 | Diagrama de blocos da construção lógica da abordagem proposta  | 12 |
| 19 | Exemplo de utilização do app de calibração da MATLAB   | 13 |
| 20 | Exemplos de tabuleiros de xadrez com anotações sobre os cantos   | 13 |
| 21 | Visualização dos parâmetros extrínsecos usando o app de calibração da MATLAB [2]   | 14 |
| 22 | Visualização qualitativa da acurácia da calibração de parâmetros extrínsecos   | 14 |
| 23 | Visualização em mosaico das perspectivas de câmera com a sobreposição das poses inferidas com as juntas de pontuação acima de 75%                              | 15 |
| 24 | Pose humana triangulada a partir das perspectivas da figura 23, pontos coloridos indicam posição das câmeras.  | 15 |
| 25 | Comparativo entre performance das linguagens de implementações do DLT  | 16 |
| 26 | visão do palco da câmera 1   | 17 |
| 27 | visão do palco da câmera 2   | 17 |
| 28 | visão do palco da câmera 3   | 17 |
| 29 | visão do palco da câmera 4   | 17 |
| 30 | visão do palco da câmera 5   | 17 |
| 31 | visão isométrica do palco em 3D  | 17 |
| 32 | visão de topo do palco em 3D   | 17 |
| 33 | Mosaico cena 1 com sobreposição de estimativa de pose 2D.  | 18 |
| 34 | Resultados da triangulação de pose da cena 1, estimativa de pose em 3D.  | 18 |
| 35 | Mosaico cena 2 com sobreposição de estimativa de pose 2D.  | 18 |
| 36 | Resultados da triangulação de pose da cena 2, estimativa de pose em 3D.  | 18 |
| 37 | Mosaico cena 3 com sobreposição de estimativa de pose 2D.  | 19 |
| 38 | Resultados da triangulação de pose da cena 3, estimativa de pose em 3D.  | 19 |
| 39 | Mosaico cena 4 com sobreposição de estimativa de pose 2D.  | 19 |
| 40 | Resultados da triangulação de pose da cena 4, estimativa de pose em 3D.  | 19 |
| 41 | Mosaico cena 5 com sobreposição de estimativa de pose 2D.  | 19 |
| 42 | Resultados da triangulação de pose da cena 5, estimativa de pose em 3D.  | 19 |
| 43 | Estimativa de pose de um frame, em vermelho, comparada ao <i>ground truth</i> do Panoptic, em verde.   | 22 |

## Lista de Tabelas

|   |   |    |
|---|---|----|
| 1 | Modelos e suas justificativas para inviabilidade de implementação. . . . .  | 12 |
| 2 | Tabela de tempos médios para cada estágio do <i>pipeline</i> de processamento. Avaliado em laptop padrão (Intel i5-10400H - 2.60GHz 11th, 8gb RAM, GTX 1650 4GB . . . . . | 21 |
| 3 | Comparação com os métodos estado da arte no <i>dataset</i> Panoptic. A métrica MPJPE foi retirado de [9] e o tempo, retirado de [10] . . . . .                            | 22 |

## 1 Introdução

A estimativa de pose 3D é uma área vital de estudo dentro da visão computacional e tem amplas aplicações que vão desde *motion capture* para a indústria do entretenimento e reconhecimento de ações até realidade aumentada e interação humano-computador. A estimativa de poses 3D é crucial para entender os movimentos e interações humanas em diversos domínios, como esportes, entretenimento, saúde e robótica.

O seu objetivo é recuperar a estrutura 3D subjacente de um corpo humano a partir das poses 2D observadas em imagens ou vídeos, monoculares ou multi câmeras. Resolver esse problema no espaço tridimensional envolve a identificação das principais articulações do corpo e suas relações espaciais, criando uma representação "esquelética" das figuras humanas.

Essa tarefa é particularmente desafiadora devido a fatores como oclusões, variações nos pontos de vista da câmera, informações de profundidade ambíguas e interações próximas entre pessoas. Para lidar com esses desafios, os pesquisadores propuseram várias técnicas e algoritmos que visam estimar poses 3D de maneira precisa e robusta.

### a Objetivos do Trabalho

O propósito deste trabalho é desenvolver um sistema de rastreamento de poses humanas 3D em tempo real, sem a necessidade de utilização de marcadores, sejam ativos ou passivos. Queremos minimizar os erros de projeção e rastreamento ocasionados por oclusões dos alvos de interesse.

Propõe-se oferecer uma solução de baixo custo em comparação com o que é adotado como padrão pela indústria do entretenimento para sistemas de captura de movimento. Mesmo que isso acarrete em uma perda de precisão do sistema como um todo, priorizamos seu baixo custo e a máxima rapidez possível para o sistema operar em tempo real com *feedback* visual. Opta-se por realizar uma implementação mais básica e rápida pois o hardware disponível para treinamento e inferência é de baixo poder de processamento, devido a restrições orçamentárias.

Neste trabalho, o escopo se restringe ao rastreamento de uma única pessoa, considerando a natureza desta pesquisa e o tempo disponível para a execução do projeto.

### b Trabalhos Relacionados

Os avanços nessa área foram impulsionados pelos progressos em *Deep Learning*, que ofereceram abordagens inovadoras para lidar com os desafios inerentes à estimativa de poses humanas.

Existem soluções que permitem rastrear uma ou múltiplas pessoas ao mesmo tempo, apresentando diferentes abordagens para o problema de coerência temporal nas identificações das pessoas reconhecidas:

Rastrear e associar diferentes visualizações do mesmo alvo em diversas câmeras em movimento é um desafio, pois sua aparência, pose e escala podem variar muito. Além disso, com vários alvos, é necessário um módulo de gerenciamento para novos alvos entrarem e alvos antigos saírem do campo de visão de cada câmera.

[S. Yang et al] em [11]

Métodos com estratégias baseadas em reconstrução [12, 13] foram propostas para lidar com o problema em 3 dimensões, primeiro detectando as pessoas e, em seguida, estimando a pose de cada uma independentemente em cada região detectada.

Outras técnicas visam resolver o problema de reconstrução com matrizes volumétricas [14]. Enquanto outros propõe implementações sem depender de tarefas intermediárias, através de algoritmos de regressão direta das poses 3D [1].

[N. D. Reddy et al] em *TesseTrack: Learnable Triangulation of Human Pose* [15], explora uma estrutura de aprendizado de ponta a ponta que estima e rastreia a pose 3D de várias pessoas utilizando uma formulação espaço-temporal e uma rede neural convolucional que considera o tempo como dimensão (4D CNN). Ela produz representações de pessoas que são vinculadas ao longo do tempo e convertidas em poses 3D.

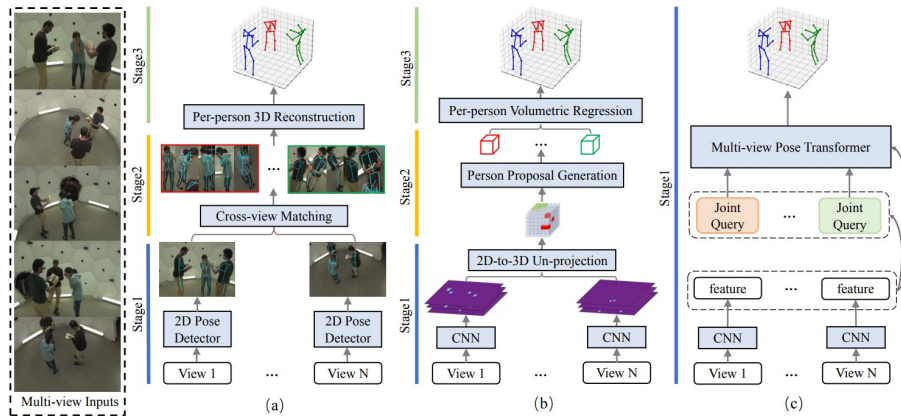


Figura 1: Pipelines existentes baseados em (a) métodos de reconstrução ou (b) representações volumétricas ou (c) problemas de regressão direta em [T. Wang et al] em [1]

Melhorando assim continuamente o estado da arte em conjuntos de dados como o *Human3.6M* [16], *MPII Human* [17], *CMU Panoptic* [6], permitindo a otimização direta da métrica de interesse, em a maioria das vezes a MPJPE (*Mean Per Joint Position Error* em milímetros).

Apesar do progresso significativo realizado, a área de estimativa de poses humanas 3D em tempo real de várias pessoas ainda apresenta muitas questões de pesquisa em aberto e oportunidades para avanços adicionais, com potencial para contribuir de forma significativa em uma ampla variedade de aplicações.



## 2 Referencial Teórico

Nessa secção serão abordadas os procedimentos matemáticos por trás das operações de imagem, modelos teóricos de câmeras e lentes, a descrição dos algoritmos de inferência necessário para captar poses e a solução para uma rápida triangularização de pontos das perspectivas bidimensionais de uma câmera.

### a Calibração de Câmeras

#### 1 Intro

A calibração da câmera é um processo crucial no *pipeline* de tratamento de dados para aplicações de visão computacional, robótica para sistemas de navegação e na reconstrução de cenas em 3D. No processamento de imagens a operação de calibração equivale a modelar o processo de formação da imagem, ou seja, encontrar a relação entre as coordenadas espaciais de um ponto no espaço 3D com o ponto associado na imagem capturada pela câmera. Ao calibrar tais sensores ganha-se grandes capacidades, como medir tamanhos de objetos em unidades do mundo real e determinar a localização relativa da câmera e outros objetos dentro de uma cena.

A motivação por trás da calibração é conseguir processar a imagem crua vinda do sensor e distorcer-la afim de conseguirmos extrair informações úteis de cada *frame* processado.

#### 2 Modelo *Pin hole*

Um dos modelos teóricos mais simples utilizados é o modelo do tipo *pinhole* [18]. Nesse modelo os raios de luz passam pela abertura e projetam uma imagem invertida no lado oposto da câmera.

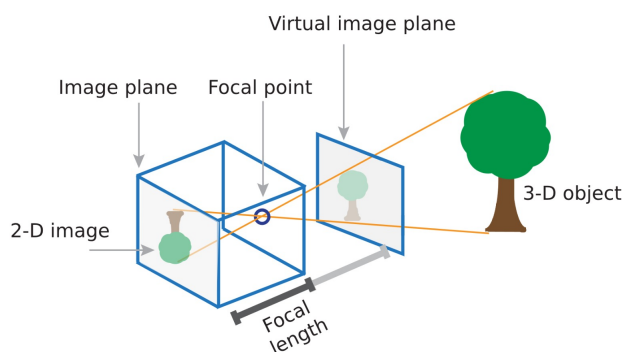


Figura 2: Modelo de câmera *Pinhole* [2]

Usando o modelo *pin hole* os parâmetros extrínsecos definem a pose da câmera (posição e orientação), enquanto os parâmetros intrínsecos especificam o formato da imagem da câmera (distância focal, tamanho do pixel e origem da imagem).

#### Matriz Extrínseca

$$E = [R \mid t]$$

$$R = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$t = [x \quad y \quad z]^T$$

#### Matriz Intrínseca

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Dentro dos parâmetros extrínsecos  $R$  é a matriz de rotação e  $t$  é a matriz de translação.

Nos parâmetros intrínsecos  $f_x$  e  $f_y$  são as distâncias focais, medidas em pixels, ao longo dos eixos  $x$  e  $y$  respectivamente. Os valores  $c_x$  e  $c_y$  são os centros focais [18], entendidos como os *offsets* entre as origens dos eixos de coordenadas da câmera em relação ao mundo.

Conseguimos a partir do modelo construir a matriz da câmera:

$$P = K \begin{bmatrix} R & | & t \end{bmatrix} \quad \left| \quad W \begin{bmatrix} x \\ y \\ z \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \right.$$

Sendo  $P$  a Matriz da câmera e a equação a direita é a correspondência dos pontos da imagem obtida em relação aos pontos de coordenadas do mundo.  $W$  é o fator de escalonamento dos pontos da imagem. Os pontos de coordenadas do mundo são transformados em coordenadas de câmera usando os parâmetros extrínsecos. Enquanto as coordenadas da câmera são mapeadas no plano da imagem usando os parâmetros intrínsecos.

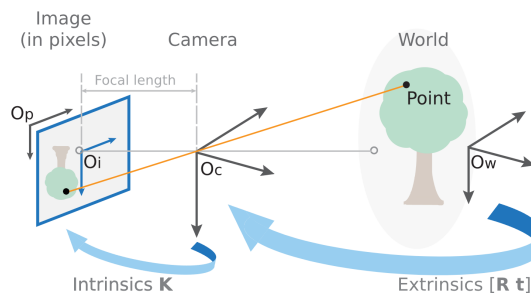


Figura 3: Correspondências das operações matemáticas com seus análogos visuais [2]

### 3 Distorção da lente

Contudo o modelo *pin hole* não leva em consideração as distorções óticas da lente. Queremos com esse processo corrigir as distorções da lente, caso eles já não sejam dados nominais fornecidos pelos fabricantes.

Existem três grandes tipos de distorção nos aparatos ópticos das lentes:

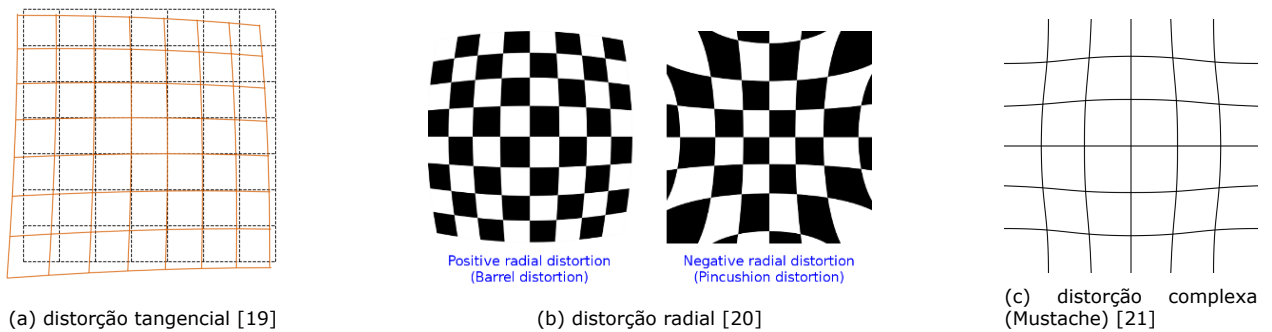


Figura 4: Exemplos de distorção

A distorção tangencial acontece quando o plano do sensor de captação não está alinhado perfeitamente com o plano diretor da lente. Enquanto a distorção radial acontece devido a posição do diafragma em relação a lente, ou então por imperfeições na manufatura do elemento óptico. Distorção complexa é quando há efeitos de distorções radiais e tangenciais acontecendo ao mesmo tempo no aparato.

Equações derivadas das aberrações ópticas do polinômio de terceiro grau de Seidel [22].

### Tangencial

$$\begin{aligned}x_{dist} &= x + [2p_1xy + p_2(r^2 + 2x^2)] \\y_{dist} &= y + [p_1(r^2 + 2y^2) + 2p_2xy]\end{aligned}$$

### Radial

$$\begin{aligned}x_{dist} &= x(1 + k_1r^2 + k_2r^4 + k_3r^6) \\y_{dist} &= y(1 + k_1r^2 + k_2r^4 + k_3r^6)\end{aligned}$$

E no vetor de distorção os componentes  $K_n$  são relativos à distorção radial enquanto os componentes  $p_n$  são relativos à distorção tangencial da lente.

Comumente se representam as distorções somente com seus coeficientes  $k_n$  e  $p_n$ :

$$Dist = (k_1, k_2, p_1, p_2, k_3)$$

## b Estimativa de Pose

Atualmente existem diversas estratégias de Estimativa de pose em tempo real, conforme discutido na introdução deste documento. A alternativa escolhida para a Estimativa de pose neste projeto foi a implementação de código aberto da Google, chamada MediaPipe [3].

O MediaPipe Pose é baseado em um *paper* de nome BlazePose [23] que consiste em um detector de pose 2D, projetado para Estimativa de pose humana em tempo real em dispositivos móveis. Ele produz 33 pontos-chave do corpo para uma única pessoa e opera em condições ótimas com mais de 30 FPS.

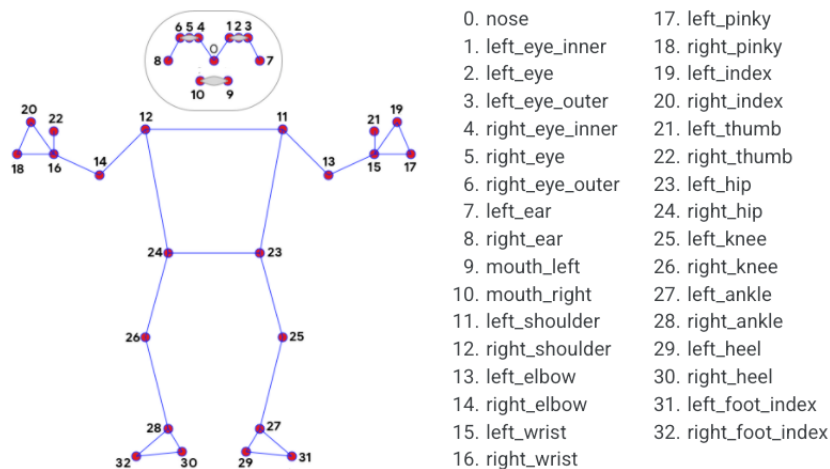


Figura 5: Modelo de esqueleto humano utilizado no MediaPipe [3]

A arquitetura tipicamente usada para esta tarefa é uma rede neural convolucional (CNN - *Convolutional Neural Network*). A entrada desse sistema é uma imagem, e sua saída é um conjunto de coordenadas indicando a localização de cada junta corporal, conforme indicado em fig.5. A CNN realiza isso através de várias camadas de convoluções e transformações não lineares, extraindo características da imagem em diferentes escalas e complexidades. As últimas camadas da rede são treinadas para mapear essas características para as coordenadas das juntas corporais.

Uma vez que a rede produziu sua saída, há uma etapa de pós-processamento para refinar as coordenadas das juntas e garantir que sejam realistas e consistentes com a imagem de entrada. Isso pode envolver a suavização das coordenadas para reduzir a tremulação, garantindo que os marcos obedeçam a certas restrições anatômicas (como não ter seu cotovelo acima do seu ombro [24]) e verificações similares.

Sua única desvantagem é que ele é limitado a rastrear somente uma pessoa por vez. Além disso, como qualquer sistema de aprendizado de máquina, o desempenho do MediaPipe depende da qualidade e representatividade de seus dados de treinamento. Se encontrar poses, tipos de corpo ou outros fatores que sejam significativamente diferentes do que foi treinado, seu desempenho pode diminuir.

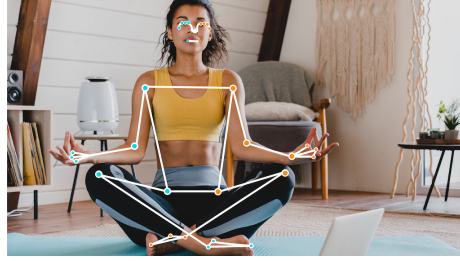


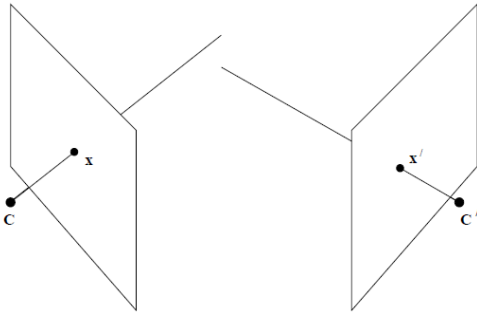
Figura 6: Exemplo de utilização do modelo de inferência [4]

O *pipeline* operacional do BlazePose consiste em dois componentes principais: um detector de pose corporal e uma rede de rastreamento de pose. O detector identifica a presença de uma figura humana, enquanto o rastreador prevê a posição dos pontos-chave do corpo e refina a área de interesse. Se nenhuma figura humana for detectada, o sistema reativa o detector no próximo quadro e assim o processo se repete.

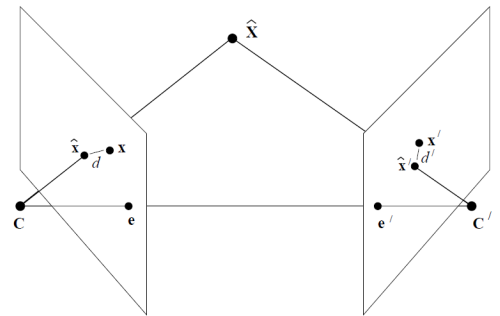
### c Reconstrução de Cena 3D

O problema principal da reconstrução 3D por uma triangularização simples são os possíveis erros nas estimativas de posição dos pontos, isso acarreta em um enviesamento dos raios retroprojetados do plano da imagem de modelo *Pin hole*, no sentido dos pontos, não satisfazendo a restrição de uma geometria epipolar fig.7a.

A Transformação Linear Direta (DLT) [25] é um método popular usado em visão computacional e fotogrametria para a triangulação de pontos e coordenadas 3D a partir de múltiplas visões de imagem 2D. Seu grande benefício é a capacidade de minimização do erro, que mesmo em presença de ruído, consegue achar uma solução próxima que satisfaça as restrições de geometria epipolar fig.7b.



(a) Exemplo de Raios retroprojetados de pontos medidos de forma imperfeita [26]



(b) Exemplo da distancia do erro,  $d$  e  $d'$ , aos pontos medidos com ruído,  $x$  e  $x'$ , e a solução próxima que obtém  $\hat{X}$  dos pontos  $\hat{x}$  e  $\hat{x}'$  [27]

Figura 7: Exemplo de projeções epipolares com e sem erros

Para cada perspectiva com seus pontos medidos perfeitamente, podemos obter as seguintes equações em função do ponto tridimensional  $X$  e das matrizes de projeção de câmera,  $P$  e  $P'$ :

$$x = PX$$

$$x' = P'X$$

Essas equações podem ser combinadas na seguinte forma, onde  $P^{iT}$  e  $P'^iT$  são as linhas de  $P$  e  $P'$ :

$$\begin{bmatrix} xP^{3T} - P^{1T} \\ yP^{3T} - P^{2T} \\ x'P'^{3T} - P'^{1T} \\ y'P'^{3T} - P'^{2T} \end{bmatrix} X = AX = 0$$

A solução não trivial, ou seja  $A \neq 0$ , é estimada usando Decomposição em Valores Singulares (SVD) [28] para resolver o sistema linear de equações, obtendo um vetor unitário singular, correspondente ao menor valor singular de A. Desta forma o DLT realiza uma triangulação de dois pontos,  $u^p$  e  $v^p$ , com um erro satisfatório comparável a outros métodos.

| Method        | Using $u^p$ and $v^p$ |
|---------------|-----------------------|
| Plane-line    | <b>0.261421</b>       |
| DLT           | 0.245759              |
| Inhomogeneous | 0.246052              |
| Optimal       | 0.245738              |

Figura 8: Erro RMS dos métodos de triangulação [5]

Além disso, também foi levada em consideração a velocidade de processamento, parâmetro crucial para realizar, em tempo real, a reconstrução dos vários pontos presentes em cada um dos múltiplos pares de câmeras. Pode se perceber que, utilizar SVD na DLT provê um algoritmo robusto e relativamente rápido, mas sua velocidade e precisão podem ser afetadas pelo número de condição da matriz a ser decomposta [18]. Na prática, o uso da SVD no DLT é frequentemente um bom equilíbrio entre velocidade e precisão. Esse método mostra resultados do erro RMS comparáveis à outros métodos fig.8.

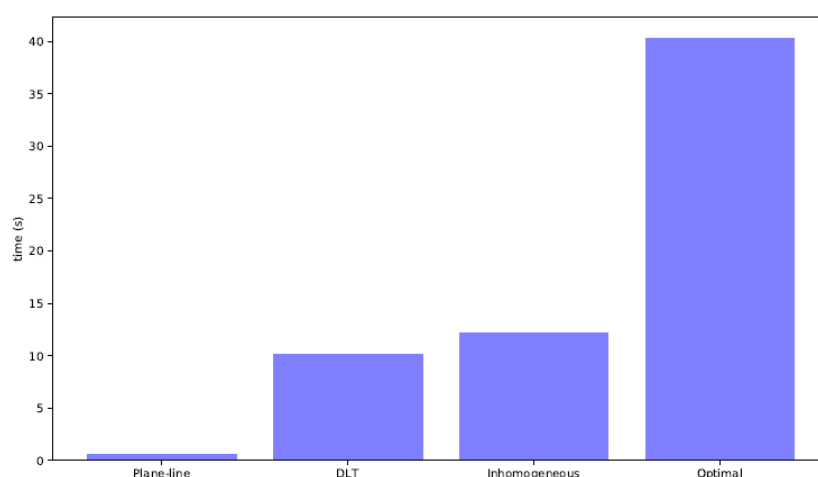


Figura 9: Tempo de execução médio dos métodos de triangulação, reconstruindo 1310720 pontos 10 vezes [5]

Em comparação com seus *peers* o método DLT tem a vantagem da simplicidade e rapidez, pois envolve uma solução direta para um sistema linear de equações. No entanto, pode ser mais sensível a ruídos e *outliers* nas correspondências de pontos. O uso de normalização no DLT pode mitigar esses problemas até certo ponto. Desta forma, é possível reconstruir de forma eficiente e versátil, a tridimensionalidade de um esqueleto humano, que mesmo chegando nas centenas de pontos para ser representado, não ira afetar a performance do sistema de forma significativa. Se mostrando assim um bom método a se utilizar.

## 3 Metodologia

Esta seção fornece uma análise detalhada das opções de hardware e software implementadas no problema de estimativa de pose humana 3D. O capítulo avalia diversas opções de câmeras, focando em parâmetros como fps, resolução, campo de visão, latência, entre outros, com a seleção da Intelbras VIP1430 B G2 como protótipo de câmera. Ele também explica a lógica do uso do Python, detalhando o processo de calibração da câmera usando o MATLAB e descreve o uso do processamento *multi-threaded* para a implementação do modelo de estimativa de pose humana. Conclui com a exploração da reconstrução de pose 3D, utilizando o algoritmo Direct Linear Transform da biblioteca OpenCV para triangulação e retornando uma visualização gráfica tridimensional em tempo real.

### a Descrição do *Dataset* Utilizado

O uso de um conjunto de dados pré-existente é crucial para o desenvolvimento de soluções de rastreamento de esqueleto humano baseadas em visão computacional. Esses conjuntos de dados fornecem exemplos rotulados amplos para modelos de aprendizado de máquina, garantindo que eles tenham um desempenho robusto em diversos cenários. Além disso, eles fornecem *benchmarks* padronizados para comparação de desempenho e economizam tempo e recursos significativos, eliminando a necessidade de coleta e anotação manual de dados.



Figura 10: Imagem exemplo do dataset [6]

O *dataset* Panoptic da Universidade Carnegie Mellon (CMU) [6] é um conjunto de dados de grande escala projetado para tarefas de entendimento visual em visão computacional. Tal conjunto de dados fornece anotações densas para diferentes cenários, mono e multi pessoas, para diversas tarefas, incluindo segmentação de instância, segmentação semântica, estimativa de pontos-chave e rastreamento. Visa preencher a lacuna entre o entendimento 2D e 3D, oferecendo dados multimodais, incluindo vídeos RGB, mapas de profundidade, poses de câmera e anotações semânticas.

Os dados disponibilizados consistem em:

- 480 câmeras VGA, 640 x 480, 25 fps, sincronizadas entre si
- 31 câmeras HD, 1920 x 1080, 30 fps, sincronizadas entre si
- 10 Sensores Kinect (RGB-D) 1920 x 1080 (RGB), 512 x 424 (profundidade-D), 30 fps, sincronizadas entre si

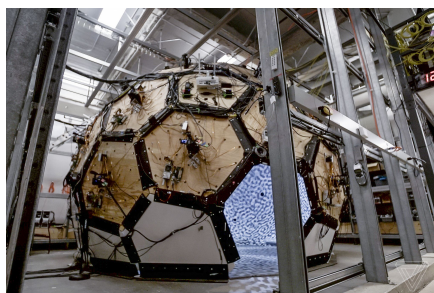


Figura 11: Imagem do sistema de captura dos dados [6]



## b Escolha do Hardware e Software Utilizados

Análise de especificações técnicas, custos e benefícios de diferentes modelos de câmeras, e implementações de hardware, assim como abordagens de software para resolução do problema.

### 1 Hardware - Câmeras

Devemos definir quais são as condições de contorno para a abordagem ótica para o problema de estimação de pose humana em 3D. O objeto de estudo mais importante da definição do hardware é a câmera escolhida. Entendendo quais são os parâmetros importantes para pesquisa levantamos:

- fps
- Resolução
- Ângulo de abertura da lente (FOV)
- Latência
- Processamento local vs centralizado
- Conexão por rede (RJ45) / USB
- Preço

Olhamos agora para o que as soluções líderes de mercado estão fazendo e quais são as especificações técnicas para as câmeras em uso, assim como um comparativo técnico para diferentes modelos.

### Optitrack PrimeX 13

- 240 fps (nativo)
- Resolução: 1280\*1024px
- FOV: 56°x45°
- Latência: 4.2ms
- Processamento local
- Shutter Global
- \$2,499 USD / un



Figura 12: Optitrack PrimeX 13 [7]

O modelo acima é um modelo custo-benefício, para aplicações de alta precisão sugerido para áreas de médio porte, com especificações técnicas média para alta nos padrões necessitados para a aplicação, contudo seu preço é contextualmente muito alto para ser utilizada neste projeto.






| Prime <sup>x</sup> 41   | Prime <sup>x</sup> 22   | Prime <sup>x</sup> 13   | Flex 13  | Flex 3  |
|---|---|---|--|---|
|  |  |  |  |  |
| \$6,499   | \$3,999   | \$2,499   | \$1,099  | \$659   |
| Image Sensor  |   |   |  |   |
| 2048 × 2048<br><small>Resolution</small>  | 2048 × 1088<br><small>Resolution</small>  | 1280 × 1024<br><small>Resolution</small>  | 1280 × 1024<br><small>Resolution</small>   | 640 × 480<br><small>Resolution</small>  |
| 180 Hz<br><small>Native Frame Rate</small>  | 360 Hz<br><small>Native Frame Rate</small>  | 240 Hz<br><small>Native Frame Rate</small>  | 120 Hz<br><small>Native Frame Rate</small>   | 100 Hz<br><small>Native Frame Rate</small>  |
| 5.5 ms<br><small>Latency</small>  | 2.8 ms<br><small>Latency</small>  | 4.2 ms<br><small>Latency</small>  | 8.3 ms<br><small>Latency</small>   | 10 ms<br><small>Latency</small>   |

Figura 13: Comparativo de mercado, modelos da Optitrack [7]

## Luxions OAK-D PoE

- 120 fps (stereo) / 60 fps (RGB)
- Resolução: 1280\*800 (stereo) / 4056x3040 (RGB)
- FOV: 80°×55° (stereo) / 66°×54° (RGB)
- Latência: Não informado
- Processamento local com acelerador de inferência de modelo neural (4 TOPS)
- Shutter Global (stereo) / Rolling Shutter (RGB)
- \$399 USD / un

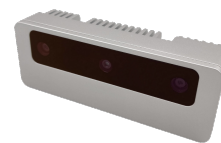


Figura 14: Luxions [8] OAK D S2 - PoE

A câmera Luxions OAK-D PoE é uma solução de visão robusta e potente, projetada para ambientes difíceis, combinando uma câmera de profundidade estéreo com uma câmera colorida de alta resolução, além de recursos de inferência de Redes Neurais e visão computacional no próprio dispositivo. A OAK-D PoE possui uma carcaça robusta, classificada como IP67, necessária para uso industrial e externo.

Entendendo as limitações de orçamento para este projeto, assim como a disponibilidade dos materiais disponíveis no mercado brasileiro, optamos por seguir com o seguinte modelo de câmera:

## Intelbras VIP1430 B G2

- 25 fps
- Resolução: 2688\*1520px
- FOV: 82°×45°
- Latência: Não informado
- Sem processamento local
- Rolling Shutter
- \$99 USD / un



Figura 15: Câmera Intelbras VIP 1430 B

A caráter de protótipo a câmera supracitada irá nos atender de maneira satisfatória para a prova de conceito e para os testes práticos, tendo o menor custo possível enquanto atendendo ao mínimo das especificações necessárias: capaz de ser conectada em rede e assim disposta em um ambiente de médio porte, com uma resolução boa e baixo custos, mesmo que comprometendo o fps ideal.

Com a Intelbras VIP1430 B G2 conseguimos prever o tamanho do cone de visão, bem como estimar quantos *pixels* por metro (ppm) teremos a cada distância de uma possível configuração de montagem, de forma a sempre estar em uma zona de reconhecimento de pose ou melhor.



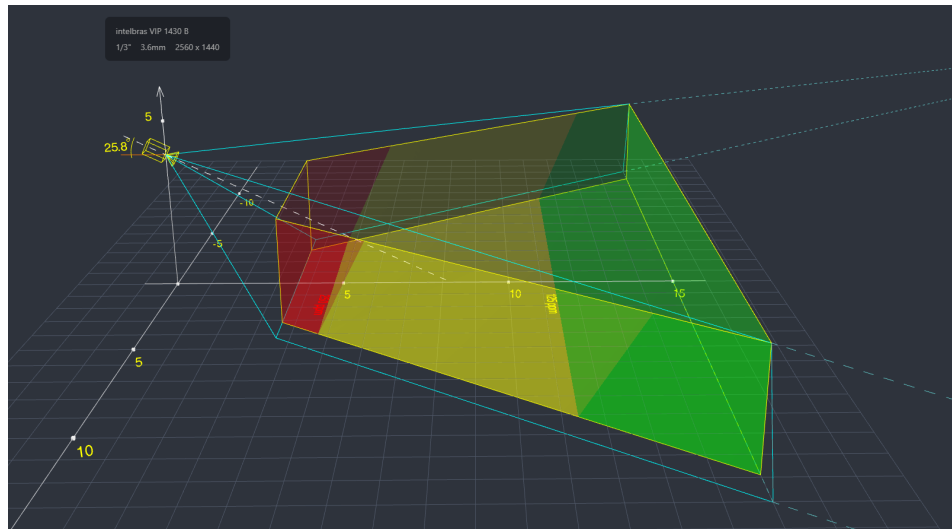


Figura 16: Calculadora de distância focal da lente da câmera, densidade de *pixels* e zonas da câmera em 3D

Conforme estimado pelo 3D, e apresentado pela tabela: temos as 3 das 4 melhores classificações de identificação possíveis, limitadas somente pelo hardware da câmera e nossa habilidade de otimizar o software.

- 1000 ppm - Forte Identificação
- 250 ppm - Identificação (vermelho)
- 125 ppm - Reconhecimento (amarelo)
- 62 ppm - Observação (verde)
- 25 ppm - Detecção
- 12 ppm - Monitoramento

## 2 Hardware - Infraestrutura

Sendo assim definimos que este projeto emprega uma configuração de hardware que inclui cinco câmeras de segurança IP (Intelbras VIP1430 B G2) conectadas a um laptop padrão (Intel i5-10400H - 2.60GHz 11th, 8gb RAM, GTX 1650 4GB) através de um switch PoE (Intelbras SF 900 Poe+) conforme o gráfico abaixo.

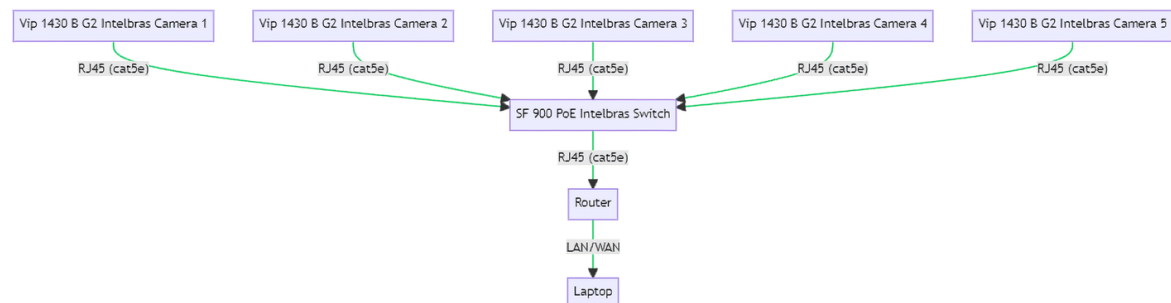


Figura 17: Infraestrutura de hardware / rede

### 3 Software

Foi escolhido *Python* [29] como a linguagem de programação para implementar o processamento das imagens e geração de dados, dentre as principais razões estão:

- **Bibliotecas e Frameworks:** Abundância de bibliotecas performáticas e estruturas de dados poderosas que simplificam a implementação e inferência de modelos de aprendizagem de máquina.
- **Facilidade e compatibilidade para prototipagem:** A natureza multi-plataforma e de tipagem dinâmica permite iterar, mesmo em diferentes sistemas operacionais, inúmeras abordagens, algoritmos e parâmetros de forma eficiente, ajudando na rapidez para otimização de um esboço de projeto.
- **Processamento e visualização de dados:** Bibliotecas robustas e otimizadas como OpenCV facilitam as operações necessários nas imagens para inferência das poses, assim como Matplotlib [30], que possibilita a visualização tridimensional dos resultados de forma prática.

Em nossa implementação de código, utilizamos o paradigma de *multithreads* assíncronas do *python* [31], para executar de forma concorrente a obtenção das várias imagens, agilizando esta tarefa que é limitada principalmente pelo tempo de espera de entrada/saída dos dados de um *frame*. Os *frames* de cada câmeras são acessados da rede local por um IP fixo, via protocolo RTSP [32].

No gráfico abaixo apresentamos a estrutura em diagrama de blocos e fluxograma de como será implementada lógica e fluxo de dados do algoritmo.

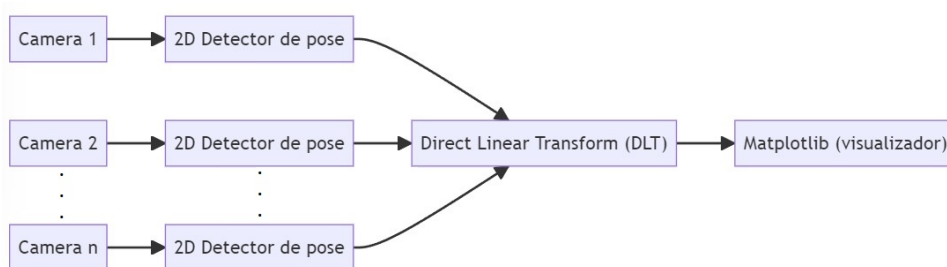


Figura 18: Diagrama de blocos da construção lógica da abordagem proposta

Dentro das técnicas abordadas nos trabalhos relacionados, foram testados algoritmos de inferência de pose 2D, tanto por uma estratégia de reconstrução de pontos de junta direta à imagem, quanto por uma detecção e segmentação de figuras humanas para uma reconstrução posterior.

Foram estudadas outras opções de modelos e *papers* existentes, contudo descartados por diferentes motivos, apresentados abaixo:

| Modelo              | Motivo  |
|---------------------|---|
| TesseTack [15]      | Não existe implementação em código.   |
| MvP/EasyMocap [1]   | Não opera em tempo real.  |
| VoxelPose [14]      | Sem hardware robusto o suficiente disponível para treinamento e falta de modelo pre-treinado. |
| OpenPose [33]       | Só opera em 2D, poderia ser uma solução alternativa ao MediaPipe, contudo é mais lento.       |
| 4D Association [34] | Dificuldades técnicas de implementação.   |
| Keras CPN [35]      | Muito mais lento em implementação de várias imagens de forma simultânea.                      |

Tabela 1: Modelos e suas justificativas para inviabilidade de implementação.

Ao final da pesquisa optamos por utilizar o modelo do Google, chamado MediaPipe, essa solução é leve e rápida, proposta para ser utilizada em aplicativos web e celulares. Funciona em tempo real e já está pre-treinada sendo distribuída de maneira *open-source* por seus criadores. Sua única desvantagem é sua capacidade limitada de reconhecer somente uma pessoa por vez em sua rede neural.

### c Procedimentos para Calibração de Câmeras

O software MATLAB, da MathWorks [36] apresenta em seu conjunto de apps internos, do pacote *Computer Vision Toolbox* [37], uma rotina de calibração de câmeras. O aplicativo *MATLAB Camera Calibrator* oferece uma maneira robusta e eficiente de estimar os parâmetros intrínsecos da câmera, como distância focal, ponto principal e coeficientes de distorção da lente. Ele emprega uma técnica de calibração onde as imagens de um padrão de tabuleiro de xadrez, de tamanho conhecido, são usadas como amostras de calibração.

Para começar, várias imagens de um padrão de tabuleiro de xadrez são tiradas a partir de vários ângulos e distâncias usando a câmera a ser calibrada. É importante garantir que o padrão de calibração (tabuleiro de xadrez) esteja completamente visível nas imagens e devem cobrir o maior campo de visão possível da câmera para melhorar a precisão da calibração.

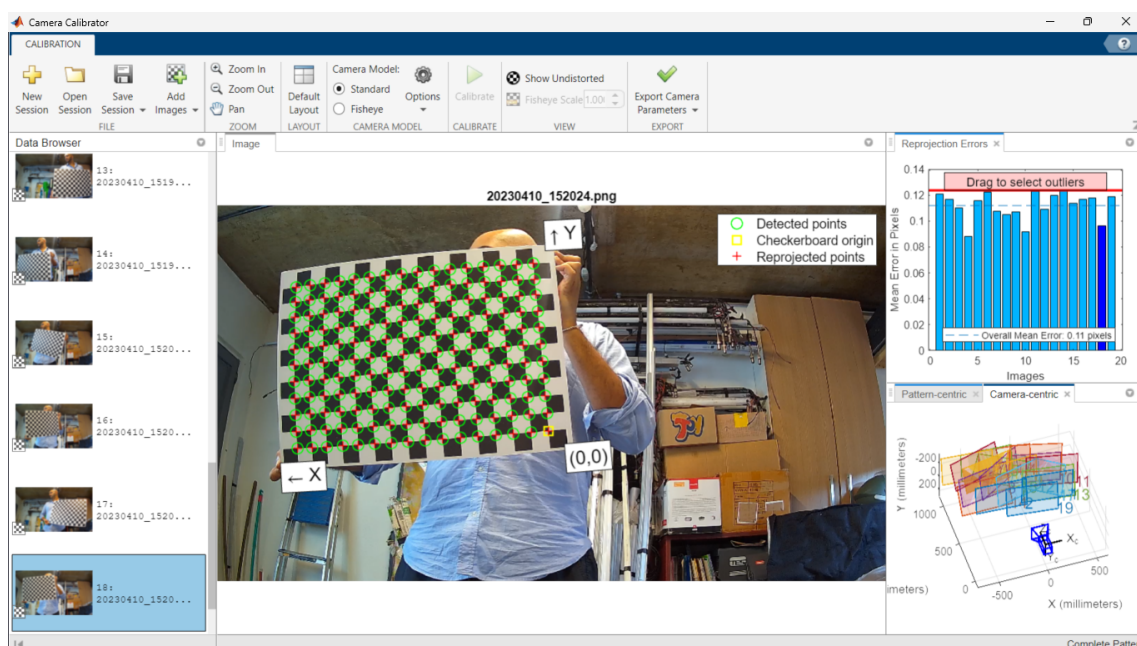
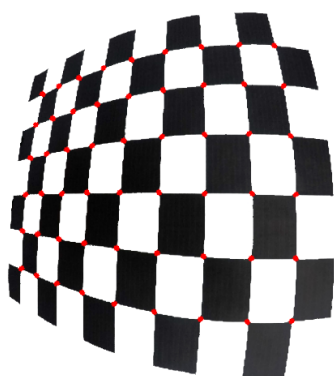
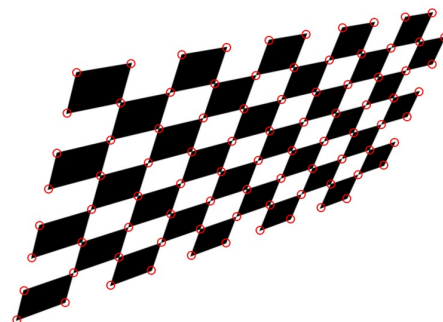


Figura 19: Exemplo de utilização do app de calibração da MATLAB

A detecção de cantos do tabuleiro de xadrez desempenha um papel fundamental no processo de calibração da câmera. O padrão de tabuleiro de xadrez é amplamente utilizado para esse propósito devido ao seu alto contraste, padrão regular e cantos facilmente identificáveis através de visão computacional. Esses cantos servem como pontos de referência conhecidos no plano de imagem 2D [20].



(a) Distorção de lente *fish-eye*



(b) Distorção linear

Figura 20: Exemplos de tabuleiros de xadrez com anotações sobre os cantos

Quando o padrão do tabuleiro de xadrez é visto de diferentes ângulos e distâncias, seu formato e tamanho parecem mudar nas imagens devido às distorções da lente, levando em consideração a posição relativa entre a câmera e o tabuleiro. Essas distorções são influenciadas pelos parâmetros intrínsecos da câmera, como a distância focal e os coeficientes de distorção. Ao analisar como as localizações dos cantos do tabuleiro de xadrez mudam em diferentes imagens, é possível estimar através de algoritmos tais parâmetros intrínsecos.

O processo envolve encontrar a transformação que mapeia pontos do mundo 3D (o padrão do tabuleiro de xadrez na vida real) para o plano de imagem 2D (as imagens capturadas pela câmera). Como as dimensões reais e o layout do tabuleiro de xadrez são conhecidos, e a localização dos cantos na imagem podem ser detectadas, esse conjunto de correspondências pode ser usado para resolver os parâmetros da câmera que melhor se ajustam a essa transformação. Este processo é frequentemente realizado através de técnicas de otimização que visam minimizar a diferença (ou erro) entre os pontos 2D projetados calculados usando os parâmetros da câmera estimados e os pontos 2D detectados na imagem.

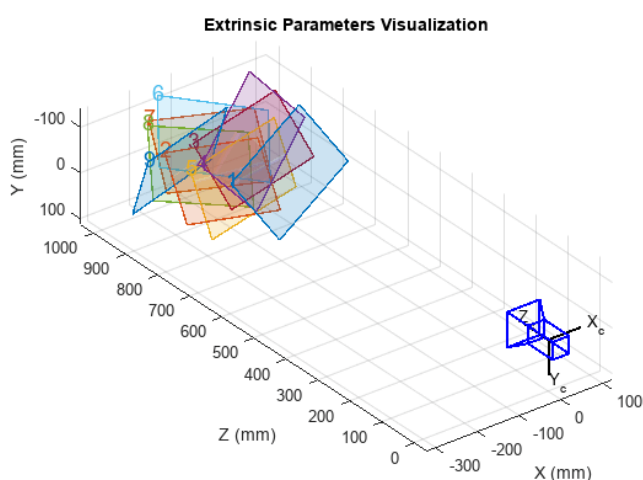


Figura 21: Visualização dos parâmetros extrínsecos usando o app de calibração da MATLAB [2]

Ao final da operação consegue-se o objeto *cameraParams* com os parâmetros intrínsecos de uma câmera, que logo após, é convertido para a mesma estrutura de dados e sistema de coordenadas espaciais da biblioteca OpenCV, exemplificado no apêndice [a].

Para obter os Parâmetros extrínsecos de forma prática, foi feito uma programa em *Python* que obtém esses parâmetros de uma única imagem de câmera. Desta forma quando estão ajustadas em suas pose finais de utilização do sistema, este programa irá gerar as matrizes de rotação e translação, assim como, uma imagem sem distorções com uma indicação e distância do eixo de origem do tabuleiro para ser avaliada qualitativamente.



Figura 22: Visualização qualitativa da acurácia da calibração de parâmetros extrínsecos



## d Implementação do Modelo de Estimativa de Pose Humana

Assim como descrito no início da seção [3], é utilizado um conjunto de *threads* com mesma lógica de processamento do apêndice [b], em cada uma delas, a medida que são obtidas novas imagens das câmeras com uma figura humana, será inferida uma detecção de pose, com seu respectivo contexto temporal em relação a sua perspectiva e ao *frame* anterior. Também é informado para cada junta da pose uma pontuação de visibilidade, indicando a confiabilidade do ponto de acordo com a rede neural.

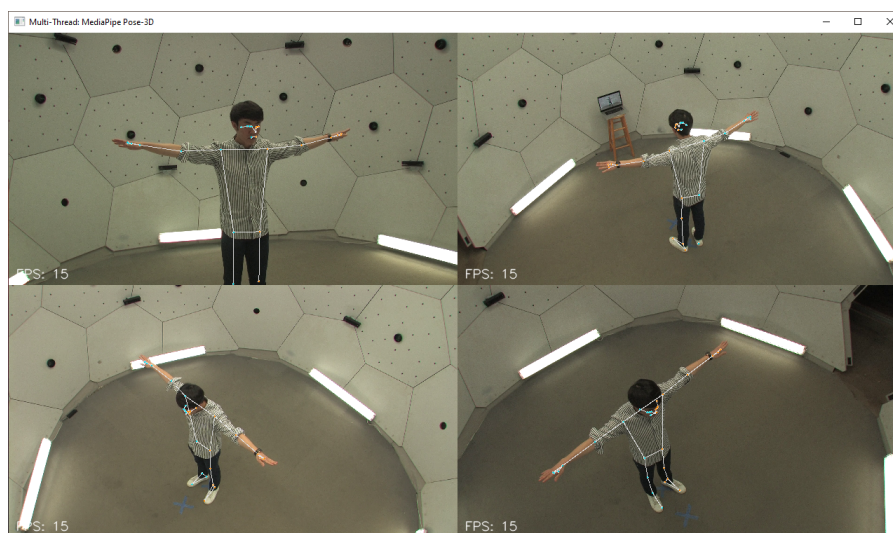


Figura 23: Visualização em mosaico das perspectivas de câmera com a sobreposição das poses inferidas com as juntas de pontuação acima de 75%

## e Processo de Reconstrução da Pose em 3D

Após cada instância de estimação de pose terminar sua inferência, são feitas triangularizações dos pontos de junta para todos as combinações em pares únicos de câmeras. Também é atrelada, a cada ponto 3D, a menor pontuação de visibilidade do par de pontos 2D utilizados, desta forma é possível obter uma pose 3D final, em que, todos seus pontos apresentam a maior confiabilidade disponível. Aplicação deste procedimento esta presente no apêndice [c].

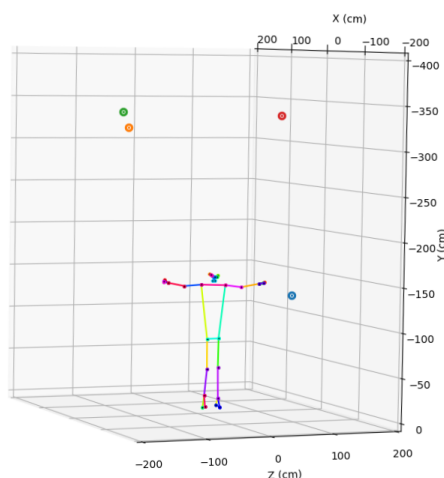


Figura 24: Pose humana triangulada a partir das perspectivas da figura 23, pontos coloridos indicam posição das câmeras.

Além disso, O algoritmo DLT, abordado na página 6, é um método de triangulação incluso na biblioteca *OpenCV* [20], isso apresenta uma vantagem, visto que, os algoritmos desta biblioteca são escritos nativamente em C++ [38,39], linguagem compilada e estaticamente tipada, dando uma vantagem performática em comparação se fosse implementado em *Python*, este aspecto também foi confirmado quantitativamente, por experimentos próprios.

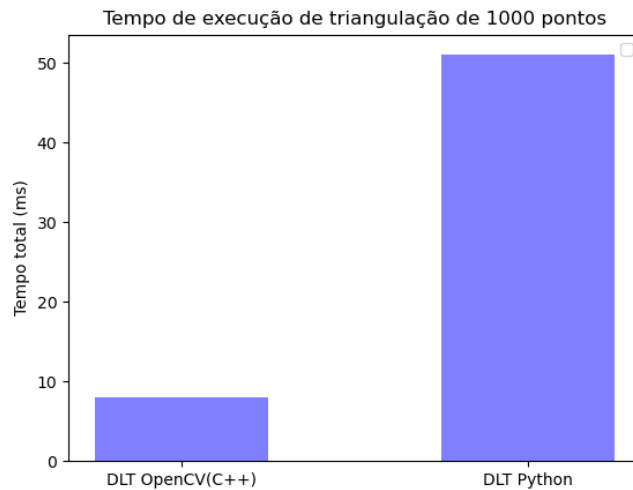


Figura 25: Comparativo entre performance das linguagens de implementações do DLT

## 4 Experimentos

### a Calibração extrínseca

Como primeiro passo para execução do algoritmo, foram obtidas as imagens para a realização da calibração extrínseca das câmeras. Visando capturar suas posições relativas ao padrão de calibração (*checkerboard*). Assim que as imagens são destorcidas e os padrões de calibração reconhecidos pode-se desenhar os eixos 3D afim de obter as seguintes imagens:



Figura 26: visão do palco da câmera 1

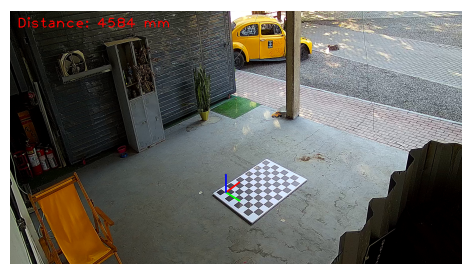


Figura 27: visão do palco da câmera 2



Figura 28: visão do palco da câmera 3

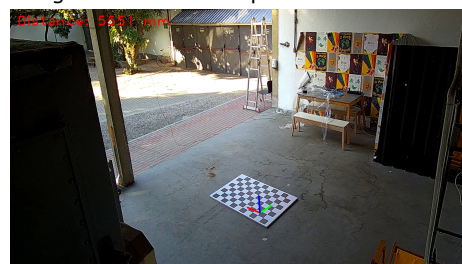


Figura 29: visão do palco da câmera 4



Figura 30: visão do palco da câmera 5

Sendo assim consegue-se estimar as posições das câmeras no mundo real, através do MATLAB [36].

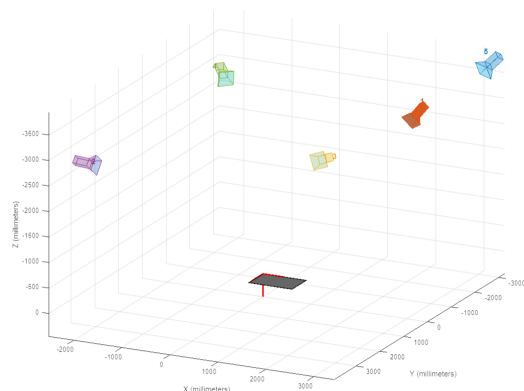


Figura 31: visão isométrica do palco em 3D

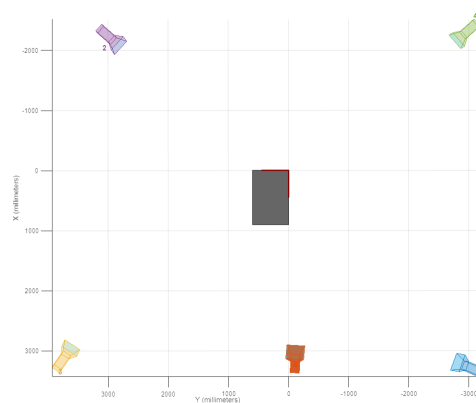


Figura 32: visão de topo do palco em 3D

## b Resultado Qualitativo (Experimento)

Resultados da inferência do modelo Media Pipe para reconhecimento dos esqueletos 2D, com pontuações de visibilidade acima de 50%, *frame a frame* e a respectiva reconstrução 3D estimada:

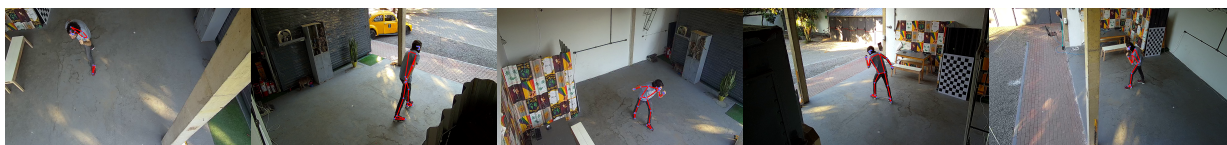


Figura 33: Mosaico cena 1 com sobreposição de estimativa de pose 2D.

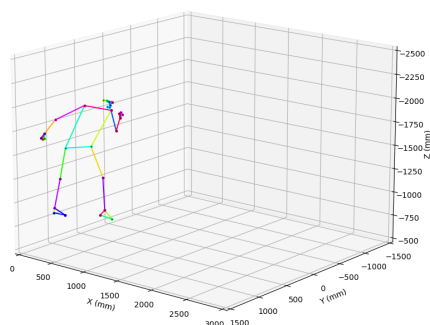


Figura 34: Resultados da triangulação de pose da cena 1, estimativa de pose em 3D.

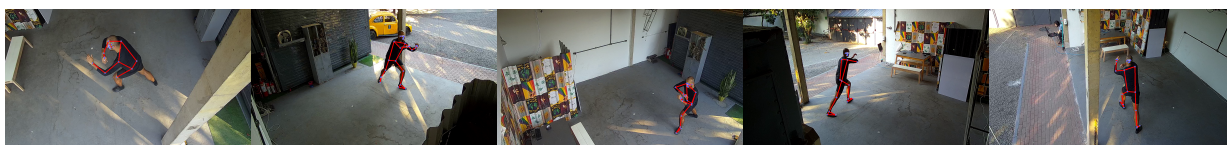


Figura 35: Mosaico cena 2 com sobreposição de estimativa de pose 2D.

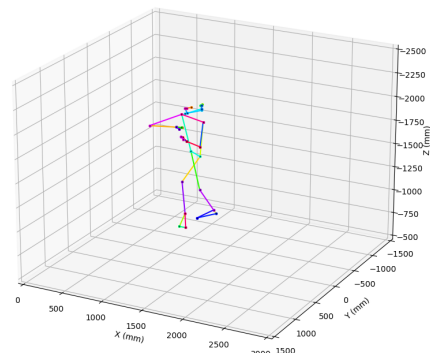


Figura 36: Resultados da triangulação de pose da cena 2, estimativa de pose em 3D.

Nas imagens acima temos exemplos de detecções e reconstruções bem-sucedidas onde o esqueleto é captado por todas as câmeras e a triangulação resulta em poses fidedignas e qualitativamente corretas.



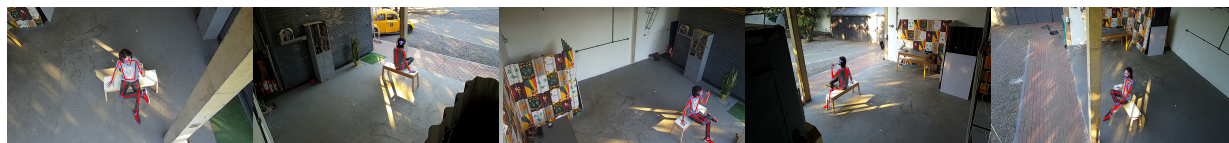


Figura 37: Mosaico cena 3 com sobreposição de estimativa de pose 2D.

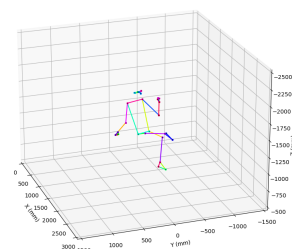


Figura 38: Resultados da triangulação de pose da cena 3, estimativa de pose em 3D.

Os bons resultados acima são atribuídos, principalmente, a uma boa pontuação de visibilidade da maioria dos pontos de juntas em duas ou mais perspectivas, que consequentemente, possibilita em uma melhor qualidade de triangulação e proximidade com a realidade da cena.

**Obteve-se também cenas onde o reconhecimento não foi bem sucedido, seguem abaixo:**

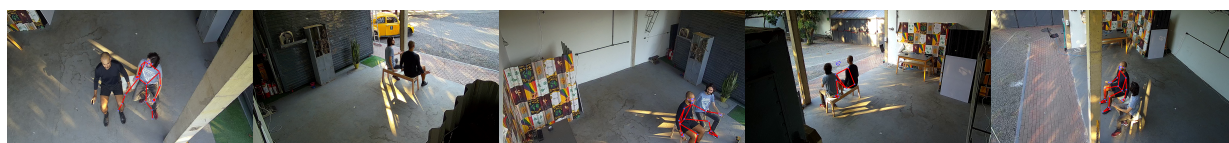


Figura 39: Mosaico cena 4 com sobreposição de estimativa de pose 2D.

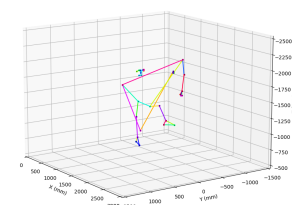


Figura 40: Resultados da triangulação de pose da cena 4, estimativa de pose em 3D.

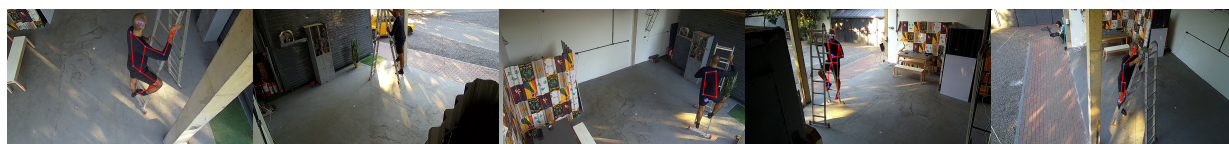


Figura 41: Mosaico cena 5 com sobreposição de estimativa de pose 2D.

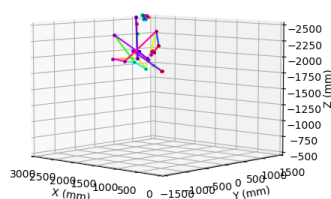


Figura 42: Resultados da triangulação de pose da cena 5, estimativa de pose em 3D.

Pose se perceber que, como indicado nas limitações do modelo do MediaPipe Pose, para mais de uma pessoa, o sistema não consegue diferenciar de forma consistente as juntas referente a cada pessoa, conforme na figura [40], assim como, uma dificuldade de reconhecimento de pose, quando objetos que formam oclusões e estruturas complexas estão muito próximos e interagindo diretamente a uma figura humana, indicado na figura [42].

Vale ressaltar que segundo a análise do Google sobre o modelo do BlazePose, utilizado na sua solução MediaPipe [40], existe uma discrepância no porcentual de acerto da inferência dado o tom de pele da pessoa de interesse em questão. Havendo uma diferença de até 4.8% de acerto total entre etnias da África Central 89.2% (acerto menor global) para a África do Sul com 94% (acerto maior global).

## 1 Resultado Temporal - FPS

Examinamos o cálculo teórico da latência envolvida na aquisição de vários *feeds* de vídeo RTSP (*Real-Time Streaming Protocol*). O sistema em consideração consiste em cinco *feeds* de vídeo RTSP, cada um com resolução de 1920x1080 *pixels* e codificado usando o *codec* H.264, que requer aproximadamente 5 Mbps de largura de banda para uma taxa de quadros de 25 fps.

Os *feeds* de vídeo são adquiridos em um laptop para jogos com processador Intel i5 de 11ª geração, com velocidade de rede de 100 MBps. Em nossa análise, consideramos vários fatores que podem afetar o tempo necessário para a aquisição do vídeo, incluindo atraso na transmissão, atraso na propagação, atraso na fila e atraso no processamento. Nosso objetivo é fornecer uma estimativa conservadora do tempo médio de aquisição, reconhecendo que as condições do mundo real podem apresentar variabilidade. Nossas descobertas fornecem informações sobre o desempenho potencial de tal sistema e podem informar as considerações de projeto de rede e sistema.

**Atraso de propagação:** Isso pode variar muito dependendo da distância entre o servidor e o cliente, bem como do meio (fibra, cobre, *wireless* etc). Uma regra prática comum é assumir um atraso de cerca de 1ms para cada 100km de cabo de fibra ótica. No entanto podemos assumir que esse valor é negligenciável dado que no experimento prático temos uma distância  $\leq 1km$ .

**Atraso na fila:** Isso depende do congestionamento da rede, mas pode facilmente adicionar vários milissegundos de atraso durante os momentos de pico de uso de banda da rede.

**Atraso de processamento:** A decodificação de vídeo H.264 não é muito exigente, mas ainda requer algum poder computacional. Em um laptop i5 de 11ª geração, isso deve ser bastante rápido, mas vamos supor que possa adicionar 5 a 10ms adicionais por *feed* de vídeo.

**Sobrecarga de rede:** RTP, RTCP, RTSP, IP, UDP e *Ethernet* adicionam seus próprios cabeçalhos a cada pacote. Esses dados indiretos precisam ser transmitidos e processados, o que adiciona um atraso adicional.

**Atraso de transmissão:** É o tempo que um bit leva para viajar do remetente ao destinatário.

Então, vamos calcular os fatores mencionados:

$$\text{Atraso de transmissão} = \frac{\text{Quantidade de dados (em bits)}}{\text{Largura de banda (em bits por segundo)}}$$

Para um único quadro do vídeo:

$$\text{Tamanho de um quadro} = \frac{5Mbps}{25fps} = 200kilobits$$

$$\text{Atraso de transmissão} = \frac{200.000bits}{800.000.000bits} = 0,00025s$$

Assuma um atraso de transmissão para um quadro calculado de 0,25ms.

Assuma um atraso de propagação negligenciável.

Assuma um atraso de fila de 10ms (esta é uma estimativa aproximada).

Assuma um atraso de processamento de 5ms por *feed* de vídeo, portanto, para cinco *feeds*, seria de 25ms.

Some tudo isso e obtém-se um valor médio, conservador, para a captura de um *frame* de cinco vídeos:

$$0,25ms + 10ms + 25ms = 35,25ms$$

Vale ressaltar que esse processamento acontece em regime *multi-thread* o que implica que esse tempo gasto não trava o processador enquanto se espera o recebimento dos dados, o deixando livre para realizar outras funções durante dito tempo.

Sendo assim podemos seguir com nosso estudo e gerar a tabela abaixo com os tempos médios medidos em software durante a prática experimental do sistema:

| Tabela de tempos médios de processamento |                  |
|--|------------------|
| Processo                                 | Resultado (ms)   |
| Aquisição                                | 35.25            |
| Distorção                                | 23.00            |
| Inferência 2D                            | 30.42            |
| Triangulação (DLT)                       | 1.62             |
| Geração de gráfico 3D                    | 54.25            |
| <b>TOTAL:</b>                            | <b>144.54 ms</b> |

Tabela 2: Tabela de tempos médios para cada estágio do *pipeline* de processamento. Avaliado em laptop padrão (Intel i5-10400H - 2.60GHz 11th, 8gb RAM, GTX 1650 4GB)

Com um tempo total de processamento de  $144,54ms$ , chegamos a uma métrica de:

$$6,92fps$$

O que corresponde a realidade prática do experimento.

## c Dataset e Métricas de Avaliação

Na seção a seguir, apresentamos a aplicação do Erro Posicional Médio por Articulação (MPJPE - *Mean Per Joint Position Error*) como uma métrica quantitativa para o processo de avaliação em nosso estudo. MPJPE é uma métrica de avaliação padrão para tarefas de estimativa de pose humana que fornece uma medida abrangente do desempenho do modelo. O cálculo do MPJPE envolve a determinação da distância entre a posição real (*Ground Truth*) e as coordenadas 3D estimadas de cada articulação e, em seguida, a média dessas distâncias em todas as articulações para uma determinada pose. Posteriormente, esse processo é repetido para todos os quadros na sequência de vídeo e os resultados são calculados novamente para fornecer o MPJPE final.

A pontuação geral é dada em milímetros, com uma pontuação mais baixa indicando uma estimativa de pose mais precisa. Essa abordagem fornece um mecanismo robusto, eficiente e matematicamente sólido para comparar diretamente o desempenho de diferentes algoritmos ou sistemas. Além disso, o MPJPE permite uma avaliação clara e transparente, fornecendo uma compreensão comparativa da eficiência de várias abordagens no tratamento do problema de estimativa de pose.

Utiliza-se o *dataset* Panoptic para comparação do modelo criado nesse *paper* com a informação correta das poses (*ground truth*) fornecido pelo *dataset*.

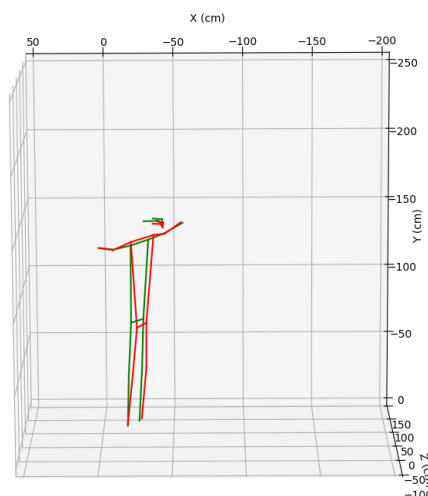


Figura 43: Estimativa de pose de um frame, em vermelho, comparada ao *ground truth* do Panoptic, em verde.

## d Comparação com o Estado da Arte

Nos últimos anos, a estimativa de pose humana 3D teve avanços notáveis, com várias metodologias de ponta ultrapassando os limites de precisão e confiabilidade. Como *TesseTrack*, *MvP* e *Voxel Pose*, entre outros, estabeleceram referências formidáveis, as usamos como comparativos na tabela a seguir:

| Dataset - CMU Panoptic   |               |               |
|--------------------------|---------------|---------------|
| Autor                    | MPJPE (em mm) | Tempo (em ms) |
| TesseTack [15]           | 7.3           | Não informado |
| MvP [1]                  | 15.8          | 278.8         |
| VoxelPose [14]           | 17.7          | 316.0         |
| FasterVoxelPose [10]     | 18.3          | 32.2          |
| <b>Nosso (4 câmeras)</b> | <b>50.8</b>   | <b>55.0</b>   |
| <b>Nosso (6 câmeras)</b> | <b>49.7</b>   | <b>58.4</b>   |

Tabela 3: Comparação com os métodos estado da arte no *dataset* Panoptic. A métrica MPJPE foi retirado de [9] e o tempo, retirado de [10]

A tabela apresenta também um métrica de tempo de execução para a inferência de uma pose 3D, desta forma, para refletir somente o tempo gasto para processar as imagens de entrada e gerar uma pose de saída, nosso tempos estão indicados sem o tempo de atraso de aquisição de imagens e de geração de gráfico 3D. Vale evidenciar que, os tempos da *MvP*, *VoxelPose* e *FasterVoxelPose* foram executados em um hardware diferente ao nosso, com GPU GeForce RTX 2080 Ti e CPU Intel(R) Xeon(R) CPU E5-2699A v4 @ 2.40GHz conforme indicado em [10].

Este relatório apresenta uma comparação abrangente do nosso trabalho com essas metodologias pioneiras, fornecendo informações sobre as compensações de desempenho quando o custo se torna um fator significativo. Essa avaliação honesta serve como um lembrete importante de que, embora nossa solução possa não oferecer a mais alta precisão, ela representa uma alternativa viável para cenários em que as limitações orçamentárias são uma consideração importante.

Nosso projeto propõe, não com uma abordagem inovadora, mas com uma abordagem mais econômica. Devido às nossas restrições orçamentárias, tomamos uma decisão consciente de priorizar a relação custo-benefício e velocidade sobre a acurácia. Como resultado, nossos resultados atuais não superam os das soluções de ponta, o que era um resultado esperado, dado o escopo, entretanto ele possibilita ao usuário uma resposta em tempo real sobre a geração e qualidade das poses.

## 5 Discussão

### a Trabalhos relacionados

Como mencionamos na introdução deste documento, estudos correlatos a este projeto já anteciparam e enfrentaram os desafios que aqui relatamos. Em comparação com as abordagens mais recentes, nosso método pode parecer obsoleto, devido à nossa estratégia de reconstrução de cena para detectar os esqueletos e em seguida, triangular sua posição. Apesar de fazer sentido do ponto de vista lógico, essa abordagem tem custos computacionais elevados.

Os trabalhos mais recentes abordam a mesma problemática utilizando técnicas matemáticas mais elaboradas e sofisticadas, que ultrapassam o escopo deste projeto de graduação.

No entanto, quando comparado com trabalhos mais antigos, oferecemos uma solução robusta e bem implementada, que funciona efetivamente na prática e pode ser usada para demonstrações. Mesmo que não seja a implementação de maior desempenho.

### b Limitações e Possíveis Melhorias

Conforme explicado anteriormente, devido a limitações de orçamento tivemos que escolher por opções mais custo-benefício para as definições do hardware utilizado. As câmeras de segurança trazem a praticidade que precisamos para fisicamente montar o sistema que precisamos (comunicação em rede), contudo trazem desvantagens grandes em relação a suas competidoras:

#### 1 fps

Nosso objetivo ideal é atingir a maior taxa de quadros por segundo (fps) possível, e garantir que o processo de captura de imagem seja o mais rápido possível. Isso nos permitirá obter um grande número de *frames* de informação, minimizando a perda de movimentos devido a imagens borradas. A câmera que estamos utilizando para os testes, no entanto, oferece um máximo de 25 fps. Isso tem um impacto negativo significativo no resultado final da detecção para movimentos rápidos ou bruscos. Idealmente o sistema operaria com um frame rate maior que 120 fps.

#### 2 Obturador / *Shutter*

A aquisição de uma imagem por um sensor digital (CCD/CMOS) pode ser descrito como o processo pelo qual o microcontrolador processa os pixels das imagens ao longo do tempo. As câmeras que estamos utilizando empregam um sistema de obturador conhecido como *rolling shutter*. Isso significa que o dispositivo lê a imagem linha por linha, escaneando o sensor de maneira sequencial, do primeiro pixel até o último, fazendo uma varredura em toda extensão do sensor.

No entanto, esse método introduz um intervalo de tempo  $\Delta t$  entre o início e o fim da varredura. Isso pode ser problemático ao tentarmos capturar movimentos rápidos, pois um único *frame* pode conter informações de diferentes momentos no tempo. O resultado são imagens borradas e de difícil interpretação pelas partes subsequentes do algoritmo.

Idealmente, buscaríamos um sensor com um obturador global. Conforme o nome sugere, esse tipo de obturador lê as informações de todos os pixels do sensor simultaneamente, eliminando a diferença de tempo entre o início e o fim da varredura.

#### 3 Processamento

Estamos utilizando uma câmera que não realiza processamento local, uma característica comum às câmeras utilizadas no mercado para tarefas similares. Isso impõe uma limitação ao nosso sistema no que diz respeito à largura de banda e ao poder de processamento que nosso computador central precisa administrar, o que acarreta impactos negativos na performance e na escalabilidade do sistema.

Para cada câmera adicionada ao sistema de captura, é necessário transmitir um fluxo de vídeo de 2K, o que corresponde a cerca de 12 Mbps de vídeo comprimido. Isso ocorre porque a câmera não possui a opção de enviar vídeo não comprimido pela rede.

Além disso, é necessário executar uma nova instância do adquiridor de *frames*, detector de pose e do algoritmo DLT para cada unidade adicional de câmera implementada no projeto.

Como medida para contrabalancear esses desafios, processaríamos localmente em cada câmera a detecção da pose em 2D, e enviaríamos as posições processadas para o computador central, assim como um *feed* de vídeo comprimido de baixa resolução só para o operador ter um *feedback* visual do que está acontecendo, mas sem necessariamente impactar na performance do sistema como um todo.

#### 4 Sincronização de Hardware

Conforme discutido na seção [3] deste documento, empregamos o protocolo RTSP (*Real Time Streaming Protocol*) para a aquisição de imagens dos *frames* das câmeras. No entanto, isso se revelou um desafio significativo no fluxo de desenvolvimento, pois o protocolo não assegura que o último *frame* solicitado pelo programa esteja sincronizado com todos os outros *frames* das  $n - 1$  câmeras em uso.

Isso resulta em um problema para o algoritmo DLT, que precisa correlacionar poses que podem ter pequenas variações temporais entre si. Esta questão impacta negativamente a performance do nosso protótipo.

Propomos que as câmeras, ou melhor, que os módulos de aquisição se tornem interconectados por um sinal de hardware de gatilho do processo para garantir a aquisição simultânea dos *frames* de todas as unidades em uso.



## 6 Conclusão

Neste relatório técnico, identificamos vários desafios e possíveis melhorias em nosso sistema de hardware para estimativa e rastreamento de poses 3D. Nossa configuração de câmera de segurança atual, embora econômica, luta para capturar movimentos rápidos devido a limitações na taxa de quadros, ao uso de um mecanismo de *rolling shutter* e à ausência de recursos de processamento local. Essas restrições geram resultados inferiores em termos de desfoque, escalabilidade e sincronização e podem ser mitigadas com a adoção de câmeras com uma taxa de quadros mais alta, sistemas de obturador global e recursos de detecção de pose local. Tais limitações críticas em nosso sistema de hardware prejudicam nossa capacidade de estimativa de pose 3D de forma robusta. Dadas as nossas restrições orçamentárias.

Propusemos a execução de uma abordagem básica para estimativa de pose 3D que demonstra os complexos problemas de associação no espaço 2D. Nosso método aproveita as informações ruidosas e incompletas de todas as visualizações da câmera e as reconstrói em um espaço 3D compartilhado, gerando um volume de recursos abrangente empregado para estimativa 3D. As validações experimentais comprovam isso, mostrando-se promissoras em nossa abordagem.

No entanto, para melhorar ainda mais o desempenho do nosso sistema e fornecer estimativas de pose 3D de alta qualidade, os aprimoramentos de hardware são indispensáveis. Tanto no quesito de um computador mais potente para treinamento do modelo quanto para a inferência.

No geral, a integração dessas possíveis melhorias de hardware com nossa nova abordagem oferece perspectivas promissoras para obter estimativas de pose 3D confiável e de alta precisão ainda oferecendo uma solução de baixo custo em comparação com o mercado atual.

## Referências

- [1] T. Wang, J. Zhang, Y. Cai, S. Yan, and J. Feng, "Direct multi-view multi-person 3d pose estimation," *35th Conference on Neural Information Processing Systems (NeurIPS)*, 2021.
- [2] MathWorks, "Camera calibration," 2023, accessed: 2023-05-30. [Online]. Available: <https://www.mathworks.com/help/vision/ug/camera-calibration.html>
- [3] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "Mediapipe: A framework for building perception pipelines," 2019.
- [4] G. Developers, "Pose landmark detection guide," 2023, [Online; accessed 1-June-2023]. [Online]. Available: [https://developers.google.com/mediapipe/solutions/vision/pose\\_landmarker](https://developers.google.com/mediapipe/solutions/vision/pose_landmarker)
- [5] J. Meza, R. Vargas, L. A. Romero, S. Zhang, and A. G. Marugo, "What is the best triangulation approach for a structured light system," in *Dimensional Optical Metrology and Inspection for Practical Applications IX*, vol. 11397. SPIE, May 2020, pp. 65–74, <https://ui.adsabs.harvard.edu/abs/2020SPIE11397E..0DM/abstract> ; <https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11397/113970D/What-is-the-best-triangulation-approach-for-a-structured-light/10.1117/12.2559119.full> ; <https://repositorio.utb.edu.co/handle/20.500.12585/9538>. [Online]. Available: <https://lens.org/157-613-314-917-512>
- [6] H. Joo, T. Simon, X. Li, H. Liu, L. Tan, L. Gui, S. Banerjee, T. S. Godisart, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh, "Panoptic studio: A massively multiview system for social interaction capture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
- [7] OptiTrack, "Prime 13 - depth camera," Online, Accessed 2023. [Online]. Available: <https://optitrack.com/cameras/prime-13/indepth.html>
- [8] Luxonis, "Luxonis OAK-D S2 POE," Online, Accessed 2023. [Online]. Available: <https://shop.luxonis.com/collections/oak-cameras-1/products/oak-d-s2-poe?variant=43146876551391>
- [9] P. W. Code, "3d multi-person pose estimation on cmu panoptic," 2023, accessed: 2023-06-24. [Online]. Available: <https://paperswithcode.com/sota/3d-multi-person-pose-estimation-on-cmu>
- [10] H. Ye, W. Zhu, C. Wang, R. Wu, and Y. Wang, "Faster voxelpose: Real-time 3d human pose estimation by orthographic projection," in *European Conference on Computer Vision (ECCV)*, 2022.
- [11] S. Yang, F. Ding, P. Li, and S. Hu, "Distributed multi-camera multi-target association for real-time tracking," *Scientific Reports*, vol. 12, no. 1, jun 2022. [Online]. Available: <https://doi.org/10.1038/s41598-022-15000-4>
- [12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018.
- [13] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," 2018.
- [14] H. Tu, C. Wang, and W. Zeng, "Voxelpose: Towards multi-camera 3d human pose estimation in wild environment," 2020.
- [15] N. D. Reddy, L. Guigues, L. Pishchulin, J. Eledath, and S. G. Narasimhan, "Tesseract: End-to-end learnable multi-person articulated 3d pose tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 15 190–15 200.
- [16] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1325–1339, jul 2014.
- [17] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [18] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003.
- [19] T. Vision, "Camera modeling: Exploring distortion and distortion models, part i," 2023, accessed: 2023-05-30. [Online]. Available: <https://www.tangramvision.com/blog/camera-modeling-exploring-distortion-and-distortion-models-part-i>
- [20] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.



- [21] Wikipedia, "Distortion (optics)," 2023, accessed: 2023-05-30. [Online]. Available: [https://en.wikipedia.org/wiki/Distortion\\_%28optics%29#Calibrated](https://en.wikipedia.org/wiki/Distortion_%28optics%29#Calibrated)
- [22] J. Ltd., "Five seidel aberrations | glossary," 2023, accessed: 2023-05-30. [Online]. Available: <https://www.jeol.com/words/emterms/20121023.035259.php#gsc.tab=0>
- [23] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "Blazepose: On-device real-time body pose tracking," *CoRR*, vol. abs/2006.10204, 2020, [Accessed 31-May-2023]. [Online]. Available: <https://arxiv.org/abs/2006.10204>
- [24] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic, "3d pictorial structures for multiple human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
- [25] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004, cap. 12.2, pag. 312.
- [26] —, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004, cap. 12.1, pag. 311.
- [27] —, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004, cap. 12.3, pag. 314.
- [28] —, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004, appendix 4, pag. 585.
- [29] G. v. Rossum, "Python," Python Software Foundation, 2021, python version 3.10, Accessed: 2023-06-01. [Online]. Available: <https://www.python.org/>
- [30] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [31] *concurrent.futures — Launching parallel task*, Python Software Foundation, oct 2021, python version 3.10, Accessed: 2023-06-01. [Online]. Available: <https://docs.python.org/3.10/library/concurrent.futures.html?highlight=threadpool#threadpoolexecutor>
- [32] A. Rao and R. Lanphier, "Real Time Streaming Protocol(RTSP)," Internet Engineering Task Force, Internet-Draft draft-rao-rtsp-00, Oct. 1996, work in Progress. [Online]. Available: <https://datatracker.ietf.org/doc/draft-rao-rtsp/00/>
- [33] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *CoRR*, vol. abs/1812.08008, 2018. [Online]. Available: <http://arxiv.org/abs/1812.08008>
- [34] Y. Zhang, L. An, T. Yu, X. Li, K. Li, and Y. Liu, "4d association graph for realtime multi-person motion capture using multiple video cameras," *CoRR*, vol. abs/2002.12625, 2020. [Online]. Available: <https://arxiv.org/abs/2002.12625>
- [35] S. K. Gupta, "3d human pose estimation using multi camera," Aug. 2020. [Online]. Available: <https://doi.org/10.5281/zenodo.4003521>
- [36] MathWorks, "Matlab," 2023, accessed: 2023-05-30. [Online]. Available: <https://www.mathworks.com/products/matlab.html>
- [37] —, "Computer vision toolbox - matlab," 2023, accessed: 2023-06-11. [Online]. Available: <https://www.mathworks.com/products/computer-vision.html>
- [38] "About : Standard c++," Standard C++ Foundation, 2023, accessed: 2023-06-11. [Online]. Available: <https://isocpp.org/about>
- [39] "About - opencv," OpenCV team, 2023, accessed: 2023-06-11. [Online]. Available: <https://opencv.org/about/>
- [40] A. Z. P. B. L. V. B. H. E. S. I. a. R. T. G. Margaret Mitchell, Simone Wu, "Model card mediapipe blazepose ghum 3d," 2019, [Accessed 27-jun-2023]. [Online]. Available: <https://storage.googleapis.com/mediapipe-assets/Model%20Card%20BlazePose%20GHUM%203D.pdf>

## A Apêndice

Segue abaixo os códigos mencionados durante o projeto.

### a Camera Calibrator

O Objeto *CameraParams*, o comando e a saída da conversão para OpenCV, feitos no MATLAB:

---

```

1
2 cameraParams =
3     cameraParameters with properties:
4
5     Camera Intrinsics
6             Intrinsics: [1×1 cameraIntrinsics]
7
8     Camera Extrinsics
9             PatternExtrinsics: [18×1 rigidtransform3d]
10
11    Accuracy of Estimation
12            MeanReprojectionError: 0.1186
13            ReprojectionErrors: [176×2×18 double]
14            ReprojectedPoints: [176×2×18 double]
15
16    Calibration Settings
17            NumPatterns: 18
18            DetectedKeypoints: [176×18 logical]
19            WorldPoints: [176×2 double]
20            WorldUnits: 'millimeters'
21            EstimateSkew: 0
22            NumRadialDistortionCoefficients: 3
23            EstimateTangentialDistortion: 0
24
25 >> [intrinsicMatrix,distortionCoefficients] = cameraIntrinsicsToOpenCV(cameraParams)
26
27 intrinsicMatrix =
28     1.0e+03 *
29
30     1.5041         0     0.9704
31         0     1.5071     0.5090
32         0         0     0.0010
33
34 distortionCoefficients =
35     -0.3929     0.1785         0         0     -0.0509
36
37

```

---

## b Código de cada *thread*

Função que cada *thread* roda continuamente, em segundo plano, para aquisição e processamento dos quadros de uma câmera (por simplicidade foram retirados parâmetros e lógicas de avaliação da performance)

---

```

1
2 def process_feed(feed, frame_width, frame_height, frame_limits, pose_2d, index, distorted_feed,
3     dist_cam_mat, dist_coeffs, undistorted_cam_mat, roi):
4
5     # Global variables
6     global mosaic
7     global pose_output_buffer
8     global stop_event
9     global thread_lock
10
11     # Get the location in the mosaic to place the frame
12     y1, y2, x1, x2 = frame_limits
13
14     while not stop_event.is_set(): # Loop over the frames from one camera
15
16         # Read a frame from the camera
17         ret, frame = feed.read()
18
19         if not ret:
20             continue # skip to the next frame
21
22         if distorted_feed:
23             # Undistort the frame
24             frame = cv2.undistort(frame, dist_cam_mat, dist_coeffs, None, undistorted_cam_mat)
25
26         # Process MediaPipe-Pose backend
27         pose_frame, landmarks_list = pose_2d.calc_2d_kpt(frame)
28
29         # Resize the frame to match the size of the visualization mosaic
30         pose_frame = cv2.resize(pose_frame, (frame_width, frame_height), interpolation=cv2.INTER_NEAREST)
31
32         with thread_lock:
33
34             # Update the pose list of 2d points
35             pose_output_buffer[index] = landmarks_list
36
37             # Update the mosaic with the frame overlayed with 2d joints
38             mosaic[y1:y2, x1:x2] = pose_frame
39
40         # Release the video capture object for this camera feed
41         feed.release()
42

```

---

### c Código de reconstrução do esqueleto

Funções dentro de classe Pose3D, que realizam o processo de triangularização (DLT) e a lógica que obtém o melhor esqueleto com base na pontuação de visualização (foi omitida a inicialização da classe onde são recebidas parâmetros das câmeras como a matriz de projeção).

---

```

1
2 def cv2_3d_pose_pair(self, pair, poses):
3     i, j = pair
4     points1 = np.array([d['pixel_xy'] for d in poses[i]].T
5     points2 = np.array([d['pixel_xy'] for d in poses[j]].T
6
7     # Triangulate (DLT) all points of a pair of poses
8     homog_pose_points_3d = cv2.triangulatePoints(self.P_mats[i], self.P_mats[j], points1, points2)
9     # Homogeneous points are returned
10    euclid_pose_points_3d = cv2.convertPointsFromHomogeneous(np.transpose(homog_pose_points_3d))
11
12    return euclid_pose_points_3d
13
14
15 def calc_best_3d_kpt(self, all_2d_poses):
16    all_3d_poses = []
17    all_min_vis = []
18    for i, j in self.pairs:
19        # triangulate 3d points per camera pair
20        pose_3d_pair = self.cv2_3d_pose_pair(pair=[i, j], poses=all_2d_poses)
21        # list o minimum visibility of the pair in each joint
22        min_visibility_pair = [min(joint1['visibility'], joint2['visibility'])
23                               for joint1, joint2 in zip(all_2d_poses[i], all_2d_poses[j])]
24        all_min_vis.append(min_visibility_pair)
25        all_3d_poses.append(pose_3d_pair)
26
27    # Finding array/pose index with best visisibility in all of all_min_vis list
28    best_pair_idx = np.array(all_min_vis).argmax(axis=0)
29    # Best 3d pose
30    best_3d_pose = [all_3d_poses[pair_idx][joint_idx, 0]
31                    for joint_idx, pair_idx in enumerate(best_pair_idx)]
32    return np.array(best_3d_pose)
33

```

---