Pontifícia Universidade Católica
do Rio de Janeiro

Luiz José Schirmer Silva

Semantic graph attention networks and tensor decompositions for computer vision and computer graphics

Tese de Doutorado

Thesis presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências – Informática.

Advisor : Prof. Hélio Côrtes Vieira Lopes
Co-advisor: Prof. Luiz Carlos Pacheco Rodrigues Velho

Rio de Janeiro
February 2021

Luiz José Schirmer Silva

Semantic graph attention networks and tensor decompositions for computer vision and computer graphics

Thesis presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências – Informática. Approved by the Examination Committee.

Rio de Janeiro, February 8th, 2021

Luiz José Schirmer Silva

Bachelor's in Computer Science (2014) at the Federal University of Santa Maria (UFSM). Masters' in Informatics (2016) at PUC-Rio with emphasis on Computer Graphics. Worked for the Applied Computing Laboratory (LaCA-UFSM) from 2011 to 2014 and Tecgraf/PUC-Rio from 2014 to 2017 in the Reservoir visualization group. Since 2017, works at GALGOS/ATD-Lab, at PUC-Rio.

Bibliographic data

Schirmer Silva, Luiz José

Semantic graph attention networks and tensor decompositions for computer vision and computer graphics / Luiz José Schirmer Silva; advisor: Hélio Côrtes Vieira Lopes; co-advisor: Luiz Carlos Pacheco Rodrigues Velho. – Rio de janeiro: PUC-Rio, Departamento de Informática, 2021.

v., 101 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Redes Neurais de convolução. 2. Pose estimation. 3. Modelos de atenção. 4. Decomposição de Tensores. 5. Redes neurais para Grafos. 6. Aplicações em tempo real. I. Lopes, Hélio Côrtes Vieira. II. Velho, Luiz. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

Acknowledgments

To my lovely wife, Letícia Fausto, for all support and patience. I will never be able to repay you fully; all I can give you is my eternal love and friendship. To my parents, José Luiz and Gilka, for the care, support, friendship, and unconditional love. You have my eternal thanks.

To prof. Hélio Lopes and prof. Luiz "Oldman" Velho. You guided me on the path to the good side of the force: computer graphics, mathematics, and art. Thank you for the help and advice. If I have arrived here today, it was because of your counsels and helped me along the way. Also, thanks for all the learning and patience!

To Djalma, the "DJ Soul," for all the help and the hilarious moments at IMPA. Certainly, without your help, this work would not be possible. Also, you are a great dancer!

To my brother and sister of a different mother, Guilherme Schardong and Josyane Almeida, for the support and counsels! I will never forget the hard times and hilarious moments when we share the apartment.

To my friends Renan Spencer and Gabi Dias for the support, counsels and memes. *J'espère que vous vé débrouillez bien en France!* Also, thank you Renan for always helping me when my boat was sinking!

To Captain Fabrício Cardoso, Leonardo Campagnolo and Abner Cardoso for the "Halo nights" on Saturdays, the counsels and talks. Oh, I almost forgot! Thanks, Fabrício, for the real Açaí from Pará that gave me the energy to complete this challenge!

To my friends from Rio de Janeiro, in special to Jonatas and Julia Grossman and Georges Spyrides and Helen.

To my friends from the Rio Grande do Sul, Laisla, Aline, and Lidiane. Although you don't really understand what I was doing, you always supported it.

To the VISGRAF Lab at IMPA and all the crazy mathematicians that I met there. Computer Graphics is about math and a lot of passion!

To GALGOS Lab, DAS Lab, and all the crazy lunatics that I met while working in PUC-Rio. This journey would certainly be boring without you people.

To Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) for partially financing this research under grant 153737/2014-0.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

Abstract

Schirmer Silva, Luiz José; Lopes, Hélio Côrtes Vieira (Advisor); Velho, Luiz (Co-Advisor). **Semantic graph attention networks and tensor decompositions for computer vision and computer graphics**. Rio de Janeiro, 2021. 101p. Tese de doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This thesis proposes new architectures for deep neural networks with attention enhancement and multilinear algebra methods to increase their performance. We also explore graph convolutions and their particularities. We focus here on the problems related to real-time pose estimation. Pose estimation is a challenging problem in computer vision with many real applications in areas including augmented reality, virtual reality, computer animation, and 3D scene reconstruction. Usually, the problem to be addressed involves estimating the 2D and 3D human pose, i.e., the anatomical keypoints or body "parts" of persons in images or videos. Several papers propose approaches to achieve high accuracy using architectures based on conventional convolution neural networks; however, mistakes caused by occlusion and motion blur are not uncommon, and those models are computationally very intensive for real-time applications. We explore different architectures to improve processing time, and, as a result, we propose two novel neural network models for 2D and 3D pose estimation. We also introduce a new architecture for Graph attention networks called Semantic Graph Attention.

Keywords

Convolutional Neural Networks; Pose estimation; Attention models; Tensor Decomposition; Graph neural networks; Real time applications;

Resumo

Schirmer Silva, Luiz José; Lopes, Hélio Côrtes Vieira; Velho, Luiz. **Redes de grafos semânticos com atenção e decomposição de tensores para visão computacional e computação gráfica**. Rio de Janeiro, 2021. 101p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Nesta tese, propomos novas arquiteturas para redes neurais profundas utlizando métodos de atenção e álgebra multilinear para aumentar seu desempenho. Também exploramos convoluções em grafos e suas particularidades. Nos concentramos aqui em problemas relacionados à estimativa de pose em tempo real. A estimativa de pose é um problema desafiador em visão computacional com muitas aplicações reais em áreas como realidade aumentada, realidade virtual, animação por computador e reconstrução de cenas 3D. Normalmente, o problema a ser abordado envolve estimar a pose humana 2D ou 3D, ou seja, as "partes" do corpo de pessoas em imagens ou vídeos, bem como seu posicionamento e estrutura. Diveros trabalhos buscam atingir alta precisão usando arquiteturas baseadas em redes neurais de convolução convencionais; no entanto, erros causados por oclusão e motion blur não são incomuns, e ainda esses modelos são computacionalmente pesados para aplicações em tempo real. Exploramos diferentes arquiteturas para melhorar o tempo de processamento destas redes e, como resultado, propomos dois novos modelos de rede neural para estimativa de pose 2D e 3D. Também apresentamos uma nova arquitetura para redes de atenção em grafos chamada de atenção em grafos semânticos.

Palavras-chave

Redes Neurais de convolução; Pose estimation; Modelos de atenção; Decomposição de Tensores; Redes neurais para Grafos; Aplicações em tempo real;

Table of contents

List of figures

List of tables

List of abreviations

CNN – Convolution Neural Networks	
CPM – Convolutional Pose Machines	
GAT – Graph attention Network	
GCN - Graph Convolutional Networks	
SGAT – Semantic Graph Attention	
GPU – Graphics Processing Unity	
FCL – Fully Connected Network	
HMR – Human Mesh Recovery	
SE-NET – Squeeze and excitation Networks	
SVD – Singular Value Decomposition	
HOSVD – High Order Singular Value Decomposition	
VBMF – Variational Bayesian Matrix Factorization	
PAF – Part Affinity Fields	
KLT – Kanade Lucas Tomasi Algorithm	
OKS – Object Keypoint Similarity	
COCO – Common Objects in Context	
MPJPE – Mean Per-Joint Position Error	
P-MPJPE – Mean per-joint position error after rigid alignment	

Vencemos! Com dificuldade! Não tínhamos a mesma velocidade, mas tínhamos o coração.

Zagalo, *Final de Copa América de Futebol, 1997.*

1 Introduction

In this thesis, we propose new architectures for deep neural networks with attention enhancement and multilinear algebra methods for increase their performance. We also explore graph convolutions and its particularities. We focus here in the problems related to real time pose estimation.

Pose estimation is a challenging problem in computer vision with many real applications in areas including augmented reality, virtual reality, computer animation and 3D scene reconstruction (ranjan2019, sindagi2018, ge2018, schwarcz2018, lin2017). Usually, the problem to be addressed involves estimating the 2D human pose, i.e., the anatomical keypoints or body “parts” of persons in images or videos(cao2017). Recent works have presented great results for multi-person pose estimation in images, but still, have some issues when considering videos (luo2018). These approaches can achieve high accuracy using architectures based on conventional convolution neural networks; however, mistakes caused by occlusion and motion blur are not uncommon, and those models are computationally very intensive for real-time applications (luo2018, lin2017). Also, despite some related work to increase performance, few papers explore the use of temporal coherence and the possibility of using such machine learning models for real-time applications involving animation (mehta2017).

OpenPose is considered the state-of-art approach on multi-person pose estimation, but it does not achieve the desired performance in terms of frames per second, which make it difficult to use in interactive applications that require frame rates close to or above 30 FPS, even in high end GPUs.

OpenPose is an evolution of convolutional Pose Machines, which is a fully convolutional neural network. Also, there is evidence that a key feature behind the success of these methods is over-parameterization. It could help in finding a good local minimum, however it also leads to a large amount of redundancy (kossaifi2019). Furthermore, models with a larger number of parameters have increased storage and are computationally intensive. Several papers focus on improving the efficiency of CNNs using tensor decompositions. Most of them consider each layer independently, where the kernel of a convolutional layer can be seen as a 4-dimensional matrix and decomposed in a set of low-rank approximations. On the other hand, we have hand-crafted decomposition methods that use pointwise and depthwise convolutions to improve performance, such as the Mobilenets (howard2017, sandler2018).

Another weakness of convolutional neural networks is that convolution operations consider only local neighborhoods thus missing global information (bello2019). Recently several papers propose the use of attention modules to leverage this problem, for example, the use of Squeeze-and-Excitation networks (hu2018), Gather-Excite for feature analysis(hu2018), and convolutional block attention modules (CBAM) (woo2018). They show consistent improvements in the result for image classification on ImageNet (deng2009) and in the COCO dataset (lin2014) across many different models and scales, proving the potential of this approach.

In one of our proposed models, we replace conventional convolution operations by successive pointwise and regular convolutions in a reduced space. As we will show, the proposed modifications is analogous to applying a tensor decomposition, more specifically a high order singular value decomposition (HOSVD). We propose a Convolutional Pose Machine following tensor decomposition models and introducing a new architecture with attention mechanisms, not only improving performance but also reducing redundancy for pose estimation tasks.

We also explore methods to infer the 3D human pose. Several projects use motion capture to produce computer-animated movies, games, medical applications, or sports analysis. However, in most of these applications, its use depends typically on multiple sensors and commercial systems. This is not only a costly technology but also depends on specialized hardware. Moreover, it is far from being accessible to most producers. We can find solutions that present alternative ways to reduce the cost and computational processing for this kind of application. To overcome these problems, several papers propose solutions using neural networks(martinez2017, zhao2019, mehta2017, mehta2020). Many of these works are based on the premise of using a simple monocular RGB camera, combining techniques to generate the 2D pose estimation and after use it as input for a 3D regression. Nowadays, 2D pose estimation methods focus on real-time performance. But, when we consider monocular RGB cameras to estimate 3D skeletal pose, this is a much harder challenge.

Convolutional Neural Networks, through a grid structure, when processing images, achieved state of art results for object detection, image segmentation and generation. However, many data structures such as 3D Meshes and human skeletons can only be represented in the form of irregular structures, such as graphs, where CNNs have limited applications.

To tackle the limitations of common CNNs, Graph Convolutional Networks (GCN)(defferrard2016, kipf2016, zhao2019) has been recently proposed. However it still have some issues when we consider a regression problem from 2D projection to 3D as in the pose estimation problem. The conventional GCN first works on nodes with arbitrary topologies but the learned kernel is shared for all edges and the internal structure of the graph is not completely exploited. Also, it only considers the first order neighbors of each node.

Semantic Graph Convolutions (SGC)(zhao2019) learn the semantic information encoded in a given graph. The idea here is to tackle the limitation of original GCN, learning channel-wise weights for edges in the graph, and then combine them with kernel matrices improving the power of graph convolutions. It also can be used in regression tasks. However, it is still have some drawbacks, for example, it does not analyse features in a specialized way to weight their relevance.

Inspired by the SGCs and Squeeze and Excitation Networks, we propose a novel gating mechanism applied to Semantic Graph Convolutions. We explore this architecture in a 2D to 3D regression applied to human pose estimation. Given a 2D human pose as input, we predict the locations of its corresponding 3D joints in a 3D coordinate space. We enhance the analysis of spatial correlation, which is crucial for understanding human actions(huang2020). Our network can learn both channel-wise weights for edges in the graph, combine them with kernel matrices, and understand global and channel interdependencies without using non-local layers. With this approach, we can achieve state-of-the-art performance considering the 2D to 3D regression, actually 4% better than the previous related works. Also, our model has 41% fewer parameters than the original SGC model.

Another issue that we study is the temporal coherence, which is not present in previous works since they do not consider the relationship between the processed frames. To track the detected persons in videos and 3D captures, we use sparse optical flow and a Kalman filter to smooth the movements. Figure 1.1 shows a teaser for all steps developed in our work following a framework for real time pose estimation.

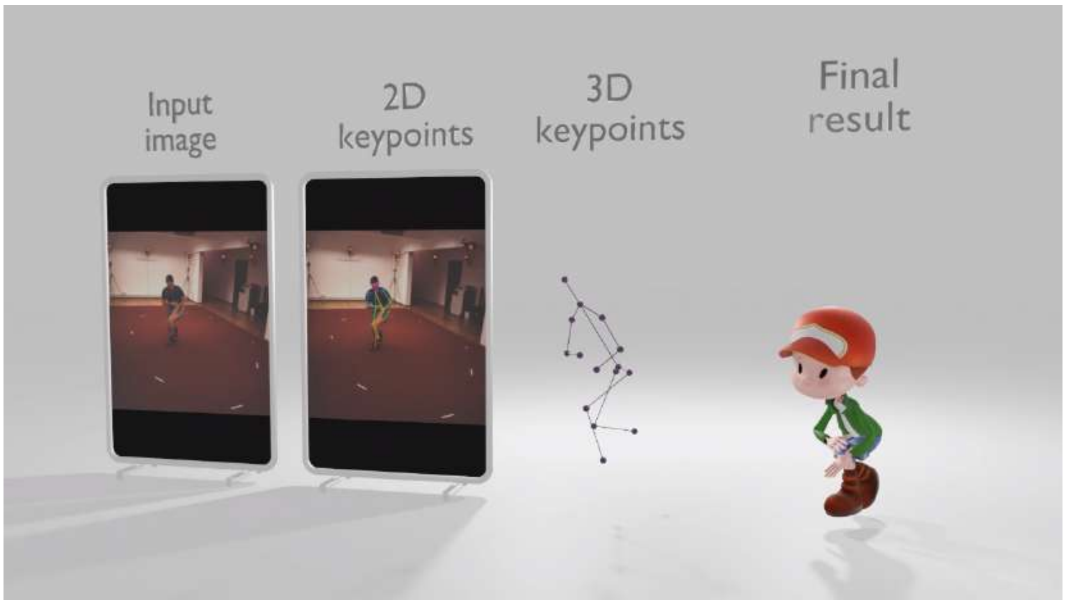

Figure 1.1: An example of our 3D human pose algorithm with a single RGB camera. The 2D data is captured with our 2D pose machine and after processed by the 3D neural network. It can be used in applications such as computer animation and game controlling.

As a result of this thesis, the methods presented here were published in the journal Computers & Graphics, in Eurographics, a top conference in the field, and in Brazilian Conference on Graphics, Patterns and Images (SIBGRAPI).

In summary, the main contributions of this thesis are:

– A novel deep neural network with streamlined architecture and tensor decomposition for 2D pose estimation with improved processing time;

– The extension of our model with Sparse Optical Flow and Kalman Filters for motion tracking in 2D;

– A novel attention layer for semantic graph convolution operations;

– A novel architecture for 3D pose estimation based on 2D to 3D joints regression.

2
Related Work

In this section we present related works that include the theoretical background of our techniques and also applications related to our domain.

2.1
Convolutional Neural Networks

Today several architectures of CNNs represents the state-of-the-art approaches for a wide range of visual tasks (krizhevsky2012imagenet, toshev2014deeppose, long2015fully, ren2015faster). CNNs are composed of layers of filters that represent neighbourhood spatial connectivity patterns. Its use of convolutions, non-linear activation functions and downsampling results in an hierarchical understanding of those features. One important aspect of this interleaving of operations is that they usually fuse spatial and channel-wise information. VGGNets (simonyan2014very) and Inception models (szegedy2015going) investigated very deep architectures in detail. Batch Normalization (BN) (ioffe2015batch) demonstrated how normalized data influences the training process, regulating the distribution of the inputs to the layers and smoothing the optimization manifold (santurkar2018does). A complementary approach is ResNets, that applied skip connections to also improve the training of deep networks (he2016deep), (he2016identity). Gates were introduced in Highway networks (srivastava2015training), which have the property of controlling the flow inside them through shortcut connections. Other works propose improvements to the learning by changing the connections between layers (chen2017dual), (huang2017densely). Another related research area is focused on methodologies for improvements in the computational aspect of the networks. For example, grouped convolutions increase the number of transformations that a network can learn (ioannou2017deep), (xie2017aggregated). Multi-branch convolutions are another approach in this matter, interesting because of their flexible compositions of operators (szegedy2015going), (ioffe2015batch), (szegedy2016rethinking), (szegedy2017inception). They can be interpreted as an extension of the grouping operator.

2.2
Attention in Neural Networks

In recent years, there has been an increasing interest by the machine learning community in attention models, especially due to their success in capturing long-distance interactions. It was introduced by for the encoder-decoder in a neural sequence translation model by Bahdanau et. al (bahdanau2014). In special, self-attentional Transformer architecture achieved state-of-the-art results applied to natural language process tasks as machine translation (vaswani2017). This architecture shows to be suitable for capturing long-distance relations between entities.

Considering feed-forward convolutional neural networks, Woo et al. (woo2018) presented an attention module where, given an intermediate feature map, the module tries to infer attention maps along the color channel and spatial dimensions, and after the attention maps are multiplied by the input feature map for adaptive feature refinement.

More recently, Bello et al. (bello2019) proposed 2D self-attention as a replacement for convolutions for image classification tasks, since self-attention captures global behavior more appropriately than convolutional layers, which are inherently local. Their results indicate that even though self-attention layers are successful in replacing convolutional layers for image classification, a combination of both techniques yields better results in a controlled environment (bello2019). However, this global form attends to all spatial locations of an input, limiting its usage to small inputs which typically require significant down sampling of the original image(ramachandran2019).

On the other hand, Squeeze-and-Excitation Networks (SE-Net) introduce an attention block for CNNs that improves channel interdependencies, not limiting the input size and presenting huge performance boost in accuracy when applied to existing architectures. Also there is almost no extra computational cost when using this kind of architecture. With its simple but effective idea of weight, each channel adaptively allows us to better interpret its outputs compared with other models, which do not have formal proof; for example, the transformer-based model presented by Bello et al. (bello2019).

2.3
Graph Neural Networks

Graph Neural Networks (GNN) was proposed as an framework to understand and explore graph structure relationships(wu2020, zhou2018, **?**). In GNNs, the representation vector of a node is computed by aggregating and transforming the data representation of its neighboring nodes. Xu et al.(xu2018) explore these concepts and have proved how powerful GNNs are. As an evolution of GNNs, Graph Convolutional Networks (GCNs)(defferrard2016, kipf2016) where introduced to deal with problems related to spectral and spatial perspective. GCNs have also limitations, for example, the kernel of an operation is shared by all nodes. Zhao et al. (zhao2019) propose the so-called Semantic Graph Convolutional Networks (SemGCN), a novel neural network architecture that operates on regression tasks with graph-structured data. It tries to capture the global and semantic information of nodes relationships. But, it also has some limitations; for example, it aims to approximate convolutions by learning a channel-wise weighting vector and for each spatial kernel it has a shared transformation matrix. However it does not consider interdependencies between channels. Veličković et al. (velivckovic2017) present the Graph Attention Networks, where they operate on graph-structured data. They use self-attentional methods by stacking layers, enabling implicitly different weights to different nodes in a neighborhood. However it does not consider regression problems and in our model, we aim to learn independent weights for the edges. Figure 2.1 show the mains structure of Graph Attention Networks.

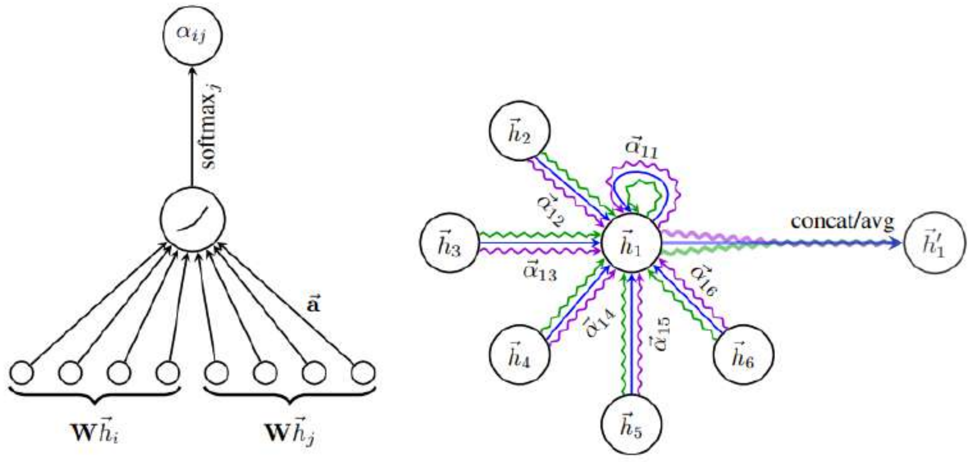

Figure 2.1: Graph Attention Networks. (velivckovic2017)

On the other hand, considering traditional convolutional neural networks, several papers aim to model global context for feature extraction (cao2019). "Squeeze-and-Excitation" (SE) networks are examples of it. Hu et al. (hu2018) propose the SE block, that analyzes channel-wise feature responses by explicitly modeling interdependencies between channels. We believe that this approach can be remodeled and adapted to other neural network techniques. Global context networks (cao2019) also try to adapt SE block to global context modeling in convolutional neural networks. In contrast with our formulation, it operates with non-local layers to process global and long-range relationships among nodes in the graph. It restricts the feature updating mechanism by computing responses between nodes based on their representations other than learning new convolution filters(zhao2019). This is out of our scope here since we just operate over structured graphs and over Semantic Graph Convolutions. As we will see in our experiments, our formulation surpasses the previous works in terms of reducing the projection error and drastically reducing the complexity of a 3D human pose regression network. We can achieve state-of-the-art performance with only 1/4 of computing operations needed by the work proposed by Zhao et al.(zhao2019).

2.4 Pose Machines and Tracking

Ramakrishna et al. proposed (ramakrishna2014) an inference machine framework and presented a method for articulated human pose estimation, called Pose Machines. A pose machine consists of a sequence of multi-class predictors, that is trained to predict the location of each part in each level of the hierarchy. Their method has the benefits of implicitly learning long-range dependencies between image and multi-part cues, tight integration between learning and inference, and a modular sequential design. The model was built on the inference machine framework so that it could learn reliable interconnections between body parts.

Wei et al. (wei2016) introduced the Convolutional Pose Machines (CPMs) for the task of pose estimation. CPMs expand the main idea of pose machine architectures and combine them with the advantages of convolutional neural networks. Their contributions were the ability to learn feature representations for both image and spatial context directly from data, and the ability to handle large training datasets efficiently. CPMs consists of a sequence of convolutional networks that repeatedly produce 2D confidence maps for the location of each body part. At each stage in a CPM, image features and the confidence maps produced by the previous stage are used as input. As said by Wei et al. (wei2016), the confidence maps provide the subsequent stage an expressive non-parametric encoding of the spatial uncertainty of location for each part, allowing the CPM to learn rich image-dependent spatial models of the relationships between elements. Figure 2.2 from Wei et al. (wei2016) show an example of the detection of heatmaps.

Cao et al. (cao2017) proposed an efficient method for 2D multi-person pose estimation with high accuracy on multiple public datasets, extending the original Convolutional Pose Machines. They created a bottom-up representation of association scores via Part Affinity Fields(PAFs), a set of 2D vector

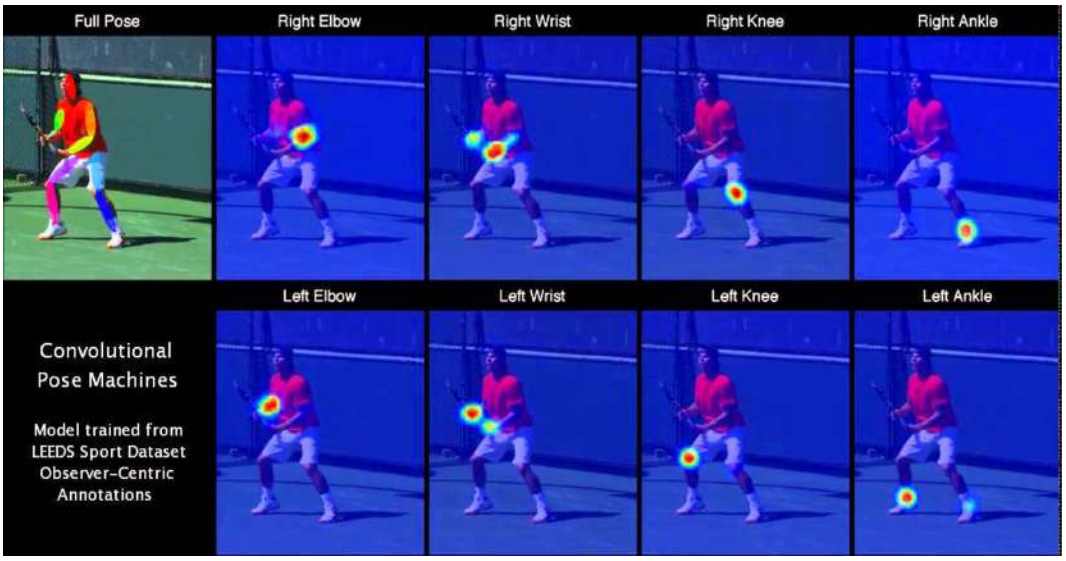
Full Pose
Right Elbow
Right Wrist
Right Knee
Right Ankle
Left Elbow
Left Wrist
Left Knee
Left Ankle
Convolutional Pose Machines
Model trained from LEEDS Sport Dataset Observer-Centric Annotations

Figure 2.2: Heatmap detections in Wei et al. (wei2016)

fields that encode the location and orientation of limbs over the image domain. They show that their approach encodes global context sufficiently well to allow for a combinatorial optimization algorithm that solves an assignment problem to detect a person's skeleton. However, they did not consider temporal coherence in the original work. When considering a video, there is no relation between persons detected in multiple frames. Also, this approach is computationally intense, demanding a lot of processing power. Several authors refers OpenPose as being one of the most lightweight multi-person 2D keypoint detectors in the literature, however, following their algorithm and considering their benchmarks, we achieved approximately only 10 FPS using an Nvidia Titan Xp GPU. Figure 2.3 from Cao et al. (cao2017) presentation in CVPR, show an example of the detection of heatmaps and also the PAFs.

In regards to the tracking problem, a solution is to use dense optical flow to adjust the predicted positions to generate a smooth movement across frames. The Thin-Slicing Network (song2017) achieved excellent results. However, this system is computationally very intensive and is slower when compared with the proposed by other papers.

Luo et al. (luo2018) achieved some improvements in terms of both accuracy and efficiency. They propose a recurrent CNN model with LSTM for video pose estimation. Although considering occlusion, some incorrect predictions can be detected when the joint is not visible for an extended period. Also, their tests do not consider multi-person pose estimation.

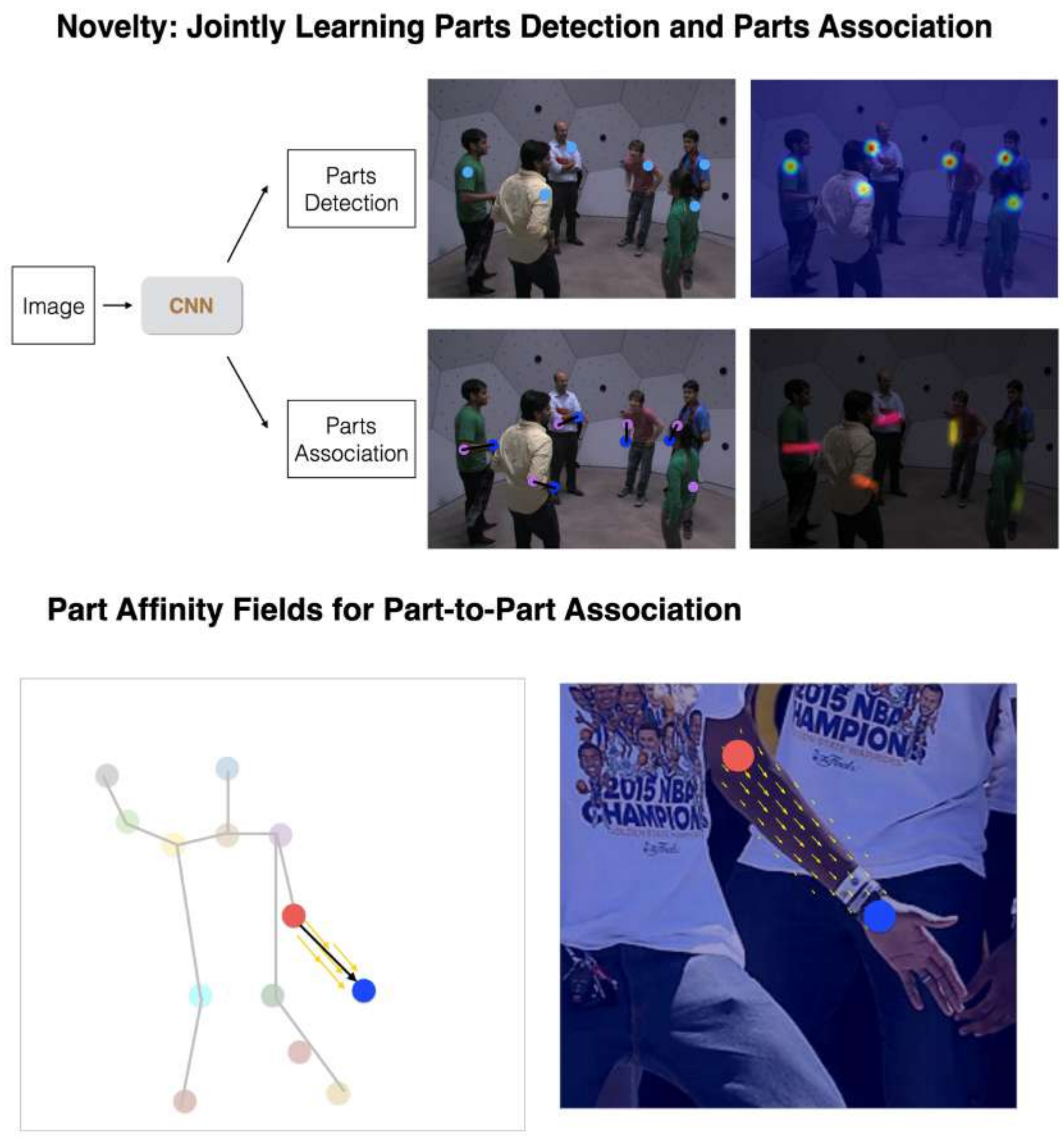
Novelty: Jointly Learning Parts Detection and Parts Association
Image
CNN
Parts Detection
Parts Association
Part Affinity Fields for Part-to-Part Association

Figure 2.3: PAFs and body points dectection by Cao et al. (cao2017)

2.5 3D Pose Estimation

Following 2D pose estimation models' success, several papers propose an end-to-end model to predict the 3D human pose given images in the wild. One of these papers, with the best results in terms of accuracy, is proposed by Wang et al. (wang2019). They present a 3D Human Pose Machine with Self-supervised Learning, where they developed a multi-stage system composed of three neural network models, involving two dual learning tasks. They generate transformations for 2D-to-3D pose and 3D-to-2D pose projection. The 2D-to-3D pose model regress intermediate 3D poses by transforming the pose representation from the 2D domain to the 3D domain, receiving as input the features extracted by a 2D pose machine. In contrast, the 3D-to-2D pose projection contributes to refining the intermediate 3D poses. However, their approach seems to be very costly, considering computational resources, wherein a high-end GPU, it takes 51 milliseconds to predict a single pose. Figure 2.4 show the archicteture of this approach from wang et al. (wang2019).

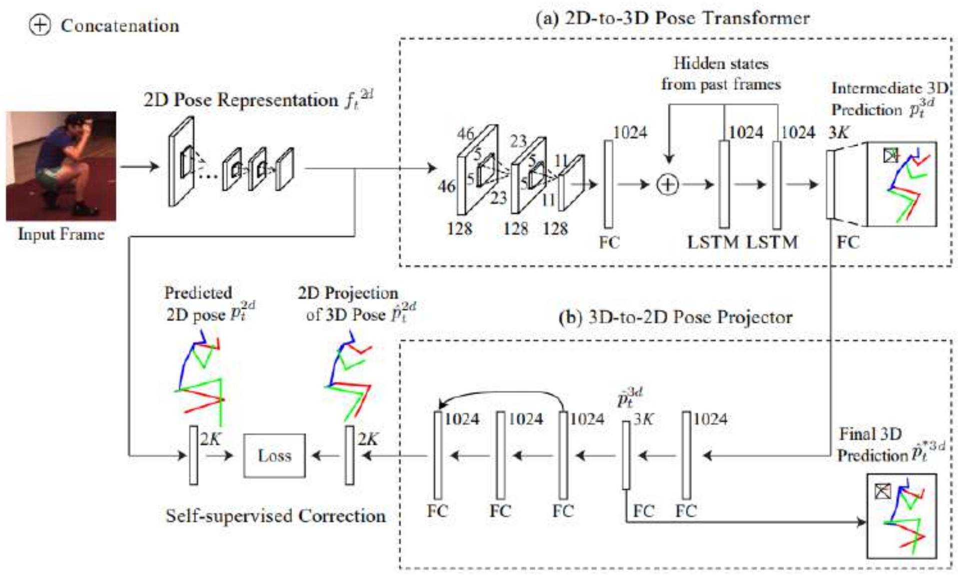
⊕ Concatenation
(a) 2D-to-3D Pose Transformer
2D Pose Representation f_t^{2d}
Input Frame
Hidden states from past frames
Intermediate 3D Prediction p_t^{3d}
46
23
11
5
128
1024
3K
FC
LSTM LSTM
Predicted 2D pose p_t^{2d}
2D Projection of 3D Pose $\hat{p}_t^{2d}$
(b) 3D-to-2D Pose Projector
2K
Loss
Final 3D Prediction $\hat{p}_t^{*3d}$
Self-supervised Correction

Figure 2.4: 3D Human Pose Machine with Self-supervised Learning by Wang et al. (wang2019)

In a different approach, Martinez et al. (martinez2017) propose a simple feedforward neural network that receives the 2d joint locations and predicts 3d positions. They “lift” the ground truth 2d joint locations to 3d space. However, this also has limitations; for example, it highly depends on the quality of 2D detections and does not maintain the bone proportion for all bodies.

Mehta et al.(mehta2017) present the first real-time method to capture the 3D pose in a stable, temporally consistent manner using a single RGB camera. Their formulation uses a regression for 2D to 3D projection in real-time and creates a kinematic skeleton fitting method for coherent kinematic analysis. The Figure 2.5 from Mehta et al.(mehta2017) show the main aspects of this approach.

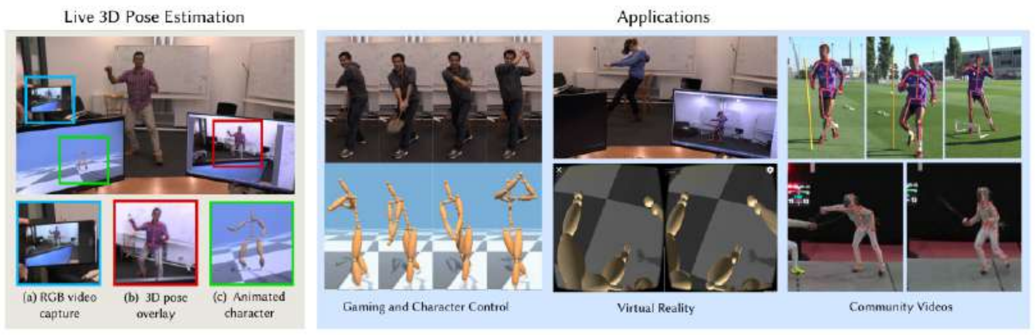

Figure 2.5: Model proposed by Mehta et al.(mehta2017) in the Vnect project

Zhao et al. (zhao2019) has expanded the proposal of Martinez et al. (martinez2017), where they use Semantic Graph Convolutions for regress the 3D pose from 2D data. However, their architecture can be also extended to use attention modules and to reduce the projection error. In our approach, to solve previous issues about reprojection error and network complexity, we propose a way to model interdependencies between features in Semantic Graph Convolutions.

2.6 Motion Capture

Zhou et al.(zhou2018) present a markerless motion capture system that combines 2D, 3D, and temporal information to infer 3D geometry. They consider the 2D joint locations as latent variables given by a deep, fully convolutional neural network. They model the 3D poses with sparse representation, and the 3D estimated parameters are processed via an Expectation-Maximization algorithm. Besides, this approach is also computationally intensive for limited hardware.

The XNet(mehta2020) is an evolution of the work proposed by Mehta et al.(mehta2017) which also predict the 3D pose of Humans and even infer the bones rotations. They claim that this is the first 3D motion capture system that uses a single monocular RGB camera. They present a real-time approach for multi-person 3D motion capture at over 30 fps. However, their model seems computationally costly and needs a high-end GPU to get the desired performance. Figure 2.6 by Mehta et al.(mehta2020) show the applications of this technique.

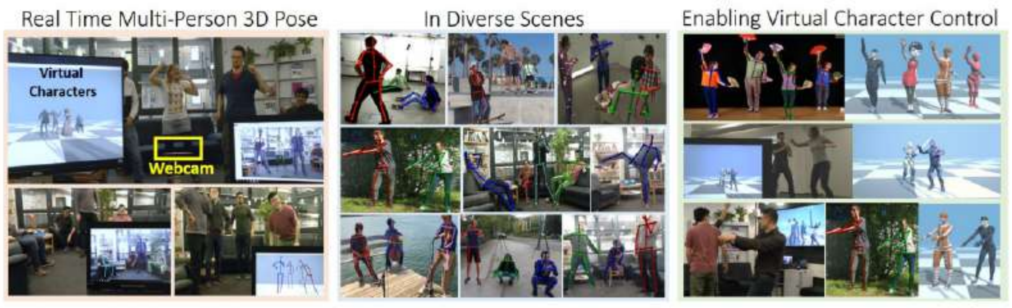
Real Time Multi-Person 3D Pose
Virtual Characters
Webcam
In Diverse Scenes
Enabling Virtual Character Control

Figure 2.6: Model proposed by Mehta et al.(mehta2020) as an evolution of the Vnect project.

The self-supervised learning of motion capture proposed by Tung et al. (tung2017) presented an optimization architecture for neural network weights that predict the 3D shape and skeleton configurations are given a monocular RGB video. Their model is trained using a combination of supervised learning from synthetic data and self-supervision from a differentiable rendering pipeline for the 3D skeleton and human shape. Although the impressive results for self-supervised learning in this task, this model still has less accuracy than traditional models. Figure 2.7 from the original paper presents the aspects of this technique (tung2017).

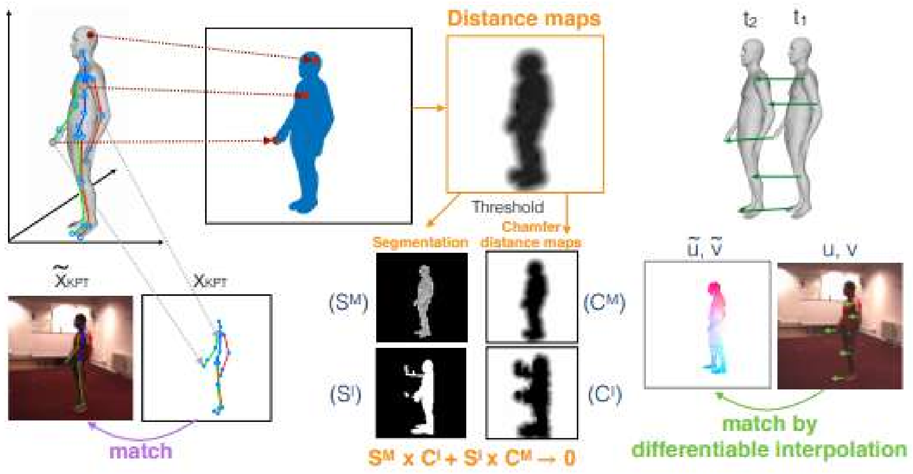

Figure 2.7: Model proposed by Tung et al. (tung2017) presenting an optimization architecture for neural network and a differentiable rendering pipeline.

In contrast with these previous works, we aim to present a lightweight framework for computer animation using 3D human pose estimation. To this, our model does not needs any specialized hardware or even high-end GPU configurations. Also our model focus only on predict the 3D keypoints location, leaving the rotation information as a future work.

3 Tensor Decomposition

This section presents Tensor analysis and its relation with Convolutional Neural Networks. This can lead us to better understand the connection between the HOSVD theory and the factorized convolutions. There is a rising interest in exploring efficient architectures for Neural Networks, either for use in embedded device applications or for better allocation of resources and services, reducing network latency and size (howard2017). Several papers propose different architectures for Convolutional Neural Networks based on factorized convolutions (howard2017, wang2017, jin2014), and according to our opinion, the hidden theory behind these applications is related to tensor algebra and high order decompositions; although few papers discuss these fundamentals. In the next section we will describe the matrix and tensor decomposition fundamentals.

3.1 Matrix Decomposition

Matrix decomposition is an essential task to perform improvements in several applications. Using such decomposition's can not only reduce processing in numerically efficient algorithms, but also,reduce the space needed to store data. In this section we demonstrate the relations between tensor decomposition and 2D matrix decomposition. Before we start explaining the mathematical concepts of matrix factorization, we need to contextualize the importance of this technique and the factor analysis. For example, in the Spearman's hypothesis, Charles Spearman, a British psychologist, propose that human intelligence can be decompose following 2 hidden factors: eductive (the ability to make sense out of complexity) and reproductive (the ability to store and reproduce information)(rabanser2017). He propose a set of tests for students where their scores were noted in a matrix M. Following his hypothesis, this matrix can be decomposed in 2 smaller matrices where each one can be evaluated for each student. This hypothesis is presented in Figure 3.1.

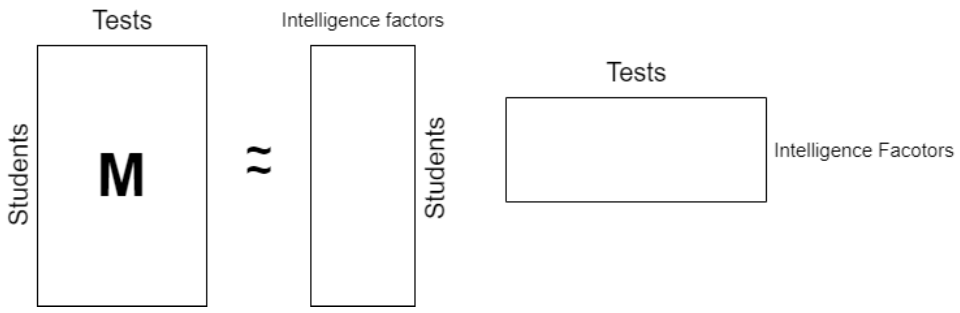

Figure 3.1: Spearman's hypothesis for intelligence factor analysis.

Let's analyse the mathematical concepts for matrix decompostion. Considering a rank decomposition, we can approximate a matrix M as in equation 3-1

$$M = AB^T, \tag{3-1}$$

where $A \in \mathbb{R}^{n \times r}$ and $B \in \mathbb{R}^{r \times m}$. Here we want to approximate M by a matrix M' of a lower rank minimizing the error between then. However the decomposition is not unique. For example, let us consider a rotation problem. We add a invertible rotation matrix R and it's inverse R^{-1} in the equation 3-1. The following equation 3-2 represent this transformation:

$$M' = ARR^{-1}B^T = (AR)(R^{-1}B^T) = (AR)(BR^T)^T = A'B'^T \tag{3-2}$$

As we can see, we can again construct two matrices A' and B'^T using this low rank approximation where $rank(M') = 2$. Generally the rank decomposition of a matrix is highly non-unique, where matrix decompositions are only unique under very stringent conditions, such as the conditions of a singular value decomposition (SVD). Before understand the SVD decomposition, let's analyse another related technique called eigen decomposition and it's limitations. Consider a symmetric matrix A with size $n \times n$. we can decompose this matrix if it is orthogonally diagonalizable as in equation 3-3:

$$A = V \Lambda V^{-1}, \tag{3-3}$$

where Λ is an $n \times n$ diagonal matrix with n eigenvalues of matrix A and V is a matrix where $V \in \mathbb{R}^{n \times n}$. The columns of V are linearly independent and correspond to the eigenvectors of A. However this decomposition only exists under some restrictions, described by the spectral theorem and the following itens:

– the eigenvectors needs to be linearly independent

– matrix A need to be squared

– V is invertible

Considering the limitations of the eigen decomposition, now we will show another approach, called singular value decomposition (SVD), which allows an exact representation of any matrix. With this technique we can easily eliminate less important parts of this representantion to produce and approximation of the original matrix with any number of dimensions. Let consider a non-symmetric matrix A. To calculate the SVD, we need to find the eigenvalues and eigenvectors of AA^T and A^TA. As in equation 3-4 we can see that AA^T is an $n \times n$ symmetric matrix:

$$(AA^T)^T = A^T(A^T)^T = A^TA \tag{3-4}$$

These two matrices represents a special role in linear algebra: they are positive semidefinite, orthonormal, symmetric and squared. Also they has the same eigenvalues and same rank of A. Considering this, the SVD theorem can be represented by equation 3-5

$$A_{n\times m} = U_{n\times n}S_{n\times m}V^T_{m\times m}, \tag{3-5}$$

where the eigenvectors of AA^T make up the columns of U and the eigenvectors of A^TA, the columns of V. S is a diagonal matrix containing the squared root of the eigenvalues of AA^T and A^TA. The singular values are the diagonal entries of the S matrix and are arranged in descending order. The singular values are always real numbers. If the matrix A is a real matrix, then U and V are also real.

We can make an approximation of the original matrix by truncating values where the resulting matrix will be quite close to the original. Suppose we want to represent a very large matrix A by its SVD components. The best way to reduce the dimensionality of the three matrices is to set the smallest singular values to zero. If we set the s smallest singular values to 0, then we can also eliminate the corresponding columns of U and V. The quality of this approximation can be evaluated using the frobenius norm.

Also is well know that with very large matrix sizes, computing the SVD may be prohibitive. The actual computational cost of a full SVD operation is $O(mn^2)$. Considering this we explore another approach that gives a computationally efficient way to compute a rank-r approximation: the randomized SVD. This technique has two major steps, where in the first one we multiply a matrix X by a Gaussian random matrix Ω in order to have random linear combinations of the columns of X.After we compute a QR factorization to obtain a good approximation of X, where $X \approx QQ^TX$. After, in a second stage we compute the SVD of a much smaller matrix defined by $Q^TX = \hat{U}_B\hat{\Sigma}\hat{V}^T$, truncate to the desired rank r and calculate $\hat{U} = Q\hat{U}_B$.

Also truncating the $\hat{\Sigma}$ and $\hat{V}$ we obatain the final approximation defined by $\hat{X} = \hat{U}\hat{\Sigma}\hat{V}^T$.All computational cost of this approach is defined by equation 3-6

$$Cost = 2(r+p)O(NonZeros(X)) + O(r^2(m+n)), \tag{3-6}$$

where p is oversampling parameter $p \geq 0$ such that $r + p \leq min\{m, n\}$. The algorithm 1 show all steps in the process to calculate the randomized SVD.

```
Algorithm 1: Randomized SVD($X, r, p$)
    Input: matrix $X \in \mathbb{R}^{m\times n}$, rank $r$, oversampling parameter $p$
    Output: matrices $\hat{U} \in \mathbb{R}^{m\times r}, \hat{\Sigma} \in \mathbb{R}^{r\times r}, \hat{V} \in \mathbb{R}^{n\times r}$
  1 Generate Gaussian random matrix $\Omega \in \mathbb{R}^{n\times(r+p)}$;
  2 $Y \leftarrow \Omega X$;
  3 $QR$ factorization of $Y$ where $Y \leftarrow QR$;
  4 $B \leftarrow Q^T X$;
  5 Calculate $SVD$ of $B$ where $B \leftarrow \hat{U}_B\hat{\Sigma}\hat{V}^T$;
  6 $\hat{U} \leftarrow Q^T\hat{U}_B[:, 1:r]$;
  7 return $\hat{U}, \hat{\Sigma}[1:r, 1:r], \hat{V}[:, 1:r]$;
```

In the next section we will demonstrate the relation between this technique and Tensor decomposition, which is one of the building blocks of our approach. A 2D convolutional layer of a neural network can be described as a 4-dimensional tensor, where its dimensions are defined by the number of columns and rows, the number of channels of an input image, i.e., the RGB channels, and the number of output channels. In this work, we are interested in improving the pose estimation network for fast applications. To do so, we decompose convolutional layers into smaller tensors, i.e., we made approximations of these layers. This approximation is essential since it reduces the number of operations performed by each layer and, in this way, it accelerates our neural network model. The following sections will discuss general aspects of tensors as well the Tucker decomposition (tucker1966, kim2015, kolda2009, smith2017).

3.2 Tensor Properties

A *tensor* can be seen as a high-dimensional matrix, i.e, with three or more dimensions. The order of a tensor $\mathcal{T}$ is the number of its dimensions, also known as ways or modes (kolda2009). In a manner analog to matrices' rows and columns, tensors have fibers. A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber (kolda2009, smith2017). Third-order tensors have column, row, and depth fibers for example.

In tensor analysis, there are operators that creates subarrays by fixing some of the given tensor's indices. In this way, when we fix all indexes leaving only one free, we create a *fiber*. In the same way, we create *slices* if we fix all indexes but leaving two free. For a third order tensor, the frontal, lateral and horizontal slices are obtained by fixing the third, the second and the first indexes, respectively.

Unfolding is similar to flattening a matrix, where we stack the fibers of a tensor in a given way to obtain a matrix representation (cichocki2009, tucker1966, kolda2009, de2000). For a tensor $\mathcal{T}$ with dimensions $(I_1, I_2, ..., I_n)$, its mode-n unfolding of $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is a matrix $\mathbf{T}_{[n]} \in \mathbb{R}^{I_n, I_M}$, with $M = \prod_{\substack{k=1, \\ k \neq n}}^{N} I_k$, mapping from element $(i_1, i_2, \cdots, i_N)$ to (i_n, j), with $j = \sum_{\substack{k=1, \\ k \neq n}}^{N} i_k \times \prod_{m=k+1}^{N} I_m$ (kolda2009).

For example, if we have a 3D tensor $\mathcal{T}$ with dimensions $(3, 3, 2)$ defined by the frontal slices:

$$T_1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, T_2 = \begin{bmatrix} 10 & 11 & 12 \\ 13 & 14 & 15 \\ 16 & 17 & 18 \end{bmatrix}$$

the mode-1 unfolding matrix is by definition given by:

$$\mathbf{T}_{[mode-1]} = \begin{bmatrix} 1 & 10 & 2 & 11 & 3 & 12 \\ 4 & 13 & 5 & 14 & 6 & 15 \\ 7 & 16 & 8 & 17 & 9 & 18 \end{bmatrix}$$

An outer product is the product of all elements of a vector. For example, an outer product of two vectors a and b is defined by equation 3-7 producing a 2D matrix X.

$$X = a \circ b = ab^T \tag{3-7}$$

A rank-one tensor is a mode-N tensor where it can be seen as the outer product of N vectors (kolda2009, smith2017) as we can see in Equation 3-8. In other words, a tensor can be strictly decomposed into into the outer product of N vectors where each element of $\mathcal{T} \in \mathbb{R}^{I_1 x I_2 x \ldots I_N}$ is the product of the corresponding vector elements defined by Equation 3-9.

$$\mathcal{T} = \mathbf{v}^{(1)} \circ \mathbf{v}^{(2)} \circ ... \mathbf{v}^{(N)} \tag{3-8}$$

$$t_{i_1 i_2 \ldots i_n} = v_{i_1}^{(1)} v_{i_2}^{(2)} ... v_{i_N}^{(N)} \tag{3-9}$$

Figure 3.2 show the process to generate a rank-1 mode-3 tensor with 3 vectors.

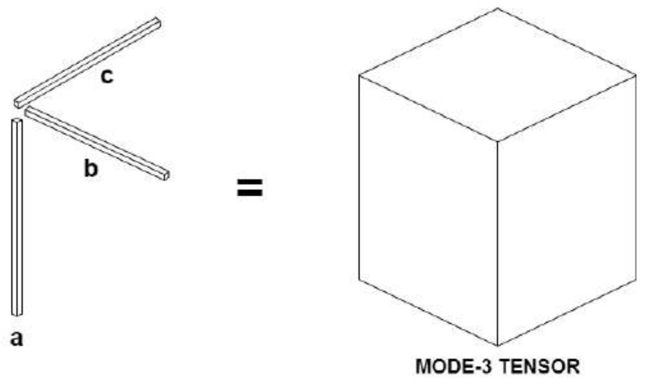

Figure 3.2: A rank-1 mode-3 tensor.

A rank of a tensor $\mathcal{T}$ is the smallest number of rank-one tensors that generate $\mathcal{T}$ by computing their sum (kolda2009). A tensor $\mathcal{T}$ of N-order is given by the equation 3-10

$$\mathcal{T} = \sum_{r=1}^{R} c_r a_r^1 \circ a_r^2 \circ \cdots \circ a_r^n = C(A^1, A^2, \cdots, A^N), \tag{3-10}$$

where the matrices A are called factor matrices and hold the combination of the vectors from the rank-one components as columns. The parameter C is often used to absorb the respective weights during normalization of the factor matrices' columns. This usually means normalizing the sum of the squares of the elements in each column to one. This is an special property and crucial to understand the tensor decomposition algorithm in the next section.

Also another important property is the $n - rank$ of a tensor, which should not be confused with the rank of a tensor $\mathcal{T}$. The $n - rank$ is the dimension of the vector space spanned by the mode-n fibers (kolda2009). Let $R_n = Rank_n(\mathcal{T})$ for $n = 1, \cdots, N$, we say that the tensor $\mathcal{T}$ is a $rank - (R_1, R_2, \cdots, R_n)$ tensor where $R_n \leq I_n$ for $n = 1, \cdots, N$. To select the rank-n we analyse each $mode - n$ unfolding of $\mathcal{T}$, in other words, we use each matrix representation following the stacked fibers.

The product of a mode-n tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ by a matrix $M \in \mathbb{R}^{J_n \times I_n}$ is a tensor $\mathcal{X}$ with dimensions $(I_1 \times I_2 \times \ldots \times I_{n-1} \times J_n \times I_{n+1} \times \ldots \times I_N)$ given by Equation 3-11 (symeonidis2010).

$$\mathcal{X} = (\mathcal{T} \times M)_{(I_1 \times I_2 \times \ldots \times I_{n-1} \times J_n \times I_{n+1} \times \ldots \times I_N)} = \sum_{i_n} t_{i_1 \ldots . i_N} m_{j_n i_n} \tag{3-11}$$

3.3 Kruskal form of a Tensor

Another property is the Kruskal form of a tensor. It is similar to matrix properties where the sum of the outer products of two vectors can generate a matrix M. Let us consider a single tensor $\mathcal{T}$. We can generate it as a sum of outer products of N vectors, where the number of terms in the sum is called the Kruskal rank of the tensor (kolda2009).

3.4 Hadamard Product

The Hadamard product is an operation that takes two tensors of same dimensions and produces another tensor of the same dimension through an element-wise multiplication. For example, consider a mode-2 Tensor, i.e., a 2D matrix. The Hadamard Product of 2D Tensors if defined as follows in equation 3-12:

$$A \circ B = \begin{bmatrix} a_{1,1}b_{1,1} & \cdots & a_{1,n}b_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1}b_{m,1} & \cdots & a_{m,n}b_{m,n} \end{bmatrix} \tag{3-12}$$

3.5 Kronecker Product

The Kronecker product is a generalization of the outer product which takes two tensors of arbitrary size. The Kronecker Product of 2D Tensors if defined as follows in equation 3-13:

$$A \otimes B = \begin{bmatrix} a_{1,1}B & \cdots & a_{1,n}B \\ \vdots & \ddots & \vdots \\ a_{m,1}B & \cdots & a_{m,n}B \end{bmatrix} \tag{3-13}$$

3.6 Khatri-Rao Product

The Khatri-Rao Product of two matrices corresponds to the column-wise Kronecker product of these matrices as in equation 3-14:

$$A \odot B = [a_1 \otimes b_1, \cdots, a_n \otimes b_n] \tag{3-14}$$

3.7 Tucker Decomposition

A Singular Value Decomposition of a 2D matrix, i.e, a 2-order tensor, can be defined as in equation 3-5 where U is a unitary matrix $m \times m$, Σ is a rectangular diagonal matrix $m \times n$ and V^T is the transpose conjugate $n \times n$. This, equation can be rewritten as follows, in Equation 3-15 (symeonidis2010):

$$M = C \times U^1 \times U^2, \tag{3-15}$$

where U^1 is defined as $(u_1^1 \times u_2^1 \times ... \times u_{I_1}^1)$ and U^2, defined as $(u_1^2 \times u_2^2 \times ... \times u_{I_1}^2)$, where both are unitary matrices, and C is a core matrix $(I_1 \times I_2)$.

The SVD decomposition can be generalized to High Order Tensors, where this approach is also know as High Order SVD or Tucker decomposition (symeonidis2010). Such mathematical tool considers the orthonormal spaces associated with the different modes of a tensor (kim2015). For example, Equation 3-15 can be extended for 3D tensors as follows:

$$\mathcal{M} = C \times U^1 \times U^2 \times U^3, \tag{3-16}$$

where U^1, U^2, U^3 contains the 1-mode, 2-mode, and 3-mode singular vectors, respectively, related to the column space of M_{mode-1}, M_{mode-2}, and M_{mode-3} matrix unfoldings. C is a core tensor with orthogonality property (symeonidis2010).

This technique can be generalized for n-dimensional tensors. Let's consider the tensor $\mathcal{M} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_n}$ where d is the order of the tensor. The Higher Order Singular Value Decomposition of this tensor is defined by the following equations 3-17 and 3-18

$$C = (U_1^T, U_2^T, \cdots, U_d^T)\mathcal{M}, \tag{3-17}$$

$$\mathcal{M} = (U_1, U_2, \cdots, U_d)C, \tag{3-18}$$

where each $U_k \in \mathbb{R}^{n_k \times n_k}$ is an unitary matrix containing a basis of the left singular vectors of the standard factor-k flattening $M_{(k)}$ of $\mathcal{M}$. The jth column of U_k corresponds the the largest jth singular value of matrix M_k as in SVD. The HOSVD algorithm 2 is described in algorithm where to compute the U_k

matrices we use the randomized SVD to reduce the complexity.

Algorithm 2: Randomized HOSVD(X, r, p)

```
Input: Tensor $\mathcal{M} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}$, target rank vector $r \in \mathbb{N}^d$,
       oversampling $p$ where
       $r_i + p \leq min\left\{I_i, \prod_{j \neq i} I_j\right\} forj = 1 \cdots d$ and $p \geq 0$
Output: matrices $\hat{U}$ and core vector $C$
1 $C \leftarrow \mathcal{M}$;
2 $U = []$;
3 for $i$ = $1{:}d$ do
4     $M_i \leftarrow flatten(\mathcal{M}, i)$;
5     $[\hat{U}, \hat{S}, \hat{V}] = \text{RandSVD}(M_i, r_i, p)$;
6     $U_i \leftarrow \hat{U}$;
7     $C = C \times U_i^T$
8 end
```

Also, as in the SVD for matrices, the HOSVD has a truncated form by selecting and arbitrary value for the rank selection. For each matrix we can reduce the dimensionality and the number of operations needed. Figure 3.3 show this operation.

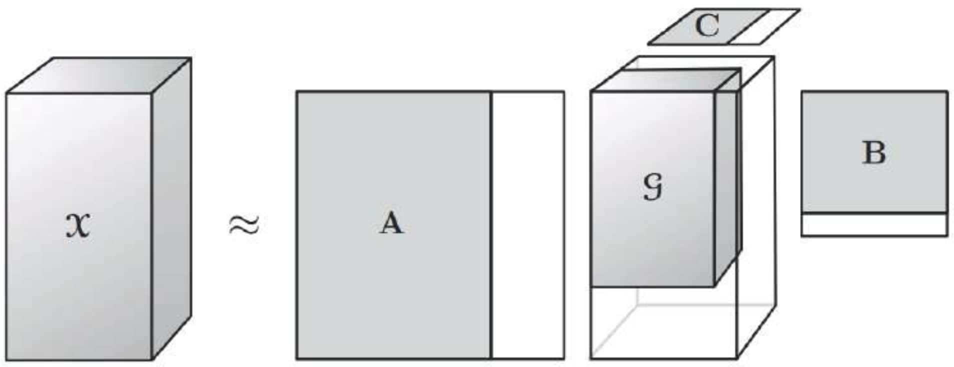

Figure 3.3: Truncated form of HOSVD (kolda2009)

Tensor decomposition has several applications. For example in Figure 3.4, we present an example were we apply HOSVD to compress volumetric data. As we can see here, the quality of the decomposition and approximation highly depends to the ranks selected.

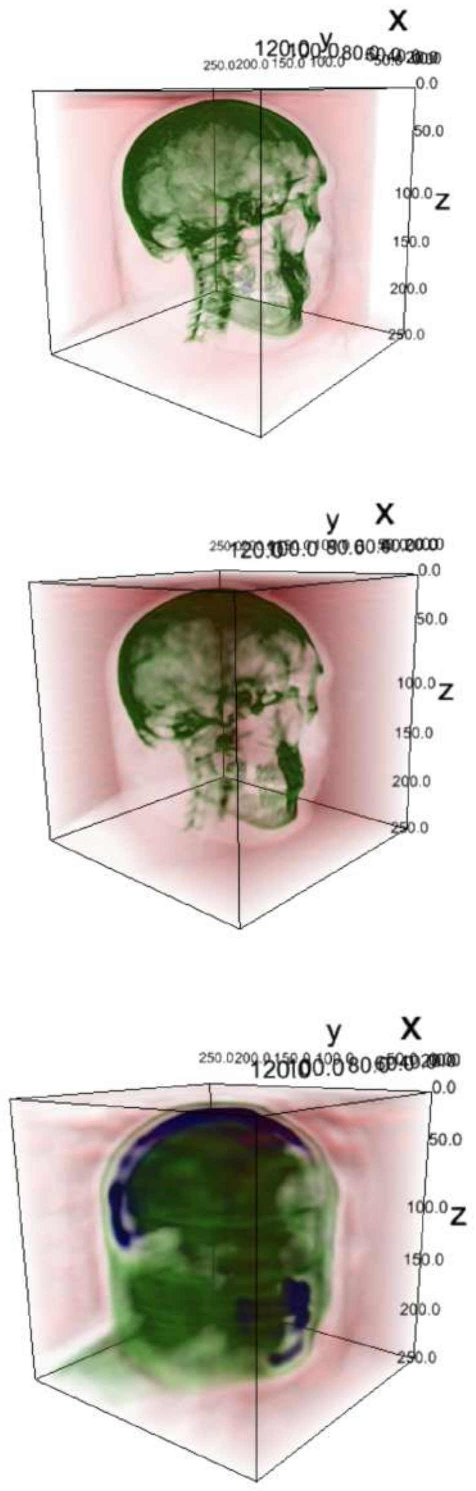

Figure 3.4: HOSVD applied to a volumetric data from MRI. The first Figure show the orignal data, the second the data reconstructed after a decomposition using target rank values of 50 and in the third Figure, target rank values of 10. The error between the original data and the approximations was 0.09 and 0.25 respectively. Also, the decomposition highly reduce the space needed to store the data, were it was by half in the second image and by 1/3 in the third.

We could now apply these Tucker decomposition concepts to our domain (kim2015). Consider a mode-4 kernel tensor $\mathcal{T}$, which can be seem as a kernel in a convolution layer. All operations for its decomposition can be written in the form described in Equation 3-19 (kim2015):

$$\mathcal{T}_{x,y,z,k} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} \sum_{r_4=1}^{R_4} C_{r1,r2,r3,r4} U^1_{x,r_1} U^2_{y,r_2} U^3_{z,r_3} U^4_{k,r_4}, \tag{3-19}$$

where C is a core tensor of size $(R_1 \times R_2 \times R_3 \times R_4)$ and the U_* matrices are factor matrices of sizes $X \times R_1$,$Y \times R_2$,$Z \times R_3$, and $K \times R_4$, respectively. Each dimension, in our domain, is associated with the dimensions defined by the number of columns and rows, RGB channels and the number of output channels in a convolutional layer of a neural network. Although this is true considering as input an RGB image, in successive layers the dimensions and number of channels vary. In this way, in intermediate layers, our tensors can be defined by the spatial dimensions X and Y of the input, the number of filters of the input data, i.e. the "depth", and the number of filters to the output data of the layer. Considering this, the decomposition is performed over the weights of each layer of convolution in our Neural Network and, as we can see, the convolution kernels are 4D tensors. We can boost the speed performance of a neural network by creating several factorized approximations of regular convolutions with only a few reductions in accuracy. The next chapter presents the details of our model as well its relation with tensor decompositions.

4
Neural Network Architectures for 2D Pose estimation

In this chapter, we describe the main aspects of our deep convolutional neural network models for 2D pose estimation. At first, we developed 2 different 2D neural network for 2D human pose estimation. A partially decomposed model with temporal coherence and a full factorized model with attention modules. In both techniques we use Tensor decomposition to improve the processing time.

4.1
2D Pose Estimation Model

This section describes the main aspects of the architecture proposed by Cao et al. (cao2017). The network architecture consists of a feedforward neural network that predicts a set S of 2D “confidence maps", from which the body parts are located in an image, and also a set L of 2D vector fields that help identify the connection between two body parts to generate a skeleton. The set $S = \{S_1, S_2, ..., S_n\}$ has n “confidence maps" one for each part of the body (hand, elbow, head, etc) and the set $L = \{L_1, L_2, ..., L_C\}$ has C 2D vector fields, each used to construct the skeleton, identifying some member of it: an arm or leg for example. N candidates for each body part are generated, creating for each pair a bipartite graph. Finally, the results are analyzed following a greedy strategy, using the Hungarian algorithm with some modifications to create a connection between two parts. This process is performed in each frame.

The neural network is divided into two branches: one responsible for generating the “confidence maps", and the other, the affinity fields. Also, these two branches are divided into t stages, where the authors of the original paper set $t = 6$ stages. At first, the images are processed by the first ten layers of a VGG-19 convolution network, generating a set of F features that will be used in the subsequent stages of the OpenPose. These features are used as input for the other layers of the network and, as result, they have a set S of confidence maps and a set L of Affinity Fields (vector fields). Let ρ and Φ represent the convolution layers for the confidence maps and affinity fields. At the end of this step, the results are concatenated with the initial set F to produce refined results. This process is performed for each subsequent stage.

To guide the network in its predictions, two objective functions are given at the end of each stage, one for each branch. The objective functions for both branches are based on the distance between the network output data and the ground truth generated for the confidence maps and vector fields, according to Equations 4-1 and 4-2 (cao2017):

$$f_S^t = \sum_{j=1}^{J} \sum_{p} W(p) \, ||S_j^t(p) - S_j^*(p)||_2^2; \tag{4-1}$$

$$f_L^t = \sum_{c=1}^{C} \sum_{p} W(p) \, ||L_c^t(p) - L_c^*(p)||_2^2, \tag{4-2}$$

where S_j^* and L_c^* represent the ground truth of the annotated data; J and C are, respectively, the confidence maps for each part of the body, and the vector fields. The $W(p)$ is a binary mask with $W(p) = 0$ when there is no data annotated at the point p of image. The mask is used because there are images in the training dataset that are not entirely annotated, so it prevents true positive data from being penalized. The overall loss function is defined by the sum of Equations 4-1 an 4-2.

The training images, as well as their annotations, were obtained through the COCO (Common Objects in COntext) dataset (lin2014), using more than 100000 images for the task.

Following the architecture model, they first analyze the set S to identify the possible candidates for a part of the body. In total, 18 confidence maps are generated by branch 1 defined by the convolution network ρ. Each confidence map of the set S can be viewed as a heatmap.

For this, they used a technique called Non-maximum suppression to identify the position of potential candidates by getting the peaks in the image. With all possible candidates (points) obtained, it is necessary to find the real connections between them. Therefore, for each connection, is built a complete bipartite graph where each node is a part of the body, and each edge is a connection representing an arm or a leg, for example. To find the right relationship, they face a well-known problem from graph theory that is finding the best match between the vertices of a bipartite graph: an assignment problem. For this task, the model uses the vector fields that were generated by the output of Φ of the network. An integral line is computed along the segment described by each pair of candidates, and since the integral line shows the influence of a given field along a line. Equation 4-3 shows how they compute the score of each segment represented by each pair of body parts. Given two points, d_1 and d_2, as candidates for two parts of a skeleton forming a connection, the score of an edge connecting these two points is:

$$E = \int_{u=0}^{u=1} L_c\left(p\left(u\right)\right).\frac{d_{j2} - d_{j1}}{||d_{j2} - d_{j1}||_2} du, \tag{4-3}$$

where L_c represents the vector field and $p(u)$ interpolates the position between two body parts d_{j1} e d_{j2},

$$p\left(u\right) = \left(1 - u\right) d_{j1} + u d_{j2}. \tag{4-4}$$

Afterward, to obtain the optimal solution, they used a classical method called Hungarian Algorithm. Regarding the detection of multiple people, determining the pose of each becomes an n-dimensional matching problem. Since this is a considered NP-Hard problem, some restrictions can be made. First, a minimum number of vertices for each body segment is chosen to generate a spanning tree instead of having the complete graph. Afterward, the problem is decomposed into a set of bipartite graphs to determine the matching between adjacent nodes independently. With all the connections defined, they make the union of all the connections that share common vertices, together with each other, to create the skeleton as a whole. Figure 4.1 illustrates the final result of the pose estimation.

4.2 2D Partially Decomposed Model

In real time applications, efficiency and approaches for reducing the computational cost are essential. Concerning the model of Cao et al. (cao2017), we propose a new deep neural network based on a streamlined architecture and Tensor decompositions.

Initially, we modified the generation of the first features using Depthwise Separable Convolution of MobileNet (howard2017), which is typically used for embedded devices. We replace the original layers of VGG network because its performance in this step represents the first bottleneck. Here, we decrease the number of operations required in the convolution network process, making an approximation through successive convolutions. The MobileNet model is based on a form of factorized convolutions which transforms a standard convolution into a depthwise and a pointwise convolution (howard2017). Depthwise Separable Convolution deals not only with the spatial characteristics of the image but also with depth, i.e., the RGB channels. In this process, a kernel is separated into other two to perform two convolutions: a depthwise and a pointwise. Essentially, the depthwise convolution applies a single filter to each input channel and the pointwise applies a $1x1$ kernel convolution to combine the outputs.

A Depthwise Separable Convolution requires approximately 8 to 9 times

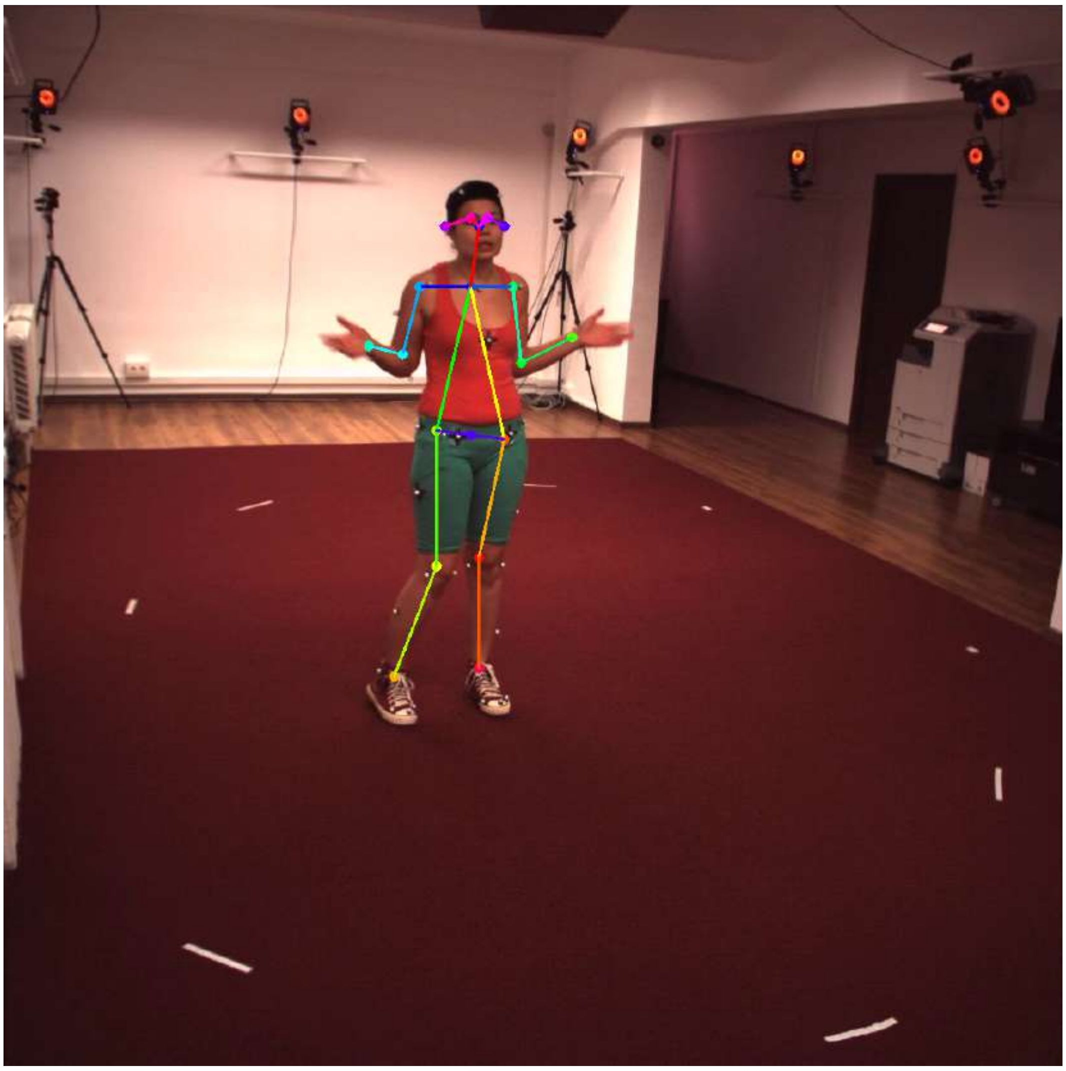
Figure 4.1: 2D pose estimation example.

less computation than a standard convolution, with only a small reduction in accuracy (howard2017). Based on the architecture of the MobilenetV2 (sandler2018), we create 12 layers of Depthwise separable convolutions instead of regular ones for feature extraction.

Following this same idea, we create another lightweight structure for intermediate layers of the network intending to increase FPS performance. In the first three stages, we perform approximations using tensors to speedup the runtime performance without having significant losses in the network accuracy. Each intermediate convolution stage was replaced by a block consisting of a pointwise, a standard convolution, but with reduced space, and another pointwise convolution. For example, in the intermediate layers of our network, convolutions with input channels and output channels of $n = 128$, kernel 3x3 and stride 1x1 were replaced by a block containing a pointwise convolution with 128 input channels and 28 output channels, a regular convolution with

28 and 32, for input and output size respectively, with a kernel 3x3 and another pointwise convolution with 32 channels for input and 128 channels for output. We use this approach to change the stages 1 to 3 of our network. This can be seen in an analogous manner as an approximation of a regular convolution described by Kim et al. (kim2015), were they used a High Order SVD for this task. Similar as in the depthwise separable convolutions, we split a regular convolution into other 3, where it drastically reduce the computation and model size. Considering the tensor decomposition's theory stated in the previous sections, let us relate it with convolutional layers. A regular convolution maps an input tensor into another with different size by successive operations as we can see in the Equation 4-5:

$$conv(x, y, z) = \sum_i \sum_j \sum_k \Theta_{(i,j,k,z)} \mathcal{W}_{(x-i,y-j,k)}, \tag{4-5}$$

where Θ is a kernel of size $IJKZ$ (for example, a kernel with size $I \times J = 3 \times 3$, with $K = 3$ channels (RGB), and $Z = 128$ convolution filters) and $\mathcal{W}$ an input tensor with size XYK, as we refer typically as an image, for example, with X and Y the image dimensions and K the number of channels. Replacing Θ by our approximation, this approach can be seen as kernel tensor decomposition (kim2015) as we can see in the Equation 4-6.

$$conv(x, y, z) = \sum_i \sum_j \sum_k approx \mathcal{W}_{(x-i,y-j,k)} \tag{4-6}$$

Regarding the equation 3-19 in our previous section, the equation 4-7 defines the tensor named *approx* computed by the HOSVD decomposition (tucker1966):

$$approx = \sum_{r=1}^{R_3} \sum_{r=1}^{R_4} C_{i,j,r_3,r_4} U^k_{r_3}(k) U^z_{r_4}(z), \tag{4-7}$$

where U_{k,r_3} and U_{z,r_4} are the factor matrices of sizes KxR_3 and ZxR_4, respectively, and C is s a core tensor of size (I, J, R_3, R_4). Notice that this equation uses only the R_3 and R_4 ranks. In our application, we suppress mode-1 (R_1) and mode-2 (R_2), which are associated with spatial dimensions of an image, because they are quite small (kim2015).

When replacing Θ in the equation 4-5 by the equation 4-7, we can rearrange the operations obtaining 3 convolutional operations used to approximate the original: 2 pointwise convolutions and 1 regular convolution with reduced space. This leads us to the convolutional block previously described. Here the first pointwise convolution reduces the number of channels from K to R_3, the regular convolution of K input and Z output channels has now R_3 input channels and R_4 output channels. Finally, the last pointwise convolution is used to get back the original output size. This process not only makes a speedup in our application, but also reduces the number of parameters needed.

A regular convolution requires D^2ZKXY *multiplication-addition* operations, where D^2 refer to the dimensions of our kernel 3×3 and X and Y refer to the spatial dimensions of our data. Considering the decomposition, our approach has a speed-up ratio S defined by the equation 4-8 (kim2015).

$$S = \frac{D^2ZKXY}{KR_3XY + D^2R_3R_4XY + ZR_4XY} \tag{4-8}$$

Regarding our model and the previous equation, each decomposed layer requires approximately 9.3 times less operations than a regular convolution. Figure 5.3 represents the architecture model of our network. Here, we have more layers of convolution, but the number of operations and weights is smaller, since each regular convolution is substituted by other 3 factorized. As we can see in Figure 5.3, for stage 1, the overall number of convolutions, considering all blocks, is $3 \times 3 + 2$ and for stages 2 and 3 is $5 \times 3 + 2$. We defined this model for the first three stages, after the feature extraction, in an empirical way, where we did not observe significant differences in the accuracy of our network when comparing to the OpenPose. If we extend this approach to the other stages, for example, to stage 4, we verified a more significant difference in the accuracy, almost 25%, leading to wrong results in inference. Also, if we apply to the whole 6 stages, we face an instability problem (kim2015, zhou2016), where it is difficult to find a good learning rate as a well how to initialize the weights. We believe that is an effect of applying stacked decomposed layers to the model, and for this reason, we limited the number of decomposed stages to the first three. We can see the precision and recall of 4 different configurations for decomposed networks in figure 4.2.Also, as we can see in Figure 4.3 and Figure 4.4, we evaluate the number of floating operations and parameters of each decomposed stage. We choose the model of 3 decomposed stages following a trade-off between precision and performance, where, following our interpretation, is the best model version.

Figure 4.12 shows some results of the inference using our network. As we can see, in major cases, we can achieve a good performance in the detection despite some small errors considering high movement shifts and blurred images.

Despite frame rate issues, another problem identified in the OpenPose was the lack of temporal coherence in the original paper. In other words, there is no relation between the objects defined in one frame to the ones defined in the subsequent frames. Therefore, the reference is lost for each identified person. For the context of our applications, it is necessary to create a module for this task. Temporal coherence enables us to actually implement real motion tracking, where the temporal correspondence between each pair of the frame

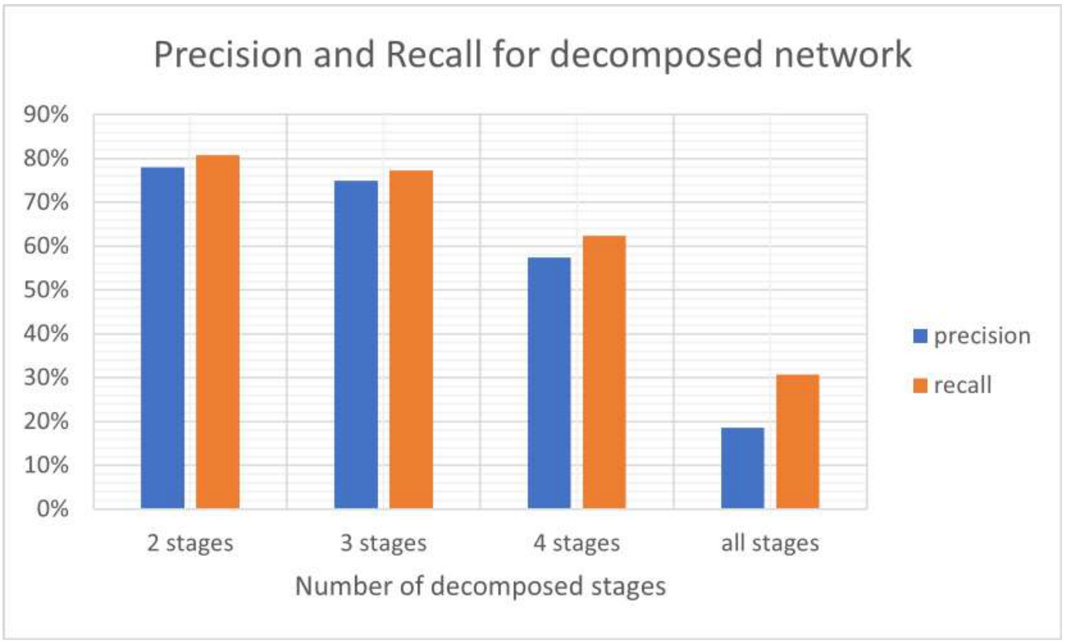

Figure 4.2: Precison and Recall considering different configurations for decomposed stages.

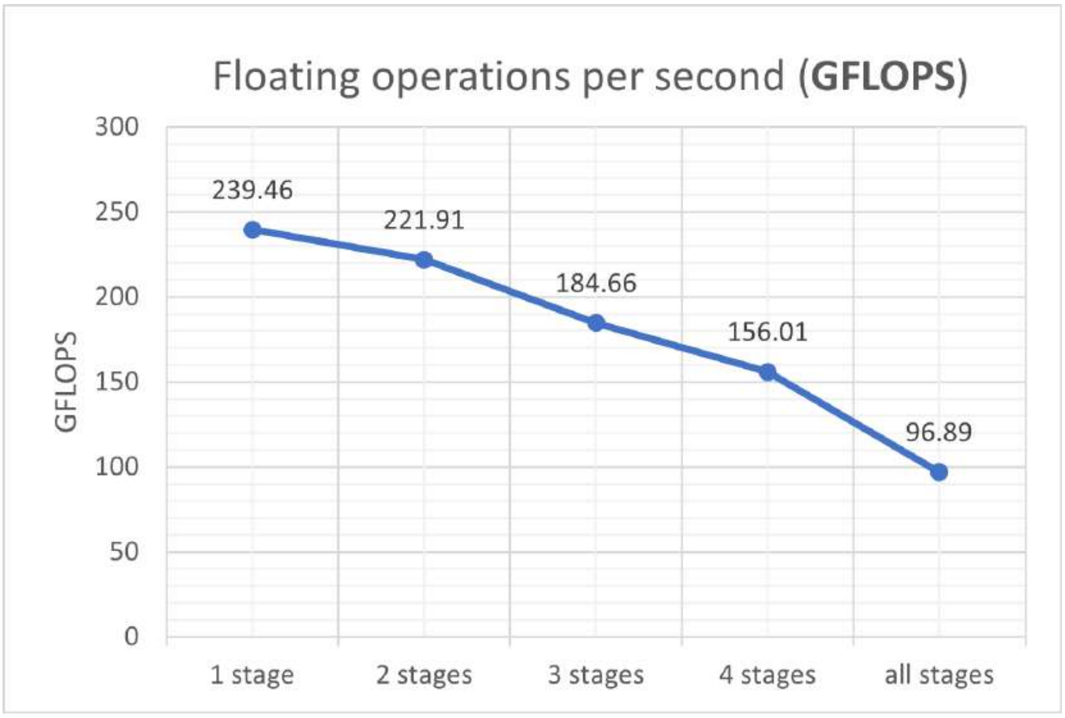

Figure 4.3: Floating Operations per second performed by each version of decomposed models.

needs to be preserved, considering motion coherence between spatial neighbors and across the temporal dimension (tsai2012). We also can use this to fix the tracking in a single person, removing problems involving people interaction, for example.

To solve this problem, in a first experiment, we create a module for

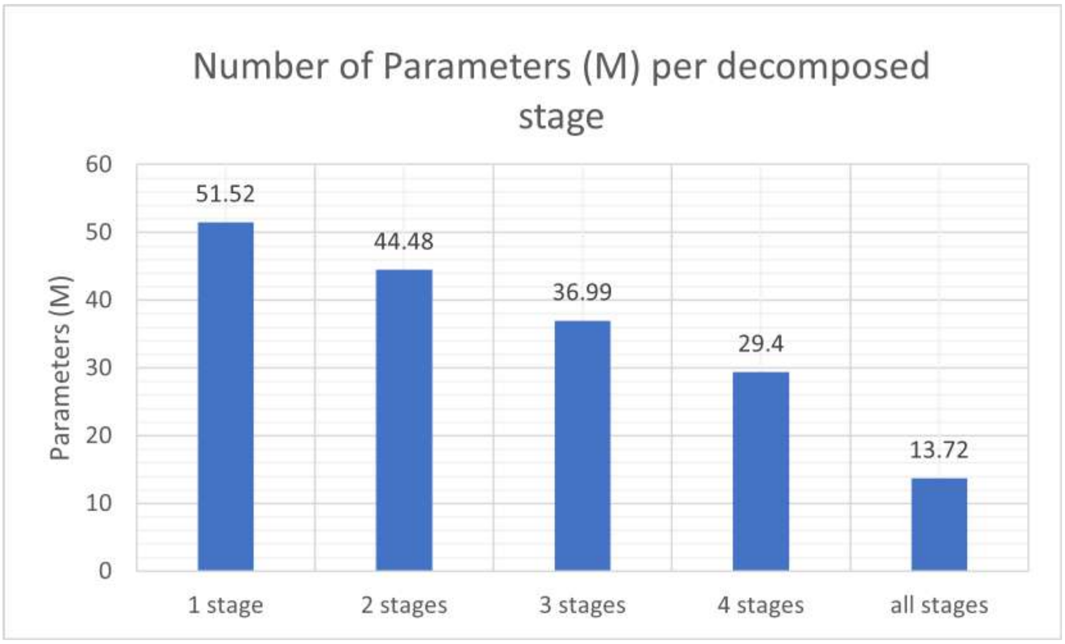

Figure 4.4: Number of Parameters of each version of decomposed models.

temporal coherence using Optical Flow. Optical Flow is a pattern of apparent motion of objects in images between two consecutive frames caused by the movement of an object or the camera. In other words, it is an approximation of image motion based upon local derivatives in a given sequence of images. Some patterns can affect the sequence of images, causing temporal variations in the brightness. In sum, it specifies how image pixels moves between subsequent frames.

Firstly, we tested the Kanade-Lucas-Tomasi (KLT) algorithm (kim2009), and our preliminary results not only presented an increase in the performance of the technique considering the frame rate, but also smoothness in the motion capture. For the KLT algorithm, we pass the points of skeletons detected by the network. At each interval of t frames, these points are iteratively tracked through the optical flow, where the previous frame, the position of the previous points, and the next frame are passed to the detection function. With each new frame, the location of each point of the tracked skeletons is updated. If any position or point are lost in the process, the tracking is stopped. Then, the network is executed again and the skeletons recalculated. The same is done at the end of the interval of t frames to guarantee the consistency of the tracking. Normally we use an interval of 10 frames, where we do not where we did not detect a greater error in the tracking when compared with the technique frame to frame. Here, the sparse optical flow deals with much less parameters than the use of the CNN and the calculation of line integral at each frame, where it tracks just the previous point detected by the CNN. The optical flow assumes

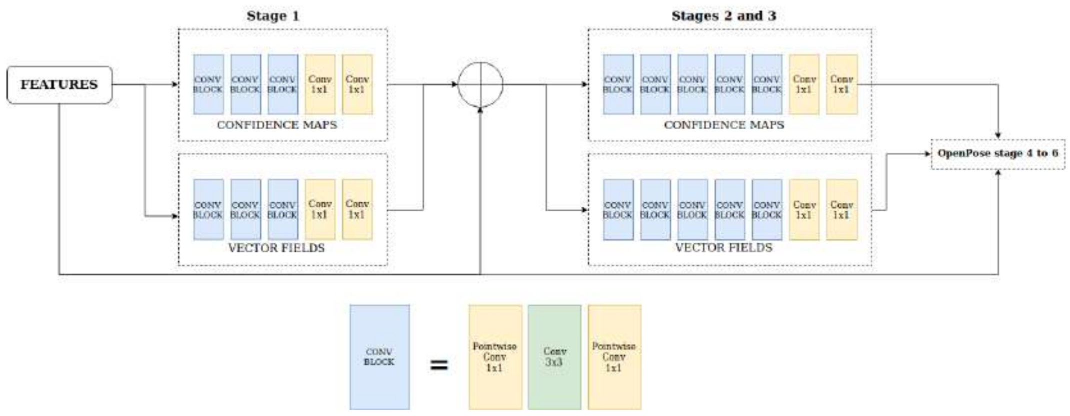

Figure 4.5: TensorPose CNN model. The initial features were extracted using the first 12 layers of a Mobilenet V2 and in the intermediary layers we replace conventional convolutions by a block containing a pointwise convolution, regular but with reduced space and another pointwise convolution. We have more layers of convolution but the number of operations and weights is smaller, which leads to a boost in performance. At the final of each stage the results are concatenated with the results of feature extraction from Mobilenet to refine the results.

that the intensity of the pixels of an object does not change with time and neighboring pixels have the same pattern of movement. Classical approaches, such as optical flow fail when it comes to environments with illumination changes and long-range motions. To solve these issues, we change the algorithm to use the Robust Local Optical Flow (Senst2016), following the framework proposed by Senst et al., while taking into account an illumination model to deal with varying illumination. Here the computational cost of this variation, used by the KLT method, is given by the upper bound of $O(nN_{large}i)$, where n is the number of computed motion vectors and N is the number of pixels of the larger image region i (Senst2016). Also, as support to the tracking process, we use the Kalman filter and Multiple Instance Learning (MIL). The Kalman filter (chui2017) is used to predict the position of the skeleton in the frame after the current one, and to ensure that the references for each skeleton detected are maintained. Given the distance between the points calculated by the network and those predicted by the Kalman filter, a threshold is given to ensure that the reference points are maintained. We fix this threshold considering a radius of 20 pixels of error. Combining Optical Flow and Kalman filter gives us the possibility to interpolate the data over subsequent frames, which enable us to smooth the captured movement. We also use the MIL (babenko2009) just to ensure that the detected person's reference is maintained. We use it as an additional feature to identify the bounding box of detected keypoints, and to maintain the reference of a person, even when the network is called. Although their use is optional. Maintaining the reference of each person along

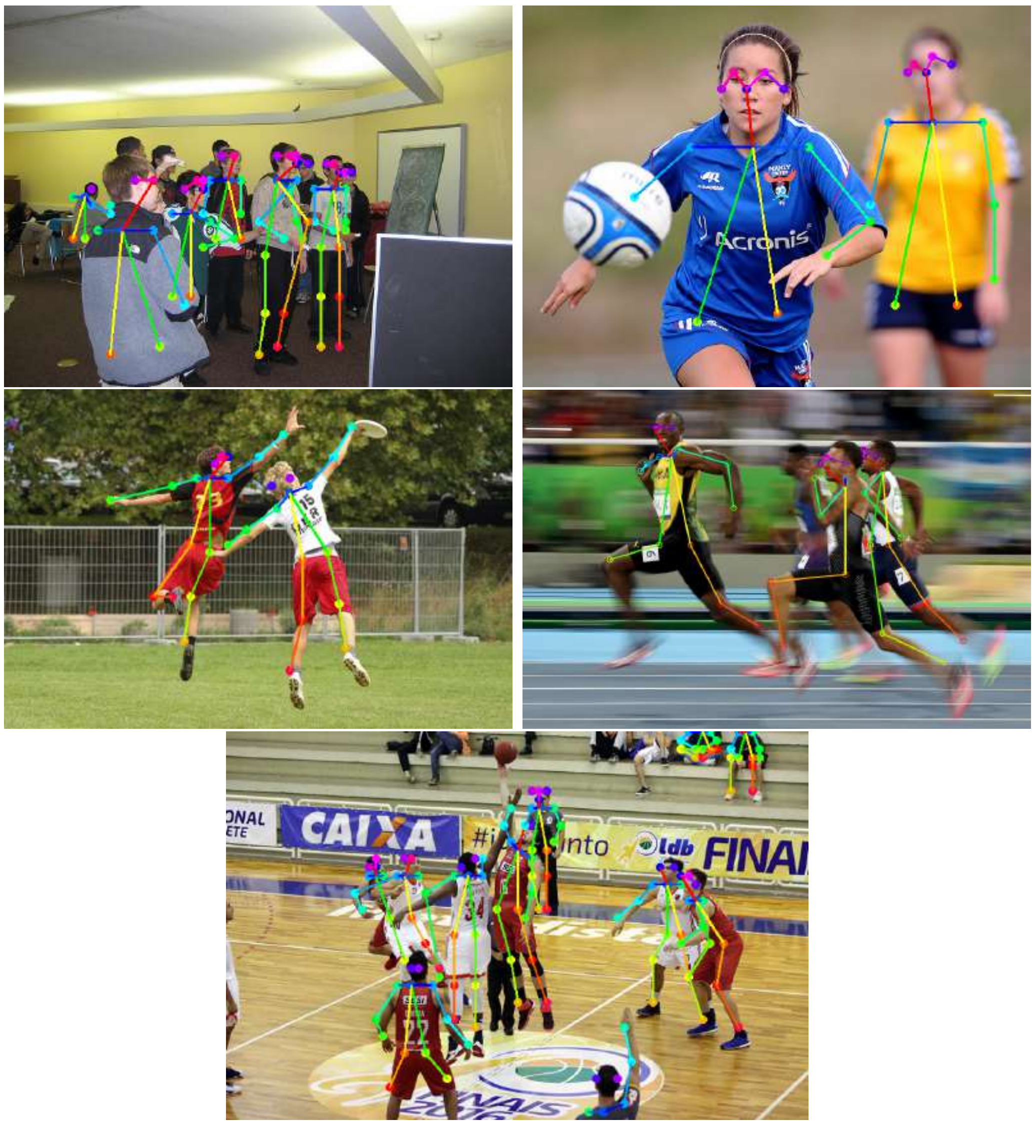

Figure 4.6: Results containing viewpoint variation and occlusion, which are common characteristics in images. Images from COCO dataset, Rio Olympics 2016 and Brazilian National Basketball league.

the frames is straightforward. We set a bounding box for keypoints coordinates, afterwards a module called *Manager* assigns an identifier for each detected person. We then employ a Kalman Filter to predict the person's future location based on his previous data and estimate the correct new position. To set the new positions, we compute the distance between the centroid of the previous bounding box with the subsequent, combining with the estimated velocity and position given by the Kalman Filter. However, this technique also faces some problems. For instance, when two persons are overlap, in some cases, their identifiers are swapped. In future works, we intend to solve this issue.

4.3 Fully Decomposed Model

This method can be seen as an improvement for pose machines. As in the previous architecture, for the output we have the set of 2D confidence maps S of body part locations and a set of 2D vector fields L (cao2018). We developed a new architecture for the data refinement using SE-Net blocks between intermediary layers. With this approach we improve the model accuracy and after, and again, we decompose convolutional layers aiming to reduce redundancy and improve processing time. Figure 5.3 represents the architecture of our model following the convolutional pose machines (wei2016), where we refine the predictions over successive stages. We built a sequential model for the first 6 convolutional blocks of each stage, adding after each pair of convolutions, attention modules. The outputs of two consecutive convolutional blocks are concatenated, following an approach similar to DenseNet. The last 6 convolutional layers form 2 branches with 3 layers each, where the first is responsible for the 38 maps for part affinity fields and the second, for 18 feature maps for body keypoints. The output of the first branch is used as input to the second in a sequential approach.

After all stages, we have a post-processing stage similar to Openpose (cao2017) and the previous approach. For assignment problem, we use a variation of the Hungarian algorithm called the Jonker-Volgenant algorithm (jonker1987) to create a connection between two parts. This process is performed in each frame.

In the next sections, we first briefly review the structure of the Attention block, the factorized convolutions, and then the detailed descriptions of our proposed modules and methods are presented.

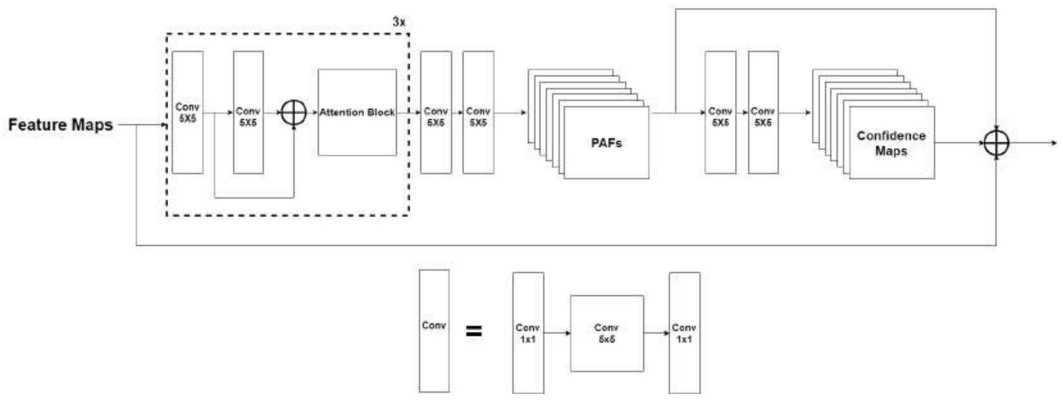

Figure 4.7: In our CNN model, we have 3 blocks containing two convolutional layers and an attention block

4.4
Attention block

The original SE-Net proposed a "Squeeze-and-Excitation" block to adaptively highlight the channel-wise feature maps by modeling weights applied to each channel(hu2018). It was proved that SE blocks bring significant improvements in performance for existing state-of-the-art CNNs at slight additional computational cost.

In the excitation step, we fully capture channel-wise dependencies. A simple gating mechanism with a sigmoid activation (S) is applied. Consider an input with ch channels. This mechanism is composed by two fully connected layers (FCL) where the first has $\frac{ch}{r}$ neurons and a LeakyReLU activation, and the second, ch neurons. Finally, the second FCL is followed by the sigmoid activation. The hyper parameter ratio r allows us to vary the capacity and computational cost of the SE blocks in the network and can be evaluated empirically. In the sequence, to generate the output, an element-wise multiplication between the result of the gating mechanism and data input is performed.

In a formal way, let $\mathcal{X} \in \mathbb{R}^{H \times W \times ch}$ be the tensor corresponding to the input of attention block and $\mathcal{X}' \in \mathbb{R}^{H \times W \times ch}$ its output. In the squeeze operation, a global average pooling is performed on the input to generate the weights z, where each element of z is computed according to the Equation 5-1:

$$z_c = \frac{1}{H \times W} \sum_{i=0}^{H} \sum_{j=0}^{W} X_c(i, j), \tag{4-9}$$

where X_c corresponds to each channel of $\mathcal{X}$. After that, it is applied the excitation step, on which the information aggregated in the squeeze operation, is used in a operation which aims to fully capture channel-wise dependencies. A simple gatting mechanism with a sigmoid activation is applied as according to this Equation 4-10:

$$\alpha = sigmoid(W_2(\beta(W_1(z)))), \tag{4-10}$$

Here, $W_1 \in \mathbb{R}^{\frac{ch}{r} \times ch}$ and $W_2 \in \mathbb{R}^{ch \times \frac{ch}{r}}$ represent two fully connected layers and β a LeakyRelu activation. The hyperparameter ratio r allows us to vary the capacity and computational cost of the SE blocks in the network and can be evaluated empirically.

In the sequence, the weights α are re-scaled on the input $\mathcal{X}$, generating the output. To do that, a Hadamard product is computed, i.e., an element-wise multiplication between α and $\mathcal{X}$ as described in Equation 4-11:

$$\mathcal{X}' = \alpha \circ \mathcal{X} \tag{4-11}$$

Similarly to Su et al. (su2019), we use spatial attention mechanism to adaptively highlight the task-related regions in the feature maps in addition to the channel-wise. Different from paying attention to the entire image, which can lead to focus on irrelevant features, spatial attention can enhance the overall results (su2019). With the spatial input $\mathcal{X} \in \mathbb{R}^{H \times W \times ch}$ and its output $\mathcal{X}'$ with same dimensions, the spatial attention is generated by a pointwise convolution followed by a sigmoid activation. The spatial attention is represented by Equation 4-12:

$$\beta = sigmoid(W\mathcal{X}), \tag{4-12}$$

where W represents the pointwise convolution. Finally, the output is also re-scaled to achieve $\mathcal{X}'$ according to Equation 4-13 (su2019):

$$x'_{i,j} = \beta_{i,j} * x_{i,j}, \tag{4-13}$$

which is an element-wise multiplication between the spatial elements of β and $\mathcal{X}$. In our architecture, each attention block is composed of spatial attention modules followed by channelwise SE-Net blocks, as we can see in Figure 4.8.

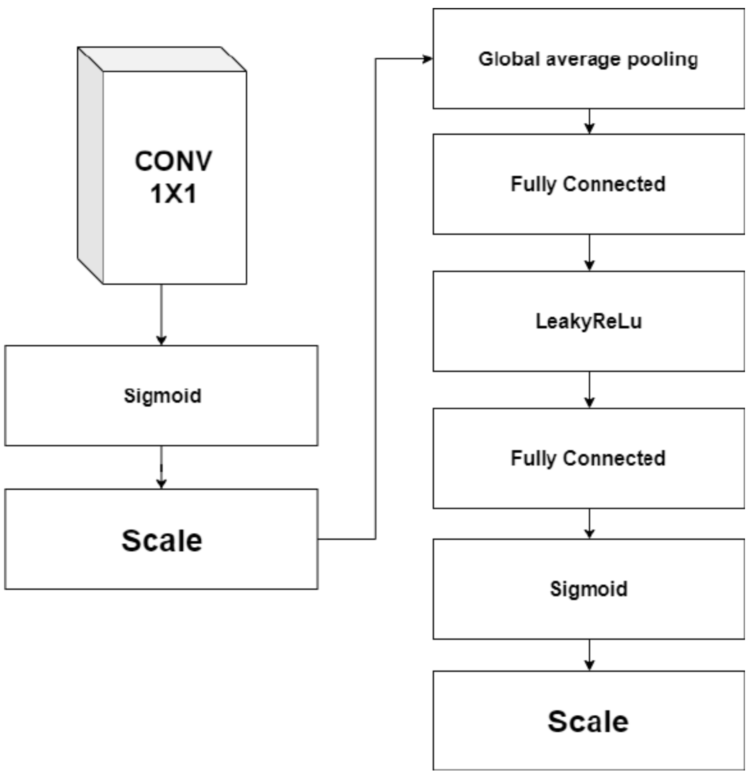

Figure 4.8: The attention block used in our architecture. Here we first process spatial attention operations before the channel-wise.

4.5
Factorized Convolutions

Similarly to the first version of our TensorPose, we apply tensor decompositions to neural networks aiming to factorize their kernels, creating approximations and improving the processing time for inference. We use this strategy as an exploratory way to model a factorized architecture, formed by pointwise convolutions combined with a regular convolution with reduced size. We aim to create a low rank approximation, where we model our architecture following an one-shot whole network compression scheme (kim2015). Here differently from the previous approach, we have two steps: rank selection and tensor decomposition. We analyze the unfolding mode-3 and mode-4 of each layer's kernel tensor with global analytic variational Bayesian matrix factorization (VBMF)(kim2015). Our experiments show that an approximation of 1/3 or 1/4 of mode-3 and mode-4 of a tensor already presented effective results. We consider just the mode-3 and mode-4 due the fact that the mode-1 and mode-2 represent just the spatial dimension of kernel and they are quite small. The VBMF tries to infer a optimal low rank selection and with these values, after, we apply the Tucker decomposition on each kernel. Even using the VBMF, it was empirically verified that an approximation of 1/3 or 1/4 of original output size of the convolution operations already presented effective results considering the performance of our network.

4.6
Proposed Architecture and Experimental details

4.7
Architecture

As said before, our architecture is based on the concepts of convolutional pose machines and in this section we present the modifications in the architecture for feature extraction and refinements stages. Again, we remove the 12 blocks of convolution of a VGG-19 used in the original and replace by a modified architecture of a Mobilenet v2. This architecture has the advantages of being a state-of-the art mobile constrained network with a huge improvement in performance while maintaining the accuracy. Also we set a number of 6 stages for intermediary refinement in the iterative prediction.

4.8 Training

We trained our model over 120 epochs using Adam optimizer with learning rate varying from $8e^{-5}$ to $7.38e^{-6}$. Also we use as activation function LeakyRelu instead of Relu. Our cost function considers a gaussian kernel distance between the annotations and the inferred data for each body part and the affinty fields, which can be written as in equation 4-16.

$$
\begin{aligned}
f^i_{key} &= \sum_{k=1}^{nkeys} \sum_{p} 1 - e^{-w(p)\left\|key^i_k - key^*_k\right\|_2}, && (4\text{-}14)\\
f^i_{paf} &= \sum_{k=1}^{paf} \sum_{p} 1 - e^{-w(p)\left\|paf^i_k - paf^*_k\right\|_2}, && (4\text{-}15)\\
loss &= \sum_{i=0}^{6} f^i_{key} + f^i_{paf} && (4\text{-}16)
\end{aligned}
$$

The kernel distances in our functions were similar to the ones proposed by Cao et al. (cao2018). We weight the loss functions spatially to address a practical issue that some datasets do not completely label all people. Here, key and paf represents the inferred maps for each keypoints and part affinty field respectively and the w is a binary mask were $w(p) = 0$ when the annotation is missing at pixel p. After the training process with the original convolution structure, we decompose our network following the HOSVD and retrain over 50 epochs to get fine tune the model and improve the precision of detections.

4.9 Experiments with partially decomposed network

Our proposal was tested and compared with the results obtained by OpenPose 1.3. The TensorPose code was implemented in Python, including the post-processing steps, such as the line integral calculation and the assignment problem. In training, 110000 images were used as well as 5000 images in validation, with a maximum of 200 iterations per epoch. In COCO dataset, approximately 150000 people and 1.7 million labeled keypoints are available. Considering the inference, we use the COCO validation dataset to compare our modified model and the OpenPose. COCO uses a metric called Object Keypoint Similarity (OKS), which measures of how close the predicted Keypoint is to the ground truth annotation. Also, COCO uses the mean average precision (AP) and average recall (AR) over 10 OKS thresholds as the main competition metrics(cao2017). In our initial tests, we consider Precision and Recall at 50% and 75% OKS (AP-50, AP-75, AR-50, AR-75) which are usually used for benchmark. We test both models with the evaluation data from COCO and submit the results to the COCO evaluation server. Equation 4-17 shows the formula used to calculate the OKS.

$$OKS = \frac{\sum_i e^{-\frac{d_i^2}{s\hat{}2k_i^2}} \delta(\upsilon_i > 0)}{\sum_i \delta(\upsilon_i > 0)} \tag{4-17}$$

where d_i are the Euclidean distances between each detected keypoint and the ground truth, υ_i are the visible flags for each annotated keypoint, and $s \times k$ ($scale \times keypointconstant$) is related to a standard deviation of a gaussian related to concentric circles where their radius varies by keypoint type. The metrics AP-50, AP-75, AR-50, AR-75 are related to how close a prediction is to annotated data considering the Gaussian standard deviation. The $s \times k$ is computed by evaluating this Gaussian function, centered on the ground-truth position of a keypoint and the standard deviation is specific to the keypoint type, which is scaled by the area of the instances, measured in pixels (ruggero2017). Table 4.8 shows our results in contrast to OpenPose and Figure 4.11 show us a visual comparison between the 2 techniques. Also, varying the OKS threshold from 0.5 to 0.95, we measure both precision and recall for the COCO validation dataset, as we can see in the Figure 4.10, where highest values mean how close our predictions are to the ground truth annotations considering each threshold.

Model	AP 0.50	AP 0.75	AR 0.50	AR 0.75
OpenPose	0.780	0.593	0.807	0.647
TensorPose	0.750	0.536	0.772	0.591

Table 4.1: Precision and Recall for OpenPose and TensorPose Models.

Here, higher OKS means higher overlap between predicted Keypoints and the ground truth. As we can see, our model has about 6.5% lower accuracy than OpenPose, which is mainly caused by our proposed simplification on the intermediary layers. Just to remember that this simplification promotes an advantage of having 9.3 times less operations in these layers than the original OpenPose. We have focus on real-time application, so this accuracy loss does not represent significant problems, since some prediction errors are acceptable.

Following the benchmarking and error diagnosis for pose estimation proposed by Ruggero et al. (ruggero2017), we analyze 4 types of localization errors using the annotations of COCO dataset:

- Jitter: small position error around the correct keypoint location;
- Miss: large localization error when the detected key point is far away of any body part;

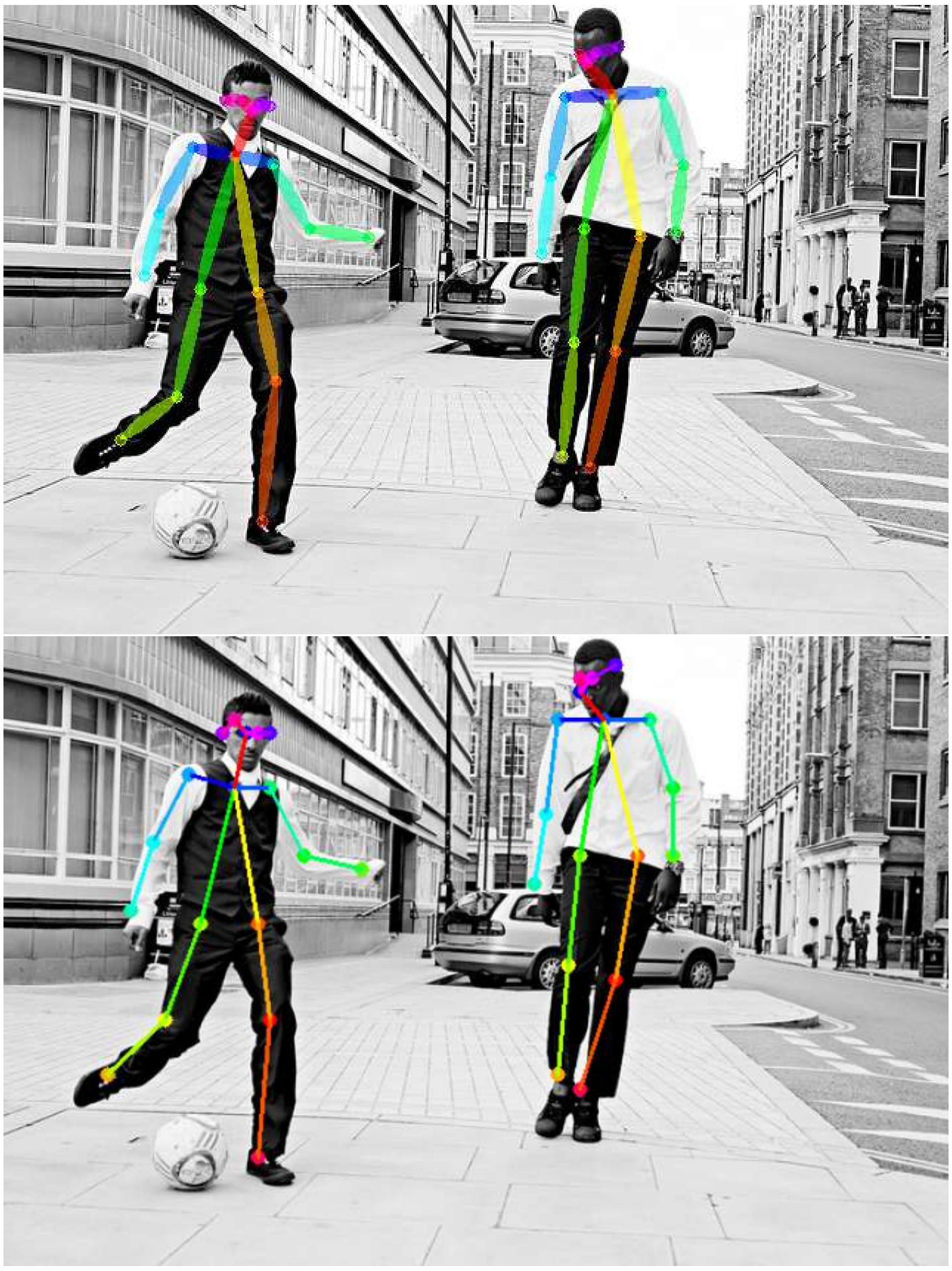

Figure 4.9: Comparison of between the OpenPose model and ours. The first Figure is generated by the OpenPose and the second by our model. As we can see, our model is slightly less accurate than the original work and fails to find the exactly position of a body joint in particular cases. Although due to the nature of our application, this issue can be considered acceptable. Image from COCO dataset.

- Inversion: confusion between semantically similar parts of a detected person,i.e, wrong body part associations for the same person due problems like self occlusion, for example;
- Swap: confusion between semantically similar parts of different persons, i.e., problems related to the association of body parts of a detected person to another.

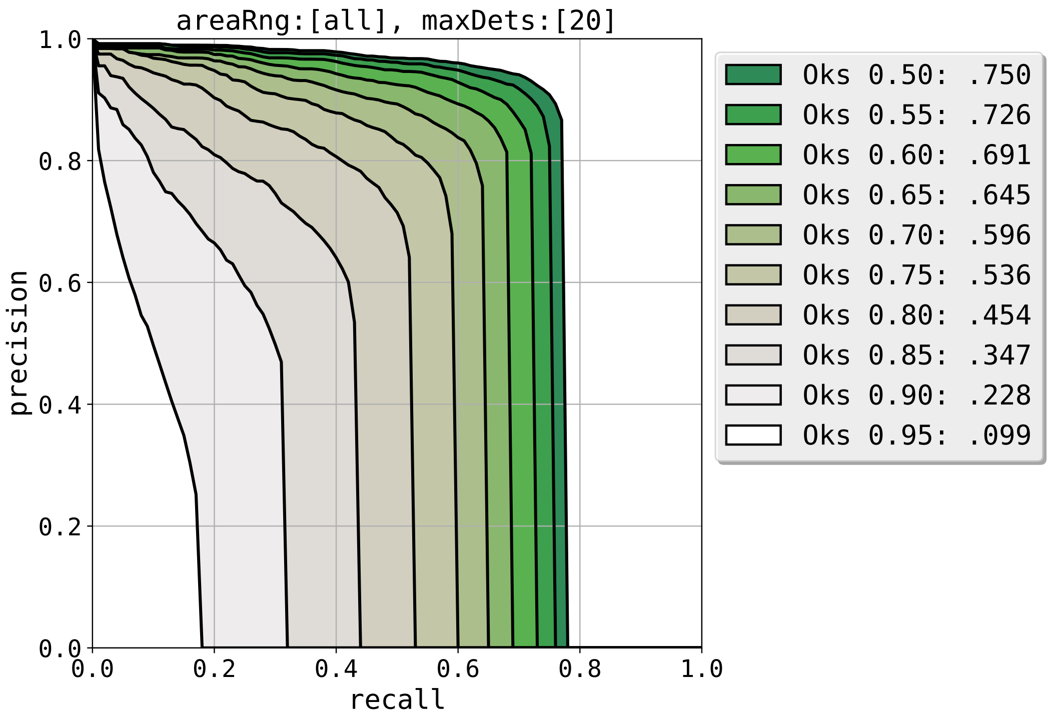

Figure 4.10: Precision and Recall for all OKS thresholds. AreaRng is related to the area of the objects annotated with size bigger than 32^2 pixels.

Table 4.2 shows the total of correct predictions and errors and Table 4.3 presents the estimate error associated to each detected joint considering our model. As we can see, most of the errors are related to the small difference between the predict keypoint and the annotated data and to significant localization error, which we believe this is due to false positives detected. We can also see that most localization errors affect keypoints more sensitive to errors related to occlusion or that usually involves some interaction between people such as the arms. As we can see in Table 4.2, our model performs better than OpenPose considering Jitter, Inversion, and Swap errors. However, it also misses more keypoints localizations than the previous related work.

Total Num. keypoints: [62790]	Tensorpose	Openpose
Good Prediction	72.1453%	73.5%
Jitter	12.80%	13.9%
Inversion	2.41%	3.7%
Swap	0.976%	1.9%
Miss	11.656%	7%

Table 4.2: Total of correct predictions and errors considering the COCO validation dataset in percentage. We compare our technique with OpenPose.

In terms of runtime performance comparisons, we begin with the test of the CNN processing. Considering just one image with 23 people, we compared

Keypoints	jitter %	inversion %	swap %	miss %
nose	9.6	0	2.4	5
eyes	16	4.8	4.2	7.4
ears	13.1	0.3	3.6	6.1
shoulders	12.8	12	21.9	7.1
elbows	12.6	4.6	17.5	14.3
wrists	9.7	10.8	15.2	21.8
hips	13.5	24.2	14	11.7
knees	7.1	22.7	11.1	13.4
ankles	5.5	20.6	10.1	13.2

Table 4.3: The frequency of localization errors over all the predicted keypoints. We have 62790 joints detected in 5000 images with this test.

our approach with OpenPose. The tests were performed in a GPU Nvidia RTX 2060 with 6GB of memory, where we vary the scale of the image tested by a factor of 0.5 and repeat each test 1000 times. As we can see in Table 4.6, in average, our approach has a better performance when compared with the OpenPose.

Image Resolution	OpenPose	TensorPose
328 x 193	~55,08 ms	~54,86 ms
656 x 386	~120.224 ms	~84,25 ms
984 x 579	~174,75 ms	~116,66 ms
1312 x 772	~320,18 ms	~210,41 ms

Table 4.4: CNN processing time for OpenPose 1.3 and TensorPose. We vary the scale of the image tested by a factor of 0.5 considering 4 image scales.

When processing videos, to avoid processing the CNN and calculating the line integral and the execution of the Hungarian algorithm at each step, we use sparse optical flow. We get not only performance improvements, with a frame rate above 30 FPS, but also smoothness in motion tracking, also due to the Kalman filter. In the first test, we defined a fixed interval of 10 frames, where the optical flow is processed and, afterward, its parameters are updated by a new processing step in the neural network. Such range was defined empirically, where we saw that there was not a significant difference in the accuracy of the tracking. Our strategy, considering both changes in the network and the use of optical flow during t frames, reduces the overall time for tracking and increases the frame rate, as we can see in Table 4.5.

Similar to the OpenPose, we analyzed some limitations when our approach fails. We face the same problems with highly crowded images where people are overlapping, and the approach tends to merge annotations from different people while missing others(cao2017). Also, our application has less

GPU	CPU	OpenPose	TensorPose
Nvidia Titan RTX	Intel(R) Core I9(R) CPU @ 3.3GHz	∼11. FPS	∼36.8 FPS
Nvidia Tesla P40	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	∼10 FPS	∼34.43 FPS
Nvidia TITAN Xp	Intel(R) Core I7(R) CPU @ 3.3GHz	∼8.28 FPS	∼27 FPS
Nvidia RTX 2060	AMD Ryzen 7 1700 @ 3.2GHz	∼6.121 FPS	∼18.27 FPS
Nvidia GTX 960	Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz	∼3.73 FPS	∼12 FPS

Table 4.5: Frame rate comparison between the OpenPose 1.3 and TensorPose.As we can see, our model surpasses in approximately 3x the performance of the OpenPose model used as base.

precision when compared to previous work due to the factorized layers. It has more difficulty to detect joints considering occlusion having a slight noise in the inferred data. As we can see in table 4.8, this is not a significant problem as the results are comparable and we still have a good performance gain.

4.10 Experiments with fully decomposed network

In terms of runtime performance comparisons, we began with the test of the CNN processing. Considering just one image with 23 people, we compared our approach with OpenPose. The tests were performed in a GPU Nvidia RTX 2060 with 6GB of memory, where we vary the scale of the image tested by a factor of 0.5 and repeat each test 1000 times. Our network achieves a performance of almost 20 frames per second when a network resolution of 656×368. We also test a non factorized version of our network. In general, for the same resolution, it has achieved a performance of almost 12 frames per second. As we can see in Table 4.6, in average, our approach has a better performance when compared with the OpenPose, considering our factorized version.

We also perform tests in CPU. As we can see in Table 4.7, in average, our approach has a better performance, considering frames per second, while the original OpenPose is unpractical to be used.

Table 4.8 shows our results in contrast to OpenPose(cao2017, cao2018) and AlphaPose(fang2017). In terms of precision and Recall, Alphapose has the best results; however, it has a different approach that uses a top-down strategy. It also is impractical to run this model in modest hardware systems. In a High-end GPU, the AlphaPose runs over only 20 FPS. Since our model

Image Resolution	OpenPose	Ours
328 x 193	~55,08 ms	~ 45 ms
656 x 368	~120.224 ms	~ 51.26 ms
984 x 579	~174,75 ms	~ 80.72 ms
1312 x 772	~320,18 ms	~ 112 ms

Table 4.6: CNN processing time for OpenPose and our Model. We vary the scale of the input tested by a factor of 0.5 considering 4 image scales. We do this only for the network input, the final result is scaled in the original image size.

Device	OpenPose	Ours
AMD Ryzen 7 1700 3.5GHZ	0.3 FPS	13 FPS
Mac Pro Intel Core I7 2.7GHZ	0.1 FPS	6 FPS

Table 4.7: CNN processing time for OpenPose and our Model in CPU.

is highly based on OpenPose and a bottom-up strategy, performance tests with AlphaPose are out of scope. Figure 4.11 shows a visual comparison between the OpenPose and our technique. We consider the original architecture of OpenPose proposed by Cao et al. (cao2017). Here, higher OKS means higher overlap between predicted Keypoints and the ground truth. As we can see, our model has a lower accuracy than OpenPose, which is mainly caused by our proposed simplification on the convolutional layers. However, this simplification promotes the advantage of having 9.3 times less operations(Schirmer2019) in these layers than the original OpenPose(Schirmer2019). We have focused on real-time application and our model's use in low power devices, so this accuracy loss does not represent significant problems. In Figure 4.11, we can see that OpenPose presents a confusion between semantically similar parts of different persons in this frame, while our method does not find this pattern.

Model	AP 0.50	AP 0.75	AR 0.50	AR 0.75
AlphaPose	0.84	0.715	0.895	0.775
OpenPose	0.782	0.594	0.807	0.650
Ours	0.743	0.501	0.702	0.580

Table 4.8: Precision and Recall for AlphaPose, OpenPose and our model.

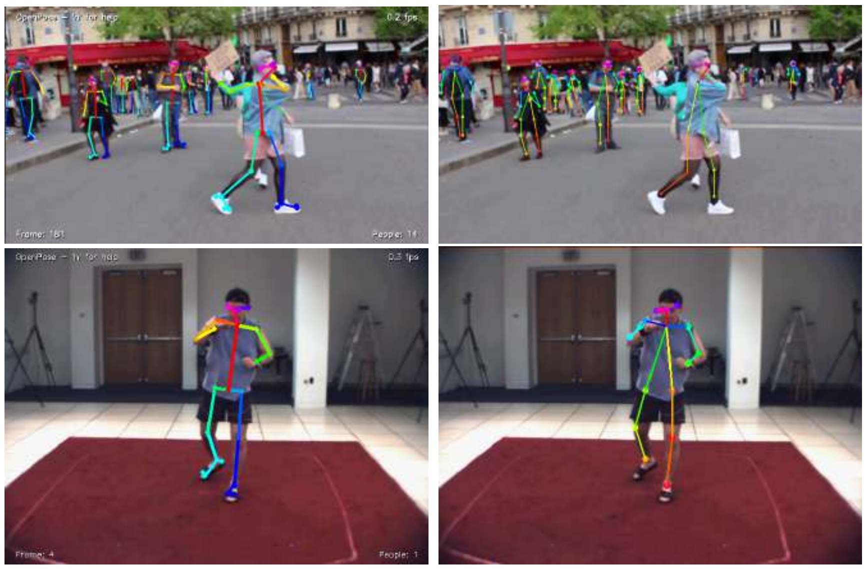

Figure 4.11: Comparison between the OpenPose model and ours. The first Figure (left) is generated by the OpenPose and the second by our model (right). As we can see, our model is slightly less accurate than the original work in particular cases, considering the exact position of a keypoint. Although, OpenPose presents a confusion between semantically similar parts of different persons in this frame. Also, our model do not consider all the feet keypoints.

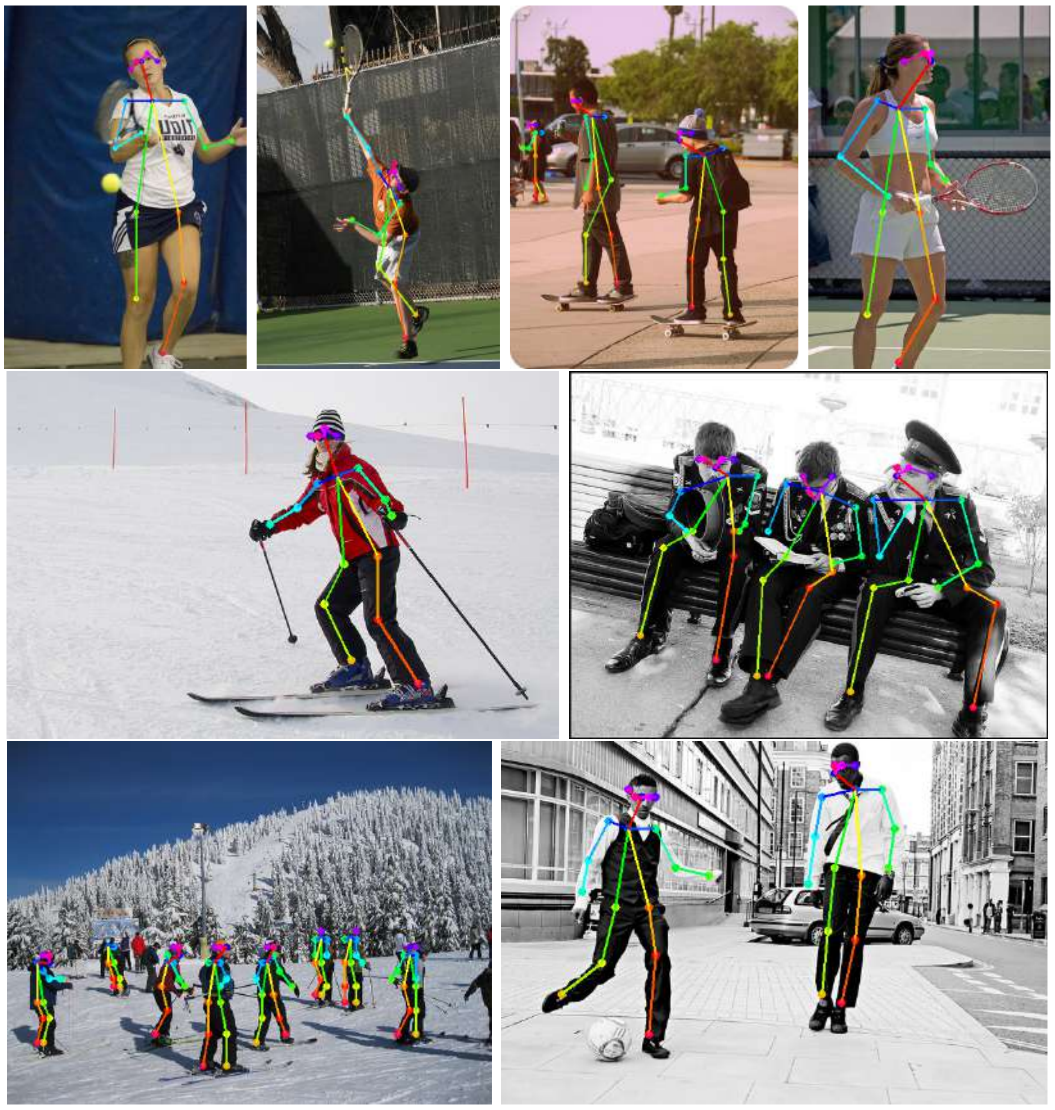
Figure 4.12: Results containing viewpoint variation and occlusion, which are common characteristics in images. Images from COCO dataset.

5
3D human pose estimation

In this chapter, we present the main aspects of our 3D pose estimation model. We also show a new attention model for Graph Convolutional Neural Networks called Semantic Graph Attention, an evolution of traditional Semantic Graph Convolutions. Here we propose an improved graph convolution operation called Semantic Graph Attention, which is derived from Semantic Graph Convolutions (zhao2019). The primary motivation is creating a new network model where we can learn both channel-wise weights for edges in the graph, combine them with kernel matrices, and understand global and channel inter-dependencies without using non-local layers. With this approach, we can achieve state-of-the-art performance considering the error in millimeters from 3D human pose regression with only 1/4 of operations needed from previous works. Finally, we present some applications for 3D computer animation, as we can see in Figure 5.1.

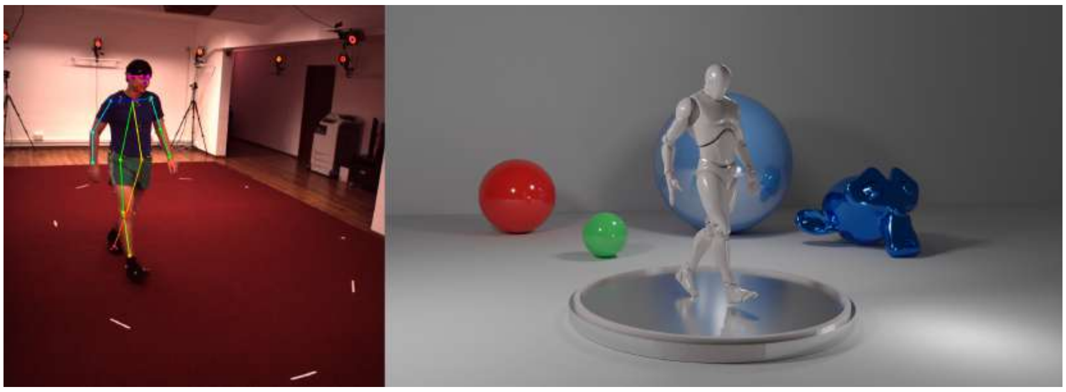

Figure 5.1: Motion Capture using 3D pose estimation neural network for Computer animation.

5.1 Graph Convolutional Networks

Following principles of regular *Convolution Neural Networks*, a *Graph Convolution Network* can be considered a way to deal with arbitrary graph structures(bruna2013, defferrard2016, kipf2016). This is highly related to our approach to analyze human pose as a structured graph.

Convolutional Graph Networks (GCNs) share the filter parameters in the graph. The *GCNs* training stage consists in learning structures capable of processing graph information from the node matrix $X \in R^{N \times D}$ (N nodes containing D features) and the adjacency matrix $A \in R^{\|N\| \times \|N\|}$ (defferrard2016, kipf2016).

Each layer is a non-linear function as in equation 5-1:

$$H^{(l+1)} = f(H^l, A), \tag{5-1}$$

with H being the output of each layer and $H^0 = X$. We can rewrite this function as follows in equation 5-2 :

$$f(H^l, A) = \sigma(AH^lW^l), \tag{5-2}$$

where σ represents the *LeakyRelu* activation function and W the weight matrix of the network layer. There are some limitations to this approach because the multiplication by the matrix A would only consider features from the neighborhood, but not from the node itself. This problem is addressed by adding the identity matrix to A ($A' = A + I$).

Furthermore, A' should be an unitary matrix to do not scale the vector of features. We reach it by normalizing A' rows using the *Normalized Laplacian Matrix* $D^{-1/2}AD^{-1/2}$, where D^{-1} is the inverse of the diagonal matrix with the degree of the graph nodes. Substituting in the previous equation, we have equation 5-3:

$$f(H^l, A) = \sigma(D^{'-1/2}A^{'}D^{'-1/2}H^lW^l). \tag{5-3}$$

There are two clear disadvantages to make the graph convolution considering a regression to work on nodes with arbitrary topologies. The first one is the kernel matrix W is shared by all the edges. As a result, the relationships of neighboring nodes, i.e. internal structure, are not well explored. This is also a limiting factor because the receptive field is fixed with ones (zhao2019), the second disadvantage.

A *CNN* with a convolution kernel of size $k \times k$ learns k^2 different transformation matrices. The transformation matrices decode features within the kernel spatial dimension. This formulation can be approximated by learning a vector of weights $\vec{a}_i$ for each position of a pixel in an image or a graph node, and then combining them with a shared transformation matrix W (zhao2019).

We can transform an image to a graph by considering the pixels as nodes, and two neighbor pixels being connected by an edge (8-connect neighborhood). So, a kernel size k affects all pixels distant less than $d = \dfrac{k-1}{2}$. We can extend this approach for *GCNs* by considering that a convolution in a graph using a kernel of size d affects all nodes in a neighborhood of size d (zhao2019).

GCNs cannot handle directly with regression problems due the issue that convolution filter shares the same weight matrix for all edges. Furthermore, the filters just operate in a one step neighborhood. As a solution, Zhao et al. (zhao2019) propose to add the weight matrix M to the graph convolution process according to the equation 5-4.

$$f(H^l, A) = \sigma(\phi(A' \odot M)H^l W^l), \tag{5-4}$$

where the matrix M is a parameter to be learned on the network and ϕ is a *softmax* function that normalizes the entries of each node, $\odot$ is an element-wise multiplication (*Hadamard*) that returns m_{ij} if $a_{ij} = 1$ or negative values with large exponents after the *softmax*. In this approach, A works like a mask that forces this to the i node in the graph.

Equation 5-5 shows how to to adapt the Equation 5-4 to adjust the idea of multiple channels.

$$f(H^l, A) = H^{l+1} = ||_{d=1}^{D+1} \sigma(\vec{w_d} H^l \phi(A^{'} \odot M_d)), \tag{5-5}$$

where $||$ represents a channel-wise concatenation and $\vec{w}$ is a vector representing each line d of the transformation matrix W.

5.2 Attention block for Semantic Graph Convolutions

We are proposing a way to adapt SE-NET concepts (hu2018) to give weights to the output features of semantic convolution. This can be related to Global Context Networks(cao2019). We aim to solve the issues of precision and computational complexity,considering both space storage and time complexity, from previous related works.

Considering the computation of features for each node, the idea of adding weights via element-by-element multiplication is natural. We intend to identify inter-dependencies between node features. For this, we propose the following gating mechanism for each channel after a regular SGC, where we learn a matrix of weights, presented in equations 5-6 and 5-7:

$$g = A^{'} \odot \phi(M_1) W_1^l H^l, \tag{5-6}$$

where g is composed by a softmax ϕ function over the entries of Matrix M_1. This hidden layer perform a dimensionality reduction, where it drastically reduces the input space by a factor r where the kernel size of $W_1^l \in R^{\frac{C}{r} \times C}$.

The equation 5-7, represents the expansion back to the original input size:

$$s(g) = \alpha(A^{'} \odot \phi(M_2) \sigma_1(g) W_2^l), \tag{5-7}$$

where kernels $W_2^l \in R^{C \times \frac{C}{r}}$, σ represent a LeakyRelu function, α represents a sigmoid activation, C represents the number of features and r is defined empirically. In our experiments, as in Hu et. al.(hu2018), we use a value $r = 16$. With the output of function $s(g)$ for each channel we perform an element-by-element multiplication operation to give weights to the input data as in equation 5-8:

$$H^{l+1} = H^l \circ s(g) \tag{5-8}$$

At the final process, the channels were also concatenated. Such a gating operation allows us to consider more relevant features after each convolution operation and thus refine our regression process for the following pose estimation case. As we will see in our experiments, this formulation enhanced our neural network's overall performance and drastically reduced its complexity.

5.3 3D neural network for pose estimation and Computer Animation Framework

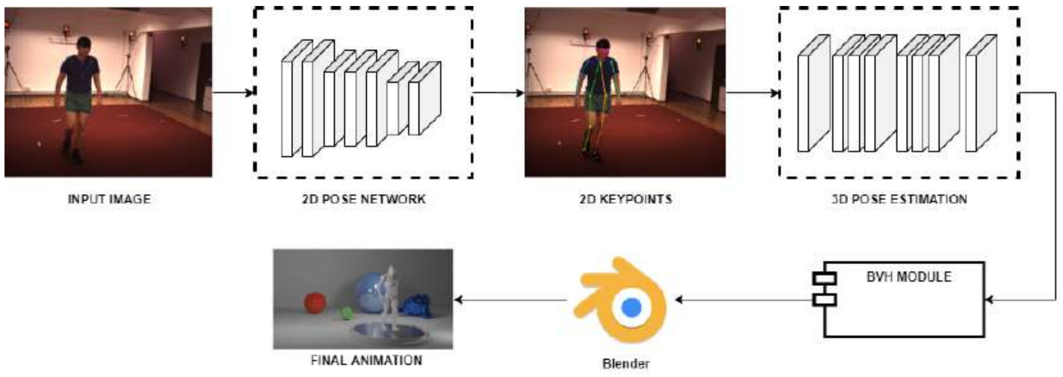

Figure 5.2: Our framework for 3D human pose estimation and computer animation. Here we capture 2D keypoints e interactively regress it to a 3D domain. After we generate 3D motion files and 3D animations in Blender.

We built a 3D human animation framework, where the extracted images from a single RGB camera pass through a 2D pose machine and, after, send the result to our 3D pose network. At the final step, we generate motion BVH capture files. Such a framework can be used in computer animation, games, and also in shared virtual experiences. One of our goals is to provide people with a way to create 3D animations without specialized hardware easily. Each module is independent in our architecture in terms of video processing, inference of captured skeletons, information transmission, and 3D animation. Since all models are decoupled, we can use different 2D pose networks to extract the 2D keypoints. It depends on the user to choose the best model for the application he is developing. Also, we built our framework aiming to be lightweight and accessible. To do so, it does not need any specialized hardware or high-end GPUs. This contributes to shared development experiences involving motion capture, wherein in the context of working from home, in a multi-user production session, creative teams can collaborate remotely. Figure 5.2 illustrates the elements of our framework.

For our 3D human pose regression, with the previous techniques, we propose a new neural network capable of inferring human pose in 3D using only a single camera. To do so, we first extract the 2D keypoints for each person using a 2D neural network. Our neural network model only needs the 2D keypoints output as its input for a 3D regression. This makes our architecture flexible and not dependent on a neural network's specific model to generate keypoints.

Afterward, we develop a new neural network following the semantic graph convolution with the gating mechanism. Our model comprises an input layer with an SGC layer followed by batch normalization and a LeakyRelu activation. The building blocks of our network's internal layers are composed of two SGC layers, also followed by batch normalization and LeakyRelu activation. The output of the second SGC layer is used as the input for our gating mechanism. This is repeated twice, and the blocks also use residual connections. We consider 128 channels for 16 graph nodes in the internal layers, where each node represents a human keypoint. The output layer comprises an SGC layer with the 16 nodes and the 3D positions as output data.

The Figure 5.3 illustrates the design of our network that still has a residual layer for refinement purposes.

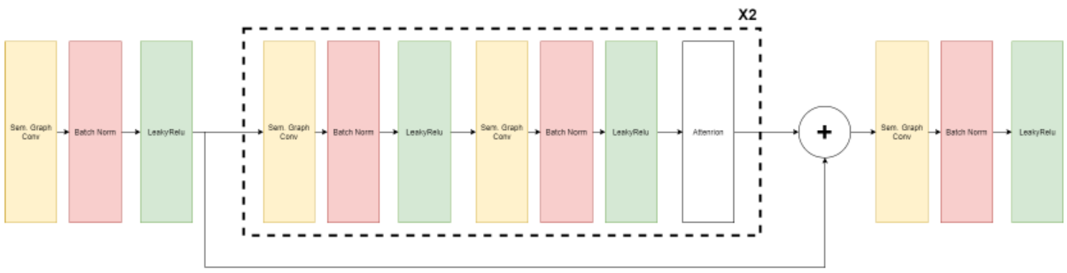

Figure 5.3: Our model for the neural network to estimate the 3D keypoints. Note that we have here 2 internal blocks that uses semantic graph structures followed by an attention block. Also, at the end of each internal block we also have a residual operation.

We sent our network's output to a module that generates a configuration file that can be used in Blender. We use it to create the necessary structure for developing an animated skeleton and applying the captured data to 3D models. This module can also generate JSON files to save our network's output configuration that includes the 2D and 3D keypoint's positions and the camera parameters. In Figure 5.4, we can see examples of our pose estimation capture.

The Rigify structure from Blender is used to automate the creation of rigging controls and bones. We consider the armature, Meta-rigging, and constraints. An armature in Blender is a type of object used for rigging. It can be seen just as similar to a real skeleton consisting of many bones. These bones can be moved around, and anything associated with them will move or deform similarly. A meta-rig is an assembly of bone chains where each chain is identified by the *Connected* attribute, specifying each bone's parent. We generate 3D normalized data for all keypoints for each analyzed frame. In Blender, our script used for converting keypoints into an animated skeleton has the following structures:

- For each keypoint there is an object of type Empty;
- We have a humanoid armature object called Meta-rig where:
  - the Bone hip has the constraint *copy location* with target for the Empty hip object;
  - The other bones have the *stretch to* constraint with the target related to the other Empty objects.
- An object type *humanoid armature* called Rig that will receive the position and rotation of each Metarig bone:
  - Bone hips has *copy location* and *copy rotation* constraints with target for Meta-rig bone hips;
  - The other bones have constraint *copy rotation* with target for the relative bone in Meta-rig.

We convert the 3D captured data to an animated skeleton following a fully automated strategy with our module called *pose3dConvert*. We generate a BVH to provide the captured information to designers, engineers, and animators, where it can be used for standalone animation or in commercial software. To generate the BVH file, for each keypoint, the following actions were performed:

- the x, y, z coordinates are copied to the *location* attribute of the *Empty* object associated with this keypoint;
- A keyframe of type *location* is created for the *Empty* object associated with the keypoint;
- The *Rig* object is selected;

– A BVH file is generated with the Export Animation Operator from Blender.

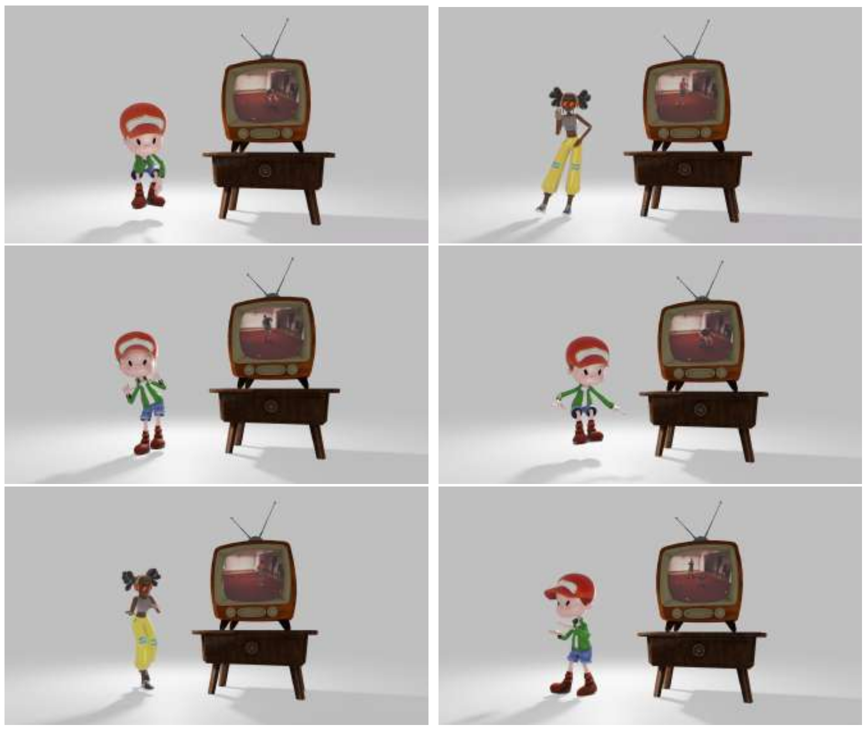
Figure 5.4: Examples of 3D animations crated with our framework.We generate a BVH to provide the captured data, where it can be used for standalone animation or in commercial softwares.As we can see here these example were created with the Human3.6 dataset. (ionescu2013)

5.4 Experimental Results

This section presents the training details of our model, including dataset particularities, camera system, 3D regression, and hyperparameter analysis.

5.4.1 Datasets

We evaluated the proposed method using the Human3.6M dataset for 3D human pose estimation, following the standard protocol. This dataset is publicly available, containing more than 3 million images and 3D data captured by a *MoCap* system and the calculated 2D joints. The dataset contains data from 7 people performing everyday activities such as walking, eating, discussing, etc.

There are two different protocols to evaluate the model trained with Human3.6M considering different approaches to split the data for training and testing. The first protocol, called *Mean Per Joint Position Error (MPJPE)*, consider all four camera views for all subjects. We used 5 subjects for training (1, 5, 6, 7 and 8) and 2 for testing (9 and 11). Furthermore, we calculate the error of the predictions and the ground-truth after aligning them with the root joint, in our experiments represented by the pelvis keypoint. The second protocol is called *Mean per-joint position error after rigid alignment (P-MPJPE)*, which differs from the first protocol only on the alignment. We used the same aforementioned division for training and testing. Moreover, we utilize a rigid transformation to align the predictions with the ground-truth data. All errors were analyzed in millimeters. We also use the COCO dataset (lin2014), a state-of-the-art dataset for 2D human pose estimation in the wild. We use this dataset for pre-training a 2D *Convolutional Pose Machine* and generate 2D input for our method in a qualitative evaluation.

In a second experiment, we use the MPI-INF-3DHP dataset. It was built over a state-of-the-art markerless motion capture system and provide ground truth 3D annotations for humam poses. This can be used as an alternative dataset to Human3.6, were it provide a large range of human motions, interactions with objects and more varied camera viewpoints. In addition to Human 3.6 and CPM detections, we test our approach with this dataset to evaluate the accuracy and generalizability of our learned model. We also use the 3D Percentage of Correct Keypoints (PCK) as evaluation metric. As proposed by Mehta et al.(mehta2017monocular) , we pick a threshold of 150mm for the error, corresponding to roughly half of head keypoint size.

5.4.2 Camera Calibration

Following the same idea proposed by Martinez et al. (martinez2017) and Zhao et al. (zhao2019), we implicitly use the camera calibration in 3D pose prediction. It is impossible to the neural network infer the 3d joint positions in an arbitrary coordinate space (martinez2017). We use the camera frame as global coordinate and implicitly enabling more training data since we consider, for a same subject, more than one camera information. We use the ground-truth 2D joint locations provided in the dataset, following the four camera views, to align the 3D and 2D poses as in the setting of Martinez et al. (martinez2017).

5.4.3 2D to 3D keypoints

Our method can be seen as an unprojection of 2d joint locations to 3D positions. First, we train our network considering the ground-truth for 2D and 3D joint positions from the Human3.6M. However, for a fair evaluation of our method, we also train our network with 2D predictions from a *Convolutional Pose Machine* (wei2016). It is natural to say that our model depends on the quality of the output of a 2d pose detector, and achieves the best results when we use as input the ground-truth 2D joint locations. We evaluate both approaches in our quantitative and qualitative experiments.

We pre-trained a *CPM* with the COCO dataset (lin2014). The COCO dataset skeleton has a different configuration for the human body structure, following the order of keypoints when compared with Human3.6M. We convert the output dictionary of this model to the Human3.6M and train our 3D network.

All 2D keypoints were previously generated in this process. The COCO skeleton has 18 joints considering five joints in the head. The Human3.6M skeleton consists of 16 joints, and we define the spine joint as the root joint. To convert to Human 3.6M and create the spine point, we consider the midpoint between the *lheap* and *rheap* of COCO, and we discoursed the thorax joint.

Also, in a second approach, we train our network trained for on MPI-INF-3DHP pose dataset (mehta2017monocular), comparing the performance of our approach with different network architectures. This dataset is more complex considering poses, clothing and skeleton structure. We use these data to evaluate the robustness and generalizability of our method.

Our 3D network model was trained over 100 epochs, using the *Adam optimizer* with a learning rate of $1e-3$, rate decay of 0.5, and batches of size 128. We also use the *Kaiming* normal function to initialize the weights of each layer. Furthermore, we use the minimum squared error as loss function. We combine joint and bone constraints in human poses and use it in our loss function, which is defined in a similar way that the one proposed by Zhao et al. (zhao2019):

$$loss(B, J) = \sum_{i=1}^{M} \left\| B_i^{'} - B_i \right\| + \sum_{i=1}^{K} \left\| J_i^{'} - J_i \right\|, \tag{5-9}$$

where J' are predicted 3D joint coordinates, B' are bones computed from J', J and B are corresponding ground-truth.

5.4.4 Ablation Study and Network evaluation

We have also analyzed the impact of the chosen hyper-parameters and architecture on the final result in testing. We trained our network with different configurations and compared it to the baseline for *SGCs* (zhao2019) and the baseline for 3D Pose Estimation (martinez2017). We considered the error analysis for protocol 1. In the first test, we compare our model with the state-of-art approaches following 2 configurations: with and without the attention layer. Table 5.1 reports the result. Also, we analyse the impact of using as input 2D prediction from a *CPM*. We use a network configuration with attention layers and 2 blocks and 128 channels per layer. Table 5.2 shows the quality of 3D predictions depends from the input, since the error increases when we use a pre-trained *CPM* in COCO dataset. We compare our approach with the stage 2 of *Xnect* (mehta2020). The *Xnect* try to infer the 3D positions in this stage. Our model surpass their results considering the ground-truth and is competitive when using data from a *CPM*.

Our technique outperforms the state-of-the-art *SGC* (zhao2019) by 3.85%. Also, our model with attention layers surpass the model only with regular *SGCs* (without attention) by almost 10%. It is noteworthy that our approach has much fewer parameters, meaning that using the attention module drastically reduces the network's computational complexity and improves the overall performance. We have approximately 41% fewer parameters than the baseline *SGC* (zhao2019) and 95% fewer parameters than the model from Martinez et al. (martinez2017).

Model	# of Parameters	MPJPE (mm)
SGC (zhao2019)	0.43 M	43.8
Martinez et al. (martinez2017)	4.29 M	45.5
Ours without attention (2 blocks and 128 ch)	0.16 M	46.71
Ours with attention (2 blocks and 128 ch)	0.18 M	**42.11**

Table 5.1: 3D pose regression errors and the parameter numbers of our networks with different settings on Human3.6M.

Model	MPJPE (mm)	P-MPJPE (mm)
Ours (Ground Truth)	**42.11**	33.29
Ours (CPM detections)	75.40	59.91
Xnect (SIGGRAPH'20) (mehta2020)	63.6	-

Table 5.2: 3D pose regression errors with different inputs. We use 2D ground-truth from Human3.6M and 2D predictions from a CPM. We compare our results with the stage 2 output of the state-of-the-art (mehta2020). The MPJPE metric for Xnect where obtained from the original paper.

In a second test, our models were trained for over 100 epochs under three configurations. The first is a model with two internal blocks and 64 channels, the second is our regular model with two inner blocks, and in the third, we use a model with four internal blocks and 128 channels. Table 5.3 shows that with the second configuration our model performs better than the baseline algorithm from Zhao et al. (zhao2019) and the other two configurations.

We evaluated our 3D unprojection model following the dataset Human 3.6M, not considering the influence from a 2D pose detector. Table 5.4 shows the result using 2D Human3.6M ground-truth for training and testing. The results are competitive and, on average, our performance is better than the state-of-the-art.

Most methods have sophisticated frameworks (pavllo2019, rayat2018, yang2018) or learning strategies. They were trained and focus on in-the-wild images, propose end-to-end frameworks to generate the 3D pose directly from images, consider temporal information and also use complex loss functions (dabral2018learning, pavllo2019). Due more data variability and their proposed constraints to reduce prediction error, it was expected that they have better performance. However, this is not true. Our model surpass the previous works, considering the *MPJPE* and *P-MPJPE*, proving the potential of the attention layer. The tests consider each action of the motion capture dataset. Table 5.4 shows the error in millimeters for each step following protocol 1 MPJPE.

Our results on Human3.6M under protocol 2 (using a rigid alignment with the ground-truth), are shown in Table 5.5. In most cases, our method surpasses the previous works and has better performance on average. Note that in some cases, our model has similar performance or worse than the model from Dabra et al. (dabral2018learning). However, our approach has fewer parameters to compute and do not need complex anatomically loss functions or a sophisticated weakly supervised learning framework. Also, in average, our model outperforms Dabra et al. (dabral2018learning) by 8.3%.

In Table 5.6, we compare the 3D pose output on the MPI-INF-3DHP

Model	# Parameters	MPJPE (mm)
Baseline ((zhao2019))	0.43 M	43.8
2 blocks and 64 channels	0.06 M	43.88
2 blocks and 128 channels	0.18 M	42.11
4 blocks and 128 channels	0.36 M	43.04

Table 5.3: Evaluation of our parameters for the 3D pose estimation model. The error is computing in the testing dataset. As we can see, our best configuration has approximately 41% fewer parameters than the baseline achieving the state-of-art performance

Protocol	Direct	Discuss	Eating	Greet	Phone	Photo	Pose	Purch.	Sitting	SittingD	Smoke	Wait	WalkD	Walking	WalkT	Average
Martinez et al. ICCV'17 (martinez2017)	51.8	56.2	58.1	59	69.5	78.4	55.2	58.1	74.0	94.6	62.3	59.1	65.1	49.5	52.4	62.9
Yang et al. CVPR'18 (yang2018)	51.5	58.9	50.4	57.0	62.1	65.4	49.8	52.7	69.2	85.2	57.4	58.4	43.6	60.1	47.7	58.6
Mehta et al. SIGGRAPH'17 (mehta2017)	62.6	78.1	63.4	72.5	88.3	93.8	63.1	74.8	106.6	138.7	78.8	73.9	82.0	55.8	59.6	80.5
Hossain & Little ECCV'18 (rayat2018)	48.4	50.7	57.2	55.2	63.1	72.6	53.0	51.7	66.1	80.9	59.0	57.3	62.4	46.6	49.6	58.3
Zhao et al CVPR'19 (zhao2019)	**37.8**	49.4	37.6	40.9	45.1	**41.4**	**40.1**	48.3	50.1	**42.2**	53.5	44.3	40.5	47.3	39.0	43.8
Pavllo et al. CVPR'19 (pavllo2019)	45.1	47.4	42.0	46.0	49.1	56.7	44.5	44.4	57.2	66.1	47.5	44.8	49.2	32.6	**34.0**	47.1
Dabra et al ECCV'18 (dabral2018learning)	44.8	50.4	44.7	49.0	52.9	43.5	45.5	63.1	87.3	51.7	61.4	48.5	**37.6**	52.2	41.9	52.1
Our Model (2 blocks, 128 ch)	39.82	**44.14**	**36.54**	**40.27**	**41.64**	48.19	42.39	**39.33**	**49.64**	59.06	**41.66**	**42.2**	41.59	**31.02**	34.15	**42.1**

Table 5.4: Results under Protocol 1 on Human3.6M (no rigid alignment in post-processing). Note that in average our model surpasses the previous state-of-the-art approach. The results of all approaches are obtained from the original papers.

Protocol	Direct	Discuss	Eating	Greet	Phone	Photo	Pose	Purch.	Sitting	SittingD	Smoke	Wait	WalkD	Walking	WalkT	Average
Martinez et al. ICCV'17 (martinez2017)	39.5	43.2	46.4	47.0	51.0	56.0	41.4	40.6	56.5	69.4	49.2	45.0	49.5	38.0	38.0	47.7
Yang et al. CVPR'18 (yang2018)	26.9	30.9	36.3	39.9	43.9	47.4	28.8	29.4	36.9	58.4	41.5	30.5	29.5	42.5	32.2	37.7
Hossain & Little ECCV'18 (rayat2018)	35.7	39.3	44.6	43.0	47.2	54.0	38.3	37.5	51.6	61.3	46.5	41.4	47.3	34.2	39.4	44.1
Pavllo et al. CVPR'19 (pavllo2019)	34.1	36.1	34.4	37.2	36.4	42.2	34.4	33.6	45.0	52.5	37.4	33.8	37.8	25.6	27.3	36.5
Dabra et al. ECCV'18 (dabral2018learning)	**28.0**	**30.7**	39.1	34.4	37.1	**28.9**	**31.2**	39.3	60.6	**39.3**	44.8	**31.1**	25.3	37.8	28.4	36.3
Our Model (2 blocks, 128 ch)	28.14	33.75	**31.49**	**31.15**	**32.92**	38.16	31.86	**30.20**	**41.52**	48.82	**33.65**	32.72	**32.87**	**24.94**	**27.08**	**33.29**

Table 5.5: Results of protocol 2 on Human3.6M under rigid alignment in post-processing. Note that in most cases, our model surpasses the previous works. The results of all approaches are obtained from the original papers.

Model	MPJPE (mm)	3D PCK
Vnect (mehta2017)	124.7	76.7
M3DHP (mehta2017monocular)	117.6	75.7
(mehta2018single)	122.2	75.2
Xnect (stage 2) (mehta2020)	98.4	82.8
Xnect (stage 3) (mehta2020)	115.0	77.8
(kanazawa2018)	124.2	72.9
Ours (2 blocks and 128 channels)	105.17	81.27

Table 5.6: Comparison on the single person MPI-INF-3DHP dataset. Top part are methods designed and trained for single-person capture.The Xnect is multi-person method, however we evaluate only single person predictions.

dataset (mehta2017monocular) using the using the 3DPCK (higher is better) and MPJPE. We prove the robustness of our method, where for both metrics we surpass the previous state-of-the-art approaches. Again, note that most of these methods are built over sophisticated frameworks and are capable to predict multi-person poses while our method can be seen as a 2D to 3D unprojection.

Moreover, considering time performance, our 3D network took, on average, 10 seconds to evaluate 1062 poses from the 2D inference. The tests were performed in a GPU Nvidia RTX 2060 with 6GB of memory, where we repeat each test 1000 times. In terms of the number of parameters, our network has 0.18M, while the model proposed by Zhao et al.(zhao2019) has 0.48M. This means that our network is lightweight and could be part of a full system that infers 3D human pose in real-time.

5.4.5 Qualitative results

Figure 5.5 illustrates some results generated using images from COCO dataset (lin2014). Our model is able to accurately predict 3D poses from these images indicating that it effectively encodes relationships among body joints and can generalize the results to different situations. The input of the method is the 2D joints generated using as *Convolutional Pose Machine*. However, our model also has some limitations as we can see in the last row of figure 5.5. For example, when using data predicted by a *CPM*, if the 2D detector output fails to detect all body keypoints, it is impossible to our model recover the missing information. Also, for in the wild examples, it is not uncommon to see images with occluded or incomplete poses. Our model has difficulty to deal with these cases. Both approaches proposed by Martinez et al. (martinez2017) and Zhao et al. (zhao2019) have the same issue.

Figure 5.6 shows the results of our technique applied on Human3.6M. In another approach, the input is generated by the method from Schirmer et. al. (Schirmer2019), and from the output we created a BVH model to generate the animation.

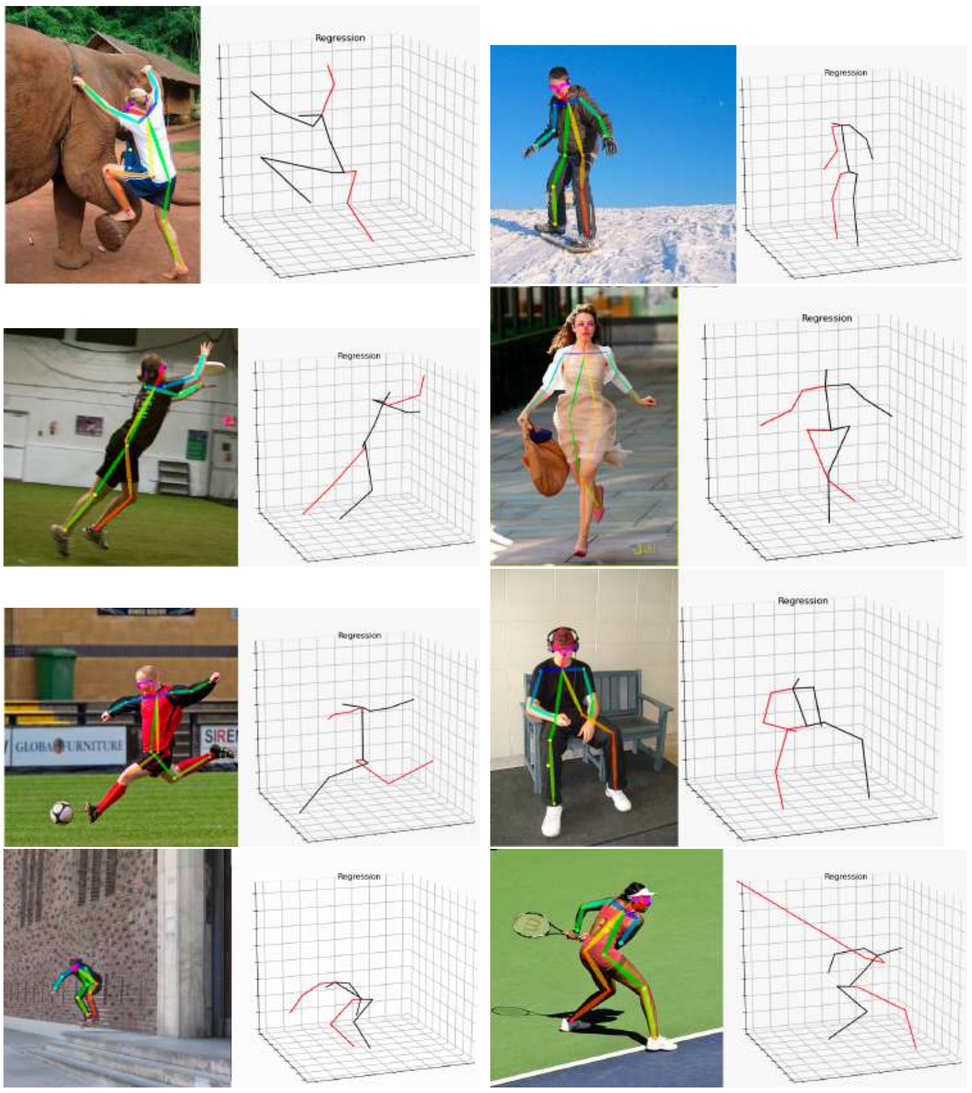

Figure 5.5: Visual results of our method on in-the-wild images from COCO dataset (lin2014) . In most cases, our technique can effectively predict 3d joints in different situations. Small errors can be seen considering the image scale and camera projection. In the last row, in an example with self-occlusion, our model cannot predict data from incomplete data.

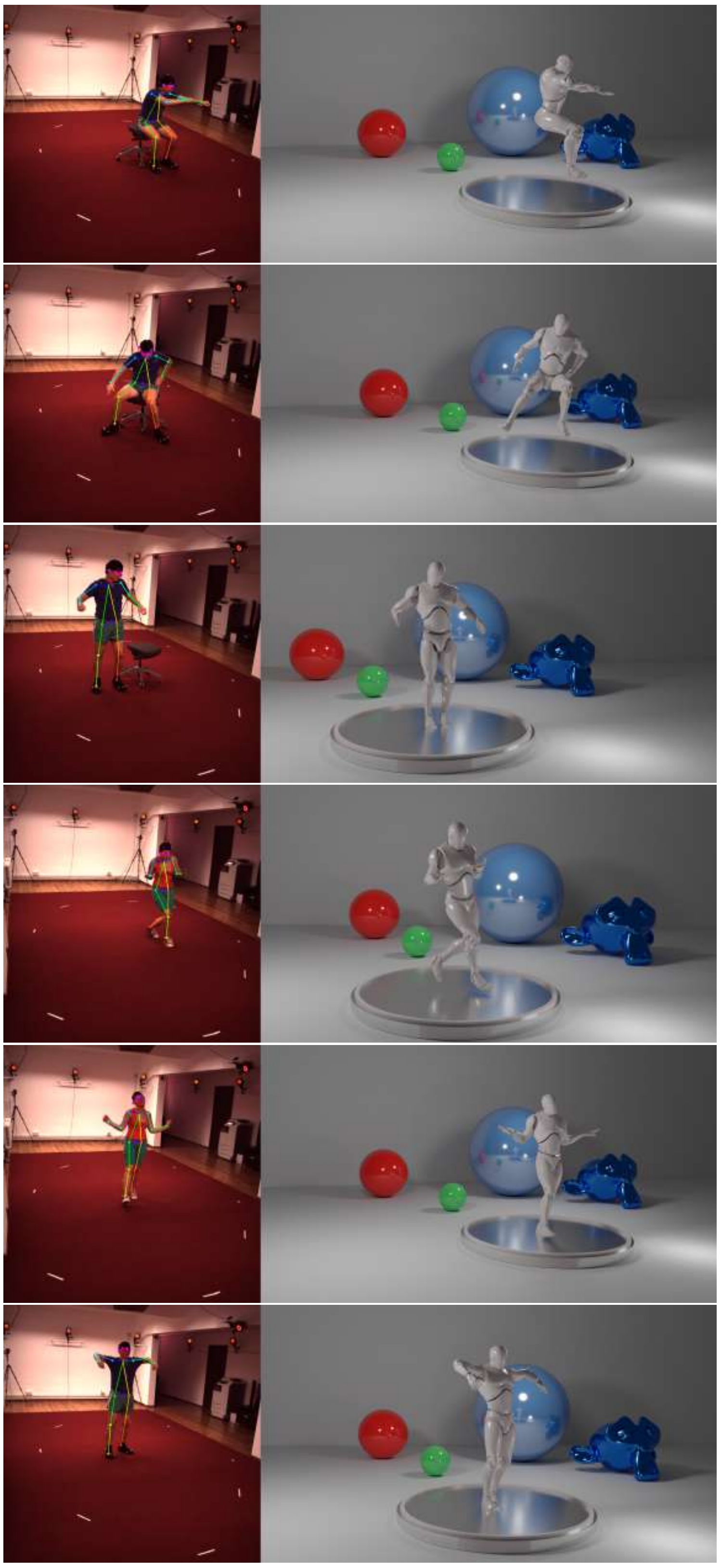

Figure 5.6: Visual results of our method on Human3.6M (ionescu2013). As we can see, our method is robust but still has minor issues considering joint rotations. As we said before, our model focus only on project the human keypoints in a 3D space.

6
Realtime Applications

In this section, we present two real-time application which uses our techniques for pose estimation. We called all modules of these applications as Tensorpose project. In the next sections, we described the architecture's main aspects of real-time applications and a framework for Human digitization, which uses our techniques.

6.1
Archictecture for realtime applications

We developed an architecture with the aim of being a platform for the development of 2D and 3D applications that involve motion capture. In this section, we will describe the general aspects of its architecture, as well as the framework for network communication of different applications. Such applications can be computer animation, games and also shared virtual experiences. One of our goals is to provide developers with a way to create environments for shared 2D or 3D experiences where users share a virtual environment, but not necessarily at the same physical environment. In the actual scenario of social distancing, this approach can enable people to work remotely in a collaborative and low cost approach.

The motion capture data can be used to track the activities of users in this kind of application. In our architecture, each module is independent in terms of video processing, the inference of captured skeletons, temporal coherence, and information transmission. Figure 6.1 represents the architecture model.

As one can see in Figure 6.1, the architecture model is composed by several modules. Next we will describe the functionality of each one.

Pose Client This module is responsible for processing video data from different cameras. Here, each frame is prepared using the OpenCV library (bradski2008), where we also do an equalization of the histogram to improve the contrast in the images and send the data to the inference module or the temporal coherence module. In this case, here, the resolution of the image is 1280x960, and the network resolution is 656x386 to fit in GPU memory.

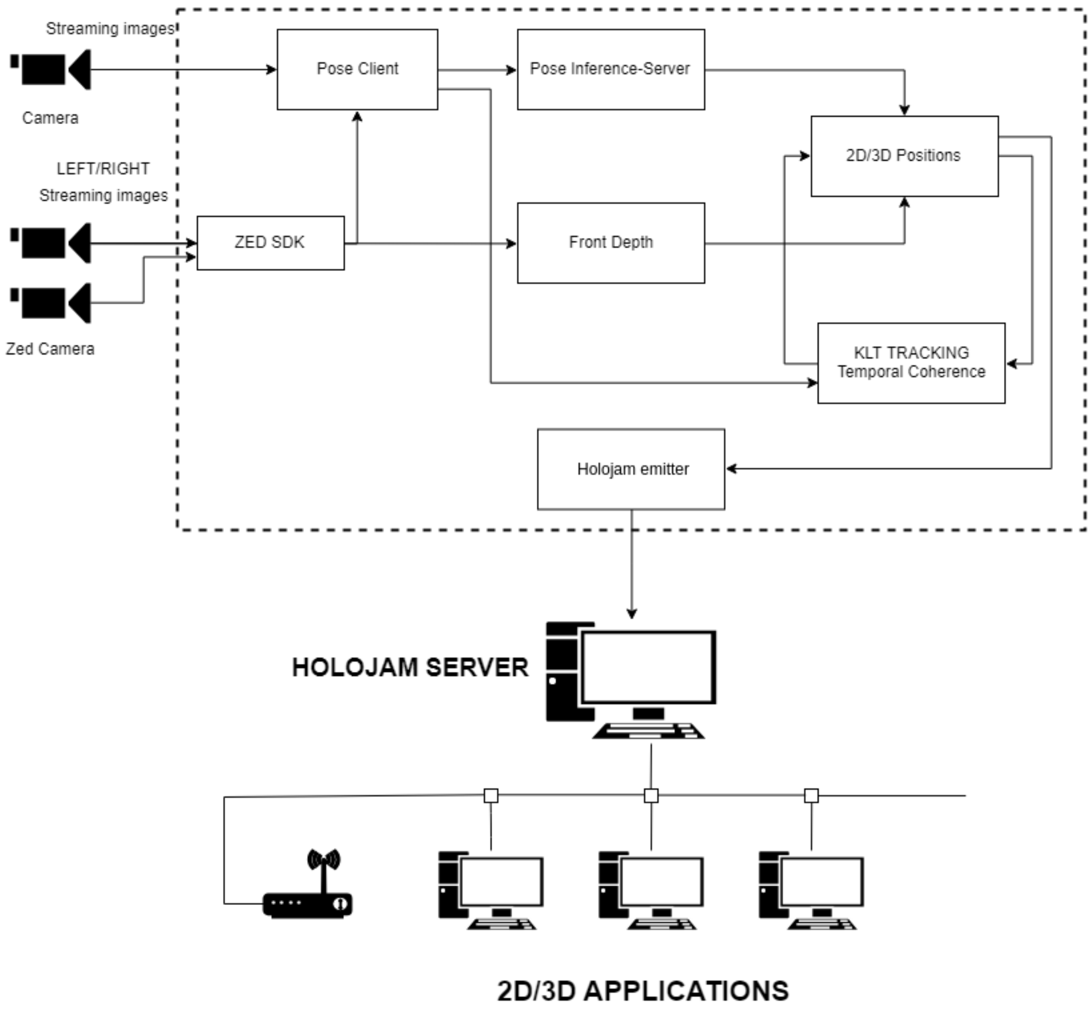

Figure 6.1: TensorPose Architecture

Pose Inference This neural network module receives data from the pose client and performs the inference of skeletons in an image. Also, a post-processing is performed to solve the assignment problem. For each set of key points detected and for each person, a dictionary is created, which contains the 2D position of each body part and the connections between them. For each recognized person is given an Id and their corresponding dictionary is added to a list of tracked persons who are sent to the module responsible by the instantiating objects, which will represent the capture of those people.

Zed SDK and Depth Front In this stage, which is an optional source, we use the ZED stereo camera SDK (burbano2016)(gupta2017) for 3D data capture without using a network for 3D prediction. This module was developed using the Python API, where data from a 2D image is sent to the Pose Client to be processed and the 3D data is sent to the depth module. However its use do not achieve the same performance and precision of using our 3D neural network approach. Also, this module is responsible for processing the depth data of one or more ZED cameras and receiving intrinsic camera data. ZED has two cameras separated by 12 cm, which capture a high-resolution 3D video of the scene and estimate depth and motion by comparing the displacement of pixels between the left and right images. This module stores a distance value Z for each pixel in the image.

2D/3D Positions This part of the architecture receives data from the positions inferred by the network and creates objects identifying each detected person. In case of 2D, keeps the positions received in coordinates of the image. In the case of 3D, it receives data from Front-Depth or from the 3D neural network. It is also responsible for passing information to the temporal coherence module, also processing the Kalman filter, identifying the points to be tracked and ensuring the reference of the detected skeletons.

Temporal Coherence This module tracks the points using the Optical Flow algorithm. It receives the initial data detected by the network, and every t frames perform the tracking of the points.

Holojam Emitter Uses the Holojam platform to send the captured data via the network. It is a Python client that communicates with the Holojam Server. The Holojam platform is described in the following section.

6.2 Virtual enviroment communication by Holojam

Holojam is a virtual space sharing platform developed by the Future Reality Lab of the New York University (masson2017). This platform consists of the Holojam Node library and the Holojam SDK project. It enables content creators to build complex location-based multiplayer VR experiences in a simple and unified Unity project. The development framework provides an extensible and clean interface, allowing rapid prototyping. Additionally, it abstracts away specific VR hardware, promoting a flexible and customizable creation of virtual reality experiences. We use the Holojam protocol framework in our architecture to communicate applications with very low latency, and to send motion capture information through the network, expanding the use to not only VR applications.

6.3 Holojam Node

We use the Holojam Node, which is a client-server library developed in Node.js and targeted to applications running on a local network or over the web. One of its main characteristics is to perform low-latency communication between the various clients and the server. It consists of the following components: relay, emitters, sinks, and clients.

The relay component is responsible for routing the messages (updates and events) in a Holojam network. It acts both as a central server, which collects data received through a preconfigured (upstream) address, as well as performs a broadcast of this data through a multicast (downstream) address.

In addition to the relay in a Holojam network, there are also several nodes, called endpoints. Endpoints are either emitters, which only send upstream data; sinks, which only receive data through the downstream; or clients that receive downstream data and send upstream data. Holojam Node also provides a WebSocket interface where you can receive and transmit data over the web.

6.4 Holojam Protocol

Packets routed through a network are either hosted or updated. An update is essentially an array of flakes (generic Holojam objects). An event has an array containing only one flake. There is also a notification that is an abstraction of an event that includes just the label.

6.5 Holojam Objects

Holojam provides two types of objects: Nuggets and Flakes. These objects are defined through Google's FlatBuffers Interface Description Language (IDL). We use Flatbuffers to serialize data, as well as streaming data, because it offers very low processing overhead. As follows, we present the structure of the objects used.

Nugget Composition:

– It is event type or update. Default is update;

– It is mandatory to have an array of flakes.

```
enum NuggetType : byte { UPDATE, EVENT }

// Message (update or event)
table Nugget {
   scope : string; // Namespace
   origin : string; // Source

   type : NuggetType = UPDATE;
   flakes : [Flake] (required);
   // Data array
}
```

Flake Composition:

– A label is mandatory

```
table Flake { // Data container
   label : string (required);

   // Optional data

   vector3s : [Vector3];
   vector4s : [Vector4];

   floats : [float];
   ints : [int];
   bytes : [ubyte];

   text : string;
}
```

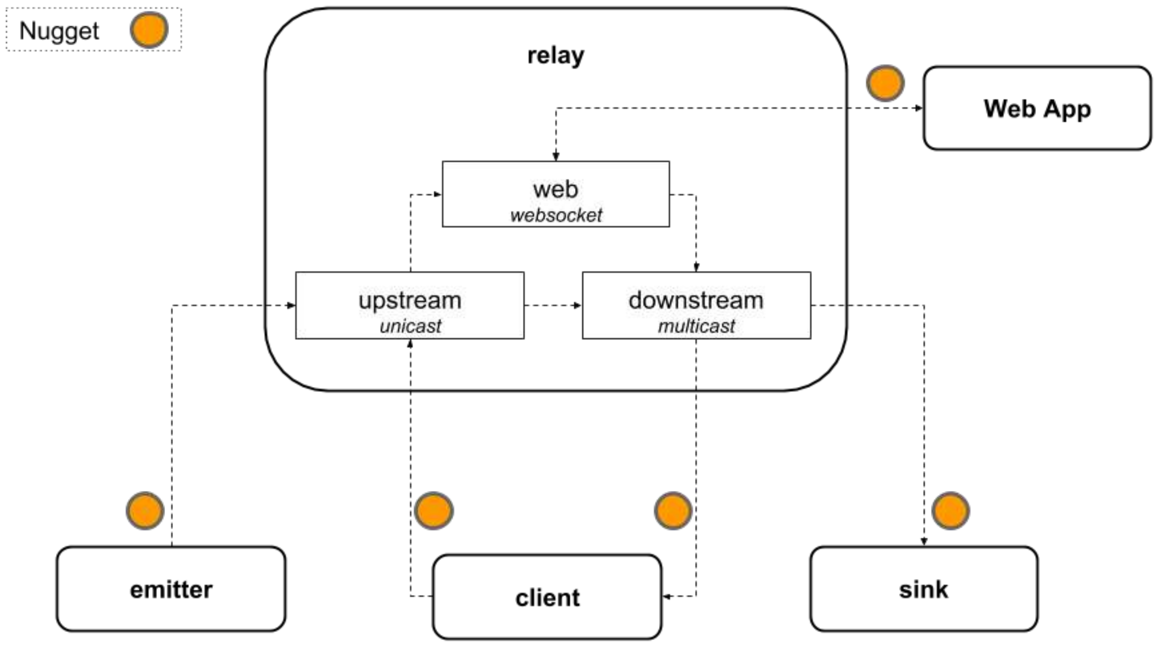

Figure 6.2: Holojam Architecture

6.6 Holojam SDK

The Holojam SDK project is a Unity3D project containing all the elements needed to create a multiplayer virtual reality application. This project provides a API that allows the application created to use or extend this API allowing for rapid prototyping. Also, it abstracts the configuration of the VR hardware used in the project. One of its main components is the implementation of the Holojam client in . Thus, all Holojam objects can be integrated easily into the Unity project.

6.7 Holojam and realtime applications

All components of the Holojam platform were used in the TensorPose project applications, both in 2D applications as well as 3D and VR applications. Since TensorPose is implemented in Python, the components needed to use Holojam in TensorPose were also deployed in Python, including a serializer for the keypoints and the emitter. The serialization of the obtained keypoints was implemented from the IDL FlatBuffers of Holojam. With this, it is possible to send the keypoints to the relay, giving access to data to any sink/client/websocket that is connected to this relay.

Figure 6.3 shows the components for various applications created in the TensorPose project and to which other component they are related to.

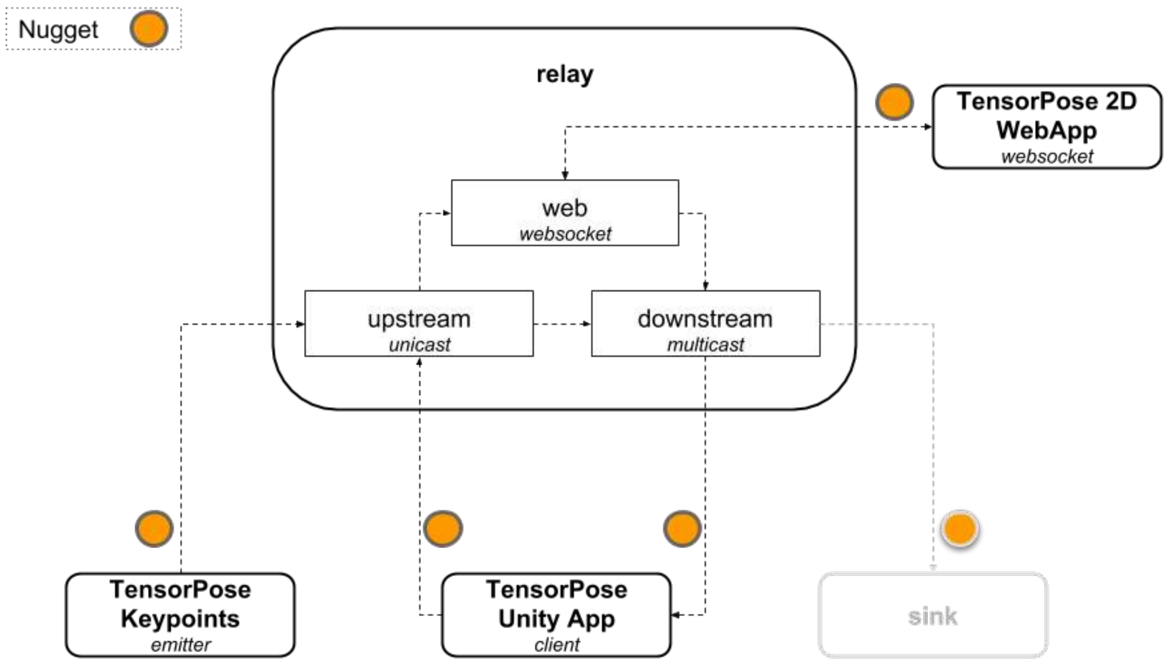

Figure 6.3: TensorPose network infrastructure. The Relay component is responsible for synchronizing and sending information over the network to the clients of our applications.

Our applications were developed in 3 steps. The initial module was developed for 2D applications where capture could be done through ordinary cameras such as webcams. Each frame is captured and sent to the detection module, which is in charge of performing the inference and detection of the people in each frame. Subsequently, the data regarding the captured persons, are sent via the net using the framework Holojam (masson2017).

In addition to the 2D capture, tests involving stereo cameras were also conducted. We used the ZED camera to map three-dimensional coordinates of the world from two-dimensional coordinates of images. The skeletal position is initially processed in the same way, but, in a later step, it is transformed into 3D space using intrinsic camera data and depth information. As in the previous process, the capture and processing modules are independent, with Holojam also being used to send information. Also, conisdering the use of just 1 monocular RGB camera, we explore our 3D neural network for pose estimation in these applications. The data were sent in the same way.

Some applications were developed as a proof of concept using the Unity 3D engine as well as WebGL. Regarding applications in Unity3D, the data is sent via the network by the Holojam Server, using multicast. Regarding the 3D scene in the Unity engine, "Holojam Unity" objects implement support for multiplayer, which includes communication with the Holojam Server, components with position recognition, and Avatar presence. These objects have a script component called "Holojam Network", which contains some fields indicating how to communicate with the server.

The "Multicast Address" field indicates the IP address used to receive data from the server. This field is not relevant if there is already a unicast connection between this client and the server since, in this case, the client will receive the data directly from the server. The Server Address field can be used to indicate the IP of the server directly. This address is used to send data to the server, but if the indicated address is not local (not starting with 192), Holojam will ping this address, which creates a connection between the server and this client. All data that is sent and received uses an update data structure. When Holojam sends an object, it uses the label “SendData" for such an update. For the virtual characters, they are created from the tracked data of the real person using inverse kinematics. A manager implemented in the application reconstructs the entire body of the Actor from pairable elements corresponding to the connections between hands, elbows, shoulders, neck, head, and so on. The reconstructed data are used as targets for a 3D model in the scene. Figure 6.4 shows the Unity3D application.

The OpenPose also has a plugin for Unity 3D, although it is limited to run the inference locally and do not achieve the necessary performance. Their plugin often encounters problems when considering low cost GPUs, where it consumes a lot of memory and frequently is necessary to run in CPU mode. Also, unlike our application, their plugin does not deals with possibility to create shared virtual experiences where the physical presence of users in the same environment is not necessary. In our proposal, we can have a server dedicated to inference, without the need to make intensive use of the local machine. The entire project was developed considering network communication using the low latency model of Holojam. All modules in the Figure 6.1 were developed to be independent.

Similarly, a web application with a unicast connection was developed, as we can see in Figure 6.5. The entire application was designed in javascript using WebGL. A script called “Manager" was implemented, responsible for receiving the data sent by the server, considering each object that identifies each person tracked and the position of their skeleton. The Manager instantiates objects called characters for the identification of actors, while still being in charge of updating their positions to each frame. It also manages the removal of actors from the scene, if they are no longer in it, or if the track has been lost. Subsequently, the characters are rendered as a ragdoll using the position of each part of their skeleton directly, not using inverse kinematics, in this case.

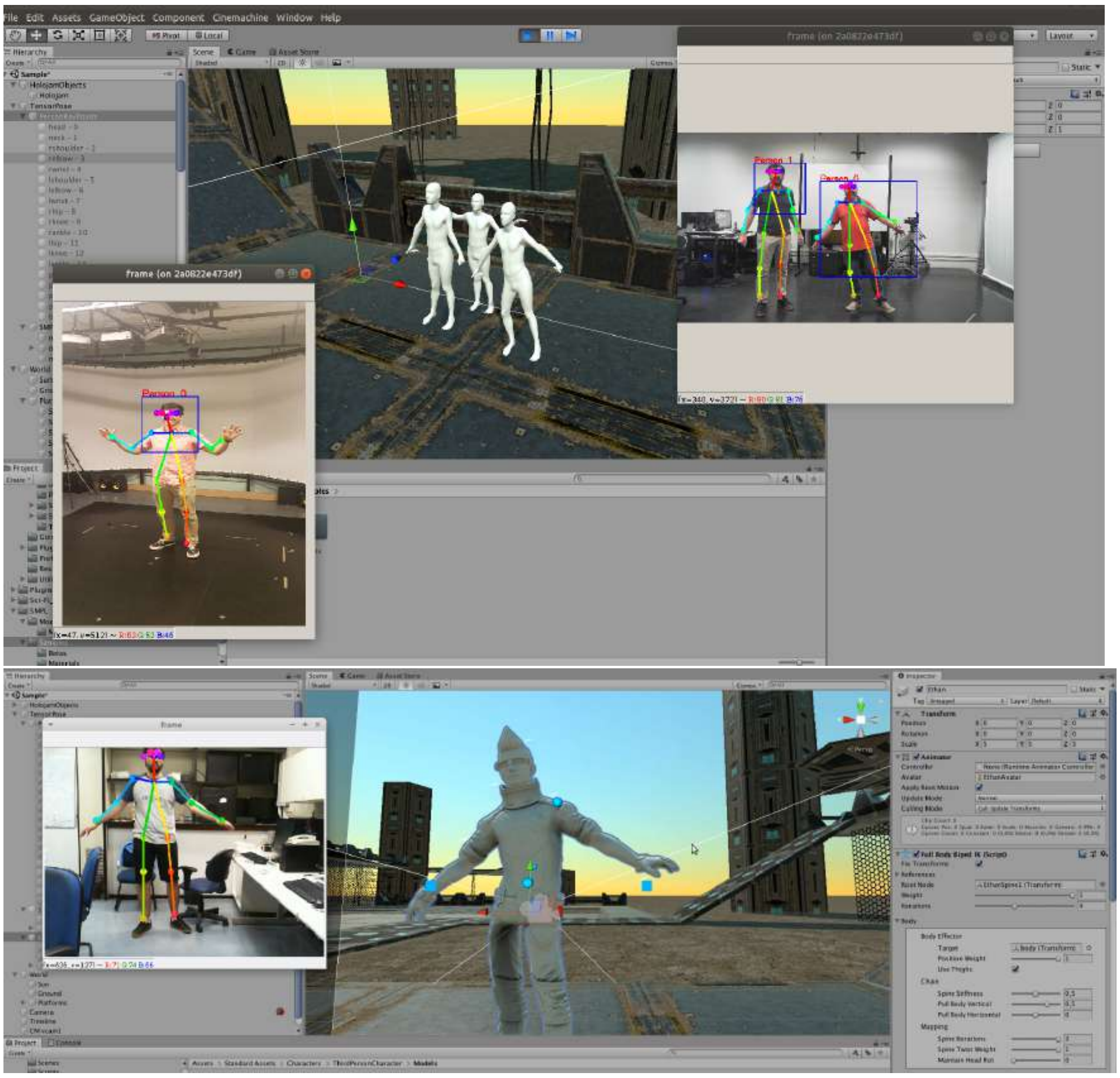
Figure 6.4: Unity3D application with TensorPose. One of our use cases is the development of applications that make use of shared virtual environments.

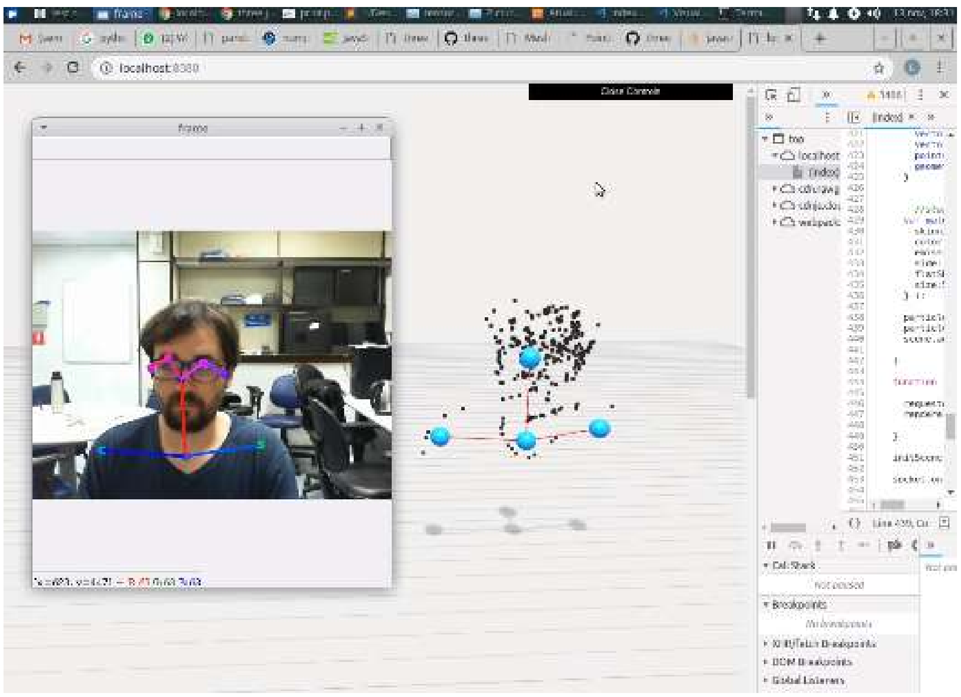
Figure 6.5: TensorPose web test.

6.8 3D Human Digitization

3D motion capture, pose estimation, texture capture, and volumetric capture are important tasks to generate content for computer animation, in particular for human Digitization. Kanazawa et al. (kanazawa2018) show an end-to-end framework for reconstructing a full 3D mesh of a human body from a single RGB image. They used the generative human body model, SMPL(loper2015), which parameterizes the mesh by 3D joint angles and low-dimensional linear shape space. An image is passed through a convolutional encoder and sent to a 3D regression module that infers the 3D representation of the human. This mesh can be useful to generate humanoid animations, which could immediately be used by animators.

When we consider volumetric capture in a studio, this is not only a costly technology but also depends on specialized hardware. Moreover, it is far from being accessible to most producers. We can find solutions that present alternative ways to reduce the cost and computational processing for this kind of application. For example, Pandey et al. .(pandey2019volumetric) proposed a method to synthesize free-viewpoint renderings using a single RGBD camera. Besides the impressive results, it seems to be far to be applicable in real situations. Also, considering texture capture, Saito et al. (saito2019pifu) introduce Pixel-aligned Implicit Function (PIFu), which is an implicit representation that locally aligns pixels of 2D images with the global context of their corresponding 3D object, although their techniques seem to be computationally intensive.

Despite the independent advances in each area, there is still no proposal that uses texture capturing, pose estimation, and mesh recovery in a unified way to generate virtual characters using an accessible and low-cost architecture. We have developed a low cost and accessible end-to-end framework for 3D modeling and texture capture of Humans using deep neural networks and a single RGB camera. We create an end-to-end approach to generate virtual characters based on image segmentation, pose estimation, and human mesh recovery(HMR). We apply the HMR method (kanazawa2018) to the captured data to generate 3D shape models and finally combine with the generated textures to obtain a full 3D reconstruction of the human body that can be used in a game engine.

We divided our method into two stages: texture extraction from a set of images I and the 3D modeling. For texture extraction, we use the DensePose model (alp2018densepose) to generate our texture atlas.They propose a variant of a Mask-RCNN, to densely regress part-specific UV coordinates within every human region at images or videos. Since we have I images with different views as input, we produce N partial atlas and after we compose them. Using this model, we can extract each pixel that relates to a specific body part of a person detected in each image. While the original article, seeks to provide a texture transfer, i.e., mapping textures previously provided to image pixels based on estimated correspondences, we aim to do the inverse. We map each pixel of an estimated coordinate into a correspondent pixel in a texture atlas. In other words, DensePose predicts the UV coordinates of 24 body parts, and we compute a look-up table to convert the DensePose UV maps to the SMPL UV parameterization (loper2015). Figure 6.6 show the processe to generate the UV maps.

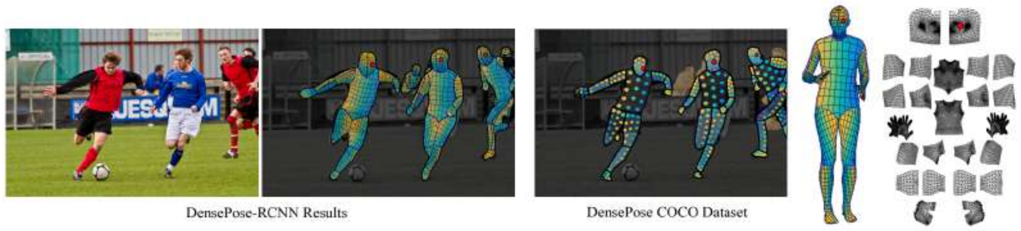

Figure 6.6: DensePose process to extract the texture atlas (alp2018densepose).

We also use partial convolutions(liu2018image) to fill small gaps in the texture since using the previous method is not possible to fill the entire UV maps. In this model, the convolution is masked and re-normalized to be conditioned on only valid pixels. We use the textures provided by the SURREAL dataset (varol2017learning) in the training process, where we create patches of size 32×32 for each image, and also do data augmentation by rotating and using noise to create different masks.

Another problem is the color discontinuity between the parts of the mapped textures. This discontinuity is caused by different illumination conditions while capturing the images due to the fact the pictures are taken by modifying the relative position between the camera and light source. To solve this problem, we subdivide the atlas following the part division of DensePose and use a method similar to presented by Junior(junior2006variational). Considering the frontier between the parts, we use a method that diffuses the color difference between the frontier zone of adjacent areas for each part. Let r be the radius distance considering each frontier edge; for each point in the line, we calculate the color difference between corresponding texels and, after with these correction factors, perform a diffusion of them over the whole texture space. As proposed by junior et al. (junior2006variational), we consider sparsely-defined texels as heat sources and solve the problem applying the diffusion equation on each heat source, which represents the flow of heat from that source. The factors between frontier edges remain fixed, and other values are relaxed across the image.

In our second stage, considering the 3D model, we aim to generate the model from an initial pose. We use our pose estimation models for this task. We use our 2D pose estimation model to capture the 2D joint positions. Subsequently, the obtained data is sent to the HMR method to infer the 3D mesh adapted from the captured person. The input for HMR is a 2D set of joints and a RGB image. Figure 6.7 show the results our approach.

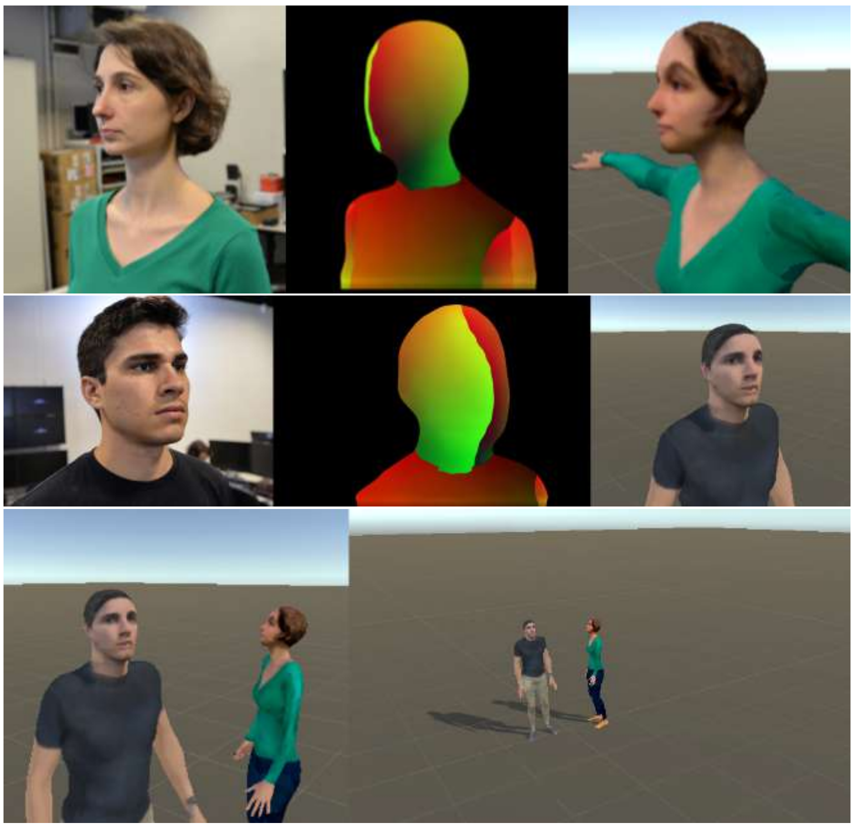

Figure 6.7: 3D reconstructed models. Here we apply the texture captured in the first step over the mesh gerenerated by the second.

7
Conclusion

In this work we proposed a novel deep neural network with streamlined architecture and tensor decomposition for pose estimation with improved processing time, named TensorPose. We adopted it on a real-time motion capture multi-user interactive application. Considering our results, we show an efficient optimization in the CNN model, where in major cases represents $3\times$ the performance of the original work. The accuracy of the detection of keypoints shows to be slightly smaller. However, this fact does not represent great problems in our applications, since our primary objective is the performance gain, when considering FPS. As we showed in Figure 4.12 some failure cases were detected, but do not represents major problems in our applications. Also, we present a statistic to each kind of error detected. In an evolution for the 2 model, we also presente a new deep neural network with a lightweight architecture, attention blocks, and tensor decomposition for pose estimation with improved processing time. We provided an efficient optimization in the CNN model when considering its use in modest GPU hardware and CPU. This is a result of our factorized convolutions involving the Tensor decomposition theory. In parallel to handcrafted approaches for factorized models, this can be considered a technique with great potential. We explored the weakness of convolutional neural networks using attention mechanisms as an auxiliary method to focus on global information for our applications besides the local neighborhood. The redundancy in the parameters of Convolutional Pose Machines following tensor decompositions and attention mechanisms was significantly reduced. However, as in OpenPose and CPMs, our model faces problems in predict keypoints considering occlusion and crowded images.

We also present a novel model for attention layers in Semantic Graph Convolutions. With this approach, we build a lightweight 3D human pose estimation model to project 2D keypoints from the output of a convolutional pose machine in a 3D space. To do that, our model can be seen as a regression from 2D to 3D keypoints. We use camera parameters implicitly, using the camera attributes from Human3.6. As we can see in the experiments, the combination of SGCs with attention layers improves the performance and reduce the overall complexity of our model and we achieve state-of-the-art performance with 41% fewer parameters. This model takes as input the 2D keypoints inferred from our 2D neural networks, which impacts the performance of the 3D prediction. As a limitation of our model, we need as input all 2D keypoints predicted. If the 2D model misses one of them, this impacts the performance of the 3D regression.

As a proof of concept, we present two main applications using Unity3D and WebGL, integrated with Holojam platform. We show that it is possible to create shared projects for motion capture with the TensorPose.

As future works, we intend to adapt our models to embedded devices, such as cellphones, and create mobile applications. We believe that our framework can be very useful for people easily create 3D animations in a simple way, without any specialized hardware.

Bibliography

[Schirmer2019] SILVA, L. J. S.; DA SILVA, D. L. S.; RAPOSO, A. B.; VELHO, L. ; LOPES, H. C. V.. **Tensorpose: Real-time pose estimation for interactive applications**. Computers & Graphics, 85:1 – 14, 2019.

[Senst2016] SENST, T.; GEISTERT, J. ; SIKORA, T.. **Robust local optical flow: Long-range motions and varying illuminations**. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, p. 4478–4482, Phoenix, AZ, USA, Sept. 2016. IEEE. IEEE Catalog Number: CFP16CIP-USB ISBN: 978-1-4673-9960-9 DOI:10.1109/ICIP.2016.7533207.

[alp2018densepose] ALP GÜLER, R.; NEVEROVA, N. ; KOKKINOS, I.. **Densepose: Dense human pose estimation in the wild**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 7297–7306, 2018.

[babenko2009] BABENKO, B.; YANG, M.-H. ; BELONGIE, S.. **Visual Tracking with Online Multiple Instance Learning**. In: CVPR, 2009.

[bahdanau2014] BAHDANAU, D.; CHO, K. ; BENGIO, Y.. **Neural machine translation by jointly learning to align and translate**. arXiv preprint arXiv:1409.0473, 2014.

[bello2019] BELLO, I.; ZOPH, B.; VASWANI, A.; SHLENS, J. ; LE, Q. V.. **Attention augmented convolutional networks**. arXiv preprint arXiv:1904.09925, 2019.

[bradski2008] BRADSKI, G.; KAEHLER, A.. **Learning OpenCV: Computer vision with the OpenCV library**. " O'Reilly Media, Inc.", 2008.

[bruna2013] BRUNA, J.; ZAREMBA, W.; SZLAM, A. ; LECUN, Y.. **Spectral networks and locally connected networks on graphs**. arXiv preprint arXiv:1312.6203, 2013.

[burbano2016] BURBANO, A.; VASILIU, M. ; BOUAZIZ, S.. **3d cameras benchmark for human tracking in hybrid distributed smart camera networks**. In: PROCEEDINGS OF THE 10TH INTERNATIONAL

CONFERENCE ON DISTRIBUTED SMART CAMERA, p. 76–83. ACM, 2016.

[cao2017] CAO, Z.; SIMON, T.; WEI, S.-E. ; SHEIKH, Y.. **Realtime multi-person 2d pose estimation using part affinity fields**. In: CVPR, 2017.

[cao2018] CAO, Z.; HIDALGO, G.; SIMON, T.; WEI, S.-E. ; SHEIKH, Y.. **OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields**. In: ARXIV PREPRINT ARXIV:1812.08008, 2018.

[cao2019] CAO, Y.; XU, J.; LIN, S.; WEI, F. ; HU, H.. **Gcnet: Non-local networks meet squeeze-excitation networks and beyond**. In: PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION WORKSHOPS, p. 0–0, 2019.

[chen2017dual]

[chui2017] CHUI, C. K.; CHEN, G. ; OTHERS. **Kalman filtering with Real-Time Applications**. Springer, 2017.

[cichocki2009] CICHOCKI, A.; ZDUNEK, R.; PHAN, A. H. ; AMARI, S.-I.. **Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation**. John Wiley & Sons, 2009.

[dabral2018learning] DABRAL, R.; MUNDHADA, A.; KUSUPATI, U.; AFAQUE, S.; SHARMA, A. ; JAIN, A.. **Learning 3d human pose from structure and motion**. In: PROCEEDINGS OF THE EUROPEAN CONFERENCE ON COMPUTER VISION (ECCV), p. 668–683, 2018.

[de2000] DE LATHAUWER, L.; DE MOOR, B. ; VANDEWALLE, J.. **A multilinear singular value decomposition**. SIAM journal on Matrix Analysis and Applications, 21(4):1253–1278, 2000.

[defferrard2016] DEFFERRARD, M.; BRESSON, X. ; VANDERGHEYNST, P.. **Convolutional neural networks on graphs with fast localized spectral filtering**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, p. 3844–3852, 2016.

[deng2009] DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K. ; FEI-FEI, L.. **Imagenet: A large-scale hierarchical image database**. In: 2009 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 248–255. Ieee, 2009.

[fang2017] FANG, H.-S.; XIE, S.; TAI, Y.-W. ; LU, C.. **Rmpe: Regional multi-person pose estimation**. In: PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, p. 2334–2343, 2017.

[ge2018] GE, L.. **Real-time 3D hand pose estimation from depth images**. PhD thesis, 2018.

[gupta2017] GUPTA, T.; LI, H.. **Indoor mapping for smart cities—an affordable approach: Using kinect sensor and zed stereo camera**. In: 2017 INTERNATIONAL CONFERENCE ON INDOOR POSITIONING AND INDOOR NAVIGATION (IPIN), p. 1–8. IEEE, 2017.

[he2016deep] HE, K.; ZHANG, X.; REN, S. ; SUN, J.. **Deep residual learning for image recognition**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 770–778, 2016.

[he2016identity] HE, K.; ZHANG, X.; REN, S. ; SUN, J.. **Identity mappings in deep residual networks**. In: EUROPEAN CONFERENCE ON COMPUTER VISION, p. 630–645. Springer, 2016.

[howard2017] HOWARD, A. G.; ZHU, M.; CHEN, B.; KALENICHENKO, D.; WANG, W.; WEYAND, T.; ANDREETTO, M. ; ADAM, H.. **Mobilenets: Efficient convolutional neural networks for mobile vision applications**. arXiv preprint arXiv:1704.04861, 2017.

[hu2018] HU, J.; SHEN, L. ; SUN, G.. **Squeeze-and-excitation networks**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 7132–7141, 2018.

[hu2018] HU, J.; SHEN, L.; ALBANIE, S.; SUN, G. ; VEDALDI, A.. **Gather-excite: Exploiting feature context in convolutional neural networks**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, p. 9401–9411, 2018.

[huang2017densely] HUANG, G.; LIU, Z.; VAN DER MAATEN, L. ; WEINBERGER, K. Q.. **Densely connected convolutional networks**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 4700–4708, 2017.

[huang2020] HUANG, Q.; ZHOU, F.; HE, J.; ZHAO, Y. ; QIN, R.. **Spatial–temporal graph attention networks for skeleton-based action recognition**. Journal of Electronic Imaging, 29(5):053003, 2020.

[ioannou2017deep] IOANNOU, Y.; ROBERTSON, D.; CIPOLLA, R. ; CRIMINISI, A.. **Deep roots: Improving cnn efficiency with hierarchical filter groups**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 1231–1240, 2017.

[ioffe2015batch] IOFFE, S.; SZEGEDY, C.. **Batch normalization: Accelerating deep network training by reducing internal covariate shift**. arXiv preprint arXiv:1502.03167, 2015.

[ionescu2013] IONESCU, C.; PAPAVA, D.; OLARU, V. ; SMINCHISESCU, C.. **Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments**. IEEE transactions on pattern analysis and machine intelligence, 36(7):1325–1339, 2013.

[jin2014] JIN, J.; DUNDAR, A. ; CULURCIELLO, E.. **Flattened convolutional neural networks for feedforward acceleration**. arXiv preprint arXiv:1412.5474, 2014.

[jonker1987] JONKER, R.; VOLGENANT, A.. **A shortest augmenting path algorithm for dense and sparse linear assignment problems**. Computing, 38(4):325–340, 1987.

[junior2006variational] JUNIOR, J. S.. **Variational Texture Atlas Construction and Applications**. PhD thesis, IMPA, 2006.

[kanazawa2018] KANAZAWA, A.; BLACK, M. J.; JACOBS, D. W. ; MALIK, J.. **End-to-end recovery of human shape and pose**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 7122–7131, 2018.

[kim2009] KIM, J.-S.; HWANGBO, M. ; KANADE, T.. **Realtime affine-photometric klt feature tracker on gpu in cuda framework**. In: 2009 IEEE 12TH INTERNATIONAL CONFERENCE ON COMPUTER VISION WORKSHOPS, ICCV WORKSHOPS, p. 886–893. IEEE, 2009.

[kim2015] KIM, Y.-D.; PARK, E.; YOO, S.; CHOI, T.; YANG, L. ; SHIN, D.. **Compression of deep convolutional neural networks for fast and low power mobile applications**. arXiv preprint arXiv:1511.06530, 2015.

[kipf2016] KIPF, T. N.; WELLING, M.. **Semi-supervised classification with graph convolutional networks**. arXiv preprint arXiv:1609.02907, 2016.

[kolda2009] KOLDA, T. G.; BADER, B. W.. **Tensor decompositions and applications**. SIAM review, 51(3):455–500, 2009.

[kossaifi2019] KOSSAIFI, J.; BULAT, A.; TZIMIROPOULOS, G. ; PANTIC, M.. **T-net: Parametrizing fully convolutional nets with a single high-order tensor**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 7822–7831, 2019.

[krizhevsky2012imagenet] KRIZHEVSKY, A.; SUTSKEVER, I. ; HINTON, G. E.. **Imagenet classification with deep convolutional neural networks**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, p. 1097–1105, 2012.

[lin2014] LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P. ; ZITNICK, C. L.. **Microsoft coco: Common objects in context**. In: EUROPEAN CONFERENCE ON COMPUTER VISION, p. 740–755. Springer, 2014.

[lin2017] LIN, M.; LIN, L.; LIANG, X.; WANG, K. ; CHENG, H.. **Recurrent 3d pose sequence machines**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 810–819, 2017.

[liu2018image] LIU, G.; REDA, F. A.; SHIH, K. J.; WANG, T.-C.; TAO, A. ; CATANZARO, B.. **Image inpainting for irregular holes using partial convolutions**. In: PROCEEDINGS OF THE EUROPEAN CONFERENCE ON COMPUTER VISION (ECCV), p. 85–100, 2018.

[long2015fully] LONG, J.; SHELHAMER, E. ; DARRELL, T.. **Fully convolutional networks for semantic segmentation**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 3431–3440, 2015.

[loper2015] LOPER, M.; MAHMOOD, N.; ROMERO, J.; PONS-MOLL, G. ; BLACK, M. J.. **Smpl: A skinned multi-person linear model**. ACM transactions on graphics (TOG), 34(6):248, 2015.

[luo2018] LUO, Y.; REN, J.; WANG, Z.; SUN, W.; PAN, J.; LIU, J.; PANG, J. ; LIN, L.. **Lstm pose machines**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 5207–5215, 2018.

[martinez2017] MARTINEZ, J.; HOSSAIN, R.; ROMERO, J. ; LITTLE, J. J.. **A simple yet effective baseline for 3d human pose estimation**. In: PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, p. 2640–2649, 2017.

[masson2017] MASSON, T.; PERLIN, K. ; OTHERS. **Holo-doodle: an adaptation and expansion of collaborative holojam virtual reality**. In: ACM SIGGRAPH 2017 VR VILLAGE, p. 9. ACM, 2017.

[mehta2017] MEHTA, D.; SRIDHAR, S.; SOTNYCHENKO, O.; RHODIN, H.; SHAFIEI, M.; SEIDEL, H.-P.; XU, W.; CASAS, D. ; THEOBALT, C.. **Vnect: Real-time 3d human pose estimation with a single rgb camera**. ACM Transactions on Graphics (TOG), 36(4):44, 2017.

[mehta2017monocular] MEHTA, D.; RHODIN, H.; CASAS, D.; FUA, P.; SOTNYCHENKO, O.; XU, W. ; THEOBALT, C.. **Monocular 3d human pose estimation in the wild using improved cnn supervision**. In: 2017 INTERNATIONAL CONFERENCE ON 3D VISION (3DV), p. 506–516. IEEE, 2017.

[mehta2018single] MEHTA, D.; SOTNYCHENKO, O.; MUELLER, F.; XU, W.; SRIDHAR, S.; PONS-MOLL, G. ; THEOBALT, C.. **Single-shot multi-person 3d pose estimation from monocular rgb**. In: 2018 INTERNATIONAL CONFERENCE ON 3D VISION (3DV), p. 120–130. IEEE, 2018.

[mehta2020] MEHTA, D.; SOTNYCHENKO, O.; MUELLER, F.; XU, W.; ELGHARIB, M.; FUA, P.; SEIDEL, H.-P.; RHODIN, H.; PONS-MOLL, G. ; THEOBALT, C.. **Xnect: Real-time multi-person 3d motion capture with a single rgb camera**. ACM Transactions on Graphics (TOG), 39(4):82–1, 2020.

[pandey2019volumetric] PANDEY, R.; TKACH, A.; YANG, S.; PIDLYPENSKYI, P.; TAYLOR, J.; MARTIN-BRUALLA, R.; TAGLIASACCHI, A.; PAPANDREOU, G.; DAVIDSON, P.; KESKIN, C. ; OTHERS. **Volumetric capture of humans with a single rgbd camera via semi-parametric learning**. arXiv preprint arXiv:1905.12162, 2019.

[pavllo2019] PAVLLO, D.; FEICHTENHOFER, C.; GRANGIER, D. ; AULI, M.. **3d human pose estimation in video with temporal convolutions and semi-supervised training**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 7753–7762, 2019.

[rabanser2017] RABANSER, S.; SHCHUR, O. ; GÜNNEMANN, S.. **Introduction to tensor decompositions and their applications in machine learning**. arXiv preprint arXiv:1711.10781, 2017.

[ramachandran2019] RAMACHANDRAN, P.; PARMAR, N.; VASWANI, A.; BELLO, I.; LEVSKAYA, A. ; SHLENS, J.. **Stand-alone self-attention in vision models**. arXiv preprint arXiv:1906.05909, 2019.

[ramakrishna2014] RAMAKRISHNA, V.; MUNOZ, D.; HEBERT, M.; BAGNELL, J. A. ; SHEIKH, Y.. **Pose machines: Articulated pose estimation via inference machines**. In: EUROPEAN CONFERENCE ON COMPUTER VISION, p. 33–47. Springer, 2014.

[ranjan2019] RANJAN, R.; PATEL, V. M. ; CHELLAPPA, R.. **Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(1):121–135, 2019.

[rayat2018] RAYAT IMTIAZ HOSSAIN, M.; LITTLE, J. J.. **Exploiting temporal information for 3d human pose estimation**. In: PROCEEDINGS OF THE EUROPEAN CONFERENCE ON COMPUTER VISION (ECCV), p. 68–84, 2018.

[ren2015faster] REN, S.; HE, K.; GIRSHICK, R. ; SUN, J.. **Faster r-cnn: Towards real-time object detection with region proposal networks**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, p. 91–99, 2015.

[ruggero2017] RUGGERO RONCHI, M.; PERONA, P.. **Benchmarking and error diagnosis in multi-instance pose estimation**. In: PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, p. 369–378, 2017.

[saito2019pifu] SAITO, S.; HUANG, Z.; NATSUME, R.; MORISHIMA, S.; KANAZAWA, A. ; LI, H.. **Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization**. arXiv preprint arXiv:1905.05172, 2019.

[sandler2018] SANDLER, M.; HOWARD, A.; ZHU, M.; ZHMOGINOV, A. ; CHEN, L.-C.. **Mobilenetv2: Inverted residuals and linear bottlenecks**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 4510–4520, 2018.

[santurkar2018does] SANTURKAR, S.; TSIPRAS, D.; ILYAS, A. ; MADRY, A.. **How does batch normalization help optimization?(no, it is not about internal covariate shift)**. arXiv preprint arXiv:1805.11604, 2018.

[schwarcz2018] SCHWARCZ, S.; POLLARD, T.. **3d human pose estimation from deep multi-view 2d pose.** In: 2018 24TH INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION (ICPR), p. 2326–2331. IEEE, 2018.

[simonyan2014very] SIMONYAN, K.; ZISSERMAN, A.. **Very deep convolutional networks for large-scale image recognition.** arXiv preprint arXiv:1409.1556, 2014.

[sindagi2018] SINDAGI, V. A.; PATEL, V. M.. **A survey of recent advances in cnn-based single image crowd counting and density estimation.** Pattern Recognition Letters, 107:3–16, 2018.

[smith2017] SMITH, S.; KARYPIS, G.. **Accelerating the tucker decomposition with compressed sparse tensors.** In: EUROPEAN CONFERENCE ON PARALLEL PROCESSING, p. 653–668. Springer, 2017.

[song2017] SONG, J.; WANG, L.; VAN GOOL, L. ; HILLIGES, O.. **Thin-slicing network: A deep structured model for pose estimation in videos.** In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 4220–4229, 2017.

[srivastava2015training] SRIVASTAVA, R. K.; GREFF, K. ; SCHMIDHUBER, J.. **Training very deep networks.** In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, p. 2377–2385, 2015.

[su2019] SU, K.; YU, D.; XU, Z.; GENG, X. ; WANG, C.. **Multi-person pose estimation with enhanced channel-wise and spatial information.** In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 5674–5682, 2019.

[symeonidis2010] SYMEONIDIS, P.; NANOPOULOS, A. ; MANOLOPOULOS, Y.. **A unified framework for providing recommendations in social tagging systems based on ternary semantic analysis.** IEEE Transactions on Knowledge and Data Engineering, 22(2):179–192, 2010.

[szegedy2015going] SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V. ; RABINOVICH, A.. **Going deeper with convolutions.** In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 1–9, 2015.

[szegedy2016rethinking] SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J. ; WOJNA, Z.. **Rethinking the inception architecture for computer vision**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 2818–2826, 2016.

[szegedy2017inception] SZEGEDY, C.; IOFFE, S.; VANHOUCKE, V. ; ALEMI, A. A.. **Inception-v4, inception-resnet and the impact of residual connections on learning**. In: THIRTY-FIRST AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2017.

[toshev2014deeppose] TOSHEV, A.; SZEGEDY, C.. **Deeppose: Human pose estimation via deep neural networks**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 1653–1660, 2014.

[tsai2012] TSAI, D.; FLAGG, M.; NAKAZAWA, A. ; REHG, J. M.. **Motion coherent tracking using multi-label mrf optimization**. International journal of computer vision, 100(2):190–202, 2012.

[tucker1966] TUCKER, L. R.. **Some mathematical notes on three-mode factor analysis**. Psychometrika, 31(3):279–311, 1966.

[tung2017] TUNG, H.-Y.; TUNG, H.-W.; YUMER, E. ; FRAGKIADAKI, K.. **Self-supervised learning of motion capture**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, p. 5236–5246, 2017.

[varol2017learning] VAROL, G.; ROMERO, J.; MARTIN, X.; MAHMOOD, N.; BLACK, M. J.; LAPTEV, I. ; SCHMID, C.. **Learning from synthetic humans**. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 109–117, 2017.

[vaswani2017] VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł. ; POLOSUKHIN, I.. **Attention is all you need**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, p. 5998–6008, 2017.

[velivckovic2017] VELIČKOVIĆ, P.; CUCURULL, G.; CASANOVA, A.; ROMERO, A.; LIO, P. ; BENGIO, Y.. **Graph attention networks**. arXiv preprint arXiv:1710.10903, 2017.

[wang2017] WANG, M.; LIU, B. ; FOROOSH, H.. **Factorized convolutional neural networks**. In: PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, p. 545–553, 2017.

[wang2019] WANG, K.; LIN, L.; JIANG, C.; QIAN, C. ; WEI, P.. **3d human pose machines with self-supervised learning.** IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(5):1069–1082, 2019.

[wei2016] WEI, S.-E.; RAMAKRISHNA, V.; KANADE, T. ; SHEIKH, Y.. **Convolutional pose machines.** In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 4724–4732, 2016.

[woo2018] WOO, S.; PARK, J.; LEE, J.-Y. ; SO KWEON, I.. **Cbam: Convolutional block attention module.** In: PROCEEDINGS OF THE EUROPEAN CONFERENCE ON COMPUTER VISION (ECCV), p. 3–19, 2018.

[wu2020] WU, Z.; PAN, S.; CHEN, F.; LONG, G.; ZHANG, C. ; PHILIP, S. Y.. **A comprehensive survey on graph neural networks.** IEEE Transactions on Neural Networks and Learning Systems, 2020.

[xie2017aggregated] XIE, S.; GIRSHICK, R.; DOLLÁR, P.; TU, Z. ; HE, K.. **Aggregated residual transformations for deep neural networks.** In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 1492–1500, 2017.

[xu2018] XU, K.; HU, W.; LESKOVEC, J. ; JEGELKA, S.. **How powerful are graph neural networks?** arXiv preprint arXiv:1810.00826, 2018.

[yang2018] YANG, W.; OUYANG, W.; WANG, X.; REN, J.; LI, H. ; WANG, X.. **3d human pose estimation in the wild by adversarial learning.** In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 5255–5264, 2018.

[zhao2019] ZHAO, L.; PENG, X.; TIAN, Y.; KAPADIA, M. ; METAXAS, D. N.. **Semantic graph convolutional networks for 3d human pose regression.** In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, p. 3425–3435, 2019.

[zhou2016] ZHOU, S.; VINH, N. X.; BAILEY, J.; JIA, Y. ; DAVIDSON, I.. **Accelerating online cp decompositions for higher order tensors.** In: PROCEEDINGS OF THE 22ND ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, p. 1375–1384. ACM, 2016.

[zhou2018] ZHOU, J.; CUI, G.; ZHANG, Z.; YANG, C.; LIU, Z.; WANG, L.; LI, C. ; SUN, M.. **Graph neural networks: A review of methods and applications.** arXiv preprint arXiv:1812.08434, 2018.

[zhou2018] ZHOU, X.; ZHU, M.; PAVLAKOS, G.; LEONARDOS, S.; DERPANIS, K. G. ; DANIILIDIS, K.. **Monocap: Monocular human motion capture using a cnn coupled with a geometric prior**. IEEE transactions on pattern analysis and machine intelligence, 41(4):901–914, 2018.