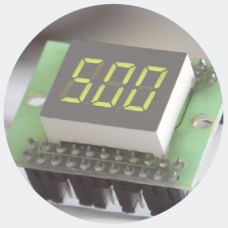
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






3D Pose and Shape Estimation from a Camera System

ANDRÉ MOREIRA DE OLIVEIRA
outubro de 2023

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

3D Pose and Shape Estimation from a Camera System

André Moreira de Oliveira

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Sistemas Autónomos

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Outubro, 2023

Esta página foi propositadamente deixada em branco

Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Sistemas Autónomos.

Candidato: André Moreira de Oliveira, Nº 1181045, 1181045@isep.ipp.pt

Orientação Científica: Lino Figueiredo, lbf@isep.ipp.pt

Coorientação Científica: Carlos Ramos, csr@isep.ipp.pt

Empresa: New Sign Solutions, Lda.

Orientador: José Ribeiro, jcribeiro@nss.com.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Outubro, 2023

Esta página foi propositadamente deixada em branco

"O que falta fazer? Eu digo tudo. O quer que seja que a gente faça, do outro lado, está sempre o infinito."

José Tenreiro Machado, 1957-2021

Esta página foi propositadamente deixada em branco

Agradecimentos

O presente projeto ilustra o culminar de um caminho realizado nos últimos anos, que contou com a contribuição, incentivo e dedicação de vários intervenientes que seguramente, no espaço limitado desta secção de agradecimentos, não me permitirá fazê-lo devidamente. Desta forma, não menosprezando algum contributo, deixo algumas palavras, com profundo sentimento de reconhecido agradecimento.

Gostaria de expressar meus sinceros agradecimentos ao orientador deste projeto, Professor Engenheiro Lino Figueiredo, pela disponibilidade constante e acompanhamento exemplar durante a evolução de todo o trabalho.

Ao meu coorientador, Professor Engenheiro Carlos Ramos, pela perspetiva complementar no enriquecimento do trabalho definida por uma visão aprofundada sobre os diferentes temas de inteligência artificial.

Um agradecimento especial à equipa da NSS, em particular ao meu orientador da empresa, Engenheiro José Ribeiro, pela partilha de experiência e cooperação constante com ideias e perspetivas complementares.

A todos os meus colegas de curso, que de forma solidária contribuíram para o sucesso mútuo.

Não posso deixar de expressar os meus mais profundos agradecimentos à minha família, nomeadamente à minha Mãe, ao meu Pai e à minha Irmã pela criação de todas as possibilidades que me permitiram finalizar esta fase e se viram frequentemente privados da minha atenção.

À minha namorada, que esteve sempre presente, apoiando-me incansavelmente e incondicionalmente mesmo nos momentos de maiores sacrifícios, fazendo-me acreditar sempre em mim. Por me ensinar o significado da amizade e companheirismo.

A todos, o meu eterno OBRIGADO!

Esta página foi propositadamente deixada em branco

Resumo

Neste trabalho, é abordada uma solução que procura estimar a posição articular 3D de várias pessoas em cenários reais, bem como a sua forma corporal e trajetória global a partir de um único vídeo RGB, gravado com uma câmara estática ou dinâmica.

Em contraste com sistemas *multi-view* complexos, esta solução prioriza a simplicidade e adaptabilidade em diferentes aplicações. Face ao cenário desafiador, desenvolveu-se um sistema baseado em diferentes *frameworks*, individualmente otimizadas para o seu propósito. Como tal, o autor procurou estender o processo realizado num *pose and shape estimator* convencional, implementando, de forma robusta, a capacidade *tracking* de humanos e uma inferência com base em coerência temporal, capaz de lidar com oclusões completas em longos intervalos de tempo.

Os humanos, presentes no cenário, são detetados e devidamente identificados ao longo do vídeo, a partir de um *Multiple Person Tracking* (MPT) (*i.e.*, Deep OC-SORT com YOLOv8x e *Re-Identication (Re-ID) model*). Esta informação, alimenta o *Human Pose and Shape (HPS) estimator* (*i.e.*, HybrIK com *backbone* da rede HRNet-W48) capaz de gerar, a partir de uma combinação da representação volumétrica das articulações com a capacidade de extração de *features* das DCNNs, uma sequência que define o movimento do humano no sistema de coordenadas da câmara (*i.e.*, *root translations*, *root rotations*, pose do corpo e os parâmetros do *shape*).

Complementarmente, o movimento humano, localmente definido, é preenchido segundo um processo iterativo, dado pela integração do *generative motion optimizer*, por sua vez organizado numa arquitetura baseada em *Transformers* e apoiado pelas relações temporais presentes na informação das deteções visíveis. Para um conjunto de parâmetros descritivos do movimento corporal de cada humano é obtido a respetiva trajetória global, propriamente relacionadas, num processo baseado na variação posicional local (posição no plano e orientação) e numa otimização iterativa dos parâmetros da câmara consistente com as evidências do vídeo, *e.g.*, *2D keypoints*.

Os resultados, obtidos no *dataset* 3DPW, demonstram que a abordagem proposta superar os métodos anteriores na reconstrução do movimento, com $68,2$ *mm* PA-MPJPE em oclusões e $46,4$ *mm* PA-MPJPE em poses visíveis.

Palavras-Chave: *3D Human pose and shape estimation*, *Camera parameters optimization*, *Deep learning*, *Global pose estimation*, *Multi-person motion reconstruction*, *Multi-person tracking*, *Occlusion-aware pose estimation*, *Transformer-based*.

Esta página foi propositadamente deixada em branco

Abstract

In this work, a solution is addressed that try to estimate the 3D joint position of several people in in-the-wild scenes, as well as their body shape and global trajectory from a single RGB video, recorded with a static or dynamic camera.

In contrast to complex multi-view systems, this solution prioritizes simplicity and adaptability in different applications. Faced with the challenging scenario, a system was developed based on different frameworks, individually optimized for their purpose. As such, the author sought to extend the process carried out in a conventional pose and shape estimator, robustly implementing the tracking capability of humans and an inference based on temporal coherence, capable of dealing with complete occlusions over long time intervals.

The humans, present in the scene, are detected and duly identified throughout the video using an *Multiple Person Tracking* (MPT) (i.e., Deep OC-SORT with YOLOv8x and Re-ID model). This information is fed into the HPS estimator (i.e., HybrIK with backbone from the HRNet-W48 network), which is able to generate, from a combination of the volumetric representation of the joints and the ability to extract features from the DCNNs, a sequence that defines the body motion of the human in the camera's coordinate system (i.e., root translations, root rotations, body pose and shape parameters).

In addition, the body motion, locally defined, is filled according to an iterative process, given by the integration of the generative motion optimizer, in turn organized in an architecture based on Transformers and supported by the temporal relationships present in the information of the visible detections. For a set of parameters describing the body motion of each human, the respective global trajectory is obtained, properly related, in a process based on local positional variation (position in the plane and orientation) and an iterative optimization of the camera parameters consistent with the video evidence, *e.g.*, *2D keypoints*.

The results, obtained in the 3DPW *dataset*, show that the proposed approach outperforms previous methods in motion reconstruction, with 68.2 *mm* PA-MPJPE in occlusions and 46.4 *mm* PA-MPJPE in visible poses.

Keywords: *3D Human pose and shape estimation*, *Camera parameters optimization*, *Deep learning*, *Global pose estimation*, *Multi-person motion reconstruction*, *Multi-person tracking*, *Occlusion-aware pose estimation*, *Transformer-based*.

Esta página foi propositadamente deixada em branco

Índice

Lista de Figuras

Lista de Tabelas

Esta página foi propositadamente deixada em branco

Glossário

2D human pose estimation

Tarefa da visão computacional focada em determinar a localização das articulações do corpo humano numa espaço relativo a duas dimensões (2D), *i.e.*, X e Y. Os métodos propostos podem ser categorizados em *regression-based methods* e *heatmap-based methods* para *single person pose estimation* ou métodos *top-down* e *bottom-up* para *multi-person pose estimation*.

3D human pose estimation

Tarefa da visão computacional focada em determinar a localização das articulações do corpo humano numa espaço relativo a três dimensões (3D), *i.e.*, X, Y e Z. Os métodos propostos podem ser segmentados em abordagens *direct estimation* e *2D to 3D lifting* para *single view single person pose estimation* ou métodos *top-down* e *bottom-up* para *single view multi-person pose estimation*.

árvore cinemática humana

A árvore cinemática humana é uma representação matemática avançada utilizada para modelar a geometria e estrutura hierárquica das articulações do corpo humano, representando a ordem e a conectividade entre essas articulações (*connectivity map* com relações de parentesco entre articulações).

backbone

No contexto do projeto, o *backbone* constitui parte essencial de uma rede neuronal, concretamente está associada uma *feature extractor network*, previamente treinada para processamento da imagem de entrada e obtenção de diversas *features*. São alguns exemplos *VGGs*, *ResNets*, *DenseNet* entre outras.

batch size

Em *machine learning*, *batch size*, é considerado um hiperparâmetro que especifica o número de amostras que deverão ser usadas em cada iteração do treino para efeito de atualização dos pesos do modelo. Um número superior exigirá uma alocação de memória maior, enquanto um *batch size* inferior poderá evitar que modelo fique preso em mínimos locais.

bottom-up methods

Métodos explorados em *2D/3D multi-person Pose Estimation* (PE) constituinte de *multi-person* HMR. De forma genérica, preveem diretamente todas as articulações das pessoas presentes na imagem de entrada (*i.e.*, através da obtenção de *local features*) e, posteriormente, agrupa-as em "esqueletos" independentes (*i.e.*, através de algoritmos de associação).

coerência temporal

Coerência temporal ou consistência temporal refere-se à capacidade do sistema de manter a coerência entre as predições em diferentes instantes de tempo. Concretamente, num algoritmo de PE, espera-se que as articulações determinadas num determinado *frame* seja coerente com a determinação no próximo *frame* para que seja levada em consideração a dinâmica natural dos movimentos humanos e mantendo a estabilidade e robustez a variações dispensáveis.

egocentric representation

Representação alternativa da *global trajectory* durante o *Generative Motion Optimizer* para simplificação da resposta da rede neuronal *Global Trajectory Predictor Network* (GTP-Net) em longos intervalos de tempo. Efetivamente, permite a representar variações de trajetória local (rotação e translação) em *heading coordinates*, isto é, com uma representação do mundo centrada no humano em análise.

epoch

Em *machine learning*, *epoch*, é considerado um hiperparâmetro e remete para uma passagem completa do *dataset* de treino pelo algoritmo. Concretamente, um *epoch* ocorre quando todos os *batches* deste *dataset* forem utilizados, pelo menos uma vez, para atualizar o modelo.

heading coordinates

Sistema de coordenadas numa *egocentric representation* obtidas pela transformação do centro de coordenadas do mundo na *root position* da pessoa, rodando, posteriormente o "mundo" em torno do eixo z para alinhar o eixo y com a orientação facial do humano (*heading vector*).

heatmap-based methods

Métodos explorados em *2D single person* PE que procuram estimar a localização aproximada das partes do corpo humano ou articulações. Cada *heatmap* contém a localização de uma articulação segundo uma distribuição Gaussiana centrada exatamente nas coordenadas dessa articulação. Esta distribuição é definida numa "mancha" de pixeis probabilisticamente distribuída.

hiperparâmetros do modelo

Parâmetros do modelo definidos pelo utilizador antes do treino podendo sofrer ajuste durante o mesmo. Dizem respeito à arquitetura e controlam o processo de aprendizagem. *Learning rate*, número de *layers* e neurónios, *batch size* e número de *epoch* são alguns exemplos de hiperparâmetros.

intermediate supervision

Também apelidado de *deep supervision* ou *auxiliary supervision* permite a incorporação de supervisão em camadas ocultas da rede neuronal com melhorias consideráveis de desempenho e reduzindo o efeito do *gradient vanishing problem*, tornando a rede ainda mais profunda pela capacidade discriminativa de diferentes *features*. Como tal, as *losses* desta supervisão devem ser combinadas com a função de perdas (*objective function*) para ajuste dos pesos.

jitter

No contexto do projeto, o ruído de alta frequência, em torno de um determinado *keypoint*, é conhecido como *jitter*. A sua origem, com presença na maioria dos algoritmos para HPE, deve-se à inconsistência nas anotações em alguns dados de treino e pela inferência ao nível do *frame*, não capturando evidências temporais ao longo do vídeo.

parâmetros do modelo

Parâmetros do modelo ajustados durante o processo de treino de forma a minimizar a *loss function*. Os pesos de *bias* são parâmetros comummente referidos.

métodos heurístico

No contexto de visão computacional, refere-se a uma abordagem de resolução de problemas baseada na experiência e no conhecimento *a priori* para encontrar soluções aproximadas ou parcialmente inferiores ao ideal. Os algoritmos heurísticos são frequentemente utilizados para detetar padrões em imagens e extrair *features* relevantes para posterior análise. Como tal, estes algoritmos são especialmente úteis em momentos em que um dado modelo matemático preciso é difícil de obter ou quando a sua resolução, por vias analíticas tradicionais, é demasiado complexa.

regression-based methods

Métodos explorados em *2D single person* PE numa *framework* de ponta a ponta capaz de mapear diretamente as coordenadas cinemáticas das articulações do corpo humano presente numa imagem ou *frames* de um vídeo. O facto de

restringir esta localização tende a diminuir o tempo de inferência e a precisão comparativamente aos *heatmap-based methods*.

self-supervised training

É um tipo de *supervised learning*, cujo método de treino recorre às *labels*, internamente criadas (idêntico ao *unsupervised learning*), pelo que a informação para treino lhe permite a execução de tarefas tradicionalmente orientadas a *supervised learning*.

semi-supervised training

No contexto do projeto, *semi-supervised learning* deve combinar o melhor dos métodos *supervised* e *unsupervised* para lidar com a escassez de *labeled data*. Durante o treino, o modelo tem acesso a uma pequena quantidade *labeled data*. Iterativamente, o modelo deve ser sujeito a uma quantidade superior de *unlabeled data* o que reduz substancialmente o tempo gasto em anotações manuais e preparação de dados.

top-down methods

Métodos categorizados em *2D/3D multi-person* PE constituinte de *multi-person* HMR. Tem por base uma localização prévia dos humanos na imagem ou *frames* de um vídeo (*bounding boxs* em torno dos mesmos) para, em seguida, com recurso a um *single person pose estimator* prever a localização dos *keypoints* referentes às principais articulações.

vanishing gradient problem

Problema verificado durante o treino de uma rede baseado *backpropagated gradients*. Naturalmente, a cada iteração do treino, cada um dos pesos da rede neuronal recebe uma atualização proporcional à derivada parcial da função de ativação face ao peso atual. Contudo, em alguns casos, esta alteração de peso não se verifica (rede aprende lentamente) uma vez que o gradiente é consideravelmente pequeno. Quanto mais camadas na rede, menor tenderá a ser o valor da derivada parcial em cada nó até a eliminação completa da sua contribuição. *Hyperbolic tangent* e *sigmoid* são alguns exemplos de funções de ativação em que se pode verificar o *vanishing gradient problem*.

weakly-supervised training

Similar ao *semi-supervised training*, no contexto do projeto, o treino tem por base a uma pequena quantidade de dados e conhecimentos, *e.g.*, *paired 2D ground-truth*, *unpaired 3D ground-truth*, parâmetros da camara, entre outros, que deve utilizar para aprimorar o colhimento (com base na pouca informação conhecida (*weak annotations*) explorar um dado paradigma).

Lista de Acrónimos

AI	*Artificial intelligence*
AP	*Average Precision*
ASM	*Active Shape Model*
BCE	*Binary Cross-Entropy*
BEV	*Bird's-Eye-View*
BMP	*Body Meshes as Points*
CAR	*Collision-Aware Representation*
CCE	*Categorical Cross-Entropy*
CNN	*Convolutional Neural Network*
ConvNet	*Convolutional Network*
CPM	*Convolutional Pose Machines*
CPN	*Cascaded Pyramid Network*
CRMH	*Coherent Reconstruction of Multiple Humans*
CSP	*Cross-Stage-Partial-connections*
CVAE	*Conditional Variational Autoencoder*
CVPR	*Computer Vision and Pattern Recognition*
DCGAN	*Deep Convolutional Generative Adversarial Network*
DCNNs	*Deep Convolutional Neural Networks*
DE	*Depth Estimation*
DL	*Deep Learning*
DNNs	*Deep Neural Networks*
DoF	*Degrees of Freedom*

DSTformer *Dual-stream Spatio-temporal Transformer*

ECCV *European Conference on Computer Vision*

FC *Fully Connected*

FK *Forward Kinematics*

FLOPS *Floating Point Operations per Second*

FN *False Negative*

FoV *Field of View*

FP *False Positive*

fps *Frames per Second*

GANs *Generative Adversarial Networks*

GCNs *Graph Convolutional Networks*

GLAMR *Global Occlusion-Aware Human Mesh Recovery*

GMI-Net *Generative Motion Infiller Network*

GPU *Graphics Processing Unit*

GRUs *Gated Recurrent Units*

GTP-Net *Global Trajectory Predictor Network*

GTRS *Graph Transformer network for human mesh ReconStruction*

HMOR *Hierarchical Multi-Person Ordinal Relations*

HMR *Human Mesh Recovery*

HOTA *Higher Order Tracking Accuracy*

HPE *Human Pose Estimation*

HPS *Human Pose and Shape*

HRNet *High-Resolution Net*

HybrIK *Hybrid Inverse Kinematics*

ICCV *International Conference on Computer Vision*

IEF *Iterative Error Feedback*

IK *Inverse Kinematics*

ILP *Integer Linear Programming*

IMC Índice de Massa Corporal

IMUs *Inertial Measurement Units*

IoU *Intersection over Union*

LoCO *Learning on Compressed Output*

LSP *Leeds Sports Poses*

LSTMs *Long Short-Term Memory*

MAE *Mean Absolute Error*

MDN *Mixture Density Network*

MHSA *Multi-Head Self-Attention*

MLP *Multilayer Perceptron*

MOT *Multiple Object Tracking*

MOTA *Multi-Object Tracking Accuracy*

MPJPE *Mean Per Joint Position Error*

MPT *Multiple Person Tracking*

MSE *Mean Squared Error*

NBF *Neural Body Fitting*

NLP *Natural Language Processing*

OCHMR *Occluded Human Mesh Recovery*

PA-MPJPE *Procrustes-Aligned MPJPE*

PAFs *Part Affinity Fields*

PandaNet *Pose estimAtioN and Detection Anchor-based Network*

PANet *Path Aggregation Network*

PARE *Part Attention REgressor*

PCA *Principal Component Analysis*

PE *Pose Estimation*

PRTR *Pose Regression TRansformers*

PSE *Pose and Shape Estimation*

PVE *Per-Vertex Error*

R-CNN *Region-Based CNN*

Re-ID *Re-Identication*

ResNet *Residual Network*

RF *Radio Frequency*

RGB *Red, Green and Blue*

RH *Relative Human*

RLE *Residual Log-likelihood Estimation*

RNNs *Recurrent Neural Networks*

RoI *Region of interest*

ROMP *Regress all meshes in a One-stage fashion for Multiple 3D People*

RPN *Region Proposal Network*

SDF *Signed Distance Field*

SHG *Stacked Hourglass*

SLAM *Simultaneous Localization and Mapping*

SMAP *Single-Shot Multi-Person Absolute 3D Pose Estimation*

SMIL *Skinned Multi-Infant Linear*

SMPL *Skinned Multi-Person Linear*

SORT *Simple Online and Realtime Tracking*

SOTA *State-of-the-Art*

SPIN *SMPL oPtimization IN the loop*

SPP *Spatial Pyramid Pooling*

SSE *Sum-Squared Errors*

STAR *Sparse Trained Articulated Regressor*

SVM *Support Vector Machines*

VIBE *Video Inference for Body Pose and Shape Estimation*

ViT *Vision Transformer*

YOLO *You Only Look Once*

Esta página foi propositadamente deixada em branco

Lista de Símbolos

Símbolo	Descrição
$\boldsymbol{\alpha}$	Ângulo *swing* (modelo HybrIK)
$\boldsymbol{\beta}$	*Shape parameters*
$\boldsymbol{B}^i$	Conjunto de *shape parameters* β da pessoa i
$\boldsymbol{C}$	Parâmetros extrínsecos do sistema visual inferido
$\boldsymbol{E}$	*Energy function* que parametriza a *global optimization*
$\boldsymbol{\epsilon}_k$	Erro de reconstrução da articulação k
$\boldsymbol{\eta}$	*Heading angle* da rotação $\boldsymbol{\gamma}$
$\boldsymbol{\gamma}$	*Root rotations* da câmara face à *root joint*
$\boldsymbol{\mathcal{I}}$	Matriz identidade 3×3
$\boldsymbol{I}$	Vídeo com H *frames*
$\boldsymbol{K}$	Número de articulações do modelo SMPL, $K = 23$
$\boldsymbol{L}$	Número de pessoas no cenário
$\mathbf{M}$	*Reconstructed Mesh*
$\boldsymbol{\mathcal{M}}$	*Generative Motion Infiller*
$\boldsymbol{N}$	Número de pontos definidos na malha $\mathbf{M}$, $N = 6980$
$\mathbf{P}^i$	*Regressed 3D pose* da pessoa i
$\boldsymbol{\Psi}$	*Egocentric trajectory* gerada pelo *Global Trajectory Predictor* $\boldsymbol{\mathcal{T}}$
$\mathtt{pa}(k)$	Índice de parentesco da k-ésima articulação
$\boldsymbol{Q}^i$	*Global motion* da pessoa i
$\boldsymbol{R}^i$	Conjunto de *root rotations* γ da pessoa i
$\mathbf{R}$	Rotações relativas das articulações
$\boldsymbol{\mathcal{S}}$	Função linear do SMPL para obtenção da malha $\mathbf{M}$
$\boldsymbol{\mathcal{D}}^{sw}$	*Closed-form solution* para determinação da rotação decomposta $\boldsymbol{R}^{sw}$
$\boldsymbol{\mathcal{D}}^{tw}$	Transformação do ângulo ϕ na rotação decomposta *twist* $\boldsymbol{R}^{tw}$
$\mathbf{T}$	*Rest pose template*
$\boldsymbol{\Theta}^i$	Conjunto de *body motion* θ da pessoa i
$\boldsymbol{\mathcal{T}}$	*Global Trajectory Predictor*
$\boldsymbol{\theta}$	*Pose parameters*
$\boldsymbol{\tau}$	*Root translation* da câmara (Z_R) face à *root joint*
$\boldsymbol{T}^i$	Conjunto de *root translations* τ da pessoa i
$\boldsymbol{\Phi}$	Conjunto de ângulos *twist* ϕ (modelo HybrIK)

Símbolo	Descrição
$\mathbf{U}^i$	*Reconstructed 3D pose* da pessoa i
$\boldsymbol{v}$	*Latent code* da *CVAE-based Global Trajectory Predictor Network*
V	Máscara de visibilidade das deteções dos humanos no vídeo I
$\boldsymbol{z}$	*Latent code* da *CVAE-based Generative Motion Infiller Network*

Capítulo 1

Introdução

O presente capítulo expõe, de forma objetiva, a contextualização do projeto, evidenciando a definição do problema que motiva o seu desenvolvimento e os objetivos orientadores do mesmo. Não obstante, face à solução preconizada e sucintamente analisada, são enumerados potenciais resultados dos quais o autor se compromete a atingir.

Complementarmente é ilustrado, numa representação em diagrama de Gantt, as diversas etapas, cronologicamente organizadas, nas diferentes fases de desenvolvimento. Posto isto, num último tópico, é apresentado, de forma sucinta, a organização da dissertação introduzindo o conteúdo abordado.

1.1 Contextualização

Conseguir estimar a disposição articular e a forma corporal que caracteriza a posição do humano, num cenário real a partir de um sistema monocular, revela-se um desafio de visão computacional com contribuições constantes deste o trabalho de Hogg [1]. *Pose and Shape Estimation* (PSE) destaca-se, atualmente, como a base de diversos sistemas visuais, procurando compreender a posição de um ou vários corpos, num conjunto de imagens ou *frames* de um vídeo, através da estimação das principais articulações dos mesmos. Na verdade, a compreensão da postura corporal e do movimento do humano ao longo do tempo revela-se fundamental para análise do seu comportamento e posterior dedução de ilações que remetam para interações entre sujeitos.

A visão humana permite a perceção visual de um conjunto de informações relativas aos limites estruturais do corpo humano, mesmo em situações de oclusão é feita uma dedução de uma possível posição. *Human Pose and Shape* (HPS) fornece, nos dias atuais, informações geométricas e de movimento do corpo humano numa variedade de aplicações, desde o reconhecimento de uma sequência de movimentos representativos de uma atividade humana, realidade aumentada, *tracking* de movimento, assistente desportivo, sistema videovigilância, *trajectory prediction* e interação homem-máquina [2].

Apesar da elevada performance conseguida em tarefas de 2D *Pose Estimation* (PE), esta representação carece da informação perdida na terceira dimensão espacial. Não obstante, 3D PE, orientada pela representação num conjunto de articulações baseada em rotações relativas, compartilha várias propriedades da determinação dos *keypoints* a duas dimensões [3]. De forma a lidar com a ambiguidade inerente do problema em causa, surgiram implementações iniciais que recorreram a sensores alternativos como *Microsoft Kinect* [4] e *Inertial Measurement Units* (IMUs) [5], tecnologia baseada em *Radio Frequency* (RF) ou ainda *depth* [6] e *point cloud sensors* [7]. Apesar dos resultados interessantes, as abordagens revelaram-se, atualmente, pouco vantajosas na relação custo-efeito, fruto do *hardware* adicional.

Alternativamente, métodos de inferência baseados em sequências provenientes de sistemas monoculares [8, 9, 10, 11] ou *multiple camera views* [12, 13, 14, 15] foram priorizados. Com a rápida evolução de soluções baseados em *deep learning*, os métodos clássicos de visão computacional em diversas tarefas (*e.g.*, *image classification* [16], *semantic segmentation* [17] e *object detection* [18]) viram-se superados. O mesmo se verificou em tarefas de *Pose and Shape Estimation* (PSE) apesar de desafios associados à oclusão visual, dados de treino insuficientes e ambiguidade na componente de profundidade do sistema.

Não obstante, decorreu uma necessidade de incorporação de modelos estatísticos para uniformização da representação da pose e forma corporal (*e.g.*, modelo paramétrico SMPL [19]) tornando-se recorrente nos trabalhos mais recentes [20, 21, 9, 22, 23, 24, 25, 11].

1.2 Definição do Problema

Os primeiros métodos envolviam o uso *hardware* adicional que, do ponto de vista computacional, aumentava a complexidade e reduzia a eficiência do sistema. Como tal, nos últimos anos verificou-se um movimento focado na redução deste mesmo *hardware*, nomeadamente com recurso a um único sensor de imagem *Red, Green and Blue* (RGB). O seu potencial é notável, permitindo a incorporação em diversas aplicações com um custo-benefício superior [3].

Contudo, surge um problema característico deste tipo sistemas, concretamente, a ambiguidade na determinação de profundidade, pela consideração do mundo a partir de uma imagem a duas dimensões. Na consequência deste facto, prevê-se um aumento do custo computacional pela necessidade de estimação de uma maior quantidade de parâmetros e implementação de relações visuais por métodos alternativos, parcialmente imprescindíveis em *multi-view systems* [26].

Face à natureza particularmente desafiadora e aos escassos conjuntos de dados, propriamente recolhidos para tal problema, algumas implementações procuram estimar o modelo corporal completo do humano no espaço, recorrendo à informação recolhida nas imagens que alimentam o sistema, *e.g.*, silhuetas, *edges* ou *2D keypoints* [27, 28, 20, 29, 30]. No entanto, em cenários reais, *e.g.*, veículos de condução autónoma ou robôs de apoio, existe a necessidade de compreensão holística do movimento humano em coordenadas globais para definir, de forma segura, potencias ações a serem tomadas. Esta tarefa vê-se dificultada, uma vez que os parâmetros descritivos do humano são estimados no sistema de coordenadas da câmara [26, 31, 32, 33, 23] ou em relação à *root joint* [21], desprezando a componente de variação dinâmica da câmara ao longo do tempo (*i.e.*, a posição da câmara relativamente ao sistema de coordenadas do mundo muda a cada *frame*).

Como tal, a trajetória do humano, estimada em coordenadas globais, será imprecisa. Este facto, deve-se à dificuldade em relacionar a dinâmica da câmara (posições no espaço determinadas através de *Simultaneous Localization and Mapping* (SLAM)) com o movimento humano a avaliar, evidenciando-se o erro proporcionado pela ambiguidade da profundidade.

Ainda assim, a performance do sistema vê-se afetada nos momentos em que ocorra oclusão visual (*e.g.*, obstrução parcial/total do humano por sobreposição ou saída deste do *Field of View* (FoV) da câmara), típico em cenários movimentados e compostos por várias pessoas, pela dependência da deteção completa das partes visíveis. Posto isto, evidenciam-se apenas alguns métodos capazes de recuperar a representação do modelo corporal em oclusões parciais para cada *frame* [9] ou oclusões totais durante um intervalo de tempo [34].

1.2.1 Objetivos

Dada a natureza complexa do sistema descrito, será mandatório o estudo e compreensão das tecnologias anteriormente desenvolvidas. Ainda assim, feita uma contextualização e definido o problema que motiva este trabalho, segue-se a enumeração dos objetivos que deverão ser atendidos ao longo do projeto:

- Desenvolvimento de uma revisão literária com base nos principais e atuais métodos de PSE;

- Estudo e documentação de conceitos e técnicas inerentes à compreensão dos principais métodos de PSE;
- Projeção justificada de uma arquitetura que acompanhe a implementação de uma solução proposta para PSE;
- Possibilitação da solução em estimar o movimento do humano ao longo do tempo pela determinação da rotação relativa das articulações, translação e orientação do/dos humanos, presentes no ambiente de estudo, em relação à câmara, a partir de um vídeo;
- Estudo e implementação de um método de otimização focado na redução do erro do movimento humano (*e.g.*, redução da variação angular das rotações relativas para as mesmas poses ao longo do tempo);
- Obtenção, de forma coerente, dos parâmetros descritivos do movimento humano e da posição da câmara num sistema de coordenadas global;
- Avaliação do sistema desenvolvido recorrendo a *datasets* obtidos em cenários reais, segundo as métricas internacionalmente estabelecidas para o tema do projeto.

1.2.2 Visão Geral do Método Proposto

Em resposta aos desafios referidos, o autor deste trabalho propôs uma solução, capaz de acompanhar, compreender e prever o movimento das várias pessoas, num ambiente real, nas coordenadas de um sistema global. Apesar da maior fiabilidade presente num sistema *multi-view*, ao lidar com oclusões parciais e na determinação da profundidade do humano no espaço, também será verdade que a sua incorporação se revelará especialmente complexa em sistemas variáveis pela necessidade de relacionamento espacial entre câmaras. Como tal, seguindo o movimento da comunidade orientado à redução de *hardware*, aposta-se num implementação baseada numa única câmara, devendo, para tal, ser estabelecidas as limitações que regem o sistema.

Posto isto, o método proposto organiza-se em três fases, numa adaptação de diferentes *frameworks*, individualmente otimizadas, para o seu propósito. Como tal, recorre-se a um *Multiple Person Tracking* (MPT) para identificação e *tracking* das pessoas presentes no vídeo de entrada. A localização do humano na imagem, quando presente, alimenta o *HPS estimator* capaz de gerar, no conjunto total de *frames*, uma sequência que define o movimento do humano no sistema de coordenadas da câmara (*i.e.*, *root translations*, *root rotations*, pose do corpo e os parâmetros do *shape*).

Naturalmente, esta sequência de informação tenderá a ser incompleta fruto de oclusões evidentes ou deteções erráticas por parte do MPT. De forma a lidar com este problema, decorre a integração do *generative motion optimizer*, concretamente,

os últimos estados proposto na implementação do GLAMR [34], numa arquitetura baseada em *Transformers*, para estimar o movimento do corpo humano sem oclusão no sistema de coordenadas do mundo. A incorporação deste modelo introduz coerência temporal de longo alcance, potencializando o preenchimento da informação corporal localmente definida (segundo a dinâmica humana aprendida) e a previsão da trajetória global de todos os humanos baseado na variação posicional local (posição no plano e orientação) e numa otimização iterativa dos parâmetros da câmara consistente com as evidências do vídeo, *e.g.*, *2D keypoints*.

1.3 Calendarização

Pretende-se que o projeto em causa seja desenvolvido ao longo de 25 semanas, com todo o planeamento devidamente definido consoante a Tabela 1.1.

Tabela 1.1: Calendarização do projeto. Melhor análise ampliado.

CALENDARIZAÇÃO DO PROJETO	Início	Fim	Duração	março				abril				maio					junho				julho				agosto			
				1ª	2ª	3ª	4ª	5ª	6ª	7ª	8ª	9ª	10ª	11ª	12ª	13ª	14ª	15ª	16ª	17ª	18ª	19ª	20ª	21ª	22ª	23ª	24ª	25ª
Introdução e formulação do problema	06/03/2023	09/03/2023	4 dias																									
Análise de requisitos/funcionalidades	10/03/2023	12/03/2023	3 dias																									
Estudo de técnicas e modelos existentes: *pose estimation*	13/03/2023	26/03/2023	2 semanas																									
Estudo de técnicas e modelos existentes: *pose and shape estimation*	20/03/2023	02/04/2023	2 semanas																									
Estudo de técnicas relativas a *monocular e multi-view systems*	03/04/2023	16/04/2023	2 semana																									
Elaboração do relatório: introdução e trabalhos relacionados	13/03/2023	23/04/2023	6 semanas																									
Elaboração do relatório: conceitos fundamentais	24/04/2023	21/05/2023	4 semanas																									
Estudo e definição da abordagem a utilizar	22/05/2023	28/05/2023	1 semana																									
Desenvolvimento da arquitetura do sistema e implementação da solução proposta	29/05/2023	09/07/2023	6 semanas																									
Elaboração do relatório: implementação	03/07/2023	23/07/2023	3 semanas																									
Avaliação e melhoria da performance do sistema	24/07/2023	06/08/2023	2 semanas																									
Recolha de resultados experimentais	03/08/2023	06/08/2023	4 dias																									
Elaboração do relatório: resultados e conclusões	07/08/2023	20/08/2023	2 semanas																									
Elaboração do relatório final	13/03/2023	27/08/2023	20 semanas																									

Assim como é possível analisar, inicialmente é feita uma contextualização do projeto e formalização o problema em causa. Para tal, deve ser feita uma análise dos requisitos e potenciais funcionalidades exigidas.

Posteriormente, espera-se a realização de um estudo científico de técnicas e modelos existentes, comummente utilizados pela comunidade, focada no desenvolvimento de sistemas análogos, para PSE. Não obstante, técnicas baseadas em *monocular vison* e *multi-view vision* deverão ser alvo de estudo. O culminar desta informação deverá ser documentado, no presente relatório, numa estrutura racionalmente organizada e desenvolvida ao longo de seis semanas.

Não obstante, é feita uma abordagem sucinta aos diferentes termos e conceitos correntemente utilizados nas técnicas referidas e na implementação a construir. Como tal, é realizado um estudo e seleção da abordagem a seguir para formalização da arquitetura do sistema final.

Resultado do trabalho produzido até então segue-se a implementação da solução proposta ao longo de seis semanas com melhorias efetivas e validação da performance

da mesma. Finalizando o desenvolvimento, é feita uma recolha objetiva de dados qualitativos e quantitativos para exposição devida no relatório final.

1.4 Estrutura do Documento

O presente documento é composto por 6 capítulos capazes de descrever todo o desenvolvimento realizado. A organização deste segue uma lógica evolutiva do projeto final, pelo que o relatório foi sempre realizado em paralelo, procurando seguir a calendarização definido na Secção 1.3.

No presente capítulo é feito uma abordagem genérica ao projeto assim como realçadas as motivações que levaram ao seu desenvolvimento. Como tal, foram enumerados os objetivos que deverão ser atendidos ao longo do projeto e, através de um diagrama de Gantt, ilustrada a calendarização e a duração de cada etapa.

No Capítulo 2 é feito um estudo e levantamento de implementações para *2D/3D HPS estimation*. Como tal, é discutido uma variedade de abordagens e técnicas que permitirão um assimilar de conhecimentos e compreensão dos conceitos inerentes. O seguinte capítulo (Capítulo 3), descreve todo o estudo adicional necessário para a continuidade do trabalho, focando nos conceitos fundamentais e tecnologias necessárias para o desenvolvimento do projeto em questão.

O Capítulo 4 apresenta a arquitetura da solução proposta, situando novamente o leitor para os objetivos do projeto. Naturalmente, é exposto todo o raciocínio, desde a criação do modelo até à contribuição do autor, segundo um desenvolvimento puramente baseado em *software*, para cumprimento dos objetivos inicialmente estipulados.

Ao longo do Capítulo 5 são reunidos os resultados provenientes da implementação, apresentada no passado capítulo, numa avaliação qualitativa e quantitativa baseada em métricas globalmente definidas para o tema principal deste projeto.

No último capítulo (Capítulo 6) estabelecem-se as considerações finais, sendo apurados os objetivos cumpridos e reunidos eventuais aspetos a serem melhorados e considerados em implementações futuras.

Capítulo 2

Trabalhos Relacionados

O ambiente real em análise poderá dificultar a perceção completa do corpo humano. Movimentos inesperados, muitas vezes associados de *motion blur* e restrições de iluminação levaram ao estudo e desenvolvimento, por parte da comunidade de visão computacional, de várias propostas, com diferentes tipo de abordagens, numa tentativa de atingir a solução ótima e superar os diversos desafios inerentes.

Neste capítulo, é apresentada uma visão geral focada em implementações de *3D Human Pose and Shape (HPS) estimation*. Como tal, será discutido uma variedade de abordagens e métodos que permitirão um assimilar de conhecimentos e compreensão maior dos conceitos inerentes.

Do ponto de vista geral, os métodos explorados abordam o problema em diferentes fases, ou somente numa fase (*two-stage vs. one-stage*). Concretamente, identificando a região de interesse (região da imagem com humanos) e posterior deteção das articulações ou, unicamente, a localização efetiva das articulações correlacionando-as posteriormente. Estas metodologias são, atualmente, fundamentas nas mais recentes abordagem para estudo de imagem, *i.e.*, *Deep Convolutional Neural Networks* (DCNNs), *Deep Convolutional Generative Adversarial Network* (DCGAN) e *Recurrent Neural Networks* (RNNs), rivalizando com arquiteturas baseadas em *Transformers* capazes de impor *coerência temporal*.

Auxiliado deste processo, estão definidos diferentes modelos corporais capazes de parametrizar o estado atual da árvore cinemática humana e descrever, de certa forma, a composição anatómica do corpo humano.

2.1 *Human Body Models*

Efetivamente, numa imagem são esperadas duas abordagens, nomeadamente:

- *2D PE* onde se procura estimar a localização das articulações do corpo humano numa espaço relativo a duas dimensões, *i.e.*, imagem ou *frame* de um vídeo. A localização é representada com coordenadas X e Y para cada *keypoint*;
- *3D PE*, convertendo o conteúdo relevante da imagem 2D num objeto 3D estimando a dimensão em falta, *i.e.*, Z. Como tal, será possível estimar a posição espacial da pessoa.

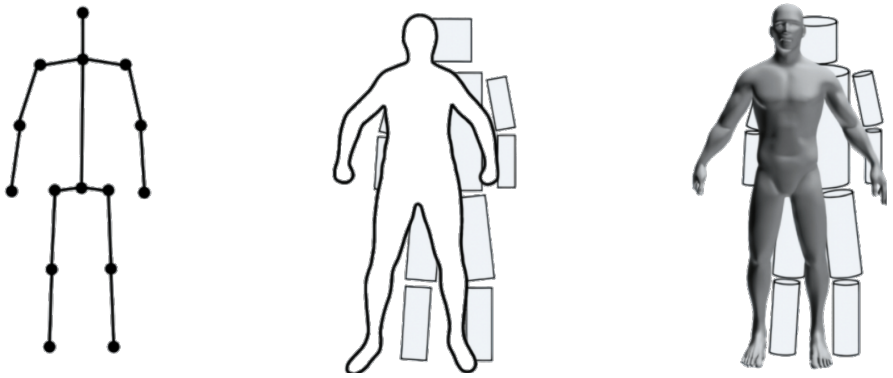
De facto, a capacidade de perceção de profundidade com base numa única câmara atua como um problema clássico em visão computacional, com contribuições constantes de investigadores em todo o mundo. Masoumian *et al.* [35] apresentam uma análise a mais de 150 artigos científicos, do desenvolvimento atual em *monocular Depth Estimation (DE)*[1], não sendo foco deste projeto o seu levantamento, a sua noção e diversidade de técnicas existentes baseadas em *Deep Learning* (DL) servirão de alicerce à compreensão de partes de algoritmos orientados a 3D PE.

Do ponto de vista anatómico, o corpo humano é uma estrutura complexa e flexível, com diversas características como estrutura cinemática, formato do corpo, textura superficial e, entre outras, a posição das articulações que o compõem [36]. Não obstante, são tipicamente referidos três tipos de modelos para *Human Pose Estimation* (HPE), *i.e.*, *skeleton-based model* (usado para *2D/3D HPE*), *Contour-based model* (usado para *2D HPE*) e *volume-based model* (usado para *3D HPE*), presentes na Figura 2.1. Estes modelos procuram representar os *keypoints* e as *features* obtida dos dados de entrada, *e.g.*, imagem ou *3D point cloud data* de forma coerente.

Chen *et al.* [36], numa publicação cientifica onde se encontram reunidos diversos métodos para HPE baseados em *deep learning*, afirma que a modelação do corpo humano é um elemento chave para atingir o objetivo aqui referido. São, como tal, evidenciados várias técnicas, numa abordagem *model-based* capazes de inferir e renderizar a posição corporal 2D/3D do humano [2]. Uma descrição mais detalhada sobre os modelos do corpo humano direcionados a esta aplicação podem ser encontradas nos artigos científicos de Liu *et al.* [37] e Gong *et al.* [38].

Skeleton-based Model: *Skeleton-based model*, também conhecido como *stick figure* ou *kinematic model* (Figura 2.1a), representa a localização de um conjunto de articulações (tipicamente entre 10 a 30, *e.g.*, tornozelos, joelhos, ombros, cotovelos e pulsos, entre outros) e a correspondente orientação dos membros [36]. Este

[1] DE, é um processo computacional aplicado a uma ou mais imagens focado em estimar a profundidade de cada elemento no cenário, eventualmente recorrendo a representações por *dense depth maps* ou *sparse depth maps* [35].



(a) *Skeleton-based model* (b) *Contour-based model* (c) *Volume-based model*

Figura 2.1: *Modelos do corpo humano tipicamente utilizados* [36].

modelo pode ainda ser representado como um gráfico, onde os vértices indicam as articulações e as linhas indicam as conexões entre as mesmas, com as respetivas restrições (*tree-structured model*) [39]. Naturalmente, esta representação é demasiado simplista, contudo é amplamente utilizada em 2D e 3D HPE [40, 41, 42] e referenciada em *datasets* de pose humana [43]. Apesar da simples representação, neste modelo, podemos contar com falta de muita informação, *i.e.*, *contour* e *shape* do corpo humano.

Contour-based Model: O *contour-based model* (Figura 2.1b), como é percetível no nome, contem informação adicional que permite estimar a largura e contorno da cabeça, tronco e membros do humano. Igualmente utilizado em HPE, os limites das partes do corpo humano são aproximadamente representadas com retângulos [36]. Anteriormente, Ju *et al.* [44] utilizou *contour-based models* ao assumir que uma pessoa poderia ser representada pela união de *planar patches* (conhecido por modelo de pessoa de papelão) enquanto Cootes *et al.* [45] apresentou o *Active Shape Model* (ASM) capaz de recolher os contornos do corpo humano aplicado a técnica *Principal Component Analysis* (PCA) para redução da quantidade de pontos de um dado conjunto representativo num menor, mantendo o conceito informativo inicial.

Volume-based Model: Também conhecido por *volumetric model* (Figura 2.1c) é usado para *3D Pose and Shape Estimation (PSE)*. Consiste em modelos e poses do corpo humano 3D representadas por formas geométricas ou malhas. Sidenbladh *et al.* [46] usou formas geométricas, como cilindros, para modelar partes do corpo humano. No entanto, modelos mais recentes são representados num formato de malha, normalmente obtidos através de *scans* 3D [36]. Concretamente, o modelo ***Skinned Multi-Person Linear* (SMPL)** [19], amplamente utilizado em 3D PSE, mostrou-se capaz de modelar uma representação ajudada a potenciais deformações dos tecidos moles face à posição natural do humano. Focado num melhor controlo da deformação, o treino modelo baseou-se num *multi-pose dataset* que continha 1786 *scans* humanos 3D de elevada resolução (891 registos de 20 mulheres e 895 registos

de 20 homens) enquanto o *multi-shape dataset* consistia no *CAESAR dataset* [47] com um total de 1700 registos de homens e 2107 de mulheres em diversas posições para otimizar os *blend weights, pose-dependent blend shapes, mean template shape* (humano parado) e um *shape regressor* para a localização das articulações de forma a minimizar o erro de vértice do modelo nos dados de treino [19].

2.2 *2D Human Pose Estimation Approaches*

Os métodos de *2D HPE* estimam a localização das articulações do corpo humano, a duas dimensões, a partir de imagens ou *frames* de vídeos. Tradicionalmente, os métodos para este fim adotavam diferentes técnicas para extração de *features* das partes do corpo humano, assumindo o mesmo como uma *stick figure* [2]. Ainda assim, implementações baseadas em DL, *e.g.*, CNNs para extração dessas *features*, resultaram num aumento considerável de performance. Neste capítulo segue-se uma análise aos métodos de *2D HPE* baseados em *deep learning* em cenários com apenas uma pessoa (*single person pose estimation*) e várias pessoas (*multi-person pose estimation*). A Figura 2.2 esquematiza, de forma sucinta, a organização desta secção com a apresentação dos métodos e variantes expostas.

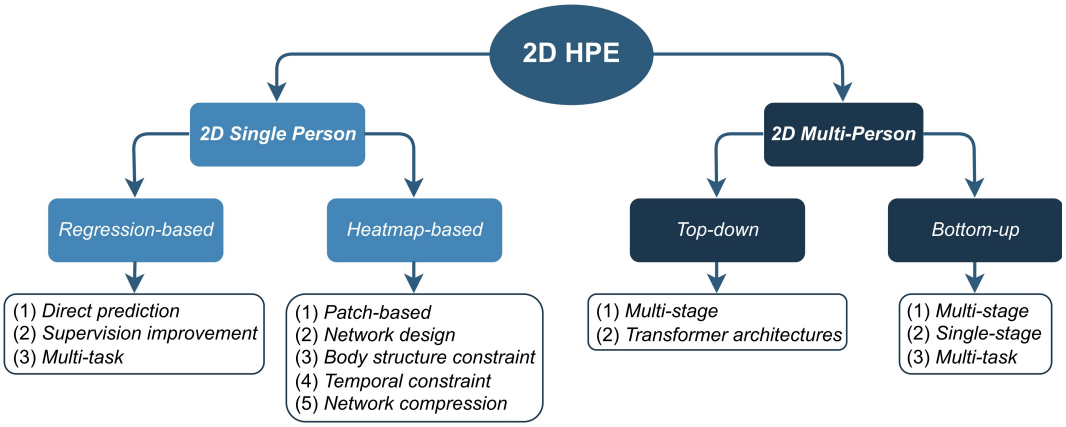


Figura 2.2: Organização categórica de *deep learning-based 2D HPE*.

2.2.1 *2D Single Person Pose Estimation*

Para uma imagem com uma única pessoa, espera-se, à saída de um algoritmo de *2D single person PE*, obter a localização da posição das suas articulações. Na existência de mais do que um humano, em certos algoritmos, a imagem é primeiramente cortada (com base no tamanho do corpo encontrado) para que exista apenas uma pessoa em cada subimagem [48]. Este processo é realizado automaticamente num *upper-body detector* ou *full-body detector* (ou *whole-body*, *i.e.*, corpo, cara, mãos e pés) [36]. Com base em diferentes técnicas para *human pose estimation*, os métodos

propostos baseados em DL podem ser classificados em duas categorias: *regression-based methods* e *heatmap-based methods* (ou *detection-based methods*).

Regression-based methods

Os *regression-based methods* mapeiam diretamente as coordenadas cinemáticas das articulações do corpo humano presente numa imagem [49]. Tipicamente compostas por menos passos não diferenciáveis, estes algoritmos são menos precisos comparativamente aos *heatmap-based methods*.

A restrição da localização da articulação a apenas um ponto dificulta o processo uma vez que se trata de um problema altamente não linear [36]. A estrutura geral dos *2D single person PE regression-based methods* encontra-se representada na Figura 2.3.

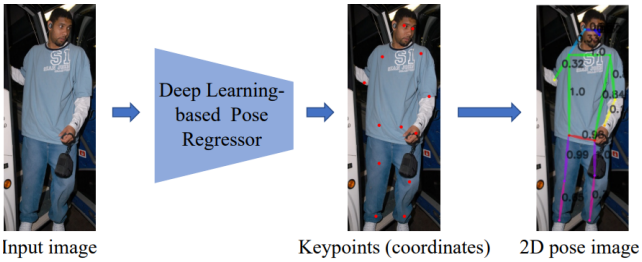


Figura 2.3: Estrutura genérica dos *regression methods*.

Toshev *et al.* [49] propuseram um *cascaded deep neural network regressor* apelidado de ***DeepPose*** para aprender a determinar a localização das articulações. Ao utilizar *AlexNet* [16] como *backbone*, constituía uma das primeiras redes para HPE fundamentada em DL. Concretamente, realizava uma deteção nas imagens inteiras, sem processamentos adicionais, *i.e.*, sem qualquer modelo corporal ou detetor de partes do corpo, assim como representado na Figura 2.4. Seguidamente, a arquitetura em cascata do *multi-stage refining regressors* (*stage* $s \in \{1, \ldots, S\}$ de um total de S *cascade stages*) é usada para ajustar as imagens cortadas no estado anterior e melhorar o desenho da janela [36].

Figura 2.4: Estrutura do DeepPose [49].

Como resposta às dificuldades inerentes de previsão das articulações humanas a partir de uma imagem surgiram redes neuronais mais poderosas. Carreira *et*

al. [50] propôs uma ***Iterative Error Feedback* (IEF)** *network* baseada na ***GoogLeNet*** [51] (Figura 2.5), contribuindo com um dos primeiros modelo dotado de supervisão. Tal como o nome sugere, trata-se de um modelo com efeito progressivo de auto-correção através da alteração da solução inicial com um erro de previsão. De facto, o *IEF* apresentou uma performance interessante nos *benchmarks MPII Human Pose dataset* [43] e *Leeds Sports Poses (LSP) dataset* [52]. Na Figura 2.5 é visível a sequência de correções esperadas neste modelo, começando numa posição inicial y_0 (esquerda) e terminando na posição verdadeira y (direita). Cada *keypoint* é movido ao longo de uma linha, pelo que as sucessivas correções em cada imagem são constantes.

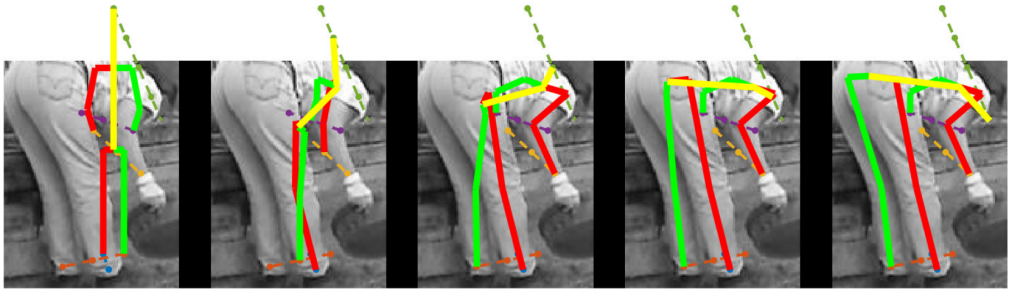


Figura 2.5: Sequência de imagens no processo de autocorreção com base no *IEF* [50]

Ainda assim, a implementação de redes focadas em *multi-task learning* veio a otimizar o processo de deteção de *features* de interesse [53]. A distribuição devida de tarefas relacionadas (*e.g.*, PE e reconhecimento de ações baseado na pose humana) permitiu ao modelo generalizar melhor a tarefa original (*pose estimation*) [2]. Neste sentido, Li *et al.* [54] propôs uma *heterogeneous multi-task framework* orientada a duas tarefas (Figura 2.6): 1) prever as coordenadas das articulações de uma imagem inteira através de um *regressor* e 2) detetar partes do corpo em imagens cortadas usando a técnica de janela deslizante.

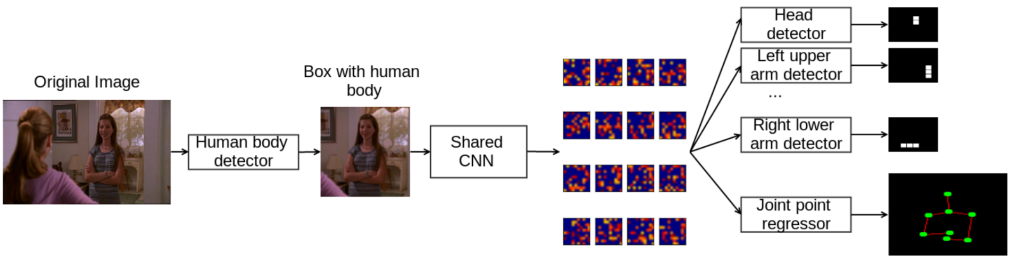


Figura 2.6: *Heterogeneous multi-task learning for pose estimation* [53]

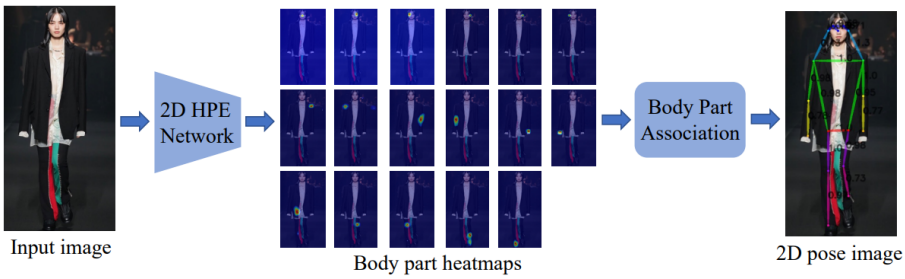
Luvizon *et al.* [55] projetou uma rede aparentemente capaz lidar com o problema de 2D/3D PE e reconhecimento de ações humanas num vídeo. A posição estimada no meio da rede pode ser usada como referência para reconhecimento de ações. Contrariamente ao analisado até ao momento neste capítulo, 2D e 3D PE

era, na literatura, realizado com tarefas distintas. Novamente, o autor frisa que uma otimização de ponta a ponta apresenta uma precisão melhorada quando comparado com um processo de aprendizagem dividido.

Heatmap-based methods

Por sua vez, *heatmap-based methods* procuram estimar a localização aproximada das partes do corpo humano [56] ou articulações [48]. Por norma, são acompanhadas por uma sequência de janelas retangulares (cada uma incluindo uma parte do corpo) [56] ou *heatmaps* (cada um indicando a localização de uma articulação segundo uma distribuição Gaussiana 2D centrada exatamente nas coordenadas dessa articulação) [48].

Comparativamente ao método descrito anteriormente, a localização da articulação é definida por uma "mancha" de píxeis, probabilisticamente distribuída, o que potencializa a sua robustez. Contudo, comparativamente ao tamanho da imagem original, esta representação apresenta menor resolução fruto da passagem pelas camadas de *pooling* nas CNNs, o que limita a precisão na de estimação da localização das articulações [36]. A estrutura geral dos *2D single person PE heatmap-based methods* encontra-se representada na Figura 2.7.



(b) Heatmap-based Methods

Figura 2.7: Estrutura genérica dos métodos baseados em *heatmaps* [2].

Concretamente, o objetivo será estimar os H *heatmaps* $\{H_1, \ldots, H_K\}$ para um total de K *keypoints*. O valor do pixel $H_i(x, y)$ em cada *heatmap* de uma articulação indica a probabilidade de que esse ponto esteja efetivamente na posição correta. O *target* (ou *ground-truth*) *heatmap* é, geralmente, e como já referido, uma distribuição Gaussiana 2D centrada na localização da articulação (*ground-truth joint location*). Como tal, estas redes são treinadas de forma a minimizar a discrepância (*e.g.*, *Mean Squared Error* (MSE)) entre os *predicted heatmaps* e os *target heatmaps* [2].

Alguns métodos iniciais recorriam a redes neuronais para fazer deteção de partes do corpo e posterior classificação [56] ou prever o mapa de confiança pertencente às várias classes. Contudo, os métodos de deteção de partes do corpo humano são sensíveis ao aumento da complexidade no fundo da imagem e à ocorrência de oclusões parciais ou totais dos seus constituintes [36].

Xiao *et al.* [57] propuseram um modelo capaz de realizar *multi-person PE* e *tracking* em vídeos baseado no vencedor [58] do *International Conference on Computer Vision (ICCV) 2017 PoseTrack Challenge* [59]. Os autores, primeiramente, realizam a PE nos *frames* através de uma ***Mask R-CNN*** adicionamento apenas três *deconvolutional layers* após as últimas camadas de convolução da ***ResNet*** [60]. A estrutura adotada, permite, segundo os autores, gerar *heatmaps* centrado em cada articulação segundo uma distribuição Gaussiana 2D, de forma simplificada, a partir de *deep and low resolution features*.

Alguns trabalhos adotaram *multi-stage networks* para melhorar os resultados através de uma previsão via *end-to-end learning* [61]. Contudo, a existência de várias camadas numa *deep feedforward neural network* torna a mesma sujeitas a diversos problemas verificados na fase de treino [62], um deles conhecido por *vanishing gradients*[2]. Wei *et al.* [61] introduziram as redes convolucionais numa estrutura sequencial apelada de ***Convolutional Pose Machines* (CPM)** para prever a localização das articulações num processamento *multi-stage*. A cada estado, num CPM, as *features* presentes na imagem e os *2D belief maps* gerados no estado anterior são usadas para produzir uma previsão, da localização das partes do corpo, iterativamente ajustada. A existência, periodicamente, de camadas intermédias de supervisão, juntamente com os CPMs, melhoravam a performance e reduziam o efeito do *vanishing gradient problem*.

A Figura 2.8 sintetiza a arquitetura convolucional e os respetivos constituintes em cada camada para um CPM com T estados. Nas subimagens que a constituem é possível ver a "pose machine", (Figura 2.8a e 2.8b) (uma sequência de *multi-class predictors* treinadas para prever a localização de cada parte em cada nível da hierarquia) e as correspondentes redes convolucionais (Figura 2.8c e 2.8d). Na Figura 2.8e, a progressão do *effective receptive field* numa imagem (centrado no joelho esquerdo) da arquitetura [61].

Newell *et al.* [48] propuseram uma *encoder-decoder network* denominada ***Stacked Hourglass* (SHG)** *network* para atingir o processamento *bottom-up* e *top-down* com *intermediate supervision*. A rede SHG é composta por camadas consecutivos de *pooling* e *upsampling* para obtenção das várias *features* a diferentes escalas e, portanto, consolidação das várias relações espaciais associadas ao corpo.

Apesar dos esforços descritos para a projeção de uma rede eficaz na tarefa de HPE, a estrutura corporal definida foi também alvo de estudo a fim de fornecer uma melhor informação supervisionada na construção destas redes. Naturalmente, com a evolução das ***Generative Adversarial Networks* (GANs)** [63] a sua aplicabilidade viu-se vigorosamente eficaz em problemas de HPE de forma a gerar previsões

[2] *Vanishing gradient problem* ocorrem pois os *backpropagated gradients* diminuem à medida que são propagados por cada camada da rede. Desenvolvimentos iniciais, focados em problemas de classificação [62], indiciam que a adição de supervisão em *layers* intermédios da rede neuronal melhora a performance do sistema.

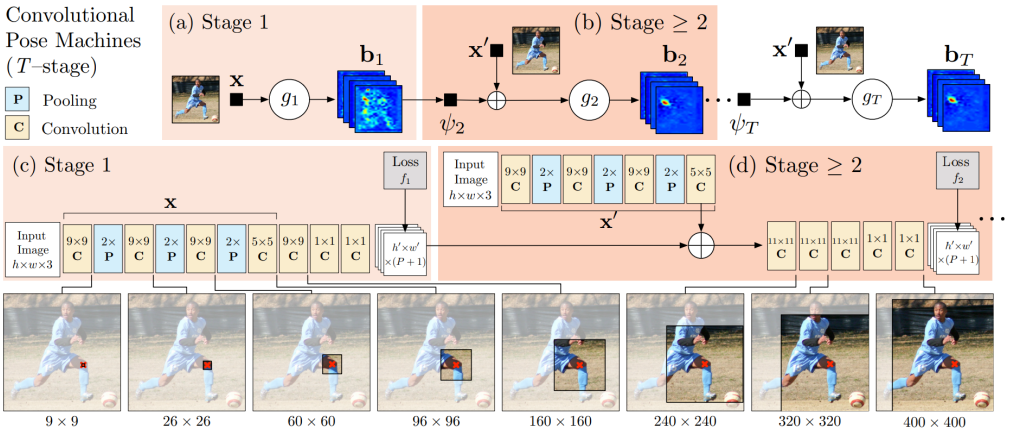


Figura 2.8: *Architecture and receptive fields of CPMs* [61].

biologicamente plausíveis e discriminar as de elevada confiança face às de menor confiança [2], fornecendo *adversarial supervision* para aprendizagem de diferentes estruturas corporais e treino de redes.

Chen *et al.* [64] propuseram uma ***Adversarial PoseNet*** composta por um *pose generator*, baseado numa *hourglass network* para determinação da posição das articulações através de *joint heatmaps* e *occlusion heatmaps*, e dois *discriminators* para distinguir entre poses reais e falsas. Assim sendo, sempre que os *discriminators* falhem na distinção entre os resultados gerados pelo *generator* e as poses reais, a rede aprenderá com sucesso. Este processo origina uma previsão com elevado grau de confiança no que à disposição anatómica das articulações do humano diz respeito.

A Figura 2.9 contém uma visão geral da *Adversarial PoseNet* proposta para HPE. São visíveis os 3 constituintes referidos, *i.e.*, a roxo a *multi-task network* G (*pose generator*), a azul o *pose discriminator* P para avaliar se a pose gerada é "real" e a verde o *confidence discriminator* C para verificar se o *generator* tem forte confiança na localização das partes do corpo. G começa por gerar *heatmaps* com baixa confiança e classificados como "falsos" perante os *discriminators*. Para tal, G é otimizado (representado no tracejado (2) e (3) com a passagem do gradiente das redes C e P, respetivamente) de forma a "enganar" os dois *discriminators* uma vez que gera *heatmaps* com maior confiança mesmo em situações de oclusão. Por fim, a linha a tracejado (1) indica os *backward gradients* para atualização da rede G.

No que diz respeito a sequências de vídeo gravadas com sistema de câmaras monocular será importante a modelação das informações temporais e espaciais a fim de realizar a tarefa de *2D human pose estimation*. Jain *et al.* [65] propuseram uma rede composta por duas ramificações de CNNs através de *multi-scale RGB frames* e *optical-flow maps* como entrada. De notar que o processamento relativo ao *optical-flow* terá sido realizado anteriormente segundo o algoritmo proposto por Weinzaepfel *et al.* [66]. No final do sistema é obtido um novo *dataset*, denominado

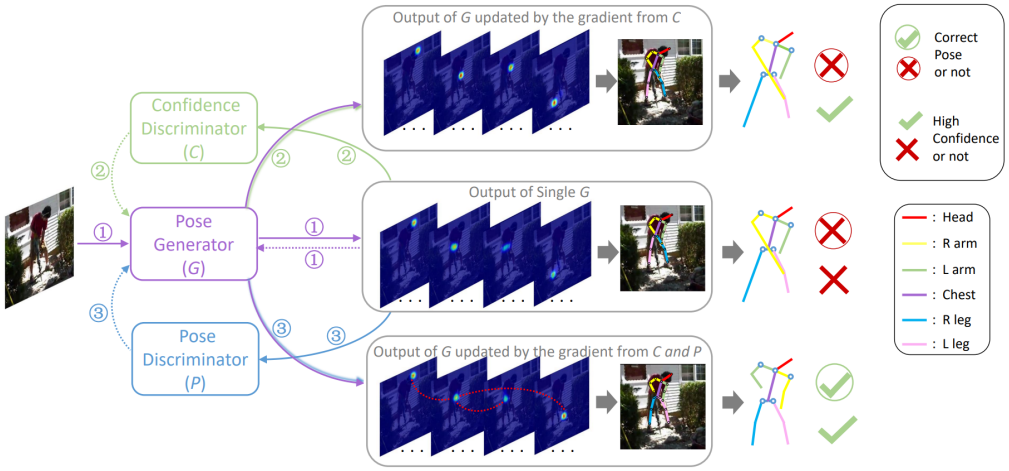


Figura 2.9: Estrutura da *Adversarial PoseNet* [64].

de ***FLIC-motion*** (adição de *motion features* ao *FLIC dataset*). A Figura 2.10 apresenta a versão final da arquitetura, baseada numa *multi-resolution ConvNet*. Cada *patch*, obtido a partir da técnica de janela deslizante na rede convolucional é devidamente normalizada segundo um ***Local Contrast Normalization*** **(LCN)** para os canais *RGB* e um ***Local Motion Normalization*** **(LMN)** para as *motion features*. As *features* obtidas (*i.e.*, *color features* e *motion features*, respetivamente) são concatenadas nas últimas camadas de convolução.

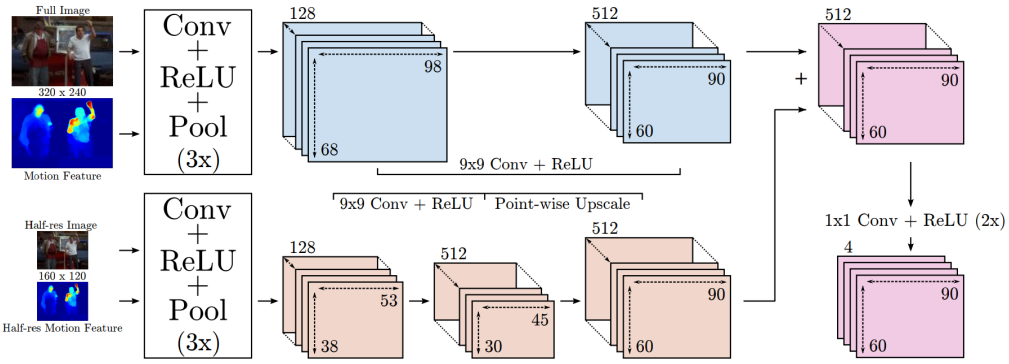


Figura 2.10: *Multi-resolution efficient sliding window model* [65].

2.2.2 *2D Multi-Person Pose Estimation*

Comparativamente com *single person PE*, *multi-person PE* é, de facto, mais complexo pelo número de passos intermédios esperados. Na verdade, o desafio revela-se maior uma vez que será necessário um pré-processamento adicional na deteção e localização das pessoas seguido da aplicação de um modelo análogo ao analisado na Subsecção 2.2.1. De forma a resolver este problema, as implementações orientadas a *2D multi-person PE* podem ser categorizadas em *top-down* e *bottom-up*. Alguns

exemplos de métodos para *2D HPE* conhecidos na literatura como ***OpenPose*** [41], ***AlphaPose*** [42] e ***HRNet*** [40] foram extensivamente utilizados como *2D pose detector* em métodos para *3D HPE*.

Top-down methods

Os métodos *top-down* recorrem, geralmente, a detetores de pessoas de forma a obter um conjunto de *bounding boxs* (em torno das pessoas na imagem de entrada). O processo seguinte aproveita desta deteção para aplicar o *single person pose estimator* e prever a localização das articulações. Naturalmente, o tempo de execução do sistema será proporcional ao número de pessoas [36] pelo que, geralmente, os resultados não são obtidos em tempo real. A *pipeline* genérica dos *2D multi-person PE top-down methods* encontra-se representada na Figura 2.11.

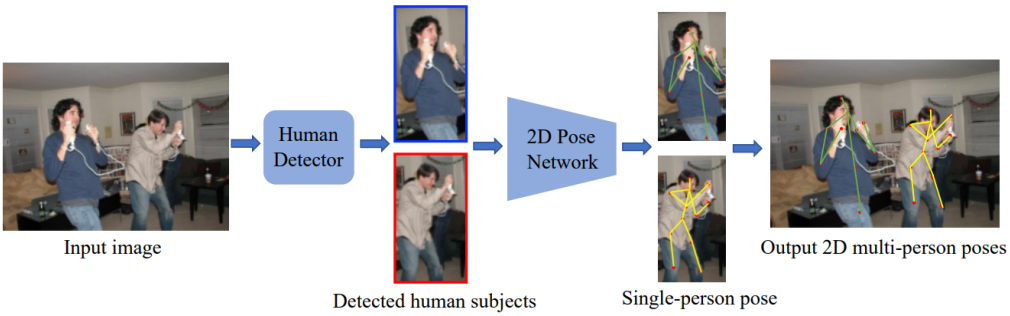


Figura 2.11: *Pipeline* genérica das abordagens *top-down* para *2D HPE* [2].

Alguns pesquisas basearam-se em detetores de humanos existentes, *e.g.*, *Faster R-CNN* [18] e *Mask R-CNN* [67]. Xiao *et al.* [57], anteriormente referido na Subsecção 2.2.1, adicionou três *deconvolutional layers* nas últimas camadas de convolucionais da ***Residual Network* (ResNet)** [60] para fazer a obter a pose dos humanos presentes no *frame* em causa. Trata-se de uma ***Mask R-CNN*** capaz de gerar *heatmaps* referentes a todos os *keypoints* obtidos de *deep* e *low resolution features*.

Muitos dos métodos para HPE, baseados em *Deep Convolutional Neural Networks* (DCNNs), repartiram o processo em sub-redes de *high-to-low resolution* conectadas em série, condicionando a resolução da imagem de entrada e reconstituindo-a, devidamente, à sua resolução inicial nas camadas finais. Veja-se o exemplo de Newell *et al.* [48], já referido, ao recuperar a alta resolução através de um processo *low-to-high* simétrico. Enquanto Xiao *et al.* [57] adota alguns *transposed convolution layers* para gerar representações de resolução superior. Tendo isto em conta, Sun *et al.* [40] apresentaram uma nova arquitetura apelada de ***High-Resolution Net* (HRNet)** (analisada propriamente na Subsecção 3.2.2), capaz de manter as representações das *features* com elevada qualidade ao longo de toda a rede.

Chen *et al.* [68] apresentaram uma nova estrutura de rede de nome ***Cascaded Pyramid Network* (CPN)**. Esta rede é dividida em duas fases: (1) ***GlobalNet***, *feature pyramid network* capaz de localizar *keypoints* comuns (*e.g.*, olhos e mãos) mas com falhas em cenários de oclusão visual, e uma (2) ***RefineNet*** para localização dos *keypoints* mais difíceis através da integração de todos os níveis de representação de *features* provenientes da *GlobalNet* junto com uma *online hard keypoint mining loss* (*training loss*) .

Os *datasets* para *multi-person pose estimation* e *tracking* são poucos diversificados e são imprecisamente anotados (foco na redução de tempo em anotação, principalmente em *frames* de vídeos, contudo a sua utilização não está indicada para treino em tarefas de *fully supervised learning* uma vez que estas exigem *dense annotations*) o que torna complica o desenvolvimento de modelos robustos capazes de lidar com problemas comuns, *e.g.*, *motion blur* e oclusões. Neste sentido, Umer *et al.* [69] propuseram uma abordagem que se baseia na correspondência entre *keypoints* para associar pessoas nos vídeos. Na verdade, em vez de treinar a rede para estimar a correspondências entre *keypoints* e as evidências do vídeo, é treinada num *datasets* com dimensão acrescida para HPE recorrendo a *self-supervised training*. Esta *top-down framework* usa a correspondência de *keypoints* para recuperar deteções de articulações dadas como perdidas através de informações temporais (associação de articulações em diferentes *frames* do vídeo).

Com base no desenvolvimento dos métodos baseados em ***Transformer***, concretamente *Vision Transformer* (ViT) (Subsecção 3.5.2), a sua aplicação em HPE tende a substituir as SOTA CNNs neste processo. Pesquisas iniciais [70] propõem um modelo apelidado de ***TransPose*** que utiliza *attention layers* para prever *heatmaps* dos *keypoints* e potencializar a melhoria ajustada desses mesmos *keypoints* em cenários de oclusão. Essencialmente, este modelo consiste em três componentes: (1) *CNN backbone* para obtenção de *low-level feature* na imagem, (2) um *transformer encoder* para obter interações espaciais maiores entre os *feature vectors* no devido local e (3) uma camada final para prever os *heatmaps*.

Seguindo Yang *et al.* [70], Li *et al.* [71] construíram um modelo baseado em *Transformer*, denominado de ***TokenPose***, focado na deteção das restrições anatómicas do humano (pormenor difícil de processar apenas com CNNs). Trata-se um modelo híbrido, Figura 2.12, com recurso a CNNs para extração e localização de *features* de baixo nível (*image feature maps*). A obtenção da sequência de *heatmaps* é conseguida por uma camada de convolução final 1×1. O restante processo é análogo ao proposto no ViT, desde a conversão destes *features maps* para vetores unidimensionais, projeção linear desta sequência e incorporação da posição 2D de cada *patch*. Ainda assim, na fase de treino, a entrada do *transformer encoder* é dividida em *visual tokens* e *target keypoints*, com a atribuição respetiva de um *token*. Estes últimos atuam como representação do que será a suposta representação

esperada dos *keypoints*.

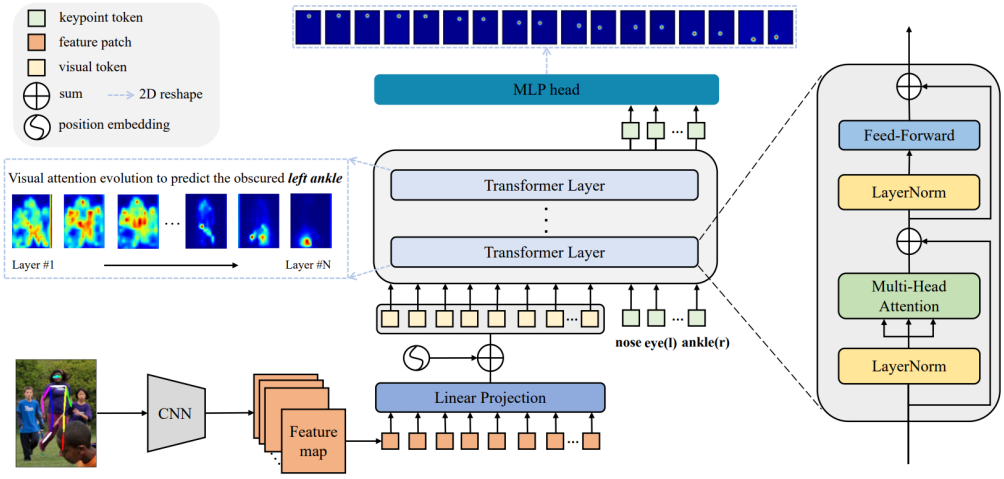


Figura 2.12: Ilustração da arquitetura do *TokenPose* [71].

Bottom-up methods

Por sua vez, os métodos *bottom-up* preveem diretamente todas as articulações 2D das pessoas presentes na imagem de entrada (*i.e.*, através da obtenção de *local features*) e, posteriormente, agrupa-as em "esqueletos" independentes (*i.e.*, através de algoritmos de associação). Esta última etapa, focada em agrupar as articulações detetadas no devido local revelou-se, especialmente em cenários complicados, uma tarefa desafiadora [36]. Tipicamente, o tempo de execução destes métodos é inferior, contudo, a sua performance pode ver-se comprometida em situações que se verifique maior sobreposição de pessoas. A *pipeline* genérica dos *2D multi-person PE bottom-up methods* encontra-se representada na Figura 2.13.

Figura 2.13: *Pipeline* genérica das abordagens *bottom-up* para *2D HPE* [2].

De forma a lidar com as duas etapas referidas, Pishchulin *et al.* [72] aplicaram um detetor de partes do corpo baseado numa *Fast R-CNN* (***Adapted Fast R-CNN***), apelidado de ***DeepCut***, para as puder categorizar. Estas mesmas partes foram reorganizadas e agregadas no devido local através de *Integer Linear Programming* (ILP) de forma a completar o "esqueleto". Foi desenvolvida uma *fully convolutional VGGNet* para estimar os *probability scoremaps* e diminuir a ambiguidade nas

partes do corpo detetadas. Apesar de prometedor e de se tratar de uma das primeiras abordagens *two-stage bottom-up*, a *DeepCut* revelou-se computacionalmente exigente.

Neste sentido, Insafutdinov *et al.* [73] introduziram ***DeeperCut***, focados na melhoria da abordagem SOTA do momento dedicada a *2D multi-person PE* (*DeepCut*). Os autores melhoraram o detetor de partes do corpo com uma implementação baseada no *backbone* ResNet [60], aparentemente melhor que a *VGGNet*. A seleção deste modelo foi guiada pelos excelentes resultados obtidos no *ImageNet Object Classification Challenge* e pela performance ao lidar com o problema dos *vanishing gradients*. Além disso, com vista a agrupar as partes num hipotética configuração válida de pose humana, foram introduzidos novos termos de *image-conditioned pairwise* (*e.g.*, ângulos das articulações). Por fim, uma estratégia de otimização incremental de forma a resolver algumas questões, levantadas pelos autores do *Integer Linear Programming* (ILP), resultando numa redução do tempo de execução de 4 a 5 vezes com melhoria na precisão da PE.

Apesar das diversas contribuições, a velocidade de processamento dos métodos *bottom-up* era reduzida, podendo apenas alguns serem executados em tempo real. Cao *et al.* [74] construiram um detetor denominado de ***OpenPose*** que recorre a *Convolutional Pose Machines* [61] para prever as coordenadas dos *keypoints* candidatos através de *heatmaps* e *Part Affinity Fields* (PAFs) (conjunto de campos vetoriais 2D com mapas vetoriais que codificam a posição e orientação dos membros) para associação aos *keypoints* de cada pessoa. A deteção das articulações e associação é feita em simultâneo, com base numa arquitetura dividida em dois ramos. A Figura 2.14 apresenta uma visão geral deste modelo. Uma imagem RGB, de tamanho $w \times h$ (Figura 2.14a), é sujeita a uma *feedforward network* (*two-branch* CNN) que simultaneamente prevê um conjunto de mapas de confiança 2D (S) da localização de partes do corpo (Figura 2.14b superior) e PAFs para associação destas (Figura 2.14b inferior). Nesta, cada pixel do campo codifica a posição e orientação do membro em causa (cada cor remete para uma orientação). Na Figura 2.14c um conjunto de *bipartite matchings* para associação das articulações candidatas e, finalmente, a combinação destas para criação das *full body poses* das pessoas na imagem (Figura 2.14d). A precisão deste modelo é mantida à medida que potencializa uma operação em tempo real independentemente do número de pessoas na imagem.

Baseado no *OpenPose* e motivado pela estrutura da *stacked hourglass network* [48], Newell *et al.* [75] introduziram *single-stage deep network* para, simultaneamente, obter a localização das articulações e agrupa-las devidamente segundo uma arquitetura que produz *pixel-wise predictions*. Contrariamente ao definido até então, estas tarefas seriam realizadas com *single-stage pipelines*. Para esse efeito são produzidos *heatmaps* para cada articulação e atribuído uma *embedding tag* (com base no pico

(a) *Input Image* (b) *Part Confidence Maps* e PAFs (c) *Bipartite Matching* (d) *Parsing Results*

Figura 2.14: Visão geral da *pipeline* do OpenPose [74].

de um *tag heatmap*) de modo a agrupa-las posteriormente através de uma rede, propriamente treinada. Este agrupamento é conseguido pela similaridade nos valores (números reais), *i.e.*, a distinção não ocorre pelo valor da *tag* mas sim pela diferença entre elas.

No decorrer deste processo, estruturas *multi-task* foram defenidas em métodos *bottom-up* para HPE [76]. Veja-se Papandreou *et al.* [76] ao introduzirem ***PersonLab*** para combinar PE e segmentação de pessoas na deteção dos *keypoints* e respetiva associação numa *deep convolutional network* tendo utilizado ResNet-101 [60] ou ResNet-152 [60] como *backbones*. A CNN do modelo deve prever: (1) *keypoint heatmaps*, (2) *short-range offsets*, (3) *mid-range pairwise offsets*, (4) mapas de segmentação de pessoas e (5) *long-range offsets*. Essencialmente, as três primeiras previsões devem ser usadas pelo *PE module* de forma a detetar as articulações do humano, enquanto as últimas duas, juntamente com o resultado da HPE são usadas no módulo de *instance segmentation* de forma a prever as *instance segmentation masks* das pessoas presentes na imagem. A performance do sistema é, desta forma, independente do número de pessoas na imagem.

2.3 *3D Human Pose and Shape Estimation Approaches*

3D HPE despertou maior interesse nos últimos anos ao propor-se a estimar a localização das articulações do corpo no espaço 3D a partir de uma imagem ou outros dados de entrada. Para o problema inverso (projeção de 2D para 3D) relativo à perda de uma dimensão várias soluções surgiram, algumas com aplicação de fusão de informação *e.g.*, produtos com *depth sensors*, *optical sensors* e *multiple cameras systems*. Apesar de tudo, todos estes sistemas operam em ambientes condicionados com *hardware* adicional. Soluções com recurso a uma única câmara apresentam uma maior diversidade de aplicabilidades, fruto das menores restrições impostas.

Comparativamente com *2D HPE*, o processo de *3D HPE* revela-se especialmente desafiador fruto da necessidade de estimar as informações de profundidade das articulações do corpo humano e da dificuldade de obtenção de dados de treino para

o efeito. Contrariamente a *2D HPE*, com maior facilidade na criação de anotações precisas, a obtenção de anotações 3D das articulações revela-se uma tarefa demorada e trabalhosa, pelo que o *labeling* manual não é praticável [2].

Posto isto, esta secção reúne um conjunto de métodos baseados em *Deep Learning* (DL) para *3D Pose and Shape Estimation (PSE)* a partir de imagens RGB ou *frames* de vídeos para *3D single person PE* e *3D multi-person PE*. A Figura 2.15 esquematiza a organização desta secção, de forma sucinta e organizada, com a apresentação dos métodos e variantes expostas.

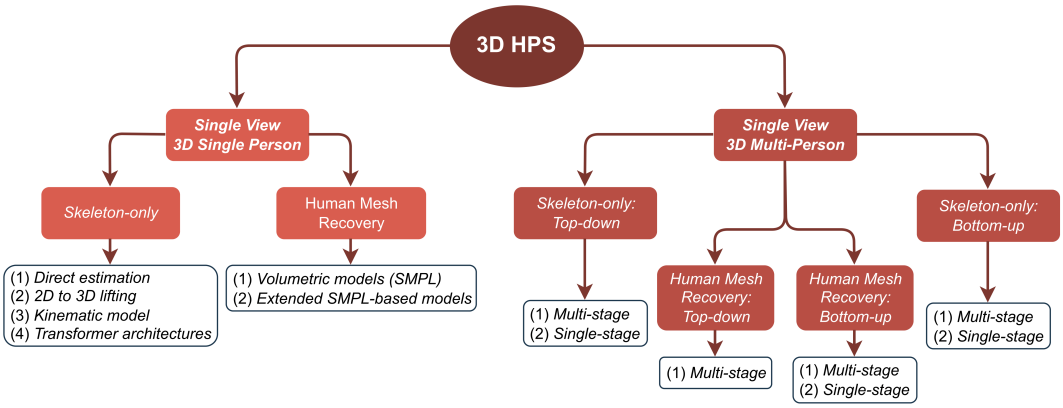


Figura 2.15: Organização dos métodos baseados em DL para estimar a pose humana 3D.

2.3.1 *Single View Single Person*

Masoumian *et al.* [35] reuniu mais de 150 artigos científicos representativos do desenvolvimento atual para DE com base em técnicas de DL, evidenciando os problemas críticos de um sistema monocular e aferindo a capacidade das redes neuronais, *e.g.*, *Deep Neural Networks* (DNNs), GANs e *Recurrent Neural Networks* (RNNs), em estimar a *dense depth maps* e *sparse depth maps*. O conhecimento retido neste documento complementa a análise aos métodos mais recentes focados em estimar a localização das articulações do humano a partir de um sistema monocular.

A eficácia verificada nos métodos para *2D HPE*, através um sistema monocular, permitiram a progressão em trabalhos focados em *3D HPE*. Contudo, apesar dos conhecidos problemas (*e.g.*, oclusão parcial ou total do corpo humano), esta tarefa vê-se fragilizada com questões adicionais, particularmente a ambiguidade da profundidade e a existência de dados de treino insuficientes. Estas questões originam um grave problema, nomeadamente a diversidade na localização das articulações humanas no espaço 3D, resultantes da projeção das poses inferidas na imagem [2].

Posto isto, esta subseção é dividida em duas categorias, sejam elas ***skeleton-only*** e ***Human Mesh Recovery*** **(HMR)** com as respetivas arquiteturas associadas e especificadas na Figura 2.15.

Skeleton-only

É possível afirmar-se que os métodos *skeleton-only* (ou *model-free*, em certos trabalhos) não empregam modelos do corpo humano para reconstruir a representação da malha corporal humana 3D. Assim sendo, estes podem ser divididos em duas abordagens: (1) ***direct estimation*** (*directly map prediction*) de uma imagem para pose 3D e (2) ***2D to 3D lifting*** com DE a partir de métodos intermédios de *2D PE* [2].

A Figura 2.16 esquematiza a forma como os métodos baseados em *direct estimation* inferem a posição das articulações humanas de uma imagem 2D, sem passos intermédios adicionais.

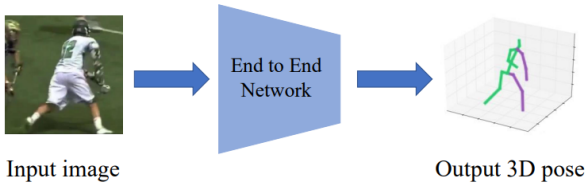
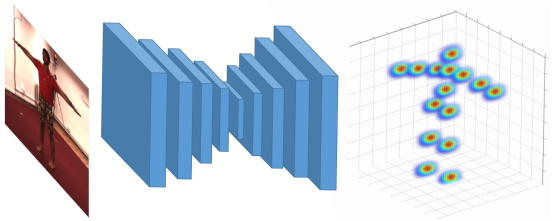


Figura 2.16: *Skeleton-only methods:* abordagem baseada em *direct estimation* [2].

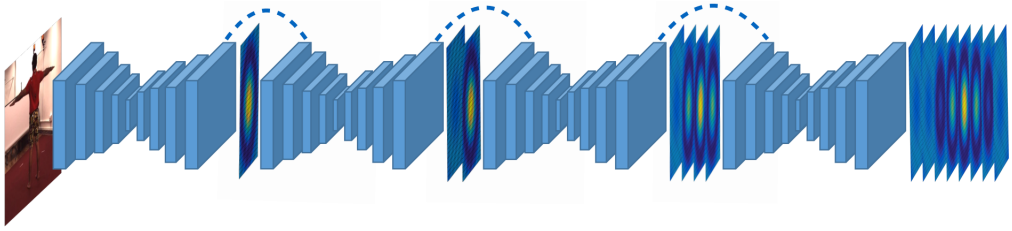
Li *et al.* [77] propuseram duas abordagens para *monocular 3D PE* com recurso a uma *deep convolutional multi-task network*. A primeira abordagem consistiu em treinar simultaneamente a rede para executar a tarefa de *pose regression* a partir de um conjunto de deteções numa estrutura *heterogeneous multi-task learning*. A segunda abordagem consistia num inicialização do *pose regressor* baseada numa rede pré-treinada para deteção de partes do corpo humano (*pose regression task*, usada sozinha). A entrada, para a execução de ambas as tarefas, será as regiões limitadas pelas *bounding boxs* que contem o humano em estudo. O objetivo da tarefa *pose regression* será estimar a posição dos *keypoints*, referentes às articulações, numa relação à posição da *root joint*. O *joint point regressor* é treinado de modo a minimizar a diferença quadrática entre a previsão e a posição verdadeira. Concretamente, são definidos um conjunto de tarefas de deteção associadas a um *keypoint* e a uma janela local. Como tal, o objetivo da tarefas de deteção será classificar onde, nesta janela, conterá (ou não) um *keypoint*. O processo de deteção é realizado com recuso à técnica de janela deslizante com tamanho de 10×10 e incrementos de 10 píxeis.

Apesar do sucesso do paradigma de uma aprendizagem de ponta a ponta, as melhores abordagens aplicavam-se em soluções de dois passos, *i.e.*, *Convolutional Network* (ConvNet) para localização 2D das articulações e uma etapa subsequente de otimização para recuperar a pose 3D. Neste sentido, Pavlakos *et al.* [78, 79] propuseram uma representação volumétrica para *3D HPE* unicamente através de uma ConvNet com contribuições associadas à validação da utilização de uma abordagem de aprendizagem ponta a ponta. Para tal, o espaço tridimensional sofre uma

discretização fina em torno da pessoa em estudo para depois, com recurso à *ConvNet*, prever a probabilidade, por *voxel*, da localização de cada articulação, Figura 2.17a. Este processo cria uma representação natural da pose 3D, com um desempenho otimizado face aos métodos de regressão anteriores. Ainda assim, de forma a aperfeiçoar a estimação inicial, os autores adotaram uma *coarse-to-fine prediction* de forma a aumentar, gradualmente, a resolução do volume de supervisão na terceira dimensão (profundidade) recorrendo a uma representação volumétrica. Na entrada da rede está uma única imagem *RGB* e uma saída de uma representação volumétrica 3D densa com probabilidades por *voxel* para cada articulação. Como referido, a rede, presente na Figura 2.17b, consiste em vários módulos *fully convolutional* supervisionados por um abordagem *coarse-to-fine*.



(a) Ilustração da representação volumétrica para *3D HPE*



(b) Ilustração da abordagem *coarse-to-fine volumetric* para *3D HPE*

Figura 2.17: Visão geral da *pipeline coarse-to-fine volumetric prediction* [78].

Adicionalmente, é percetível nas linhas a tracejado, o processo de fusão dos *heatmaps* intermédios com as *features* extraídas da imagem para criação da entrada de dados no próximo módulo convolucional. Este processo de supervisão permite obter resultados ainda mais precisos.

Ainda assim, motivado pelo desenvolvimento em *2D HPE*, abordagens *2D to 3D lifting*, capazes de inferir a pose humana 3D a partir da pose humana 2D, tornaram-se uma solução popular. A Figura 2.18 esquematiza a forma como os métodos ***2D to 3D lifting*** inferem a posição das articulações humanas, tipicamente em duas etapas. Numa primeira fase, modelos *off-the-shelf 2D HPE* são aplicados para estimar a pose 2D e, posteriormente, *2D to 3D lifting* é usado para obter a pose 3D [2]. Fruto da notável performance dos melhores detetores de pose humana 2D, as abordagens *2D to 3D lifting* geralmente superam as abordagens de *direct estimation*.

Li *et al.* [80] abordam a questão de *3D HPE* como um problema inverso com múltiplas soluções viáveis. O modelo em causa gera várias hipóteses de poses 3D a partir de poses 2D aplicando *ranking networks* para escolher a melhor pose 3D.

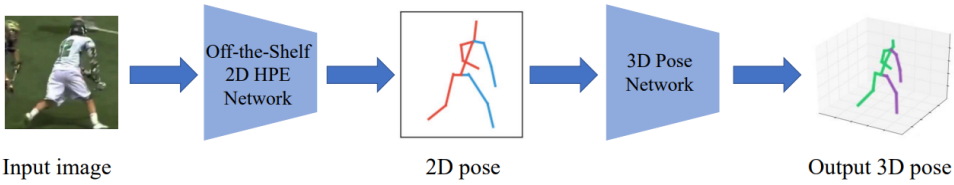


Figura 2.18: *Skeleton-only methods - abordagens 2D to 3D lifting* [2].

Novamente, a rede neuronal para PE 2D tem como *backbone* uma *stacked hourglass network* pré-treinada com o *MPII dataset* [43]. A Figura 2.19 apresenta a estrutura geral desta implementação baseada numa *deep two-stage network* capaz de gerar várias hipóteses para *3D human pose estimation*. Recorre-se a uma *Mixture Density Network* (MDN) para extração de *features* e a um gerador de hipóteses de poses 3D. Contrariamente às abordagens de DL existentes, focadas em minimizar o *negative log-likelihood* de distribuição Gaussiana unimodal, *i.e.*, erro quadrado médio, os autores procuraram estimar várias hipóteses minimizando o *negative log-likelihood* de uma *multimodal mixture-of-Gaussians*. À saída do modelo de mistura, obtém-se um conjunto de parâmetros que caracterizam os *kernels* Gaussianos, *i.e.* *mixing coefficients* (α), médias (μ) e variâncias (σ).

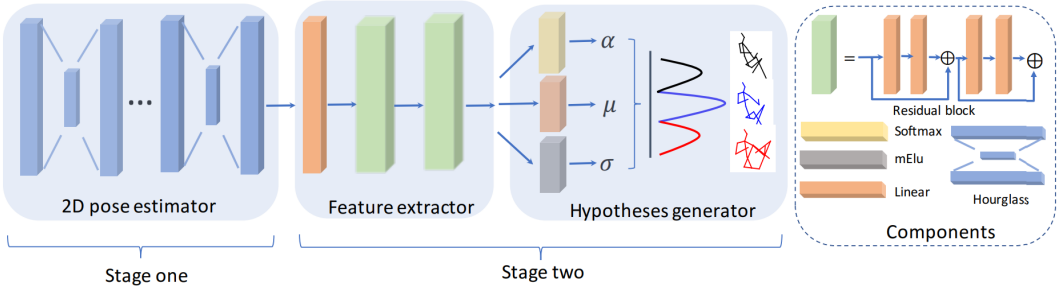


Figura 2.19: *Two-stage network for feature extractor and 3D pose hypotheses generator* [80].

Contudo, uma pose humana pode ser representada como um gráfico onde as articulações são os nós (*nodes*) e os ossos são as linhas de união (*edges*). Assim, *Graph Convolutional Networks* (GCNs) foram aplicadas ao problema de *2D-to-3D pose lifting* com desempenhos relevantes. O proposta de Zhou *et al.* [81] baseia-se numa *GCN* capaz de modelar a relação entre diferentes parte do corpo humano. A implementação procura solucionar os problemas de técnicas anteriores na dificuldade que a rede poderia ter em aprender diferentes relações entre articulações do corpo. Como tal, a ***Modulated GCN*** proposta consiste em duas componentes: (1) *weight modulation* para aumentar a recetibilidade na aprendizagem de diferentes *modulation vectors* de *nodes* diferentes e (2) *affinity modulation* para ajustar a estrutura do gráfico de forma a ser possível modelar diferentes *edges* além do estrutura óssea do humano.

Não obstante, abordagens mais recentes procuram a incorporação do conhecimento associado às restrições anatómicas do corpo humano com base no seu ***kinematic model***, *e.g.*, informação de conexão entre articulações humanas, propriedades de rotação individual das articulações e proporções no tamanho dos ossos, para realizar uma estimação, do esqueleto humano, lógica [2].

Nie *et al* [82] propuseram um método focada na redução da ambiguidade em prever a profundidade das articulações humanas por base na localização 2D das mesmas e numa hierarquia a dois níveis de redes *Long Short-Term Memory* (LSTMs). Num primeiro nível, o *skeleton-LSTM* foca-se em aprender noções de profundidade a partir de *features* globais do esqueleto humano, enquanto o *patch-LSTM* recorre às evidências da imagem cortada em torno da articulação detetada (*local features*). Num segundo nível, as *features* locais e globais são integradas para prever a profundidade das articulações. Naturalmente, a pose 3D é obtida pela agregação destes valores de profundidade com a pose 2D. Ambas as redes têm uma estrutura em árvore definida na relação cinemática do esqueleto humano. A Figura 2.20 ilustra o processo de treino dos dois níveis da rede e a estrutura geral da mesma. O primeiro nível, *skeleton-LSTM*, é pré-treinado com *2D-3D pose pairs* para prever a profundidade através de *features* globais do esqueleto humano, enquanto no segundo nível, *patch-LSTM*, prevê a profundidade a partir de evidências locais da *image patch* de partes do corpo humano. Desta forma, é possível afirmar que estrutura em árvore da rede é definida segundo as relações cinemáticas das articulações humanas. Um pormenor relevante, considerado pelos autores, encontra-se na atenção à redução do *overfitting* do *patch-LSTM*, nomeadamente com um treino com *multi-task loss*: *depth prediction loss* and *2D pose regression loss*.

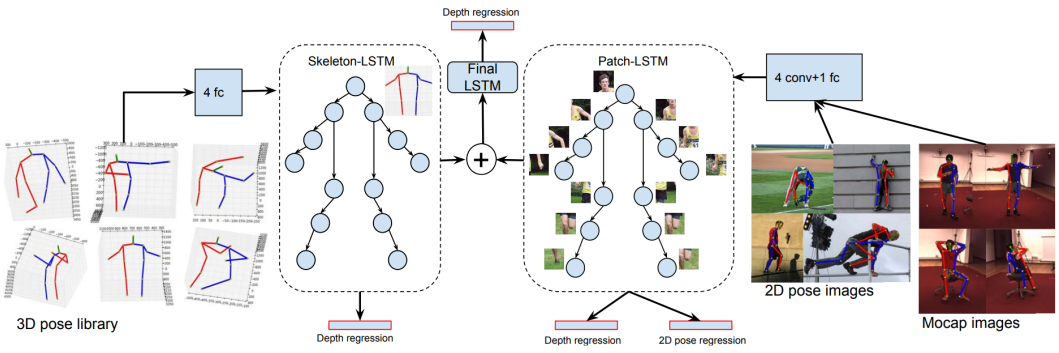


Figura 2.20: Visão geral da rede LSTM com dois níveis hierárquicos [82].

Não obstante, soluções atuais, baseadas na arquitetura dos ***Transformers***, destacaram-se na pressuposta tarefa de *3D HPE*, fruto do seu desenvolvimento na área de visão computacional [83]. A Secção 3.5 contempla um estudo generalizado à tecnologia, mantendo este estudo atualizado e num sentido em que se prevê a substituição das CNNs e RNNs pela adoção do mecanismo de *self-attention* em diversas

tarefas de classificação de imagem [84], deteção de objetos, segmentação e HPE [85].

Zheng *et al.* [85] apresentaram a primeira abordagem puramente baseada em *Transformers* para *3D HPE* em vídeos, sem uso de qualquer operação de convolução, apelada de ***PoseFormer***. Inspirado no recente desenvolvimento em ViT [84], os autores projetaram um *spatial transformer module* para extrair *features* com base em correlações das articulações do corpo humano e um *temporal transformer module* para captar correlações temporais entre *frames* numa dada sequência. A Figura 2.21 reúne os três módulos que compõe a arquitetura do *PoseFormer*. Com base no processo de *2D-to-3D lifting*, o primeiro módulo, *spatial transformer*, é alimentado com uma sequência de poses 2D provenientes do *2D pose detector* em causa, *i.e.*, *Cascaded Pyramid Network* (CPN) [68]. Um segundo módulo, *temporal transformer*, igualmente introduzido e, por fim, um *regression head module* para definir a posição 3D final centrada num *frame*.

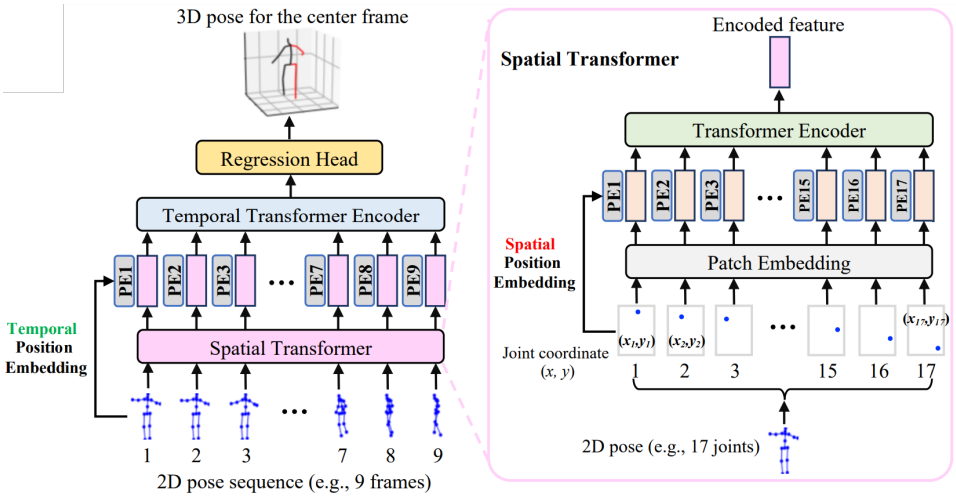


Figura 2.21: *Spatial-temporal transformer (PoseFormer) architecture* [85].

Human mesh recovery

Os métodos ***Human Mesh Recovery*** **(HMR)** (ou *model-based*) tipicamente empregam um modelo corporal paramétrico (anteriormente referido na Subsecção 2.1) ou *template* para estimar a *Human Pose and Shape* (HPS) em imagens. Por norma, os parâmetros estimados pela *3D HPS network* são fornecidos a um *model regressor* para reconstruir a malha corporal humana 3D, assim como esquematizado na Figura 2.22.

Para tal, os ***volumetric models*** são empregues na recuperação da malha corporal humana 3D, fornecendo informações adicionais relativas à forma do corpo humano [2]. O modelo ***Skinned Multi-Person Linear*** **(SMPL)** [19] tornou-se popular neste processo pela facilidade na incorporação em *rendering engines*.

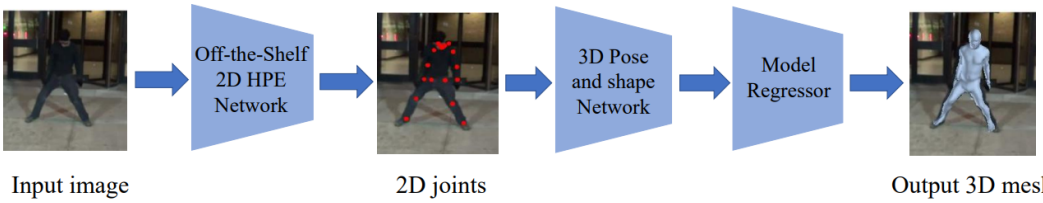


Figura 2.22: *HMR methods:* abordagem baseada *volumetric models* [2].

Pavlakos *et al.* [28] exploram o problema de estimar *3D HPS* a partir de uma única imagem. O projeto apresentado na *Computer Vision and Pattern Recognition* (CVPR) 2018 manteve uma estrutura baseada em *ConvNets*, apesar das suas falhas, *e.g.*, exigência de uma grande quantidade de dados para treino e potenciais *3D predictions* de baixa resolução. Para tal, a solução incorpora um *parametric statistical body shape model* (SMPL) como elemento central da *framework*. Deste modo, é possível obter resultados da malha 3D detalhados (6890 vértices) de uma forma direta pela simples estimação de um pequeno número de parâmetros, *i.e.*, 72 para a pose θ e 10 para o *shape* β. Ainda assim, o modelo permite o mapeamento da malha de vértices para as articulações do corpo através de um *linear regressor* (articulações expressas como uma combinação linear de vértices da malha). Adicionalmente, os autores verificaram que os parâmetros em causa podem ser calculados, com elevado grau de confiança, recorrendo apenas a *keypoints* 2D e *masks* (silhuetas dos humanos), dados de saídas típicos de *ConvNets* na análise 2D de um humano na imagem.

A Figura 2.23 ilustra a *framework* desenvolvida. Inicialmente, (a) a *ConvNet* (*Human2D*) estima os *heatmaps* 2D e as *masks* a partir de uma imagem RGB. As duas redes (b) estimam os parâmetros para do modelo estatístico *SMPL*. Concretamente, a rede *PosePrior* estima os parâmetros de pose (θ) a partir dos *keypoints* obtidos dos *heatmaps* e a rede *ShapePrior* estima os parâmetros do *shape* (β) a partir das silhuetas. Para isto, a metodologia de treino deve ser otimizada a cada uma das redes. Isto é, com recurso às anotações de imagens 2D (*e.g.*, *MPII Human Pose dataset* [43]) é possível treinar a inferência *image-to-2D* (rede *Human2D*) enquanto com base na instâncias de *human shape* do modelo paramétrico SMPL treinar a inferência *2D-to-3D shape*. Desta forma, será possível gerar uma malha 3D a partir dos parâmetros estimados e, naturalmente, (c) otimizar diretamente a superfície com base na *3D per-vertex loss* (melhor *correlation* com o erro 3D *vertex-to-vertex*) sem necessidade de imagens com *3D shape ground truth*. Por fim, com recurso a um *differentiable renderer* foi possível projetar a malha 3D novamente na imagem 2D. Esta projeção permite ajustar a rede pela comparação direta entre a malha gerada e as anotações 2D (*i.e.*, *keypoints* 2D ou *masks*).

De igual forma, Omran *et al.* [20] integraram um modelo corporal estatístico dentro de uma CNN para reconstruir uma malha corporal humana 3D numa nova

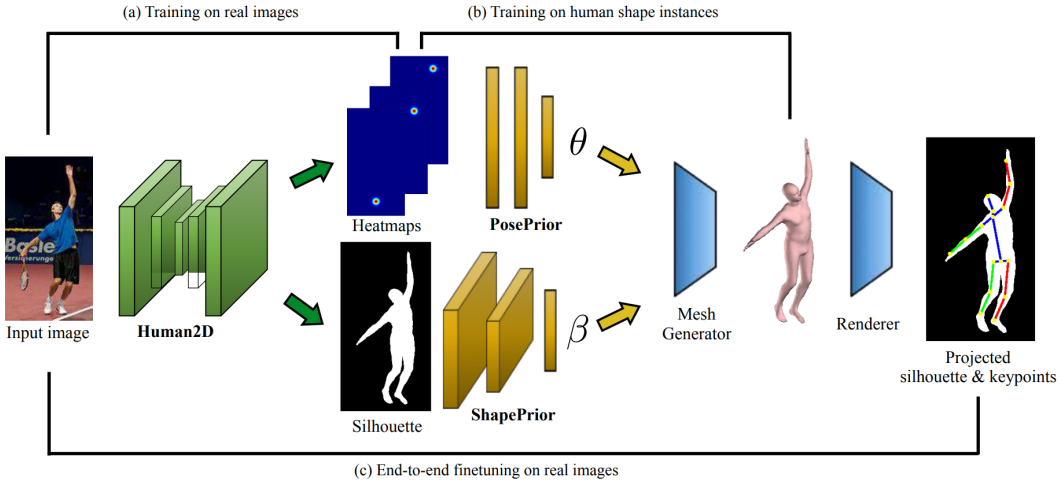


Figura 2.23: Representação da *framework* desenvolvida por Pavlakos *et al.* [28].

abordagem apresentada na 3DV 2018 e apelada de ***Neural Body Fitting* (NBF)**. A Figura 2.24 ilustra a *pipeline* da abordagem *two-stage* proposta. Numa primeira fase (*first stage*), dada uma imagem RGB com uma única pessoa, com recurso à *RefineNet* [86] (*bottom-up semantic body part segmentation* baseada na ResNet-101 [60]) são estimadas as partes do corpo da mesma (12 partes semânticas representadas a diferentes cores). A segunda fase (*second stage*) é dividida em duas partes: (1) *regression network* (ResNet-50 [60]) para processar os mapas de probabilidade das partes semânticas e gerar os 226 parâmetros SMPL (pose e *shape*), e um conjunto de camadas (não treináveis) para implementação do modelo SMPL e, posteriormente, obter uma projeção dos pontos referentes à pose 2D. Em suma, estas camadas são capazes de produzir uma malha 3D, e representação das articulações 2D e 3D.

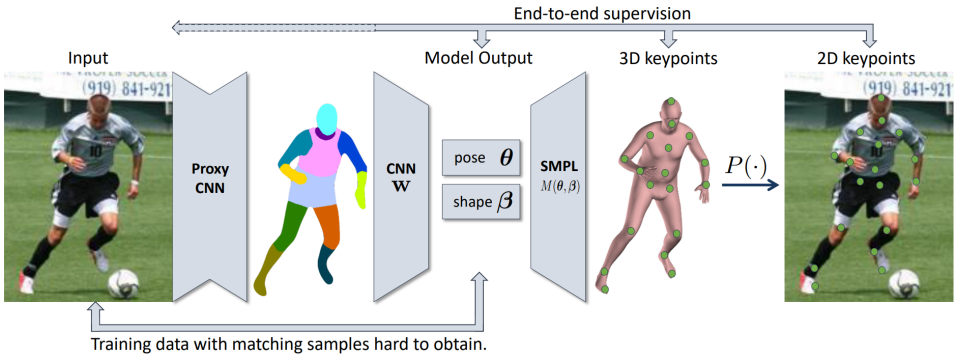
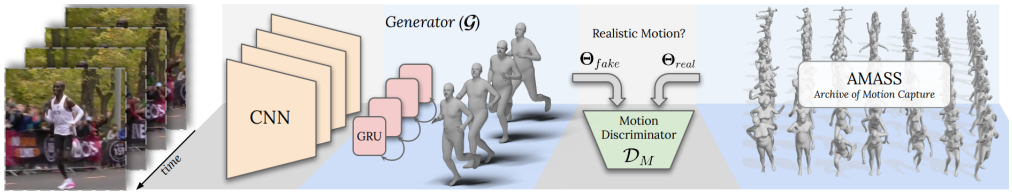


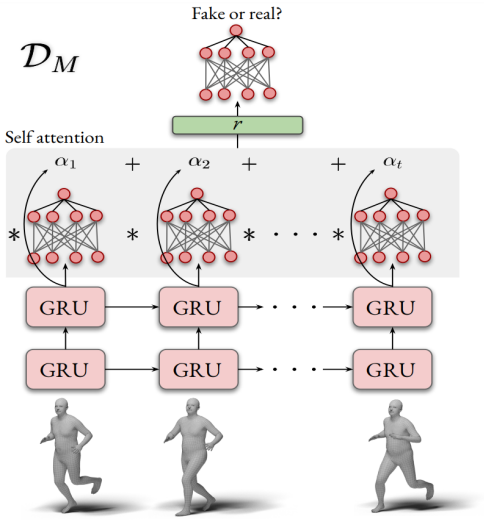
Figura 2.24: Representação da *framework* desenvolvida por Omran *et al.* [20].

Apesar destes progressos, métodos dedicados a inferência em vídeo falhavam na produção de sequências de movimento precisas e naturais fruto da falta de *ground-truth 3D motion data* para treino. Para resolver este problema, Kocabas *et al.* [21] apresentaram na CVPR 2020 o **VIBE**. Na verdade, a combinação de *indoor 3D*

datasets com vídeos, em trabalhos anteriores, tendo apenas a *ground-truth* 2D ou *pseudo-ground-truth keypoint annotations* arca com várias desvantagens, nomeadamente: (1) os *indoor 3D datasets* são limitados no número de candidatos, amplitude de movimentos e na complexidade da imagem, (2) número insuficiente de anotações em vídeo com *ground-truth 2D poses* para treino de DNNs e (3) existe pouca fiabilidade nos *pseudo-ground-truth 2D labels* para modelar o movimento humano no espaço. Sabendo isto, os autores, inspirado em Kanazawa *et al.* [87], aproveitaram um *large-scale motion capture (MoCap) dataset*, denominado de AMASS [88] (suficientemente rico para aprender um modelo representativo da locomoção humana), numa abordagem de *adversarial training* e anotações de *unpaired 2D keypoint* (uma vez que na **HPE baseada em vídeos** a localização destes *keypoint* pode mudar rapidamente entre *frames*) para um treino do *pose estimator*. A estrutura geral do *VIBE* encontra-se exposta na Figura 2.25a.



(a) Representação da arquitetura do VIBE



(b) Representação da arquitetura do *motion discriminator*

Figura 2.25: Visão geral da *framework* VIBE [89].

No processamento do vídeo, são extraídas as *features* de cada *frame* com recurso a uma CNN pré-treinada seguida de um *encoder* temporal e *body parameter regressor*. Neste processo, o *regressor* aprende a estimar as sequências de HPS 3D, uma vez que esta *framework* aproveita o AMASS *dataset* [88] para discriminar entre os movimentos humanos reais e os movimentos produzidos pela *temporal pose and shape regression network* (*generator*). Como tal, o *discriminator* introduz uma *weak supervision*. Tanto a implementação do *encoder* temporal como a do *motion*

discriminator recorrem a *Gated Recurrent Units* (GRUs) para compreender a natureza da sequência do movimento humano (*outputs* composto por *latent variables* representativas de informação temporal de anteriores e próximos *frames*). Ainda assim, o *motion discriminator*, Figura 2.25b, incorpora um *self-attention mechanism* de forma intensificar a contribuição (*i.e.*, atribuição de pesos diferentes) dos *frames* mais importantes numa dada representação, mostrando ser possível a criação de movimentos cinematicamente plausíveis sem *ground-truth 3D labels*. Toda o modelo é supervisionado por uma *adversarial loss* juntamente com as *regression losses* para minimizar o erro entre o previsto e os *ground-truth keypoints*, parâmetros pose e *shape*.

Complementarmente, surgiram várias soluções baseadas em ***extended SMPL*** *models* focado na redução das limitações do modelo SMPL, *e.g.*, elevado custo computacional e falha na projeção de referências faciais e mãos [2]. Neste sentido, ainda antes do lançamento do VIBE, Bogo *et al.* [90] propuseram o ***SMPLify***, apresentado na *European Conference on Computer Vision* (ECCV) 2016, numa estrutura *two-stage*. Concretamente, recorreram do método *DeepCut* [72] para inferir a localização 2D das articulações humanas numa abordagem *bottom-up*. Posteriormente, numa segunda fase, aproveitando as articulações 2D previstas, ajustam o modelo estatístico corporal SMPL para inferir a 3D HPS do humano (abordagem *top-down*). Ainda assim, os resultados do HPS são diretamente otimizados através de uma *loss function* representativa do erro entre as articulações 3D projetadas e as articulações 2D estimadas pela CNN. De forma a automatizar complementarmente o método, *i.e.*, distinção dos parâmetros do *shape* das mulheres e dos homens, introduziram um *gender-neutral model*, fulcral nos momentos em que o género ser desconhecido. Na eventualidade de ser definido o género aplica-se um *gender-specific model* para melhores resultados.

Pavlakos *et al.* [91] apresentaram, na CVPR 2019, um novo modelo apelidado de **SMPL-X** capaz de prever, além do modelo 3D da pose humana, todas as articulações nas mãos assim como *landmarks* faciais. Seguindo a abordagem do *SMPLify*, são obtidas *features* 2D, com informação descritiva do corpo, mão, pés e cara, recorrendo a uma abordagem *bottom-up* (*OpenPose* [41]) e, posterior, otimização dos parâmetros do modelo SMPL-X de forma a corresponder a estas *features* com um novo método denominado de ***SMPLify-X***. Em causa estão melhorias significantes no modelo inicial *SMPLify*, nomeadamente com o treino de uma nova rede neuronal para inferência da pose 3D, ***VPoser***, utilizando um *large-scale MoCap dataset*, *e.g.*, AMASS *dataset* [88]. Não obstante, foi definido um novo penalizador de colisões com precisão acrescida e treinado um novo detetor de género para, automaticamente, compreender qual modelo corporal deve ser utilizado (masculino, feminismo ou neutro). Segundo os autores, o *SMPLify* apresentava performances baixas na estimação dos parâmetros do modelo SMPL, como tal, numa implementação deste

modelo em PyTorch, verificou-se um aumento de velocidade em pelo menos 8 vezes.

Kolotouros *et al.* [29] abordou o problema de HPS como uma "colaboração" de métodos entre os dois paradigmas que o descrevem, nomeadamente: (1) métodos baseados em otimização focados em ajustar o modelo de corpo paramétrico às observações 2D de forma iterativa (geralmente modelos precisos mas com tempos de inferência superiores e sensível à sua inicialização) e (2) *regression-based methods* que recorrem a DNNs para estimar diretamente os parâmetros do modelo a partir dos píxeis da imagem (tendem a ser menos precisos mas com menor tempo de inferência, além da exigência de métodos de supervisão). Concretamente, o ***SMPL oPtimization IN the loop* (SPIN)**, apresentado na ICCV 2019, concilia o melhor dos dois cenário *i.e.*, realizar uma *direct estimation* (HMR) para uma inicialização confiante do modelo de otimização iterativa (maior velocidade e precisão de adaptação) enquanto o ajuste preciso ao nível do pixel na otimização iterativa (aplicando *SMPLify*) poderá atuar como modelo de supervisão para a rede. Ou seja, a *deep network* inicializa a rotina de otimização iterativa, responsável por ajustar o modelo às articulações 2D previstas e, subsequentemente, o modelo ajustado é usado para supervisionar a performance da rede. Deste modo, podemos verificar um sistema com auto-aperfeiçoamento uma vez que as melhores estimativas geradas na rede podem levar a melhores soluções de otimização e, portanto, uma supervisão melhorada da própria rede. Ainda assim, pode afirmar-se que este projeto redefiniu um nova limite de performance dos modelos SOTA, contudo o método é sensível a oclusões parciais, o que exigirá progressos em trabalhos futuros, principalmente pelo recurso a técnica de *data augmentation*.

Osman *et al.* [92] desenvolveram ***Sparse Trained Articulated Regressor* (STAR)**, apresentada na ECCV 2020, como um modelo compacto, completo e potencial substituto do modelo SMPL, em resposta às limitações impostas pelo mesmo. O elevado número de parâmetros do SMPL, proveniente de *global blend shapes* e representativo de uma *dense pose-corrective* relaciona vértices da malha com todas as articulações na *árvore cinemática humana*. Para solucionar estas correlações de longo alcance, os autores do STAR definiram *pose correctives* por articulação e procuraram perceber qual subconjunto de vértices da malha é influenciado pelo movimento da articulação (fatorizarão da deformação dependente de pose num conjunto de *spatially local pose-corrective blendshape functions*). Naturalmente, é esperada uma deformação mais realista, além de que reduz significativamente o número de parâmetros do modelo SMPL em cerca de 20%. Não obstante, foi revisto a implementação do modelo estrutural do indivíduo, uma vez que pessoas com diferentes *shapes* deformam-se de forma diferente. No caso, os autores relacionaram esta potencial variação de deformação com a pose humana e o Índice de Massa Corporal (IMC). Como tal, foi necessário treinar novamente a rede com 10000 *scans* de homens e mulheres para que seja possível captar o maior número de variações na população

em geral.

Apesar deste progresso significativo, Kocabas *et al.* [30] verificou que estes métodos permanecem sensíveis à oclusão parcial podendo induzir em previsões erradas. Este cenário é visível na Figura 2.26, nomeadamente com a circulação de um pequeno *occluding patch* a cinzento, o que leva a previsões erradas no SPIN [29] (Figura 2.26 (b,c)) e ao aumento considerável do erro médio por articulação (Figura 2.26 (d) com zonas a vermelho evidenciando uma maior sensibilidade a oclusão). Por sua vez, o modelo desenvolvido e apresentado na ICCV 2021, apelidado de ***Part Attention REgressor* (PARE)**, manteve-se praticamente indiferente a estes dois cenários de oclusão.

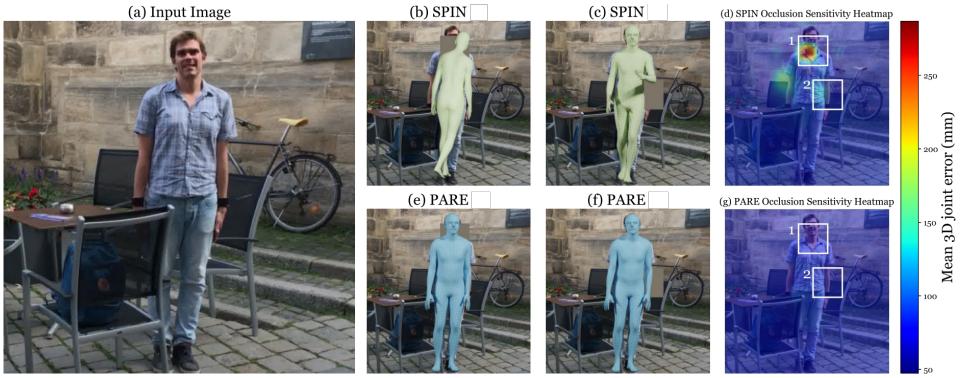


Figura 2.26: *Occlusion Sensitivity Analysis: comparação entre SPIN e PARE* [30].

Contrariamente a métodos anteriores, dependentes de *features* globais obtidas da imagem, este mecanismo de atenção suave é capaz de prever uma máscara de atenção (*body-part-guided*) de forma a explorar as informações sobre a visibilidade de partes do corpo individuais, enquanto aproveita as referências providencias pelas partes vizinhas para prever potenciais zonas ocultas. Assim sendo, o PARE possuiu duas tarefas, originando duas ramificações na rede. Para tal, recorre a uma CNN (com *backbone* ResNet-50 [60]) para extrair *features* volumétricas, *i.e.*, *3D body branch* responsável pela determinação dos parâmetros SMPL e *2D part branch* onde estima os pesos de atenção para cada parte do corpo (atribuição de *labels* para cada parte do corpo segmentada). Naturalmente, como um modelo de supervisão, o PARE utiliza a segmentação em partes, análogo à *framework* desenvolvida por Omran *et al.* [20], como *soft attention masks* de forma a que cada articulação na árvore cinemática do *SMPL* tem a sua corresponde parte do corpo atribuída.

Li *et al.* [23] propôs uma solução, apresentada na CVPR 2021, que procura atenuar a não linearidade na determinação dos parâmetros necessários para construir uma malha 3D completa. O **HybrIK** transforma diretamente as articulações 3D precisas, obtidas no referencial da câmara, para rotações relativas de partes do corpo, *i.e.*, com vista na reconstrução da malha 3D humana decorre uma decomposição das partes do esqueleto envolventes em rotações *twist* (rotação sobre o eixo longitudinal

do membro) e *swing* (rotação no plano). Este modelo decompõe a rotação de um corpo rígido em duas componentes separadas: (1) componente *twist* que descreve uma primeira rotação em torno do eixo fixo (1-*Degrees of Freedom* (DoF)) e (2) componente *swing* que descreve a rotação em torno de eixo capaz de mudar de direção.

Uma análise superficial à arquitetura da *HybrIK*, Figura 2.27, permite compreender os vários conceitos associados. A cada imagem RGB são obtidas as representações das articulações 3D através de *heatmaps* 3D gerados por *deconvolution layers* numa ResNet-34 [60]. Por sua vez, os parâmetros do *shape* (β) e o *twist angle* (Φ) são determinados por análise visual nas camadas *Fully Connected* (FC) enquanto o ângulo *swing* é obtido analiticamente. Os resultados são enviados para o bloco *HybrIK* para determinação dos parâmetros da pose (θ). Com todos o parâmetros conhecidos é então possível reconstruir a malha M e a pose Q através de um *linear regressor* ou *Forward Kinematics* (FK). Embora o processo de estimação cinemático das poses seja capaz de gerar resultados anatomicamente plausíveis, em certas situações são verificados movimentos fisicamente impossíveis com artefactos, *e.g.*, *foot sliding*, interseções humano-cenário e *jitter*.

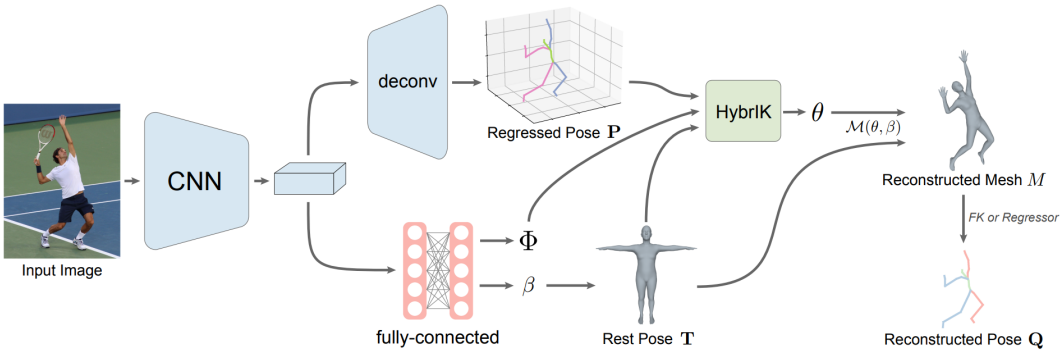


Figura 2.27: Visão geral da solução analítico-neural do HybrIK [23].

Recentemente, Zhu *et al.* [93] desenvolveram uma abordagem híbrida que aproveita os pontos fortes dos métodos baseados em imagens RGB (*shape* preciso) e a informação proveniente do movimento humano. Para tal, propuseram um *refiner module* passível de ser integrado nos métodos baseados em imagens ou vídeos existentes como um *backbone* genérico, *e.g.*, *HybrIK framework* [23]. Concretamente, apresenta-se como uma implementação a dois estados dividida num pré-treino e ajuste fino. A primeira fase dedica-se ao treino de um *motion encoder* (***Dual-stream Spatio-temporal Transformer* (DSTformer)**) para recuperar a movimento 3D a partir dos esqueletos humanos 2D parcialmente determinados (estimação dos esqueletos 2D de diversos *motion datasets* para a adição de mascaras aleatórias e ruído de forma a introduzir perturbações temporais e espaciais) levando a mesma

a inferir as estruturas humanas 3D durante transições temporais repentinas (*temporal Multi-Head Self-Attention* (MHSA)) e recuperar potencias falhas por oclusão ou problemas intrínsecos (*spatial* MHSA), *i.e.*, interligações articulares, restrições anatômicas e dinâmica temporal. A segunda fase aufere o *human motion encoder*, universal com a adição de alguns *layers* para *downstream tasks*, *i.e.*, 3D PE, reconhecimento de ações e recuperação da malha corporal. Os resultados experimentais em vários *benchmarks* demonstraram a versatilidade e adaptabilidade em métodos atuais.

2.3.2 *Single View Multi-Person*

Assim como na Subsecção 2.2.2, os métodos de *3D multi-person HPS* a partir de uma única imagem RGB ou vídeo também podem ser categorizados em duas abordagens: *top-down* e *bottom-up*.

Skeleton-only: top-down methods

Tipicamente, os métodos *3D multi-person PE top-down* fazem a deteção dos humanos presentes na imagem para, com recurso a uma detetor de pose 3D, determinar as coordenadas absolutas das articulações que definem o indivíduo. Num processo adicional, pode ocorrer uma projeção das coordenadas, inicialmente definidas, no referencia da câmara, para o referencial do mundo. A *pipeline* geral destes métodos encontra-se representada na Figura 2.28.

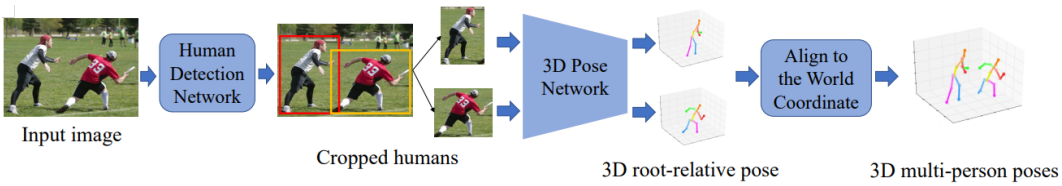


Figura 2.28: *Pipeline* genérica das abordagens *top-down* para *3D HPE* [2].

Moon *et al.* [26] apresentaram, na ICCV 2019, uma abordagem ***multi-stage top-down*** para determinar a pose 3D de várias pessoas presentes numa imagem RGB. O método em causa segue a estrutura presente na Figura 2.29 sendo que, primeiramente, deteta as *bounding boxes* em torno das pessoas nessa imagem com recurso à ***DetectNet***. Posteriormente, a informação posicional dos humanos na imagem alimenta a ***RootNet***, desenvolvida de forma a estimar as coordenadas 3D absolutas (relativas ao centro ótico da câmara) para cada indivíduo e ajustar a área efetiva para, posteriormente, estimar a profundidade (estratégia *camera distance-aware*). Em paralelo, uma terceira rede, apelada de ***PoseNet***, estima individualmente a *root-relative 3D pose*. O culminar desta informação deve gerar uma representação espacial nas coordenadas do referencial do mundo, uma vez que decorreu uma

adaptação no processamento rede com base na distância estima entre a câmara e o humano.

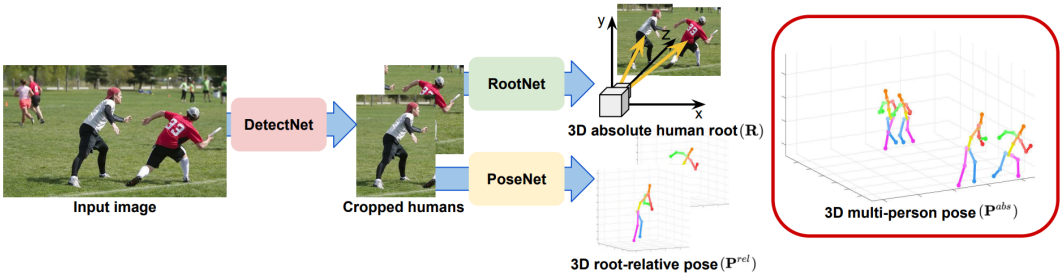


Figura 2.29: Visão geral da *pipeline* da *framework* proposta por Moon *et al.* [26].

A solução introduzida por Li *et al.* [94] na CVPR 2021, apelada de HMOR, procura melhorar a precisão da tarefa de *3D PE*, baseando-se nas relações espaciais (relações de profundidades e ângulos) relativas entre articulações de cada pessoa. Esta informação é disposta hierarquicamente, *i.e.*, *instance* → *part* → *joint*, permitindo a perceção da distância relativa entre articulação e em que ordem aparecem. Desta forma, a rede neuronal encontra-se disposta em dois estados (Figura 2.30). Um primeiro estado onde uma rede neuronal, com um *backbone* ResNet-50 [60], extrai *Region of interest (RoI) features* e o mapa de profundidade inicial. Esta informação é entregue a dois módulos, *i.e.*, ***PoseHead*** *e* ***DetHead***, responsáveis por estimar a posição relativa das articulação de cada humano e obter as *bounding boxes* em torno do humano a partir do *Region of interest* (RoI), respetivamente. E um segundo estado, onde um módulo conhecido por ***DepthHead***, recorre ao mapa de profundidade inicial e às *RoI features* para gerar um valor residual de modo a ajustar a profundidade final das articulações. Finalmente, o *HMOR*, como modelo de supervisão, realiza o ajuste definitivo das poses 3D finais.

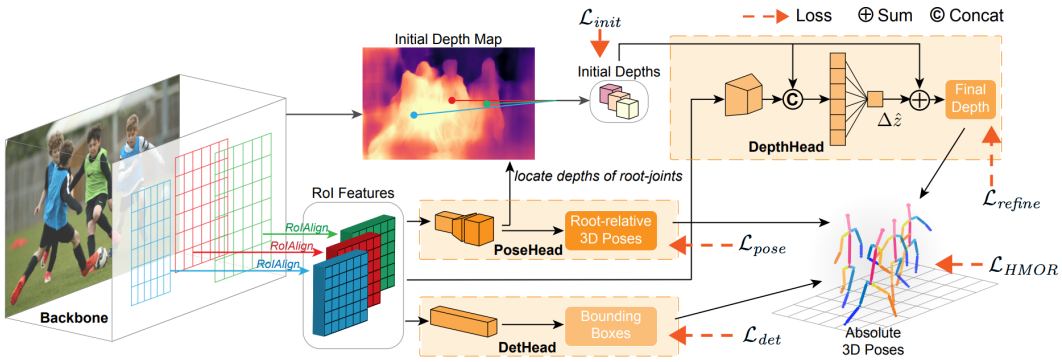


Figura 2.30: Arquitetura proposta por Li *et al.* para integração da solução HMOR [94].

Por sua vez, Benzine *et al.* [95] propuseram uma abordagem *single-stage* apelada ***Pose estimAtioN and Detection Anchor-based Network* (PandaNet)** para *multi-person 3D PE* a baixa resolução. Na verdade, o modelo permite a obtenção

da pose 2D e 3D de pessoas com diferentes tamanhos numa única passagem pela rede. Contrariamente aos métodos anteriores, que trabalham sobre cenários que possibilitam elevada resolução para cada indivíduo, esta estratégia, *Pose-Aware Anchor Selection*, baseada em poses 2D sobrepostas, procurou resolver o problema de sobreposição ao descartar ancoras ambíguas (sobreposição de ancoras na representação da pessoa) durante a inferência. Apesar dos resultado promissores, o artigo não menciona diretamente como lida com oclusão, referindo apenas que não recorre a qualquer tipo de pós-processamento uma vez que a rede prevê a pose 3D completa para cada *bounding box*, fruto de um treino com *dataset* de diferentes níveis de oclusão. Apesar de tudo, os *benchmark* realizados em três *datasets* diferentes, JTA [96], CMU-Panoptic [97] e MuPoTS-3D [98] mostraram que o modelo superou os resultados de SOTA em termos de precisão e eficiência, evidenciando a sua robustez em situações de oclusão por objetos ou pessoas.

Skeleton-only: bottom-up methods

Por sua vez, os métodos *3D multi-person PE bottom-up* fazem, primeiramente, a deteção da localização de todas as articulações e respetivos mapas de profundidade. Posteriormente, associam-se as partes do corpo de cada pessoa de acordo com *root depth map* e *part relative depth map*. Naturalmente, um dos maiores desafios destas abordagens prende-se na dificuldade em agrupar as articulações detetadas que a cada pessoa pertencem. A *pipeline* geral destes métodos encontra-se representada na Figura 2.13.

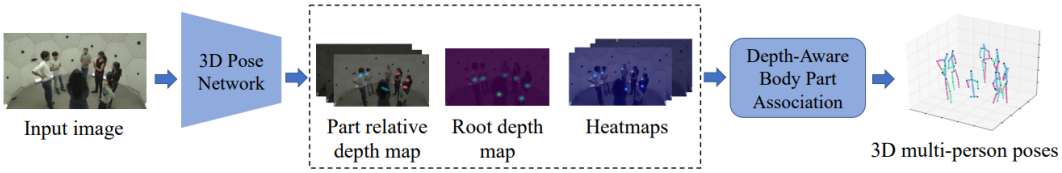


Figura 2.31: *Pipeline* genérica das abordagens *bottom-up* para *3D HPE* [2].

Mehta *et al.* [8] propuseram uma abordagem em tempo real a mais de 30 *Frames per Second* (fps) numa estrutura a três estados capaz de operar em cenários genéricos com oclusão por objetos ou pessoas. Concretamente, o método apelidado de ***XNect***, num primeiro estado recorre à CNN proposta, ***SelecSLS Net*** para estimar rapidamente a pose 2D e a pose 3D intermédia visíveis (*per body joint*). Numa segunda fase, uma *lightweight FC network*, executada em paralelo para cada pessoa detetada, reconstrói a pose 3D completa, incluindo articulações ocultas (noção de contexto global, fruto de uma *prior 3D pose structure* aprendida). Por fim, numa terceira fase, determina a localização relativa ao centro ótico da câmara e ajusta, através de uma parametrização do ângulo da articulação, o esqueleto cinemático

de cada indivíduo (*temporally coherent motion*). De facto, revelou-se um projeto promissor com uma precisão e tempos de inferência que redefinem o SOTA.

Apesar do tempo de inferência superior, Fabbri *et al.* [99] propuseram uma solução apelada de ***Learning on Compressed Output*** **(LoCO)** capaz de processar imagens *full HD* com mais de 50 pessoas a 8 fps. O método recorre a *volumetric heatmaps* de alta resolução para determinação da localização das articulações de forma a reduzir consideravelmente o tamanho desta representação. No centro do método está um *Volumetric Heatmap Autoencoder* (VHA) (representado na Figura 2.32, *Encoder* e), uma *fully-convolutional network* responsável pela compressão dos *ground-truth heatmaps* em representações intermédias mais densas/menores, mantendo as *features* relevantes (posição dos picos da distribuição Gaussiana nas várias articulações) enquanto reduz os requisitos computacionais do algoritmo em geral. Por sua vez, um segundo módulo, o *Code Predictor* (representado na Figura 2.32 f) terá sido treinado para reduzir as MSE *losses* entre as *alternative ground-truth representation* e a saída da rede em causa. Finalmente, o *decoder* (Figura 2.32 d), numa arquitetura espelhada do *encoder* e alimentado com a representação comprimida, recuperar a representação original, nomeadamente a estimação da localização 3D das várias articulações. Para tal, a associação de articulações é feita tendo por base uma distância heurística, começando pela deteção das cabeças (*i.e.*, *keypoint* com maior confiança) e conectando, posteriormente as restantes $(N-1)$ articulações pela seleção das que se encontram mais perto em termos de distância Euclidiana 3D. O sistema mostrou ser robusto na operação em diferences cenários com diferentes graus de oclusão fruto de uma aposta numa abordagem baseada *encoder-decoder* e no potencial providenciado pela representação volumétrica dos *heatmaps*.

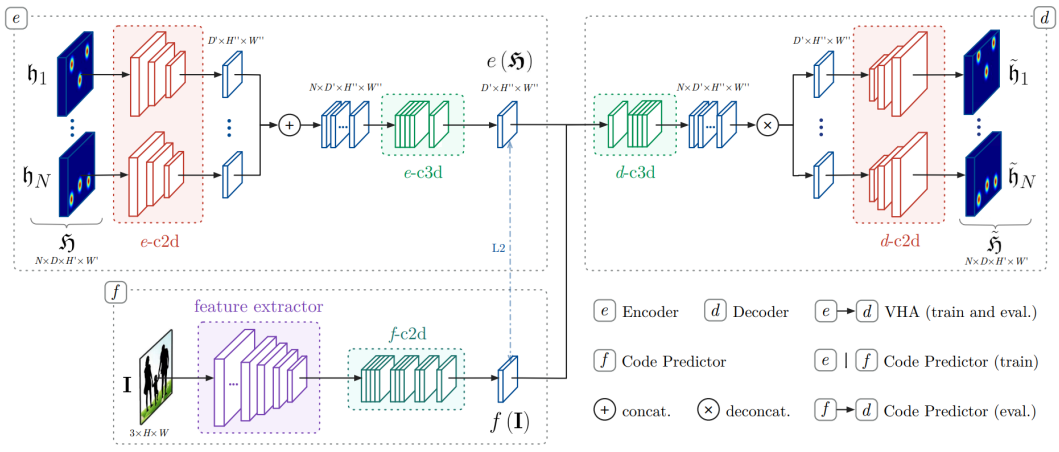


Figura 2.32: Representação da *pipeline* da implementação LoCO [99].

Zhen *et al.* [31] apresentaram o ***Single-Shot Multi-Person Absolute 3D Pose Estimation*** **(SMAP)** e, contrariamente a sistemas anteriores que dependiam de *multi-stage pipelines*, este método obtém diretamente a pose 3D a partir

de uma imagem RGB numa única passagem pela rede. O método proposto recorre a uma *Hourglass network* como *backbone* tendo sido modificada numa estrutura de aprendizagem *multi-task* de forma a estimar as várias representações presentes na Figura 2.33. Como tal, esta CNN faz, simultaneamente, a previsão da informação 2D (*keypoint heatmaps* e PAFs), *root depth map* e *part relative-depth map* (arquitetura projetada para análise de *features* em toda a imagem em vez de apenas em torno de uma determinada *bounding box*, *e.g.*, tamanhos de corpo, *scene layouts*, e relacionamentos interpessoais) que reúne a profundidade relativa entre duas articulações da mesma parte do corpo. Por suas vez, com base no PAFs, os *2D keypoints* detetados são associados à pessoa em causa através de um algoritmo *part association* (avaliação sobre oclusão interpessoal e restrições do comprimento ósseo com base na profundidade). Resultado desta associação, juntamente com os *root depth map* e *part relative-depth maps* são recuperadas as poses 3D. A *RefineNet* referida é opcional, contudo a sua aplicação destina-se ao refinamento acrescido das poses 3D previstas, assim como a determinação de potenciais *keypoints* invisíveis. O método foi avaliado em vários *benchmark* e atingiu, à data, a SOTA performance em termos de precisão e velocidade.

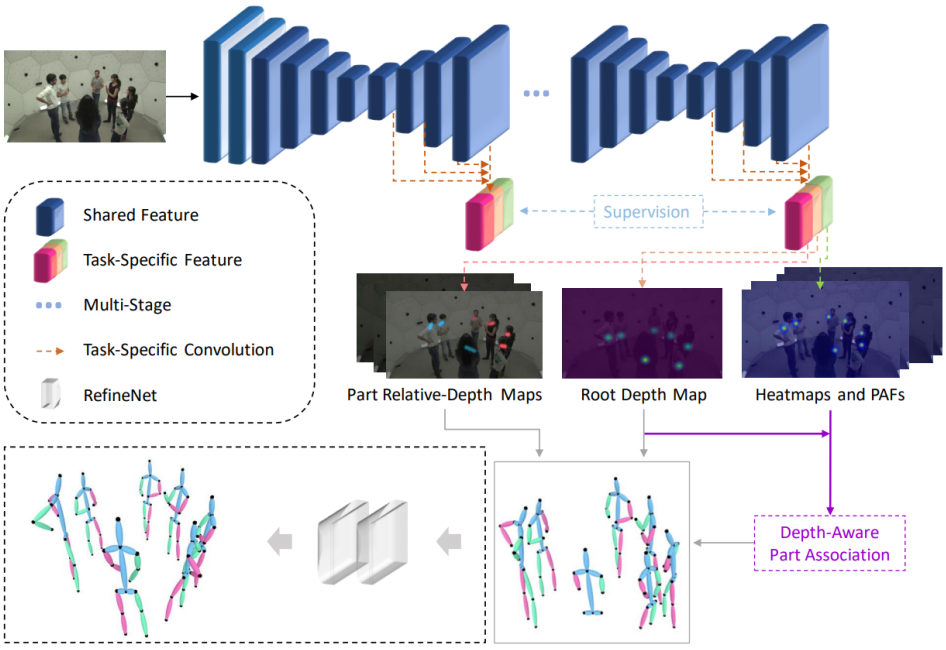


Figura 2.33: Visão geral da abordagem SMAP [31].

Human mesh recovery: top-down methods

Naturalmente, *3D HPS* através de única imagem RGB representa um maior desafio comparativamente à estimação única das articulações humanas na forma de esqueleto, especialmente em momentos que envolva interação entre pessoas fruto da

ambiguidade introduzida pela oclusão humana. Tipicamente, as abordagens *top-down* detetam, primeiramente, o humano na imagem em causa (identificados com *bounding boxes*) e posteriormente estimam os parâmetros da malha 3D para cada indivíduo (similar à *pipeline* de uma *single person 3D mesh reconstruction* com *loss functions* dedicadas ao cenário com densidade de pessoas) [100].

Neste sentido, Jiang *et al.* [9] optaram por rever a *pipeline* das abordagens *top-down* de forma a atenuar as causas de erros recorrentes que levam a resultados incoerentes, *e.g.*, interseção e inconsistência na profundidade entre as pessoas no cenário fruto de uma reconstrução individual independente. Para tal, os autores comprometeram-se a desenvolver uma rede que procura lidar com estas situações. Trata-se de uma *Faster R-CNN* [18] (ResNet-50 [60] como *backbone* para determinar todas as pessoas na imagem) alterada para obter os parâmetros do modelo SMPL (parâmetros pose θ e *shape* β da corresponde *bounding box* i) assim como os parâmetros da câmara ajustados à posição de cada pessoa no cenário $\boldsymbol{\pi}_i = \left(s_i, t_{x(i)}, t_{y(i)}\right)$, tal como Kanazawa *et al.* [87]. A contribuição do projeto encontra-se na incorporação de duas restrições durante a fase de treino que promovem uma reconstrução coerente das pessoas no espaço, nomeadamente uma *interpenetration loss* para evitar sobreposição de humanos e uma *depth ordering-aware loss* para reduzir a diferença entre a renderização da malha prevista de todas as pessoas e das *annotated instance masks* produzindo uma noção de profundidade própria. Apesar destes sinais de supervisão, esta abordagens ambiciosa é limitada comparativamente às mais recentes *top-down frameworks*.

Choi *et al.* [25] realçaram a dependência dos ambientes internos e, por vezes, controlados nos métodos existentes, propondo uma *framework* a dois estados, que combina uma *2D PE network* com uma *3D mesh regression network* apelada de ***3DCrowdNet***. Numa primeira fase, a rede determina a localização 2D das articulações humanas, contrariamente aos outros métodos que se focam na geometria 3D (dispensa de um *MoCap dataset* com *3D labels*), sendo o treino baseado em *datasets* com ambientes reais e com imagens repletas de multidões de humanos (robustez fulcral para performance da *3DCrowdNet*). Numa segunda fase, o *joint-based regressor* proposto, capaz de distinguir as *feature* da pessoa alvo e excluir as restantes pessoas não relevantes. De facto, este *regressor* é capaz de manter a perceção espacial da pessoa alvo num *deep CNN feature map* que deverá ser combinado com localização 3D das articulações previstas. Com esta informação, são estimados os parâmetros do modelo humano. Para o seu treino, os autores recorreram ao *CrowdPose dataset* [101] , tendo usado outros *datasets* para aumentar a eficiência e robustez do mesmo (*e.g.*, MSCOCO [102], MPII [43]), enquanto mantêm o desempenho para situações controladas (além de multidões e potenciais cenários de oclusão) com o 3DPW [103] e *Human3.6M* [104] *dataset*. Ainda assim propuseram o *3DPW-Crowd dataset* que reúne um conjunto de desafios em ambientes externos (espaços lotados

de pessoas no mundo real), como sobreposição de *bounding boxes* e oclusão entre pessoas. Os autores mostraram que a sua abordagem redefiniu novos resultados, ultrapassando os métodos SOTA em 3D HPS *estimation* em cenários com uma grande quantidade de pessoas.

Khirodkar *et al.* [24] propuseram uma nova abordagem *top-down mesh recovery*, apelada de ***Occluded Human Mesh Recovery* (OCHMR)**, que incorpora o contexto espacial da imagem para lidar com as limitações assumidas nas oclusões parciais de humanos. Na verdade, o OCHMR alcançou uma precisão ao nível do pixel no alinhamento da malha corporal estimada, possível em métodos *top-down* e robustez para oclusões verificada em métodos *bottom-up* ao usar informação do *global center-map* para compreender o contexto geral da cena. Para tal, os autores desenvolveram o módulo denominado de *Context Normalization* (CoNorm), um mecanismo adaptável a qualquer *backbone* focado na extração de *features*, para processamento dos *2D global* e *local center maps* e injeção de informação contextual de elevada resolução nos vários níveis de profundidade da rede extração de *features* (ResNet-50 [60]). A malha corporal humana é então otimizada iterativamente com esta informação adicional. Não obstante, o modelo terá sido treinado com *multi-person mesh interpenetration* e *depth ordering losses*, similar a *Coherent Reconstruction of Multiple Humans* (CRMH) [9], com penalização na interseção entre as malhas reconstruídas e uma *differentiable depth-ordering loss* para consistência na malha corporal humana definida. Apesar de tudo, a implementação não se adequa a aplicações em tempo real podendo, contudo ser destacado a eficácia em vários *datasets e.g.*, CrowdPose [101] e OCHuman [105].

Numa abordagem inovadora, Yuan *et al.* [34] apresentaram o ***Global Occlusion-Aware Human Mesh Recovery* (GLAMR)** e tal como o nome evidência, esta abordagem potencializa a obtenção da malha corporal humana, nas coordenadas do referencial do mundo a partir de câmaras dinâmicas com especial incorporação de uma variante da arquitetura *Transformer* [106]. Na verdade, o método destaca-se em situações que verifiquem oclusões severas e no *tracking* de humanos quando os mesmos deixam de estar visíveis *Field of View* (FoV) da câmara. Naturalmente, uma arquitetura desta complexidade decorre da combinação de diferentes blocos e contribuições da comunicada de visão computacional. A *pipeline* do GLAMR é dividida em quatro estados, evidenciados na Figura 2.34. Numa primeira fase, no pré-processamento, é utilizado um *Multiple Object Tracking* (MOT) [107] e um algoritmo de reidentificação para obter a sequência de *bounding boxes* de cada pessoa que, por sua vez, será a entrada de um modelo de HMR (*e.g.*, KAMA [32] ou SPEC [33]) para perceção do movimento $\widetilde{\boldsymbol{Q}}^{i}$ de cada pessoa (i-ésima *bounding box*) nas coordenadas da câmara. Naturalmente, $\widetilde{\boldsymbol{Q}}^{i}$ tende a ser incompleto, com poses pouco viáveis fruto da oclusão no humano em causa e falha na criação de *bounding boxes* no MOT. Como tal, num segundo estado, os autores propuseram

um método que procura completar o movimento (*generative motion infiller*), isto é, lidar com potenciais oclusões no movimento humano estimado $\tilde{\boldsymbol{\Theta}}^i$ e produzir um movimento corporal sem oclusão $\hat{\boldsymbol{\Theta}}^i$. Numa terceira fase, o módulo *global trajectory predictor* recorre do movimento do corpo preenchido $\hat{\boldsymbol{\Theta}}^i$ e gera uma trajetória global $\left(\hat{\boldsymbol{T}}^i, \hat{\boldsymbol{R}}^i\right)$, para cada pessoa. Por fim, decorre uma otimização da trajetória global de todas as pessoas e dos parâmetros da câmara a fim de produzir movimentos globais consistentes $\check{\boldsymbol{Q}}^i$ no cenário real.

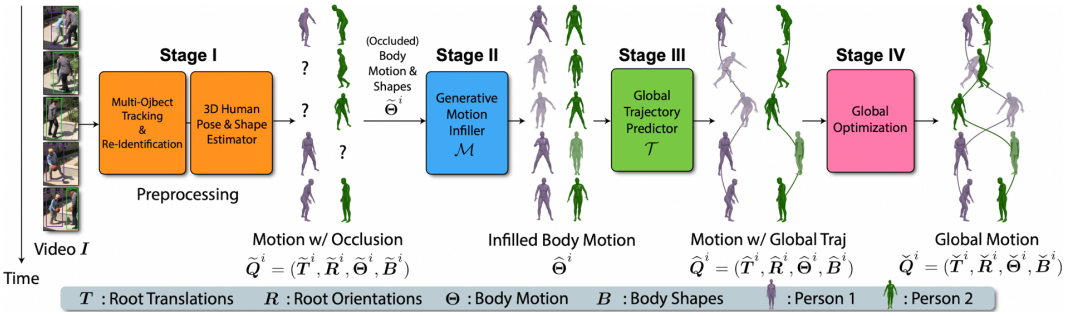


Figura 2.34: Representação da *pipeline* da implementação GLAMR [34].

De notar que os elementos que compõem o vetor $\tilde{\boldsymbol{\Theta}}$ e $\tilde{\boldsymbol{B}}$ correspondem, respetivamente aos parâmetros de pose e *shape* do modelo SMPL, sendo os restantes referentes ao reposicionamento da pose humana no sistema de coordenadas do mundo (obtidos do *3D HPS estimator*). Ainda assim, o objetivo do projeto exige a divisão em três fases, expostas na Figura 2.35.

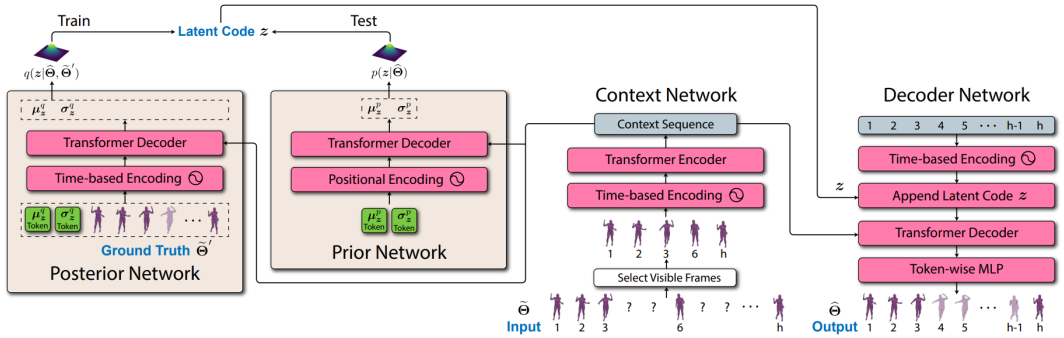


Figura 2.35: *Generative Motion Infiller Network* [34].

A rede *Generative Motion Infiller Network* (GMI-Net) $\boldsymbol{\mathcal{M}}$, executada na segunda fase, recorre a um *Conditional Variational Autoencoder* (CVAE) para gerar probabilisticamente o *latent code* $\boldsymbol{z}$. Isto é, numa *sequence-to-sequence* baseada em *Transformers* [106] onde o *encoder* aceita exclusivamente os *fames* com movimento visível $\tilde{\boldsymbol{\Theta}}$ (através de uma máscara de visibilidade V) para, juntamente com um *latent code* $\boldsymbol{z}$, no *decoder*, gerar o movimento sem oclusão $\hat{\boldsymbol{\Theta}}$. Este *latent code* $\boldsymbol{z}$ remete para uma representação compacta de características latentes que capturam informações relevantes sobre o movimento humano e auxiliam no preenchimento das

regiões ausentes ou ocultas. Como tal, a partir de *prior* (inferência) e *posterior networks* (treino) são geradas *prior* e *posterior distributions* para o *latent code* $\boldsymbol{z}$ (saída condicionada com base numa distribuição gaussiana) tendo por base a codificação do tempo em vez da posição da implementação original [106]. Naturalmente, variando este *latent code* são obtidos diferentes movimentos livres de oclusão $\widehat{\boldsymbol{\Theta}}$.

Não obstante, a rede do *Global Trajectory Predictor* $\boldsymbol{\mathcal{T}}$ adota uma arquitetura similar à GMI-Net com uma pequena diferença, os autores adotam LSTMs para modelação temporal em vez de *Transformers* uma vez que a mudança de trajetória remete exclusivamente em movimentos do corpo verificados em *frames* próximos, não exigindo uma consideração temporal de longo alcance.

Ainda que a origem dos dados de entrada provenham de vídeos RGB obtidos em cenários não controlados (*in-the-wild*), implementações mais recentes procuram estimar a pose 3D e *body shape* dos vários humanos no cenário mantendo a *coerência temporal* e uma localização global, não restrita ao espaço visível [34, 11]. Recentemente, Luvizon *et al.* [11] desenvolveram um sistema para perceção do movimento humano 3D a partir de uma câmara estática num culminar de conhecimentos emergentes em visão computacional. A arquitetura da proposta encontra-se presente na Figura 2.36, dividida em duas fases.

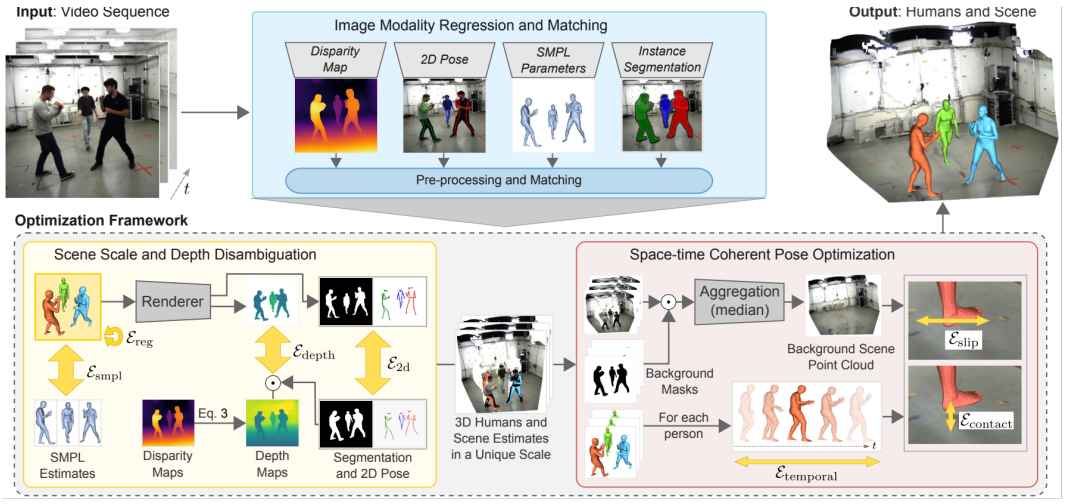


Figura 2.36: Visão geral da abordagem proposta por Luvizon *et al.* [11].

Numa primeira fase, *i.e.*, *Image Modality Regression and Matching* são obtidas e agregadas *human-related predictions* (*e.g.*, *disparity map*, pose 2D, parâmetros do modelo SMPL e *instance segmentation*) na sequência do vídeo. Numa segunda fase, igualmente dividida em duas componentes, atua a *optimization framework*: (1) *Scene Scale and Depth Disambiguation* recupera a profundidade do cenário (a partir de uma reconstrução de uma *point-cloud* do cenário estático) e as escalas humanas únicas, assim como a posição absoluta 3D das suas articulações (a partir de mapas de disparidade normalizados, articulações do corpo 2D e das rotações relativas das

articulações) e (2) *Space-time Coherent Pose Optimization* para um ajuste coerente *space-time* da pose corporal de cada indivíduo (rotações relativas das articulações, para redução de potenciais artefactos e garantir plausibilidade temporal, espacial e física, *e.g.*, *foot sliding*, interseções humano-cenário e *jitter*). De facto, a combinação de vários modalidades (*multi-modal inputs*), historicamente desenvolvidas na área de visão computacional, permitiram aos autores a redução drástica no erro da reconstrução da postura humana. Como tal, a comparação direta com algoritmos de estimação da pose 3D, *e.g.*, XNect [8] e ROMP [22] (falha na determinação da dimensão dos humanos, com *prediction* instáveis) e HMR, *e.g.*, GLAMR [34] e *Bird's-Eye-View* (BEV) [108] (falha ao prever posições coerentes fruto de diferentes tamanho dos humanos no cenário).

Human mesh recovery: bottom-up methods

Por outro lado, as abordagens *bottom-up* permitem a deteção das várias pessoas e a reconstrução da malha 3D simultaneamente, podendo ser disposta numa *multi-stage* [109, 110, 10] ou *single-stage* [111, 22, 108] *pipeline*. Contudo, são frequentemente afetados pela variação de escala dos humanos presentes no cenário, *e.g.*, distinção de uma pessoa pequena e de uma pessoa presente no fundo da imagem [100].

Como tal, Zhang *et al.* [111] desenvolveram um modelo *single-stage* chamado de ***Body Meshes as Points* (BMP)**, focado na redução de processamento redundante. O método recorre a uma *feature pyramid network* para extrair da imagem *multi-scale features*, *e.g.*, *lower scales* para representação de instâncias de maior dimensão (perto da câmara), em que cada instância é associada a um ponto numa dada grelha. Naturalmente, a aplicação do *BMP model* deve lidar com os problemas típicos destas abordagens: (1) reconstrução coerente das instâncias com noção de profundidade correta e (2) lidar com os problemas de oclusão já referidos. Para tal, os autores assumiram a utilização de um modelo pré-treinado para determinar a profundidade de cada pessoa e o ponto central (*3D space*) treinando-o com *dataset* completos (Human3.6M [104] e MuCo-3DHP[98]). Para o segundo desafio e atenuação da elevada sensibilidade do modelo na recuperação dos parâmetros SMPL em oclusões parcial, os autores, propuseram uma estratégia de *keypoint-aware occlusion augmentation* capaz de gerar oclusões sintéticas, durante o treino, e simular verdadeiras situações de oclusão (abstenção à utilização de dados de treino adicionais). Face à representação do ponto central, a *pipeline* do modelo é dividida em dois ramos paralelos para fazer a *instance localization* de cada pessoa na imagem e *body mesh recovery* com base nas *features* anteriormente associadas ao ponto que define o humano em causa. Esta primeira solução *single-stage* alcançou os métodos SOTA com performance e tempos de inferência dita de uma implementação em tempo real. Contudo, os resultados estão longes do modelo proposto por Luvizon *et al.* [11].

Sun *et al.* [22] propuseram o **ROMP**, uma implementação livre de *bounding box*, capaz de representar várias pessoas ao nível do pixel. Na verdade, o método estima diretamente múltiplos mapas, Figura 2.37 que, ao serem combinados, descrevem a malha corporal 3D de todas as pessoas no ambiente, concretamente: (1) *Body Center heatmap* (probabilidade de cada articulação 2D ser o centro do corpo) e (2) *Mesh Parameter map* (concatenação do *Camera map* e do *SMPL map*).

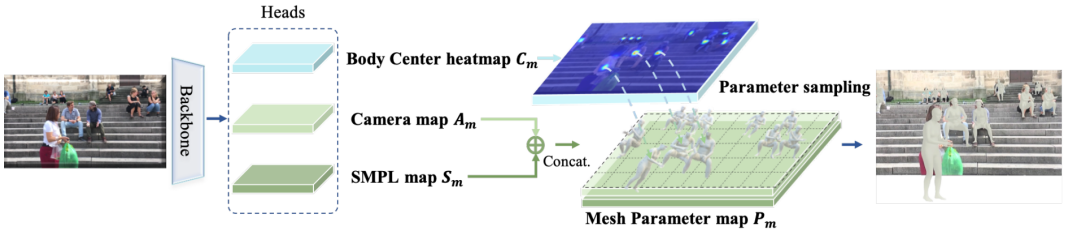


Figura 2.37: Visão geral da abordagem ROMP [22].

O *Camera map* agrega os parâmetros da câmara (análogo a implementações anteriores, (s, t_x, t_y), respetivamente a escala 2D (tamanho do corpo e "profundidade" até certo ponto) e translação $\boldsymbol{t} = (t_x, t_y)$ da pessoa em cada centro) enquanto o *SMPL map* contem os parâmetros do modelo SMPL referentes ao corpo. Através de um processo de amostragem, é possível obter do *Mesh Parameter map* toda a informação necessária (*mesh parameter vectors*) para estimar a malha corporal 3D e respetiva localização. Ainda assim, em cenários que se verifique sobreposição de pessoas e, portanto, colisão de diferentes centros de corpo no mesmo plano, os autores desenvolveram o ***Collision-Aware Representation* (CAR)**, cujo ideia se assemelha a uma força de repulsão magnética existente entre duas cargas positivas, sendo afastadas mutuamente. De facto, em situações de oclusão severa, esta solução promove o deslocamento do centro corporal para uma área visível, levando o modelo a amostrar os parâmetros da malha 3D a partir da melhor zona centrada nas partes visíveis. A não dependência de *features* obtidas exclusivamente na área limitada pela *bounding box* aumentou a precisão e eficiência do método ao lidar com situações de oclusão em cenários lotados de pessoas, ultrapassando a SOTA performance em vários *benchmarks* bem como uma velocidade de inferência em tempo real. Como tal, serviu de base para o desenvolvimento de muitos outras implementações nos anos seguintes.

Ainda assim, a implementação de Sun *et al.* [108], apelada de **BEV**, surge de uma melhoria ao ROMP, ao introduzir uma nova representação imaginária na análise da altura das pessoas com diferentes idades, *e.g.*, crianças e adultos. A arquitetura *multi-head* do BEV encontra-se ilustrada na Figura 2.38 e apresenta os cinco diferentes mapas obtidos a cada imagem RGB na entrada da rede.

Uma vez que o *2D bird's-eye-view map* imaginário representa uma potencial localização dos centros de cada corpo em profundidade é passível de ser inferida para

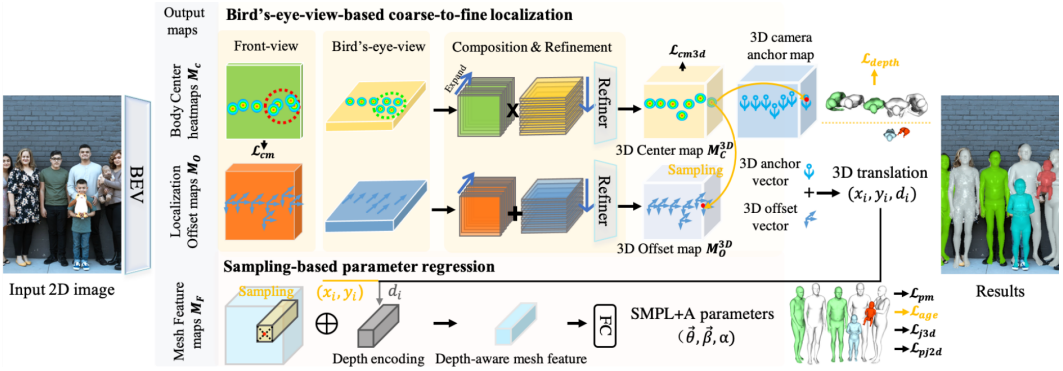


Figura 2.38: Visão geral da abordagem proposta por Sun *et al.* [108].

estimar *heatmaps* 2D nos diferentes planos. A combinação da informação obtida no plano frontal e na vista de cima (*bird's eye views*) origina uma relação profundidade/altura expressa em *heatmaps* 3D (*Body Center heatmaps*) e *3D Offset maps* para ajuste minucioso das deteções grosseiras. Esta solução permite ao BEV aprender de anotações 3D e 2D existentes. A soma destes elementos origina uma predição de translação 3D (x_i, y_i, d_i) que deverá ser combinada com o *Mesh Feature map* (capaz de diferenciar pessoas a diferentes profundidades, sobretudo em situações de oclusão) para obtenção dos parâmetros da malha 3D. Para isso, o valor de profundidade d_i é mapeado para um *depth encoding* com a adição das amostras do vetor de *feature* obtido do *Mesh Feature map*. E como tal, para um poderoso método de representação foi necessário rever o esquema de treino para garantir uma generalização do modelo. Uma vez que a altura de um indivíduo varia com a idade e a sua não consideração leva a erráticas previsões de profundidade, principalmente para crianças e bebés, o *weakly-supervised training* terá sido tido em atenção com um *dataset* próprio apelidado de ***Relative Human* (RH)** com *weak labels* associadas a diferentes idades (adultos, adolescentes, crianças e bebés) e *depth layers* para diferentes ordens de profundidade das pessoas na imagem. Não obstante, a utilização do modelo *Skinned Multi-Infant Linear* (SMIL) [112], baseado no modelo SMPL e dimensionado para parametrizar as malhas do corpo 3D de bebés, foi essencial. Na verdade verifica-se a sua utilização (SMIL) $\mathcal{M}_{\mathcal{I}(\vec{\theta},\vec{\beta})}$ para bebés quando um dado *offset* de idade α é superior a um dado *threshold* t_α, caso contrário é utilizado o modelo SMPL+A $\mathcal{M}(\vec{\theta}, \vec{\beta}, \alpha)$, proposto pelos autores. Os resultados são satisfatórios, consideravelmente superiores ao ROMP, mas ainda assim, inferiores à implementação desenvolvida por Luvizon *et al.* [11] ficando pendente a inclusão de variações na representação dos humanos além da idade, *e.g.*, diferentes pesos, género e etnia, entre outros.

Capítulo 3

Conceitos Fundamentais

O presente capítulo contempla o estudo preliminar à implementação propriamente dita, abordando conceitos fundamentais e tecnologias necessárias para o desenvolvimento do projeto em questão.

Em conformidade com os objetivos previamente estipulados e perante o estado atual das diversas soluções propriamente analisadas no Capítulo 2, serão introduzidos princípios complementares orientados às arquiteturas que os descrevem assim como às ferramentas das quais estes dependem.

O desejo incessante da comunidade pela criação de um sistema inteligente proporcionou a surgimento da *Artificial intelligence* (AI), uma área em crescimento com uma variedade de aplicações [113]. Ainda assim, o verdadeiro desafio da AI aloca-se na resolução de tarefas simples de serem resolvidas pelo humano mas complicadas de descrever computacionalmente, *e.g.*, construir um discurso lógico ou identificar objetos.

Esta questão abre a porta a um novo paradigma de programação. Contrariamente às abordagens tradicionais de programação, muitas vezes baseadas em modelos matemáticos complexos e métodos heurístico, o humano definia totalmente as ações a serem executadas, dividindo grandes problemas em várias tarefas pequenas. Por outro lado, uma rede neuronal apresenta uma lógica de implementação diferente ao aprender a partir diretamente de dados e ao descobrir a sua própria solução, muitas vezes potencializada pela perceção de padrões complexos e relações não-lineares, para o problema em questão [113].

As técnicas de aprendizagem em **DNNs**, atualmente conhecidas como *Deep Learning* (DL), potencializaram a utilização intensiva destes dados e substituição de métodos clássicos em muitos problemas de visão computacional. Na verdade, o estudo intensivo exposto em artigos científicos e o aumento do poder computacional permitiram à sociedade a sua aplicação em tarefas tão distintas como as propostas neste trabalho [114].

Deep learning é então um tipo particular de *machine learning* representado por um número sucessivo de *layers*, podendo, nas implementações mais recentes, envolver dezenas ou centenas de camadas empilhadas representas em redes neuronais. Apesar de alguns conceitos nativos do DL apontarem para referências neurobiologia, a sua mecânica não representa um modelo do cérebro, sendo neste processo associado a uma estrutura matemática capaz de aprender de uma dada representação de dados [114].

Neste secção, é analisada as vantagens e potenciais limitações do uso de DNNs em *3D PSE* a partir de *multi-view systems*, focando na sua aplicabilidade. Especificamente, é apresentado uma visão geral das arquiteturas das CNNs, GANs, RNNs e outras técnicas de DL levanto em atenção os trabalhos desenvolvidos até ao momento.

3.1 Introdução às *Deep Neural Networks*

A disposição das camadas das DNNs, estruturalmente definidas, permitem a execução de um determinado processo computacional diferente. ***Multilayer Perceptron* (MLP)** ou ***feedforward deep network*** é um exemplo de um modelo de DL representativo de uma função matemática para mapear os valores das entradas em novas saídas [113]. A compreensão dos dados é conseguida em *multi-stage*, sendo o número de instruções passíveis de execução em sequência proporcional à sua "profundidade".

Os ***perceptrons*** assumem-se como um elemento fundamental das DNNs que admite várias entradas binárias, $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\ldots \boldsymbol{x}_n$ e produzem uma única saída binária. Eventualmente podem ser representados em termos algébricos (Equação 3.1) como o produto escalar entre os vetores representativos dos **pesos** $\boldsymbol{w}$ e os valores das **entradas** $\boldsymbol{x}$ somado do ***bias*** $\boldsymbol{b}$ para ajuste no processamento dos dados de entrada.

$$output = \begin{cases} 0 & if \quad w \cdot x + b \leqslant 0 \\ 1 & if \quad w \cdot x + b > 0 \end{cases} \tag{3.1}$$

Naturalmente e assim como presente na Figura 3.1, diferentes camadas possuem diferentes interpretações dos dados consoante o peso, aleatoriamente atribuído no início do treino, a cada *perceptron*.

O ***input layer*** admite variáveis passíveis de serem interpretadas por humanos. À medida que nos aproximamos do interior da rede, mais abstratas são as *features*

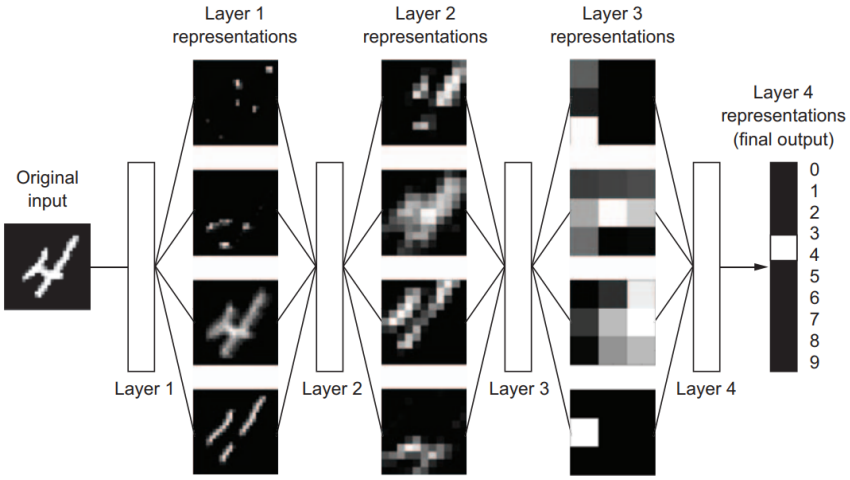


Figura 3.1: Representação de um modelo de DL para classificação de dígitos [114].

obtidas. Nomeadamente, é nos ***hidden layers*** que decorre todo o tipo de cálculos computacionais para deteção de *features* simples, como cantos e *edges*, nos primeiros e *features* mais complexas, como formas e objetos, perto do ***output layer***.

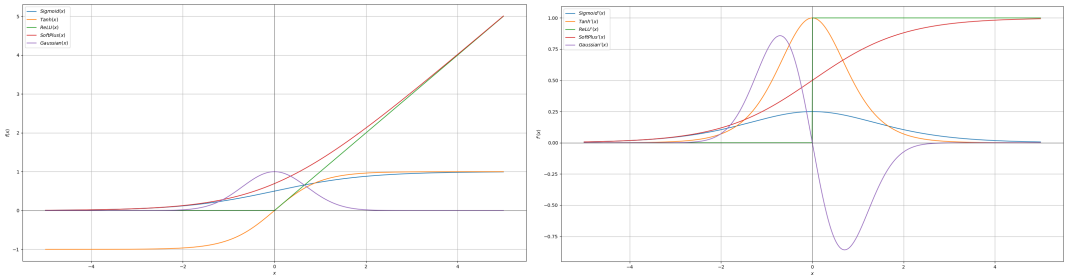
3.1.1 Funções de ativação

Eventualmente uma função de ativação (*e.g.*, *Sigmoid*, *ReLU*, *Step* e *Tanh*) condiciona o valor real à saída do *perceptron* para que estes não assumam exclusivamente valores binários. O estudo da aplicabilidade de cada uma destas funções nas camadas intermédias assim como a consequência da sua utilização para efeitos de treino remete para uma área complementar das DNNs. A não linearidade proporcionada por algumas das funções, representadas nos gráficos da Figura 3.2, possibilitam a definição de relações não lineares entre as entradas e saídas nos *hidden layers*. Nas camadas de saída, estas funções devem ser estabelecidas consoante o problema, *e.g.*, função *softmax* utilizada em problemas de classificação.

Como tal, a função ***softmax***, Equação 3.2, normaliza um determinado vetor $\boldsymbol{z}$ com $\boldsymbol{K}$ números reais numa distribuição com probabilidade proporcional à exponencial do valor de entrada, garantindo um somatório unitário nas saídas da camada, tipicamente um *FC layer*.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \; for \; i = 1, \ldots, K \; and \; \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K \tag{3.2}$$

A função *Tanh*, Equação 3.3, com um formato semelhante à função *sigmoid*, diferenciando no intervalo dos valores de saída ($[-1, 1]$ e $[0, 1]$ respetivamente) sofre igualmente do *vanishing gradient problem*. Apesar de apresentarem uma variação suave do gradiente, *i.e.*, evitam saltos repentinos nos valores de saída, ambas as derivadas se aproximam de zero à medida que o valor de entrada aumenta. Como tal,



(a) Funções de ativação (b) Derivada das funções de ativação

Figura 3.2: Alguma das funções de ativação utilizadas em trabalhos anteriores.

a sua utilização deve ser restringida, por norma, aos *output layers* com preferencial utilização da função *Tanh* nas RNNs.

$$\tanh(x) = \frac{\text{senh}(x)}{\cosh(x)} = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{3.3}$$

Por sua vez, a função ***rectified linear unit*** ou *ReLU*, Equação 3.4, destaca-se na utilização em camadas intermédias pela sua eficiência computacional (ativação de apenas alguns neurónios) e rápida convergência do gradiente descendente da *loss function* fruto da sua propriedade linear. Contudo, o lado negativo do gráfico torna o gradiente nulo resultado em "neurónios mortos" (*dying ReLU problem*) uma vez que durante o processo de *backpropagation* os pesos e *bias* de alguns neurónios não são atualizados.

$$\text{ReLU}(x) = \max(0, x) \tag{3.4}$$

O surgimento de variantes como ***Leaky ReLU***, Equação 3.5, ou ***Parametric ReLU*** procuraram atenuar esta questão quando é verificada dificuldade na convergencia da rede durante o treino. Esta última função apresenta uma equação análoga à *Leaky ReLU* com valência do coeficiente a ser aprendido junto com os outros parâmetros da rede neuronal.

$$\text{LReLU}(x) = \max(ax, x) \quad \text{sendo } a \text{ tipicamente } 0{,}01 \tag{3.5}$$

Assim como referido, *softmax* e *sigmoid* tem aplicabilidade na última camada de ativação sendo a escolha pendente do objetivo da rede e da respetiva *loss function*.

3.1.2 *Loss functions*

A perceção do quão longe está a saída da rede do esperado é parametrizado através de uma *loss function* (ou *objective function*) que pode ser definida como o produto das derivadas parciais das funções de ativação, pelo que a escolha da mesma pode afetar indiretamente o processo de ajuste [113].

Com base na *chain rule* torna-se fácil escrever a expressão algébrica para o gradiente de um escalar em relação a qualquer nó. Dividindo a rede em pequenos *symbolic graphs* e assumindo $\boldsymbol{u}^{(n)}$ como sendo a *loss* de um exemplo de treino com respeito aos nós de entrada $\boldsymbol{u}^{(1)}$ a $\boldsymbol{u}^{(n_i)}$ entradas, a derivada deste gráfico ficaria definida como $\frac{\partial u^{(n)}}{\partial u^{(i)}}$ para todos os $i \in \{1, 2, \ldots, n_i\}$. Como tal, a divisão em subgráficos para análise nó a nó potencializam a determinação da derivada em causa partindo do nó $\boldsymbol{u}^{(j)}$ através da Equação 3.6. Neste caso, assume-se a existência de uma única *edge* entre os nós $\boldsymbol{u}^{(j)}$ e o $\boldsymbol{u}^{(i)}$, simbolizados pelo termo $\frac{\partial u^{(i)}}{\partial u^{(j)}}$ [113].

$$\frac{\partial u^{(n)}}{\partial u^{(j)}} = \sum_{i:j} \frac{\partial u^{(n)}}{\partial u^{(i)}} \frac{\partial u^{(i)}}{\partial u^{(j)}} \tag{3.6}$$

Na verdade, este processo atua um algoritmo de otimização, conhecido como gradiente descendente, permitindo, a cada iteração (*epochs*), um ajuste meticuloso dos pesos e *bias* proporcional à derivada de uma *loss function* [114]. O ciclo é realizado até que seja atingido o mínimo analítico e minimizado o erro.

Não obstante, será necessário considerar um *learning rate* $\boldsymbol{\eta}$ a fim de definir o tamanho dos "passos" executados para atingir este mínimo mantendo o gradiente negativo. Por norma, este valor é pequeno e varia conforme o comportamento da *loss function*. Naturalmente, um valor baixo exigirá um maior número de iterações, comprometendo a eficiência geral, mas atingindo o mínimo com maior precisão.

A seleção da *loss function* dependerá do objetivo da rede podendo, segundo a literatura, ser dividida em problemas de regressão e de classificação. Daqui em diante, com base na nomenclatura apresentada por diferentes autores, assume-se $\boldsymbol{\mathcal{L}}$ como a *loss function* específica de um componente ou parte do modelo enquanto $\boldsymbol{L}$ remeterá para a *loss function* geral do modelo.

MSE e ***Mean Absolute Error* (MAE)** são recorrentemente usadas para medição do erro entre um dado valor real y_i e o previsto $\hat{y}_i$. Concretamente, MSE, Equação 3.7, permite a análise segundo uma função quadrática com apenas um mínimo. Por sua vez, MAE, Equação 3.8, assume o mesmo peso para todos os erros e portanto, uma maior imunidade ao surgimento de *outliers*.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.7}$$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3.8}$$

Contudo, a função MAE não é diferenciável quando $y_i = \hat{y}_i$, apresentando uma derivada unitária positiva quando $\hat{y}_i > y_i$ e uma derivada unitária negativa quando $\hat{y}_i < y_i$. Nesta situação é verificada a anulação da otimização do gradiente descendente. A utilização da *Huber Loss* agrega as vantagens das funções MSE e MAE

uma vez que condiciona o intervalo de utilização de cada uma em função de um dado *threshold*.

Para problemas de classificação, a *Cross-Entropy Loss* (*Negative Log-Likelihood*) apresenta-se como uma escolha comum para medição do erro verificado entre as previsões e as verdadeiras classes. Neste cenário a rede produz um vetor de probabilidades (p_i obtidas da função *softmax* para a i^{th} classe definida) para seleção da mais provável.

***Binary Cross-Entropy* (BCE)** *loss*, Equação 3.9, é utilizada para classificação binária (apenas duas classes) com n número de amostras obtidas do treino da rede segundo uma penalização logarítmica negativa ($\log(p_i) < 0 \ \forall \ p_i \in [0, 1[$) e um valor binário (0 ou 1) representativo da classe escolhida t_i.

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n} t_i \log\left(p_i\right) = -\frac{1}{n}\sum_{i=1}^{n}\Big[t_1 \log\left(p_1\right) + t_2 \log\left(p_2\right)\Big] \tag{3.9}$$

Na eventualidade do número de classes ser superior a duas é utilizado ***Categorical Cross-Entropy* (CCE)** *loss*. O cálculo, apresentado na Equação 3.10, é idêntico a menos de um somatório representativo do número de classes m em consideração no processo de treino. Como tal, BCE apresenta-se como um caso especial da CCE quando $m = 2$ (apenas duas classes).

$$L_{CCE} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} t_{ij} \log\left(p_{ij}\right) \tag{3.10}$$

3.1.3 *Learning with gradient descent*

Implicitamente, as funções de ativação devem ser diferenciáveis, centradas em zero e monótonas uma vez que durante o treino, nos algoritmos de *backpropagation*, as redes neuronais ajustam os pesos e *bias* através da sua derivada parcial (*backpropagated gradients*).

Existem três principais variantes do algoritmo do gradiente descendente com base na quantidade de dados usado para determinar o gradiente de uma *loss function*, cada um com características diferentes pelo que a seleção deve ser ajustada ao problema (*trade-off* entre precisão do ajuste dos parâmetros e tempo necessário para este ajuste), nomeadamente: (1) ***Batch Gradient Descent***, (2) ***Stochastic Gradient Descent*** e (3) ***Mini-Batch Gradient Descent***.

O *batch gradient descent* determina o gradiente da *loss function* para ajuste dos parâmetros do modelo $\boldsymbol{\theta}$ (*e.g.*, pesos e *bias*) de forma a minimizar a função em causa (Equação 3.11). O algoritmo em causa impede o modelo de ser atualizado durante o processo de treino (*online*) até que todas as observações do *dataset* tenham sido avaliadas (atualização dos parâmetros a cada *epoch*). Este facto poderá tornar o treino impraticável principalmente em situações em que se verifique que todo o

dataset não possa ser armazenado em memória. Como tal, o algoritmo revela-se eficiente ao lidar com pequenos *dataset* mas lento para convergir para a solução ótima em *dataset* maiores (mínimo local poderá não ser o mínimo absoluto).

$$\theta = \theta - \eta \cdot \nabla_\theta L(\theta) \tag{3.11}$$

Por sua vez, o *stochastic gradient descent* (Equação 3.12) diminui a redundância do algoritmo anterior ao calcular o gradiente da *loss function* e atualizar os parâmetros a cada observação. Como tal, deverão ser realizadas um determinado número de iterações (proporcional ao número de exemplos x_i e *label* y_i do *dataset*) por cada *epoch*. O facto de apenas ser necessário armazenar uma observação em memória torna-o computacionalmente eficiente e permite a realização de um treino *online*, contudo, as atualizações sofrem de uma alta variância (*noisy gradients* com *overshooting*), resultando em flutuações da *loss function*. Não obstante, esta consequência poderá ser útil para "escapar" do mínimo local e encontrar o mínimo absoluto.

$$\theta = \theta - \eta \cdot \nabla_\theta L\left(\theta, x_i, y_i\right) \tag{3.12}$$

O *mini-batch gradient descent* (Equação 3.13) é a combinação dos dois métodos acima. Ao utilizar um pequeno subconjunto do *training dataset* (*batch size*) aproveita a precisão do *batch gradient descent* e a eficiência do *stochastic gradient descent*. Na verdade, o *batch size* n pode ser ajustado para atender as necessidades do problema em questão.

$$\theta = \theta - \eta \cdot \nabla_\theta L\Big(\theta, x_{(i:i+n)}, y_{(i:i+n)}\Big) \tag{3.13}$$

3.1.4 Desafios do gradiente descendente

Apesar do gradiente descendente ser a abordagem mais comum em problemas de otimização, esta vem agregada de um conjunto de desafios. Nos problema de otimização convexos, o gradiente descendente pode encontrar o mínimo absoluto com facilidade, mas noutros cenários o mínimo encontrado poderá ser relativo e neste caso estar a atingir uma solução subótima, isto é, ficar preso em mínimos locais, não sendo este o mínimo absoluto desejado. Assim como referido, gradientes com presença de ruído poderão ajudar a evitar a permanência neste mínimo.

Não obstante, o desempenho do algoritmo será sensível à inicialização dos parâmetros. Na verdade, uma inicialização pouco cuidada levará a uma convergência lenta ou, novamente, a uma solução subótima. Neste sentido foram introduzidas técnicas avançadas além da ***random initialization*** ou inicialização com base numa distribuição normal como ***Xavier initialization*** e ***He initialization***.

Fruto disto, os problemas de *vanishing gradients* e *exploding gradients* também são passíveis de ocorrer, dependo da função de ativação selecionada e da estrutura da rede neuronal. A primeira, já referida, ocorre em momentos que se verifique um gradiente demasiado pequeno e que tenda a diminuir à medida que avançamos nas camadas da rede evitando o processo de aprendizagem da mesma. Já o segundo ocorre para gradientes demasiado grandes, criando um modelo instável para cada alteração de parâmetros. Nestes casos, a definição de um limite máximo ou mínimo de gradiente, redução do *learning rate*, simplificação da rede neuronal (número de camadas) ou alteração da função de ativação (*e.g.*, *Leaky ReLU*) poderá ajudar na redução deste problema.

3.1.5 Princípio da aprendizagem

A maioria dos algoritmos de DL, descritos no Capítulo 2, poderiam ser divididas em categorias de *supervised learning* e *unsupervised learning*. Este processo é o culminar de vários algoritmos, nomeadamente algoritmos de otimização, *loss function*, a estrutura do modelo em si e o *dataset*. Ainda assim, parte dos hiperparâmetros do modelo introduzidos anteriormente devem ser definidos antes do treino para que a cada iteração seja novamente ajustados.

Este campo de desenvolvimento permite a compreensão do princípio do fundamento da inteligência. Mitchell [115] definiu o processo de aprendizagem de uma máquina: "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*"

As tarefas aprendidas são, por norma, complicadas de resolver com base em programas fixos projetados por humanos. Apesar de o parecer, o processo de aprendizagem não é considerado uma "tarefa" mas sim um meio para atingir a tarefa T. Uma tarefa é então uma definição de como o algoritmo deve lidar com dado exemplo (conjunto de *features* quantitativamente medidas). Os algoritmos de DL potencializam a execução de diversas tarefas, sendo as mais recorrentes:

- **Classificação**: identificação a qual das categorias definidas a entrada pertence;
- **Regressão**: previsão de um valor com base nos dados de entrada;
- **Segmentação**: separação e identificação de objetos ou regiões numa imagem;
- **Processamento de linguagem natural**: processamento de texto e fala para tarefas como reconhecimento de fala, tradução automática e contexto.

Não obstante, *multi-task learning* pode conduzir à generalização do modelo, redução do *overfitting* e a uma maior eficiência computacional ao partilhar recursos entre as diferentes tarefas. Constituindo um subcampo do *machine learning*, exigirá

a reformulação da arquitetura do modelo (implementação de um *shared backbone* e várias *heads* cada uma dedicada a uma tarefa) e do processo de treino (definição de uma nova *loss function* como a soma ponderada da contribuição de cada tarefa e do *optimizer*.

A complexidade do sistema de treino não invalida o estabelecimento de um método de avaliação através de uma medida quantitativa do seu desempenho. Esta medida P, é específica à tarefa T do modelo em causa devendo ser avaliada num conjunto de teste. Como tal, *accuracy*, complemento do *error rate* (proporção de exemplos incorretamente estimados pelo modelo) são comummente usadas para tarefas de classificação.

Ainda assim, em função da experiência prévia que o algoritmo possui durante a fase de treino, estes poderão ser classificados em *unsupervised* e *supervised*. A principal diferença entre os dois reside no facto do ***supervised learning*** ser realizado com recurso à *ground truth*, *i.e.*, existe um conhecimento prévio de qual o resultado esperado na inferência da rede neuronal. Apesar de consistir principalmente em *classification* e *regression*, *object detection* e *image segmentation* também são consideradas tarefas pacíficas de *supervised learning* [114].

Unsupervised learning, por sua vez, aprende as propriedades essenciais de um *dataset* repleto de *features*, sem conhecimento concreto do resultado esperado. *Clustering* e *dimensionality reduction* são alguns exemplos de tarefas realizas por estes algoritmos.

Este métodos constituem apenas a "ponta o iceberg", sendo *machine learning* uma área extensa com uma taxonomia repleta de subcampos complexos. Métodos alternativos incluem *self-supervised learning* e *weakly supervised learning*.

Self-supervised learning é um tipo de *supervised learning* sem acesso a anotações humanas mas com supervisão realizada por intermédio de *labels* internamente criadas. Os *autoencoders* são redes neuronais treinados, por norma, em *self-supervised learning*.

Weakly supervised learning procura otimizar o treino com base numa pequena quantidade de dados (*weak/noisy annotations*) e conhecimentos prévios. Este método é comum em algoritmos para *HPS estimation* fruto da dificuldade em reunir uma grande quantidade de *datasets* com informações estruturalmente coerentes.

3.1.6 *Capacity, overfitting* e *underfitting*

Um dos problemas centrais em *machine learning* encontra-se na dificuldade da generalização do modelo para manter a performance em exemplos não observados durante treino. Tipicamente, um *dataset* encontra-se dividido em três subconjuntos, *i.e.*, conjunto de treino, validação e teste.

Naturalmente, o primeiro conjunto será utilizado para treinar o modelo. Durante este processo será possível calcular um determinado **erro de treino**, indicativo do progresso do treino e potencial ajuste dos parâmetros previamente definidos.

O conjunto de validação dá origem a um **erro validação** para avaliação da generalização do modelo a novos dados e acompanhar processo de treino (ajuste dos hiperparâmetros) a fim de evitar *overfitting* (*early stopping*).

Por sua vez, o conjunto de teste permite avaliar o desempenho do modelo no final do treino dando origem a um **erro de teste** (*generalization error*). No entanto, este erro é apenas uma estimativa do verdadeiro erro do modelo ao lidar com dados não observados, podendo ser afetado pelo tamanho do *dataset* de treino e da sua capacidade de representação de uma dada realidade. Portanto, é importante usar técnicas como *cross-validation* ou *bootstrap* para estimar a variabilidade do erro de teste e obter uma ideia mais precisa do desempenho do modelo [113].

Os dois principais desafios durante o processo de treino, *i.e.*, *underfitting* e *overfitting*, estão associados à determinação de dois fatores representativos do desempenho do algoritmo durante treino, nomeadamente [113]:

1. Redução do erro de treino;
2. Redução do espaço entre o erro de treino e o erro de teste.

Overfitting ocorre quando o espaço entre o erro de treino e o erro de teste é demasiado grande, podendo ser indicativo de um treino excessivamente prolongado ou fruto de uma arquitetura complexa. Nesta fase, o modelo "memoriza" o ruído e ajusta-se ao conjunto de dados de treino, não mantendo a mesma performance para a tarefa na qual foi projetado [116]. Eventualmente, o problema será oriundo de um *dataset* de treino redundante com presença de um elevado número de imagens com características muito próximas.

Por sua vez, o ***underfitting*** lida com o problema oposto, sendo que o modelo não apresenta a performance esperada para o qual foi projetado, fruto de um treino pobre (número de imagens do *dataset* de treino baixo ou tempo de treino reduzido) [116].

Desta forma, será possível controlar a probabilidade de um modelo de sofrer de *overfit* ou *underfit* em função da sua *capacity*. Concretamente, modelos com baixa *capacity* podem ter dificuldades na atualização devida dos parâmetros para um dado *dataset* de treino. O contrário pode evidenciar um modelo com tendências a sofrer de *overfit*.

O formato em "U" da curva, tipicamente representativa do erro de teste, pode ser visto na Figura 3.3. À medida que a *capacity* aumenta, o erro de treino diminui mas o erro de teste aumenta (aumenta o *generalization gap*).

Não obstante, a área de estatística disponibiliza diversas ferramentas que podem auxiliar na generalização do modelo, nomeadamente com a estimação de *bias* e variância de um determinado *estimator*. Ao aprender a partir da informação obtida

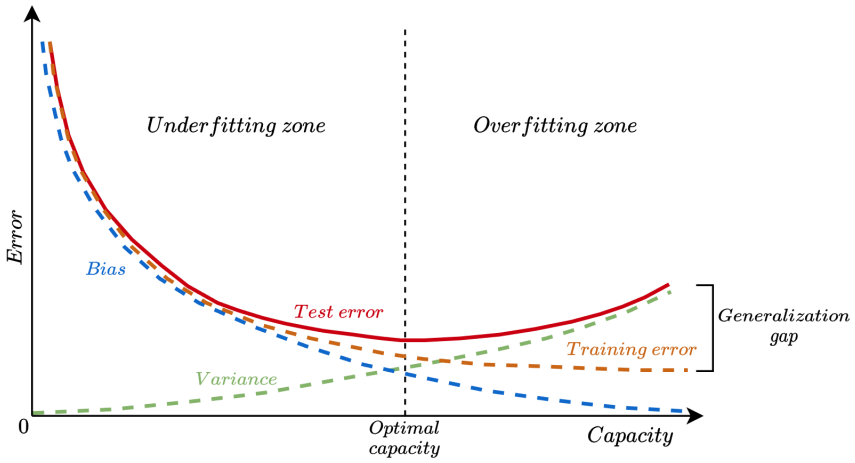


Figura 3.3: Variação da *capacity* ($eixo-x$) e do erro de teste ($eixo-y$) com o *bias* e *variance* em duas zonas divididas pela *optimal capacity* [113].

durante o treino, ajusta os parâmetros para diminuição do valor do erro de treino. *Bias* representa a diferença entre as previsões do modelo e os valores verdadeiros, enquanto a variância remete para a sensibilidade do modelo a variações nos dados de treino.

Novamente, na Figura 3.3, à medida que a capacidade aumenta, o *bias* tende a diminuir e a variância tende a aumentar. Uma arquitetura simplificada poderá originar um *bias* elevado pela distância ao valor esperado. Da mesma forma, uma variância elevada indica dificuldade na generalização do modelo (*overfitting*).

Ainda assim, uma comparação direta entre a curva do erro de validação e erro de treino permitirão visualizar estas zonas. Na zona de *overfitting* o decline da curva do erro de validação inverte-se, afastado-se da curva do erro de treino e na zona de *underfitting* a curva do erro de validação, com *offset* considerável à curva do erro de treino, tende a reduzir-se [116].

3.1.7 Regularização em *deep learning*

Atualmente, existe diversas técnicas aplicadas durante o treino que procuram otimizar a generalização da rede neuronal e evitar o *overfitting* em modelos de *deep learning*. Estas estratégias são conhecidas como **regularização**.

Regularização pode ser entendida como qualquer modificação (*e.g.*, restrições no valor dos parâmetros ou adição de novos termos à *loss function*) que seja feita no algoritmo de aprendizagem que visa a redução do erro de teste mas não do erro de treino [113].

Estas estratégias são baseadas em *regularizing estimators* para encontrar um ponto de equilíbrio entre o aumento do valor do *bias* e redução da variância.

Existe duas principais técnicas que recorrem à adição de ***parameter norm penaltie*** $\Omega(\boldsymbol{\theta})$ à *loss function* L para limitar a capacidade dos modelos, *i.e.*, incentivar

a rede a usar pesos menores e devidamente distribuídos, nomeadamente L^1 *regularization* (*Lasso Regression*) e L^2 *regularization* (*Ridge Regression*).

Para m número de amostras $(X^{(train)}, y^{(train)})$, simplificado a (X, y), assumindo $\tilde{L}$ como a *loss function* regularizada e $\alpha \in [0, \infty[$ o hiperparâmetro que controla a penalização do termo $\Omega(\boldsymbol{\theta})$ (sendo θ o vetor dos parâmetros afetados/a regularizar, *e.g.*, pesos) temos que:

$$\tilde{L}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = L(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \alpha\Omega(\boldsymbol{\theta}) \tag{3.14}$$

Sucintamente, L^1 ***regularization*** adiciona uma penalização proporcional à soma absoluta dos parâmetros da rede (o que permite uma penalização equivalente à soma dos valores absolutos estimados e à anulação do valor resultante da *loss function*). Por sua vez, L^2 ***regularization***, realiza uma penalização proporcional à soma dos quadrados dos parâmetros da rede (permite uma aproximação do valor nulo). A primeira é especialmente interessante em modelos com vários pesos próximos de zero, simplificando o custo computacional da rede. Ainda assim, a segunda técnica é mais utilizada pela capacidade imposta no algoritmo de aprendizagem em "detetar" maior variância nas entradas X (reduzindo os pesos) [113].

Assumindo a MSE como *loss function* para a regressão de uma equação linear $y = w_1x_1 + w_2x_2 + \ldots + w_mx_m + b$, sendo $x_1, x_2, \ldots x_m$ os valores de entrada (*features*) e $w_1, w_2, \ldots w_m$ os pesos, a *loss function* para *Lasso regression* e *Ridge regression* pode ser dada, respetivamente pela Equação 3.15 e Equação 3.16.

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{m} w_j \cdot x_{ij} \right)^2 + \alpha \sum_{j=0}^{m} |w_j| \tag{3.15}$$

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{m} w_j \cdot x_{ij} \right)^2 + \alpha \sum_{j=0}^{m} {w_j}^2 \tag{3.16}$$

A regularização dos parâmetros do *bias* pode contribuir para o *underfitting* do modelo. Ainda assim, pode ser relevante a utilização de diferentes penalizações (diferentes α) para cada camada da rede.

Técnicas alternativas como *data augmentation*, *early stopping*, *batch normalization* e *dropout* podem ser eficazes, se devidamente aplicadas, na redução *overfitting*.

A melhoria do conteúdo e da qualidade do *dataset* é a melhor forma para treinar um modelo, neste sentido ***data augmentation*** propõe a criação de *fake data* para adição ao *dataset* de treino. Este processo é especialmente benéfico em tarefas de *object recognition*, estando em causa alterações como de rotação, inversão nos dois eixos, aplicação de *blur* e ruído, alteração dos níveis de brilho e saturação, entre outros.

Por sua vez, a técnica de ***dropout***, aplicada a uma camada consiste em *dropping out* aleatoriamente um certo número de *output features* (neurónios nomeadamente) durante o treino permitindo reduzir a complexidade da rede neuronal (eliminando a sua contribuição na *forward* e *backpropagation*) e obrigando a rede a aprender representações do *dataset* de treino por vezes redundantes, mas mais robustas. Para tal, deverá ser definido um hiperparâmetro, o *dropout rate*, *i.e.*, fração de *features* que serão anuladas (tipicamente um valor entre 0,2 e 0,5) [114].

Por último, ***batch normalization***, uma técnica focada na otimização para *deep neural networks*, não sendo considerada propriamente um algoritmo. Na verdade é um método que propõe uma alteração dos parâmetros de forma adaptativa motivado pela dificuldade verificada no treino de *deep models* (várias camadas intermédias). Ao normalizar o resultado da função de ativação evita potenciais gradientes instáveis e permite que a rede neural seja treinada com maior velocidade [113].

3.2 *Convolutional Neural Networks*

Convolutional neural networks, também conhecidas por *ConvNets* ou CNNs são um tipo de redes orientadas ao processamento de dados no formato matricial (*2D tensors*) com histórico sucesso em reconhecimento e classificação de objetos em imagens.

Esta classificação surge da implementação de um modelo de uma CNN disposto em três estados: (1) um determinado número de camadas para realizar convoluções em paralelo e produzir um conjunto de ativações lineares, (2) sujeitas posteriormente a uma função de ativação não linear (***detector*** *stage*) e por último, (3) uma **função de *pooling*** para modificar ainda mais a saída.

Com a evolução estrutural das DNNs diversos trabalhos focaram-se na extração de *features* mais complexas, com vários níveis de representação e fulcral para a tarefa de HPE. *DeepPose* [49] foi um dos primeiro projetos a aplicar CNNs nesta tarefa, com um sistema em cascata com 7 camadas de CNNs para perceção do contexto completo de cada articulação na imagem.

3.2.1 Estrutura Base das CNNs

O nome desta rede deriva de uma operação matemática aplicada ao longo da mesma, *i.e.*, **convolução**. Apesar disso, a operação de convolução nas CNNs diferencia ligeiramente da convolução matemática normal. Em vez disso, nesta camada, realiza-se um produto matricial de um dado filtro (***kernel***) $\boldsymbol{K}(m \times n)$ ao longo da matriz de píxeis representativa da imagem $\boldsymbol{I}(i \times j)$ (através de uma técnica conhecida como janela deslizante), resultando numa saída apelada de *feature map* $\boldsymbol{S}$.

Por definição, a operação de convolução ($*$) exige a rotação da matriz *kernel* em 180° para que se verifique a comutatividade, podendo ser expressa segundo a Equação 3.17. Apesar disso, várias bibliotecas, focadas no desenvolvimento de redes

neuronais, implementam uma função idêntica, *i.e.*, ***cross-correlation*** apelando-o de convolução. Nesta, não se verifica a inversão do *kernel*.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{3.17}$$

Ainda assim, a convolução não é completamente imposta, devendo, no entanto, de ser mantida a consistência ao longo da rede para que durante o treino sejam definidos os valores do *kernel* na devida posição.

A deslocação do *kernel* ao longo da matriz representativa da imagem de entrada é definida segundo um determinado número de *strides*. Concretamente, um ***stride s*** unitário desloca esta matriz 1 pixel de cada vez, enquanto um *stride* de 2, move o *kernel* de 2 em 2 pixeis. Naturalmente, esta redução de custo computacional, proporcionada pelo avanço de algumas posições, resultará numa diminuição da matriz de saída. Admitindo uma matriz de entrada quadrada ($i = j = q$) e um filtro quadrado ($m = n = k$) será possível redefinir o tamanho do *tensor* de saída, obrigatoriamente quadrado ($g \times g$), pela expressão:

$$g = \frac{q-k}{s} + 1 \tag{3.18}$$

Consequentemente, neste formato, a imagem tenderá para resoluções inferiores à original à medida que mais camadas de convolução sejam aplicadas, uma vez que a informação das bordas da imagem é parcialmente desaproveitada. A solução a este problema é conseguida, usualmente, pela inclusão de mais pixeis nas bordas da imagem original, com uma quantidade $\boldsymbol{p}$ seguidamente definida:

$$p = \frac{k-1}{2} \tag{3.19}$$

Vale a pena rever três casos especiais de *zero-padding*. ***Valid convolution***, sem preenchimento das bordas ($p = 0$) e com uma matriz quadrada de saída de largura $q - k + 1$, resultando, com o excessivo número de camadas convolucionais, numa redução a 1×1. ***Same convolution*** com um preenchimento tal que garanta uma matriz de saída de dimensões iguais à matriz de entrada, não havendo propriamente restrições ao número de camadas convolucionais. Por último, ***full convolution***, ao permitir que cada pixel seja "visitado" k vezes em cada direção resultado numa matriz de saída de largura $q + k - 1$. Tem isto em atenção, a largura do resultado da convolução na matriz de saída fica definido por:

$$g = \frac{q-k+2p}{s} + 1 \tag{3.20}$$

A última fase do processo de deteção de *features*, o *pooling layer*, responsável por reduzir o numero de parâmetros da imagem, mantendo a informação importante mas reduzindo o custo computacional. Esta otimização, posterior à camada de

convolução, torna o modelo mais robusto a variações das posições das *features* da imagem de entrada uma vez que são considerados valores aproximados em vez dos valores precisos. Tipicamente existem três tipos de *pooling layers*:

- *Max pooling*;
- *Average pooling*;
- *Sum pooling*.

O ***max pooling*** seleciona o elemento máximo de uma determinada região (*kernel*). Por sua vez, o ***average pooling*** realiza um processo idêntico anotando desta vez a média enquanto o ***sum pooling*** realiza a soma da intensidade dos pixeis nas regiões respetivas.

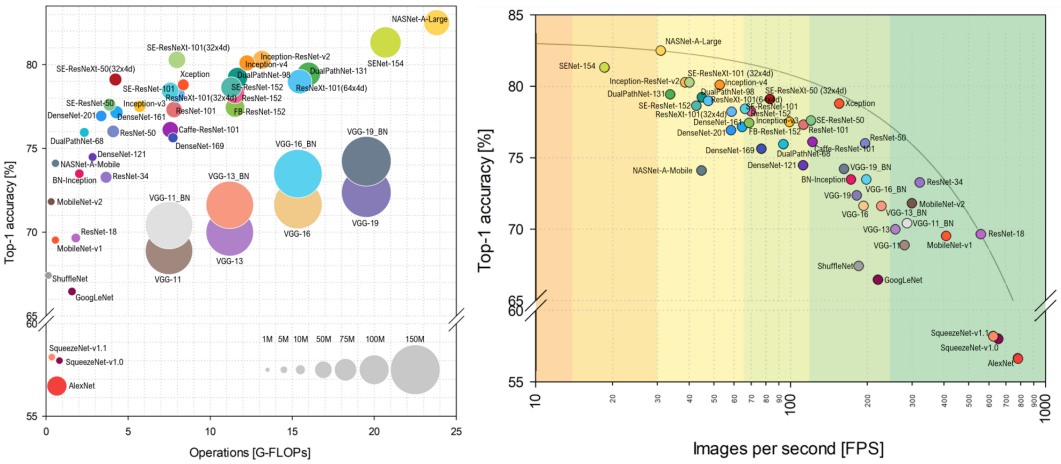
No que diz respeito ao processo de classificação, a adição de um *Fully Connected (FC) layer* permite relacionar as *features* de alto nível extraídas nas camadas anteriores com um determinado número de classes. Contudo, será necessário transformar este *feature map* num vetor unidimensional (*flatten*). Como tal, no *FC layer* são feitas todas as conexões possíveis da camada anterior à seguinte, resultando num influência direta nos valores do vetor de saída por parte de cada valor do vetor de entrada. Por fim, com o intuito já descrito a camada da função de ativação *softmax*.

3.2.2 *Backbones* para Extração de *Features*

A evolução das CNNs, atualmente desenvolvidas como *Deep Convolutional Neural Networks* (DCNNs), mostraram-se capazes de extrair *features* mais complexas, resultando numa melhor performance nas diferentes tarefas em que são aplicáveis. Cada arquitetura diferencia na sua complexidade ao variar o número de camadas e número de filtros usados. Ao longo dos anos, várias redes foram propostas, tornando-se famosas pela adaptabilidade a diferentes modelos de DL. Concretamente, a sua aplicabilidade na extração de *features* no início da rede apela-as como *backbones*. AlexNet, *VGGNet* e *ResNet* são exemplos de *backbones* utilizadas nas soluções presentes no Capítulo 2.

Existem, na literatura diversos artigos focados na análise do custo computacional, complexidade, precisão e velocidade de inferência [117, 118], como tal, baseiam-se em vários *benchmarks* e *datasets* para avaliação, *e.g.*, *ImageNet-1k*. Veja-se a Figura 3.4a, onde é possível analisar a relação entre a *Top-1 accuracy* e a complexidade do modelo (tamanho das bolas indicativo do número de parâmetros) medida em *Floating Point Operations per Second* (FLOPS) numa única passagem pelo mesmo através de uma *Graphics Processing Unit* (GPU) NVIDIA Titan X Pascal. O gráfico contempla 44 variantes de arquiteturas, incluindo os vencedores da competição anual ImageNet, *e.g.*, AlexNet, ResNet e Inception. Aparentemente não existe uma relação entre a complexidade do modelo e a precisão em tarefas de classificação,

e.g., *SENet-154* necessita de cerca do triplo do número de operações comparativamente ao *SE-ResNeXt-101(32x4d)*, tendo a mesma precisão. O mesmo se aplica na relação entre número e parâmetros e precisão, *e.g.*, *VGG-13* com elevado número de parâmetros comparativamente ao ResNet-18 para a mesma precisão. Destaque para o *SE-ResNeXt50 (32x4d)* ao atingir a maior precisão mantendo baixo o nível de complexidade (aproximadamente 2,76 milhões de parâmetros).



(a) *Top-1 accuracy vs.* complexidade computacional

(b) *Top-1 accuracy vs.* fps processados (*batch size* de 1)

Figura 3.4: Análise de performance de 44 CNNs numa NVIDIA Titan X Pascal [117].

Ainda na Figura 3.4b é comprovada a evidência de que arquiteturas com maior precisão tendem a aumentar o tempo de inferência. Num extremo, ResNet-34 destaca-se com uma precisão de 73,27% para mais de 250 fps enquanto no outro extremo, a cerca de 30,96 fps, o *NASNet-A-Large* com uma precisão de 82,50% evidenciando a sua potencialidade numa aplicação em tempo real.

Ao longo desta subsecção serão mencionadas as principais famílias de *backbones* e arquiteturas para diferentes tarefas, *e.g.*, classificação de imagem, deteção de objetos e outras tarefas.

AlexNet

AlexNet [16] foi a primeira DCNN a ganha a competição anual ImageNet de 2012. Desenvolvida Krizhevsky *et al.*, o modelo, Figura 3.5, é composto por 8 camadas: 5 convolucionais com *max pooling* para redução do número de parâmetros, 2 *FC layers* com 4096 neurónios e uma última camada de *softmax* para classificação nas 1000 classes diferentes. O tamanho da imagem de entrada é fixo $227 \times 227 \times 3$ com 96 *kernels* de tamanho $11 \times 11 \times 3$, *stride* de 4 pixeis e *padding* de 0, seguindo toda uma lógica dimensional conforme analisado nas equações da Subsecção 3.2.1.

Sabendo que a camada de entrada e de *pooling* numa CNN tradicional não apresenta parâmetros passíveis de serem treinados então, a cada camada de convolução com um determinado número de filtros/canais de saída ($n_{filters}$) quadrados ($k \times k$)

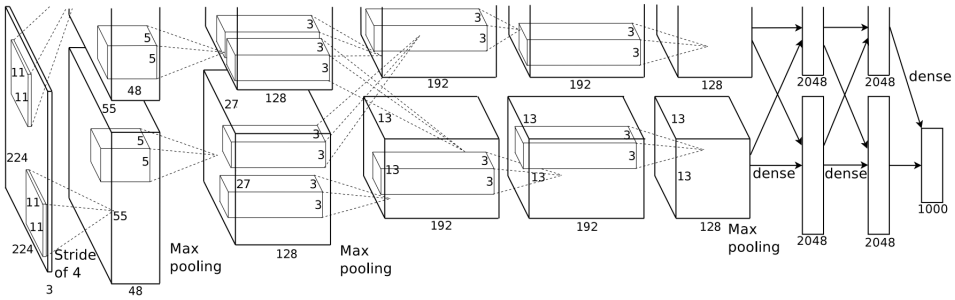


Figura 3.5: Arquitetura da AlexNet [16].

e uma determinada profundidade/número de canais na entrada da camada de convolução ($n_{channels}$) correspondente ao número de filtros da camada de convolução anterior, temos um número de parâmetros ($P_{conv} = 0$) dado por:

$$P_{conv} = (k^2 \times n_{channels} + 1) \times n_{filters} \tag{3.21}$$

Eventualmente, a soma de 1 diz respeito aos termos das *bias* para cada filtro. Por sua vez, os *FC layers* apresentam um valor bastante superior pelo facto de todos o neurónios da camada anterior ($previous_{neurons}$) estarem conectados aos neurónios da camada atual ($current_{neurons}$). Entre um *convolutional layer* (com $n_{filters}$ e um *tensor* $(g \times g)$) e um *FC layer* temos:

$$P_{conv,FC} = g^2 \times n_{filters} \times current_{neurons} + 1 \times current_{neurons} \tag{3.22}$$

E entre dois *FC layers* temos:

$$P_{FC,FC} = previous_{neurons} \times current_{neurons} + 1 \times current_{neurons} \tag{3.23}$$

A título de exemplo, na primeira camada de convolução $P_{conv} = (11^2 \times 3 + 1) \times 96 = 34.944$ parâmetros, no primeiro *FC layer* $P_{conv,FC} = 6^2 \times 256 \times 4096 + 4096 = 37.752.832$ parâmetros e no *softmax layer* $P_{FC,FC} = 4096 \times 1000 + 1000 = 4.097.000$ parâmetros.

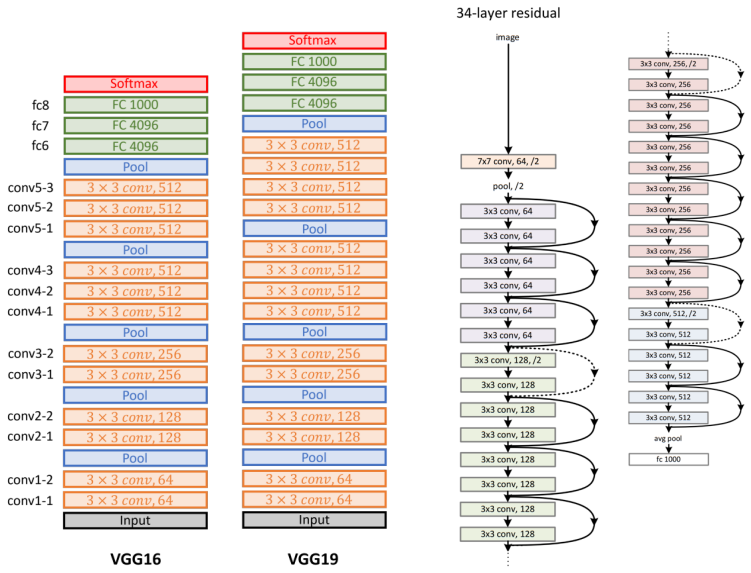
Durante o treino foi mantida a função de ativação ReLU com um método de de regularização (*dropout*) para reduzir o *overfitting* nas camadas FC. Em suma, a AlexNet conta com 60 milhões de parâmetros e 650 mil neurónios. E como tal, apesar da estrutura simples e da estratégia para evitar *overfitting*, o número elevado de parâmetros pode revelar-se uma desvantagem em pequenos *datasets*.

A sua utilização é frequente nos dias atuais, sendo o *backbone* de muitos modelos de deteção e segmentação, *e.g.*, *Region-Based CNN* (R-CNN) [119] e também o selecionado para extração de *features* no projeto *DeepPose* [49].

VGGs

A família VGGs [120] (VGG-16 e VGG-19) são dos *backbones* mais utilizados em tarefas de análise e visão computacional (*e.g.*, classificação de imagens e deteção de objetos). Assim como a AlexNet, viu-se aplicada em arquiteturas mais complexas como *Faster* R-CNN [18]. Desenvolvidas em 2014, apresentam-se como modelos mais profundos do que a AlexNet, sendo idênticos e unicamente distinguíveis pelo número de camadas de convolução presentes. Assim sendo, a VGG-16, Figura 3.6a, contem 13 camadas convolucionais e 5 *max pooling layers*, finalizando com 3 *FC layers*, originando um total de 16 camadas. Tal como a rede analisada anteriormente, recorre à função de ativação ReLU durante o treino com o *dataset* ImageNet e finaliza com uma camada de *softmax* para classificação nas 1000 classes diferentes. Naturalmente, a VGG-19 conta com um maior número de parâmetros, sendo 144 milhões comparativamente aos 138 milhões da VGG-16 [118].

A sua simplicidade e aplicabilidade a diversas tarefas com várias camadas de convolução e pequenos filtros torna-a relativamente eficaz, contudo, pouco eficiente com custos computacionais potencialmente elevados dado o número considerável de parâmetros (propicio a *overfitting* se não for devidamente regularizada)



(a) VGG (b) ResNet

Figura 3.6: Arquitetura da VGG-16, VGG-19 e ResNet-34 [118].

ResNets

A introdução das ResNets [60] por He *et al.* em 2015 definiram um novo marco no campo do *deep learning*. A adição excessiva de camadas de convolução tendia a saturar a precisão do modelo e potencializava a degradação da performance da rede. Para tal, as ResNets, Figura 3.6b, introduziram o conceito *skip connections*

(*residual blocks*) entre camadas e *batch normalization* após a convolução e antes da função de ativação para evitar o problema de *vanishing gradients*. Desta forma, a rede seria capaz de perceber se uma determinada camada é passível de ser ignorada pela baixa necessidade nos recursos aprendidos.

Concretamente, cada *residual block* é compostos por um determinado número de camadas de convolução seguidas de camadas de ativação ReLU e um "atalho" que deve permitir a soma da entrada original à saída da última camada de convolução. Este facto evidencia a sua modularidade para diferentes níveis de profundidade. Como tal, trata-se de uma rede extremamente profunda, treinada em vários *datasets*, *e.g.*, ImageNet para classificação e *COCO* para deteção e segmentação de objetos. Ainda assim, conta com diversas variantes, como ResNet-50, ResNet-101 e ResNet-152 respetivamente com 25,6, 44,5 e 60,2 milhões de parâmetros [118].

Com o passar dos anos, variantes como *ResNeXt* e *SE-ResNeXt* surgiram. Nesta arquitetura ocorreu uma substituição das camadas consecutivas em cada bloco por ramificações de camadas em paralelo para uma redução do número de hiperparâmetros ajustáveis e uma concatenação de *features* no final de cada *residual block*.

Ainda assim, o modelo demonstrou a sua intemporalidade com diversas aplicações de PSE (*e.g.*, PARE [30], HybrIK [23] e OCHMR [24]).

Inception

Inception é uma série de DCNNs proposta numa primeira versão (*Inception-v1* ou *GoogleNet* [51]) em 2014. Projetada para ser mais eficiente do que os modelos existentes, segue uma lógica análoga às ResNet. Como tal, é composta por blocos de Inception (disposta em 22 camadas), Figura 3.7, constituídos por várias camadas de convolução com diferentes tamanhos de filtros (1×1, 3×3, 5×5) executados em paralelo e devidamente concatenados na saída final [118]. Esta abordagem permite que a rede aprenda a extrair *features* a diferentes escalas e resoluções de forma eficiente e com o menor número de camadas de convolução, o que se revela essencial para tarefas de deteção de objetos numa imagem. De realçar a adoção por camadas de *average pooling* em vez das convencionais *max pooling* usadas nas ResNets e VGGs. Todo este processo recorre a apenas 5 milhões de parâmetros.

Este modelo procura reduzir o efeito do *vanishing gradients*, contudo é verificado uma grande sensibilidade à inicialização dos pesos da rede, pelo que uma inicialização inadequada poderá resultar em gradientes instáveis e baixo desempenho. O mesmo se verifica com o processo de regularização inapropriado que potencializar o *overfitting*.

Assim como nas arquiteturas anteriores, outras variantes, com melhorias de performance surgiram. *BN-Inception* adiciona *batch normalization* às camadas de convolução acelerando a eficiência do treino e melhorando a generalização da rede. *Inception-v2* e *Inception-v3* foram propostos com objetivo de aumentar a eficiência da rede mantendo o baixo número de parâmetros (21,8 milhões de parâmetros) e

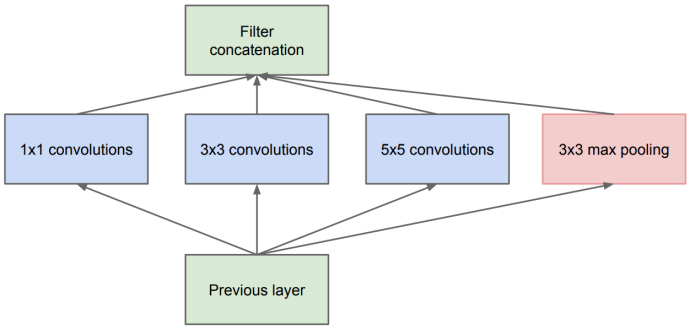


Figura 3.7: *Inception module* [51].

substituição dos filtros das várias camadas de convolução (utilização de filtros com tamanhos de 1×1 e 3×3). Por sua vez, num modelo híbrido, a *Inception-ResNet*, proposto em 2016, que utiliza os *residual connections* dentro dos blocos *inception* para melhor eficiência durante o treino e desempenho da rede. Concretamente, Inception-ResNet-v2 é composta com 160 camadas de convolução e 4 de *max pooling* para um total de 55 milhões de parâmetros.

DenseNet

Numa CNN tradicional o número de *layers* L era indicativo do número de ligações. Contudo, na DenseNet são verificadas $L(L+1)/2$ conexões. Isto é, o *feature map* à saída da cada camada é usado como entrada nas próximas camadas, permitindo que a rede opere com um menor número de filtros, reduzindo, implicitamente, o número de parâmetros durante o treino e a generalização do modelo pelo facto de lidar com *features* obtidas em diferentes níveis. Concretamente, o número de filtros usados em cada camada de convolução é aumentando, dentro de cada *dense block*, segundo uma constante k apelada de "*growth rate*", tornando o sistema modelar e facilmente ajustável uma vez que o número de parâmetros depende desta constante (*e.g.*, DenseNet-100 ($k = 12$) com 7 milhões de parâmetros e DenseNet-100 ($k = 24$) com 27,2 milhões de parâmetros).

Assim como na Figura 3.8, cada camada recebe um agregar de conhecimento (concatenação) proveniente das camadas anteriores. Entre os *dense blocks* decorre uma camada de transição composta por um *batch normalization layer* e uma camada de convolução 1×1 seguido de um 2×2 *average pooling layer*.

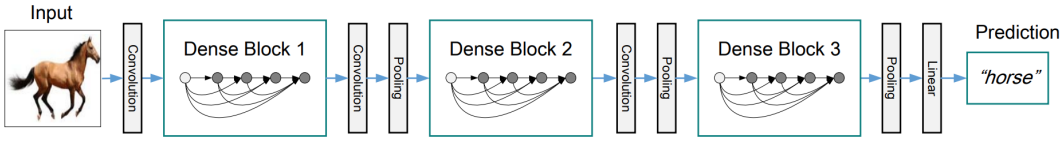


Figura 3.8: DenseNet com três *dense blocks* [51].

Naturalmente, este elevado número de conexões pode aumentar o tempo de treino, comparativamente às anteriores arquiteturas. Ainda assim, o método permite a introdução do conceito de *deep supervision* numa influencia direta das diferentes saídas intermédias ao longo da rede na *loss function*. Na verdade, um único classificador no final da rede fornece supervisão direta a todas as camadas através de dois ou três *transition layers*.

MobileNet

MobileNet [121] surgiu como um modelo de *deep learning* adequado a equipamentos com menor desempenho, *e.g.*, *edge devices* e *embedded applications* como *NVIDIA Jetson*, *Raspberry Pi* e *Google Coral*.

Esta arquitetura introduziu a técnica de *"separable convolutions"* para redução do número de parâmetros do modelo. Nesta, a operação de convolução é dividida em *depthwise convolution* e convoluções 1×1 (*pointwise convolution*). Para tal, a *pointwise convolution* combina as saídas originadas na *depthwise convolution* pela aplicação de um filtro a cada canal de entrada.

Deste modo, o método revela-se extremamente eficiente pela independência da primeira convolução dos canais de saída (filtros da camada em causa). Numa CNN tradicional seria dedutível $k^2 \times n_{channels} \times n_{filters}$ pesos (Equação 3.21), contudo, nesta arquitetura, cada camada de convolução admite $k^2 \times n_{channels} + n_{channels} \times n_{filters}$, sendo a primeira parcela referente aos pesos da *depthwise convolution* e a segunda referente à *pointwise convolution*.

Ainda assim, esta rede terá sido treinada e testada com o *dataset* ImageNet para classificação de imagens, tendo os autores definido dois novos hiperparâmetros que possibilitam a alteração da natureza do modelo: baixa latência ou "leve" em termos de custo computacional.

Atualmente conta com mais duas versões (MobileNet-v2 e MobileNet-v3) tendo sido introduzido na MobileNet-v2 [122] a possibilidade de deteção de objetos e a adição de *inverted residual blocks* (análogo aos *residual blocks*) na sua arquitetura para potencializar liberdade à rede de "salta" determinadas camadas de convolução. A *SOTA performance* foi atingida pelo MobileNet-v3 [123] com diferentes tamanhos de modelos orientados para aplicações processadas em CPUs de telemóveis.

A sua aplicação é atual e combinada com arquiteturas inovadoras como *Transformer networks* atingem resultados promissores, *e.g.*, *Graph Transformer network for human mesh ReconStruction* (GTRS) de Zheng *et al.* [124].

HRNet

A HRNet [40] ou *HRNet-v1* é uma arquitetura de CNN completa, desenvolvida para manter elevada qualidade do *feature map* ao longo de todo o treino. Na sua

primeira versão, proposta em 2019 por Sun *et al.*, verifica-se uma organizada em várias ramificações paralelas para operar a diferentes escalas e evitar a perda de informação proporcionada pelas técnicas de *downsampling* e *pooling*.

Por natureza, uma CNN tradicional tende a comprimir a entrada numa abordagem *downsampling* e, portanto, a reconstrução da imagem é realizada pelo processo inverso, o *upsampling* (*feature map* de saída superior, em termos de dimensão, ao *feature map* de entrada) através de um *transposed convolutional layer*. Neste sentido, a profundidade do *backbone*, Figura 3.9, dividida em diferentes sub-redes, é passível de ser ajustada em função da tarefa em causa.

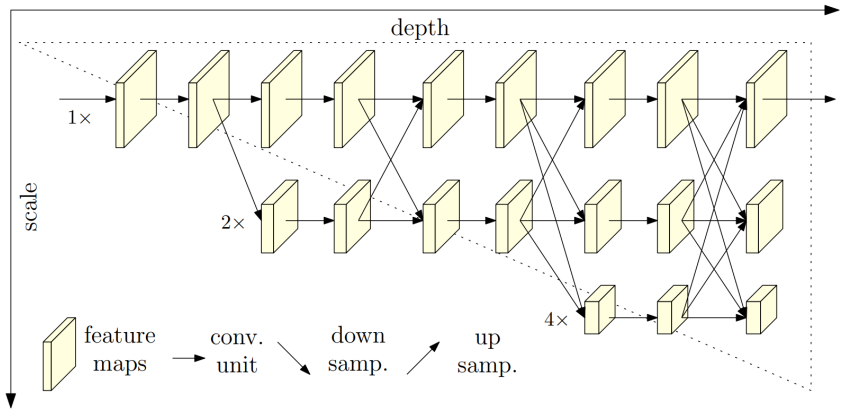


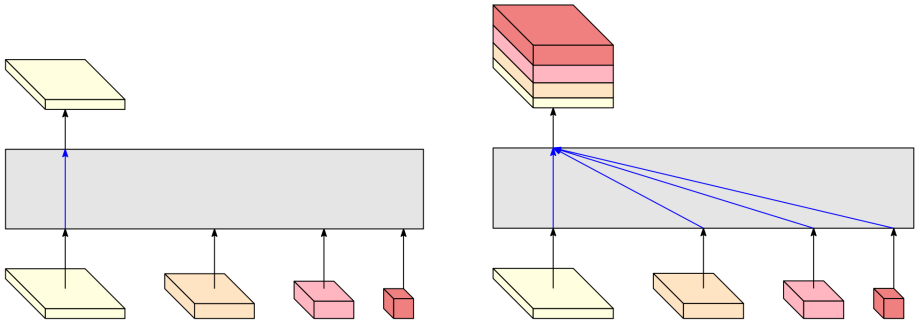
Figura 3.9: Ilustração da arquitetura da HRNet proposta por Sun *et al.* [40].

O processo é iniciado com uma sub-rede de alta resolução (que deverá ser mantido até ao fim), seguido de um número de estados síncrono com o número de *high-to-low resolution subnetworks*, conectadas em paralelo. Como tal, as *features* obtida em diferentes escalas são partilhadas e agregadas com as diversas camadas numa estrutura completamente conectada.

Porém, a complexidade da rede e custo computacional é escalável, potencializando um tempo de treino elevado mediante do tamanho da rede (*e.g.*,HRNet-W32 eHRNet-W48 com 28,5 e 63,6 milhões de parâmetros respetivamente.

Esta arquitetura permitiu à HRNet alcançar surpreendentes desempenhos na deteção de poses humanas em tempo real. A sua precisão aumentou a confiança nos *keypoints* das articulações obtidos de *heatmaps* com maior resolução. Numa segunda versão HRNet-v2 permitiu a deteção de objetos, segmentação de imagens e e até mesmo reconhecimento facial.

Na verdade, a HRNet-v2 [125] introduziu uma estratégia de fusão entre as sub-redes de diferentes resoluções, Figura 3.10b. Ao invés de gerar unicamente na saída um *feature map* de elevada resolução, Figura 3.10a, realiza uma concatenação cruzada das diferentes escalas (através de um *upsampling* prévio).



(a) Representação da alta resolução proposta em *HRNet-v1*

(b) Concatenação dos vários *feature maps* (upsampled) proposta em HRNet-v2

Figura 3.10: *Feature maps* de saída entre sub-redes nas diferentes implementações [125].

3.3 Deteção de Objetos e Segmentação

3.3.1 Deteção de Objetos

Os *backbones* referidos são constituintes essenciais em parte das implementações analisadas no Capítulo 2. Factualmente, a sua necessidade deve-se à eventual capacidade de deteção e reconhecimento de pessoas e/ou *features* relevantes (*e.g.*, articulações e forma do corpo) nas imagens.

A Figura 3.11 mostra uma arquitetura de rede simples para deteção de objetos. Naturalmente, é iniciada com um conjunto de camadas de convolução para extração de *features* e alguns *FC layers* para posteriormente se dividir em duas ramificações (*heads*). Um ramo dedicado à classificação com uma estrutura análoga à analisada anteriormente e um outro ramo para resolução do problema de regressão (*bounding box regression*) ao determinar os parâmetros da *bounding box*, *e.g.*, duas coordenadas (x, y) para o canto superior esquerdo e dois parâmetros de largura e altura (w, h). De notar a necessidade de *FC layers*, neste último ramo, sem necessidade de uma função de ativação (saída linear).

Esta subsecção dedica-se ao estudo superficial de modelos baseados na estrutura das CNNs para deteção de objetos, dividido-se em abordagens *multi-stage*, *e.g.*, variantes da R-CNN, e *single stage*, *e.g.*, família *You Only Look Once* (YOLO).

Abordagens *multi-stage*: R-CNN, *Fast* R-CNN e *Faster* R-CNN

A ***Region-Based CNN*** **(R-CNN)** proposta por Girshick *et al.* [119] combina o DL com técnicas tradicionais de visão computacional numa *pipeline* descrita na Figura 3.12. Em vez de uma abordagem *sliding window*, o método, num pré-processamento inicial, identifica aproximadamente 2000 regiões quadradas que poderão conter objetos de interesse. Este algoritmo vinha a ser substituído, no *Faster* R-CNN, por uma técnica baseada em DL.

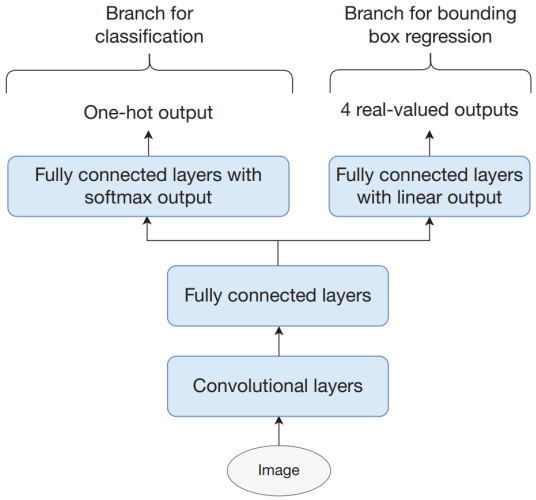


Figura 3.11: *Pipeline* de uma rede neuronal para deteção de objetos [126].

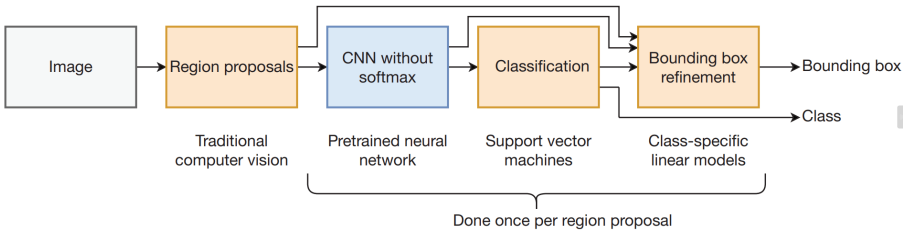


Figura 3.12: Visão geral da arquitetura da R-CNN. [126].

A essência desta arquitetura aloca-se nos seguintes três constituintes. Cada região proposta é submetida a uma CNN pré-treinada no ImageNet com base na arquitetura AlexNet. Eventualmente, a última camada terá sido removida obtendo na saída um *feature vector* com 4096 elementos usado como entrada para o processo de classificação (*Support Vector Machines* (SVM) para classificação binário) e *bounding box regression*.

Ainda que o número de regiões propostas seja inferior ao resultado da aplicação de uma *sliding window*, métricas como *Intersection over Union* (IoU) foram incorporadas para lidar com regiões sobrepostas. Para $\boldsymbol{K}$ classes disponíveis, são treinados $\boldsymbol{K}$ *linear regression models* para que a cada classificação seja possível determinar os parâmetros das *bounding boxes* pelo respetivo *linear regressor*.

A elevada passagem de regiões propostas pela CNN contribui para atenuação da performance do modelo. O ***Fast* R-CNN** [127] apresentou consideráveis mudanças estruturais ao substituir o *backbone AlexNet* pela VGG-16 e, em vez de SVM e *linear regression*, recorrer a redes neuronais para classificação e ajuste dos parâmetros das *bounding boxes*.

A primeira etapa do *Fast* R-CNN, Figura 3.13, é a passagem da imagem inteira por um conjunto de camadas convolução e *max pooling* da VGG-16 pré-treinada.

Novamente, os autores removeram os dois últimos FC e *softmax layers*, resultando num *feature map* com dimensões $\frac{W}{32} \times \frac{H}{32}$, sendo a imagem de entrada definida com as dimensões $W \times H$[1].

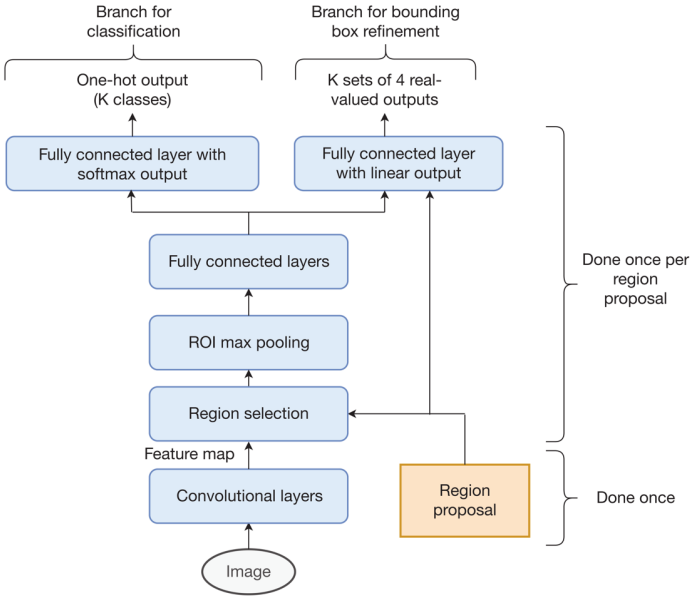


Figura 3.13: Visão geral da arquitetura da *Fast* R-CNN [126].

Apesar do progresso, a *Fast* R-CNN ainda se baseia na proposta das 2000 regiões. Contudo, é passível a projeção destas regiões no *feature map* por um processo apelidado de *RoI projection*. Serão então estas *features* que deverão ser usadas como entrada da nossa rede de classificação.

Não obstante, os autores adicionaram um *RoI max pooling layer* para converter o RoI do *feature map* para uma dimensão fixa de 7×7 que será exatamente o tamanho esperado nas entradas dos *FC layers* removidos. Nesta fase, a rede é capaz de classificar o objeto numa das várias classes ou *background* através de um *softmax layer*. Enquanto isto, uma *FC network* opera em paralelo para estimar os parâmetros das diferentes *bounding boxes*.

Recordando, a R-CNN apenas utiliza técnicas de DL para extração de *features* enquanto *Fast* R-CNN recorre ao DL nas etapas de classificação e previsão das *bounding boxes*. Contudo, as regiões propostas continuam a ser definidas por abordagens tradicionais. Neste sentido, a implementação da ***Faster* R-CNN** [18] introduz uma RPN, Figura 3.14 (direita), num modelo que recorre ao *feature map* estimado pelas camadas convolucionais da imagem completa.

A RPN, apresenta uma estrutura idêntica à do processo final de classificação e *bounding box refinement*, como tal, é composta por um *FC ReLU layer* seguido de dois *FC layers* em paralelo para gerar K regiões e respetivas localizações. A

[1] O denominador (32), apelidado de *sub-sampling ratio*, provém dos cinco *pooling layers*, cada um com uma redução da dimensão por um fator de dois.

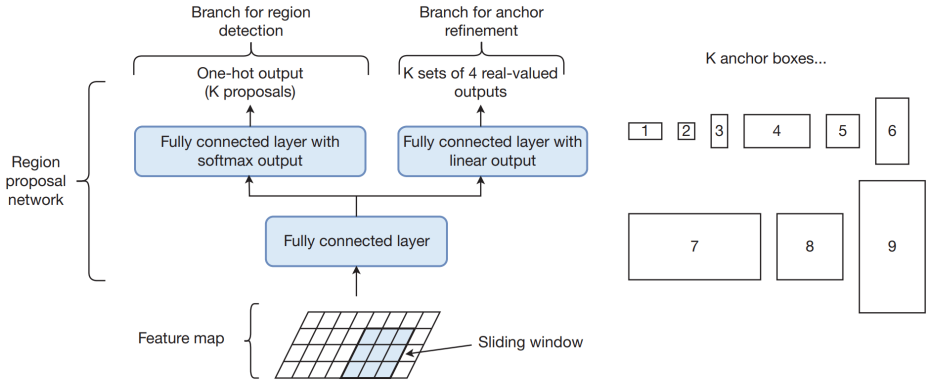


Figura 3.14: Equerda: RPN. Direita: *K Anchor boxes* [126].

rede inclui um mecanismo conhecido por *anchor boxes* com um dado tamanho e rácio centrado num dada posição. Concretamente, os autores usaram 128, 256 e 512 pixeis com rácios de 1:2, 1:1 e 2:1, resultando nas combinações presentes na Figura 3.14 (esquerda) e, como tal, a região final proposta é dada pela combinação das coordenadas determinadas pela rede com uma *anchor box* específica [126].

Tendo as regiões propostas, o restante processo é idêntico à *Fast* R-CNN, segundo a arquitetura da Figura 3.15. Neste cenário, a técnica de *sliding window* com 9 *anchor boxes* (apenas 9 classes nesta rede) mostrou-se eficiente pela diminuição da área de procura e das dimensões do *feature map* (32 vezes inferior à imagem de entrada).

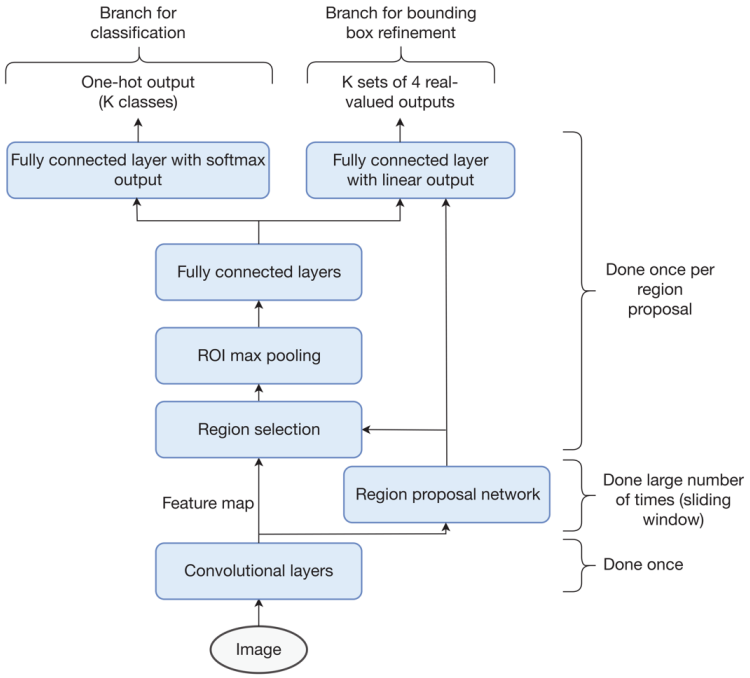


Figura 3.15: Visão geral da arquitetura da *Faster* R-CNN [126].

Abordagens *one stage*: Família YOLO

Em abordagens modernas a deteção de objetos baseia-se em modelos *single shot/one stage* pelo seu baixo tempo de latência e desempenho, por vezes superior a modelos *multi-stage*, em várias métricas. A enumeração e descrição das diferentes arquiteturas baseadas em CNNs é atualmente extensa, pelo que o propósito desta subsecção se restringe à apresentação do conceito geral das mais relevantes versões da família YOLO.

You Only Look Once **(YOLO)** é uma família de modelos de visão computacional para deteção e segmentação de objetos em tempo real. Atualmente, encontra-se na oitava versão, com melhorias continuas (Figura 3.16) desde o primeiro lançamento no CVPR 2016 por Redmon *et al.* [128]. Terven *et al.* [129] fundamentou uma análise abrangente desta evolução, ao referir as inovações e contribuições a cada iteração. Como tal, no decorrer desta subsecção serão abordadas evidencias técnicos de algumas versões que motivaram a sua aplicação em diferentes soluções revistas no Capítulo 2 e direcionadas à implementação no Capítulo 4.

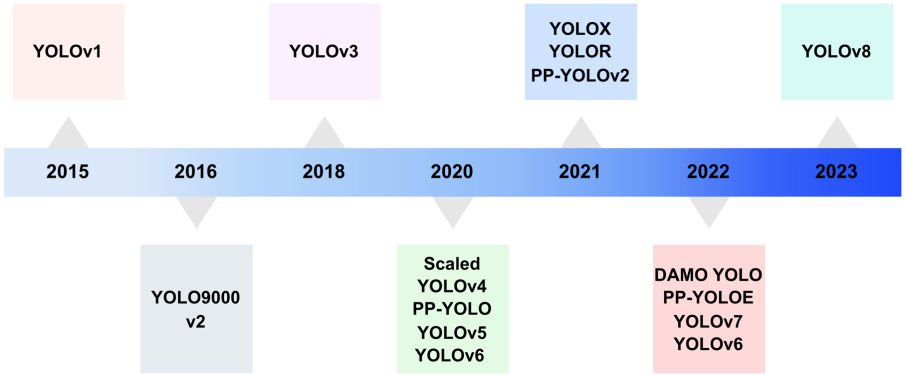


Figura 3.16: Evolução temporal das versões YOLO.

O YOLO realiza a tarefa de deteção de objetos com uma única passagem da imagem pela rede, contrariamente às abordagens anteriores que recorriam a *sliding windows* seguidos de classificadores com inúmeras passagens pela rede. Além disso, o YOLO apresenta a saída unicamente baseada numa *single bounding box regression*, em oposição às arquiteturas R-CNN com duas saídas (classificação e *bounding box regression*).

Uma análise à arquitetura do **YOLOv1** (Figura 3.17) permite compreender como este processo é realizado. Efetivamente organiza-se em 24 camadas convolucionais seguidas por dois *FC layers*, sendo que todos têm implementado a função *LReLU*, à exceção da última camada com uma função de ativação linear para classificação. Verifica-se que adoção por camadas de convolução 1×1 permitem a redução do tamanho do *feature map* enquanto mantém baixo o número de parâmetros a estimar.

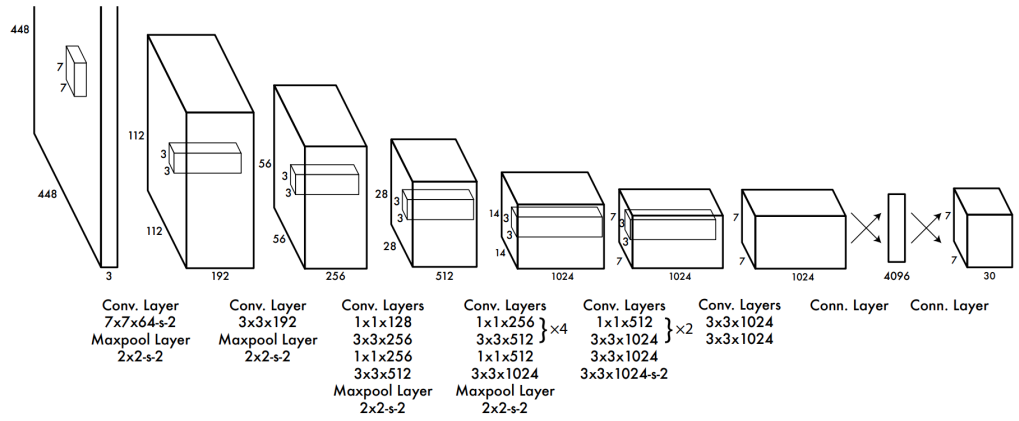


Figura 3.17: Arquitetura do YOLOv1 [128].

Neste primeiro modelo, o funcionamento é simples e constituiu-se como base para as seguintes versões. A imagem é dividida num conjunto numa grelha $S \times S$ para prever B *bounding box* da mesma classe, com uma dada confiança, de todas as classes possíveis C. O resultado da rede é um *tensor* com dimensões $S \times S \times (B \times 5 + C)$. A constante 5, da parcela que define a profundidade da saída da rede, remete para o número de parâmetros identificáveis em cada *bounding box* estimada: $\boldsymbol{P}_c, \boldsymbol{b}_x, \boldsymbol{b}_y, \boldsymbol{b}_h$ e $\boldsymbol{b}_w$, respetivamente *confidence score* da caixa que potencialmente contem um objeto, com centro $(\boldsymbol{b}_x, \boldsymbol{b}_y)$ face ao centro da imagem e altura $\boldsymbol{b}_h$ e largura $\boldsymbol{b}_w$ fracionaria das dimensões da imagem.

Esta implementação assumiu um treino fracionado, tendo as primeiras 20 camadas de convolução sido treinadas com o *ImageNet dataset* (224×224), seguidos as 4 camadas com *PASCAL VOC 2007* e *VOC 2012 datasets* (448×448) para aumentar o detalhe na deteção de objetos. Além disso, a complexa *loss function* é dada pela soma de múltiplos *Sum-Squared Errors* (SSE) com diferentes contribuições dos elementos constituintes: *localization, confidence e classification loss* [129].

Contudo, a simplicidade do modelo refletiu-se na dificuldade em detetar objetos de pequena dimensão fruto do número reduzido de classes passíveis de serem identificadas por cada célula da grelha definida. Neste sentido, **YOLOv2** ou **YOLO9000** potencializou a deteção de um total de 9000 categorias. Efetivamente contou com melhorias estruturais ao incluir *batch normalization* em todas as camadas convolucionais, remoção do *FC layer* para uma arquitetura puramente convolucional, uso de *anchor boxes* para estimar as *bounding boxes* e um treino com imagens a diferentes escalas (320×320 a 608×608) para atenuar a principal desvantagem do YOLOv1. Para cada célula eram estimadas cinco *bounding boxes* com cinco parâmetros: $\boldsymbol{t}_x, \boldsymbol{t}_y, \boldsymbol{t}_w, \boldsymbol{t}_h$ e $\boldsymbol{t}_o$. As quatro distâncias definidas no YOLOv1 eram deduzíveis destes parâmetros, sendo $\boldsymbol{t}_x$ e $\boldsymbol{t}_y$ um deslocamento face ao centro da *bounding box* e os parâmetros de dimensão desta dados pelo produto valor escalar pelo exponencial

respetivo (e^{t_w} ou e^{t_h}). Com estas alterações YOLOv2 atingiu uma *Average Precision* (AP) de 78,6% a 40 fps frente aos 63,4% do YOLOv1 no *PASCAL VOC2007 dataset* [129], ambos numa Geforce GTX Titan X (modelo original).

Ainda assim, é na terceira versão desta família, o **YOLOv3** [130], apresentado por Redmon *et al.* em 2018 que se verificaram as maiores alterações a nível de arquitetura. Além dos valores estimados pelo YOLOv2 para cada *bounding box*, ainda é determinado um *objectness score* (valores entre 0 e 1) que deverá ser tão alto quanto maior for o sobreposição da *anchor box* sobre a *ground truth*. A cada localização são feitas três *predictions*, com *anchor boxes/prior boxes* a diferentes escalas, compostas pelos parâmetros da *bounding box*, o *objectness score* e as 80 classes (definidas no *COCO dataset*), *i.e.*, $N \times N \times [3 \times (4+1+80)]$ *predictions*. A camada de classificação dada pela função *softmax* foi substituída por BCE, permitindo que várias *labels* sejam atribuídas à mesma caixa (*e.g.*, mulher e pessoa). Tudo isto, com um novo *backbone*, a *Darknet-53* com 53 camadas de convolução e *residual connections*.

Destacam-se a *multi-scale predictions* do YOLOv3 que, apesar de provir de um aumento de complexidade da rede, originou um AP_{50} (*threshold* IoU de 0.5) de 57,9 % a aproximadamente 20 fps numa Geforce GTX Titan X frente aos 44 % do YOLOv2 no *MS COCO dataset*. Por esse motivo, atuou com detetor de humanos ou poses 2D [131] em diversos projetos de PSE.

YOLOv4, com aplicabilidade em vários projetos de PSE [15, 100], surge em 2020 por Bochkovskiy *et al.* [132]. Autores diferentes, mantiveram o princípio, mas aumentaram a complexidade da rede. Esta admite uma camada intermédia apelada de *"neck"* para interligação do *backbone* (*Darknet-53* para extração de *features*) com a *head* (usada para prever classes e *bounding boxes* de objetos). Concretamente, para o *neck* foi utilizado uma versão modificada do *Spatial Pyramid Pooling* (SPP) e da *Path Aggregation Network* (PANet) para combinar características de diferentes níveis de resolução numa única saída. Já a *head* é uma *anchor based* do YOLOv3. Ainda assim, com a adição de um *Cross-Stage-Partial-connections* (CSP) ao *backbone* permite uma redução do cálculos computacionais, mantendo a mesma precisão. Por estes motivos, o modelo também é conhecido por *CSPDarknet53-PANet-SPP*.

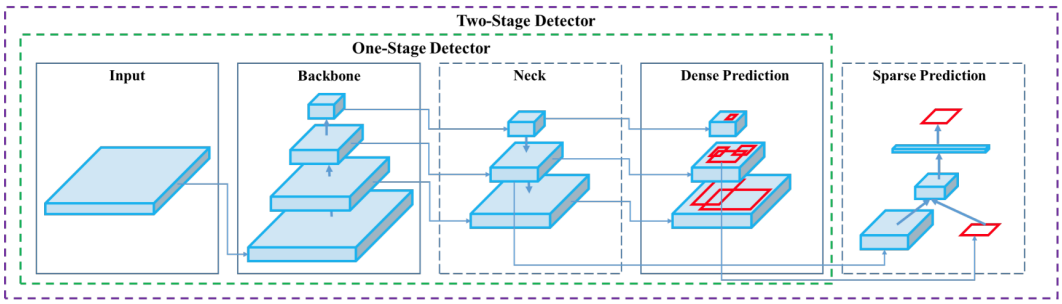


Figura 3.18: Arquitetura do YOLOv4 comparativamente a *two-stage object detector* [132].

Os resultados espelham as alterações evidentes, conseguindo um AP_{50} de 65,7% a pelo menos 30 fps numa *Pascal GPU* (valores similares numa Maxwell e Volta GPU) para o *MS COCO dataset* (test-dev 2017) [132].

Posteriormente é lançado o **YOLOv5**, pela Ultralytics, com a definição de um novo marco na família YOLO. A partir deste momento, a maioria do desenvolvimento seria realizado em Pytorch em vez Darknet (*open source neural network framework* escrita em C e CUDA). Atualmente, na sua versão 7.0 potencializa, além da classificação a realização de tarefas de *instance segmentation*. Disposta em 5 modelos: *nano*, *small*, *medium*, *large* e *extra large*, permitiu a adaptação e integração em versões móveis para iOS e Android.

Ainda do ponto de vista de análise, uma visualização gráfica da evolução do número de parâmetros descritivos do modelo e o tempo de inferência para uma métrica como o $mAP^{50\text{-}95}$ permitirá compreender a evolução efetiva das mais recentes versões da família YOLO. O repositório oficial sintetiza esta diferença, com respeito à tarefa de classificação, num simples gráfico representado na Figura 3.19. Fica evidente a diferença da última versão da Ultralytics face às restantes, mantendo um número de parâmetros semelhantes, mas com uma eficiência e mAP superior.

Concretamente, a métrica aqui apresentada contempla uma média simples da AP de todas as classes detetadas. Trata-se da métrica de avaliação oficial do *MS COCO dataset* que atua sobre diferentes IoUs de 0,5 a 0,95 com passos de 0,05.

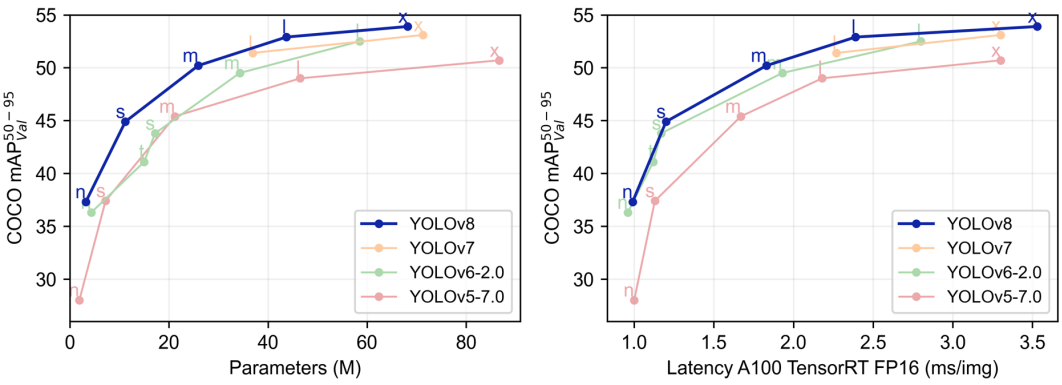


Figura 3.19: Performance Ultralytics YOLOv8 *vs.* outras versões [133].

Ainda assim, sintetizado na Tabela 3.1, uma comparação com base em resultados obtidos do *MS COCO dataset* e um treino a 300 *epochs*. A aceleração gráfica providenciada pela compilação do modelo do **YOLOv8** em TensorRT ficou evidente à medida que o mesmo se tornou mais complexo [133].

No decorrer, a sétima versão, disponibilizada pelos mesmos autores do YOLOv4 e YOLOR, surgiu em 2022 com uma contribuição que potencializou ao modelo inicial evoluir para ***YOLOv7 Pose***. Baseado no original YOLO-Pose [134], é capaz de determinar a pose 2D de vários humanos definidas em 17 *keypoints*. O modelo original é por si só baseado no YOLOv5 e YOLOX, treinado de ponta a ponta,

Tabela 3.1: performance YOLOv5 *vs.* YOLOv8 [133].

Model Size	mAP^{val} 50-95	Speed A100 (ms)	Params (M)	FLOPs (B)
YOLOv5n	28,0	0,6	1,9	4,5
YOLOv8n	37,3	0,99	3,2	8,7
YOLOv5s	37,4	0,9	7,2	16,5
YOLOv8s	44,9	1,2	11,2	28,6
YOLOv5m	45,4	1,7	21,2	49,0
YOLOv8m	50,2	1,83	25,9	78,9
YOLOv5l	49,0	2,7	46,5	109,1
YOLOv8l	52,9	2,39	43,7	165,2
YOLOv5x	50,7	4,8	86,7	205,7
YOLOv8x	53,9	3,53	68,2	257,8

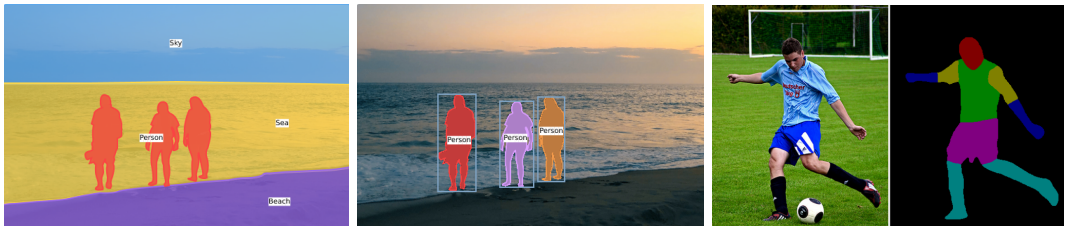
permite detetar as *bounding boxes* e as poses de cada pessoa num *single stage*, combinando o melhor das abordagens *top-down* e *bottom-up* (sem necessidade de pós processamento para agregar os *keypoints* dos esqueletos humanos).

3.3.2 Segmentação

A segmentação de imagem é uma técnica de processamento de visão computacional que procura agrupar ou classificar regiões ou segmentos semelhares numa imagem. A sua aplicação é recorrente em passos intermédios a fim de melhorar a precisão dos modelos que estimam a pose e a forma humana. A possibilidade de identificação do humano ao nível do pixel através de uma máscara estende a capacidade da rede. A combinação desta informação com o conhecimento estatístico prévio potencializado pelo SMPL permite a incorporação de um sistema de supervisão adicional, como sobreposição da silhueta humana em 2D [11]. Dificuldades a lidar com a variedade de posições corporais, formas, roupas e ambiente podem ser atenuadas com a adoção deste pré-processamento evidenciado situações que usufruam de subtração de fundo e limitação de profundidade.

A segmentação, presente nas soluções analisadas no Capítulo 2, pode ser categorizada em ***semantic segmentation*** (Figura 3.20a) e ***instance segmentation*** (Figura 3.20b). Eventualmente, um terceira hipótese, *i.e.*, *body part segmentation* (Figura 3.20c), não representativa de uma categoria, é considerada por vários autores como uma variante de *semantic segmentation* com aplicabilidade no processamento intermédio de um sistema de *HPS estimation*.

A principal diferença entre as categorias de segmentação reflete-se na forma como cada uma descreve os elementos constituintes da imagem. *Semantic segmentation* identifica unicamente os objetos pertencentes a um determinado número de classes enquanto *instance segmentation* reúne a classificação e localização individual de cada elemento de interesse. Por sua vez, *body segmentation* concretiza a divisão de um



(a) *Semantic segmentation* (b) *Instante segmentation* (c) *Body part segmentation*

Figura 3.20: Métodos de segmentação usados em *HPS estimation*.

corpo, gerando uma máscara que poderá ser associada aos *keypoints*, previamente detetados, do humano [30].

Esta subsecção dedica-se ao estudo superficial de modelos baseados na estrutura das CNNs para segmentação, focando essencialmente os principais constituintes que potencializam a incorporação deste pré-processamento para PSE.

Semantic segmentation

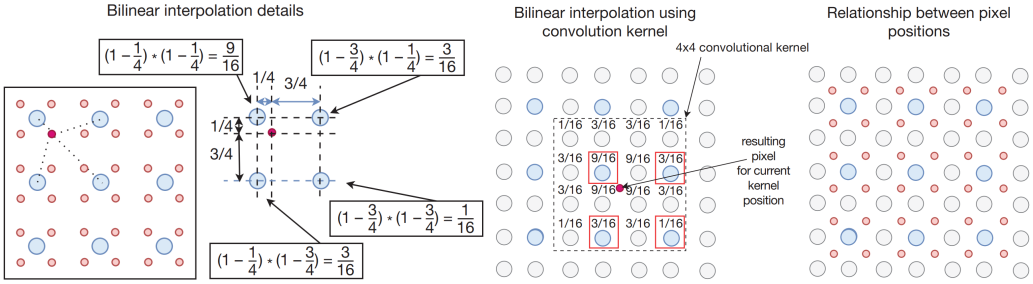
A tarefa de *semantic segmentation* envolve a classificação de cada pixel, de uma imagem, numa das classes existentes com base na textura representativa. A sua aplicabilidade verificou em [28, 105, 9], para perceção da silhueta e forma humana ou divisão do corpo em partes [30, 86, 87]. Objetos pertencentes à mesma classe serão identificados com a mesma cor. Do ponto de vista funcional, a imagem de saída terá a mesma resolução da imagem de entrada, porém o número de canais (inicialmente três) deverá agora ser tanto maior quanto mais classes houver. Por norma, este tipo de redes não apresenta *pooling layers* nem *strides* nas camadas de convolução.

Como tal, para a aplicação das tradicionais CNNs (*pooling layers* tendem a diminuir a dimensão da imagem) será necessário aumentar novamente a resolução do *feature map* nas camadas finais. Esta técnica, conhecida por ***upsampling***, pode ser realizada diferentes formas, sendo que a maioria não recorrem de DL. ***Nearest neighbor interpolation*** e ***Bilinear interpolation*** destacam-se nesta tarefa.

Naturalmente, a primeira providenciará um aumento de resolução pela atribuição do valor do pixel mais próximo na imagem original ao pixel correspondente na imagem redimensionada. Apesar da eficiência computacional deste algoritmo pode espera-se uma perceção que tende a induzir uma aparência "pixelizada" [126].

Neste sentido, *bilinear interpolation* apresenta resultados visualmente mais naturais. O processo computacional é idêntico ao apresentado na Figura 3.21a. Os novos pixeis gerados (vermelhos) são definidos com base numa relação espacial entre os pixeis originais (azuis). Felizmente, este processo pode ser implementado usando convoluções (Figura 3.21b), sendo necessário a introdução de pixeis fictícios (cinzentos) de valor 0 entre os pixeis originais. Neste momento, um determinado *kernel*

(4 × 4 no exemplo), preenchido com as relações espaciais referidas (pesos), pode percorrer a nossa imagem aumentada a fim de gerar a localização dos novos pixeis. Com filtros de tamanhos superiores, como o da Figura 3.21b, será então possível preservar a informação das bordas da imagem original.



(a) Detalhes da interpolação bilinear com 2× *upsampling*

(b) *Bilinear interpolation* implementada com recurso a convolução

Figura 3.21: *Bilinear interpolation*. *Computed (interpolated) pixel* (vermelho), *original pixel* (azul) e *zero padding* (cinzento) [126].

Usando a mesma técnica, é possível construir um *kernel* convolucional que implemente a *nearest neighbor interpolation*. Simplesmente, o *kernel* deve ser 2 × 2 com um 1 em todos os elementos [126].

Posto isto, a determinação dos pesos do *kernel* pode integrar no treino da CNN. Ainda assim, todo este processo de adição de um pixel com valor nulo e posterior convolução da imagem com um determinado *kernel* para aumento de resolução é referido, em DL, por ***transposed convolution***. Deste modo, o *feature map*, sujeito a uma camada *max pooling*, pode ser recuperado pela operação contrária, isto é, *unpooling*. Os primeiros passos desta técnica são análogos à *bilinear interpolation*, contudo, a colocação dos pixeis de valor nulo faz uso de informação anterior à operação de *max pooling* (colocação pixeis nulos em zonas diferentes da posição do máximo).

De notar que *deconvolution* é uma operação matemática capaz de reverter o efeito da convolução. Por outro lado, *transposed convolution* apenas reconstrui as dimensões da imagem original. Deste modo, as operações originam resultados diferentes. A menos que seja percetível pelo contexto, deve-se evitar o uso rotineiro do termo [126].

Posto isto, será interessante analisar a parte integrante dos modelos orientados a segmentação semântica que exigem o uso de camadas de menores resoluções no meio da rede. *SegNet*, *U-Net*, *deconvolution network* (*DeconvNet*) e FCNs são alguns exemplos de potenciais extensões lógicas das noções descritas até ao momento. O foco recairá sobre a *DeconvNet* [135] e *Hourglass network* [48] pela sua aplicabilidade corrente no tema em estudo.

Efetivamente, a ***DeconvNet*** desenvolvida por Noh *et al.* [135] contempla os conceitos apresentados. Parte da rede, Figura 3.22, é baseada numa VGG-16, com

operações de convolução e *max-pooling*, para posteriormente reconstruir as dimensões do *feature map* com um múltiplas operações de *unpooling* e *deconvolution* numa estrutura simétrica.

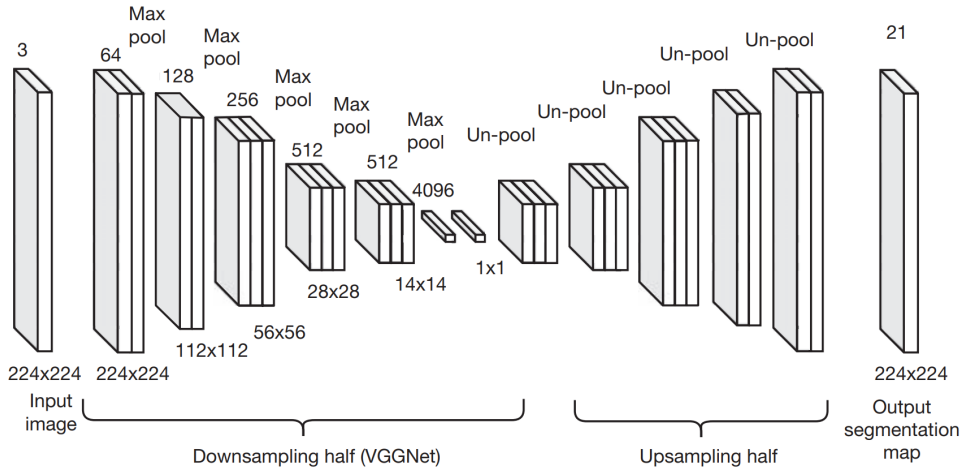


Figura 3.22: Arquitetura da *DeconvNet* e respetiva divisão baseada na *VGGNet-16* [135].

A metade final da VGG-16 é exclui da última camada de *softmax*, permanecendo unicamente os dois *FC layers*, *i.e.*, os 4096 neurónios são vistos na rede como uma camada de convolução com dimensões 1×1 e 4096 canais. E tal como referido, para a rede em causa, é obtido na saída um *segmentation map* de dimensões $224 \times 224 \times 21$ com uma imagem RGB de entrada de tamanho $224 \times 224 \times 3$, evidenciando a possibilidade de classificação de cada pixel em 20 classes ou *background*.

Ainda com foco nas soluções apresentadas no Capítulo 2, será de realçar a ***Stacked Hourglass* (SHG) *network*** [48] pela sua relação com as arquiteturas *encoder-decoder* e *convolution-deconvolution* e viabilidade ao processar informação espacial a diferentes escalas.

Por esta razão, previamente à compreensão da SHG, deve estar assente a noção de ***autoencoders***. Sucintamente, *autoencoders* são usados para reduzir uma imagem de entrada de elevada dimensão numa inferior com base em *downsampling (encoding) layers* e, posteriormente, tentar reconstruir essa representação usando *upsampling (decoding) layers*. Nesta processo, o modelo deve aprender como manter apenas a informação relevante através de uma representação mais compacta. Assim sendo, espera-se uma otimização da eficiência computacional e redução do *overfitting*.

De facto, a SHG apresenta uma topologia simétrica idêntica mas com uma essência de operação ligeiramente diferente. As camadas de *unpooling* e *deconvolution* são substituídas pelo simples processo de *nearest neighbor upsampling* e *skip connections*. Não obstante, esta implementação adota uma estrutura composta por um agregar de múltiplos *hourglass modules* para uma inferência *bottom-up* e *top-down*. Naturalmente, esta metodologia potencializou a aplicação de supervisão intermédia ao longo *heatmaps* gerados, passíveis de serem sujeitos a uma determinada *loss*.

Instance segmentation

Em *instance segmentation*, instâncias diferentes devem apresentar cores diferentes, mesmo que pertençam à mesma categoria. Fica evidente que o problema em causa é, de certa forma, a combinação de deteção de objetos e segmentação semântica uma vez que existe a necessidade de identificar o pixel que está associado a um dado objeto.

Posto isto, ***Mask R-CNN*** [67] apresenta-se como o *SOTA model* para *instance segmentation*. Esta extensão do *Faster R-CNN* permite a deteção com *bounding box* e segmentação de objetos através de uma terceira ramificação que opera em paralelo. Tal como presente na arquitetura da Figura 3.23, o *feature map* é usado como entrada numa rede responsável por fazer o *upsampling* e gerar na saída uma *instance mask* referente ao objeto identificado.

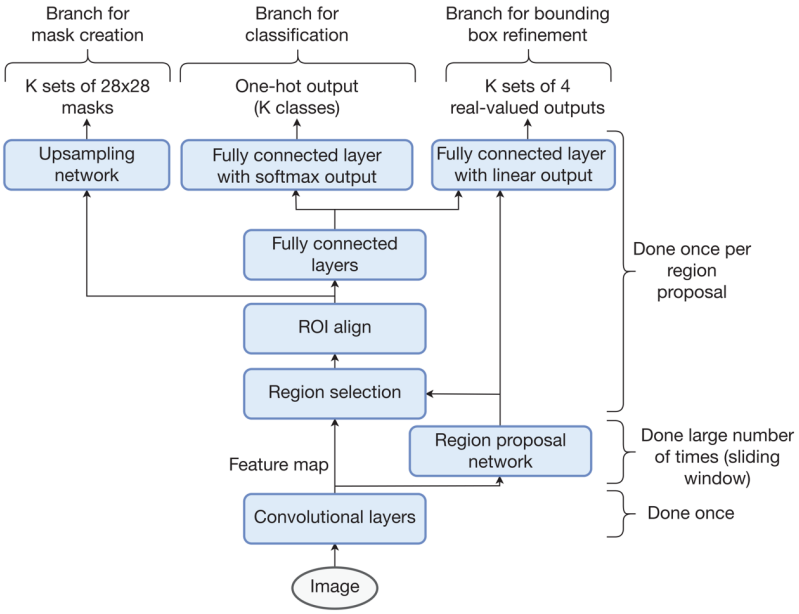


Figura 3.23: Visão geral da arquitetura da *Mask* R-CNN [126].

O processo é idêntico à *DeconvNet*, sendo que a máscara de saída remete para segmentação semântica, em cada região proposta, com indicação da classe a que objeto pertence. A segmentação fornece então um canal para cada classe, com a identificação concreta dos pixeis referentes. Esta informação deve ser combinada com a saída da ramificação de classificação para selecionar o canal de interesse [126]. Complementarmente, existe a informação obtida na saída da primeira ramificação indicativa dos parâmetros que definem a *bounding box* em torno do objeto.

Não obstante, os autores implementaram uma ferramenta que viria a substituir o *RoI max pooling layer*, nomeadamente o *RoI align layer* e preservação das informações do *feature map* [67]. Esta operação aplica interpolação entre os valores de um

determinado *kernel* (em vez de definir o máximo) o que origina um ajuste cuidado da máscara em zonas intermédias (destaque dos pixeis entre os limites das células).

Contudo, durante a inferência, será necessário o *upscaling* da máscara de saída em situações que se verifique um grande discrepância à dimensão da *bounding box* uma vez que a saída da ramificação adicionada é limitada a 28×28 pixeis.

A sua aplicabilidade é corrente em HPS, pela que a combinação da informação extraída da imagem (pose do esqueleto e instâncias de humanos) pode fortalecer a eficácia do método ao lidar com oclusões visuais drásticas [105]. Além disso, pode, de certa forma, durante a fase de treino, atuar como *ground-truth* e reduzir a diferença entre uma *predicted mask* (projeção da malha humana) e a *instance mask* para avaliação de profundidade dos vários indivíduos [9].

3.4 *Generative Adversarial Networks*

As ***Generative Adversarial Networks*** **(GANs)** tiveram impacto na melhoria da capacidade de aprendizagem das informações estruturas do corpo [64], bem como ajuste dos *heatmaps* para previsão de poses humanas biologicamente plausíveis [136].

Contrariamente à *pipeline* das CNNs, as GANs, introduzidas por Goodfellow *et al.* [63], apresentam uma abordagem diferente. O modelo transforma um conjunto de variáveis latentes $\boldsymbol{z}$ (caracterizado numa distribuição do *generator* p_g), pela adição de um ruído de entrada $p_{\boldsymbol{z}}(\boldsymbol{z})$, em amostras $\boldsymbol{x}$ (ou distribuições da mesma) recorrendo a uma função diferenciável $x = G(\boldsymbol{z}; \theta_g)$ (falsas amostras) representada, tipicamente pela rede neuronal (MLP) e pelos respetivos parâmetros θ_g do modelo. Uma segunda rede neuronal (MLP) $D(\boldsymbol{x}; \theta_d)$ estabelece, no final, um escalar $D(\boldsymbol{x}) \in [0, 1]$ indicativo da probabilidade de $\boldsymbol{x}$ ser um exemplo do conjunto de treino em vez de uma amostra falsa extraída do modelo (p_g). O treino sobre D é focado em maximizar a probabilidade da correta classificação aos exemplos de treino e amostras do G. Simultaneamente, o treino do G é motivado pela maximização da probabilidade de D cometer um erro.

Noutras palavras, a GAN, compostas, tipicamente, por dois módulos interligados numa estrutura análoga ao *minimax game* verifica, a cada iteração, um "vencedor" e um "perdedor" através de um função $V(G, D)$ [63]:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} [\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})} [\log(1 - D(G(\boldsymbol{z})))] \quad (3.24)$$

Como tal, o primeiro termo da Equação 3.24 remete para a instâncias provenientes da distribuição dos dados de treino enquanto o segundo remete para as instâncias obtidas da *noisy distribution* que caracteriza o *generator* [63]. Por norma, o primeiro termo desta equação não tem influência direta nas perdas do *generator*, pelo que se

pretende minimizar $V(G)$, *i.e.*, incentivando à tomada de decisões erráticas do *discriminator* com a probabilidade obtida $D(G(\boldsymbol{z})) \rightarrow 1$, isto é:

$$\min_G V(G) = \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})} \left[\log(1 - D(G(\boldsymbol{z})))\right].$$

Do ponto de vista do *discriminator* pretende-se maximizar $V(D)$, isto é, o primeiro termo $[\log D(\boldsymbol{x})] \rightarrow 1$ enquanto o segundo termo $\log(1 - D(G(\boldsymbol{z}))) \rightarrow 0$.

Posto isto, a restante secção assume como exemplo a rede proposta por Chen *et al.* [64], Figura 2.9, com aplicabilidade prática de um *generator* G (com uma estrutura de rede G enquadrada numa *fully convolutional network* com *residual blocks* e uma arquitetura *convolution-deconvolution*) e dois *discriminators*: *pose discriminator* P e *confidence discriminator* C.

Concretamente, a *generative network*, similar à SHG [48] possibilita processamento *bottom-up* e *top-down*. Projetada para ser uma *multi-task network*, gera, na sua saída, 32 *heatmaps* para imagem RGB (metade usados para estimar os *keypoints* e outra metade para a corresponde às oclusões) com uma confiança entre $[0, 1]$. A falta dos *discriminators* exige a otimização do *generator* exclusivamente com base nas *forward* e *backward propagations* dele mesmo (linhas marcadas a 1 na Figura 2.9). Naturalmente, são esperados resultados com baixa confiança e até incorretos.

Com a adição dos *discriminators* o cenário muda. A atualização do G por um *adversarial training* com C e P (linhas 2 e 3, respetivamente) tende à criação de poses mais confiantes e cinematicamente plausíveis.

3.4.1 *Multi-Task Generative Network*

No cenário em causa, o *generator* deverá produzir *heatmaps* com uma veracidade tal que será capaz de induzir em erro os *discriminators*. Ainda assim, a capacidade de identificar a parte do corpo oculta providencia informação relevante durante a inferência da pose humana. Por este motivo, uma *multi-task generative network* deve ser capaz de aprender uma determinada função G que procura estimar, a cada imagem $\boldsymbol{x}^i$, os *heatmaps* da pose $\boldsymbol{y}$ e da oclusão $\boldsymbol{z}$ (reunidos num *dataset* de treino $\left\{\boldsymbol{x}^i, \boldsymbol{y}^i, \boldsymbol{z}^i\right\}_{i=1}^{M}$ com M imagens), *i.e.*, $G(\boldsymbol{x}) = \{\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}\}$, sendo $\hat{\boldsymbol{y}}$ e $\hat{\boldsymbol{z}}$ os respetivos *heatmaps* previstos.

Não obstante, áreas maiores permitem a localização partes do corpo enquanto evidencias locais são essenciais para identificar *features* para a cara e mãos. Para este objetivo, uma arquitetura *encoder-decoder* com *skip connections* entre as camadas espelhadas permite a perceção da informação a diferentes escalas [64]. Similar à SHG [48], a *multi-task generator network* é composta por N módulos expressos por:

$$\begin{cases} \{\boldsymbol{Y}_n, \boldsymbol{Z}_n, \boldsymbol{X}\} = G_n\left(\boldsymbol{Y}_{n-1}, \boldsymbol{Z}_{n-1}, \boldsymbol{X}\right) & \text{se } n \geqslant 2 \\ \{\boldsymbol{Y}_n, \boldsymbol{Z}_n, \boldsymbol{X}\} = G_n(\boldsymbol{X}) & \text{se } n = 1 \end{cases}$$

sendo $\boldsymbol{Y}_n$ e $\boldsymbol{Z}_n$ os *tensors* de saída do n-ésimo módulo para estimação da pose e partes ocultas, respetivamente e $\boldsymbol{X}$ o *tensor* que circulará ao longo da rede, proveniente da imagem original. Portanto, esta *multi-task generator network* pode ser formulada numa sequência:

$$\{\boldsymbol{Y}_N, \boldsymbol{Z}_N, \boldsymbol{X}\} = G_N\left(G_{N-1}\left(\cdots\left(G_1(\boldsymbol{X}), \boldsymbol{Y}_1, \boldsymbol{Z}_1\right)\right)\right).$$

A cada módulo, os *heatmaps* finais $\hat{\boldsymbol{y}}$ e $\hat{\boldsymbol{z}}$ são obtidos de $\boldsymbol{Y}_n$ e $\boldsymbol{Z}_n$ por uma camada de convolução 1×1 com *stride* unitário e sem *padding*. Deste modo, para o *dataset* de treino definido, a *loss function* deste *generator* pode ser expressa por:

$$\mathcal{L}_G(\theta_g) = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \left((\boldsymbol{y}_n^m - \hat{\boldsymbol{y}}_n^m)^2 + (\boldsymbol{z}_n^m - \hat{\boldsymbol{z}}_n^m)^2 \right) \tag{3.25}$$

3.4.2 *Discriminator Network*

Pose discriminator

A fim de estimular um treino eficaz e uma previsão correta das configurações das articulações do corpo humano, os autores projetaram o *pose discriminator* P [64]. Inspirado no objetivo geral de um *discriminator*, este P deve ser capaz destingir as poses falsas (poses que não respeitam as restrições das articulações do corpo humano) das poses verdadeiras.

Análogo ao *generator*, esta rede assume uma arquitetura *encoder-decoder*, com *skip connections* entre camadas paralelas, para lidar com a informação local (baixo nível) e global (alto nível). Neste caso, o *pose discriminator* é alimentado com imagem RGB original e com os mapas da pose e oclusão gerados pelo *generator*. Como tal, a *loss function* para esta rede pode ser expressa numa variante da Equação 3.24, como:

$$\mathcal{L}_P\left(G, P\right) = \mathbb{E}[\log P\left(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{x}\right)] + \mathbb{E}\left[\log\left(1 - \left(P\left(G\left(\boldsymbol{x}\right), \boldsymbol{x}\right) - \boldsymbol{p}_{fake}\right)\right)\right] \tag{3.26}$$

onde $\boldsymbol{p}_{\text{fake}}$, abordado na Subsecção 3.4.3, remete para a *ground truth pose discriminator label* determinada sobre um *threshold* $\boldsymbol{\delta}$.

Confidence discriminator

Contrariamente a outras soluções, esta arquitetura possui um *discriminator* adicional. Os autores verificaram que a existência de oclusões alterava a natureza da distribuição Gaussiana no *heatmap* previsto, pelo que o ponto central não remeteria mais para a articulação local [64]. Neste sentido torna-se relevante distinguir as

previsões de baixa confiança das previsões de elevada confiança (através de um *threshold* $\boldsymbol{\varepsilon}$, descrito na Subsecção 3.4.3) a fim de obter uma relação geométrica entre as articulações não ocultas. Para esta rede, a *loss function* pode ser expressa como:

$$\mathcal{L}_C(G, C) = \mathbb{E}[\log C(\boldsymbol{y}, \boldsymbol{z})] + \mathbb{E}[\log(1 - (C(G(\boldsymbol{x})) - \boldsymbol{c}_{fake}))] \tag{3.27}$$

onde $\boldsymbol{c}_{\text{fake}}$, abordado na Subsecção 3.4.3, remete para a *ground truth confidence label* determinada sobre o *threshold* $\boldsymbol{\varepsilon}$ referido.

3.4.3 Treino de *Adversarial Networks*

Uma análise superficial ao treino de uma *adversarial networks* revela-se pertinente para conclusão devida desta secção. O Algoritmo 1 demonstra todo o processo de treino em pseudocódigo para o exemplo em causa [64].

Algorithm 1 Processo de treino de uma *adversarial networks* para HPE

Require: Imagens de treino: $\boldsymbol{x}$ e os respetivos *ground-truth heatmaps* $\boldsymbol{y}, \boldsymbol{z}$;

1: **while** Precisão do *dataset* de validação estiver a aumentar **do**
2: *Forward* G através de $\{\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}\} = G(\boldsymbol{x})$ e otimização de G de acordo com a Equação 3.25;
3: *Forward* P através de $\{\hat{\boldsymbol{p}}_{fake}\} = P(\boldsymbol{x}, G(\boldsymbol{x}))$ e otimizar P maximizando o segundo termo da Equação 3.26;
4: *Forward* P através de $\{\hat{\boldsymbol{p}}_{real}\} = P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ e otimizar P maximizando o primeiro termo da Equação 3.26;
5: *Forward* C através de $\{\hat{\boldsymbol{c}}_{fake}\} = C(G(\boldsymbol{x}))$ e otimizar C maximizando o segundo termo da Equação 3.27;
6: *Forward* C através de $\{\hat{\boldsymbol{c}}_{real}\} = C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ e otimizar C maximizando o primeiro termo da Equação 3.27;
7: Otimizar G pela Equação 3.28;
8: **end while**
9: **return** G

É sabido que *heatmaps* de baixa confiança, obtidos pelo *generator*, serão classificados como um resultado falso. Por isso, à medida que G é otimizado, C tende a acreditar que os falsos *heatmaps* são reais, originando previsões mais confiantes. Na verdade, o vetor de confiança $\boldsymbol{c} \in [0, 1]$ (16×1) é representativo de qual das partes a rede está efetivamente mais confiante. $\boldsymbol{c}_{real}$ é obtido diretamente do *dataset* de treino enquanto $\boldsymbol{c}_{fake}$ é obtido da análise aos *heatmaps* provenientes do G para a i-ésima parte do corpo, definido da seguinte forma:

$$\boldsymbol{c}^i_{fake} = \begin{cases} 1 & \text{se } \|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i\| < \varepsilon \\ 0 & \text{se } \|\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i\| \geqslant \varepsilon \end{cases}$$

Posto isto, $\boldsymbol{p}_{real}$ (vetor 16×1) é obtido de forma análoga enquanto $\boldsymbol{p}_{fake}$ é definido segundo uma distancia normalizada d_i entre a prevista e a *ground-truth* da i-ésima parte do corpo, segundo:

$$\boldsymbol{p}^i_{fake} = \begin{cases} 1 & \text{se } d_i < \delta \\ 0 & \text{se } d_i \geqslant \delta \end{cases}$$

Para o exemplo em causa, a *loss function* final pode ser definida através da Equação 3.28, sendo as perdas do *discriminator* limitadas pelos hiperparâmetros $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$ compreendidos entre $[0, \infty[$:

$$\min_{G} \max_{P,C} \mathcal{L}_G(\theta_g) + \alpha \mathcal{L}_C(G, C) + \beta \mathcal{L}_P(G, P) \tag{3.28}$$

3.5 *Transformer Neural Networks*

Os ***Transformers*** [106], originalmente introduzidos para *Natural Language Processing* (NLP), *e.g.*, interação homem-máquina com identificação e classificação do enredo para respostas coerentes (traduções automáticas, resumos de texto e análise emocional), vira-se aplicáveis, fruto de uma arquitetura modelável e vocacionada no processamento de longas sequências de dados, a tarefas tão distintas como prova de teoremas matemáticos e classificação de imagens.

A variante apresentada Dosovitskiy *et al.* [84], ViT, veio reformular as abordagem convencionais em visão computacional, evidenciando a independência dos *Transformers* numa aplicação direta. Efetivamente, a combinação de uma arquitetura baseada numa CNN com o mecanismo de *self-attention*, providenciado pelos *Transformers*, resultou numa melhoria significativa do desempenho nos modelos *deep learning*, perceção do contexto e acompanhamento temporal. Contudo, segundo autor, a criação de padrões de atenção especializados verificados nestes cenários, apesar de eficientes, podem comprometer a eficiência computacional dada a lacuna no dimensionamento para *hardware* moderno [84].

Motivado por esta evolução, diferentes estruturas baseadas em *Transformers* para a tarefa de HPE surgiram. Muita destas adotam uma CNN como *backbone* e, posteriormente, usam *Transformers* para fortalecer e modelar relações entre *keypoints* do corpo humano [71, 70]. Assim como o *Pose Regression TRansformers* (PRTR) [137] com incorporação de *transformer encoders* e *decoders* para, gradualmente ajustar a localização das articulações. Apesar disso, é evidente a dependência de CNNs para extração de *feature*. Como tal, soluções alternativas aplicaram o ViT como *backbone* para extrair *feature maps* numa dada instância (*i.e.*, pessoa) [85, 93], evoluindo ainda *graph Transformers*, *e.g.*, GTRS [124], permitindo uma melhoria estrutural da localização 3D das articulações com base na informação cinemática do humano.

Esta secção pretende-se elevar o conhecimento associada à arquitetura representativa dos *Transformers* e às várias ferramentas que a complementam focando, essencialmente, no objetivo deste trabalho.

3.5.1 Estrutura Base dos *Transformers*

Qualquer sistema de NLP e *speech recognition* precisa de lidar com grandes sequências de dados e transforma-las em saídas específicas. As abordagens tradicionais, *e.g.*, RNNs, têm várias limitações o que as impede de ser adotadas em aplicações do mundo real.

Antes da publicação de Vaswani [106], os mecanismos de atenção combinados com RNNs predominavam em tarefas de NLP. Os *Transformers* tornaram-se essenciais, cobrindo estas limitações e atingindo resultados vigorosos. Tudo possível através de uma arquitetura dotada de camadas *self-attention* (*i.e.*, *encoder/decoder attention* e *multi-head attention*) combinadas com *feed-forward networks* e *residual connections*.

RNNs e problemas associados

As aplicações, direcionadas a NLP, com arquiteturas *sequence-to-sequence*, *e.g.*, RNNs baseadas em *encoder-decoder*, destacaram-se com resultados aparentemente promissores. Vocalizando o problema ao processamento textual, uma frase de entrada poderá ser convertida em palavras e estas palavras, mapeadas em *feature vectors*, serão as entradas de um *encoder* [138]. Dado um $\boldsymbol{x}_t$ *token* de uma sequência de entrada representada por $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$, posteriormente mapeados em $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$ vetores e, num instante $\boldsymbol{t}$ com o *hidden state* (conhecido por *thought vector*) anterior $\mathbf{h}_{t-1}$ para uma entrada $\mathbf{x}_t$, é gerado um novo *hidden state* dado por:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) \tag{3.29}$$

Deste modo, o estado $\mathbf{h}_t$ (variável de contexto com informações relevantes para o mecanismo de atenção) codifica toda a informação na sequência de entrada e, no caso mais simples, mapeia (função m) a variável de contexto para o último *hidden state*:

$$c = m(\mathbf{h}_1, \ldots, \mathbf{h}_T) = \mathbf{h}_T \tag{3.30}$$

Eventualmente, com aumento de complexidade, numa RNN bidirecional o *hidden state* não dependerá apenas do estado anterior $\mathbf{h}_{t-1}$ e da entrada $\mathbf{x}_t$, mas também do próximo $\mathbf{h}_{t+1}$ [138].

O *decoder* assume a saída do *encoder*, a variável de contexto $\mathbf{c}$ e uma dada sequência de saída $\boldsymbol{y}_1, \ldots, \boldsymbol{y}'_T$ para gerar as respetivas saídas [138]. E como tal, o

hidden decoder state, no instante de tempo t', é agora função, não apenas do *hidden state* passado $\mathbf{s}_{t-1}$ e saída do *decoder* $\mathbf{y}_{t'-1}$, mas também do vetor de contexto $\mathbf{c}_t$:

$$\mathbf{s}_{t'} = g\left(\mathbf{s}_{t-1}, \mathbf{y}_{t'-1}, \mathbf{c}\right) \tag{3.31}$$

Ficam evidentes potenciais problemas de inicialização, sendo que algumas abordagem tendem a inicializar o *decoder* recorrendo ao último *hidden state* do *encoder* (variável de contexto).

Não obstante, os LSTMs e GRUs surgem como solução ao *vanishing gradient problem* existente nas RNNs através de mecanismos internos (*gates*) para regular o fluxo de informação ao longo da rede [138]. São estes mecanismos que possibilitam à rede compreender que informação é relevante à chegada de cada *token*.

Apesar de tornar o modelo robusto a *vanishing* e *exploding gradients*, o processamento sequencial dos *tokens* e manutenção do contexto/informação em memória complicava-se à medida que o tamanho da entrada aumenta [138]. A impossibilidade processamento paralelo dificultou o escalabilidade e eficiência computacional, *e.g.*, cada passagem pela rede é condicionada pelo facto de o modelo ter visto amostras anteriores da sequência.

Mecanismo de atenção generalizado

Contudo, tal como nas simples RNNs, o tempo de treino é elevado e não recorre da toda a capacidade computacional paralela de uma GPU. Além disso, o *decoder* deve ser capaz de gerar uma sequência de saída com base num *thought vector* (codificado com toda a informação da sequência de entrada). As RNNs tradicionais, nestes *hidden states*, levavam em atenção a informação dos últimos elementos da sequência. Esta desvantagem seria atenuada com os LSTMs, mas não eliminada. Aparentemente, "*Attention Is All You Need*" [106] procura preservar o conteúdo importante e fundamentar uma resposta contextualizada "independente" da quantidade de informação na entrada.

O **mecanismo de atenção** permite que o modelo atribua importância diferenciada a partes específicas dos dados de entrada (*e.g.*, vetores de palavras ou *features* extraídas), com base numa ***query*** para um processamento ponderado. Na verdade, tal como presente na Figura 3.24, este mecanismo mapeia uma *query* $\mathbf{q}$ e um conjunto de pares *key-values* numa saída. Sendo todos estes elementos vetores, a saída será calculada como uma soma ponderada dos *values* com uma influência definida por um *attention weights*. Estes pesos serão tanto maiores quando maior a compatibilidade entre a *query* e a *key* correspondente ao *value* em causa [138].

Formalizando, considere-se uma unidade de memória composta por n pares de *key-values* $(\mathbf{k}_1, \mathbf{v}_1), \ldots, (\mathbf{k}_n, \mathbf{v}_n)$ com $\mathbf{k}_i \in \mathbb{R}^{d_k}$ e $\mathbf{v}_i \in \mathbb{R}^{d_v}$, onde as *keys* são vetores e os *hidden states*, anteriormente referidos, são os *values*. A camada de atenção recebe

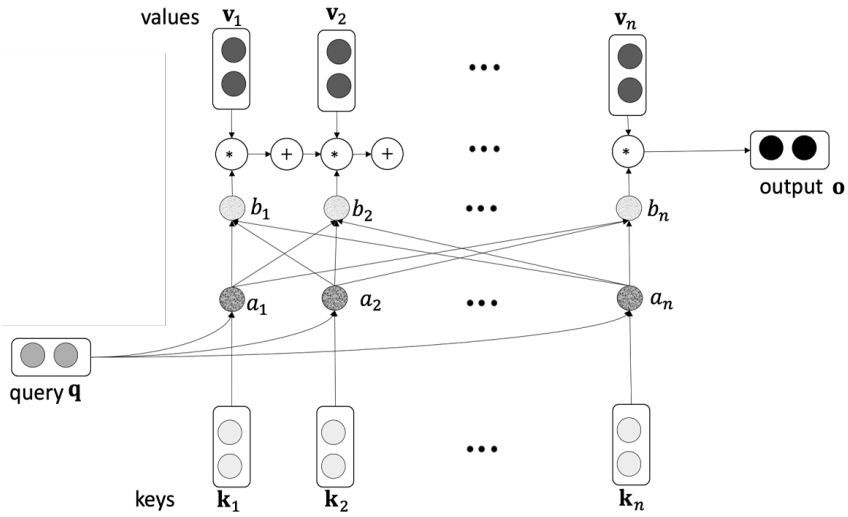


Figura 3.24: Mecanismo de atenção: *query*, *keys*, *values* e vetor de saída [138].

uma *query* $\mathbf{q} \in \mathbb{R}^{d_q}$ e retorna um vetor e saída $\mathbf{o} \in \mathbb{R}^{d_v}$ (mesmo tamanho do vetor $\mathbf{v}$) [138].

As *keys* $\mathbf{k}_1, \ldots, \mathbf{k}_n$ relacionam a similaridade entre estes *values* e a *query* $\mathbf{q}$ através de uma função diferenciável α, Equação 3.32 (treinada em *backpropagation* com o resto do sistema) que retorna uma pontuação $a_1, \ldots, a_n$. Isto é, quanto maior a pontuação obtida, maior a similaridade entre a *query* em causa e o *value* associado.

$$a_i = \alpha(\mathbf{q}, \mathbf{k}) \tag{3.32}$$

Eventualmente, a soma de todos os *attention weights* ($\mathbf{b} = [b_1, \ldots, b_n]^{\mathrm{T}}$) deverá ser unitária [138], como tal, podemos determinar cada elemento de $\mathbf{b}$ com recurso base numa função *softmax* a partir das pontuações geradas:

$$b_{\mathbf{q},\mathbf{k}_i} = \frac{e^{a_{\mathbf{q},\mathbf{k}_i}}}{\sum_{j=1} e^{a_{\mathbf{q},\mathbf{k}_j}}} \tag{3.33}$$

Cada iteração (*decoder step*), gera um único vetor de saída (vetor de contexto) dado pela soma ponderada dos *attention weights* e *values* (*encoder hidden states*):

$$\mathit{Attention}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \mathbf{o}_t = \sum_{i=1} b_{\mathbf{q},\mathbf{k}_i} \mathbf{v}_{\mathbf{k}_i} = \sum_{i=1} \frac{e^{a_{\mathbf{q},\mathbf{k}_i}}}{\sum_{j=1} e^{a_{\mathbf{q},\mathbf{k}_j}}} \mathbf{v}_{\mathbf{k}_i} \tag{3.34}$$

Deste modo, o *encoder* tem acesso a todos os estados da sequência de entrada a cada iteração t, o que remove a necessidade de codificar toda a informação da sequência original num vetor de tamanho fixo, como seria realizado no modelo *sequence-to-sequence* dito regular [138].

Por sua vez, a função de pontuação $\alpha(\mathbf{q}, \mathbf{k})$ pode assumir diferentes formas, cada um com vantagens em específico. Concretamente, o produto escalar apresenta-se como o mais simples e sem qualquer parâmetro passível de treinar:

$$\alpha(\mathbf{q}, \mathbf{k}) = \mathbf{q} \cdot \mathbf{k}^{\mathrm{T}} \tag{3.35}$$

Uma variante divide o produto escalar pela raiz da dimensão do vetor *query* $\sqrt{d_k}$. Esta normalização, segundo os autores, evita gradientes pequenos verificados em dimensões superiores [106].

$$\alpha(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q} \cdot \mathbf{k}^{\mathrm{T}}}{\sqrt{d_k}} \tag{3.36}$$

Ainda assim, implementações alternativas optaram por projetar a *query* e as *keys* para uma camada intermédia com dimensão h e pesos passíveis de serem treinados $(\mathbf{W}_k, \mathbf{W}_q)$ usando, posteriormente, uma função *sigmoid* para os combinar com os *values* segundo [138]:

$$\alpha(\mathbf{q}, \mathbf{k}) = \mathbf{v} \tanh\left(\mathbf{W}_k \mathbf{k}^{\mathrm{T}} + \mathbf{W}_q \mathbf{q}\right) \tag{3.37}$$

Transformer model

Assim como apresentado na Figura 3.25, a arquitetura dos *Transformers* é baseado num *encoder-decoder* combinando as vantagens das CNNs em computação paralela e as RNNs com a perceção de contexto em grandes sequências de dados.

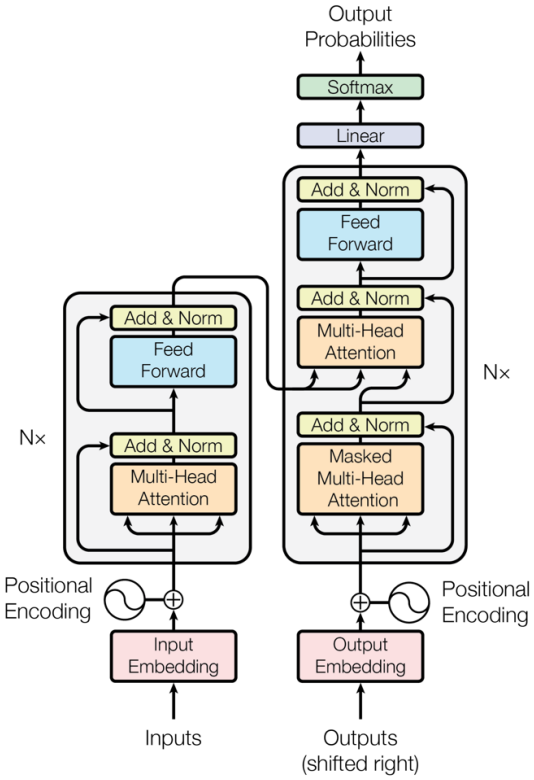


Figura 3.25: *Transformer architecture* [106].

O *encoder* (*pipeline* da esquerda na Figura 3.25) mapeia uma sequência de entrada $(x_1, \ldots, x_n)$ para uma sequência de representações contínuas $\mathbf{z} = (z_1, \ldots, z_n)$. Composto por $N = 6$ camadas idênticas, divididas cada uma em duas subcamadas. A primeira, remete para o mecanismo de *self-attention*, enquanto a segunda representa uma *FC feed-forward network*. Análogo às ResNets, foi implementado *residual connections* em torno das subcamadas, para uma soma devida dos elementos constituintes, seguido de uma camada de normalização, *i.e.*, a saída de cada subcamada é $LayerNorm(x + SubLayer(x))$, sendo $SubLayer(x)$ a função implementada pela subcamada em causa [106].

O *decoder* (*pipeline* da direita na Figura 3.25), também composto por $N = 6$ camadas idênticas, gera uma sequência $(y_1, \ldots, y_m)$ a partir da sequência de saída do *encoder*. É composto por duas subcamadas superiores, análogas às presentes no *encoder*, responsáveis pela *multi-head attention* sobre as saídas do *encoder*. Complementarmente, uma terceira subcamada, concede-lhe a propriedade *auto-regressive*, tendo sido modificada para realizar *self-attention* unicamente aos elementos passados, eventualmente conhecidos pela rede, evitando influenciar a saída com resultados conhecidos durante a fase de treino. Não obstante, é visível as *residual connections* em torno de cada subcamada assim como as camadas de normalização [106].

Tal como referido, as *source* e as *target words* são convertidas em *tokens* que serão passados por uma camada ***word embedding*** e *positional encoding*. É nesta primeira camada, *word embedding*, que decorre a conversão das palavras/*tokens* numa representação vetorial (numérica). Concretamente, é gerada uma matriz $\mathbf{W} \in \mathbb{R}^{n \times d}$ com dimensões (n, d) referentes ao número de elementos n da frase de entrada (*sentence*) num *embedding vector* de dimensão d. Este processo é conseguido com base numa *embedding function*, treinada para perceção de informação semântica entre as palavras. Como tal, cada palavra corresponde a um ponto num *embedding space*, sendo que pontos próximos evidenciam similaridade entre palavras [138].

O processamento sequencial das palavras, crucial nos mecanismos tradicionais de NLP, veio a ser substituído por computação paralela nos *Transformers*, pela incorporação de *multi-head attention layers*. Contudo, será necessário a adoção de método de indexação, conhecido por ***positional encoding***, para adição da informação referente à posição relativa ou absoluta da palavra na frase [138]. Existem diversas hipóteses de *positional encodings*, tendo os autores definido a utilização da função cos() e sin() com diferentes frequências para preencher o vetor posicional. Sendo $\mathbf{P} \in \mathbb{R}^{n \times d}$ a matriz, com dimensões iguais a $\mathbf{W}$, que agrega os *positional encodings* de todas os n *tokens* e i e j os indexes ao longo de n e d respetivamente, temos um ajuste posicional dado por:

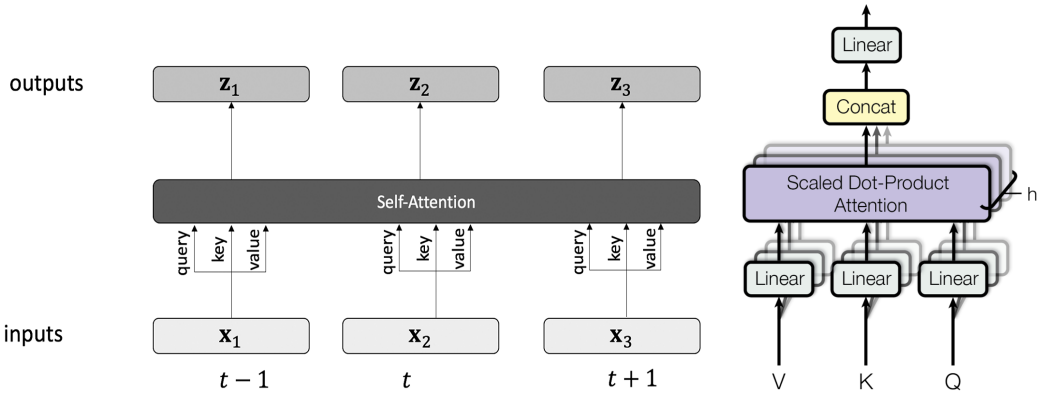
$$\begin{aligned} \mathbf{P}_{i,2j} &= \sin\left(i/1000^{2j/d}\right) \\ \mathbf{P}_{i,2j+1} &= \cos\left(i/1000^{2j/d}\right) \end{aligned} \tag{3.38}$$

Este processo evita compensações exagerados que afastem indevidamente os *tokens*. Tal como visível na Figura 3.25, as duas matrizes, *i.e.*, *word embeddings* $\mathbf{W}$ e *positional encoding* $\mathbf{P}$, são somadas para gerar um *tensor* de entrada $\mathbf{X} = \mathbf{W} + \mathbf{P} \in \mathbb{R}^{n \times d}$ [106].

Camadas de atenção

Tanto no *encoder* como no *decoder*, *self-attention* é, apesar de algumas variações, assumido como o bloco de construção base.

Cada bloco de *self-attention*, Figura 3.26a, converte três vetores $(\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i)$ gerados a partir do vetor de entrada $\mathbf{x}_i$ num vetor de saída $\mathbf{z}_i$. Concretamente, estes três vetores são obtidos pela projeção do vetor $\mathbf{x}_i$, no instante i, nos pesos das matrizes $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$ que, por sua vez, são determinados durante o treino [138].



(a) Mecanismo de *Self-Attention* [138]

(b) *Multi-Head Attention* [106]

Figura 3.26: Projeção linear das entradas para *query*, *keys* e *values* afim para análise através *multi-head attention* e concatenação das saídas.

O processo é análogo ao descrito numa das subsecções passadas, como tal:

1. O *query vector* do *token* i, $\mathbf{q}_i$, é combinado com todos os *key vectors* $\sum_{j=0}^{l} \mathbf{q}_i \mathbf{k}_j^{\mathrm{T}}$ para influenciar os pesos para a própria saída $\mathbf{z}_i$ gerada.

2. O *key vector* do *token* i, $\mathbf{k}_i$, deve ser combinado com todos os *query vectors* a fim de obter a similaridade com este último vetor e influenciar a saída com base na pontuação gerada pela Equação 3.35.

3. O *value vector* do *token* i, $\mathbf{v}_i$, combinado com os *attention weights*, dá origem ao vetor de saída $\mathbf{z}_i$.

Na prática, em vez de um cálculo vetorial para cada *token*, a atenção pode ser calculada pela matriz de entrada $\mathbf{X}$, combinada com cada uma das matrizes das várias *querys* $\mathbf{Q}$, *keys* $\mathbf{K}$ e *values* $\mathbf{V}$:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\mathbf{V} \tag{3.39}$$

Porém, os autores optaram por implementar ***multi-head attention*** com ***h*** *single self-attention heads* em paralelo, Figura 3.26b. Este processo permite que cada *head* aprenda algo diferente, uma vez que *multi-head attention* providência diferentes representações do subespaço (em vez de apenas uma representação) [138]. Porém, não é nada mais do que concatenação de cada matriz de saída $\mathbf{Z}_i$ com a multiplicação por uma matriz de pesos adicional $\mathbf{W}_O$ para obter uma única matriz de saída $\mathbf{Z}$ [106]:

$$\begin{aligned} MultiHead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= Concat\left(head_1, \ldots, head_h\right)\mathbf{W}_O \\ \text{onde } head_i &= Attention(\mathbf{Q}\mathbf{W}_q^i, \mathbf{K}\mathbf{W}_k^i, \mathbf{V}\mathbf{W}_v^i) \end{aligned} \tag{3.40}$$

O objetivo será melhorar o *decoder* para que ele aprenda com a sequência do *encoder* e resultados passados do próprio *decoder*. Semelhante à arquitetura *sequence-to-sequence*, apenas o último *token* precisa de estar presente, sendo que os futuros deverão ser escondidos (*masked*) [138]. Na verdade, não seria um treino viável se o *decoder* conhecesse o resultado a obter na saída. Esta alteração resulta num ***masked multi-head attention***, sendo implementada com recurso a uma *masking weight matrix* $\mathbf{M}$ que atribui $-\infty$ a *tokens* futuros e 0 aos *tokens* passados. Analisando a função de atenção respetiva (Equação 3.41), *tokens* passados resultam no valor escalar previsto, enquanto *tokens* futuros resultam no valor 0, dada a natureza da função *softmax* [106].

$$maskedAttention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\mathbf{V} \tag{3.41}$$

Nesta fase, o resultado destes *attention vectors* e dos vetores, provenientes do bloco do *encoder*, são sujeitos a uma nova camada de *multi-head attention*. É deste modo que o *decoder* pode aprender a relação de atenção entre todas as entradas e a saída esperada.

Feed-forward networks

Tanto o *encoder* como o *decoder* contêm uma *FC feed-forward network* após as subcamadas de atenção. Esta camada é aplicada a todos os *attention vectors* de forma idêntica (com base nos mesmos parâmetros ($\mathbf{W}_1, \mathbf{W}_2, b_1$ e b_2), análogo à Equação 3.1), para os transformar num formato aceitável para o próximo *encoder* ou *decoder layer* [106]. Consiste em duas transformações lineares, com a função de ativação *ReLU* entre elas, aplicado a cada sequência $\mathbf{x}$ separadamente:

$$FFN(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2 \tag{3.42}$$

Os *Transformers* utilizam a técnica de regularização *dropout* na saída de cada subcamada antes da adição da entrada desta camada e posterior normalização.

Em resumo, a vantagem do mecanismo de *self-attention* é o imediato acesso a todos os elementos da sequência de entrada, em oposição aos modelos RNNs. Por sua vez, em termos de complexidade computacional, as camadas de *self-attention* são mais rápidas do que os *recurrent layers* quando o comprimento da sequência n é menor do que a dimensionalidade da representação d, verificado na maioria dos casos [106]. Contudo, o foco do projeto será a combinação dos *Transformers* com as técnicas de visão computacional, conhecido por *Vision Transformer* (ViT).

3.5.2 *Vision Transformers*

Os *Transformers*, inicialmente desenvolvidos para NLP, atingiram resultados impressionantes em diferentes tarefas de visão computacional (*e.g.*, classificação de imagem, deteção de objetos, segmentação e HPE) desde a exposição do modelo *Vision Transformer* (ViT) [84] em 2021.

Ainda que o mecanismo de atenção tenha sido combinado com CNNs, *e.g.*, *augmenting feature maps* para classificação de imagens, processamento adicional do resultado gerado numa CNN ou acompanhamento temporal, a proposta em causa [84], foi a primeira a demonstrar como os *Transformers* podem substituir "totalmente" as operações convolucionais com apenas algumas alterações.

Fica evidente o problema associado ao custo computacional na determinação da relação entre cada pixel na imagem. Tal como indicado no título do documento [84], uma imagem vê-se dividida numa sequência de *image patches*, num *flattened vector*, a fim de aplicação do mecanismo de atenção e posterior classificação. Contudo, nesta implementação concreta, a exigência de uma grande quantidade de dados (*e.g.*, *JFT-300M dataset*, propriedade da Google com 300 milhões de imagens) é imposta para que sejam atingidos resultados substancialmente superiores, durante o treino, em termos de recursos computacionais e performance em diversos *benchmarks* de classificação de imagem (*e.g.*, ImageNet).

O trabalho da comunidade é contínuo, impulsionando a modelação deste conceito com alterações estruturais categoricamente classificadas, segundo Khan *et al.* [83] em *uniform scale ViTs* (mantendo a escala na representação das *features* ao longo da rede), *multi-scale ViTs* (orientado a *dense prediction tasks* capaz de aprender através de *hierarchical features*) e *Hybrid ViTs* (com operações convolucionais). Estes modelos exigiriam todo um estudo dedicado, porém, face ao objetivos estabelecidos neste trabalho, esta subsecção dedicar-se-à unicamente à abordagem inicial proposta por Dosovitskiy *et al.* [84] dotada dos vários conceitos transversais ás restantes variantes.

ViT architecture overview

O modelo ViT, Figura 3.27, divide a imagem na entrada $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ numa sequência de *flattened 2D patches* $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 C)}$ (pequenas regiões quadradas) com tamanho fixo, sendo (H, W) a resolução da imagem original, C o número de canais, (P, P) a resolução de cada ***image patch*** e $N = HW/P^2$ o número resultante de *patches* gerados a partir da imagem $\mathbf{x}$.

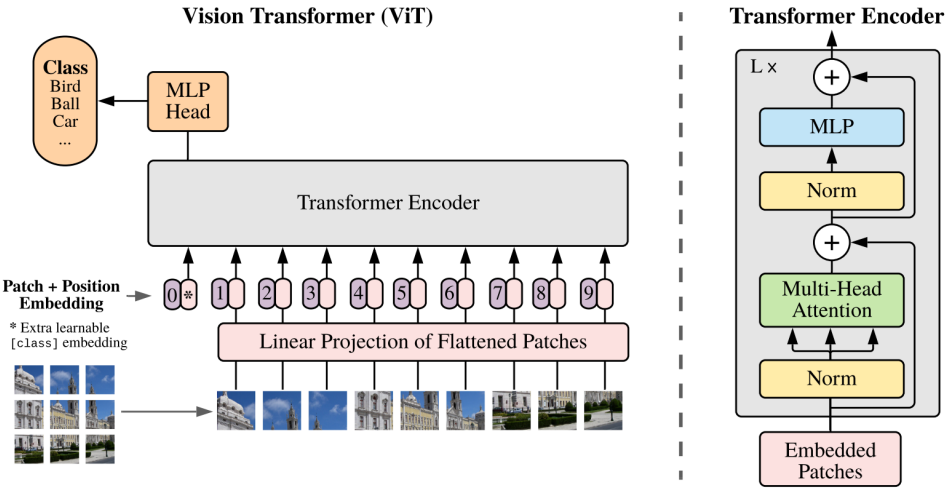


Figura 3.27: Arquitetura ViT [84].

Uma vez que os *Transformer*, em todas as suas camadas, operam com vetores de tamanho D, terá sido necessário converter a sequência de *image patches* em ***flattened patches*** e posteriormente, redimensiona-los, segundo uma projeção linear, para a dimensão D pretendida. Esta projeção linear, Equação 3.43, é treinável e dada pelo produto entre a matriz referente a cada *patch* e uma matriz de pesos $\mathbf{E}$ (***patch embedding projection***) composta por parâmetros ajustados durante o treino. As saídas destas projeções são conhecidas por ***patch embeddings***.

$$\mathbf{z}_0 = \left[\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \ldots; \mathbf{x}_p^N\mathbf{E}\right] + \mathbf{E}_{pos}, \quad \mathbf{E} \in \mathbb{R}^{\left(P^2C\right) \times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{3.43}$$

Os autores adotaram uma arquitetura composta por apenas um *encoder*, devidamente treinado pela anexação de um "*token* de classificação" (`[class]` *token*) $\mathbf{z}_0^0 = \mathbf{x}_{\text{class}}$, cujo estado de saída do *encoder* $(\mathbf{z}_L^0)$ atua como representação da imagem $\mathbf{y}$ (Equação 3.44). Este *token* adicional é responsável por extrair informações globais ou do contexto geral da imagem.

$$\mathbf{y} = LayerNorm\left(\mathbf{z}_L^0\right) \tag{3.44}$$

Tanto durante o pré-treino como durante o *fine-tuning*, a *classification head* é conectada diretamente ao `[class]` *token* $\mathbf{z}_L^0$. Tal como visível na arquitetura, a *classification head* é implementada com um MLP. Durante o pré-treino, este é composta por apenas um *hidden layer* responsável pela realização de transformações não lineares nas representações de entrada (para que o modelo aprenda representações mais complexas e discriminativas dos dados durante esta fase). Durante o *fine-tuning* este *hidden layer* é substituído por uma camada linear para submissão dos dados de entrada a transformações lineares simples.

A posição/localização de cada região na imagem (***position embedding***) é adicionada ao *patch embedding* respetivo para consideração da informação posicional. Este ***embedding vectors*** resultantes servem como entrada para o *encoder*.

De forma sucinta, o *transformer encoder*, apresentado na subsecção anterior, consiste em camadas alternadas de *multiheaded self-attention* (MSA) e blocos MLP (Equação 3.45 e 3.46 respetivamente). *LayerNorm* é aplicado antes de cada bloco e *residual connections* entre subcamadas.

$$\mathbf{z}'_\ell = \text{MSA}\left(LayerNorm\left(\mathbf{z}_{\ell-1}\right)\right) + \mathbf{z}_{\ell-1}, \quad \ell = 1 \dots L \tag{3.45}$$

$$\mathbf{z}_\ell = \text{MLP}\left(LayerNorm\left(\mathbf{z}'_\ell\right)\right) + \mathbf{z}'_\ell, \quad \ell = 1 \dots L \tag{3.46}$$

Como alternativa, a sequência de *image patches* na entrada pode ser formada por *feature maps* de CNNs. Nesse modelo híbrido, a matriz de projeção E, Equação 3.43, é aplicado às regiões obtidas do *feature map* da CNN para conversão em *flattened vectors* e posterior redimensionamento na dimensão D, exigida nos *Transformers*.

Pré-treino e *fine-tuning* dos ViT

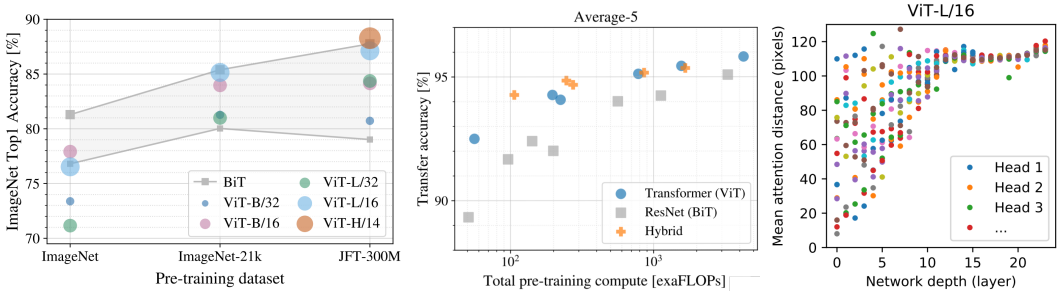
Quando pré-treinado em grandes quantidades de dados e transferido (*fine-tuning*) para tarefas especificas (*downstream tasks*), *e.g.*, reconhecimento de imagem com *datasets* menores, os ViT atingem excelentes resultados comparados com as SOTA CNNs [83].

Concretamente, o modelo terá sido pré-treinado com um o *JFT-300M dataset*, num total de 300 milhões de imagens com alterações estruturais durante a fase de *fine-tuning*. Ainda assim, os autores verificaram que o ViT apresenta um menor número de *inductive bias* comparativamente a uma CNN (ótimas para extração de *features* na imagem mas incapazes de modelar dependência entre elas). Isto resulta em menores suposições prévias sobre a estrutura dos dados e flexibilidade do modelo em aprender uma maior variedade de relações nos dados, porém, tende a aumentar a complexidade de treino [84].

Por sua vez, o processo de *fine-tuning* para tarefas específicas é otimizado quando verifica um ajuste de parâmetros através da utilização de imagens de resolução superior, comparativamente à fase de treino. Neste cenário, mantendo o *patch size*, resulta, naturalmente numa sequência maior de *image patchs*. Recordando, os ViT são capazes de lidar com diferentes comprimentos de sequências distribuídas arbitrariamente (*position embedding* é perdida após divisão em pequenas regiões), no entanto, a incorporação das *position embeddings* pré-treinadas podem não ser viáveis ao cenário em causa. Como tal, tornou-se viável a realização de uma interpolação 2D nos *position embeddings* pré-treinadas de acordo com a localização da imagem original. Efetivamente, estão definidos os únicos *inductive bias* adicionados manualmente no ViT: ajuste de resolução com interpolação 2D nos *position embeddings* pré-treinados e extração de *image patchs* [84].

Ainda assim, foram gerados três variantes do ViT: *Base* (ViT-B), *Large* (ViT-L) e *Huge* (ViT-H) com tempos de treino substancialmente inferiores e definidos em *TPUv3-core-days* (número de *TPU v3 cores* multiplicado pelos dias de treino) por ser o *hardware* utilizado para esse efeito. Os resultados foram obtidos com *fine-tuning* em quatro pequenos *datasets* para classificação de imagem (ImageNet, *CIFAR-10/100*, *Oxford-IIIT Pets* e *Oxford Flowers-102*), ultrapassando BiT-L (ResNet 152 × 4).

A Figura 3.28a sintetiza o referido. O pré-treino foi realizado, experimentalmente, em três *datasets* com dimensões crescentes. É visível que a ResNet (BiT CNNs) apresenta melhor performance em regimes com acesso a poucos dados (ImageNet), potencialmente pelos *inductive bias* definidos que são traduzidos *receptive fields* maiores à medida a imagem se aprofunda na rede.



(a) Resultados após *fine-tuning* no *ImageNet dataset*

(b) Desempenho *vs.* custo computacional para diferentes arquiteturas

(c) Área atendida por *head* e *network depth*

Figura 3.28: Comparação entre SOTA em *benchmarks* populares para classificação de imagem [84].

Complementarmente, a Figura 3.28b apresenta o custo computacional referido durante a fase de pré-treino em três arquiteturas diferentes. Aparentemente, ViT híbridos apresentam melhores resultados em modelos menores, mas tende a diminuir à medida que a complexidade do modelo aumenta.

Ainda assim, a "*attention distance*", similar ao *receptive field* locais das CNNs, é visivelmente superior na perceção do contexto, Figura 3.28c, ao longo das camadas que a compõem (*network depth*). Cada ponto remete para uma valor médio de *attention distance* entre a imagens, para cada uma de 16 *attention heads*, em cada camada (24 no total). Na verdade, a curva de variação do *receptive field* das CNNs com a *network depth* seria algo similar a uma rampa, com aumento linear, tangente ao limite inferior dos pontos do gráfico e de valor máximo atingível quando cada píxel atender a qualquer pixel na imagem. Fica então evidente atenção global na imagem verificada em algumas *heads*, ainda nas camadas iniciais.

Capítulo 4

Implementação Proposta

4.1 Contextualização da Solução

A presente secção deverá formalizar o problema em causa, devendo os conceitos anteriormente expostos atuar como complemento de uma implementação num sistema robusto e coerente, descrevendo a contribuição desenvolvida pelo autor.

Apesar da maior fiabilidade presente num sistema *multi-view*, ao lidar com oclusões parciais e na determinação da profundidade do humano no espaço, também será verdade que a sua incorporação se revelará especialmente complexa em sistemas variáveis, pela necessidade de relacionamento espacial entre câmaras (*e.g.*, matriz de rotação e vetor de translação numa relação entre sistemas de coordenadas), além do sincronismo na obtenção de imagens úteis.

Algumas questões veem-se simplificadas quando a responsabilidade de perceção do mundo é realizada com apenas uma câmara. Definidas as limitações que regem o sistema, será possível acondicionar uma solução viável. Ainda assim, as oclusões devem ser processadas e propriamente complementadas recorrendo a toda a perceção de informação captada num passado recente e, num pós processamento, da informação de um futuro próximo.

Ainda assim, a recuperação dos parâmetros descritivos do humano (*i.e.*, conjunto de articulações e malha humana), num dado momento e numa vertente global, vê-se complicada quando o conjunto de imagens monoculares, que alimentam o modelo, proveem de câmaras dinâmicas (*i.e.*, a posição da câmara, relativamente ao sistema de coordenadas do mundo, muda a cada *frame*).

É neste sentido que o projeto será orientado, adaptando diferentes *frameworks*, individualmente otimizadas, para o seu propósito. Este novo modelo, procura estender o processo realizado num convencional *pose estimator* para obtenção precisa das articulações dos vários humanos no sistema de coordenadas do mundo. Não obstante, pela incorporação de *coerência temporal* é potencializada a ponderação de evidências visuais, presentes no vídeo inicial, para otimização iterativa dos parâmetros da câmara e obtenção da trajetória global dos humanos em causa.

Conforme descrito na Figura 4.1, a solução proposta divide-se em três módulos. Numa primeira fase, ***Multi-Object Tracking and Re-Identication*** (Secção 4.2), para deteção de cada humano no cenário, expresso em sequências de *bounding boxes*. Esta informação é propriamente acondicionada para atuar como entrada do segundo módulo, ***3D Human Pose and Shape Estimation*** (Secção 4.3) e recuperar o movimento $\widehat{\boldsymbol{Q}}^i$ de cada pessoa (incluindo a translação) no sistema de coordenadas da câmara. Naturalmente, este movimento $\widetilde{\boldsymbol{Q}}^i$ tenderá a ser incompleto fruto de oclusões evidentes ou deteções erráticas por parte do MPT (*e.g.*, a pessoa sair do *Field of View* (FoV) da câmara ou falhas no *tracking*). Para tal, decorre uma integração do *Generative Motion Optimizer* (Secção 4.4) dividido nos três últimos estados propostos na implementação GLAMR [34] (exposto na Subsecção 2.3.2) para estimar a *body motion* sem oclusão $\widehat{\boldsymbol{\Theta}}^i$ com base pose do corpo temporalmente definida $\widetilde{\boldsymbol{\Theta}}^i$ e, iterativamente, otimizar a trajetória global de todos os humanos e os parâmetros da câmara a fim de produzir as *global motions* $\breve{\boldsymbol{Q}}^i$ consistente com as evidências do vídeo.

Concretamente, na entrada do modelo está presente um vídeo $\boldsymbol{I} = (\boldsymbol{I}_1, ..., \boldsymbol{I}_H)$ com H *frames* obtidos a partir de uma câmara dinâmica ou estática. Seguindo a nomenclatura proposta por Yuan *et al.* [34], à saída do sistema é obtido uma sequência de poses (*global motion*) $\{\boldsymbol{Q}^i\}_{i=1}^{L}$ de L pessoas no sistema de coordenadas do mundo. Generalizando, a ***global motion*** $\boldsymbol{Q}^i = (\boldsymbol{T}^i, \boldsymbol{R}^i, \boldsymbol{\Theta}^i, \boldsymbol{B}^i)$, para a pessoa i, é definida por uma ***root translations*** $\boldsymbol{T}^i = (\boldsymbol{\tau}_1^i, \ldots, \boldsymbol{\tau}_H^i)$, ***root rotations*** $\boldsymbol{R}^i = (\boldsymbol{\gamma}_1^i, \ldots, \boldsymbol{\gamma}_H^i)$, ***body motion*** (pose do corpo) $\boldsymbol{\Theta}^i = (\boldsymbol{\theta}_1^i, \ldots, \boldsymbol{\theta}_H^i)$ e ***shape*** $\boldsymbol{B}^i = (\boldsymbol{\beta}_1^i, \ldots, \boldsymbol{\beta}_H^i)$. Esta informação é inicialmente obtida do *HPS estimator* (Subsecção 4.3.2) que, na falha de deteções, procura, com base num modelo autorregressivo, preencher e definir o trajeto deste ao longo do vídeo (Secção 4.4).

Por convenção, serão consideradas 23 articulações humanas das 24 presentes no modelo SMPL considerado na *body pose* (Subsecção 4.3.1), assumindo que a primeira (#0), será, no modelo inicial a *root joint* (*i.e.*, *pelvis*). Em particular, cada pose $\boldsymbol{\theta}_h^i \in \mathbb{R}^{23\times3}$ e *shape* $\boldsymbol{\beta}_h^i \in \mathbb{R}^{10}$ do corpo humano correspondem aos parâmetros que definem o modelo estatístico SMPL [19] (Subsecção 4.3.1). Determinados a *root translation* $\boldsymbol{\tau}_h^i \in \mathbb{R}^3$ e a (*axis-angle*) *root rotations* $\boldsymbol{\gamma}_h^i \in \mathbb{R}^3$, o modelo corporal SMPL [19] permite

obter a representação da malha humana através de uma função linear $\mathcal{S}(\boldsymbol{\theta}_h^i, \boldsymbol{\beta}_h^i)$[1] para mapear uma *global pose* $\boldsymbol{q}_h^i = (\boldsymbol{\tau}_h^i, \boldsymbol{\gamma}_h^i, \boldsymbol{\theta}_h^i, \boldsymbol{\beta}_h^i)$ numa ***articulated triangle mesh*** $\mathbf{M}_h^i \in \mathbb{R}^{N\times 3}$, sendo $N = 6980$ vértices. Desta forma, é recuperada a sequência de *global meshs* de cada humano através da sua *global motion* $\boldsymbol{Q}^i$.

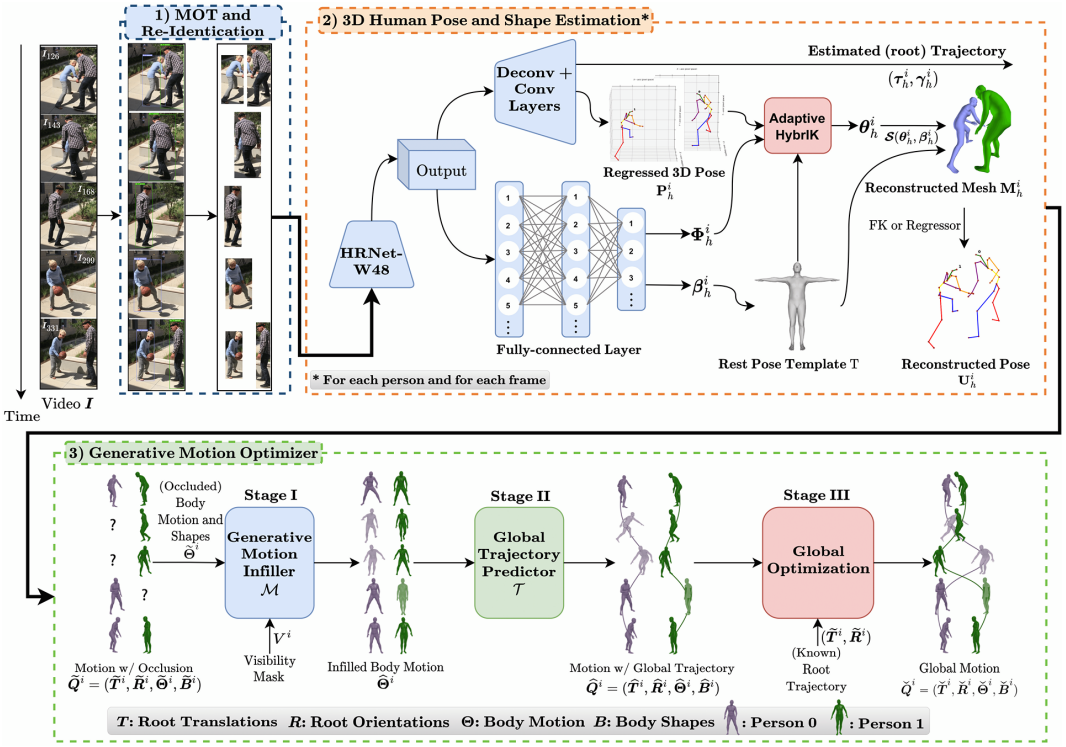


Figura 4.1: Arquitetura do sistema. O vídeo I é pré-processado com **MPT** e ***re-identification*** para alimentar o ***HPS estimator*** capaz de gerar, em cada *frame* I_h (exemplo para o *frame* $h = 126$), para cada humano i, o movimento com eventual oclusão $\widetilde{\boldsymbol{Q}}^i$ no referencial da câmara. Terminado este processo, esta informação é sujeita ***Generative Motion Optimizer***, dividido em três módulos, para uma otimização iterativa dos parâmetros da câmara enquanto são recuperados os parâmetros que mapeiam a trajetória, sem oclusões, do corpo humano $\widehat{\boldsymbol{Q}}^i$, definida no sistemas de coordenadas do mundo. Melhor interpretação da figura a cores e ampliada.

4.2 *MPT and Re-identification*

A avaliação em ambientes reais, compostos por vários humanos, em diferentes posições e com oclusões cruzadas, revela-se desafiante, principalmente quando se trata de um sistema monocular. A identificação do humano e acompanhamento do movimento a fim de estimar os parâmetros que o definem ao longo do tempo não deve sofrer de uma avaliação errática do cenário. Para este efeito, ao longo desta secção

[1] Função representada por $\boldsymbol{M}$ no artigo original [19]. Necessidade de reutilização no *Generative Motion Infiller* segundo a nomenclatura do GLAMR [34].

é descrito o MOT adotado, concretamente numa variante de *Multiple Person Tracking* (MPT) divida na deteção de humanos (Subsecção 4.2.1) e *tracking* otimizado a partir de *Re-Identication (Re-ID) models* (Subsecção 4.2.2).

Não sendo objetivo a projeção de um sistema de deteção e *tracking* de humanos, a seleção do modelo a incorporar baseou-se nas métricas comparativas para cada objetivo. Em suma, nesta etapa, naturalmente associada a pré-processamento, foi utilizado o detetor de humanos, previamente treinado, **YOLOv8x** [133]. O *human tracker*, robusto o suficiente para lidar com as não linearidades no movimento, trata-se do **Deep OC-SORT** [139] com uma ResNet-50 [60] como *Re-ID model*. Ambos as *frameworks* garantem uma resposta passível de execução em tempo real, com resultados que não devem perturbar a performance do restante processamento.

4.2.1 Deteção de Humanos

O vídeo, na entrada do sistema, é propriamente dividido com recurso à *multimedia framework* FFmpeg, nativamente executada do sistema operativo (*i.e.*, Ubuntu 20.04.6 LTS), em diferentes *frames* para alimentarem o detetor de humanos. Em causa, decorreram testes em diferentes *human detectors*, *e.g.*, YOLOX, YOLOv8 e YOLO-NAS com variações nas dimensões dos modelos.

O facto de se tratar de um sistema impossível de treinar de ponta a ponta poderá implicar propagação de erro por deteções incoerentes ao longo do tempo. O condicionamento dos parâmetros que regem o detetor poderão sofrer alterações em função das características do cenário e dos requisitos de precisão desejados. Neste sentido, poderia ser interessante aumentar a confiança do detetor na eventualidade da solução dispensar do módulo *generative motion optimizer* (tendo em atenção a possibilidade de aumento de falsos negativos) e garantia de deteções estruturalmente dispostas. Contudo, durante a inferência, face ao proposto e aos resultados verificados em diferentes cenários no *dataset* 3DPW [103], foi definido uma confiança de 0,60 na versão mais complexa do detetor, *i.e.*, YOLOv8x. Os resultados numéricos destes testes estão definidos na Secção 5.2, podendo ser feita uma avaliação qualitativa no Anexo A.

4.2.2 *Tracker and Re-identication Model*

As falhas do detetor de humanos devem ser compensadas para reduzir a suscetibilidade do mesmo em gerar *bounding boxes* erradas, *i.e.*, falta delas ou com uma área inferior ao visualmente possível de definir. Não obstante, face ao sucesso dos *human trackers*, foi incorporado na solução um algoritmo capaz de lidar com este tipo de problemas, além de oclusões parciais ou totais em cenários com elevada densidade populacional.

Foram incorporados diferentes *trackers*, *e.g.*, StrongSORT [140], OC-SORT [141], Deep OC-SORT [139] e ByteTrack [142], com inclusão de alguns dos mais usados *Re-ID models*. Na verdade, apenas alguns métodos combinam a aparência com os modelos heurísticos simples (*e.g.*, filtro de Kalman) para processamento do movimento na imagem.

Maggiolino *et al.* [139], numa análise experimental em dois dos mais conhecidos *benchmarks* para MOT na classe *"person"* (MOT17 [143] e MOT20 [144]), documentou resultados de métricas relevantes entre diferentes *trackers* (Tabela 4.1), que permitiram a escolha consciente do *re-identication model*. ***Multi-Object Tracking Accuracy* (MOTA)** é calculado com base nos ***False Positive* (FP)**, ***False Negative* (FN)** e **IDs**, focando-se exclusivamente no desempenho global no processo de deteção [140]. Em comparação, **IDF1** quantifica a performance do *tracker* em manter os IDs (corretos) atribuídos ao longo do tempo, enquanto ***Higher Order Tracking Accuracy* (HOTA)** reúne, numa única métrica, a precisão na deteção, associação e, efetivamente, trajetória. Ainda assim, a fragmentação (**Frag**) expressa a capacidade do algoritmo em manter a trajetória da deteção do humano, sem o perder ao longo dos *frames*, mesmo que o detetor falhe ocasionalmente [141].

Tabela 4.1: Resultados dos métodos testados em MPT nos *benchmarks* MOT17 e MOT20 [139].

MOT17							
Tracker	**HOTA↑**	**MOTA↑**	**IDF1↑**	**FP(10^4)↓**	**FN(10^4)↓**	**IDs↓**	**Frag↓**
ByteTrack [142]	63,1	**80,3**	77,3	2,55	**8,37**	2196	2277
StrongSORT [140]	64,4	79,6	79,5	2,79	8,62	1194	**1866**
OC-SORT [141]	63,2	78,0	77,5	**1,51**	10,8	1950	2040
Deep OC-SORT [139]	**64,9**	79,4	**80,6**	1,66	9,88	**1023**	2196
MOT20							
Tracker	**HOTA↑**	**MOTA↑**	**IDF1↑**	**FP(10^4)↓**	**FN(10^4)↓**	**IDs↓**	**Frag**
ByteTrack [142]	61,3	**77,8**	75,2	2,62	**8,76**	1223	1460
StrongSORT [140]	62,6	73,8	77,0	**1,66**	11,8	**770**	**1003**
OC-SORT [141]	62,1	75,5	75,9	1,80	10,8	913	1198
Deep OC-SORT [139]	**63,9**	75,6	**79,2**	1,69	10,8	779	1536

Neste sentido, com base no desenvolvimento realizado pela comunidade, o pós-processamento da deteção de humanos num determinado conjunto de imagens foi realizado com recurso ao **Deep OC-SORT** [139]. Trata-se de uma otimização do método OC-SORT [141] que, por sua vez, atuava como uma extensão ao *Simple Online and Realtime Tracking* (SORT), numa proposta exclusivamente baseada no movimento (*Kalman-filter-based*) e sem necessidade de realização de qualquer tipo de treino.

A adição de um *Re-ID model*, *e.g.*, ResNet-50 [60], e a inclusão de módulos para compensação da qualidade das deteções (*i.e.*, *camera motion compensation*), permitiram a incorporação, de forma adaptativa, da aparência dos objetos detetados para correspondência com os resultados do OC-SORT.

Para as mesmas deteções, o Deep OC-SORT mostrou melhorias consideráveis nas métrica HOTA, IDF1 e no número de IDs gerados, revelando o sucesso do algoritmo no *tracking* de humanos e na capacidade de reduzir o número de identificadores falsos criados durante ao longo da sequência.

A informação de cada pessoa detetada (*i.e.*, parâmetros que definem a localização da *bounding box* 2D) é devidamente armazenada ao longo de todos os *frames* do vídeo. É nesta fase que decorre o preenchimento da máscara de visibilidade $\boldsymbol{V}$ com indicação da presença do identificador num determinado *frame*.

Não obstante, alguns resultados qualitativos podem ser encontrados no Anexo A a fim de aferir visualmente a otimização de performance entre métodos.

4.3 *3D HPS Estimation*

A presente secção expõe o desenvolvimento efetuado no segundo módulo da arquitetura do sistema proposto (Figura 4.1) baseada numa solução analítico-neural de cinemática inversa, *i.e.*, HybrIK, inicialmente estudada na Subsecção 2.3.1 e propriamente implementada ao longo do projeto em causa. De uma forma sucinta, numa primeira fase, é utilizada uma rede neuronal para prever a localização das articulações $\mathbf{P}_h^i$, o ângulo *twist* Φ_h^i e os parâmetros de *shape* β_h^i. Numa segunda fase, os parâmetros de *shape* são usados para gerar a *rest pose* $\mathbf{T}$ (*blend shape*), adequada ao humano detetado, através do modelo SMPL. Combinando $\mathbf{P}_h^i$, $\mathbf{T}$ e Φ_h^i, são determinadas as rotações relativas $\mathbf{R}$ para a pose 3D, *i.e.*, parâmetros de pose θ_h^i, recorrendo ao método *Adaptive HybrIK*. Por fim, através da função linear do SMPL $\mathcal{S}(\theta_h^i, \beta_h^i)$, é obtida a ***articulated triangle mesh*** para então, a partir de FK ou de um *linear regressor*, obter a *reconstructed pose* $\mathbf{U}_h^i$.

Para clarificação da organização desta secção, numa primeira fase, Subsecção 4.3.1, é feita uma contextualização prática do modelo estatístico corporal SMPL, seguindo-se a Subsecção 4.3.2, com uma introdução simplificada da solução de cinemática inversa *Adaptive HybrIK* ajustada ao projeto. Por fim, na Subsecção 4.3.3, são apresentados os detalhes de implementação necessários.

4.3.1 *SMPL Model*

Neste trabalho foi aplicado o modelo paramétrico SMPL para representação do corpo humano. O SMPL estabelece uma descrição física do humano, visualmente estabelecida numa malha 3D, através de parâmetros de pose e *shape*. A *template mesh* $\boldsymbol{T}$, que define a topologia usada pelo modelo SMPL (*e.g.*, número de vértices, polígonos e articulações do esqueleto), pode ser condicionada pelos parâmetros do *shape* $\boldsymbol{\beta}_h^i \in \mathbb{R}^{10}$, compreendido num vetor com 10 valores escalares (expansão/compressão numa dada direção, *e.g.*, altura, tamanho da cintura, largura dos ombros, entre

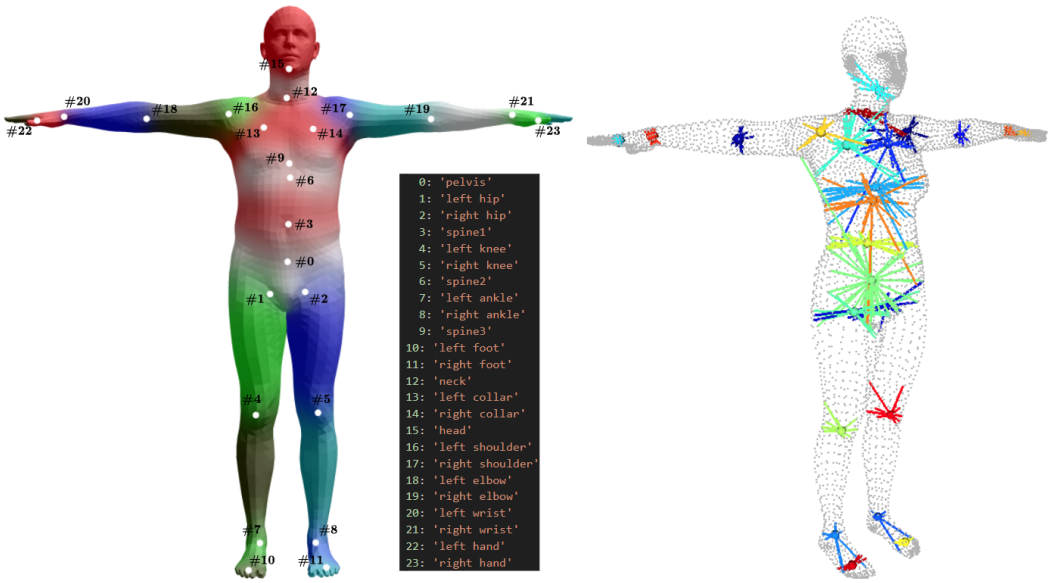
outros) e $K = 23$ articulações $\boldsymbol{\theta}_h^i = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_k)$ (assumindo que a primeira articulação (#0), será, no modelo inicial, a *root joint*, *i.e.*, *pelvis*). Concretamente, através da função linear do SMPL $\mathcal{S}(\boldsymbol{\theta}_h^i, \boldsymbol{\beta}_h^i)$, que aceita os parâmetros de pose $\boldsymbol{\theta}_h^i$ e *shape* $\boldsymbol{\beta}_h^i$ como entrada, é obtida uma ***articulated triangle mesh*** $\mathbf{M}_h^i \in \mathbb{R}^{N\times 3}$, com $N = 6980$ vértices. Como tal, a vertente articulada da malha triangular permite a rotação e alteração posicional (*e.g.*, "dobrar" um conjunto de pontos) dos pontos que a compõe face a uma dada articulação.

A hierarquia das 24 articulações está presente na Figura 4.2a através de pontos brancos. Factualmente, é definida com base numa rotação relativa à *parent joint*, disposta numa matriz 23×3. Isto é, cada rotação, contem uma representação compacta e conveniente de uma ***axis-angle rotation*** (*Rodrigues formulation*), por 3 valores escalares, em relação à *parent joint*. No entanto, será importante ressaltar que a representação simplificada com apenas 3 valores para descrever a variação angular das articulações, pode, em modelos mais sofisticados, ver-se convertida em quaterniões ou matrizes de rotação 3×3 que, *a priori*, levam em consideração mais informações sobre a orientação tridimensional das articulações.

Ainda assim, a relação entre articulações diz respeito ao modelo SMPL utilizado e é definida num mapa de conectividade. A variação posicional de cada articulação tem efeito direito nos vértices circundantes, mas com influências diferentes, pelo que, quando maior a proximidade à articulação afetada, maior o efeito na transformação. O mesmo efeito é verificado no sentido contrário, pelo que a localização local das articulações sofre influência da variação da malha, indicado nas linhas coloridas da Figura 4.2b.

É segundo uma combinação linear dos vértices da malha ou segundo a aplicação um processo de *Forward Kinematics* (FK) (contrário ao processo explicado na Subsecção 4.3.2) que é obtida a ***reconstructed pose*** $\mathbf{U}_h^i$ presente no final do segundo módulo da arquitetura da Figura 4.1. Não obstante, durante o processo de otimização, a sequência de malhas que acompanham a variação posicional do humano no mundo é igualmente obtida a partir da *global motion* $\boldsymbol{Q}^i$, tal como referido na contextualização da solução.

De notar que uma implementação com SMPL-X ou SMPLify-X [91] auferia de melhores resultados visuais com a variação do género, explicado pelo aumento de articulações e parâmetros de *shape* (300 parâmetros) que descrevem o humano. Ainda assim, desconhecendo o género do humano, durante o projeto foi considerado o modelo SMPL neutro, anteriormente treinado *CAESAR dataset* [47] composto por 1700 elementos masculinos e 2107 elementos femininos.



(a) Inicialização da localização das 24 articulações

(b) *Joint regression* após otimização: relação ponderada entre a malha e as articulações do humano

Figura 4.2: *SMPL human model* [19].

4.3.2 Hybrid Analytical-Neural: Inverse Kinematics

Os conceitos por de trás da solução proposta por Li *et al.* [23] suscitaram interesse ao autor do presente projeto pela abordagem cinemática implementada e pelos notáveis resultados à data da sua exposição. De facto, trata-se de um sistema híbrido (à semelhança do MPT adotado) que propõe combinar, conhecimentos analíticos predominantemente descritivos do movimento humano no espaço com propriedades visuais passíveis de serem processadas na flexibilidade de uma rede neuronal.

Como tal, a contribuição principal fundamenta-se numa inovadora solução *Inverse Kinematics* (IK) através de uma *twist-and-swing decomposition* sucintamente analisada nesta subsecção. Não obstante, a escolha deste método deve-se à possibilidade de obtenção das poses humanas 3D no sistema de coordenadas da câmara com inclusão da referida *root translations* absoluta (*depth estimation* com base na RootNet [26]), exigida no algoritmo do *generative motion optimizer*, enquanto muitos outros métodos para HPS não determinam esta informação.

Todo o processamento, esquematicamente representado no segundo módulo da arquitetura presente na Figura 4.1, visa a redução do erro entre a malha corporal projetada e os *keypoints* 3D estimados. Na verdade, em vez de estimar diretamente a posição das articulações 3D, o modelo adota *volumetric heatmaps* para, em parceria com a malha corporal definida no modelo paramétrico SMPL, fechar o ciclo com a determinação das articulações 3D do esqueleto humano.

Concretamente, este módulo deve aceitar cada elemento da sequência de *image*

patches dos humanos detetados (obtido do MPT) e gerar, respetivamente, a pose e *shape* inicial. Ainda assim, a arquitetura do modelo que origina os resultados referidos deve ser alvo de estudo para correta incorporação e adaptação no projeto.

Forward and Inverse Kinematics

A colaboração entre as articulações 3D e a malha corporal atua como um elemento essencial na *pipeline* do método. Por um lado, a precisão das articulações 3D facilitam a determinação da malha humana, por outro lado, o modelo paramétrico tende a corrigir a questão da estrutura corporal irrealista (*i.e.*, dependência de modelações explícitas de distribuições do comprimento dos ossos tendem a gerar assimetria estrutural do humano) presente nos métodos de estimação diretos. Ao considerar o modelo paramétrico do corpo, a forma final deve adaptar-se melhor ao humano real.

Inverse Kinematics (IK) é um processo matemático que procura determinar as rotações relativas $\mathbf{R} = \{R_{\mathtt{pa}(k),k}\}_{k=1}^{K}$ sendo conhecidas as posições das articulações humanas $\mathbf{P} = \{p_k\}_{k=1}^{K}$. Trata-se de um problema possível de resolver com um número indeterminado de soluções. Ainda assim, assumindo o *rest pose template* $\mathbf{T} = \{t_k\}_{k=1}^{K}$, o processo de determinação destas rotações relativas pode ser formalizado por:

$$\mathbf{R} = \mathrm{IK}(\mathbf{P}, \mathbf{T}), \tag{4.1}$$

onde $t_k \in \mathbb{R}^3$ remete para a k-ésima articulação do *rest pose template* $\mathbf{T}$ e $p_k \in \mathbb{R}^3$ diz respeito à k-ésima articulação da pose de entrada $\mathbf{P}$, definida em K *keypoints*, do IK. Seguindo a mesma lógica apresentada anteriormente na Subsecção 4.3.1, $\mathtt{pa}(k)$ retorna o índice de parentesco da k-ésima articulação e $R_{\mathtt{pa}(k),k}$ será a rotação relativa da k-ésima articulação em relação à sua *parent joint*. Devido à natureza da cinemática inversa, os autores do método impuseram a seguinte condição para seleção da uma solução única:

$$p_k - p_{\mathtt{pa}(k)} = R_k(t_k - t_{\mathtt{pa}(k)}) \quad \forall 1 \leqslant k \leqslant K, \tag{4.2}$$

onde $R_k \in \mathbb{SO}(3)$ remete para a rotação da k-ésima articulação em relação ao *canonical rest pose space* (conjunto de todas as configurações de articulações que definem uma *rest pose* válida para o modelo, isto é, posição neutra). Neste sentido, a condição evidência que a solução do ângulo relativo de uma determinada articulação k, quando aplicado a partir da *rest pose* $(t_k - t_{\mathtt{pa}(k)})$, leva o modelo à posição desejada $(p_k - p_{\mathtt{pa}(k)})$ no espaço tridimensional.

Tal como referido anteriormente, foi assumindo que a primeira articulação (#0), será, no modelo inicial, a *root joint* (*i.e.*, *pelvis*) e, portanto, $p_0 = t_0$ sem qualquer relação de parentesco.

Por sua vez, *Forward Kinematics* (FK), no contexto HPE, refere-se ao processo matemático capaz de calcular a *reconstructed pose* $\mathbf{U} = \{u_k\}_{k=1}^{K}$ na saída do segundo módulo da arquitetura presente na Figura 4.1, sendo conhecidas as rotações relativas à respetiva *parent joint*. Mantendo a mesma nomenclatura, mas admitindo como entrada do sistema a rotação relativa $\mathbf{R}$ e o *rest pose template* $\mathbf{T}$, a *reconstructed pose* pode ser obtida por:

$$\mathbf{U} = \mathrm{FK}(\mathbf{R}, \mathbf{T}), \tag{4.3}$$

onde $u_k \in \mathbb{R}^3$ remete para a localização da k-ésima articulação. Naturalmente, FK é conseguido pela rotação recursiva da parte do corpo definida no *template* desde a *root joint* até à última articulação:

$$u_k = R_k(t_k - t_{\mathtt{pa}(k)}) + u_{\mathtt{pa}(k)}, \tag{4.4}$$

Complementar ao apresentado no cálculo IK e pela relação de parentesco entre articulações, podemos concluir que a rotação da k-ésima articulação é decomposta no produto da rotação face ao *rest pose template* da *parent joint* com a rotação relativa à mesma *parent joint*:

$$R_k = R_{\mathtt{pa}(k)} R_{\mathtt{pa}(k),k}. \tag{4.5}$$

Novamente, a *root joint* (*i.e.*, *pelvis*) não possui qualquer *parent joint*, pelo que considera-se $u_0 = t_0$.

Twist-and-Swing Decomposition

Face ao pressuposto, será questionável como decorrerá a determinação destas rotações relativas. Para tal, existe uma cooperação efetiva entre as potencialidades analíticas e neuronais e, a partir dos *keypoints* 3D estimados, otimizar a malha corporal.

Uma vez que a partir das pose 3D determinada não é possível obter diretamente a rotação relativa de cada articulação em relação à *rest pose*, pela diversidade de rotações resultantes que origem a mesma configuração de articulações, os autores do HybrIK decompuseram a suposta rotação original em *twist* (rotação com apenas 1-DoF sobre o eixo longitudinal do membro) e *swing* (rotação no plano). Em suma, as articulações, inicialmente estimadas, são utilizadas para calcular, analiticamente, a rotação *swing* enquanto a rotação *twist* é determinada com base numa análise neuronal de factos visuais na imagem.

Apesar da limitação anatómica das articulações do corpo humano (*i.e.*, algumas articulações estão restritas a apenas 1-DoF ou 2-DoF, como o caso do joelho e tornozelo, respetivamente), os autores do método generalizaram o processo e atribuíram

a qualquer articulação 3-DoF. Tal como ilustrado na Figura 4.3, a decomposição desta rotação $\boldsymbol{R} \in \mathbb{SO}(3)$ é dada por *twist* $\boldsymbol{R}^{tw}$ e *swing* $\boldsymbol{R}^{sw}$ igualmente definidas numa representação *axis-angle*. De facto, esta representação é conveniente em modelos matemáticos que lidem com a dinâmica de corpos rígidos, podendo estes ser parametrizados pelo eixo fixo de rotação e o ângulo efetivo de rotação.

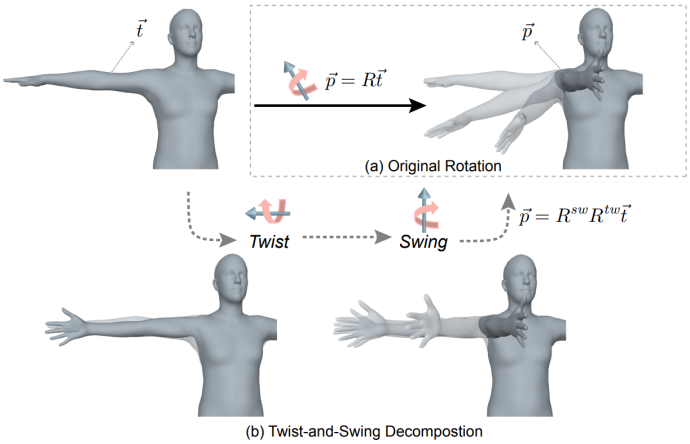


Figura 4.3: Ilustração da *twist-and-swing decomposition* [23].

Sendo conhecido o vetor do modelo inicial da parte do corpo alvo, $\vec{t}$ e o vetor final pretendido $\vec{p}$, o processo para determinação da nova rotação é propriamente formulado por:

$$\boldsymbol{R} = \boldsymbol{\mathcal{D}}(\vec{p}, \vec{t}, \phi) = \boldsymbol{\mathcal{D}}^{sw}(\vec{p}, \vec{t})\boldsymbol{\mathcal{D}}^{tw}(\vec{t}, \phi) = \boldsymbol{R}^{sw}\boldsymbol{R}^{tw}, \tag{4.6}$$

onde ϕ remete para o ângulo *twist* determinado pelo rede neuronal (*FC layers* no segundo módulo da arquitetura da Figura 4.1), $\boldsymbol{\mathcal{D}}^{sw}(\cdot)$ é a solução analítica (*closed-form solution*) da rotação *swing* $\boldsymbol{R}^{sw}$ e $\boldsymbol{\mathcal{D}}^{tw}(\cdot)$ o processo capaz de realizar a transformação do ângulo ϕ para a rotação *twist* $\boldsymbol{R}^{tw}$. Novamente, esta nova rotação $\boldsymbol{R}$ deverá respeitar a condição da Equação 4.2, *i.e.*, $\vec{p} = \boldsymbol{R}\vec{t} = \boldsymbol{R}^{sw}\boldsymbol{R}^{tw}\vec{t}$

É, portanto, percetível, o movimento de rotação resultante desta decomposição, na medida em que a rotação *swing* apresenta um determinado eixo $\vec{n}$ perpendicular a $\vec{t}$ e $\vec{p}$ normalizado pela seguinte equação:

$$\vec{n} = \frac{\vec{t} \times \vec{p}}{\|\vec{t} \times \vec{p}\|}, \tag{4.7}$$

pelo que o ângulo *swing* α originado deverá satisfazer as seguintes igualdades:

$$\cos\alpha = \frac{\vec{t} \cdot \vec{p}}{\|\vec{t}\|\|\vec{p}\|}, \quad \sin\alpha = \frac{\|\vec{t} \times \vec{p}\|}{\|\vec{t}\|\|\vec{p}\|}, \tag{4.8}$$

E tal como indicado anteriormente, a *closed-form solution* da rotação *swing* R^{sw} provem da *Rodrigues formulation*:

$$\boldsymbol{R}^{sw} = \boldsymbol{\mathcal{D}}^{sw}(\vec{p}, \vec{t}) = \boldsymbol{\mathcal{I}} + \sin\alpha[\vec{n}]_\times + (1 - \cos\alpha)[\vec{n}]_\times^2, \tag{4.9}$$

sendo $[\vec{n}]_\times$ a matriz antissimétrica de $\vec{n}$ e $\boldsymbol{\mathcal{I}}$ é a matriz identidade 3×3.

Por sua vez, a rotação *twist* $\boldsymbol{R}^{tw}$ remete para uma rotação em torno do vetor $\vec{t}$, sendo este o eixo de rotação, é possível determinar $\boldsymbol{R}^{tw}$ por:

$$\boldsymbol{R}^{tw} = \boldsymbol{\mathcal{D}}^{tw}(\vec{t}, \phi) = \boldsymbol{\mathcal{I}} + \frac{\sin\phi}{\|\vec{t}\|}[\vec{t}]_\times + \frac{(1 - \cos\phi)}{\|\vec{t}\|^2}[\vec{t}]_\times^2, \tag{4.10}$$

admitindo, novamente, que $[\vec{t}]_\times$ será matriz antissimétrica de $\vec{t}$.

Os detalhes adicionais do processo analítico estão presentes no artigo original do método adotado [23], ficando desta forma evidenciado uma orientação que permitirá acompanhar o raciocínio nas restantes secções. Ainda assim, o processo de aprendizagem no dimensionamento do ângulo *twist* é consideravelmente simplificado uma vez que esta variação angular é definida por uma única variável (1-DoF) e limitada por restrições anatómicas do humano.

Adaptive HybrIK Model

No modelo SMPL, os parâmetros da pose $\boldsymbol{\theta}$ controlam a rotação das partes rígidas do corpo. As três articulações apeladas `left hip`, `right hip` e `spine1` formam a parte rígida do corpo, que é controlada pela *root rotation* $\boldsymbol{R}_0$, definida efetivamente como *root orientation* na saída do *HPS estimator* para o humano i no *frame* h por $\boldsymbol{\gamma}_h^i$ e entrada do *generative motion optimizer* da arquitetura da Figura 4.1 por $\widetilde{\boldsymbol{R}}^i$. Neste sentido, a *root rotation*, no referencial da câmara, pode ser obtida pela rotação efetiva que minimiza a diferença no alinhamento entre a localização das articulações estimadas $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ e $\boldsymbol{p}_3$, respetivamente, e a localização das mesmas no *rest pose template* dadas por $\boldsymbol{t}_1$, $\boldsymbol{t}_2$ e $\boldsymbol{t}_3$. Esta otimização é realizada em três pares possíveis com referência à *root joint*, formalizada por:

$$\boldsymbol{R}_0 = \arg\min_{R \in \mathbb{SO}^3} \sum_{i=1}^{3} \|p_i - \boldsymbol{R}t_i\|_2^2 \tag{4.11}$$

A determinação da *root rotation* é comum aos dois métodos, desenvolvidos pelos autores do HybrIK [23], para obtenção da *reconstructed pose* **U** apeladas de ***Naive HybrIK*** e ***Adaptive HybrIK***. Por ventura, o segundo método decorre de uma otimização que procura minimizar os erros verificados no *Naive HybrIK*. A Figura 4.4 esquematiza a diferença acumulada de erro entre os dois métodos. A *rest pose* é rodada para u_1 e u_2 em dois passos. O erro $\vec{\boldsymbol{\epsilon}_1}$ gerado no primeiro passo é igual em ambos os métodos, porém, no segundo passo, surge um novo erro $\vec{\boldsymbol{\epsilon}_2}$, singular no *Adaptive HybrIK* mas aditivo a $\vec{\boldsymbol{\epsilon}_1}$ no *Naive HybrIK*. A eliminação desta propagação

de erro deve-se à seleção adaptativa da $u_{\mathtt{pa}(k)}$ para definição de um novo vetor corretivo dado por $p_k - u_{\mathtt{pa}(k)}$, *i.e.*, seleção de u_1 para o vetor de direção $p_2 - u_1$.

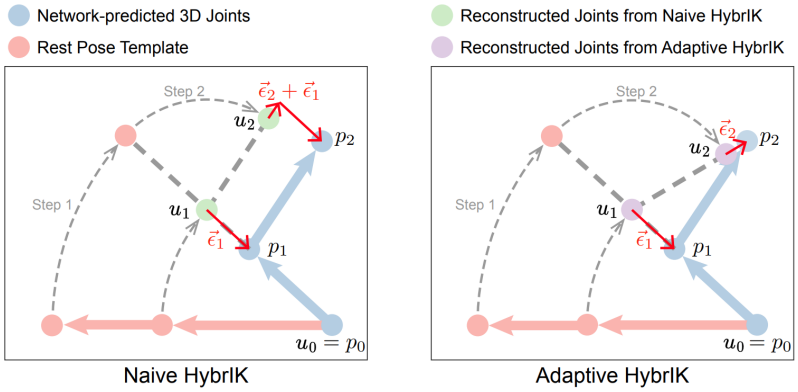


Figura 4.4: Exemplo da projeção do erro de reconstrução: *Naive vs. Adaptive HybrIK* [23].

Para análise do erro de reconstrução $\boldsymbol{\epsilon}_k$ na articulação k decorreu uma comparação entre a pose de entrada $\mathbf{P}$ e a *reconstructed pose* $\mathbf{U}$:

$$\|\mathbf{P} - \mathbf{U}\| \Leftrightarrow \sum_{k=1}^{K} \|p_k - u_k\|, \tag{4.12}$$

onde $\mathbf{U} = \mathrm{FK}(\mathbf{R}, \mathbf{T}) = \mathrm{FK}(\mathrm{IK}(\mathbf{P}, \mathbf{T}), \mathbf{T})$. Para evitar a acumulação de erro verificado no processo inicial, o *Adaptive HybrIK* atualiza o vetor a atingir de forma adaptativa através da recente determinada *parent joint* $u_{\mathtt{pa}(k)}$. Formalizando, para o vetor $\vec{p}_k = R^{-1}_{\mathtt{pa}(k)} \left(p_k - u_{\mathtt{pa}(k)}\right)$ e $\vec{t}_k = \left(t_k - t_{\mathtt{pa}(k)}\right)$ a condição do *Adaptive HybrIK* (Equação 4.2) pode ser reformulada para:

$$p_k - u_{\mathtt{pa}(k)} = R_k \left(t_k - t_{\mathtt{pa}(k)}\right) + \vec{\epsilon}_k. \tag{4.13}$$

Neste sentido, é possível estabelecer que:

$$p_k - u_{\mathtt{pa}(k)} = u_k - u_{\mathrm{pa}(k)} + \vec{\epsilon}_k \Rightarrow p_k - u_k = \vec{\epsilon}_k. \tag{4.14}$$

Recordando a Figura 4.4, no *Naive HybrIK*, o erro de reconstrução apenas dependerá da articulação em causa no momento da aplicação do FK, não acumulando dos antecessores.

A implementação pode ser sintetizada a partir do Algoritmo 2. Síncrono com o conteúdo apresentado na arquitetura do projeto, este método deverá admitir na entrada a pose estimada $\mathbf{P}$, a *rest pose template* $\mathbf{T}$ o conjunto dos ângulos *twist* $\boldsymbol{\Phi} = \{\phi_k\}_{k=1}^{K}$. A saída deverá ser o conjunto de rotações relativas $\mathbf{R} = \{R_{\mathtt{pa}(k),k}\}_{k=1}^{K}$.

Como tal, o *Adaptive HybrIK* ocorre num ciclo que deverá percorre toda a *árvore cinemática humana*, sendo necessário a determinação prévia da *root rotation* R_0. Para cada articulação estimada determina-se a localização da sua *parent joint*

Algorithm 2 Adaptive HybrIK

Input: P, T, Φ
Output: R

1: Determine R_0;
2: **for** k ao longo da árvore cinemática do humano **do**
3: $u_{\mathtt{pa}(k)} \leftarrow R_{\mathtt{pa}(k)}(t_{\mathtt{pa}(k)} - t_{\mathtt{pa}^2(k)}) + u_{\mathtt{pa}^2(k)}$;
4: $\vec{p}_k \leftarrow R^{-1}_{\mathtt{pa}(k)}(p_k - u_{\mathtt{pa}(k)})$;
5: $\vec{t}_{\mathtt{k}(k)} \leftarrow (t_k - t_{\mathtt{pa}(k)})$;
6: $R^{sw}_{\mathtt{pa}(k),k} \leftarrow \mathcal{D}^{sw}(\vec{p}_k, \vec{t}_k)$;
7: $R^{tw}_{\mathtt{pa}(k),k} \leftarrow \mathcal{D}^{tw}(\vec{t}_k, \phi_k)$;
8: $R_{\mathtt{pa}(k),k} \leftarrow R^{sw}_{\mathtt{pa}(k),k} R^{tw}_{\mathtt{pa}(k),k}$;
9: **end for**

reconstruída $u_{\mathtt{pa}(k)}$. Importa ressalvar que $\mathtt{pa}^2(k)$ denota o índice da *parent joint* de $\mathtt{pa}(k)$. Definidos os vetores $\vec{p}_k$ e $\vec{t}_k$, pelas expressões anteriormente referidas, procede-se uma decomposição da rotação relativa nas suas componentes $R^{sw}_{\mathtt{pa}(k),k}$ (Equação 4.9) e $R^{tw}_{\mathtt{pa}(k),k}$ (Equação 4.10). O produto destas deverá originar a rotação relativa $R_{\mathtt{pa}(k),k}$ para a articulação em causa.

4.3.3 Informação Complementar do *HPS Estimator*

Esta secção destaca conceitos complementares na implementação do *HPS Estimator*. Neste sentido, alguns dos resultados qualitativos, reunidos no Anexo B e C podem ser avaliados em paralelo com conteúdo aqui presente.

Numa versão inicial, o HybrIK adotava uma ResNet-34 [60] como *backbone* da rede. Mais recentemente, durante o estudo deste projeto, os autores deste método publicaram um novo *backbone*, previamente treinado para operar no modelo paramétrico SMPL e SMPL-X [145]. Em causa está uma HRNet-W48 [40] que promete otimizar a performance do sistema, minimizando as incoerências estruturais humanas e reduzindo o *jitter* verificado ao longo de um vídeo. Na Subsecção 5.3.1 será avaliada esta melhoria que evidência aumento da suavidade de movimento previamente e após o *generative motion optimizer*. Contudo, seria de todo relevante analisar os resultados tabelados pelos autores nos diferentes *backbones*. A Tabela 4.2 contempla exatamente essa informação.

Mean Per Joint Position Error (MPJPE) calcula o erro médio da posição de cada articulação com base na distância euclidiana entre as posições das articulações verdadeiras (*ground truth joints*) e as posições estimadas ao longo de todos os *frames* de um *dataset*, expressa em milímetros para poses 3D e *pixel* para poses 2D. Por sua vez, *Procrustes-Aligned MPJPE* (PA-MPJPE) atua como uma extensão do *Mean Per Joint Position Error* (MPJPE) ao incorporar um processo de alinhamento antes de calcular o erro (*Procrustes alignment*). Trata-se de uma técnica usada para

Tabela 4.2: *Benchmark* HybrIK com diferentes *backbones* nos *datasets* 3DPW e Human3.6M [145].

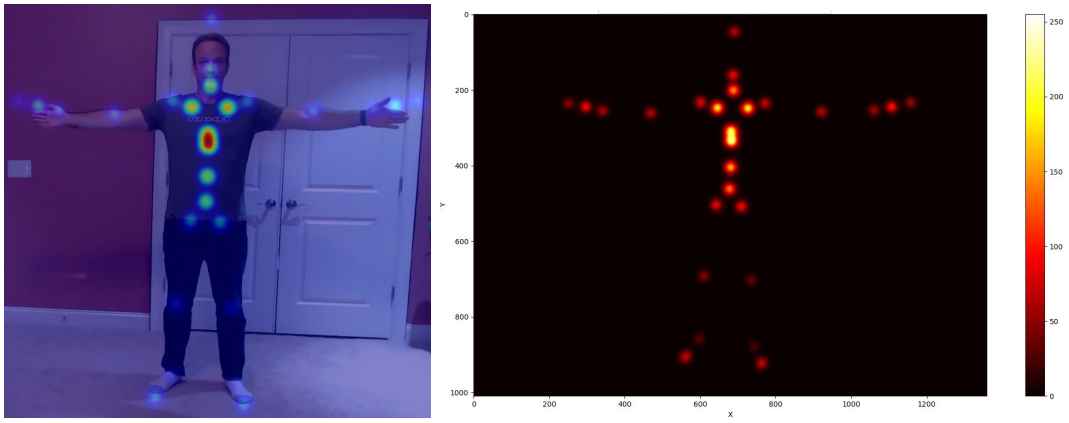
Backbone	Training Data	3DPW		Human3.6M	
		PA-MPJPE↓	MPJPE↓	PA-MPJPE↓	MPJPE↓
ResNet-34	w/ 3DPW	44,6	72,5	33,7	55,3
HRNet-W48	w/o 3DPW	48.6	88.0	**29.5**	50.4
HRNet-W48	w/ 3DPW	**41.8**	**71.6**	29.8	**47.0**

alinhar dois conjuntos de pontos de forma a minimizar a soma dos quadrados das diferenças entre os mesmos. Desta forma, para determinação do PA-MPJPE, as posições estimadas das articulações são primeiro alinhadas com as posições verdadeiras usando a análise de Procrustes e, em seguida, o erro médio por articulação é calculado com base nas posições alinhadas, mitigando os problemas inerentes de escala, rotação e translação.

Inicializado com os pesos pré-treinados do *ImageNet*, o treino da HRNet-W48 resultou num ajuste de pesos para determinação da pose, *shape* e ângulo *twist*. Esta aceleração da convergência do treino, pelo conhecimento prévio adquirido na fase de pré-treino, revelou valores de PA-MPJPE e MPJPE consideravelmente inferiores ao *backbone* inicial, sendo, portanto, alvo de escolha na restante implementação. O Anexo C reúne um conjunto de resultados qualitativos que evidenciam a coerência anatómica, por vezes não verificada, na implementação do HybrIK com a ResNet-34.

A saída da HRNet-W48 é ramificada em dois processos distintos. O primeiro ramo foca-se na obtenção dos *heatmaps* 3D (*volumetric heatmaps*) através das *feature* presentes na imagem, com dimensão $C \times 64 \times 64$. Neste caso, $C = 64$ é a dimensão do canal. A imagem em causa remete para a *image patch*, em cada humano, definida pela área da *bounding box* no MPT. Esta sofre um conjunto de transformações para adequação ao modelo e otimização na determinação da profundidade, *e.g.*, a imagem de entrada é redimensionada a 256×256 enquanto a *bounding box* sofre um redimensionada na mesma proporção.

Inspirados noutros trabalhos, os autores propuseram 3 *deconvolution layers* com uma função de ativação *ReLU* seguidos de 1×1 *convolution layer* para gerar os *volumetric heatmaps*. Esta *pipeline* permite aumentar a resolução das *features* inicialmente detetadas e não compromete a característica diferenciável do modelo. Os *heatmaps* são propriamente normalizadas pela aplicação da função *softmax* ao longo dos valores que descrevem a distribuição Gaussiana 3D. Como tal, a localização efetiva da articulação encontra-se no ponto mais provável definindo nesta distribuição. A Figura 4.5 apresenta o aspeto visual destes *heatmaps* a duas dimensões. O resultado visual é obtido num pós-processamento, maximizando (ou numa média) os valores de cada *heatmap* ao longo da componente de profundidade.



(a) Representação com 3 canais

(b) Representação com cores de gradiente

Figura 4.5: Visualização dos *heatmaps* 2D de todas as articulações. A intensidade reduzida na representação das distribuições Gaussianas em algumas articulações deve-se ao processo de normalização ao longo da componente de profundidade, remetendo para uma menor certeza da sua localização face às de maior intensidade.

De notar que nas Figuras 4.5a e 4.5b se encontra representado 29 *heatmaps*, contrariamente ao definido pelo modelo esquelético do SMPL. Na verdade, este conjunto de *heatmaps* dão origem a cinco articulações adicionais (*distal joints*), necessárias para a determinação das 24 rotações relativas na representação da pose humana durante o processo de cinemática inversa (IK). E como tal, no resultado final deste módulo, Figura 4.6, apenas são exibidas as 24 articulações.

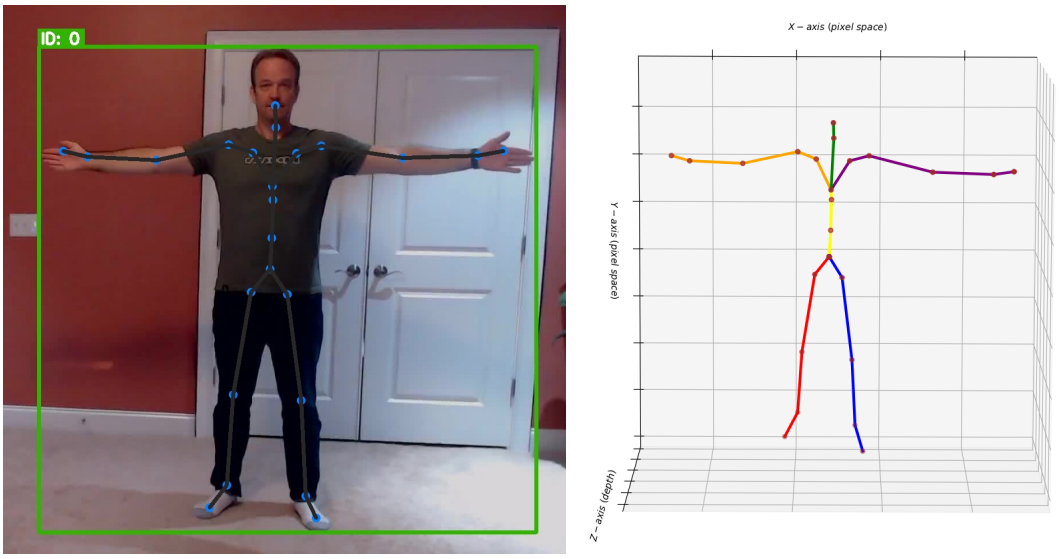
Por sua a vez, a segunda ramificação é composta por um *average pooling* para o calculo da média da intensidade dos pixeis, reduzindo a dimensionalidade das *features* extraídas e destacando características relevantes para obtenção dos ângulos *twist* $\mathbf{\Phi}$ e dos parâmetros de *shape* $\boldsymbol{\beta}$ pertencente ao modelo SMPL. Para tal, as *features* extraídas são ponderadamente correlacionadas através dois *FC layers*. Uma primeira camada com 1024 neurónios (com uma camada de *dropout*, para um *dropout rate* de 0,5) e uma última camada com 56 neurónios (10 para $\boldsymbol{\beta}$ e 46[2] para $\mathbf{\Phi}$).

No que diz respeito à determinação da profundidade decorre um processo, com uma estrutura similar à *RootNet* [26], para o cálculo do valor absoluto da distância à *root joint* $\mathbf{A}= (x_R, y_R, Z_R)$. Efetivamente, os pontos 2D estimados na imagem são retroprojetados (*back-projection*) para o referencial da câmara (relativas ao centro ótico desta) recorrendo a um valor de profundidade estimado Z_R (valor final da terceira componente). Para este efeito, os autores da *RootNet* fizeram uso das informações presentes na imagem, relacionando a área do humano real no espaço A_{real} (mm^2) e na imagem A_{img} $(pixel^2)$ e, após uma normalização da área efetiva dada pela *bounding box* do humano em causa (A_{real} normalizada 2000 $mm \times 2000$ mm, sendo A_{img} parte da área em A_{real}), inclusão de um fator de correção para lidar com

[2] *Tensor* referente à rotação no eixo de um dado osso, convertido para $(23, 2)$, com base na representação *axis-angle*.

situações passíveis de ocorrer em ambientes reais, *i.e.*, uma criança com A_{img} inferior à área dada pela *bounding box* de um adulto, ambos à mesma distância da câmara, permitindo aumentar a A_{img} do adulto e prever devidamente a profundidade da *root joint* (*i.e.*, *pelvis*) para cada humano.

Posto isto, os dados obtidos à saída do modelo podem ser propriamente trabalhos, permitindo obter, a partir dos *heatmaps* apresentados na Figura 4.5, um conjunto de articulações projetadas na imagem inicial, Figura 4.6a, ou reconstruídas no sistema de coordenadas da câmara **U** pela aplicação de cinemática direta FK, Figura 4.6b. Implementação realizada em PyTorch[3].



(a) Identificação do humano 0 e projeção das articulações e ossos do humano na imagem 2D original

(b) Reconstrução das articulações 3D e ossos do humano no cenário

Figura 4.6: Resultado visual da reconstrução de pose do *Adaptive HybrIK*.

4.4 *Generative Motion Optimizer*

A implementação podia ver-se finalizada nesta fase, contudo, o autor deste trabalho procurou compensar as vulnerabilidades expostas num sistema monocular. O GLAMR [34], anteriormente apresentado, reúne as condições para processar a informação que descreve o movimento, com oclusão, das L pessoas no cenário, obtida no *HPS estimator* e inferir, durante um processo de otimização, potenciais posições em falta, sendo conhecidas um conjunto de poses aleatoriamente estabelecidas no espetro temporal. O modelo é especialmente complexo mas passível de ser incorporado em sistemas que cumpram determinados requisitos, *i.e.*, a sequência de pose e *shape* (movimento corporal) $\widehat{\boldsymbol{Q}}^{i}$ da pessoa i visível deve ser determinada no sistema de coordenadas da câmara segundo o modelo SMPL.

[3]Detalhes concretos das bibliotecas e *frameworks* utilizadas descrito no Anexo F.

Esta integração é cumprida em três fases, sucintamente descritas nesta secção e devidamente referidas na contextualização da solução (Secção 4.1). Face aos resultados apresentados no artigo oficial do projeto (e.g., *long-term occlusions* na determinação da pose dos humanos) e aos recursos computacionais disponibilizados à equipa de desenvolvimento da NVIDIA, não foi efetuado qualquer tipo de treino adicional para *motion infiller* $\boldsymbol{\mathcal{M}}$ ou para o *trajectory predictor* $\boldsymbol{\mathcal{T}}$ uma vez que o autor deste trabalho acredita que não seria acrescentada qualquer tipo de mais-valia. Não obstante, será relevante evidenciar de que forma decorre a *pipeline* do modelo para uma correta adaptação ao *HPS estimator* desenvolvido e maximização dos resultados em alguns *benchmarks* realizados pelos autores do GLAMR.

O Anexo D reúne alguns resultados qualitativos após o processo de otimização que permitiu a otimização da trajetória global de todas as pessoas e dos parâmetros da câmara a fim de produzir *global motions* $\breve{\boldsymbol{Q}}^i$, sem oclusão e no sistemas de coordenadas do mundo.

4.4.1 *Generative Motion Infiller*

Cabe ao *Generative Motion Infiller* $\boldsymbol{\mathcal{M}}$ a tarefa de preencher, para cada humano, as pose do corpo (*body motion*) $\breve{\boldsymbol{\Theta}}^i$ e gerar uma *body motion* sem oclusões $\widehat{\boldsymbol{\Theta}}^i$. Por muito que seja tentador, o *motion infiller* não deve inferir outras componentes no movimento estimado $\widehat{\boldsymbol{Q}}^i$, *i.e.*, *root trajectory* $(\widetilde{\boldsymbol{T}}^i, \widetilde{\boldsymbol{R}}^i)$ e *shapes* $\widetilde{\boldsymbol{B}}^i$. Em vez disso, este processo é realizado no *global trajectory predictor* proposto no GLAMR, a partir da pose corporal preenchida $\widehat{\boldsymbol{\Theta}}^i$, dando origem a uma trajetória global sem oclusão $(\widehat{\boldsymbol{T}}^i, \widehat{\boldsymbol{R}}^i)$ enquanto os *shapes* sem oclusão $\widehat{\boldsymbol{B}}^i$ proveem de uma interpolação linear ao longo da sua variação temporal em cada *frame* [34].

Para sistemas com câmaras dinâmicas é notória a dificuldade na determinação da *root trajectory* baseada na dinâmica de movimento humano aprendida pela rede neuronal, uma vez que os dados provenientes do *HPS estimator* se encontram no referencial da câmara. Ainda assim, a *root trajectory* $(\widetilde{\boldsymbol{T}}^i, \widetilde{\boldsymbol{R}}^i)$ será utilizada no *global optimizer* (Secção 4.4.3).

De forma generalizada, a ideia será obter a *body motion* sem oclusão $\widehat{\boldsymbol{\Theta}} = (\widehat{\boldsymbol{\theta}}_1, \ldots, \widehat{\boldsymbol{\theta}}_H)$ em H *frames* a partir de uma máscara de visibilidade $\boldsymbol{V} = (\boldsymbol{V}_1, \ldots, \boldsymbol{V}_H)$ e da *body motion* com oclusão $\widetilde{\boldsymbol{\Theta}} = (\widetilde{\boldsymbol{\theta}}_1, \ldots, \widetilde{\boldsymbol{\theta}}_H)$:

$$\widehat{\boldsymbol{\Theta}} = \boldsymbol{\mathcal{M}}(\widetilde{\boldsymbol{\Theta}}, \boldsymbol{V}, \boldsymbol{z}), \tag{4.15}$$

onde $\boldsymbol{z}$ corresponde a um *latent code* obtido de uma distribuição Gaussiana (Figura 2.35) que permitirá variar a resposta do *motion infiller* $\boldsymbol{\mathcal{M}}$ e, portanto, gerar diferentes *body motion* sem oclusão $\widehat{\boldsymbol{\Theta}}$.

O *motion infiller* $\boldsymbol{\mathcal{M}}$ foi treinado a partir do *dataset* AMASS [88], para que a rede sintetizasse a dinâmica humana em diferentes cenários, contudo, a fim de

garantir a abrangência em intervalos de oclusão superior aos de treino, foi proposto um **Autoregressive Motion Infilling** (Figura 4.7) durante a fase de teste.

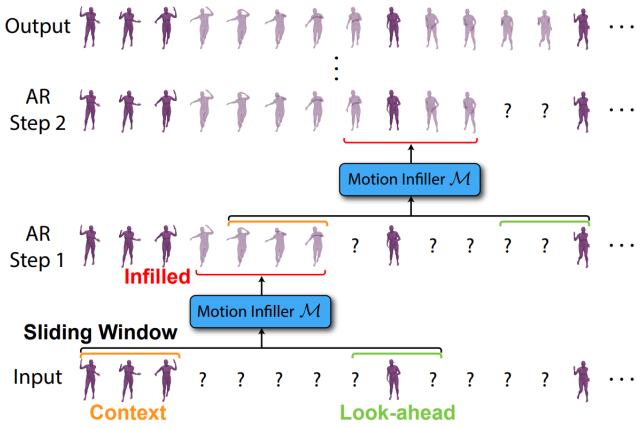


Figura 4.7: *Autoregressive Motion Infilling* [34].

Tal como visível, o método recorre a uma janela deslizante com H *frames*, onde a pose dos h_c *frames* é conhecida (ou anteriormente preenchida), atuando como contexto (*context*) enquanto os últimos h_l *frames* atuam como *look-ahead*, uma guia para a posição final na janela deslizante definida. Neste cenário serão inferidos h_o *frames*, *i.e.*, $h_o = h - h_c - h_l$, exatamente o número de *frames* que devem ser avançados na janela deslizante. Para o projeto em causa foi considerada uma janela deslizante $h = 50$, com $h_c = 10$ e $h_l = 10$, resultando num salto $h_o = 30$ *frames*.

A *Generative Motion Infiller Network* (GMI-Net) foi anteriormente mencionada (Secção 2.3.2), relembrando somente a importância da inclusão da *coerência temporal* passível de ser interpretada pelo *encoder* numa variante da arquitetura *Transformer* [106] (*context network*) e o *CVAE decoder* que recorre ao *latent code* $\boldsymbol{z}$ e à informação dos *frames* visíveis para, com base na lógica do *autoregressive motion infilling*, gerar poses sem oclusão $\widehat{\boldsymbol{\Theta}}$ através de um MLP.

Os detalhes concretos da arquitetura da rede não serão incluídos nesta secção, sendo recomendado ao leitor deste relatório a consulta do artigo oficial do projeto GLAMR [34] (Anexo C) para aprofundamento de conhecimentos.

4.4.2 *Global Trajectory Predictor*

O *Global Trajectory Predictor* $\boldsymbol{\mathcal{T}}$ procura preencher as propriedades descritivas do movimento em falta no sistema de coordenadas do mundo, *i.e.*, trajetória global sem oclusão $(\widehat{\boldsymbol{T}}^i, \widehat{\boldsymbol{R}}^i)$ a partir da *body motion* local estimada $\widehat{\boldsymbol{\Theta}}^i$. A formulação deste *global trajectory predictor* é igualmente definida com base um *CVAE decoder* (para capacitar o sistema de uma modelação condicional no processo de preenchimento da trajetória) com um *latent code* $\boldsymbol{v}$ para uma *body motion* sem oclusão $\widehat{\boldsymbol{\Theta}} = (\widehat{\boldsymbol{\theta}}_1, \ldots, \widehat{\boldsymbol{\theta}}_H)$:

$$\boldsymbol{\Psi} = \boldsymbol{\mathcal{T}}(\hat{\boldsymbol{\Theta}}, \boldsymbol{v}), \tag{4.16}$$

$$(\boldsymbol{T}, \boldsymbol{R}) = \texttt{EgoToGlobal}(\boldsymbol{\Psi}), \tag{4.17}$$

sendo a saída do *global trajectory predictor* $\boldsymbol{\mathcal{T}}$ dada por uma representação alternativa da *global trajectory* sem oclusão $(\hat{\boldsymbol{T}}^i, \hat{\boldsymbol{R}}^i)$, *i.e.*, ***egocentric trajectory*** $\boldsymbol{\Psi} = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_H)$, posteriormente convertida para as variáveis requeridas através de uma função `EgoToGlobal`, *i.e.*, *root translations* $\boldsymbol{T} = (\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_H)$ e *root rotations* $\boldsymbol{R} = (\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_H)$.

Esta *egocentric trajectory* otimiza o processo de previsão de trajetória em longos intervalos de tempo, uma vez que a rede neuronal apenas necessita de gerar a mudança (variação) de trajetória local para cada *frame*, em vez de uma potencial variação de trajetória global acrescida. Esta variação de trajetória local representa a rotação e translação em *heading coordinates*[4], permitindo que a trajetória estimada seja invariante à orientação e translação absoluta da pessoa no plano xy.

Em particular, num determinado *frame* h, a *egocentric trajectory* $\boldsymbol{\psi}_h = (\delta x_h, \delta y_h, z_h, \delta\eta_h)$ é calculada por:

$$\boldsymbol{\psi}_h = (\boldsymbol{\delta x}_h, \boldsymbol{\delta y}_h, \boldsymbol{z}_h, \boldsymbol{\delta\eta}_h)$$

$$(\delta x_h, \delta y_h) = \texttt{ToHeading}\left(\boldsymbol{\tau}_h^{xy} - \boldsymbol{\tau}_{h-1}^{xy}\right), \quad z_h = \boldsymbol{\tau}_h^z, \tag{4.18}$$

sendo esta variação dada pela diferença da componente xy da translação $\boldsymbol{\tau}_h$ em dois tempos consecutivos ($\boldsymbol{\tau}_{h-1}$ e $\boldsymbol{\tau}_h$) enquanto a componente $\boldsymbol{z}$ (altura) se mantém constante uma vez que a altura da pessoa em relação ao chão não varia consideravelmente, estando efetivamente correlacionada com o movimento corporal da mesma,

$$\eta_h = \texttt{ToHeading}\left(\boldsymbol{\gamma}_h\right), \tag{4.19}$$

$$\delta\eta_h = \eta_t - \eta_{h-1}, \tag{4.20}$$

enquanto η_h corresponde ao *heading angle* da rotação $\boldsymbol{\gamma}_h$ em torno do eixo vertical e no plano horizontal em causa. Ainda assim, fica pendente o metodologia de inicialização das variáveis $(\delta x_0, \delta y_0)$ e η_0. Concretamente, durante a inferência foram inicializadas aleatoriamente (uma vez que a trajetória poderá começar em qualquer posição ou orientação). Contudo, para correção de potenciais desvios na trajetória, na Subsecção 4.4.3, a *global trajectory* será otimizada (Equação 4.26)

[4] As *heading coordinates*, obtidas a partir da aplicação da função `ToHeading` à translação ou rotação, são definidas colocando o centro de coordenadas do mundo na *root position* da pessoa e rodando o "mundo" em torno do eixo z para alinhar o eixo y com a orientação facial do humano (*heading vector*) [34].

para, juntamente com as evidencias presentes no vídeo, produzir a *global motion* $\breve{\boldsymbol{Q}}^i$.

Tal como mencionado anteriormente, a *Global Trajectory Predictor Network* (GTP-Net) adota uma arquitetura similar à GMI-Net com diferença na modelação temporal da informação realizada por LSTMs em vez de *Transformers*. Esta decisão deve-se ao facto da mudança de trajetória local (*egocentric representation*) no humano ser verificada em *frames* próximos, não exigindo uma consideração temporal de longo alcance.

De notar, que apesar da nomenclatura evidenciar a utilização das rotações relativas das articulações $\widehat{\boldsymbol{\Theta}}$, na entrada da GTP-Net foi utilizada as posição 3D das articulações (localmente, sem consideração de $\widehat{\boldsymbol{T}}$ ou $\widehat{\boldsymbol{R}}$) pela conversão prévia recorrendo às funcionalidades do SMPL.

4.4.3 *Global Optimization*

Após a aplicação do *generative motion infiller* e *global trajectory predictor* decorre um processo de otimização analítico da trajetória global de cada humano (Equação 4.25 e 4.26) e dos parâmetros extrínsecos da câmara (Equação 4.27), os quais devem ser relacionados com as evidências obtidas do vídeo (*e.g.*, *2D keypoints* a partir do termo $E_{\texttt{2D}}$). O resultado final desta otimização são as *global motions* $\breve{\boldsymbol{Q}}^i = (\breve{\boldsymbol{T}}^i, \breve{\boldsymbol{R}}^i, \breve{\boldsymbol{\Theta}}^i, \breve{\boldsymbol{B}}^i)$ onde $(\breve{\boldsymbol{\Theta}}^i, \breve{\boldsymbol{B}}^i) = (\widehat{\boldsymbol{\Theta}}^i, \widehat{\boldsymbol{B}}^i)$, obtidos diretamente das técnicas anteriores.

Com este intuito, são evidenciadas dois conjuntos de variáveis passíveis de serem otimizadas. A primeira remete para a *egocentric trajectory*[5] $\{\breve{\boldsymbol{\Psi}}^i\}_{i=1}^L$, relativamente à trajetória global $\{(\breve{\boldsymbol{T}}^i, \breve{\boldsymbol{R}}^i)\}_{i=1}^L$, permitindo a otimização da trajetória mesmos em poses ocultas (anteriormente inferidas), com um impacto considerável na trajetória resultante. Por sua vez, os parâmetros da câmara $\boldsymbol{C} = (\boldsymbol{C}_1, \ldots, \boldsymbol{C}_H)$, onde $\boldsymbol{C}_h \in \mathbb{R}^{4\times4}$, constituem o segundo conjunto de variáveis a otimizar nesta implementação.

Formalizando o pressuposto, os autores do GLAMR [34] desenvolveram uma ***Energy Function***[6] que promove esta otimização e a qual se procura minimizar:

$$E\left(\left\{\breve{\boldsymbol{\Psi}}^i\right\}_{i=1}^{L}, \boldsymbol{C}\right) = \lambda_{\texttt{2D}}E_{\texttt{2D}} + \lambda_{\texttt{traj}}E_{\texttt{traj}} + \lambda_{\texttt{reg}}E_{\texttt{reg}} + \lambda_{\texttt{cam}}E_{\texttt{cam}}, \tag{4.21}$$

controlados por quatro coeficientes de perda ($\lambda_{\texttt{2D}}, \lambda_{\texttt{traj}}, \lambda_{\texttt{reg}}$ e $\lambda_{\texttt{cam}}$) do respetivo termo (*energy term*).

[5] Esta representação permite a correção no *frame* em causa para que esta se propague nos futuros *frames*.

[6] Os autores da função de otimização consideraram um quinto termo $E_{\texttt{pen}}$ que não terá sido incorporado nesta implementação.

No que diz respeito ao primeiro termo, $E_{\texttt{2D}}$, quantifica-se o erro entre a projeção 2D $\breve{\boldsymbol{x}}_h^i$ (provenientes dos *keypoints* 3D $\breve{\boldsymbol{X}}_h^i \in \mathbb{R}^{K\times 3}$ otimizados, Equação 4.23) e os *keypoints* 2D $\tilde{\boldsymbol{x}}_h^i$ proveniente do *HPS estimator*[7]:

$$E_{\texttt{2D}} = \frac{1}{LHK}\sum_{i=1}^{L}\sum_{h=1}^{H} V_h^i \left\|\breve{\boldsymbol{x}}_h^i - \tilde{\boldsymbol{x}}_h^i\right\|_F^2, \tag{4.22}$$

$$\breve{\boldsymbol{x}}_h^i = \Pi\left(\breve{\boldsymbol{X}}_h^i, \boldsymbol{C}_h, \boldsymbol{K_{cam}}\right), \quad \breve{\boldsymbol{X}}_h^i = \mathcal{J}\left(\breve{\boldsymbol{\tau}}_h^i, \breve{\boldsymbol{\gamma}}_h^i, \breve{\boldsymbol{\theta}}_h^i, \breve{\boldsymbol{\beta}}_h^i\right), \tag{4.23}$$

onde V_h^i identifica o estado de visibilidade da pessoa i no *frame* h, $\|\cdot\|_F$ remete para norma matricial de Frobenius, Π simboliza a projeção da câmara com os parâmetros extrínsecos C_h e os parâmetros intrínsecos K_{cam}, e os *keypoints* 3D $\breve{\boldsymbol{X}}_h^i$ calculados a partir da função SMPL $\mathcal{J}$ com base na pose global otimizada $\breve{\boldsymbol{q}}_h^i = \left(\breve{\boldsymbol{\tau}}_h^i, \breve{\boldsymbol{\gamma}}_h^i, \breve{\boldsymbol{\theta}}_h^i, \breve{\boldsymbol{\beta}}_h^i\right) \in \breve{\boldsymbol{Q}}^i$.

Durante a inferência, para imagens com dimensões $[\texttt{w}, \texttt{h}]$ foi considerada a distância focal $f_x = f_y = 1000\ mm$ e ponto central (*principal point*) $[c_x, c_y] = [\texttt{w}/2, \texttt{h}/2]$ o centro exato da imagem para aproximação à matriz dos parâmetros intrínsecos K_{cam}:

$$K_{cam} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.24}$$

De notar que durante o processo de otimização, os parâmetros intrínsecos K_{cam} foram mantidos fixos, enquanto durante os *benchmarks* (Secção 5.3.3), foram considerados os parâmetros definidos no sistema visual do *dataset* em causa. No que diz respeito aos parâmetros extrínsecos C_h desconhecidos, estes foram inicializados a partir da trajetória global para pessoas visíveis[8].

O segundo termo, E_{traj}, mede a diferença entre a trajetória global otimizada $(\breve{\boldsymbol{T}}^i, \breve{\boldsymbol{R}}^i)$ vista no sistema de coordenadas da câmara e a *root trajectory* $(\tilde{\boldsymbol{T}}^i, \tilde{\boldsymbol{R}}^i)$ obtida na saída do *HPS estimator*:

$$E_{\text{traj}} = \frac{1}{LH}\sum_{i=1}^{L}\sum_{h=1}^{H} V_h^i \left(\left\|\Gamma\left(\breve{\boldsymbol{\gamma}}_h^i, \boldsymbol{C}_h\right) \ominus \tilde{\boldsymbol{\gamma}}_h^i\right\|_a^2 + w_h \left\|\Gamma\left(\breve{\boldsymbol{\tau}}_h^i, \boldsymbol{C}_h\right) - \tilde{\boldsymbol{\tau}}_h^i\right\|_2^2\right), \tag{4.25}$$

onde a função $\Gamma(\cdot, \boldsymbol{C}_h)$ transforma a rotação $\tilde{\boldsymbol{\gamma}}_h^i$ ou translação $\breve{\boldsymbol{\tau}}_h^i$ para o sistema de coordenadas da câmara pela matriz C_h e w_h um fator de ponderação para diferença quadrática das translações. Seguindo a nomenclatura dos autores

[7] Os pontos 2D são obtidos de uma transformação projetiva dos pontos 3D, inicialmente estimados a partir dos *volumetric heatmaps*, para o plano da imagem.

[8] A inicialização da matriz dos parâmetros extrínsecos C_h seguiu o processo definido pelos autores do GLAMR [34] (Anexo E do projeto original).

do GLAMR [34], $\ominus$ evidência o cálculo de uma rotação relativa[9], $\|\cdot\|_a$ o cálculo da diferença angular (ângulo de rotação) e $\|\cdot\|_2$ remete norma vetorial Euclidiana.

O terceiro termo, E_{reg}, com efeito corretivo na penalização do anterior termo, procura regularizar a *egocentric trajectory* $\breve{\boldsymbol{\Psi}}^i$ para que esta se aproxime do valor obtido no *global trajectory predictor* $\widehat{\boldsymbol{\Psi}}^i$, obtido pelo processo contrário ao descrito nas Equação 4.17, *i.e.*, função $\widehat{\boldsymbol{\Psi}}^i = \texttt{GlobalToEgo}(\widehat{\boldsymbol{T}}^i, \widehat{\boldsymbol{R}}^i)$:

$$E_{\text{reg}} = \frac{1}{LH} \sum_{i=1}^{L} \sum_{h=1}^{H} \left\| \boldsymbol{w}_\psi \circ \left(\breve{\boldsymbol{\psi}}_h^i - \widehat{\boldsymbol{\psi}}_h^i \right) \right\|_2^2, \tag{4.26}$$

onde $\circ$ remete para um produto elemento a elemento e $\boldsymbol{w}_\psi$ um vetor de ponderação para diferença quadrática de cada elemento das *egocentric trajectories*.

O último termo desta otimização, E_{cam} mede a suavidade da variação dos parâmetros extrínsecos da câmara C_h:

$$E_{\text{cam}} = \frac{1}{H-1} \sum_{h=1}^{H-1} \left\| \boldsymbol{C}_{h+1}^{\gamma} \ominus \boldsymbol{C}_h^{\gamma} \right\|_a^2 + \left\| \boldsymbol{C}_{h+1}^{\tau} - \boldsymbol{C}_h^{\tau} \right\|_2^2, \tag{4.27}$$

sendo que $\boldsymbol{C}_h^{\gamma}$ e $\boldsymbol{C}_h^{\tau}$ remetem para a rotação e translação presentes na matriz dos parâmetros extrínsecos da câmara C_h. Tal como referido na nota em rodapé 8, C_h sofre uma inicialização baseada nas trajetórias globais das pessoas visíveis (se uma pessoa estiver oculta são utilizados os parâmetros extrínsecos C_h do *frame* mais recente em que esta estava visível) , como tal, o primeiro elemento não deve ser contabilizado neste termo de otimização pelo facto de estar sujeito a um processo de obtenção diferente.

Finalizando esta secção, durante a inferência, foi atribuído aos coeficientes de perda ($\lambda_{\texttt{2D}}, \lambda_{\texttt{traj}}, \lambda_{\texttt{reg}}$ e $\lambda_{\texttt{cam}}$), Equação 4.21, os valores $(1, 1\times10^5, 1\times10^2, 1\times10^5)$. Relativamente ao fator de ponderação w_h (Equação 4.25) foi considerado o valor 0 pois foi verificado que a translação estimada pelo *HPS estimator* apresentava bastante ruído e instabilidade, pelo que a sua consideração não era benéfica no processo de otimização. Por sua vez, no vetor de ponderação $\boldsymbol{w}_\psi$ (Equação 4.26) foram definidos os valores $(3\times10^2, 3\times10^2, 1\times10^4, 3\times10^3)$ para cada elemento da *egocentric trajectory* $\boldsymbol{\psi}_h = (\delta x_h, \delta y_h, z_h, \delta\eta_h)$ com uma penalização superior nas variações de altura z_h. No que diz respeito ao processo de minimização da função de perdas (Equação 4.21), este terá sido realizado com base no *Adam optimizer* segundo um *learning rate* de 1×10^{-3}. Toda a implementação foi realizada em PyTorch, enquanto a projeção no mundo das sequências de malhas triangulares de cada humano assim como a posição da câmara dinâmica foi realizada com recurso à biblioteca PyVista. Contudo, para mais detalhes sobre as bibliotecas e *frameworks* utilizadas recomenda-se a análise do Anexo F.

[9] A diferença angular é realiza a partir de uma representação 6D [146] da rotação efetiva, previamente convertida.

Esta página foi propositadamente deixada em branco

Capítulo 5

Resultados Experimentais

O presente capítulo reúne um conjunto de testes e resultados realizados durante o desenvolvimento e complementares à análise da implementação do Capítulo 4. A Secção 5.1 apresenta, de forma sucinta, os *datasets* utilizados assim como as métricas para a avaliação da solução desenvolvida. A viabilidade do MPT adotado é comprovado, qualitativamente, na Secção 5.2, enquanto na Secção 5.3 é explorado o comportamento do sistema ao lidar cenários dinâmicos e o efeito otimizado do *generative motion optimizer* com as métricas previamente consideradas.

5.1 Métricas de Avaliação e Detalhes dos *Datasets*

5.1.1 *Datasets*

Durante o desenvolvimento foram recorrentemente utilizados os seguintes *datasets*:

1. **AMASS** [88]: trata-se de um *MoCap dataset* com mais de 11000 movimentações captadas em humanos. Estes conjuntos de dados foram utilizados durante a fase de treino, previamente realizada, do *motion infiller* e *trajectory predictor* para que cada modelo da rede neuronal pudesse replicar a dinâmica humana aprendida em cenários desconhecidos;

2. **MPI-INF-3DHP** [147]: consiste num *dataset* que captura a movimentação individual de 8 humanos (4 homens e 4 mulheres) num estúdio com tela verde e no exterior através de um sistema configurado com de várias câmaras. A

sua utilização destinou-se ao treino do *backbone* do *HPS estimator* (HRNet-W48 [40]) e testes qualitativos;

3. **3DPW** [103]: trata-se de um *dataset* com 60 vídeos que capta um conjunto de movimentos de humanos em ambientes reais (*in-the-wild*). A precisão das poses 3D verdadeiras é conseguida através do posicionamento de diferentes IMUs (*wearable IMUs*), mesmo quando a pessoa está oculta (*e.g.*, sai do FoV da câmara). Assim sendo, o 3DPW também fornece a trajetória global das pessoas no *dataset*, sendo a componente de teste utilizadas durante a fase de avaliação do método. Não obstante, o treino do *backbone* do *HPS estimator* (HRNet-W48) foi baseado no *dataset* 3DPW. Parte dos resultados qualitativos recorreram a este *dataset*;

4. **Dynamic 3DPW**: trata-se de um novo *dataset*, criado pelo autor deste trabalho a partir do 3DPW [103], que procura adicionar uma componente dinâmica aos vídeos que apresentavam pouca ou nenhuma variação horizontal na posição da câmara. Para tal, realiza-se um recorte de cada *frame* numa pequena janela de visualização de 516×1350 que oscila horizontalmente, com incrementos de 12 pixeis, em torno do centro da *bounding box* da pessoa em causa. Deste modo, é promovido um aumento no número de oclusões do humano no cenário, revelando-se um desafio acrescido para a solução proposta.

5.1.2 Métricas de Avaliação

Por sua vez, a avaliação da performance da solução proposta foi baseada nas seguintes métricas:

1. ***Procrustes-Aligned MPJPE* (PA-MPJPE)**: anteriormente referida na Subsecção 4.3.3, dividiu-se em amostras visíveis e invisíveis. Contudo, dada a natureza probabilística do *CVAE decoder* no *motion infiller* e à quantidade de poses passíveis de serem inferidas quando o MPT falha na deteção do humano, mantivemos a abordagem do GLAMR [34] a fim de calcular a melhor PA-MPJPE;

2. **G-MPJPE** e **G-PVE**: estas métricas procuram estender o *Mean Per Joint Position Error* (MPJPE) e *Per-Vertex Error* (PVE), respetivamente, calculando estes erros nas coordenadas globais. Contudo, tratando-se de um método de reconstrução em malha aberta (sem *feedback* de trajetória temporalmente definido) que evidenciará acumulação de erro, os autores do GLAMR [34] adotaram um processo de avaliação padronizado em métodos de navegação (*e.g.*, *inertial odometry* e SLAM) baseado numa janela temporal. Como tal, a cada 10 segundos a *root translation* e *root rotation* é alinhada com os valores definidos na *ground truth*;

3. **Accel**: permite calcular o erro médio de aceleração de cada articulação, tipicamente utilizado para parametrizar o *jitter* nas poses estimadas. É medida em $mm/frame^2$ para poses 3D e $pixel/frame^2$ para poses 2D.

5.2 *Multiple Person Tracking*

Tendo por base a proposta apresentada na Secção 4.2, segue-se uma análise qualitativa do primeiro módulo da arquitetura do sistema (Figura 4.1). O objetivo será justificar, a seleção e a robustez do Deep OC-SORT com YOLOv8x e *Re-ID model* (ResNet-50 [60]) na deteção e *tracking* de humanos.

Para este teste optou-se pela escolha de um *dataset* com um conjunto de pessoas acrescido, num cenário consideravelmente complicado e com diversos momentos de oclusão. Desta forma, recorrendo ao *dataset* de teste do 3DPW [103], concretamente ao conjunto de 731 *frames* `downtown_runForBus_00`, com uma confiança de 0,60 (valor típico utilizado pela comunidade), um IoU *threshold* de 0,3 (necessário para o processo de associação), restrição no *tracking* exclusiva da classe *"person"* e o mesmo *Re-ID model* foi possível obter, sem qualquer tipo de filtragem, os resultados da Tabela 5.1. De notar que os tempos de inferência médios referidos foram obtidos de um ajuste à imagem de entrada para a resolução 384×640.

Tabela 5.1: Deep OC-SORT em diferentes detetores de humanos com o mesmo *Re-ID model* (ResNet-50) no *dataset* 3DPW.

Detector	Inference time (ms)	Tracking time (ms)	IDs
YOLOX-S	50,2	42,6	140
YOLOX-L	104,6	103,5	340
YOLOv8-S	17,8	13,9	49
YOLO-NAS-S	73,9	33,5	110
YOLOv8-L	75,8	13,9	49
YOLO-NAS-L	107,3	38,7	127
YOLOv8-X	85,7	16,3	60

O tempo de inferência superior em modelos de deteção maiores (*e.g.*, YOLOv8-X, YOLOX-L e YOLO-NAS-L) é compreensível fruto da sua complexidade e quantidade de parâmetros que regem o seu comportamento. É notório um aumento do tempo de *tracking* nos modelos que geram um maior número de identificadores. Ainda assim, o valor de confiança definido interferiu consideravelmente na enumeração dos humanos detetados. Apesar da solução global deste projeto beneficiar de uma técnica complementar às deteções erráticas do MPT, não será de todo interessante apostar num modelo instável (*e.g.*, YOLOX gera um número de deteções acrescida que tendem a prejudicar o *tracker*), pelo que neste trabalho será adotado o detetor YOLOv8-X. A Figura 5.1 apresenta o resultado do MPT segundo as configurações referidas.

Figura 5.1: Resultados qualitativos do MPT Deep OC-SORT com YOLOv8x e *Re-ID model* ResNet-50 no *dataset* 3DPW [103]. Linha superior remete o *tracking* de todos os humanos detetados. A linha inferior contempla uma filtragem aos humanos (ID 1 e 2) utilizados na avaliação do modelo final do modelo.

Complementarmente, no Anexo A, encontram-se um conjunto de situações, comparativas entre os vários *trackers* referidos, que evidenciam a atribuição errada de IDs ou fragmentação na atribuição dos mesmos. Nestes cenários o Deep OC-SORT mantém-se estável e com um número reduzido número de identificadores falsos.

5.3 Avaliação do Método Proposto

Esta secção apresenta resultados visuais e gráficos relativos à solução desenvolvida, validando a capacidade de previsão de movimento em coordenadas globais, para longos intervalos com oclusão, a partir de câmaras dinâmicas e a suavidade (Subsecção 5.3.1) na variação posicional do esqueleto humano ao longo do vídeo (rotações relativas $\breve{\boldsymbol{\Theta}}^{i}$ inicialmente estimadas pelo algoritmo do *Adaptive HybrIK*).

Os testes foram realizados maioritariamente nos *datasets* 3DPW [103] e *Dynamic 3DPW* (Subsecção 5.3.2), fruto da elevada precisão na obtenção rotações relativas verdadeiras de cada articulação em cada humano. É então possível validar a performance do *generative motion optimizer*, mantendo os parâmetros intrínsecos fixos e, iterativamente, corrigindo os parâmetros extrínsecos, inicialmente desconhecidos, para reconstrução das relações espaciais entre os humanos no vídeo.

Posto isto, na Subsecção 5.3.3 são expostos os resultados das métricas referidas (PA-MPJPE, G-MPJPE, G-PVE e Accel) aplicadas ao *dataset* 3DPW, demonstrando a capacidade global do sistema em reconstruir o movimento humano e produzir variações posicionais (ainda que visíveis no FoV da câmara) mais suaves.

5.3.1 Análise da Suavidade do Movimento

Para uma análise completa à suavidade do movimento gerado no final do sistema proposto e da capacidade em inferir poses, quando o MPT deixa de conseguir detetar o humanos, foi utilizado o conjunto de 783 *frames* `downtown_runForBus_01` proveniente do *dataset* de teste do 3DPW [103]. Concretamente, este *dataset* reúne diversas situações, num cenário não controlado, em que os humanos deixam de ser detetados por oclusão ou por saírem do FoV da câmara.

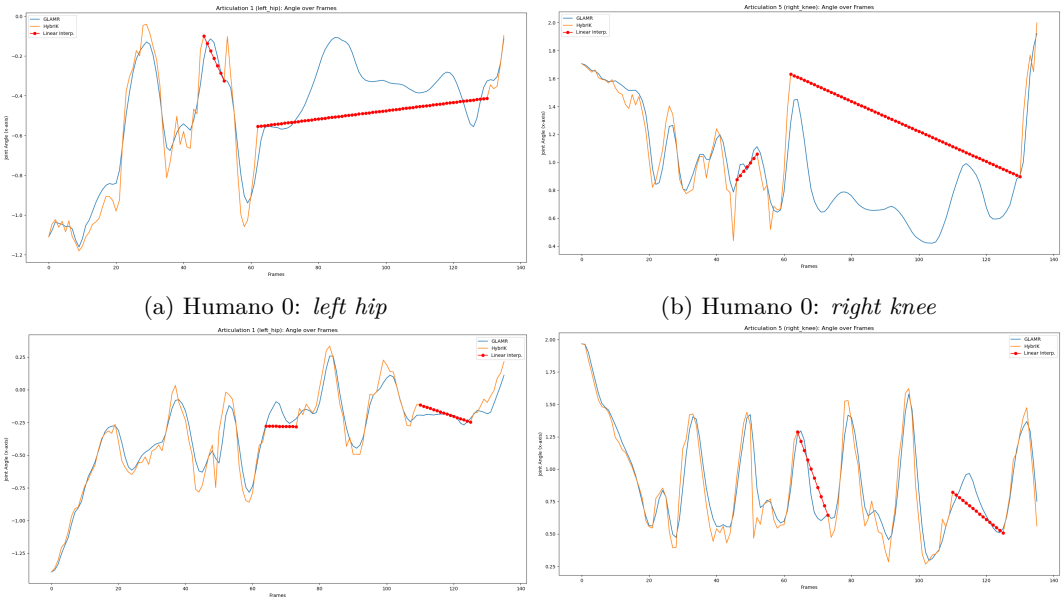
Com foco no intervalo de *frames* 485 a 555, representando cerca de 2,3 segundos do *dataset* completo e uma filtragem do MPT (configuração referida na Subsecção 5.2) nos IDs 0 (verde) e 1 (roxo), Figura 5.2 (linha superior), foram geradas do *Adaptive HybrIK* as respetivas malhas, devidamente projetadas para o plano da imagem, Figura 5.2 (linha intermédia).

Figura 5.2: Resultados qualitativos gerados pelos três módulos da solução proposta no intervalo de *frames* 485 a 555 no *dataset* 3DPW [103]. Primeira linha reune as *bounding boxes* geradas pelo MPT e as respetivas poses no plano da imagem. A segunda linha inclui a malha projetada nos instantes em que o humano terá sido detetado. A última linha apresenta o resultado final da otimização, com a malha (visível e preenchida) no sistema de coordenadas do mundo (centro definido pelos 3 eixos no chão) e possível posição da câmara (esfera amarela).

O sincronismo na falha de atribuição da *bounding box* e na obtenção da malha corporal é notório e, como tal, cabe ao *motion infiller* fazer a inicialização da janela deslizante (Figura 4.7) a partir dos *frames* iniciais com as poses do corpo conhecidas (*context*) enquanto os últimos *frames* atuam como uma guia para a posição final (*look-ahead*).

Efetivamente, o movimento preenchido (transparente) durante o *generative motion optimizer* revela-se mais natural especialmente para as pernas. Esta afirmação é passível de ser interpretada graficamente a partir da Figura 5.3. Nesta, encontra-se exposta a variação angular, que descreve a posição de cada articulação, ao longo dos *frames* avaliados.

Na primeira linha, Figura 5.3a e5.3b, referente ao humano 0 (verde), é realizada uma interpolação linear (linha a vermelho definida por pontos) para atenuar a variação posicional no algoritmo do *Adaptive HybrIK* na próxima aparição do sujeito (face a anteriores soluções que adotam a última posição). Contudo, esta solução não



(a) Humano 0: *left hip*

(b) Humano 0: *right knee*

(c) Humano 1: *left hip*

(d) Humano 1: *right knee*

Figura 5.3: Variação angular das articulações dos humanos avaliados na Figura 5.2. A amarelo as poses estimadas no final do *Adaptive HybrIK*. A vermelho, numa linha com pontos, a interpolação linear das poses nos *frames* com deteções em falta. A azul o resultado da otimização do método adotado (GLAMR [34]).

leva em consideração a dinâmica do movimento humano aprendida pelo *generative motion optimizer*.

Apesar do aumento de precisão e menor erro durante a inferência conseguidas pela adoção do novo *backbone* na rede do *HPS estimator* (HRNet-W48 [40]), é percetível a atenuação considerável na variação angular após otimização. Sinalizado com linha amarela, as rotações relativas das articulações referidas (*left hip* e *ight knee*) apresentam um *jitter* superior, com picos visualmente não esperados e atenuados nas novas poses geradas (linha azul). Isto acontece por o *motion infiller* aprendeu a dinâmica humana a partir de um grande *dataset* (AMASS [88]).

5.3.2 Câmara Dinâmica no 3DPW

Para aumento da componente dinâmica no *dataset* 3DPW e aumento no número de oclusões do humano no cenário, foi proposto um algoritmo capaz de impor uma variação horizontal na posição da câmara, segundo uma oscilação horizontal em torno do centro da *bounding box* da pessoa presente. O *dataset* foi apelidado de *Dynamic 3DPW*. Processamento realizado com recurso à biblioteca Pillow[1].

A Figura 5.4 apresenta este condicionamento visual ao longo de um intervalo de *frames* do *dataset* `courtyard_basketball_01`. Na segunda linha, podemos observar o sincronismo entre as poses estimadas e movimento reconstruido.

[1]Detalhes concretos das bibliotecas e *frameworks* utilizadas descrito no Anexo F.

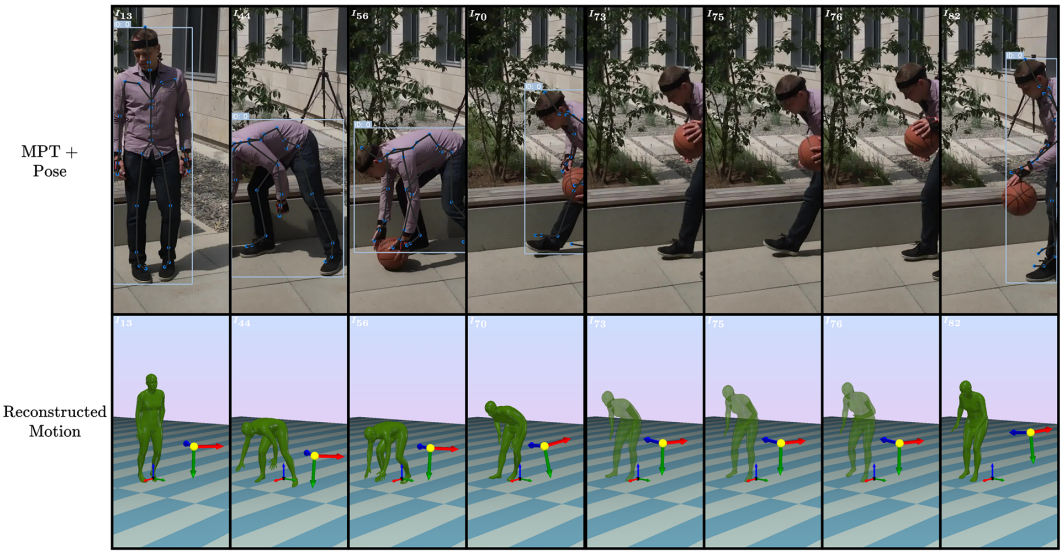


Figura 5.4: Resultado qualitativo do sistema no *Dynamic 3DPW* proposto. Procura-se aumentar os momentos de oclusão para validação efetiva da capacidade no preenchimento do movimento do *generative motion optimizer* face à inferência do *Adaptive HybrIK*. Novamente, é visível uma variação posicional suavizada e temporalmente coerente com base na informação que preenche a janela do *autoregressive motion infilling*.

Destaca-se novamente a imposição de um movimento natural, focando desta vez a variação posicional da mão esquerda entre o *frame* 70 e o *frame* 83, pelo que, contrariamente a outras soluções, não mantém o sujeito estático (fixo na última posição visível) ou aplica uma interpolação linear, mas recorre ao conhecimento da dinâmica do humano aprendido para projetar uma possível variação na árvore cinemática.

5.3.3 Comparação com SOTA

Os resultados da execução no *dataset* 3DPW encontram-se quantitativamente expostos na Tabela 5.2. Durante este processo foram avaliados dois humanos (ou apenas um) em determinados conjuntos de imagens, conforme definido pelo protocolo do *dataset*, a fim de aferir a qualidade dos resultados gerados pelo método proposto na reconstrução do movimento ao longo do tempo.

Importa realçar que esta avaliação recorreu da informação de *tracking* disponibiliza pelo *dataset*. Tendo isto em conta, podemos excluir a propagação errática de deteções dos humanos, avaliando unicamente a capacidade de reconstrução e preenchimento de movimento.

Posto isto, fica claro o benefício desta técnica de otimização, superando métodos anteriores suportados por interpolações lineares ou ConvAE [148] (*Convolutional autoencoders* para preenchimento do movimento humano). Destaca-se os resultados

Tabela 5.2: Comparação com métodos SOTA no *dataset* 3DPW. Informação obtida para os *frames* com oclusão (*invisible*), sem oclusão (*visible*) e todos (*all*). Resultados do método proposto na última linha da tabela com o novo *backbone* HRNet-W48 [40].

Method	**(Invisible) PA-MPJPE↓**	**(Visible) PA-MPJPE↓**	**(All) Accel↓**
KAMA [32] + Linear Interpolation	87,5	50,8	24,2
KAMA [32] + Last Pose	96,3	50,8	25,4
KAMA [32] + ConvAE [148]	84,5	56,4	19,6
SPEC [33] + Linear Interpolation	85,6	53,3	33,1
SPEC [33] + Last Pose	92,4	53,3	34,2
SPEC [33] + ConvAE [148]	86,9	59,3	24,0
GLAMR w/ SPEC [33]	79,1	54,9	9,5
GLAMR w/ KAMA [32]	73,6	51,1	**8,9**
GLAMR w/ HybrIK (solução proposta)	**68,2**	**46,4**	10,6

obtidos em PA-MPJPE em pessoas ocultas (*invisible*) que, apesar da pequena redução de erro, justifica-se pelo aumento de precisão conseguido no *HPS estimator*, nos *frames* com o humano visível e, naturalmente, pela capacidade de preenchimento de movimento do modelo probabilístico GLAMR [34] (proximidade aos movimentos verdadeiros baseada na dinâmica humana aprendida). O mesmo já não é notório no parâmetro da aceleração (Accel), sendo, contudo, superior aos anteriores métodos, que não recorrem ao GLAMR, mas compõe o SOTA.

Ainda assim, apesar de não serem conhecidos noutros modelos, para o *dataset* em causa, os valores de G-MPJPE e G-PVE, foram obtidos, nos testes realizados (*dataset* 3DPW), respetivamente 763,4 *mm* e 778,1 *mm*. No artigo oficial de Yuan *et al.* [34] foram registados, no *dataset* Human3.6M [104], respetivamente 899,1 *mm* e 913,7 *mm* para o método GLAMR combinado com SPEC [33], enquanto no método GLAMR combinado com KAMA [32] 806,2 *mm* e 824,1 *mm*. É, portanto, possível prever, dada a natureza complexa dos dados que compõe o *dataset* 3DPW, uma melhoria performance quando inferido no Human3.6M.

De notar que os resultados terão sido obtidos pela execução numa única GPU NVIDIA GeForce GTX 1650 para um CPU Intel Core i5-9300H no sistema operativo Ubuntu 20.04.6 LTS. Como tal, o tempo médio de inferência por *frame* e por pessoa aproxima-se dos 333 *ms* (3 fps) no *HPS estimator*. O mesmo cálculo não terá sido realizado no *generative motion optimizer*, uma vez que se trata de um processo de otimização, não sendo atualmente o objetivo a execução em tempo real.

Complementarmente, é feita uma análise sucinta aos resultados qualitativos presente em anexo (Anexo B a D) evidenciando a capacidade de reconstrução de movimento e *tracking* de pessoas em ambientes reais.

Capítulo 6

Conclusão

Ao longo deste projeto foi desenvolvida uma abordagem focada na obtenção da posição articular 3D dos humanos detetados em cenários reais, bem como a sua forma corporal com recurso a um sistema monocular. O método destaca-se pela incorporação de ferramentas promissoras (*e.g.*, *Transformers* e LSTMs) e pela recuperação da trajetória global de cada humano de forma otimizada.

A solução foi dividida em três módulos, maximizando a performance do algoritmo que os define. A deteção dos humanos, ao longos dos *frames* do vídeo, foi realizada com recurso a um MPT, *i.e.*, Deep OC-SORT [139], e com um *Re-ID model*, devidamente incorporados e aptos a lidar com as não linearidades do movimento humano, apresentando uma resposta passível de execução em tempo real.

A saída do MPT deve alimentar o *HPS estimator*, desenvolvido com base numa solução analítico-neural de cinemática inversa, *i.e.*, HybrIK [23], adotando o mais recente *backbone* HRNet-W48 [40, 145] que minimiza as incoerências estruturais humanas e promove a determinação dos elementos que caracterizam o modelo paramétrico SMPL [19]. Este sistema híbrido propôs a combinação dos conhecimentos analíticos predominantemente descritivos do movimento humano no espaço com propriedades visuais passíveis de serem processadas na flexibilidade de uma rede neuronal para produção de uma malha corporal humana e localização indireta das articulações 3D, no sistema de coordenadas da câmara.

Definidas as vulnerabilidades de sistemas análogos, sucedeu-se um processo de otimização da trajetória do humano detetado para lidar com situações de oclusão e

dificuldade na correção do movimento globalmente definido. Posto isto, uma arquitetura baseada em *Transformers* fundamentou a base do sistema para preenchimento do movimento humano (*generative motion infiller*) nos *frames* com falta de informação, permitindo a incorporação de uma metodologia autorregressiva regrada pela coerência temporal. Por sua vez, a ambiguidade na reconstrução da trajetória humana, estimada no sistema de coordenadas da câmara, terá sido ultrapassada pela incorporação de um módulo capaz de lidar com as variações locais do movimento do corpo e gerar trajetórias globais (*global trajectory predictor*). Ainda que o movimento dos humanos esteja globalmente definido, foi necessário correlacioná-los, segundo um processo de otimização analítico, minimizando uma função de perdas.

O método em causa apresentou resultados notáveis, quando comparado diretamente com as soluções que compõe a base deste projeto (*i.e.*, GLAMR [34] com *HPS estimator* KAMA [32] ou SPEC [33]), evidenciando a robustez dos *Transformers* e LSTMs no que à análise temporal em visão computacional diz respeito.

6.1 Limitações e Trabalho Futuro

Face ao desenvolvimento realizado e conhecidas as vulnerabilidades que impactam a performance do sistema, são destacadas, nesta secção, as principais intervenções que poderão otimizar os resultados apresentados. Complementarmente, no Anexo E, encontra-se uma discussão detalhada sobre as mesmas.

Realça-se a possibilidade de propagação de erro entre os diferentes constituintes da arquitetura, fruto da natureza iterativa do modelo. Numa abordagem a esta questão, equaciona-se a integração conjunta dos diferentes constituintes do modelo.

A consideração da informação presente no cenário (*e.g.*, representação numa *point cloud*) pode providenciar evidências que permitirão aproximar os pés do humano do chão (*foot contact*) [11].

Poderá ainda ser relevante o desenvolvimento de uma abordagem que procure relacionar os movimentos e trajetórias, analisados individualmente nas primeiras duas fases do *generative motion optimizer*, a fim de obter informação potencialmente complexa entre humanos, *e.g.*, danças ou abraços. Neste sentido, ficou pendente a implementação um termo energético que penalize a *inter-person penetration* e as interseções entre malhas distintas $E_{\texttt{pen}}$ [24, 34].

Finalmente, baseado na análise sistemática, o autor acredita que seria uma mais valia o desenvolvimento e publicação de um artigo científico reunindo os mais recentes métodos focadas na determinação da pose e forma corporal baseados num sistema visual. Em resultado da performance da solução desenvolvida, importa referir que código estará disponível em formato *open-source* no seguinte repositório do GitHub: https://github.com/AndreOliveira00/3D-Pose-and-Shape-Estimation-with-a-Camera-System.git.

Referências

[1] D. Hogg, "Model-based vision: a program to see a walking person," *Image and Vision Computing*, 1983. [Citado na página 1]

[2] C. Zheng, W. Wu, C. Chen, T. Yang, S. Zhu, J. Shen, N. Kehtarnavaz, and M. Shah, "Deep learning-based human pose estimation: A survey," 2022. [Citado nas páginas ix, x, 2, 8, 10, 12, 13, 15, 17, 19, 22, 23, 24, 25, 26, 27, 28, 31, 35 e 37]

[3] G. Bilbeisi, "Survey on 3D Pose and Shape Estimation using Deep Learning," 2019. [Citado na página 2]

[4] A. Weiss, D. Hirshberg, and M. J. Black, "Home 3D body scans from noisy image and range data," *IEEE International Conference on Computer Vision*, 2011. [Citado na página 2]

[5] T. von Marcard, B. Rosenhahn, M. J. Black, and G. Pons-Moll, "Sparse inertial poser: Automatic 3D human pose estimation from sparse IMUs," *Eurographics Symposium on Geometry Processing*, 2017. [Citado na página 2]

[6] T. Yu, J. Zhao, Z. Zheng, K. Guo, Q. Dai, H. Li, G. Pons-Moll, and Y. Liu, "DoubleFusion: Real-Time Capture of Human Performances with Inner Body Shapes from a Single Depth Sensor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. [Citado na página 2]

[7] H. Jiang, J. Cai, and J. Zheng, "Skeleton-aware 3D human shape reconstruction from point clouds," *IEEE International Conference on Computer Vision*, 2019. [Citado na página 2]

[8] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, M. Elgharib, P. Fua, H. P. Seidel, H. Rhodin, G. Pons-Moll, and C. Theobalt, "XNect: Real-time Multi-Person 3D Motion Capture with a Single RGB Camera," *ACM Transactions on Graphics*, 2020. [Citado nas páginas 2, 37 e 44]

[9] W. Jiang, N. Kolotouros, G. Pavlakos, X. Zhou, and K. Daniilidis, "Coherent reconstruction of multiple humans from a single image," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. [Citado nas páginas 2, 3, 40, 41, 78, 82 e 160]

[10] N. Ugrinovic, A. Ruiz, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "Body Size and Depth Disambiguation in Multi-Person Reconstruction from Single Images," *2021 International Conference on 3D Vision*, 2021. [Citado nas páginas 2 e 44]

[11] D. Luvizon, M. Habermann, V. Golyanik, A. Kortylewski, and C. Theobalt, "Scene-Aware 3D Multi-Human Motion Capture from a Single Camera," 2023. [Citado nas páginas x, 2, 43, 44, 46, 77, 132 e 159]

[12] H. Tu, C. Wang, and W. Zeng, "VoxelPose: Towards Multi-camera 3D Human Pose Estimation in Wild Environment," *Lecture Notes in Computer Science*, 2020. [Citado na página 2]

[13] Z. Dong, J. Song, X. Chen, C. Guo, and O. Hilliges, "Shape-aware Multi-Person Pose Estimation from Multi-View Images," *IEEE International Conference on Computer Vision*, 2021. [Citado na página 2]

[14] Y. Zhang, Z. Li, L. An, M. Li, T. Yu, and Y. Liu, "Lightweight Multi-person Total Motion Capture Using Sparse Multi-view Cameras," *IEEE International Conference on Computer Vision*, 2021. [Citado na página 2]

[15] J. Dong, Q. Fang, W. Jiang, Y. Yang, Q. Huang, H. Bao, and X. Zhou, "Fast and Robust Multi-Person 3D Pose Estimation and Tracking From Multiple Views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. [Citado nas páginas 2 e 75]

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM*, 2017. [Citado nas páginas x, 2, 11, 62 e 63]

[17] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. [Citado na página 2]

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. [Citado nas páginas 2, 17, 40, 64 e 71]

[19] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," *ACM Transactions on Graphics*, 2015. [Citado nas páginas xi, 2, 9, 10, 27, 100, 101, 106 e 131]

[20] M. Omran, C. Lassner, G. Pons-Moll, P. Gehler, and B. Schiele, "Neural body fitting: Unifying deep learning and model based human pose and shape estimation," 2018. [Citado nas páginas ix, 2, 3, 28, 29 e 33]

[21] M. Kocabas, N. Athanasiou, and M. J. Black, "Vibe: Video inference for human body pose and shape estimation," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. [Citado nas páginas 2, 3 e 29]

[22] Y. Sun, Q. Bao, W. Liu, Y. Fu, M. J. Black, and T. Mei, "Monocular, One-stage, Regression of Multiple 3D People," *IEEE International Conference on Computer Vision*, 2021. [Citado nas páginas x, 2, 44 e 45]

[23] J. Li, C. Xu, Z. Chen, S. Bian, L. Yang, and C. Lu, "HybrIK: A Hybrid Analytical-Neural Inverse Kinematics Solution for 3D Human Pose and Shape Estimation," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021. [Citado nas páginas ix, xi, 2, 3, 33, 34, 65, 106, 109, 110, 111, 131 e 161]

[24] R. Khirodkar, S. Tripathi, and K. Kitani, "Occluded Human Mesh Recovery," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2022. [Citado nas páginas 2, 41, 65, 132 e 160]

[25] H. Choi, G. Moon, J. K. Park, and K. M. Lee, "Learning to Estimate Robust 3D Human Mesh from In-the-Wild Crowded Scenes," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2022. [Citado nas páginas 2 e 40]

[26] G. Moon, J. Y. Chang, and K. M. Lee, "Camera distance-aware top-down approach for 3D multi-person pose estimation from a single RGB image," *IEEE International Conference on Computer Vision*, 2019. [Citado nas páginas x, 3, 35, 36, 106 e 114]

[27] L. Sigal, A. Balan, and M. J. Black, "Combined discriminative and generative articulated pose and non-rigid shape estimation," *Advances in Neural Information Processing Systems 2007*, 2007. [Citado na página 3]

[28] G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis, "Learning to Estimate 3D Human Pose and Shape from a Single Color Image," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. [Citado nas páginas ix, 3, 28, 29 e 78]

[29] N. Kolotouros, G. Pavlakos, M. Black, and K. Daniilidis, "Learning to reconstruct 3D human pose and shape via model-fitting in the loop," in *IEEE International Conference on Computer Vision*, 2019. [Citado nas páginas 3, 32 e 33]

[30] M. Kocabas, C. H. P. Huang, O. Hilliges, and M. J. Black, "PARE: Part Attention Regressor for 3D Human Body Estimation," *IEEE International Conference on Computer Vision*, 2021. [Citado nas páginas ix, 3, 33, 65 e 78]

[31] J. Zhen, Q. Fang, J. Sun, W. Liu, W. Jiang, H. Bao, and X. Zhou, "SMAP: Single-Shot Multi-person Absolute 3D Pose Estimation," in *Lecture Notes in Computer Science*, 2020. [Citado nas páginas x, 3, 38 e 39]

[32] U. Iqbal, K. Xie, Y. Guo, J. Kautz, and P. Molchanov, "KAMA: 3D Keypoint Aware Body Mesh Articulation," 2021. [Citado nas páginas 3, 41, 130, 132 e 160]

[33] M. Kocabas, C. H. P. Huang, J. Tesch, L. Müller, O. Hilliges, and M. J. Black, "SPEC: Seeing People in the Wild with an Estimated Camera," 2021. [Citado nas páginas 3, 41, 130, 132 e 160]

[34] Y. Yuan, U. Iqbal, P. Molchanov, K. Kitani, and J. Kautz, "GLAMR: Global Occlusion-Aware Human Mesh Recovery with Dynamic Cameras," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2022. [Citado nas páginas x, xi, xii, 3, 5, 41, 42, 43, 44, 100, 101, 115, 116, 117, 118, 119, 120, 121, 124, 128, 130, 132, 154, 160 e 161]

[35] A. Masoumian, H. A. Rashwan, J. Cristiano, M. S. Asif, and D. Puig, "Monocular Depth Estimation Using Deep Learning: A Review," *Sensors*, 2022. [Citado nas páginas 8 e 22]

[36] Y. Chen, Y. Tian, and M. He, "Monocular human pose estimation: A survey of deep learning-based methods," *Computer Vision and Image Understanding*, 2020. [Citado nas páginas ix, 8, 9, 10, 11, 13, 17 e 19]

[37] Z. Liu, J. Zhu, J. Bu, and C. Chen, "A survey of human pose estimation: The body parts parsing based methods," *Journal of Visual Communication and Image Representation*, 2015. [Citado na página 8]

[38] W. Gong, X. Zhang, J. Gonzàlez, A. Sobral, T. Bouwmans, C. Tu, and E. H. Zahzah, "Human pose estimation from monocular images: A comprehensive survey," *Sensors (Switzerland)*, 2016. [Citado na página 8]

[39] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *International Journal of Computer Vision*, 2005. [Citado na página 9]

[40] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019. [Citado nas páginas x, 9, 17, 67, 68, 112, 124, 128, 130 e 131]

[41] Z. Cao, G. Hidalgo, T. Simon, S. E. Wei, and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. [Citado nas páginas 9, 17 e 31]

[42] H. S. Fang, J. Li, H. Tang, C. Xu, H. Zhu, Y. Xiu, Y. L. Li, and C. Lu, "AlphaPose: Whole-Body Regional Multi-Person Pose Estimation and Tracking in Real-Time," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. [Citado nas páginas 9 e 17]

[43] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2D human pose estimation: New benchmark and state of the art analysis," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014. [Citado nas páginas 9, 12, 25, 28 e 40]

[44] S. X. Ju, M. J. Black, and Y. Yacoob, "Cardboard people: a parameterized model of articulated image motion," *International Conference on Automatic Face and Gesture Recognition*, 1996. [Citado na página 9]

[45] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models - their training and application," *Computer Vision and Image Understanding*, 1995. [Citado na página 9]

[46] H. Sidenbladh, F. De La Torre, and M. J. Black, "A framework for modeling the appearance of 3D articulated figures," *2000 IEEE International Conference on Automatic Face and Gesture Recognition*, 2000. [Citado na página 9]

[47] K. Robinette, S. Blackwell, H. Daanen, M. Boehmer, and S. Fleming, "Civilian american and european surface anthropometry resource (caesar), final report. volume 1. summary," 2002. [Citado nas páginas 10 e 105]

[48] A. Newell, K. Yang, and J. Deng, "Stacked Hourglass Networks for Human Pose Estimation," in *Lecture Notes in Computer Science*, 2016. [Citado nas páginas 10, 13, 14, 17, 20, 79, 80 e 83]

[49] A. Toshev and C. Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014. [Citado nas páginas ix, 11, 59 e 63]

[50] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. [Citado nas páginas ix e 12]

[51] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015. [Citado nas páginas x, 12, 65 e 66]

[52] S. Johnson and M. Everingham, "Clustered pose and nonlinear appearance models for human pose estimation," in *British Machine Vision Conference*, 2010. [Citado na página 12]

[53] S. Ruder, "An Overview of Multi-Task Learning in Deep Neural Networks," 2017. [Citado nas páginas ix e 12]

[54] S. Li, Z. Q. Liu, and A. B. Chan, "Heterogeneous Multi-task Learning for Human Pose Estimation with Deep Convolutional Neural Network," *International Journal of Computer Vision*, 2015. [Citado na página 12]

[55] D. C. Luvizon, D. Picard, and H. Tabia, "2D/3D Pose Estimation and Action Recognition using Multitask Deep Learning," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. [Citado na página 12]

[56] X. Chen and A. Yuille, "Articulated Pose Estimation by a Graphical Model with Image Dependent Pairwise Relations," in *Advances in Neural Information Processing Systems*, 2014. [Citado na página 13]

[57] B. Xiao, H. Wu, and Y. Wei, "Simple Baselines for Human Pose Estimation and Tracking," in *Lecture Notes in Computer Science*, 2018. [Citado nas páginas 14 e 17]

[58] R. Girdhar, G. Gkioxari, L. Torresani, M. Paluri, and D. Tran, "Detect-and-Track: Efficient Pose Estimation in Videos," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. [Citado na página 14]

[59] M. Andriluka, U. Iqbal, E. Insafutdinov, L. Pishchulin, A. Milan, J. Gall, and B. Schiele, "PoseTrack: A Benchmark for Human Pose Estimation and Tracking," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. [Citado na página 14]

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. [Citado nas páginas 14, 17, 20, 21, 29, 33, 34, 36, 40, 41, 64, 102, 103, 112, 125 e 147]

[61] S. E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional Pose Machines," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. [Citado nas páginas ix, 14, 15 e 20]

[62] C. Y. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Journal of Machine Learning Research*, 2015. [Citado na página 14]

[63] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Citado nas páginas 14 e 82]

[64] Y. Chen, C. Shen, X. S. Wei, L. Liu, and J. Yang, "Adversarial PoseNet: A Structure-aware Convolutional Network for Human Pose Estimation," in *IEEE International Conference on Computer Vision*, 2017. [Citado nas páginas ix, 15, 16, 82, 83, 84 e 85]

[65] A. Jain, J. Tompson, Y. LeCun, and C. Bregler, "MoDeep: A deep learning framework using motion features for human pose estimation," in *Lecture Notes in Computer Science*, 2015. [Citado nas páginas ix, 15 e 16]

[66] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *IEEE International Conference on Computer Vision*, 2013. [Citado na página 15]

[67] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. [Citado nas páginas 17 e 81]

[68] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Cascaded Pyramid Network for Multi-Person Pose Estimation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. [Citado nas páginas 18 e 27]

[69] U. Rafi, A. Doering, B. Leibe, and J. Gall, "Self-supervised Keypoint Correspondences for Multi-Person Pose Estimation and Tracking in Videos," in *Lecture Notes in Computer Science*, 2020. [Citado na página 18]

[70] S. Yang, Z. Quan, M. Nie, and W. Yang, "TransPose: Keypoint Localization via Transformer," in *IEEE International Conference on Computer Vision*, 2021. [Citado nas páginas 18 e 86]

[71] Y. Li, S. Zhang, Z. Wang, S. Yang, W. Yang, S. T. Xia, and E. Zhou, "TokenPose: Learning Keypoint Tokens for Human Pose Estimation," in *IEEE International Conference on Computer Vision*, 2021. [Citado nas páginas ix, 18, 19 e 86]

[72] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele, "DeepCut: Joint subset partition and labeling for multi person pose estimation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. [Citado nas páginas 19 e 31]

[73] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," in *Lecture Notes in Computer Science*, 2016. [Citado na página 20]

[74] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," 2017. [Citado nas páginas ix, 20 e 21]

[75] A. Newell, Z. Huang, and J. Deng, "Associative embedding: End-to-end learning for joint detection and grouping," in *Advances in Neural Information Processing Systems*, 2017. [Citado na página 20]

[76] G. Papandreou, T. Zhu, L. C. Chen, S. Gidaris, J. Tompson, and K. Murphy, "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model," in *Lecture Notes in Computer Science*, 2018. [Citado na página 21]

[77] S. Li and A. B. Chan, "3D human pose estimation from monocular images with deep convolutional neural network," in *Lecture Notes in Computer Science*, 2015. [Citado na página 23]

[78] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Coarse-to-fine volumetric prediction for single-image 3D human pose," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017. [Citado nas páginas ix, 23 e 24]

[79] G. Pavlakos, X. Zhou, and K. Daniilidis, "Ordinal Depth Supervision for 3D Human Pose Estimation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. [Citado na página 23]

[80] C. Li and G. H. Lee, "Generating Multiple Hypotheses for 3D Human Pose Estimation with Mixture Density Network," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019. [Citado nas páginas ix, 24 e 25]

[81] Z. Zou, T. Liu, D. Wu, and W. Tang, "Compositional Graph Convolutional Networks for 3D Human Pose Estimation," *2021 IEEE International Conference on Automatic Face and Gesture Recognition*, 2021. [Citado na página 25]

[82] B. X. Nie, P. Wei, and S. C. Zhu, "Monocular 3D Human Pose Estimation by Predicting Depth on Joints," in *IEEE International Conference on Computer Vision*, Institute of Electrical and Electronics Engineers Inc., dec 2017. [Citado nas páginas ix e 26]

[83] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in Vision: A Survey," *ACM Computing Surveys*, 2022. [Citado nas páginas 26, 94 e 96]

[84] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," 2020. [Citado nas páginas xi, 27, 86, 94, 95, 96 e 97]

[85] C. Zheng, S. Zhu, M. Mendieta, T. Yang, C. Chen, and Z. Ding, "3D Human Pose Estimation with Spatial and Temporal Transformers," in *IEEE International Conference on Computer Vision*, 2021. [Citado nas páginas ix, 27 e 86]

[86] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017. [Citado nas páginas 29 e 78]

[87] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-End Recovery of Human Shape and Pose," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. [Citado nas páginas 30, 40 e 78]

[88] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. Black, "AMASS: Archive of motion capture as surface shapes," in *IEEE International Conference on Computer Vision*, 2019. [Citado nas páginas 30, 31, 116, 123 e 128]

[89] N. Kolotouros, G. Pavlakos, and K. Daniilidis, "Convolutional mesh regression for single-image human shape reconstruction," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019. [Citado nas páginas ix e 30]

[90] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, "Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image," *Lecture Notes in Computer Science*, 2016. [Citado na página 31]

[91] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. Osman, D. Tzionas, and M. J. Black, "Expressive body capture: 3D hands, face, and body from a single image," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019. [Citado nas páginas 31 e 105]

[92] A. A. Osman, T. Bolkart, and M. J. Black, "STAR: Sparse Trained Articulated Human Body Regressor," in *Lecture Notes in Computer Science*, 2020. [Citado na página 32]

[93] W. Zhu, X. Ma, Z. Liu, L. Liu, W. Wu, and Y. Wang, "Learning Human Motion Representations: A Unified Perspective," 2022. [Citado nas páginas 34 e 86]

[94] C. Wang, J. Li, W. Liu, C. Qian, and C. Lu, "HMOR: Hierarchical Multi-person Ordinal Relations for Monocular Multi-person 3D Pose Estimation," *Lecture Notes in Computer Science*, 2020. [Citado nas páginas x e 36]

[95] A. Benzine, F. Chabot, B. Luvison, Q. C. Pham, and C. Achard, "Pandanet: Anchor-based single-shot multi-person 3D pose estimation," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. [Citado na página 36]

[96] M. Fabbri, F. Lanzi, S. Calderara, A. Palazzi, R. Vezzani, and R. Cucchiara, "Learning to Detect and Track Visible and Occluded Body Joints in a Virtual World," in *Lecture Notes in Computer Science*, 2018. [Citado na página 37]

[97] H. Joo, T. Simon, X. Li, H. Liu, L. Tan, L. Gui, S. Banerjee, T. 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*, 2019. [Citado na página 37]

[98] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, S. Sridhar, G. Pons-Moll, and C. Theobalt, "Single-shot multi-person 3D pose estimation from monocular RGB," in *2018 International Conference on 3D Vision*, 2018. [Citado nas páginas 37 e 44]

[99] M. Fabbri, F. Lanzi, S. Calderara, S. Alletto, and R. Cucchiara, "Compressed Volumetric Heatmaps for Multi-Person 3D Pose Estimation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. [Citado nas páginas x e 38]

[100] J. Cha, M. Saqlain, G. U. Kim, M. Shin, and S. Baek, "Multi-Person 3D Pose and Shape Estimation via Inverse Kinematics and Refinement," *Lecture Notes in Computer Science*, 2022. [Citado nas páginas 40, 44 e 75]

[101] J. Li, C. Wang, H. Zhu, Y. Mao, H. S. Fang, and C. Lu, "Crowdpose: Efficient crowded scenes pose estimation and a new benchmark," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019. [Citado nas páginas 40 e 41]

[102] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Lecture Notes in Computer Science*, 2014. [Citado na página 40]

[103] T. von Marcard, R. Henschel, M. J. Black, B. Rosenhahn, and G. Pons-Moll, "Recovering accurate 3D human pose in the wild using IMUs and a moving

camera," in *Lecture Notes in Computer Science*, 2018. [Citado nas páginas xi, 40, 102, 124, 125, 126, 127 e 160]

[104] 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*, 2014. [Citado nas páginas 40, 44 e 130]

[105] S. H. Zhang, R. Li, X. Dong, P. Rosin, Z. Cai, X. Han, D. Yang, H. Huang, and S. M. Hu, "Pose2Seg: Detection free human instance segmentation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019. [Citado nas páginas 41, 78 e 82]

[106] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017. [Citado nas páginas xi, 41, 42, 43, 86, 87, 88, 90, 91, 92, 93, 94 e 117]

[107] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *International Conference on Image Processing*, 2018. [Citado na página 41]

[108] Y. Sun, W. Liu, Q. Bao, Y. Fu, T. Mei, and M. J. Black, "Putting People in their Place: Monocular Regression of 3D People in Depth," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2022. [Citado nas páginas x, 44, 45 e 46]

[109] A. Zanfir, E. Marinoiu, and C. Sminchisescu, "Monocular 3D Pose and Shape Estimation of Multiple People in Natural Scenes: The Importance of Multiple Scene Constraints," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. [Citado na página 44]

[110] A. Zanfir, E. Marinoiu, M. Zanfir, A. I. Popa, and C. Sminchisescu, "Deep network for the integrated 3D sensing of multiple people in natural images," *Advances in Neural Information Processing Systems*, 2018. [Citado na página 44]

[111] J. Zhang, D. Yu, J. H. Liew, X. Nie, and J. Feng, "Body Meshes as Points," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021. [Citado na página 44]

[112] N. Hesse, S. Pujades, J. Romero, M. J. Black, C. Bodensteiner, M. Arens, U. G. Hofmann, U. Tacke, M. Hadders-Algra, R. Weinberger, W. Müller-Felber, and A. Sebastian Schroeder, "Learning an infant body model from RGB-D data for accurate full body motion analysis," *Lecture Notes in Computer Science*, 2018. [Citado na página 46]

[113] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," *Foreign Affairs*, 2016. [Citado nas páginas x, 47, 48, 50, 51, 56, 57, 58 e 59]

[114] P. A. Miceli, W. D. Blair, and M. M. Brown, *Deep Learning with Python*. 2018. [Citado nas páginas x, 48, 49, 51, 55 e 59]

[115] M. Tom, *Machine Learning*. 1997. [Citado na página 54]

[116] T. Dietterich, "Overfitting and undercomputing in machine learning," *ACM Computing Surveys*, 1995. [Citado nas páginas 56 e 57]

[117] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark Analysis of Representative Deep Neural Network Architectures," 2018. [Citado nas páginas x, 61 e 62]

[118] O. Elharrouss, Y. Akbari, N. Almaadeed, and S. Al-Maadeed, "Backbones-Review: Feature Extraction Networks for Deep Learning and Deep Reinforcement Learning Approaches," 2022. [Citado nas páginas x, 61, 64 e 65]

[119] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014. [Citado nas páginas 63 e 69]

[120] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015. [Citado na página 64]

[121] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017. [Citado na página 67]

[122] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. [Citado na página 67]

[123] A. Howard, M. Sandler, B. Chen, W. Wang, L. C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, Q. Le, and H. Adam, "Searching for MobileNetV3," in *IEEE International Conference on Computer Vision*, 2019. [Citado na página 67]

[124] C. Zheng, M. Mendieta, P. Wang, A. Lu, and C. Chen, "A Lightweight Graph Transformer Network for Human Mesh Reconstruction from 2D Human Pose," 2022. [Citado nas páginas 67 e 86]

[125] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang, "High-Resolution Representations for Labeling Pixels and Regions," 2019. [Citado nas páginas x, 68 e 69]

[126] M. Ekman, "Learning Deep Learning," *Deep Learning Institute NVIDIA*, 2022. [Citado nas páginas x, xi, 70, 71, 72, 78, 79 e 81]

[127] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision*, 2015. [Citado na página 70]

[128] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. [Citado nas páginas x, 73 e 74]

[129] J. Terven and D. Cordova-Esparza, "A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond," 2023. [Citado nas páginas 73, 74 e 75]

[130] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018. [Citado na página 75]

[131] H. Chu, J. H. Lee, Y. C. Lee, C. H. Hsu, J. D. Li, and C. S. Chen, "Part-aware measurement for robust multi-view multi-human 3d pose estimation and tracking," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2021. [Citado na página 75]

[132] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020. [Citado nas páginas x, 75 e 76]

[133] Ultralytics, "GitHub - Ultralytics YOLOv8," 2023. [Citado nas páginas xi, xiii, 76, 77 e 102]

[134] D. Maji and M. Mathew, "YOLO-Pose Enhancing YOLO for Multi Person Pose Estimation Using Object," 2022. [Citado na página 76]

[135] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *IEEE International Conference on Computer Vision*, 2015. [Citado nas páginas xi, 79 e 80]

[136] C. J. Chou, J. T. Chien, and H. T. Chen, "Self Adversarial Training for Human Pose Estimation," in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2017. [Citado na página 82]

[137] K. Li, S. Wang, X. Zhang, Y. Xu, W. Xu, and Z. Tu, "Pose Recognition with Cascade Transformers," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021. [Citado na página 86]

[138] U. Kamath, K. L. Graham, and W. Emara, *Transformers for Machine Learning*. 2022. [Citado nas páginas xi, 87, 88, 89, 90, 91, 92 e 93]

[139] G. Maggiolino, A. Ahmad, J. Cao, and K. Kitani, "Deep OC-SORT: Multi-Pedestrian Tracking by Adaptive Re-Identification," 2023. [Citado nas páginas xiii, 102, 103 e 131]

[140] Y. Du, Z. Zhao, Y. Song, Y. Zhao, F. Su, T. Gong, and H. Meng, "StrongSORT: Make DeepSORT Great Again," *IEEE Transactions on Multimedia*, 2023. [Citado na página 103]

[141] J. Cao, J. Pang, X. Weng, R. Khirodkar, and K. Kitani, "Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking," 2022. [Citado na página 103]

[142] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "ByteTrack: Multi-object Tracking by Associating Every Detection Box," in *Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*, 2022. [Citado na página 103]

[143] A. Milan, L. Leal-Taixé, T. Taixé, I. Reid, S. Roth, and K. Schindler, "MOT16: A Benchmark for Multi-Object Tracking," [Citado na página 103]

[144] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "MOT20: A benchmark for multi object tracking in crowded scenes," 2020. [Citado na página 103]

[145] J. Li, S. Bian, C. Xu, Z. Chen, L. Yang, and C. Lu, "HybrIK-X: Hybrid Analytical-Neural Inverse Kinematics for Whole-body Mesh Recovery," 2023. [Citado nas páginas xiii, 112, 113, 131 e 160]

[146] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," IEEE Conference on Computer Vision and Pattern Recognition, 2019. [Citado na página 121]

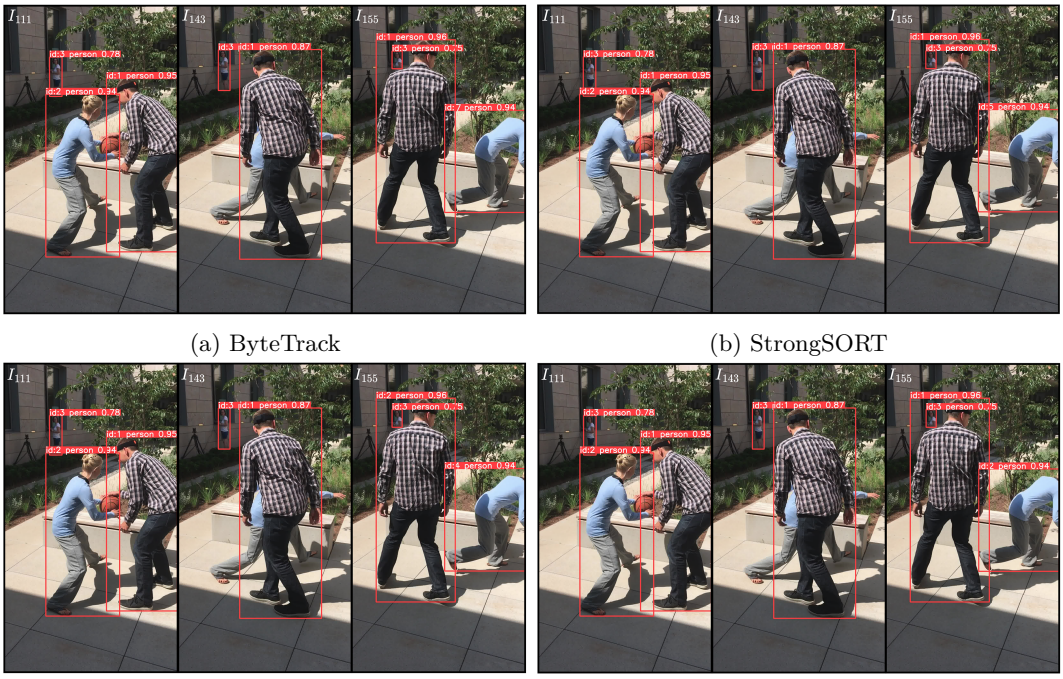
[147] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt, "Monocular 3d human pose estimation in the wild using improved cnn supervision," 2017. [Citado na página 123]

[148] M. Kaufmann, E. Aksan, J. Song, F. Pece, R. Ziegler, and O. Hilliges, "Convolutional autoencoders for human motion infilling," in *2020 International Conference on 3D Vision (3DV)*, IEEE, 2020. [Citado nas páginas 129 e 130]

[149] J. Li, S. Bian, A. Zeng, C. Wang, B. Pang, W. Liu, and C. Lu, "Human Pose Regression with Residual Log-likelihood Estimation," in *IEEE International Conference on Computer Vision*, 2021. [Citado na página 160]

Anexo A

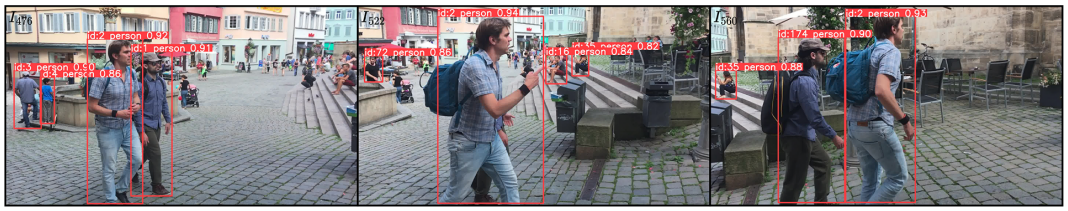
MPT: Resultados Qualitativos em Diferentes Métodos



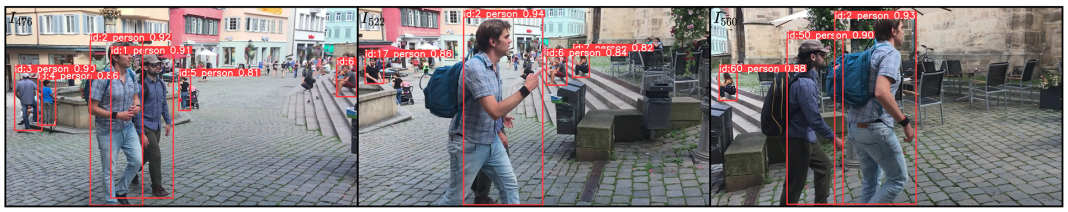
(a) ByteTrack (b) StrongSORT

(c) OC-SORT (d) Deep OC-SORT

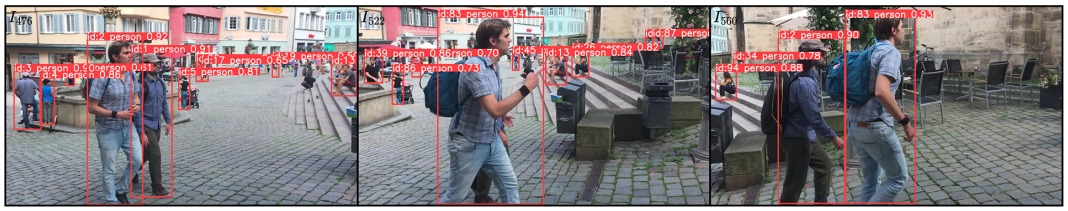
Figura A.1: Resultados visuais do MPT com diferentes *trackers* e mantendo o mesmo *Re-ID model* (ResNet-50 [60], nos modelos que assim o exijam) no *dataset* 3DPW (`courtyard_basketball_00`). Confiança de 0,6 e IoU *threshold* de 0,3. É notória a troca de IDs nos vários *trackers* no momento de oclusão (*frame* 143) da pessoa com ID inicial 2, com ênfase nos resultados do (c) OC-SORT a sofrer de uma troca recursiva de IDs. Novamente, o (d) Deep OC-SORT mostrou um comportamento robusto e estável à não linearidade do movimento humano. Melhor interpretação da figura a cores e ampliada.



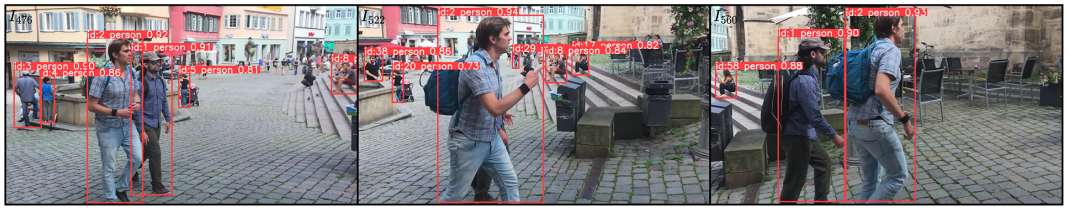
(a) ByteTrack



(b) StrongSORT



(c) OC-SORT



(d) Deep OC-SORT

Figura A.2: Resultados visuais do MPT com diferentes *trackers*, mantendo a mesma configuração, no *dataset* 3DPW (`downtown_cafe_002`). É notória a troca de IDs no vários *trackers* no momento de oclusão (*frame* 522) da pessoa com ID inicial 1. A errática facilidade na criação de novos IDs do (a) ByteTrack e (c) OC-SORT é evidenciada nestas amostras. Melhor interpretação da figura a cores e ampliada.

(a) ByteTrack

(b) StrongSORT

(c) OC-SORT

(d) Deep OC-SORT

Figura A.3: Resultados visuais do MPT com diferentes *trackers*, mantendo a mesma configuração, no *dataset* 3DPW (`downtown_crossStreets_00`). É notória a troca de IDs no vários *trackers* em diversos momento de oclusão (oclusão total no *frame* 522) das pessoa com ID inicial 1 e 3. Destaque na capacidade de acompanhamento do (b) StrongSORT no ID 3. Novamente, o (d) Deep OC-SORT superar os restantes que, apesar do tempo de *tracking* superior, tenderá a acrescentar robustez à solução final. Melhor interpretação da figura a cores e ampliada.

Esta página foi propositadamente deixada em branco

Anexo B

HPS: Resultados Qualitativos

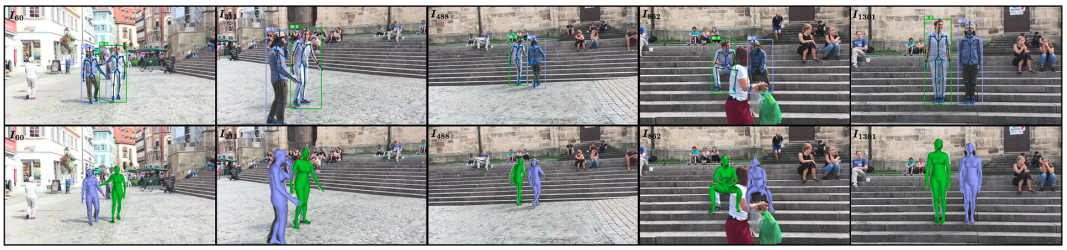


Figura B.1: Resultados visuais do *HPS estimator* com *backbone* HRNet-W48 no *dataset* 3DPW (`downtown_sitOnStairs_00`). Foco no *frame* 862 pela capacidade de prever a localização das articulações do esqueleto humano apesar da oclusão parcial. Melhor interpretação da figura a cores e ampliada.

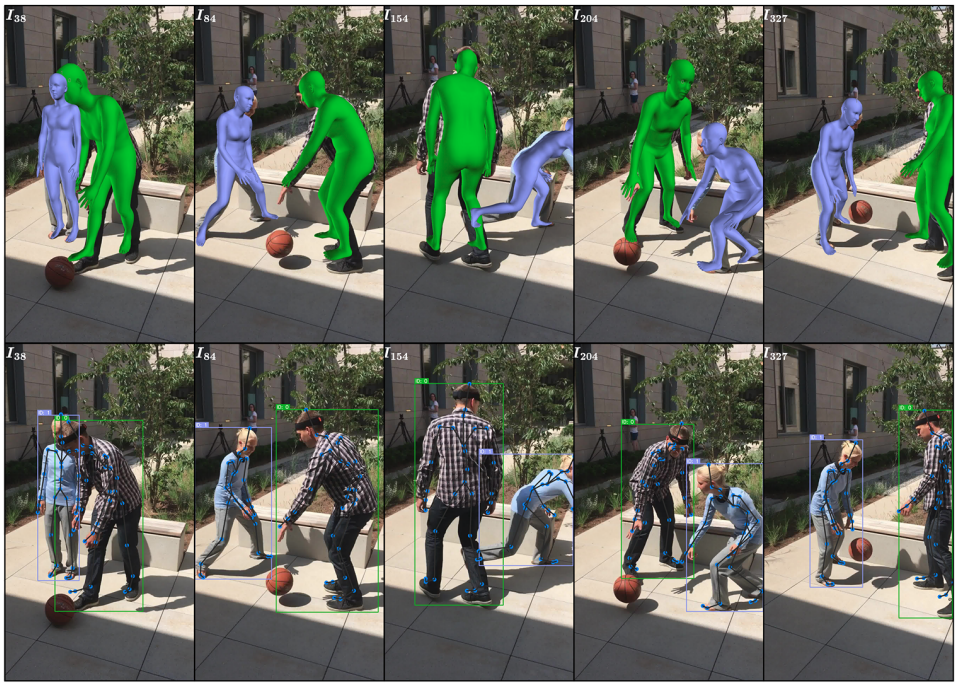


Figura B.2: Resultados visuais do *HPS estimator* com *backbone* HRNet-W48 no *dataset* 3DPW (`courtyard_basketball_00`). Importa realçar que o algoritmo não falha na consideração da profundidade de cada indivíduo, simplesmente, a ordem de projeção da malha corporal no plano de imagem é a mesma da atribuição de IDs, resultando numa sobreposição. Contudo, é percetível um pequeno deslocamento nas articulações dos pés da pessoa com ID 0 (verde). Este erro deve-se à determinação errática do comprimento dos ossos (parâmetros do *shape* β), dos *3D keypoints* ou na consideração de parâmetros intrínsecos distintos dos reais para esta câmara. Melhor interpretação da figura a cores e ampliada.

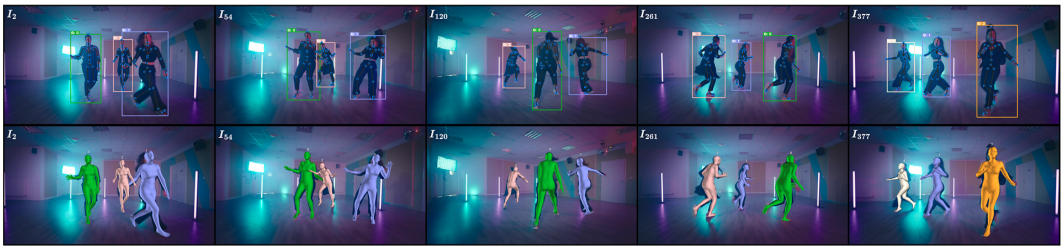


Figura B.3: Resultados visuais do *HPS estimator*, com *backbone* HRNet-W48, num vídeo da *internet* (*Havoc Ladies*). Apesar das posições complexas dos humanos e da inferior resolução do vídeo fica evidente a capacidade de determinação dos *3D keypoints* e projeção da malha reconstruída. Ainda que supere positivamente a determinação dos parâmetros que definem os humanos no fundo do cenário, tende a falhar na proximidade do pé ao chão, *i.e.*, distância euclidiana não nula quando o sujeito tem o pé completamente pousado (*frame* 120). Este problema é abordado no Anexo E.Melhor interpretação da figura a cores e ampliada.

Anexo C

HybrIK: ResNet-34 *vs.* HRNet-W48

Figura C.1: Resultados visuais do *HPS estimator* com diferentes *backbones* no *dataset* 3DPW (`downtown_cafe_002`). Resultados com o *backbone* ResNet-34 (primeira linha). Resultados com o *backbone* HRNet-W48 (segunda linha). Foco na orientação da cabeça do humano com ID 1 (roxo) e no posicionamento dos pés, corretamente projetados, face à oclusão parcial da cadeira. Dada à consideração exclusiva de 24 articulações por parte do modelo estatístico utilizado para representar o humano, por vezes, a rotação do pulso e da mão não é a adequada (naturalmente, a contração dos dedos não deve ser considerada, fruto das limitações do modelo corporal SMPL), contudo, é percetível uma melhoria no novo *backbone* utilizado. Melhor interpretação da figura a cores e ampliada.

Figura C.2: Resultados visuais do *HPS estimator* com diferentes *backbones* no *dataset* 3DPW (`courtyard_basketball_00`). Resultados com o *backbone* ResNet-34 (primeira linha). Resultados com o *backbone* HRNet-W48 (segunda linha). Novamente, atenção à orientação da cabeça do humano com ID 1 (roxo). Nesta sequência de imagens é percetível que a errada determinação dos *3D keypoints* das articulações que compõe os pés não terá sofrido melhorias. Melhor interpretação da figura a cores e ampliada.

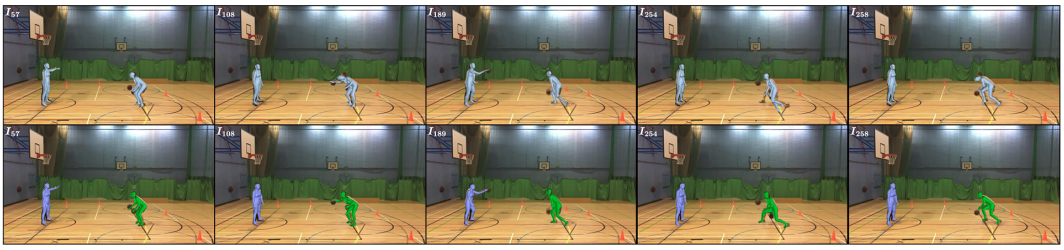


Figura C.3: Resultados visuais do *HPS estimator* com diferentes *backbones* num vídeo do repositório oficial do GLAMR [34]. Resultados com o *backbone* ResNet-34 (primeira linha). Resultados com o *backbone* HRNet-W48 (segunda linha). Este vídeo, de baixa resolução, destaca o aumento de precisão e redução de erro do *backbone* utilizado pelas consecutivas falhas na determinação das articulações dos humanos no *backbone* ResNet-34. Realça-se a orientação da mão esquerda do humano com ID 1 (roxo) no *frame* 57 ou falha no posicionamento das pernas do humano com ID 0 (verde) no *frame* 254. Melhor interpretação da figura a cores e ampliada.

Anexo D

Resultados após *Optimizer*

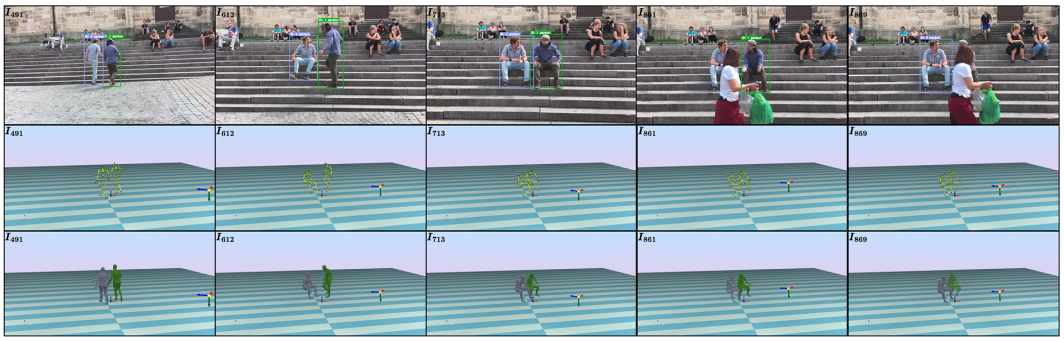


Figura D.1: Resultados finais da reconstrução de movimento no *dataset* 3DPW (`downtown_sitOnStairs_00`). É visível um deslocamento dinâmico da câmara e das pessoas face ao referencial do mundo, elevando-se quando os sujeitos avaliados começam a subir as escadas. Não obstante, face ao visualizador (*i.e.*, leitor, externo no momento de recolha das imagens), o centro da câmara tende a afastar-se pela aproximação aos humanos. Nota para as últimas duas imagens, com previsão dos parâmetros descritivos dos dois sujeitos. Melhor interpretação da figura a cores e ampliada.

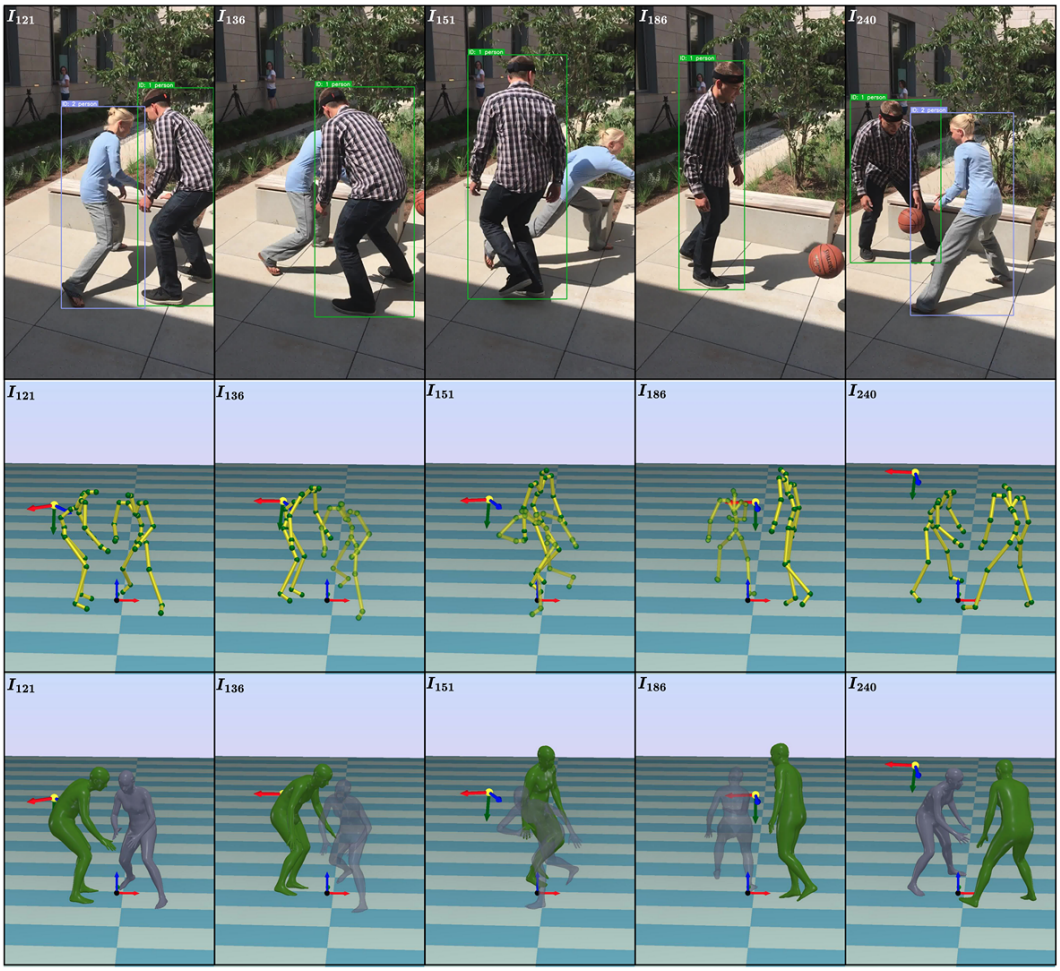


Figura D.2: Resultados finais da reconstrução de movimento no *dataset* 3DPW (`courtyard_basketball_00`). Destaca-se a capacidade do MPT a manter o identificador do humano a roxo (ID 2) e da robustez no preenchimento de movimento deste mesmo sujeito. Apesar da complexidade do posicionamento entre os *frames* 121 e 240, o *generative motion optimizer* conseguiu projetar uma rotação da árvore cinemática, face ao eixo longitudinal deste, enquanto se encontrava fora do FoV da câmara. Melhor interpretação da figura a cores e ampliada.

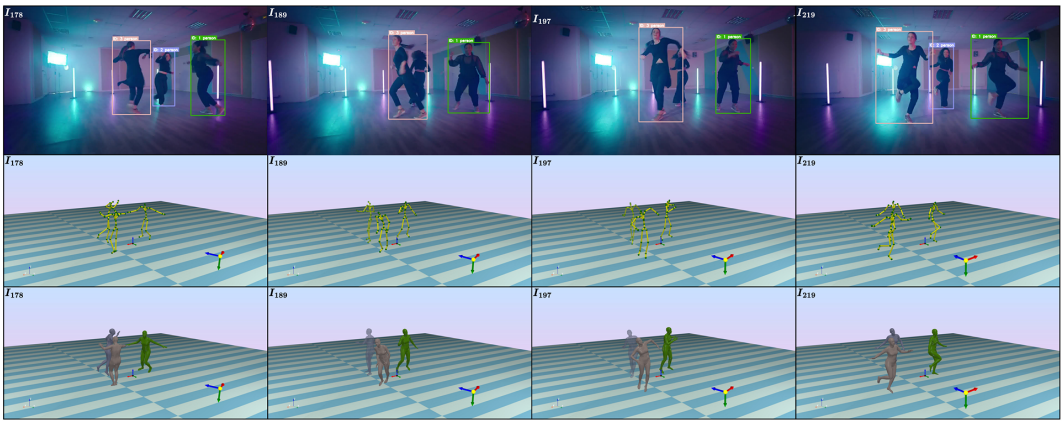


Figura D.3: Resultados finais da reconstrução de movimento num vídeo da *internet* (*Havoc Ladies*). Neste vídeo de baixa resolução, é visível a robustez do algoritmo na determinação da distância da câmara aos humanos, no fundo do cenário, face à sua altura ao longo do tempo (mantendo se constante). Realça-se o correto preenchimento do movimento no humano com ID 2 (roxo), enquanto se encontra oculto por outra pessoa e realiza uma rotação em torno do seu eixo longitudinal. Melhor interpretação da figura a cores e ampliada.

Esta página foi propositadamente deixada em branco

Anexo E

Discussão sobre as Limitações

Durante o desenvolvimento foram evidenciadas algumas limitações que serão relevantes para trabalhos futuros. Realça-se a dependência entre as diferentes fases do modelo, isto é, tratando-se de um modelo iterativo existe a propagação de erro ao longo dos vários estados (*e.g.*, más deteções no MPT levam à determinação de parâmetros erráticos que caracterizam o movimento humano e tendem a prejudicar a resposta à sequência do *generative motion optimizer*. A abordagem proposta a este problema passava pela integração conjunta de todos ou partes constituintes do modelo.

O tempo de otimização impede a operação em tempo real, devido ao elevado processamento de informação nas diferentes fases. Trabalhos futuros poderão reavaliar o processo de otimização e procurar reduzir o tempo de inferência.

A consideração da informação presente no cenário (*e.g.*, representação numa *point cloud*) pode providenciar evidências que permitirão aproximar os pés do humano do chão (*foot contact*) [11], segundo um termo energético, na função de perdas do processo de otimização, para minimizar a distância euclidiana da malha corporal local ao chão.

Num pré-processamento, fazer a deteção do género e outra informação que permita a atribuição do modelo paramétrico correto (*i.e.*, masculino, feminino ou neutro). Contudo, foi verificado que o modelo SMPL não apresentava alterações consideráveis na malha humana gerada para os diferentes géneros, não tendo este processo sido incorporado. Assim sendo, num proposta mais complexa, podia ser revista a

estrutura do *generative motion optimizer* de forma a operar com o modelo paramétrico SMPL-X, suportado no *backbone* mais recente do HybrIK [145], e com um controlo superior das poses e malhas geradas (*e.g.*, maior número de parâmetros de *shape*).

Não obstante, o método apresentado apenas recupera os parâmetros SMPL que omitem os detalhes visíveis da malha real humana, *e.g.*, roupas. A incorporação de um método capaz de projetar estas variações ao nível da malha poderia trazer uma valência ao projeto.

Tal como referido, ficou pendente de implementação um termo energético que penalize a *inter-person penetration* e as interseções entre malhas distintas $E_{\texttt{pen}}$ [24, 34]. Abordagens recentes apostam numa abordagem análogo a [9] com a incorporação do cálculo do *Signed Distance Field* (SDF).

Relativamente ao *HPS estimator*, a determinação dos *keypoints* com base num representação probabilística em *volumetric heatmaps* revelou-se uma solução com um custo computacional acrescido. Além disso, limita os resultados obtidos à *bounding box* de entrada, que pode falhar em momentos com piores previsões do MPT, *e.g.*, parte do corpo da pessoa fora da imagem. Uma representação alternativa pode ser feita com base no paradigma *Residual Log-likelihood Estimation* (RLE) [149] para determinar as coordenadas da articulações.

Tendo por base as informações não exploradas do *dataset* 3DPW [103], *e.g.*, medições inerciais providenciadas pelos IMUs em cada sujeito, poderia ser realizado um estudo aprofundado da *inertial odometry* nos humanos com intuito comparativo do *global trajectory predictor*, contribuindo, eventualmente durante a fase de treino do modelo.

Finalmente, dado as dificuldade do autor deste trabalho em aceder ao *dataset* Human3.6M e *Dynamic Human3.6M*, não terá sido possível realizar quais tipos de *benchmarks* no mesmo e como tal, não foi possível comparar com os resultados das soluções que compõe a base deste projeto (*i.e.*, GLAMR [34] com *HPS estimator* KAMA [32] ou SPEC [33]).

Anexo F

Libraries and Framework Versions

O presente anexo procura auxiliar o leitor na reimplementação do projeto, reunindo numa tabela as diferentes bibliotecas e *frameworks* utilizadas. Recomenda-se, contudo, a leitura cuidada da informação presente no repositório *open-source* do GitHub (https://github.com/AndreOliveira00/3D-Pose-and-Shape-Estimation-with-a-Camera-System.git) desenvolvido pelo autor.

Relembrando, os resultados terão sido obtidos pela execução numa única GPU NVIDIA GeForce GTX 1650 para um CPU Intel Core i5-9300H no sistema operativo Ubuntu 20.04.6 LTS.

Para efeito de implementação recomenda-se a utilização de três ambientes propriamente definidos (*i.e.*, configurados para o MPT, HybrIK [23] e GLAMR [34]) com uma gestão potencializada pela Anaconda e comutados pela utilização do módulo `subprocess`.

Assume-se que existe uma placa gráfica dedicada durante a implementação, como tal, deve ser feita a instalação do PyTorch com a versão CUDA correta assim como as dependências presentes no repositório respetivo (*i.e.*, `requirements.txt`).

Posto isto, as seguintes tabelas apresentam os principais *packages*, dividindo-se nos três ambientes referidos. Complementarmente, no repositório do projeto existe três ficheiros YAML passíveis de ser importados como ambiente no Anaconda.

Tabela F.1: *MPT: packaged versions.*

Package	Version
cython	0.29.35
ffmpeg	4.2.2
ffmpeg-python	0.2.0
filterpy	1.4.5
gdown	4.7.1
gitpython	3.1.31
imageio	2.30.0
imutils	0.5.4
joblib	1.2.0
lap	0.4.0
loguru	0.7.0
matplotlib	3.7.1
ninja	1.10.2
numpy	1.23.0
onnx	1.13.0
onnxruntime	1.13.1
onnx-simplifier	0.4.31
opencv-python	4.7.0.72
opendr	0.78
pandas	2.0.2
pillow	9.3.0
pyqt	5.9.2
python	3.9.16
pytorch	1.9.1
pytorch-cuda	11.7
pyyaml	5.3
qt	5.9.7
scikit-learn	1.2.2
scipy	1.9.1
sqlite	3.41.2
tb-nightly	2.14.0
tensorboard	2.13.0
torchaudio	0.9.1
torchvision	0.10.1
wheel	0.40.0
yacs	0.1.6
yaml	0.2.5

Tabela F.2: *HybrIK: packaged versions.*

Package	Version
chumpy	0.70
cython	0.29.35
easydict	1.10
ffmpeg	4.3
ffmpeg-python	0.2.0
future	0.18.3
fvcore	0.1.5
iopath	0.1.9
joblib	1.3.1
matplotlib	3.7.1
ninja	1.10.2
numpy	1.23.1
nvidiacub	1.10.0
opencv-python	4.8.0.74
opendr	0.78
pillow	9.3.0
pycocotools	2.0.6
pyqt	5.9.2
python	3.9.16
pytorch	1.9.1
pytorch3d	0.7.4
pytorch-cuda	11.7
pyyaml	6.0
qt	5.9.7
scipy	1.11.1
six	1.16.0
smplx	0.1.28
sqlite	3.41.2
tb-nightly	2.14.0
tensorboard	2.13.0
terminaltables	3.1.10
torchaudio	0.9.1
torchvision	0.10.1
tqdm	4.65.0
wheel	0.38.4
yacs	0.1.8
yaml	0.2.5

Tabela F.3: *GLAMR: packaged versions.*

Package	Version
apptools	5.2.0
autograd	1.5
chumpy	0.70
cython	0.29.35
easydict	1.10
ffmpeg	4.3
ffmpeg-python	0.2.0
future	0.18.3
fvcore	0.1.5
graphviz	0.20.1
h5py	3.8.0
imutils	0.5.4
iopath	0.1.9
joblib	1.2.0
lmdb	1.4.1
matplotlib	3.7.1
multiprocess	0.70.14
ninja	1.10.2
numpy	1.23.0
nvidiacub	1.10.0
opencv-python	4.7.0.72
opendr	0.78
pathos	0.3.0
pillow	9.3.0
progress	1.6
protobuf	3.19.6
pycocotools	2.0.6
pynvml	11.5.0

Package	Version
pyqt	5.9.2
pyrender	0.1.45
python	3.9.16
pytorch	1.9.1
pytorch3d	0.7.4
pytorch-cuda	11.7
pytorch-lightning	1.3.5
pytube	15.0.0
pyvista	0.40.dev0
pyyaml	5.4.1
qt	5.9.7
scikit-image	0.20.0
scikit-learn	1.2.2
scipy	1.9.1
seaborn	0.12.2
six	1.16.0
sklearn	0.0.post5
smplx	0.1.28
sqlite	3.41.2
tb-nightly	2.14.0
tensorboard	2.13.0
terminaltables	3.1.10
torchaudio	0.9.1
torchvision	0.10.1
tqdm	4.65.0
wheel	0.40.0
yacs	0.1.6
yaml	0.2.5