

Imperial College London
Department of Electrical and Electronic Engineering

Human Pose Estimation from Sparse 3D Data on Low Power Systems

Manolis Vasileiadis

Supervised by Christos-Savvas Bouganis

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Electrical and Electronic Engineering of Imperial College and
the Diploma of Imperial College, August 2020

Abstract

This thesis deals with the investigation of novel techniques for human pose estimation (HPE) using sparse depth/3D data, in order to develop a standalone, high-accuracy, low-latency human pose estimation module, suitable for deployment in systems with limited processing resources.

Based on the existing work and motivated by the significant progress that has been achieved in the relevant fields, two novel methods for the estimation and tracking of the human pose utilising sparse depth/3D data, are proposed.

First, a real-time human pose estimation and tracking framework is developed, which builds upon an already established human-template-tracking based approach, utilising the 3D Signed Distance Function (SDF) data representation. A series of complementary tracking features are introduced, tackling specifically the issues of free space violation, body part visibility and leg intersection, which are typically encountered under real-life monitoring conditions. The method is experimentally evaluated on a series of publicly available datasets, achieving state-of-the-art (SOA) performance, while also successfully utilised for human behavioural modelling on an autonomous robotic platform.

Due to inherent limitations of this tracking-based approach, such as the requirement for clearly segmented human/background data and the use of an out-of-the-box initialiser, a second, deep learning-based architecture is investigated. Specifically, a detection-based 3D-CNN architecture for 3D human pose estimation from 3D data is introduced, following the sequential network architecture paradigm. It utilises a volumetric data representation, and generates 3D heatmaps corresponding to potential locations of the human joints in the scene, achieving state-of-the-art accuracy. Additionally, a 3D body-part detector is incorporated, extending the architecture

towards multi-person 3D pose estimation, the first such method for 3D data.

However, the 3D CNN architecture comes at a steep computational cost, making it unsuitable for implementation on low power systems. Thus, the final contribution of this thesis includes the investigation of computationally efficient 3D CNN design guidelines, in order to reduce the computational complexity of the developed model. The result of this investigation is a novel 3D-CNN architecture for multi-person pose estimation from 3D data, composed mainly of 3D depthwise residual bottleneck units, SE blocks and a decomposed strided input layer. This optimised version performs comparably to SOA methods on two public datasets, while requiring significantly fewer computational resources and achieving a speedup of over 100x on a modern low power mobile device, and a reduction in model size of approximately 50x.

Acknowledgements¹

First and foremost, I would like to thank my supervisor, Dr. Christos Bouganis, for his priceless support and care throughout my studies. The present work would not have been possible without his invaluable academic guidance and advice. Under his supervision I was able to develop important skills, recognise and work on my strengths and weaknesses and cope with the often overwhelming demands, all of which helped me in becoming a better researcher.

I would also like to thank my collaborative supervisor at the Information Technologies Institute (ITI), Dr. Dimitrios Tzovaras, for his supervision and support throughout the whole period of my research.

Finally, I would like to thank my family, my friends and my colleagues in ITI, for always having been on my side throughout all the difficulties of the past four years.

¹A major part of this thesis was supported by the European Union's Horizon 2020 Research and Innovation Programme through the RAMCIP project under Grant Agreement No. 643433, and the SMILE project under Grant Agreement No. 740931. The opinions expressed in this thesis are those of the author and do not necessarily reflect the views of the European Commission.

Declarations

Declaration of Originality

I herewith certify that the work presented in this thesis is my original own. All material in this thesis which is not my own work has been appropriately referenced.

Declaration of Copyright

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC).

Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

Abbreviations

ADE	Average Distance Error
AR	Augmented Reality
BFGS	Broyden–Fletcher–Goldfarb–Shanno
CNN	Convolutional Neural Network
DBN	Dynamic Bayesian Network
DL	Deep Learning
DWS	DepthWise Separable
EM	Expectation Maximisation
FLOP	Floating Point Operation
FOV	Field of View
FPGA	Field Programmable Gate Array
GMM	Gaussian Mixture Model
HCI	Human Computer Interaction
HOG	Histogram of Oriented Gradients
HPE	Human Pose Estimation
ICP	Iterative Closest Point
MAP	Mean Average Precision

MEM	Memory Access Operation
NIR	Near InfraRed
PAF	Part Affinity Field
PC	Point Cloud
PCA	Principal Component Analysis
RF	Random Forest
RGB-D	Red Green Blue - Depth
RoI	Region of Interest
SDF	Signed Distance Function
SE	Squeeze and Excitation
SGD	Stochastic Gradient Descent
SoA	State of the Art
SVM	Support Vector Machine
ToF	Time of Flight

Contents

Abstract	i
Acknowledgements	iii
1 Introduction	1
1.1 Motivation	3
1.2 Problem Formulation	4
1.3 Thesis Contributions	7
1.4 List of Publications	9
1.5 Thesis Outline	10
2 Literature Survey	12
2.1 Human Pose Estimation Algorithms	13
2.1.1 Algorithm Taxonomy	13
2.1.2 Pose Accuracy Measures	15
2.1.3 2D Human Pose Estimation from RGB Data using Handcrafted Features	17
2.1.4 3D Human Pose Estimation from Depth/3D Data using Handcrafted Features	19

2.1.5	2D Human Pose Estimation from RGB Data via Deep Learning	24
2.1.6	3D Human Pose Estimation from RGB Data via Deep Learning	30
2.1.7	3D Human Pose Estimation from Depth/3D Data via Deep Learning	32
2.2	Optimisation for Systems with Limited Computational Resources	35
2.2.1	Hardware-Specific Machine Learning Acceleration	36
2.2.2	Hardware-Agnostic Deep Learning Acceleration	41
2.3	Summary	45
3	Preliminary Concepts and Notations	47
3.1	Application Scenarios	47
3.2	Depth Imaging	49
3.3	Datasets	52
3.4	Computational Complexity Metrics	56
3.4.1	Framework-Agnostic Metrics	57
3.4.2	Metrics Calculation in CNNs	59
3.5	Summary	60
4	Template-Based Human Pose Estimation and Tracking	61
4.1	Data Input and Output	63
4.2	Articulated Human Template	63
4.3	SDF-Based Tracker	64
4.4	Model Representation	65

4.5	Data Association	66
4.6	Optimisation	67
4.7	Complementary Tracking Features	68
4.7.1	Free Space Violation	69
4.7.2	Body Part Visibility	69
4.7.3	Leg Intersection	70
4.8	Framework Initialisation	71
4.9	Tracking Failure Detection	72
4.10	Large Object Handling	73
4.11	Implementation Details	74
4.11.1	Input Data Preprocessing	75
4.11.2	Articulated-SDF Human Model	76
4.11.3	Optimisation Process	77
4.11.4	Tracker Initialisation	79
4.11.5	Data Export	79
4.12	Experimental Evaluation	80
4.12.1	Individual Components Evaluation	81
4.12.2	Public Datasets	82
4.12.3	Realistic Human Motion Dataset	83
4.13	Implementation on Autonomous Robotic Platform	85
4.14	Summary	86

5	Deep Learning-Based Human Pose Estimation	89
5.1	Preliminary Testing on the 2D Depth Image Plane	91
5.1.1	The OpenPose Architecture	91
5.1.2	Experimental Evaluation	92
5.2	3D-CNNs for 3D Human Pose Estimation from 3D Data	96
5.3	Input Data Representation	97
5.4	Network Architecture	98
5.4.1	Feature Extraction	100
5.4.2	Joints Detection	100
5.4.3	Extension Towards Multi-Person Pose Estimation	101
5.5	Multi-Person Pose Composition	102
5.6	Network Training	105
5.6.1	Ground Truth Generation	105
5.6.2	Training Loss Function	106
5.6.3	Optimisation	106
5.6.4	Data augmentation	107
5.7	Implementation Details	107
5.7.1	Input Voxel Grid	107
5.7.2	Network Implementation	110
5.7.3	Training	111
5.8	Experimental Evaluation	111

5.8.1	Single person 3D pose estimation	111
5.8.2	Multi-Person 3D Pose Estimation	116
5.8.3	Inference	118
5.9	Discussion	119
5.9.1	Comparison to the Template-Based Tracker	121
5.10	Summary	125
6	3D CNN Architecture Optimisation for HPE on Low Power Devices	128
6.1	Network Design Optimisation	130
6.1.1	Optimisation Protocol	132
6.1.2	Efficient Convolutional Building Blocks	132
6.1.3	Input Layer	134
6.1.4	Kernel Size	136
6.1.5	End-of-Stage Pointwise Layers	136
6.1.6	Squeeze-and-Excitation Blocks	137
6.1.7	Input Data Representation	138
6.1.8	Design Guidelines Overview	140
6.2	Network Dimensionality Parameterisation	141
6.3	Experimental Evaluation	143
6.3.1	3D Human Pose Estimation	143
6.3.2	3D Object Classification	145
6.3.3	Computational Complexity Evaluation	146

6.3.4	On-Device Benchmarking	147
6.4	Summary	152
7	Conclusions and Future Work	154
7.1	Thesis Summary and Achievements	154
7.2	Research Conclusions	158
7.3	Future Work	160
	Bibliography	162
A	Experimental Evaluation of the Template Based Pose Tracking Framework	197
B	Experimental Evaluation of Deep Learning Based Pose Estimation	201
C	Experimental Evaluation of the Optimised 3D CNN Architecture	205

List of Tables

2.1	Experimental evaluation of early RGB-based discriminative HPE approaches using handcrafted features.	19
2.2	Experimental evaluation of early non-DL Depth/3D-based approaches, as reported by their authors.	23
2.3	Experimental evaluation of DL-based 2D single human pose estimation methods from RGB data, as reported in the respective papers.	26
2.4	Experimental results of the DL-based 2D multi-person pose estimation methods from RGB data, as reported in the respective papers.	28
2.5	Experimental results of the DL-based 2D human pose estimation methods from RGB videos, as reported in the respective papers.	30
2.6	Experimental results of the DL-based 3D human pose estimation methods from RGB data, as reported in the respective papers.	32
2.7	Experimental results of the DL-based human pose estimation methods from depth/3D data, as reported in the respective papers.	34
3.1	Technical specifications of the most common commercially available depth cameras	52
3.2	FLOPs per operation [Huss and Pennline, 1987, Chimmalgi et al., 2019]	58

3.3	Calculation of the computational complexity metrics for common CNN layers. D is the input's spatial dimension per axis, n is the number of fully connected nodes, C_{in}, C_{out} are the input and output channels, k is the kernel size, g is the number of convolution groups, s is the stride and d is the input dimensionality (1 for 1D, 2 for 2D, 3 for 3D). For simplicity we consider that the input's spatial dimensions, kernel size and stride are equal across all axes.	60
4.1	Experimental evaluation of the body part visibility component, on artificially occluded sequences of the PDT dataset. Enabling the body part visibility component significantly decreases the ADE for the occluded lower body parts, increasing the probability of the tracker correctly tracking them once they reappear	81
5.1	Comparison ($mAP@0.1m$) of the OpenPose [Cao et al., 2017] architecture to state-of-the-art-methods on the ITOP (front) and EVAL datasets	96
5.2	Experimental evaluation ($mAP@0.1m$) of the single-person 3D pose estimation architecture on the ITOP - front, ITOP - top, EVAL and PDT datasets	112
5.3	Experimental evaluation ($mAP@0.1m$) of the multi-person 3D pose estimation architecture on the <i>Haggling</i> and <i>Band</i> sequences of the CMU PanopticStudio 3D PointCloud dataset	117
5.4	Runtime analysis of all the individual components of the proposed methodology, benchmarked on the NVIDIA GTX 970 (CUDA 7) and 1080Ti (CUDA 10) GPUs.	119
5.5	Accuracy comparison of the Template-based tracker and the 3D-CNN estimator on the EVAL and PDT datasets.	122
5.6	Computational complexity estimation of the template-based human pose tracker, for 3000 3D input points, optimisation iterations $it_{max} = 10$ and linesearch steps $ls = 20$	123

5.7	Computational complexity comparison between the template-based human pose tracker and 3D CNN estimator.	123
6.1	Computational complexity breakdown of the baseline 3D CNN estimator	131
6.2	mAP (0.1m) on the ITOP-front dataset and overall computational complexity for different efficient convolutional building blocks.	134
6.3	mAP (0.1m) on the ITOP-front dataset and overall computational complexity for different input layer layouts. s is the stride size.	135
6.4	mAP (0.1m) on the ITOP-front dataset and overall computational complexity for different kernel size	136
6.5	mAP (0.1m) on the ITOP-front dataset and overall computational complexity with and without the second-to-last pointwise layer at each detection stage. . . .	137
6.6	mAP (0.1m) on the ITOP-front dataset and overall computational complexity with and without the utilisation of Squeeze - and - Excitation blocks (SE) [Hu et al., 2018]. r is the SE block squeeze rate.	138
6.7	mAP (0.1m) on the ITOP-front dataset and overall computational complexity for different data representation models. K is the fixed number of points per voxel, l is the truncation limit and D is the voxel grid spatial resolution	140
6.8	Comparison (mAP@0.1m) of the proposed optimised architecture to the baseline 3D CNN and state-of-the-art on the ITOP and CMU PanopticStudio datasets .	144
6.9	Accuracy comparison of the baseline and optimised VGG13 3D architectures, to state-of-the-art volumetric 3D-CNN models on the ModelNet10 dataset.	146
6.10	Computational complexity comparison of the proposed optimised architecture, the baseline 3D CNN estimator and the state-of-the-art V2V Posenet by Moon et al. [Moon et al., 2018]	146
6.11	Technical specifications of the benchmarking configuration	148

6.12	Runtime in seconds, for a single frame inference, benchmarked on a desktop configuration. x1 and x8 correspond to single and multi core performance	149
6.13	Runtime in seconds, for a single frame inference, benchmarked on two mobile devices (Xiaomi Redmi Note 5 Pro and Samsung Galaxy Note 8). x1 and x8 correspond to single and multi core performance	149
A.1	Experimental evaluation of the proposed template-based pose tracking algorithm on three publicly available datasets.	198
A.2	Experimental evaluation of the proposed template-based pose tracking algorithm on the novel realistic human motion dataset and comparison to the Kinect 1 built-in pose estimator (NiTE 2.2). To ensure consistency between the two systems, the distance between the ground truth positions and the skeleton joints was used, instead of the corresponding mesh point, which may have resulted in a slightly higher Average Distance Error	199
A.3	Experimental evaluation of the leg intersection fix. Tested on a subset of the PDT dataset, for sequences with complex leg movement	199
A.4	Experimental evaluation of the body-part visibility feature. Tested on a subset of the PDT dataset, where lower body motion was of moderate complexity. The lower body parts were artificially occluded, in order to simulate a “acting behind a table” scenario	200
B.1	Experimental evaluation of the 2D-CNN based OpenPose architecture on two publicly available datasets.	202
B.2	Experimental evaluation of the proposed 3D-CNN based pose estimation algorithm on three publicly available single person datasets.	203
B.3	Experimental evaluation of the proposed 3D-CNN based pose estimation algorithm on the CMU PanopticStudio 3D PointCloud multi person dataset. . . .	204

C.1	Experimental evaluation of the proposed optimised 3D-CNN pose estimation architecture on the single person ITOP and multi-person CMU Panoptic PtCloud datasets	206
-----	--	-----

List of Figures

1.1	3D human pose estimation: Given as input a sparse 3D point cloud, the objective is to extract the coordinates in 3D space of the skeletal joints, for all the people present in the scene.	5
2.1	Block diagram of the taxonomy of state of the art marker-less pose estimation and tracking algorithms	13
2.2	Block diagram of the taxonomy of hardware-agnostic techniques for Deep Learning acceleration	42
2.3	Block diagram of the ResNet [He et al., 2016] and ResNeXt [Xie et al., 2017] bottleneck units (image taken from [Xie et al., 2017])	43
3.1	Left: Sample depth image from the Micorosoft Kinect 1 camera (normalized for better visualization). Right: The corresponding 3D point cloud produced by back-projecting the depth image	50
3.2	Sample frames from the realistic human motion dataset, with manually annotated ground truth data.	56
4.1	Human template customisation. Left: $[g=1, s=1]$, Right $[g=1.2, s=1.5]$	64

4.2	Representation of the human model in the tracker. Left: a planar slice through the human model with the isocontours of the SDF superimposed on the plane, Right: isocontours of the SDF, with the zero level set shown in bold (image taken from [Schmidt et al., 2015])	65
4.3	Left: global SDF, Right: composition of local SDFs (image taken from [Schmidt et al., 2015])	66
4.4	Pose correction through the complementary tracking features. Left: free space violation, Center: leg intersection, Right: body part visibility in occluded view; without the visibility feature both legs converge to the visible right leg, while with the feature enabled, the left occluded leg remains at the default position and is not rendered	68
4.5	The initialisation pipeline of the human pose tracker	72
4.6	High (left) and low (right) confidence pose tracking. Deformed human template in red, input human silhouette in grey	73
4.7	Data preprocessing for large object removal. The input image is split in three regions: blue - overlap with the last tracked human silhouette, red - buffer zone, green - candidate large object area which provides the seeds for the flood-fill algorithm. The image on the right is the processed image passed to the optimiser	74
4.8	Block diagram of the developed human pose tracking algorithm	75
4.9	Tracking performance in relationship to the input cloud subsampling, for sample sequences from the EVAL dataset. Left: average distance error (10 iterations, 20 linesearch steps), Right: optimization execution time for a single iteration (20 linesearch steps)	76
4.10	Tracking accuracy in relationship to the optimisation parameters, for sample sequences from the EVAL dataset. Left: maximum iterations (20 linesearch steps), Right: linesearch steps (10 maximum iterations)	78

4.11	The skeleton joints extracted by the developed human tracking implementation .	80
4.12	Framework performance on three public datasets. Top - comparison to state-of-the-art methods (results as reported by their authors), Bottom - <i>ADE</i> per-joint	83
4.13	Sample frames from the evaluation on public datasets.	83
4.14	Comparison of per-joint <i>ADE</i> and <i>mAP</i> with the Kinect built-in pose estimator (NITE 2.2 middleware [PrimeSense, 2013]) on the Realistic Human Motion Dataset.	84
4.15	Sample frames from the evaluation on the Realistic Human Motion Dataset: Red - proposed framework, Green - Kinect built-in pose estimator (NITE 2.2 middleware [PrimeSense, 2013])	84
4.16	The first and second iterations of the RAMCIP autonomous robotic platform .	85
4.17	Sample frames of the template-based pose tracker used for action recognition in realistic settings	85
5.1	Architecture of the two-branch multi-stage CNN. First, the image is analysed through a VGG-19 [Simonyan and Zisserman, 2015] based network structure. Next, the exported feature maps $\mathbf{F}$ are passed to the two branch network structure. At each stage t the top branch produces the joint confidence maps $\mathbf{S}_t$, while the lower branch produces the Part Affinity Fields $\mathbf{L}_t$. After each stage, the predictions from the two branches, along with the original image features, are concatenated for the next stage (image taken from [Cao et al., 2017])	92
5.2	Per-joint accuracy from the experimental evaluation of OpenPose on the ITOP depth dataset (front)	93
5.3	Per-joint accuracy from the experimental evaluation of OpenPose on the EVAL depth dataset	95

5.4	Sample frames from the experimental evaluation of OpenPose on the ITOP-front (top) and EVAL (bottom) depth datasets	95
5.5	Generating a hit grid from a 3D point cloud. The 3D space is quantised using a fixed-size occupancy grid. Voxels that include at least one data point within their volume are assigned a fixed non-zero value.	98
5.6	The proposed network architecture. Top: The feature extraction part (light blue), takes as input a $112 \times 112 \times 112$ 3D voxel grid (grey) and produces a set of features maps $\mathbf{F}$. Bottom: The sequential multi-stage structure produces the per-voxel likelihood maps $\mathbf{J}^t$, $\mathbf{B}^t$ for N_J joints (light red) and N_B body parts (light green) respectively. At the first stage $t = 1$ the likelihood maps are inferred only from the features $\mathbf{F}$, while in subsequent stages the predictions from the previous stage, along with the original features $\mathbf{F}$ are concatenated in order to refine the predictions	99
5.7	Left: Ground truth joint likelihood map for the shoulder, elbow and hand joints. Each joint is modelled by a 3D Gaussian, centred at the ground truth 3D coordinates of the joint. Right: Ground truth body part likelihood map for the upper and lower arm body parts. Each body part is modelled by a cylinder of a predefined radius, lying along the line segment generated from the corresponding ground truth joint positions	106
5.8	Top: Relative increase in accuracy when decreasing the voxel dimension of the input 3D voxel grid. Bottom: Relative increase in total memory usage, runtime and FLOPS when increasing the dimensions of the input 3D voxel grid	108
5.9	Top-down view of the input grid area (green) and the effective field of view of the Kinect 1 (light blue) and Kinect 2 (blue) depth cameras	109
5.10	Relative increase in accuracy, number of model parameters, runtime, total memory usage and FLOPS when increasing the number of stages in the sequential multi-stage structure	110

5.11	Qualitative results from experimental evaluation of the single person 3D pose estimation architecture on the ITOP - front (top), EVAL (middle) and PDT (bottom) datasets. Background data have been removed for better visualization	114
5.12	Comparison ($mAP@0.1m$) of the single person 3D pose estimation architecture to-state-of-the-art-methods on the ITOP - front, ITOP - top, EVAL and PDT datasets (results taken from the corresponding papers)	115
5.13	Qualitative results from experimental evaluation of the multi person 3D pose estimation architecture on the on the <i>Haggling</i> and <i>Band</i> sequences of the CMU PanopticStudio 3D PointCloud dataset. Background data have been removed for better visualisation	118
5.14	Left: Out-of-FOV clipping: the partial skeleton pose is estimated, even though the whole human is not visible and may not be detected. Right: Limbs in close proximity: the algorithm can handle multiple instances of the same joint (right hand) in close proximity, correctly assigning each detected joint to the corresponding skeleton	120
6.1	The basic convolutional building blocks investigated, based on the Mobilenets [Howard et al., 2017, Sandler et al., 2018], Shufflenet [Zhang et al., 2018, Ma et al., 2018] and Residual Bottleneck [He et al., 2016] architectures.	133
6.2	Integration of a Squeeze - and - Excitation block [Hu et al., 2018] inside a Shufflenet v2 basic block	138
6.3	2D illustrations of the <i>Hitgrid</i> [Maturana and Scherer, 2015], <i>PointGrid</i> [Le and Duan, 2018] and <i>projective D-TSDF</i> [Song and Xiao, 2016] 3D data representation models.	139

6.4	mAP (0.1m) on the ITOP-front dataset and computational complexity in giga-FLOPs, for different width multiplier values a and number of sequential stages t	142
6.5	The final optimised version of the 3D CNN human pose estimation architecture	143
6.6	The baseline VGG13 3D architecture. It takes as input a $32 \times 32 \times 32$ voxel grid, and predicts the label of the object depicted in the voxel grid	145
6.7	Average inference speed gain and model size reduction of the optimised architectures, relative to the baseline 3D CNN architecture, across all the benchmark devices	151
6.8	Maximum framerate of the optimised architectures on the GPU, CPU x86-64 and ARMv8-A benchmark platforms	151

Chapter 1

Introduction

Humans, being a highly social and vision-dominant species, have historically relied on gesture, expression and body language interpretation in order to perceive the moods, actions and intentions of other people [Blake and Shiffrar, 2007]. As Darwin noted in his seminal work, *The Expression of Emotions in Man and Animals* [Darwin, 1872], actions can provide more reliable and compelling information towards understanding what others are doing, compared to static images. While these perceptual skills are mastered and refined by humans from a very young age, achieving a similar level of perception in autonomous machines has proven to be non-trivial, especially for the low level task of estimating the human pose from a visual input, which serves as the basis of such skills.

From a technical perspective, human pose estimation (HPE) can be defined as the process of detecting the positions of the joints of the human body in RGB and depth images or 3D point-clouds, in order to reconstruct the skeletal structure and provide information about body posture, body motion and human gestures. It is considered one of the major challenges in the field of computer vision and has been intensively studied, in the last few decades, by the computer vision community [Moeslund et al., 2006, Liu et al., 2015, Sarafianos et al., 2016], due to its fundamental importance in various scientific fields.

Human pose estimation techniques have found usage in a large variety of technology domains, aiming towards creating automated real-life applications, which will require minimal human

intervention in order to function properly, as well as focusing towards providing a more natural, human-centric application interface. Some of the areas where human pose estimation techniques have been utilised include:

- **Surveillance:** One of the earliest applications of human detection and pose estimation can be found in the area of video surveillance, which aims at automatically monitoring humans and recognising aggressive and potentially dangerous behaviours [Sedai et al., 2009].
- **Human - computer interaction (HCI):** Efforts have been made to enhance traditional input methods, with more natural, life-like methods of interaction [Palacios et al., 2013, Song and Davis, 2015] . Body-motions and gestures similar to those used by people when communicating with each other, such as head nodding, arm waving or pointing, are employed as substitutes for various commonly used HCI actions (i.e. scrolling, mouse clicking etc.).
- **Entertainment:** The video-game industry has shifted its focus towards using the human body as a controller (i.e. Microsoft Kinect, Nintendo Wii), in order to increase the immersiveness of modern video-games, especially in the fast-developing area of Augmented and Virtual Reality games [Pugliese et al., 2015]. Moreover, human motion capture has been widely used in video-games and movies to achieve life-like animations, by rigging virtually designed models on the recorded moving skeleton of real actors.
- **Human Motion Analysis:** Video-based Human Motion Analysis has been of special interest in sports as it can help to recover playing techniques from legacy video content and analyse the evolution of athletes' performance through time [Fastovets et al., 2013, Unzueta et al., 2014].
- **Healthcare:** Human pose tracking is utilised for automatic human monitoring and activity detection, in order to extract behavioural and movement patterns which can contribute towards the early diagnosis of medical issues [Leo et al., 2017].
- **Robotics:** One of the basic prerequisites for mobile robotic platforms to achieve autonomous operation is to fully comprehend the surrounding environment, including any

humans present. Robust human pose estimation can contribute, among others, towards safety, human-robot communication and efficient decision making [Kostavelis et al., 2016, McColl et al., 2011], thus making it an integral part a robotic platform’s operation. Moreover, the use of robotic service applications in various domains, including the ones mentioned above, further increases the need for effective human pose estimation in mobile robotic platforms.

1.1 Motivation

Estimating the human pose is an intricate and complex task. The human body presents high variability in shape, size and texture, while the articulated joints that make up the human skeleton offer many degrees of freedom, providing a large range of motion for each rigid body part. Marker-based motion capture systems have been effectively used for body pose estimation and tracking, in controlled laboratory environments [Park and Hodgins, 2006]. However, the intricate installation process and obtrusive nature, as they require the user to wear a set of special reflective markers and a network of multiple calibrated cameras with overlapping FOVs to track them, as well as the high cost, have prevented the wide adoption of such systems in real-life applications. As a result, the focus of the research community has shifted towards marker-less human pose estimation techniques, using low-cost consumer-grade RGB and depth cameras [Liu et al., 2015]. While these techniques do offer technically and financially feasible solutions, they also have to deal with various problems which hinder their performance, such as body part occlusions due to the presence of obstacles, partial-body views due to constraints in the available FOV of the camera, sensor noise, interaction with objects etc., which further add to the problem’s complexity.

Moreover, most research efforts deal with human pose estimation as a single, discrete process, requiring substantial computing resources, usually high-end CPUs or GPUs for parallel processing, in order to achieve real time operation. This has led to the development of implementations which are successful in accurately tracking the human body as stand-alone applications, but

are not suitable for deployment in systems with limited processing power and hardware-based constraints in installation space and cooling resources (i.e. mobile phones, autonomous robotic platforms etc.). Meanwhile, low power devices have experienced a growing adoption in various applications during the last few years, shifting the research focus towards the implementation of HPE methods on such systems, thus making it necessary to reduce the computational complexity of the proposed approaches while maintaining a high level of accuracy.

1.2 Problem Formulation

The problem that this research is addressing is *human pose estimation from sparse 3D data, on low power systems*. Given as input a 3D point cloud, produced from a single or multiple depth cameras, the objective is to estimate the positions, in 3D space, of all the human skeleton joints that are inside the cameras' field of view, in order to extract the body poses of all the humans. The recognition process must be robust to a specific set of common environmental challenges, including self-occlusions, partial/cropped views, non-typical (i.e. top-down) viewpoints, cluttered scenes, sensor noise and moving cameras. Additionally the algorithm design should allow the seamless and efficient implementation of the developed human pose estimation algorithm to a variety of platforms, targeting mainly low-power mobile devices.

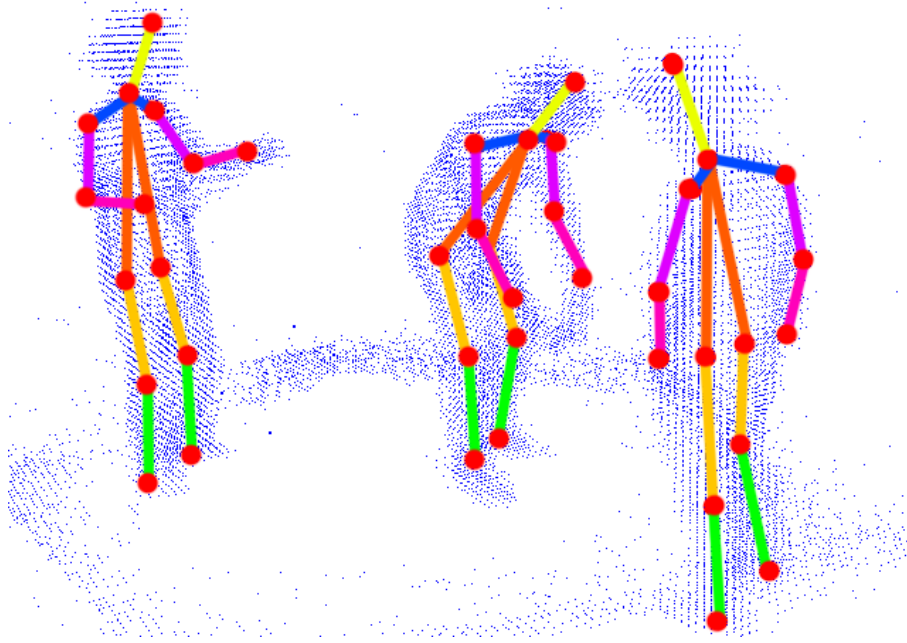


Figure 1.1: 3D human pose estimation: Given as input a sparse 3D point cloud, the objective is to extract the coordinates in 3D space of the skeletal joints, for all the people present in the scene.

The task of human pose estimation is limited, in this research, to sparse 3D data generated from depth images. This limitation has been partially imposed due to the more privacy invasive nature of RGB imaging, along with its susceptibility to luminosity-changing conditions, both important issues especially in the cases of non-consenting or in-home monitoring. Depth/3D imaging, on the other hand, overcomes these issues while maintaining a discriminative prowess comparable to RGB imaging. Depth maps, which are intrinsically a representation of the 3D world projected on the image plane through a non-linear mapping, provide a 3D reconstruction of the scene, offering rich geometrical information about the humans present. A potential HPE system can see the actual appearance of objects without perspective distortion, in contrast to 2D images. This allows for more robust and accurate 3D pose estimation, as the depth values of the images correspond to real life 3D coordinates, leading to closer-to-reality detection results. This notion has been further validated in recent work for 3D hand pose estimation [Yuan et al., 2018b, Bilbeisi and Malik, 2019], where when utilised under the same conditions, RGB-only approaches fail to reach the accuracy of methods using depth data. Moreover, recent research has shown that the privacy-preserving, at least at a superficial level¹, representation

¹It is not possible for a person to easily recognise someone by simply looking at a depth image

of the environment by depth data, can increase the acceptance of such systems by the general public [Vasileiadis et al., 2016].

The actual challenge of the current research is twofold. First, it is imperative to achieve accurate and robust pose estimation. This requires developing a method that either outperforms, or performs comparably to other SOA methods, when evaluated on publicly available datasets, while simultaneously maintaining this level of accuracy when employed in unconstrained realistic settings. While many earlier pose estimation techniques are reported to achieve high joint estimation accuracy and real-time performance [Shotton et al., 2013, Ye and Yang, 2014], they are usually evaluated using datasets captured under ideal recording conditions in controlled laboratory environments (SMMC-10 [Ganapathi et al., 2010a], EVAL [Ganapathi et al., 2012b], PDT [Helten et al., 2013a]). As a result their accuracy and robustness degrades in real life applications, where various problems arise such as body part occlusions due to the presence of obstacles, partial-body views due to constraints in the available FOV of the camera, interaction with objects etc. [Obdrzalek et al., 2012, Plantard et al., 2015]. Moreover, the limited diversity of these datasets (small variation in body shape and size, fixed camera-human distance) makes them unsuitable for training large classifiers for pose estimation. Recently, the increased utilisation of Deep Learning for HPE [Toshev and Szegedy, 2014, Wei et al., 2016], along with the availability of large diverse 3D HPE datasets [Joo et al., 2019b] have partially alleviated those issues; however deep-learning based research specifically for HPE from 3D data has remained relatively limited [Haque et al., 2016b, Nishi and Miura, 2017].

The second issue which is addressed is related to the design of the algorithms from a computational complexity optimisation standpoint. Most of the current state-of-the-art pose estimation and tracking methods do not present any kind of flexibility regarding their design and implementation, as they are usually developed targeting, and thus constrained to, the best hardware available in order to achieve real-time performance and state-of-the-art accuracy. As a result, implementing these algorithms on systems of different architecture, especially ones with limited processing resources, becomes cumbersome and usually requires significant alterations of the original design in order to achieve the optimal accuracy/performance trade-off. On the other hand, a low computational complexity and parameterisable algorithm design can allow

the seamless and efficient implementation of the developed algorithms to a variety of platforms, especially low power devices.

While there are available lower computational complexity implementations of basic machine learning [MacLean, 2005], optimisation [Xu and Ziavras, 2003] and deep learning algorithms [Venieris and Bouganis, 2016, Howard et al., 2017], upon which most pose estimation techniques are based, very few research approaches can be found that target specifically the domain of human pose estimation on low power systems. This current PhD research aspires to fill this void by building upon state-of-the art human pose estimation and tracking algorithms while simultaneously utilising modern techniques for reducing the overall computational load, in order to develop a standalone high-performance low-powered human pose estimation module, suitable for deployment in systems with limited processing resources. Such systems can include embedded devices, low-end notebooks and modern mobile phones, which should be able to achieve close to real time ($>5\text{fps}$) processing rates in order to support higher level tasks where HPE acts as a low level data input. Applications in which the results of the current research can be used include autonomous robotic platforms, applications on handheld devices, autonomous driving, etc.

1.3 Thesis Contributions

The main contributions of the present thesis can be summarised as follows:

- A novel approach for real-time human pose estimation and tracking from depth/3D data. The proposed method employs a human-template based tracker utilising the Signed Distance Function (SDF) data representation. Additionally, a series of complementary tracking features are introduced, partially inspired by earlier works, tackling specifically the issues of free space violation, body part visibility, leg intersection, template initialisation and large object interaction, which are typically encountered under real-life monitoring conditions. The incorporation of all these individual mechanisms into a single HPE framework results in robust, state-of-the-art accuracy in both publicly available datasets

and unconstrained real-life settings. The proposed method’s efficiency is further demonstrated through its implementation within the context of daily activities recognition with a robotic agent.

- A deep learning based algorithm for multi person 3D human pose estimation, the first such published method for 3D data. The architecture follows the sequential network architecture paradigm, employing a fully-convolutional 3D-CNN network, which utilises a volumetric data representation, and generates 3D heatmaps corresponding to potential locations of the human joints in the scene. Additionally, a 3D body-part detector is incorporated, extending the architecture towards multi-person 3D pose estimation, without its runtime complexity being affected by the number of people within the scene. The developed architecture is evaluated on a series of publicly available datasets, outperforming other deep-learning based 3D HPE methods.
- A series of guidelines, in terms of network architecture design and input 3D data representation, that can reduce the computational complexity of already established 3D-CNN architectures, while maintaining comparable accuracy. These guidelines incorporate computationally efficient building blocks, such as 3D depthwise bottleneck units, SE blocks and decomposed convolutional layers. Their utilisation is investigated in depth, for the first time for 3D CNNs, leading to a novel, highly optimised version of the developed multi-person 3D HPE CNN, which performs comparably to the baseline model on two public datasets, while requiring significantly fewer computational resources. A speedup of up to 100x is achieved, along with a 50x smaller model, making it suitable for deployment on low power mobile devices.
- A novel human motion dataset simulating unconstrained realistic conditions. The dataset incorporates various challenges commonly encountered in realistic settings, including interaction with large objects, occlusion due to obstacles, out-of-FOV clipping etc., and has been made available to the scientific community.

1.4 List of Publications

The original contributions of this thesis have been published in the following peer-reviewed journals, conference and workshop proceedings:

Journals

[J2] M. Vasileiadis, C.S. Bouganis and D. Tzovaras, “Multi-Person 3D Pose Estimation from 3D Cloud Data Using 3D Convolutional Neural Networks”, *Computer Vision and Image Understanding*, vol. 185, pp. 12-23, 2019 [Vasileiadis et al., 2019b]

[J1] I. Kostavelis, M. Vasileiadis, E. Skartados, A. Kargakos, D. Giakoumis, C.S. Bouganis and D. Tzovaras, “Understanding of Human Behavior with a Robotic Agent through Daily Activity Analysis”, *International Journal of Social Robotics*, vol. 11, pp. 437-462, 2019 [Kostavelis et al., 2019]

Conference Proceedings

[C1] M. Vasileiadis, C.S. Bouganis, G. Stavropoulos and D. Tzovaras, “Optimising 3D-CNN Design towards Human Pose Estimation on Low Power Devices”, in *British Machine Vision Conference*, BMVA, 2019 [Vasileiadis et al., 2019a]

Workshop Proceedings

[W2] I. Kostavelis, D. Giakoumis, G. Peleka, A. Kargakos, E. Skartados, M. Vasileiadis and D. Tzovaras, “RAMCIP Robot: A Personal Robotic Assistant; Demonstration of a Complete Framework”, in *European Conference on Computer Vision Workshops*, 2018 [Kostavelis et al., 2018]

[W1] M. Vasileiadis, S. Malassiotis, D. Giakoumis, C.S. Bouganis and D. Tzovaras, “Robust

Human Pose Tracking for Realistic Service Robot Applications”, in *IEEE International Conference on Computer Vision Workshops*, pp. 1363-1372, IEEE, 2017 [Vasileiadis et al., 2017b]

1.5 Thesis Outline

The rest of the Thesis is organized as follows:

Chapter 2, contains a comprehensive literature review of existing work in fields related to the current research, covering the topic of human pose estimation from both the algorithmic and computational complexity viewpoint. Parts of the contents of this chapter have appeared in the following papers: [Vasileiadis et al., 2017b, Vasileiadis et al., 2019b, Vasileiadis et al., 2019a].

In **Chapter 3**, some basic concepts which are used throughout the thesis are introduced, in order to familiarise the reader with the notation of the proposed methods. The datasets and evaluation metrics used for the experiments are presented, setting the context within which the proposed methods are used. Parts of the contents of this chapter have appeared in the following papers: [Vasileiadis et al., 2017b, Vasileiadis et al., 2019a].

In **Chapter 4** the template-based human pose estimation and tracking framework is presented, including the main SDF tracker and the introduced complementary features and mechanisms. The implementation details are thoroughly described, along with extensive results from the experimental evaluation. The contents of this chapter have been published in the following papers: [Vasileiadis et al., 2017b, Kostavelis et al., 2019].

Chapter 5 presents the proposed deep-learning based approach for multi-person 3D HPE. The 3D-CNN architecture is analysed, along with details regarding the network implementation, training process and multi-person pose composition. The comparative experimental results are also presented, along with a discussion over the potential advantages of the DL-based design. The contents of this chapter have been published in [Vasileiadis et al., 2019b].

Based on the conclusions of the previous chapter, **Chapter 6** describes in detail the investigation of efficient CNN design techniques within the scope of 3D-CNNs, in order to reduce their

computational complexity. The proposed computationally efficient 3D-CNN for multi person 3D HPE is presented, along with detailed experimental and benchmarking results. Parts of the contents of this chapter have appeared in [Vasileiadis et al., 2019a].

Chapter 7 summarises the work conducted, discusses potential directions for future work and concludes the thesis.

Finally, an **Appendix** section is included, providing additional detailed results from the experimental evaluation conducted during the PhD.

Chapter 2

Literature Survey

This chapter presents a comprehensive survey of existing work in the fields relevant to the current research. It reviews earlier and modern methods for human pose estimation from RGB, depth and 3D data, as well as hardware-specific and hardware-agnostic acceleration strategies for the machine learning algorithms employed in these methods.

*Parts of the contents of this chapter have appeared in the following peer-reviewed papers:
[Vasileiadis et al., 2017b, Vasileiadis et al., 2019b, Vasileiadis et al., 2019a].*

The problem formulation, as stated in Section 1.2, implies that two different research areas need to be studied:

- Human pose estimation, which refers to the algorithms that are used to estimate the position of the skeletal joints from the input RGB and depth/3D data.
- Implementation on systems with limited computational resources, that is, how the HPE algorithms, either whole or in parts, can be redesigned and optimised, in terms of computational complexity, in order to be suitable for deployment in modern low power devices

Results of research efforts on both of these areas must be studied and combined, in order for the research problem to be addressed. Extensive work can be found in the literature regarding the above areas, with a review of the main state-of-the-art approaches given below.

2.1 Human Pose Estimation Algorithms

2.1.1 Algorithm Taxonomy

State of the art marker-less human pose tracking algorithms tend to fall mainly into three categories (Figure 2.1) [Ye and Yang, 2014]. Discriminative estimation approaches use large training datasets and machine learning techniques in order to map the extracted features from the input data to body parts and poses. Generative tracking approaches, on the other hand, try to match the input data to articulated body templates by minimising an objective function, utilising various optimisation techniques. There are also hybrid approaches which combine discriminative and generative techniques towards pose estimation.¹

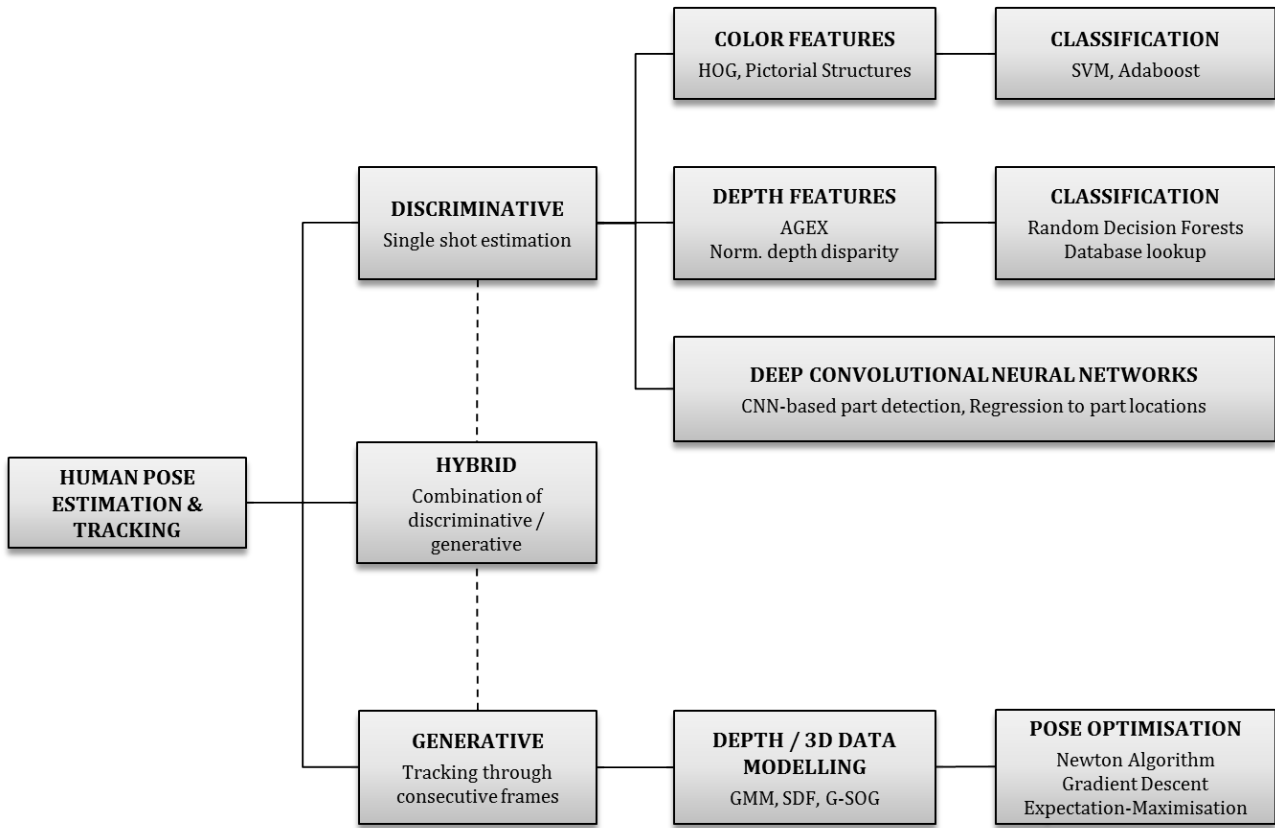


Figure 2.1: Block diagram of the taxonomy of state of the art marker-less pose estimation and tracking algorithms

¹The definitions provided herein are a simplified description of methods specifically from the limited point of view of human pose estimation and tracking. They should not be considered strict definitions of general discriminative and generative methods in machine learning

Discriminative approaches, also known as single shot pose estimators, have been successfully used for human pose estimation, utilising both RGB and Depth images [Liu et al., 2015]. They rely on large datasets and machine learning techniques in order to directly train the conditional probability of a body part within an image, thus estimating the skeletal joints' positions in new images by classifying each pixel, either on its own or in clusters, to a potential body part. As a result, they provide robust human pose estimation from a single frame, without the need of model fitting and explicit point correspondences, and most importantly, without requiring any prior knowledge regarding the human's position within the scene. This feature also ensures that any potential mistakes or failures in the detection of body parts in a frame will not affect the estimation accuracy in subsequent frames. However, discriminative approaches require extremely large and diverse datasets in order to train recognition models, which can be hard to acquire or generate, while the large amount of training data require considerable computing capacity to carry out the training process itself (e.g. in [Shotton et al., 2013] the authors state that training 3 randomised decision trees from 1 million images takes about a day on a 1000-core cluster). Moreover, they fail to efficiently handle poses and scenes not commonly found in their training data, such as occlusions or top-down views, leading to lower overall detection accuracy.

Generative approaches, also known as articulated body pose trackers, try to track the human pose by fitting a human body template to the observed data. The body template is usually comprised of an articulated skeleton which is made up by skeletal joints and fixed-length body parts, represented in the form of a kinematic tree, and a 3D mesh rigged to it, which represents the shape of the human. The pose of the body template is described by a pose vector which includes the rotation of each rigid body part relative to its parent, and the global rotation and translation of the whole template with reference to the camera frame. As a result, the fitting process involves the estimation of the optimal values of the pose vector that minimise an objective function, which expresses the similarity between the input data and the human template for a given pose. This is usually achieved by calculating the partial derivatives of the objective function with respect to the pose vector and using numerical optimisation algorithms to estimate the optimal solution. Generative approaches, in general, offer higher detection

accuracy, more consistent joint estimations through consecutive frames and better occlusion handling compared to discriminative techniques, mainly due to the use of a life-like human template. While they don't require any prior training, generative approaches do need a rough initial pose of the body. In order for the initial pose estimation to be successful, the starting pose of the body template must be similar to the actual pose of the human, so that the optimisation algorithm will be able to converge to the actual pose. Once the model converges to the tracked human's starting pose, these methods proceed with the tracking of the subsequent poses of the monitored subject, based on the prior knowledge and the new observations. Moreover, if there are large body movements between consecutive observations, the optimisation process can get trapped in local minima and fail to recover properly, thus making it necessary to achieve a high framerate in order ensure smooth and limited body motion between frames.

Combinations of generative and discriminative approaches have also been proposed, in order to further increase the pose estimation accuracy. In these *hybrid approaches*, pose tracking is usually performed through generative techniques, while the discriminative algorithms work complementary to the main tracker to propose potential joint positions, initialise the pose tracker or recover it from failure if it gets trapped in local minima. Hybrid methods combine the robustness of single-shot estimators with the accuracy of articulated trackers, offering an efficient solution that can work not only in controlled laboratory environments, but under real-life conditions as well. However, the combination of two discrete modalities, vastly different in design, can increase the computational complexity of the implementation.

2.1.2 Pose Accuracy Measures

In order to quantitatively evaluate the accuracy of the reviewed state-of-the-art methods, allowing for direct performance comparison between them, it is necessary to use evaluation measures that can provide a clear and insightful representation of a method's pose estimation capacity. In the following the definitions of the most common evaluation measures for human pose estimation, which are used in relevant literature and throughout this report, are given. All measures are based on the Euclidean distance between the ground-truth and the estimated

positions of the human joints and body parts. In the case of 3D poses the distance is calculated in absolute values (meters), while for 2D poses a normalisation factor is usually introduced, since the distance in pixels is relative to the dimensions of the image and the camera's point of view.

Accuracy Measures in 2D Space

Percentage of Correctly Localized Parts (PCP) [Ferrari et al., 2008] is defined as the percentage of body parts where the estimated segment endpoints lie within 50% of the length of the ground-truth segment from their annotated location.

Probability of Correct Keypoint (PCK) [Yang and Ramanan, 2013] is defined as the proportion of joints where the distance in pixels between the estimated joint positions and their respective ground truth joint positions is within a certain maximum distance, which is defined as a fraction (usually 20%) of the person's bounding box size. A modified version of this measure is introduced in [Andriluka et al., 2014], **PCKh**, where the matching threshold is defined as 50% of the head segment length.

Average Precision of Object Keypoint Similarity (AP^{O_KS}) [Lin et al., 2014] is defined as the average precision for $O_{KS} = 0.50, 0.05, 0.95$. O_KS is calculated as:

$$O_{KS} = \sum_{i=1}^n \exp\left(\frac{-d_i^2}{2s^2k_i^2}\right) \delta(v_i > 0) / \sum_{i=1}^n \delta(v_i > 0) \quad (2.1)$$

where n the total number of joints in an image, d_i the Euclidean distance in pixels between the estimated joint position and the corresponding ground truth joint position, s the object scale, k_i a per-joint constant and v_i a visibility flag.

Accuracy Measures in 3D Space

Average Distance Error (ADE) is defined as the average Euclidean distance between the estimated joint positions and their respective ground truth joint positions.

Mean Average Precision (mAP) is defined as the proportion of joints where the Euclidean distance between the estimated joint positions and their respective ground truth joint positions is within a certain maximum Euclidean distance, usually set to $0.1m$.

Percent of Detected Joints (PDJ) [Wang et al., 2016] is defined as the percentage of joints where the distance between the ground truth and the estimated joint positions is within a certain fraction of the torso diameter. The final accuracy is the average accuracy over four fraction values: 0.05, 0.1, 0.15, 0.2.

2.1.3 2D Human Pose Estimation from RGB Data using Hand-crafted Features

Early human pose estimation approaches utilised handcrafted features in order to detect human body joints in RGB images. [Bourdev and Malik, 2009] introduce the concept of “poselets”, which refer to pieces of human poses that are tightly clustered in both appearance and configuration spaces, for human body part detection from RGB images. HOG [Dalal and Triggs, 2005] features are extracted and linear SVMs are used to train classifiers, in order to detect poselets for each body part in images, through multi-scale scanning. A two-step training process is employed; an initial classifier is trained using the positive and a random set of negative examples, which is then used to scan over images not containing people and collect false positives. Next a second round of training follows including the detected hard false positives into the negative training set. [Wang et al., 2011] extend [Bourdev and Malik, 2009] by introducing a structured hierarchical model where each poselet can correspond to a single rigid body part or a larger piece of the human body comprised by

multiple hierarchically connected body parts. [Andriluka et al., 2009] use a Pictorial Structure Model [Felzenszwalb and Huttenlocher, 2005] for human pose estimation. The human body is represented through a kinematic tree, while the appearance of body parts is modelled using densely sampled shape context descriptors [Mikolajczyk and Schmid, 2005] and discriminatively trained AdaBoost classifiers [Freund and Schapire, 1995]. A two-step training process is employed, similar to [Bourdev and Malik, 2009].

Combining pictorial structures and poselets, in [Pishchulin et al., 2013a] a conditional model is defined, in which all body parts are connected a-priori, but which becomes a tractable tree-structured Pictorial Structure model once the image observations are available, relying on poselet-based features to derive a set of conditioning variables. [Yang and Ramanan, 2013] propose a novel representation of deformable part models, modelling articulation using a mixture of small, non-oriented parts. A flexible mixture model jointly captures spatial relations between part locations and co-occurrence relations between part mixtures, augmenting standard Pictorial Structure Models, while all the parameters are learned through a structured SVM solver. Meanwhile, in [Pishchulin et al., 2013b] the authors investigate the augmentation of Pictorial Structure Models through the incorporation of various complementary local appearance representations which build on findings from the literature.

Table 2.1 presents the results of the experimental evaluation of the above methods, as reported by their authors. From the results, it becomes apparent that while there is an incremental increase in prediction accuracy, these initial RGB-based methods are not sufficient for high-accuracy human pose estimation, as the limited discriminative capacity of the handcrafted features and classifiers, along with the lack of large and diverse datasets at the time, severely hinder their pose estimation potential especially in challenging poses. More recent methods, utilising deep-learning for RGB-based HPE, are presented in section 2.1.5.

Table 2.1: Experimental evaluation of early RGB-based discriminative HPE approaches using handcrafted features.

Method	Dataset	PCP
[Bourdev and Malik, 2009]	PASCAL VOC2007 [Everingham et al., 2007]	36.5%
[Andriluka et al., 2009]	UIUC People [Tran and Forsyth, 2010]	32.1%
	Sport Image [Wang et al., 2011]	37.4%
[Wang et al., 2011]	UIUC People	47.0%
	Sport Image	37.3%
[Yang and Ramanan, 2013]	Leeds Sports Dataset [Johnson and Everingham, 2010]	60.8%
	Image Parse [Ramanan, 2007]	60.7%
[Pishchulin et al., 2013a]	Leeds Sports Dataset	62.9%
	Image Parse	63.1%
[Pishchulin et al., 2013b]	Leeds Sports Dataset	69.2%
	Image Parse	69.4%

2.1.4 3D Human Pose Estimation from Depth/3D Data using Handcrafted Features

Following the launch of the Microsoft Kinect RGB-D camera in 2011 [Zhang, 2012], the research focus shifted towards the utilisation of depth/3D data, for body part detection. [Plagemann et al., 2010] propose a novel interest point detector suitable for mesh and depth data. The interest points, called “Accumulative Geodesic EXtrema” (AGEX), are computed in 3D space by incrementally maximising geodesic distances on the surface of the 3D mesh, in an iterative way using Dijkstra’s algorithm [Dijkstra, 1959]. Small depth image patches (41x41px) surrounding these points are then used as local descriptors in order to train a boosted classifier. The final training set is comprised of a set of positive examples (patches around interest points) and a large quantity of negative examples (patches around background/no-interest

points). [Shotton et al., 2013] use randomised decision trees and forests for body part detection and treat the body part segmentation as a per-pixel classification task. Each pixel in the depth image is evaluated separately and is classified into one of 32 human body parts, employing a computationally inexpensive depth comparison feature. Next, a weighted mean shift-based [Comaniciu and Meer, 2002] approach is used to estimate the body pose based on the inferred body part probability on each pixel. Training of the decision trees is performed using a large synthetic dataset following the training procedure described in [Lepetit et al., 2005]. Similarly, [Pons-Moll et al., 2015] also utilise randomised decision forests, but propose an alternative training approach by employing the “Metric Space Information Gain” (MSIG) training objective. Targeting high performance on low-powered hardware, [Jung et al., 2015] also employ randomised decision trees for pose estimation. However, they achieve a large computation gain by substituting pixel-wise classification with the estimation of the probability distribution to the direction towards the particular joint [Xiong and De la Torre, 2013], relative to the current position. This process, known as “Random Tree Walks” (RTW), greatly reduces the computation time of the pose estimation process, since instead of taking into consideration all the available pixels, it uses a randomly selected sample around each joint.

Moving towards generative approaches, trying to track the human pose through consecutive frames, [Ganapathi et al., 2012b] use a Dynamic Bayesian Network to model human motion and introduce an enhanced ICP-based model [Demirdjian et al., 2003] which utilises free space constraints, termed “Ray-Constrained ICP Model”, by applying Chamfer distance transforms [Balan et al., 2007]. The human body is modelled as a collection of linked 3D capsules, while two complementary features, regarding self-collisions and limb movement constraints, are also proposed. A generalisation of Signed Distance Functions (SDF) [Felzenszwalb and Huttenlocher, 2004] for articulated objects is introduced in [Schmidt et al., 2015]. Objects are represented by a symmetric version of the articulated SDF in 3D space, which can be precomputed, allowing for a dense representation of the data. The human body is modelled by a detailed realistic human mesh and gradient based optimisation is employed to estimate the optimal pose, taking into consideration free space constraints similar to [Ganapathi et al., 2012b]. [Ye and Yang, 2014] relate the observed data to a realistic skinned

body template by assuming that the observed point cloud follows a Gaussian Mixture Model (GMM) whose centroids are the deformed template vertices. Simultaneously to pose estimation, shape adaptation is also performed by optimising the human template in regard to limb length and body part geometry. The pose estimation and shape adaptation problem is then solved iteratively using the Expectation-Maximisation (EM) algorithm [Dempster et al., 1977]. Towards achieving high performance for non-GPU optimised implementations, a Generalised Sum-of-Gaussians (G-SOG) model for human shape modelling is presented in [Ding and Fan, 2015]. The observed data are represented by isotropic Gaussians through octree partitioning, and multivariate Gaussian kernel correlation is employed as a similarity measure between the body template and the observation. Body part visibility and movement smoothness terms are also added to the objective function, which is optimised by employing a Quasi-Newton algorithm (L-BFGS [Liu and Nocedal, 1989]).

Combining both discriminative and generative approaches, [Wei et al., 2012] use a gradient-based optimiser to track the human pose, and combine it with a randomised decision trees-based body part detector similar to [Shotton et al., 2013], used only for initialisation and tracking failure recovery. Tracking failure is detected by estimating the average 3D error and the overlap on the 2D plane between the reconstructed and the observed human in the scene. In similar fashion, [Ganapathi et al., 2010a] also utilise a gradient-based optimiser, modelling the system as a DBN, similar to [Ganapathi et al., 2012b], in conjunction with the body part detector from [Plagemann et al., 2010]. However, in contrast to [Wei et al., 2012], the body part proposals are taken into consideration in every frame and not only during the initialisation/reset phase. In both cases, the correspondence between the estimated joint positions and the pose of the human template is figured through inverse kinematics. In [Taylor et al., 2012], the authors further extend the random forests-based approach by learning to predict direct correspondences between input image pixels and a 3D articulated mesh model, with the optimisation process being handled by a Quasi-Newton algorithm (L-BFGS). [Baak et al., 2013] provide a rough pose estimation for each frame by finding the pose most similar to the observation, from a database of pre-rendered mesh models, using a matching process based on the feature extraction algorithm from [Plagemann et al., 2010]. The final human pose is extracted by fusing the two alternative

pose hypotheses, from the generative and the discriminative components, through a novel voting scheme based on sparse Hausdorff distance [Huttenlocher et al., 1993]. Database lookup is also employed in [Ye et al., 2011], where PCA of normalised depth images is used to find an initial rough pose, which is then further refined through non-rigid registration between the observed point cloud and the estimated human pose.

The results of the experimental evaluation of the depth-based human pose estimation and tracking methods described in this Section, as reported by their corresponding authors, are presented in Table 2.2. The SMMC-10 [Ganapathi et al., 2010a], EVAL [Ganapathi et al., 2012b], PDT [Helten et al., 2013a] and ITOP [Haque et al., 2016b] datasets are publicly available datasets that include real recordings along with ground truth data from marker-based pose tracking, while the Taylor Dataset [Taylor et al., 2012] is generated by synthetic human meshes. The motion complexity of SMMC-10 is relatively simple, which leads to most methods achieving a Mean Average Precision (mAP) score of over 95%, while the other 4 datasets present more challenging motion sequences. Currently [Ye and Yang, 2014] achieves the highest mAP score on the relatively complex EVAL dataset, with [Schmidt et al., 2015] a close second; the small difference in accuracy can be attributed to the body template shape adaptation component of [Ye and Yang, 2014]. More recent methods, utilising deep-learning for depth and 3D point cloud-based HPE, are presented in section 2.1.7.

Table 2.2: Experimental evaluation of early non-DL Depth/3D-based approaches, as reported by their authors.

	Method	Data Type	Dataset	ADE	mAP
Discriminative	[Shotton et al., 2013]	Depth map	SMMC-10 [Ganapathi et al., 2010a]	-	95%
			PDT [Helten et al., 2013a]	0.080	-
			ITOP (front) [Haque et al., 2016b]	-	65.8%*
			ITOP (top) [Haque et al., 2016b]	-	47.4%*
	[Pons-Moll et al., 2015]	Depth map	Taylor Dataset [Taylor et al., 2012]	-	78%
	[Jung et al., 2015]	Depth map	SMMC-10	-	96%
			EVAL [Ganapathi et al., 2012b]	-	74%**
			ITOP (front)	-	80.5%*
			ITOP (top)	-	68.2%*
Generative	[Ganapathi et al., 2012b]	3D point cloud	SMMC-10	-	97%
			EVAL	-	85%
	[Schmidt et al., 2015]	3D point cloud	EVAL	-	90%
	[Ye and Yang, 2014]	3D point cloud	SMMC-10	0.035	96%
			EVAL	-	92%
			PDT	0.046	92%
	[Ding and Fan, 2015]	3D point cloud	SMMC-10	0.039	96%
	[Ganapathi et al., 2010a]	Depth + 3D	SMMC-10	0.070	91%
	[Ye et al., 2011]	3D point cloud	SMMC-10	0.040	97%
Hybrid	[Taylor et al., 2012]	Depth + 3D	Taylor Dataset	0.092	-
			SMMC-10	0.039	-
	[Wei et al., 2012]	Depth + 3D	SMMC-10	-	96%
	[Baak et al., 2013]	3D point cloud	SMMC-10	0.060	-
			PDT	0.070	-

*results reported in [Haque et al., 2016b]

**original results retracted in [Jung, 2017]

2.1.5 2D Human Pose Estimation from RGB Data via Deep Learning

Recently, methods utilising Convolutional Neural Networks [LeCun et al., 1998a] have achieved remarkable results in a variety of Computer Vision tasks [Krizhevsky et al., 2012] significantly outperforming prior work, including the task of human pose estimation. A large number of methods have been proposed towards 2D and 3D human pose estimation from single images, for either single or multi-person cases. These CNN-based methods can generally be classified into two categories: *Detection-based* methods, which rely on powerful CNN-based part detectors which are then combined using graphical models or refined using regression, and *Regression-based* methods that try to learn a direct mapping from the image and CNN features to part locations.

Single Human Pose Estimation

The first application of CNNs towards 2D single human pose estimation from a RGB image can be found in [Toshev and Szegedy, 2014], where a 7-layer network, based on [Krizhevsky et al., 2012], is used to regress to the location of each body joint. [Tompson et al., 2014] propose a hybrid architecture, that consists of a deep Convolutional Network and a Markov Random Field, which can exploit structural domain constraints such as geometric relationships between body joint locations. Similarly, in [Chen and Yuille, 2014] a graphical model for human pose is utilised, exploiting the fact the local image measurements can be used both to detect parts (or joints) and also to predict the spatial relationships between them (“Image Dependent Pairwise Relations”).

[Wei et al., 2016] introduce “Convolutional Pose Machines”, which consist of a sequence of convolutional networks that repeatedly produce 2D belief maps with increasingly refined estimates for the locations of each part, while in [Carreira et al., 2016] the “Iterative Error Feedback” process is presented; a self-correcting model that progressively changes an initial solution by feeding back error predictions. In [Newell et al., 2016] the “stacked hourglass network” ex-

tends Fully Convolutional [Long et al., 2015] and Deconvolutional [Zeiler et al., 2011] networks within the concept of residual learning [He et al., 2016]. [Newell et al., 2016] is further extended in [Chu et al., 2017] through the introduction of a novel generic hourglass residual unit along with a CRF-based attention mechanism, and in [Yang et al., 2017] through the utilisation of multi-scale feature pyramids. [Belagiannis and Zisserman, 2017] combine a feed forward module with a recurrent module, which can be run iteratively to improve performance. [Bulat and Tzimiropoulos, 2016] propose a detection-followed-by-regression CNN cascade where the first part of the cascade outputs part detection heatmaps and the second part performs regression on these heatmaps. Towards human pose estimation with limited resources, a lightweight design is proposed in [Bulat and Tzimiropoulos, 2017], inspired by the Binarized Neural Networks architecture [Courbariaux et al., 2016].

Table 2.3 presents the experimental results of the deep-learning based 2D single human pose estimation methods, as reported in the respective papers. From the results it becomes evident that there is an incremental increase in accuracy as newer more powerful models are proposed, with the latest ones achieving extremely high accuracy scores of over 90%. Moreover, when compared with earlier non-DL HPE algorithms, described in Section 2.1.3, a significant improvement can be observed, clearly showcasing the superiority of deep CNNs over handcrafted features.

Table 2.3: Experimental evaluation of DL-based 2D single human pose estimation methods from RGB data, as reported in the respective papers.

Method	Dataset	PCP	PCK	PCKh
[Toshev and Szegedy, 2014]	Leeds Sports Dataset [Johnson and Everingham, 2010]	61.0%	-	-
	Image Parse	69.0%	-	-
[Tompson et al., 2014]	Leeds Sports Dataset	66.6%	72.3%	-
	MP II [Andriluka et al., 2014]	-	-	79.6%
[Chen and Yuille, 2014]	Leeds Sports Dataset	75.0%	73.4%	-
	FLIC [Sapp and Taskar, 2013]	91.9%	-	-
[Bulat and Tzimiropoulos, 2016]	Leeds Sports Dataset	-	90.7%	-
	MP II	-	-	89.7%
[Wei et al., 2016]	Leeds Sports Dataset	72.5%	90.5%	-
	MP II	-	-	88.5%
[Newell et al., 2016]	MP II	-	-	90.9%
[Carreira et al., 2016]	Leeds Sports Dataset	-	73.1%	-
	MP II	-	-	81.3%
[Chu et al., 2017]	Leeds Sports Dataset	-	92.6%	-
	MP II	-	-	91.5%
[Belagiannis and Zisserman, 2017]	Leeds Sports Dataset	-	85.2%	-
	MP II	-	-	86.7%
[Yang et al., 2017]	Leeds Sports Dataset	-	93.9%	-
	MP II	-	-	91.8%
[Bulat and Tzimiropoulos, 2017]	MP II	-	-	78.1%

Multi-Person Pose Estimation

Moving towards inferring the pose of multiple people in images, two types of approaches are proposed. *Top-down* methods initially employ a person detector in order to extract the bounding box of each person in the image, with single person pose estimation being performed next on each detection. These methods, however, present some drawbacks, as it is impossible to recover from a potential failure of the initial person detection, thus making it necessary to incorporate post processing techniques to compensate for missed or poor human detections. Moreover their

runtime is proportional to the number of people in the scene, since for each person detection a single-person pose estimation has to be run.

[Iqbal and Gall, 2016] employ a faster-RCNN human detector [Ren et al., 2015] along with CPM [Wei et al., 2016] to generate heatmaps. [Papandreou et al., 2017] also utilise the faster-RCNN, on top of a ResNet 101 CNN [He et al., 2016] which produces heatmaps and 2D offset vectors. Meanwhile [Fang et al., 2017, Chen et al., 2018, Xiao et al., 2018] achieve SoA accuracy by introducing improved versions of the stacked hourglass model [Newell et al., 2016], in combination with faster-RCNN and FPN [Lin et al., 2017b] detectors.

Bottom-up approaches, in which keypoint proposals are grouped together into person instances, alleviate the issues of top-down approaches. However they may suffer from efficiency issues as they can be extremely taxing computationally, with a runtime of multiple seconds per image. [Pishchulin et al., 2016] propose an approach that jointly labels part detection candidates and associates them to individual people, while [Insafutdinov et al., 2016] build on [Pishchulin et al., 2016] by employing part detectors based on the concept of residual learning [He et al., 2016] and image-dependent pairwise scores [Chen and Yuille, 2014]. Meanwhile, in [Cao et al., 2017] an efficient real-time design is presented, representing association scores via “Part Affinity Fields” (PAFs), a set of 2D vectors that encode the location and orientation of limbs over the image domain. [Papandreou et al., 2018] extend [Papandreou et al., 2017] by incorporating short and mid-range 2D offset vectors, which are used to compose the detected joints into poses.

The experimental results of the multi-person pose estimation approaches, as reported by their authors, are presented in Table 2.4. Top-down methods in general slightly outperform bottom-up methods, indicating that they can successfully overcome potential misses of the human detectors. However the all-in-one nature of bottom-up methods, as they don’t require a separate network for detection, can potentially provide a better speed/accuracy trade-off, especially since the latest methods [Cao et al., 2017, Kocabas et al., 2018] have achieved real-time performance.

Table 2.4: Experimental results of the DL-based 2D multi-person pose estimation methods from RGB data, as reported in the respective papers.

	Method	Dataset	PCKh	AP
Top-Down	[Iqbal and Gall, 2016]	MPII [Andriluka et al., 2014]	43.1%	-
	[Papandreou et al., 2017]	MSCOCO [Lin et al., 2014]	-	64.9%
	[Insafutdinov et al., 2017]	MPII [Andriluka et al., 2014]	74.3%	-
	[Fang et al., 2017]	MPII	76.7%	-
		MSCOCO	-	71.0%
	[Chen et al., 2018]	MSCOCO	-	72.1%
	[Xiao et al., 2018]	MSCOCO	-	73.7%
Bottom-Up	[Insafutdinov et al., 2016]	MPII	59.5%	-
	[Cao et al., 2017]	MPII	75.6%	-
		MSCOCO	-	64.2%
	[Newell et al., 2017]	MPII	77.5%	-
		MSCOCO	-	65.5%
	[Papandreou et al., 2018]	MSCOCO	-	68.7%
	[Kocabas et al., 2018]	MSCOCO	-	69.6%

Pose Estimation from Video

The methods described above generate predictions from single images, addressing, in the case of videos, every individual frame independently, without taking into consideration potential spatio-temporal information that may be available in sequential data. Towards this end, novel methods have been proposed, incorporating temporal information into the prediction process, in order to increase human pose estimation accuracy in videos.

[Jain et al., 2014] propose the use of motion features, generated from a pair of consecutive

frames, to improve the accuracy in video sequences, with the combination of the RGB frame and its corresponding Optical flow image achieving the best results. [Pfister et al., 2014] directly stack consecutive RGB frames at the input of a deep CNN, allowing the network to learn the temporal coherence and directly regress the pose. A more complex approach is employed in [Pfister et al., 2015]: first for each frame in the input a separate pose confidence heatmap is predicted. The heatmaps are then warped to the target frame using optical flow, and pooled through a convolutional layer, which learns how to weigh the warped heatmaps from nearby frames. A similar approach is proposed by [Song et al., 2017], with the heatmap warping, however, now performed through deep warping layers. In the final step a spatio-temporal inference layer is trained, to conduct inference between body parts spatially and temporally, producing the final joint position estimates.

[Charles et al., 2016] employ a 4-stage process in order to generate high quality personalised pose estimators which adapt themselves to a person’s appearance. First the pose estimator from [Pfister et al., 2015] is used to generate high quality initial annotations for a few frames. Patches from the remaining frames are then matched to patches of the annotated ones, and the pose annotations are spread temporally to neighbouring frames. Finally, through iterative self-evaluation, false new annotations are discarded, and the resulting annotations are used to train a personalized pose estimator. [Gkioxari et al., 2016] adapt the sequence-to-sequence model [Sutskever et al., 2014] for structured vision tasks. The output variables for a given input are predicted sequentially, depending not only on the current input image but also on all the previous predictions. Meanwhile, in [Xiaohan Nie et al., 2015, Iqbal et al., 2017] action and activity priors are employed in order to refine initial pose estimations.

The experimental results of the aforementioned methods, as reported in the respective papers, are presented in Table 2.5.

Table 2.5: Experimental results of the DL-based 2D human pose estimation methods from RGB videos, as reported in the respective papers.

Method	Dataset	PCK
[Pfister et al., 2014]	BBC Pose [Pfister et al., 2014]	62.0%
[Pfister et al., 2015]	BBC Pose	85.0%
[Xiaohan Nie et al., 2015]	Penn Action [Zhang et al., 2013b]	48.0%
[Gkioxari et al., 2016]	Penn Action	91.8%
[Iqbal et al., 2017]	Penn Action	81.1%
[Song et al., 2017]	Penn Action	96.4%

2.1.6 3D Human Pose Estimation from RGB Data via Deep Learning

Further expanding the 2D pose estimation techniques, various DL-based methods have been proposed for 3D human pose estimation from monocular RGB images, attempting to simultaneously enforce 2D spatial relationships and 3D geometry constraints, utilizing datasets that include both RGB and motion capture data [Ionescu et al., 2014]. Recovering the 3D pose from 2D images is an extremely challenging task, when compared to 2D pose estimation, due to the much larger 3D pose space, more ambiguities and the problem of irreversible perspective projection.

[Li and Chan, 2014] are the first to apply CNNs for 3D human pose estimation presenting a multi-task framework that simultaneously trains the regression and detection tasks as well as pre-training the regression task with detection tasks. In [Du et al., 2016] additional built-in knowledge is introduced for reconstructing the 2D pose thus formulating an objective function to estimate the 3D pose from the detected 2D pose, while in [Zhou et al., 2016b] a kinematic object model is directly embedded into the deep learning process.

Exploiting the temporal information in sequences of images and videos, [Tekin et al., 2016] utilize motion information to regress to the 3D pose, while [Zhou et al., 2016c] introduce a novel synthesis method, combining a CNN-based 2D part detector, a sparsity-driven 3D reconstruction approach and a temporal smoothness prior. In [Lin et al., 2017a] the authors propose a model that recurrently integrates spatial and temporal long-range dependencies, based on LSTM layers [Hochreiter and Schmidhuber, 1997], achieving important gains in terms of pose detection accuracy. [Bogo et al., 2016] perform simultaneous 3D pose estimation and shape estimation by combining the 2D pose estimator [Pishchulin et al., 2016] with the SMPL statistical body shape model [Loper et al., 2015]. [Pavlakos et al., 2017] discretise of the 3D space around the subject and train a ConvNet to predict per voxel likelihoods for each joint, followed by a coarse-to-fine prediction scheme. [Chen and Ramanan, 2017] estimate the 2D pose using a ResNet 101 CNN [He et al., 2016] and look up a suitable 3D pose from a dictionary, while [Mehta et al., 2017] also follow a two stage approach, adopting transfer learning from 2D to 3D HPE. Similarly [Fang et al., 2018] and [Yang et al., 2018] utilize the 2D/3D paradigm employing a grammar model [Han and Zhu, 2008] and adversarial learning [Goodfellow et al., 2014] respectively to further boost accuracy.

The experimental results of the aforementioned methods, as reported in the respective papers, are presented in Table 2.6.

Table 2.6: Experimental results of the DL-based 3D human pose estimation methods from RGB data, as reported in the respective papers.

Method	Dataset	ADE(m)
[Li and Chan, 2014]	Human3.6M [Ionescu et al., 2014]	0.121*
[Du et al., 2016]	Human3.6M	0.126
[Tekin et al., 2016]	Human3.6M	0.125
[Zhou et al., 2016c]	Human3.6M	0.113
[Zhou et al., 2016b]	Human3.6M	0.107
[Bogo et al., 2016]	Human3.6M	0.069
[Lin et al., 2017a]	Human3.6M	0.073
[Pavlakos et al., 2017]	Human3.6M	0.072
[Mehta et al., 2017]	Human3.6M	0.069
[Chen and Ramanan, 2017]	Human3.6M	0.069
[Fang et al., 2018]	Human3.6M	0.060
[Yang et al., 2018]	Human3.6M	0.059

*tested on a limited subset of the dataset

2.1.7 3D Human Pose Estimation from Depth/3D Data via Deep Learning

While a multitude of approaches have been proposed towards CNN-based pose estimation from RGB images, as described in Section 2.1.5 above, respective work on depth data has been rather limited. The relative lack of CNN-based pose estimation techniques from depth data can be attributed to various factors such as the rather limited size and diversity of publicly available depth datasets [Ganapathi et al., 2010a, Ganapathi et al., 2012b, Helten et al., 2013a], when compared to similar RGB datasets [Lin et al., 2014], which may not be sufficient for training

deep neural network architectures, potentially leading to over-fitting. Moreover, the much larger integration of RGB imaging in everyday tasks, as well as the large amount of work already done on depth-based pose estimation (Section 2.1.4) prior to the advent of CNNs, have reduced the interest of the research community towards this task.

Currently available non-DL depth-based approaches, however, present one major drawback, as they require human / background segmentation in order to perform efficiently, which severely hinders their performance in unconstrained settings, especially in the case of a moving camera (i.e. on mobile robotic platform). This issue, along with the recent introduction of large-scale synthetic [Varol et al., 2017] and real [Joo et al., 2019b] depth datasets, make CNN-based pose estimation from depth data an attractive solution, since depth imaging can provide much higher detection accuracy in the case of 3D human pose estimation, as it becomes evident from the experimental results presented in Tables 2.2 and 2.6.

[Wang et al., 2016] achieve real time performance combining an 8-layer FCN with a novel inference MatchNet [Han et al., 2015c] structure, which enhances appearance and geometric compatibility between body part proposals and templates. [Haque et al., 2016b] investigate the issue of pose estimation from extremely challenging top-down viewpoints, adopting an iterative refinement technique inspired by [Carreira et al., 2016], while [Moon et al., 2017] improve on the results of [Haque et al., 2016b] by using a 2D CNN detector to find regions of interest on the depth map which are then represented in the form of a 3D occupancy grid [Maturana and Scherer, 2015] and fed into a 3D CNN architecture to estimate per-voxel likelihood for body joints localisations. A similar approach is followed in [Moon et al., 2018] where a 2D human detector is combined with a 3D CNN architecture based on the stacked hourglass network [Newell et al., 2016]. In [Nishi and Miura, 2017] a network architecture similar to [Oliveira et al., 2016] is employed to detect challenging sitting and lying poses.

Table 2.7 presents the experimental results, as reported by their authors, of the DL-based pose estimation techniques from depth data. The relatively lower accuracy reported in the EVAL dataset, in comparison to the non-CNN methods described in Section 2.1.4, indicates that there is plenty of room for improvement, since CNNs have repeatedly demonstrated their

superior learning capacity in similar RGB-based detection tasks. Moreover, similar to non-DL methods, approaches that utilise 3D data outperform pure 2D-based ones, as 3D point clouds provide a representation of the scene rich in geometrical features. However, since both 3D based methods [Moon et al., 2017, Moon et al., 2018] rely on 2D ROI detectors, a pure 3D approach can potentially lead to further improvements.

Table 2.7: Experimental results of the DL-based human pose estimation methods from depth/3D data, as reported in the respective papers.

Method	Data Type	Dataset	mAP	PDJ
[Toshev and Szegedy, 2014]	Depth map	K2HGD [Wang et al., 2016]	-	60.3%*
[Chen and Yuille, 2014]	Depth map	K2HGD	-	67.0%*
[Wang et al., 2016]	Depth map	K2HGD	-	71.6%
		EVAL [Ganapathi et al., 2012b]	74.1%	-
[Haque et al., 2016b]	Depth map	ITOP (front) [Haque et al., 2016b]	77.4%	-
		ITOP (top)	75.5%	-
		ITOP (front)	71.4%**	-
[Carreira et al., 2016]	Depth map	ITOP (top)	51.2%**	-
		EVAL	79.8%	-
[Moon et al., 2017]	Depth + 3D	ITOP (front)	83.4%	-
		ITOP (front)	84.9%	-
[Guo et al., 2017]	Depth map	ITOP (top)	75.5%	-
		ITOP (front)	88.7%	-
[Moon et al., 2018]	Depth + 3D	ITOP (top)	83.4%	-

*results reported in [Wang et al., 2016]

**results reported in [Haque et al., 2016b]

2.2 Optimisation for Systems with Limited Computational Resources

Estimating and tracking the human pose is a computationally taxing process, both in terms of algorithmic complexity and memory bandwidth. It often requires the utilisation of high performance hardware components in order to achieve high accuracy and real time processing rate.

Research, however, specifically in the area of HPE on low power systems has been rather minimal, as most state-of-the-art methods utilise powerful processing units without any consideration for run-time efficiency; e.g. the initial non-optimised version of [Cao et al., 2017] required a 4-GPU configuration to achieve real time performance. Few methods have attempted to achieve high performance with limited resources, usually approaching the issue from an algorithmic point of view, introducing novel software designs such as Random Tree Walks [Jung et al., 2015] and Binarized Convolutional Networks [Bulat and Tzimiropoulos, 2017], or through optimised hardware-specific implementations [Oberg et al., 2012].

A lot of research effort, on the other hand, has been put towards the design and development of highly optimised implementations of the machine learning, optimisation and general computer vision algorithms that state-of-the-art pose estimation and tracking techniques are based on. This challenging aspect has been tackled through the progressive advancement, in terms of raw processing power, ease of use and availability, of GPU-based solutions, significantly boosted by the release of Nvidia’s CUDA API model [Nvidia, 2007] which provided direct access to the GPU’s virtual instruction set and parallel computational elements, allowing developers to use a CUDA-enabled devices for general purpose processing.

Moreover, reconfigurable hardware, mainly in the form of Field Programmable Gate Arrays (FPGAs), has recently become increasingly attractive for digital signal processing problems, including image processing and computer vision tasks. In [MacLean, 2005] an evaluation of the suitability of FPGAs for embedded vision systems is presented, with the authors pointing out towards the reconfigurability and the ability to exploit parallelism in vision problems which can

lead to substantial performance improvements. On the other hand, some major drawbacks are also reported, including fixed-point arithmetic representation and relatively complicated design process.

Additionally, the increased utilisation of deep learning for various computer vision tasks, has also led to the development of a multitude of methods targeting the reduction of the computational complexity of such models [Cheng et al., 2017]. These methods follow a framework-agnostic approach for optimising the efficiency of neural networks, not exploiting any hardware-specific design properties of the target devices, but rather attempting to reduce the overall number of necessary arithmetic and memory access operations, thus allowing the seamless deployment of DL-based architectures on a variety of low power devices.

In the following Sections, a description of the computationally optimised implementations, relative to the machine-learning based and optimisation based pose estimation and tracking algorithms, described in Section 2.1, is given.

2.2.1 Hardware-Specific Machine Learning Acceleration

Machine-Learning Approaches

Machine learning algorithms play a significant role in the domain of discriminative human pose estimation, especially support vector machines (SVM) and convolutional neural networks for RGB-based and decision forests for depth-based human body part detection. Hardware accelerated approaches of these methods are available in literature, both as general-use implementations and targeting specifically the problem of human detection.

Feature extraction and classification from RGB data Within the context of manually selected feature-based human recognition, plenty of approaches are available towards hardware-accelerated extraction and classification of features, suitable for human detection in RGB images. [Kadota et al., 2009] tackle the the issue of pedestrian detection, through the combination of HOG features and SVM classifiers, achieving real time performance on VGA

video, utilising in parallel 10 FPGA devices. The HOG/SVM architecture is also utilised in [Bauer et al., 2010], where an FPGA-GPU pipeline is proposed, while in [Negi et al., 2011] a deep pipelined one-chip FPGA implementation is introduced, using an Adaboost classifier instead of SVMs, and achieving 7.5 times faster throughput compared to a generic software implementation. Meanwhile, [Hiromoto and Miyamoto, 2009] present an FPGA implementation for the estimation of the computationally expensive co-occurrence HOG [Watanabe et al., 2009] features, reporting significant performance gains.

In [Papadonikolakis et al., 2009] a performance comparison of GPU and FPGA architectures for the SVM training problem is presented, focusing on the exploitation of Gilbert’s algorithm [Martin, 2005] parallelisation potential, in order to efficiently utilise the available hardware resources, while the authors in [Papadonikolakis and Bouganis, 2010] propose a fully scalable heterogeneous FPGA architecture for the acceleration of the SVM classification. The issue of accelerating the training of nonlinear SVMs is investigated in [Rabieah and Bouganis, 2015, Rabieah and Bouganis, 2016], while [Cadambi et al., 2009] tackle both the SVM training and classification tasks, introducing a massively parallel FPGA-based co-processor, with the proposed implementation achieving a 20-fold increase in speed compared to a state-of-the-art CPU. Meanwhile, a series of progressively faster GPU-accelerated implementations for SVM training and classification [Catanzaro et al., 2008, Carpenter, 2009, Herrero-Lopez et al., 2010, Athanasopoulos et al., 2011] are described in [Drozda and Sopyła, 2016].

Feature extraction and classification from Depth data [Oberg et al., 2012] present an FPGA-based implementation of the discriminative pose estimator in [Shotton et al., 2013], providing a Verilog implementation of the decision tree pixel classification algorithm (Forest Fire), along with novel algorithmic and architectural optimisations for processing randomised decision trees that account for the limitations of real-world memory systems. The final implementation, with data capture and image segmentation being performed on a host PC, achieves an average processing speed of 181fps for an image of two average sized adults. [Pittman et al., 2013] further extend [Oberg et al., 2012], by incorporating the image segmentation process into the hardware forest classifiers, leading to a fully-embedded realisation of the Microsoft Kinect

single-shot human pose estimation pipeline, without using powerful CPUs or GPUs, utilizing commercially available FPGAs.

An FPGA implementation of decision tree classification is presented in [Narayanan et al., 2007] with further optimisation achieved by reordering the computations and by using a bit-mapped data structure. [Barbareschi et al., 2015] present a novel approach for an efficient hardware implementation of the majority voting combining rule, having a Decision Tree as a base classifier and a majority voting rule as a combiner. [Saqib et al., 2015] propose a pipelined architecture for the hardware implementation of axis-parallel binary decision tree classification that meets the state-of-the-art demands of increased throughput with minimal resource utilisation, while in [Mishina et al., 2015] “Boosted Random Forests” are introduced, utilising 47% less memory than conventional random forests when implemented on an FPGA board. Meanwhile, [Cheng and Bouganis, 2013] investigate the acceleration of the random forest training process using FPGAs.

[Sharp, 2008] describes a method for implementing the evaluation and training of general purpose decision trees and forests entirely on a GPU. The proposed method maps naturally to the GPU, exploiting its parallelism and cache, yielding a significant speed increase over a CPU version by around two orders of magnitude. [Marron et al., 2014] achieve even greater performance gains, within the more challenging context of evolving data streams, minimising the communication between CPU and GPU by building the trees directly inside the GPU.

Convolutional neural networks [Farabet et al., 2009] present an efficient implementation of convolutional networks on a single, low-end, DSP-oriented FPGA, exploiting the inherent parallelism of CNNs and taking full advantage of the multiple hardware multiply-accumulate units on the FPGA. Progressively faster frameworks for mapping Convolutional Neural Networks on FPGAs are presented in [Peemen et al., 2013, Zhang et al., 2015, Venieris and Bouganis, 2016]. In [Peemen et al., 2013] CNN applications are accelerated by reducing external data access with delicate data reuse, while in [Zhang et al., 2015] first the CNN’s computation and memory access are optimised, followed by the modelling of all the po-

tential designs in a “Roofline” model [Williams et al., 2009] and estimation of the best design for each layer. [Venieris and Bouganis, 2016] present the “fpgaConvNet” framework, proposing a methodology that captures CNNs by means of an analytical SDF model and introducing for the first time FPGA reconfiguration as a design option for CNN FPGA implementation. Meanwhile, G[Gupta et al., 2015] investigate the effect of limited numerical precision in deep learning, substituting floating-point units with fixed-point arithmetic circuits, and describe an FPGA-based hardware accelerator for matrix-matrix multiplication.

Regarding the GPU-based acceleration of CNNs, the extremely large amount of research work utilising CNNs, in the last few years, has led to the development of multiple frameworks allowing for the easy highly optimised implementation of such networks on modern GPUs. The most common used frameworks include Tensorflow [Abadi et al., 2016], Caffee [Jia et al., 2014], Torch [Collobert et al., 2002] and Theano [Bergstra et al., 2010], with all of the latest implementations utilising Nvidia’s cuDNN [Chetlur et al., 2014], a library of efficient implementations of deep learning primitives that provides reliable performance across a wide range of input sizes, and takes advantage of highly-optimised matrix multiplication routines to achieve high performance, without requiring any auxiliary memory.

Optimisation Approaches

Generative human pose tracking algorithms estimate the human pose utilising iterative optimisation algorithms (i.e. Newton’s method, EM etc.). To this end, various hardware-accelerated optimisation approaches have been proposed, although not specifically in the context of human pose tracking.

Iterative closest point Towards 3D point cloud registration through the Iterative Closest Point (ICP) [Besl et al., 1992] algorithm, commonly employed when trying to find the closest points between two datasets with respect to a distance measure, [Qiu et al., 2009] present a GPU-accelerated implementation of the Nearest Neighbor Search (NNS) algorithm, using a fixed length queue to search for the nearest point. [Wu et al., 2015a] further enhance the effi-

ciency of GPU-NNS by proposing an improved priority search method using a search stack and adopting the trim optimisation approach for efficient search. In [Park et al., 2011] a GPU-accelerated implementation of the point-to-plane ICP [Chen and Medioni, 1992] algorithm is proposed, while [Belshaw and Greenspan, 2009] present an FPGA implementation of ICP within the context of object tracking from range data.

Iterative gradient-based solvers A considerable amount of effort has also been put towards the hardware-acceleration of iterative numerical algorithms, used in solving systems of linear and non-linear equations. The GPU-acceleration of the biconjugate gradient algorithm [Fletcher, 1976] is investigated by Garcia [Garcia, 2010], while highly optimised GPU-implementations of Jacobian-free and preconditioned Newton-Krylov [Knoll and Keyes, 2004] methods are presented in [Pethiyagoda et al., 2014, Serban et al., 2015]. Additionally, the quasi-newton L-BFGS algorithm [Liu and Nocedal, 1989] is implemented on GPU in [Fei et al., 2014, D’Amore et al., 2015].

[Xu and Ziavras, 2003] explore the use of iterative methods for solving linear systems of equations on FPGA-based machines, focusing on three widely used iterative algorithms: Newton-Raphson, Conjugate-Gradient and Chebyshev. In [Morris and Prasanna, 2005] an FPGA-based 64-bit floating-point version of the Jacobi iterative method is introduced, achieving a 2.7 speedup for multiple iterations. Moreover, an FPGA implementation of the conjugate gradient method is presented in [Roldao and Constantinides, 2010], with the authors introducing a parametrisable, deeply pipelined hardware design for solving systems of linear equations and reporting a speedup of 5x compared to the peak theoretical performance of a single CPU implementation. Towards the solution of nonlinear least squares problems, two highly-efficient FPGA implementations of the Gauss-Newton [Wedderburn, 1974] and Levenberg-Marquardt [Moré, 1978] algorithms are presented in [Abba et al., 2009] and [Abba et al., 2010] respectively.

Expectation-maximization [Kumar et al., 2009] present a parallel implementation of the iterative Expectation Maximisation algorithm [Dempster et al., 1977] for finding maximum like-

likelihood estimates of parameters of Gaussian mixture models (EM-GMM), similar to the data association model used in [Ye and Yang, 2014] for human pose tracking, designed for the multi-core architecture of GPUs, with the authors reporting a speedup of 164 times in comparison to a single threaded CPU implementation. The hardware acceleration of EM-GMM algorithm is also explored in [Guo et al., 2012], where a FPGA-based pipeline-friendly variant of EM-GMM is implemented, achieving a maximum of 517 times speedup over a CPU-based solution, and 28 times speedup over a GPU-based solution.

Stochastic gradient descent Stochastic gradient descent (SGD) is a gradient descent optimisation strategy used to minimise certain kinds of objective functions, notably used as part of the back-propagation algorithm for training neural networks [Bottou, 1991], which has received plenty of attention from the research community due to the large utilisation of neural networks in a variety of research fields. While SGD, initially, was deemed unsuitable for GPUs [Dean et al., 2012] due to the need for fine-grained synchronisation, which can be expensive, efficient asynchronous approaches have been proposed towards its parallelisation. [Zhang et al., 2013a] and [Paine et al., 2013] both present their own versions of asynchronous SGD exploiting model and data parallelism to speed up neural network training. [Kaleem et al., 2015] examine several synchronisation strategies for SGD on GPUs, proposing a dynamically scheduled implementation, while the authors in [De Sa et al., 2017] study the performance of the low-precision A-SGD variant, introducing a new conceptual framework for classifying precision and presenting design features to enhance performance on FPGAs.

2.2.2 Hardware-Agnostic Deep Learning Acceleration

Deep Learning is widely used towards human pose estimation from both RGB and depth data, significantly outperforming earlier non-DL approaches. This immense increase in accuracy, however, has come at the cost of increased computational complexity [Justus et al., 2018]. The complexity of deep learning models has been steadily growing, increasing the required computational resources during both training and inference. As a result many approaches have

been proposed attempting to reduce the complexity of deep learning models by either shrinking the models or minimising the overall cost of computations (Figure 2.2). In contrast to the hardware-specific implementations described in Section 2.2.1 above, the approaches presented herein provide more generic, framework-agnostic acceleration strategies.

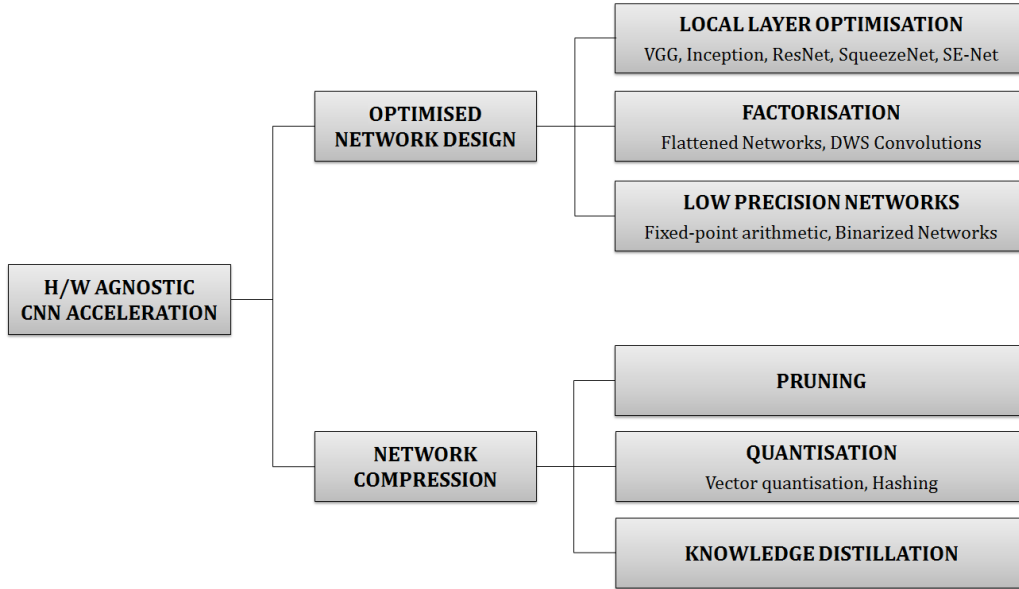


Figure 2.2: Block diagram of the taxonomy of hardware-agnostic techniques for Deep Learning acceleration

Optimised Network Design

Multiple novel architectures have been proposed trying to improve the performance of earlier CNNs [Krizhevsky et al., 2012], by introducing updated designs which allow them to maintain a comparable level of accuracy while reducing the computational cost.

Local Layer Optimisation [Simonyan and Zisserman, 2015] replace the large convolutional filters of AlexNet [Krizhevsky et al., 2012] with multiple small 3×3 filters of a similar receptive field size, while [Szegedy et al., 2015, Szegedy et al., 2016, Szegedy et al., 2017] decompose the 3×3 filters into two consecutive 1×3 and 3×1 filters. [He et al., 2016] introduce the bottleneck block, employing 1×1 filters to reduce and then expand the dimensionality of the feature space, while [Xie et al., 2017] extend [He et al., 2016] by utilising group convolutions inside the bottleneck block (Figure 2.3). A bottleneck approach is also proposed in SqueezeNet

[Iandola et al., 2016] where 1×1 filters “squeeze” the network, followed by multiple 1×1 and 3×3 filters which expand it. Meanwhile, in [Hu et al., 2018] the authors introduce the Squeeze-and-Excitation (SE) blocks which adaptively recalibrate channel-wise feature responses by explicitly modelling interdependencies between channels.

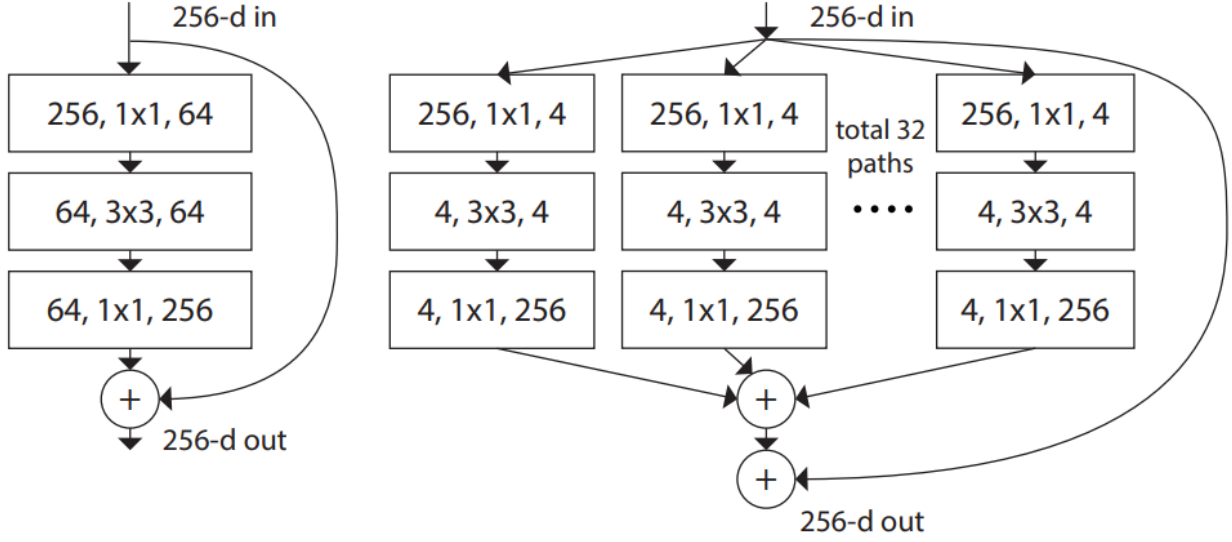


Figure 2.3: Block diagram of the ResNet [He et al., 2016] and ResNeXt [Xie et al., 2017] bottleneck units (image taken from [Xie et al., 2017])

Factorisation techniques factorise convolutional layers, by decomposing standard convolutions. [Jin et al., 2014] separate convolutions into consecutive 1D filters, across the channel, horizontal and vertical direction, while [Wang et al., 2017] unravel standard convolutions into individual 2D spatial convolutions which are performed sequentially, followed by a linear channel projection across all channels.

A special form of factorized convolutions are DepthWise Separable (DWS) convolutions [Sifre and Mallat, 2014] which consist of a spatial convolution performed independently over each channel of an input (depthwise), followed by a 1×1 convolution (pointwise), projecting the channel outputs of the depthwise convolution onto a new channel space. Xception [Chollet, 2017] and Mobilenets [Howard et al., 2017] utilize DWS convolutions in order to generate extremely efficient models for mobile devices. Shufflenets [Zhang et al., 2018] employ DWS convolutions inside residual bottleneck units, along with group pointwise convolutions and channel shuffle, while [Ma et al., 2018] argue that maintaining the feature dimensionality along

the bottleneck unit, through channel splitting, improves the performance of [Zhang et al., 2018] in practice. Meanwhile, an inverted residual bottleneck unit with DWS convolutions is proposed in the 2nd version of Mobilenets [Sandler et al., 2018].

Low Precision Networks employ lower bit-depth representations for CNN weights and/or activations during both training and inference. [Courbariaux et al., 2014] investigate the use of low-precision multiplication for training neural networks. [Gupta et al., 2015] and [Zhou et al., 2016a] employ different bitwidths for weights, activations and gradients, while in [Wu et al., 2016, Jacob et al., 2018] unified quantised frameworks for fixed-point training and inference are proposed. [Zhou et al., 2017] further push the parameters representation to less than 8 bits, while in [Hubara et al., 2017, Li et al., 2016] 2-bit representations are explored.

An extreme version of low precision networks are Binarized Networks, where activations and network weights are binary [Kim and Smaragdis, 2016, Hubara et al., 2016, Rastegari et al., 2016, Bulat and Tzimiropoulos, 2019], thus allowing the the binary convolution to be efficiently implemented using the XNOR gate.

Network Compression

Network compression strategies aim to accelerate inference while preserving the accuracy of a pretrained model, by reducing the number of parameters and layers.

Pruning reduces the number of parameters and operations in pretrained CNNs by dropping less important connections [Hanson and Pratt, 1989], enabling smaller networks to maintain comparable performance to larger predecessor networks. [Han et al., 2015a, Han et al., 2015b] introduce fine-grained sparsity in a network by a magnitude-based pruning approach where any weights with absolute magnitude under a threshold are pruned. [Luo et al., 2017] and [He et al., 2017] utilise statistics information computed from the next layer to guide a greedy layer-by-layer pruning, while in [Liu et al., 2017] a scaling factor is imposed during the training process to facilitate one channel-level pruning.

Quantisation strategies reduce the arithmetic precision of the pretrained model parameters. [Vanhoucke et al., 2011] explore a fixed-point implementation with 8-bit integer (vs 32-bit floating point) activations, while [Gong et al., 2014] compress deep convnets using vector quantisation. Meanwhile, HashNets employ a hash function to group connection weights into hash buckets [Chen et al., 2015, Spring and Shrivastava, 2017], with all connections within the same hash bucket sharing a single parameter value.

Knowledge Distillation is a framework that can compress a large deep network (teacher) into a smaller student network. [Hinton et al., 2015] introduce the student-teacher paradigm, in which the student is penalised according to a softened version of the teacher’s output. The framework compresses an ensemble of deep networks (teacher) into a student network of similar width, being trained to predict the output of the teacher, as well as the true classification labels. In [Romero et al., 2014] the student also learns intermediate representations of the teacher network, allowing the training of thinner student networks. [Yim et al., 2017] exploit flow between layers computed as the inner product of the corresponding feature maps, while [Polino et al., 2018] jointly leverage weight quantisation and distillation to train quantised thinner students. [Crowley et al., 2018] propose the usage of corresponding efficient group convolutions when designing the student network, and [Wang et al., 2018] employ adversarial learning to improve accuracy and training speed.

2.3 Summary

In this section, a survey of recent work in the fields of human pose estimation and their optimised implementations for low power systems, relevant to the problem which the current research addresses, has been presented. The survey is separated in two distinct parts. In the first part, a comprehensive description of current state-of-the-art human pose estimation algorithms is given, including classic handcrafted feature-based pose estimation from RGB data, discriminative and generative methods for estimation and tracking of the pose from depth and 3D data and recent estimation techniques for 2D and 3D poses which utilise deep convolutional neural

networks. In the second part, hardware-accelerated implementations of the algorithms relevant to the pose estimation and tracking methods are presented, targeting mainly the domain of GPUs and FPGAs, along with hardware-agnostic strategies for deep learning optimisation for low power devices.

Although the fields of human pose estimation and computational optimisation have been well studied separately, there has been little work on combining these two fields towards human pose estimation and tracking on systems with limited resources, as modern techniques often utilise high performance hardware, with little consideration for run-time efficiency. Moreover, specifically in the area of 3D pose estimation from sparse depth/3D data, research has been relatively limited, usually evaluated in laboratory datasets generated in constrained environments, thus presenting limitations when employed in real life unconstrained conditions.

In an attempt to fill the gaps in the research so far, in the field of human pose estimation, the present research aims at proposing novel methods and frameworks for human pose estimation from sparse 3D data on low power systems. In the following sections the relevant research work that has been carried out is presented, including a complete human template-based pose estimation and tracking framework, a deep learning-based approach utilising 3D CNNs for multi-person HPE and the optimisation process of the latter, in terms of computational complexity reduction, which has allowed its efficient deployment on modern low power mobile devices.

Chapter 3

Preliminary Concepts and Notations

This chapter presents basic concepts and notations used throughout the Thesis, including target application scenarios, depth imaging basics datasets and computational complexity measures.

*Parts of the contents of this chapter have appeared in the following peer-reviewed papers:
[Vasileiadis et al., 2017b, Vasileiadis et al., 2019a].*

In this chapter, some preliminary concepts and notations are presented, which are used throughout the rest of the Thesis, in order for the reader to be familiarised with the purpose of the methods proposed in the subsequent chapters. First, some application scenarios are presented, in order to better define the goals of this research. Next, an introduction to depth images, which are used in this research, is given, covering some of the basic aspects of the format. Finally, Sections 3.3 and 3.4 describe in detail the datasets and measures which are used for the performance evaluation and training of the proposed methods.

3.1 Application Scenarios

The results of this research are expected to be used within the context of applications where the estimation of the human pose is necessary towards the extraction of information, either direct

or inferred, relevant to the monitored person, while implemented on hardware with limited resources. Specifically, the following scenarios were kept under consideration while carrying out this research:

- Human monitoring from assistive robotic platforms** A typical application scenario for human pose estimation from depth data on embedded systems, are applications of assistive living, utilising mobile robots. Robust human pose estimation and tracking plays an integral role in assistive service robot applications, as it provides information regarding the body pose and motion of the user in a scene, which are necessary for understanding the user's actions and activities, thus enhancing the assistive capacity of the robotic agent. Moreover, the efficient hardware-optimized implementation of such an algorithm, can contribute towards the robot's autonomy, as reduced computational load and installation size needs can increase the, usually limited, autonomous range of the robot. Additionally, the utilisation of depth data provides a detailed representation of the environment, even under extremely low illumination conditions, while the less privacy-invasive nature of depth images compared to RGB, makes the adoption of such systems easier for technologically illiterate older population, who are typically the target end-users of such systems and tend to feel uneasy having a "live" camera within their homes [Vasileiadis et al., 2016].
- Large-area monitoring through interconnected low-cost devices** Human-presence monitoring, within the context of security, occupancy modeling etc., is often performed through a network of overlapping visual sensors, covering large areas (i.e. offices, shops, airports etc.). Key to the wide adaptation of such monitoring systems is the availability of low cost devices capable of providing accurate human motion information, as the large amount of visual devices, needed to cover all the monitoring areas, can quickly add up to the total cost of the system. Since low cost usually correlates to low computational capabilities, human pose estimation and tracking on embedded devices can provide a practical solution, with depth/3D data tackling any low illumination issues, while simultaneously allowing for the easier and more accurate calibration of the overlapping visual sensors,

providing a real-life 3D grid of the monitored area.

- **Motion-based HCI on mobile applications** Mobile applications utilising motion-based HCI, within the context of mobile games and general gesture-based controls, have achieved great commercial success in the last few years. Developing a computationally efficient pose estimation implementation, allows the utilisation of such systems within the limited capacity of mobile processors and GPUs. Additionally, the fact that depth images are less affected by illumination changes than RGB images, even in outdoor scenes at short range, ensures robust performance in diverse lightning conditions.

The final developed 3D HPE module, described in Chapters 5 and 6, reliably addresses the first two scenarios. It can handle moving cameras, cluttered scenes and partial views, which are common in the human monitoring scenario, as well as non-frontal views and data from multicamera setups, related to the second scenario. Additionally its computationally efficient implementation (Chapter 6) enables it to achieve the necessary processing rate (>5 fps) on the low power devices typically used in such systems. Regarding the motion-based HCI on mobile devices, while none of the developed methods were trained/evaluated in these constrained settings (i.e hand tracking from very close distance), the overall architectures can be easily adapted, with small tweaks (training data, sampling resolution etc.), to handle the specific viewpoints of this scenario.

3.2 Depth Imaging

In the last few years, the commercial availability of consumer-grade depth sensors, such as the Microsoft Kinect, has shifted the focus of the research community towards the utilization of depth data for various computer vision tasks, including body part detection. Depth images, contrary to RGB images, are not affected by changes in lighting conditions, allowing them to work even in total darkness, and can provide a 3D representation of the scene, offering rich geometrical information about the humans present in it, especially in cases where there is overlapping of body parts on the 2D image plane (e.g. human facing towards the camera with

the arm extended to the front). Moreover, depth images allow for privacy-preserving human monitoring, as it is not possible for a human to recognize the identity of a monitored person simply using the visual information provided from a depth image.

On the other hand, it should be noted that depth imaging does present some limitations. Most consumer grade RGB-D cameras usually have a maximum range of $\sim 8m$, with an effective range of $\sim 6m$, as in larger distances the accuracy of the data begins to degrade. Moreover, some surfaces and materials cannot be captured by the depth sensors (i.e. curved surfaces, transparent or reflective materials etc.) and can add noise to the collected data. Similarly, direct exposure to intense sunlight has been reported to interfere with the data accuracy, for some depth sensor architectures. However, these limitations rarely affect the human pose tracking process, as the human body, due to its size, shape and texture, can be robustly captured by depth sensors, with the exception of human hair which can add some negligible sensor noise around the head area.

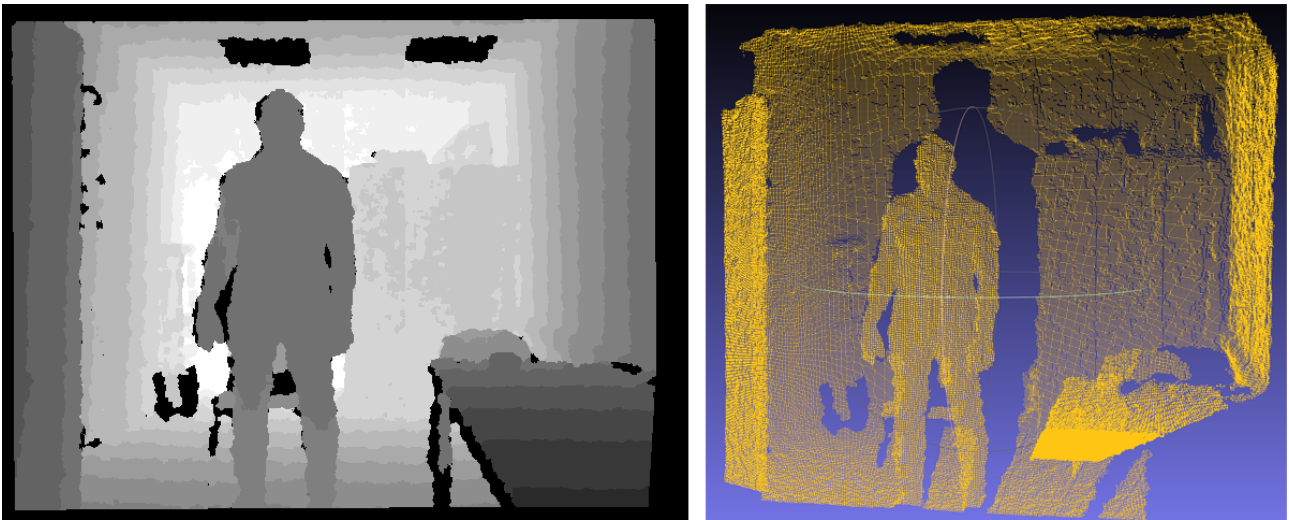


Figure 3.1: Left: Sample depth image from the Microsoft Kinect 1 camera (normalized for better visualization). Right: The corresponding 3D point cloud produced by back-projecting the depth image

Commercial depth sensors fall into two design architectures [Zanuttigh et al., 2016]:

- **Structured light depth cameras** use an IR projector to project a specific geometric pattern and an IR camera to capture the pattern projection on the surfaces. The disparities between corresponding points in the projected pattern and the captured pattern are

then calculated, allowing for the estimation of the distance between the sensor and the surface point (“depth”).

- **Time-of-flight depth cameras** utilize a transmitter, consisting of an array of laser or LED emitters, which emits light pulses in the the near infrared (NIR) range and a receiver which can be interpreted as a matrix of single ToF receivers. Each pixel of the receiver sensor matrix typically estimates the phase difference between the emitted and received light pulse, thus providing an estimation of the distance between the camera and the surface point where the incoming pulse was echoed from.

Both architectures provide a similar output: a single channel 2D image where each pixel includes the depth value of the corresponding surface point. Since the depth values correspond to real life measurements, it is possible to transform the 2D image into a 3D point cloud by back-projecting it using the pinhole camera model. For a pixel $\mathbf{p} = [u, v]^T$ with a depth value d , the 3D scene coordinates of the corresponding point $\mathbf{P} = [x, y, z]^T$ are:

$$\begin{aligned} x &= \frac{u - c_x}{f_x} d \\ y &= \frac{v - c_y}{f_y} d \\ z &= d \end{aligned} \tag{3.1}$$

where f_x , c_x and f_y , c_y the focal length and center of projection of the x and y axis respectively. Table 3.1 presents the technical specifications of the most common commercially available depth cameras

Table 3.1: Technical specifications of the most common commercially available depth cameras

Camera Model	Architecture	Output	Effective Range* [Gonzalez-Jorge et al., 2015]
Primesense Sensor			
Microsoft Kinect v1	Structured	320x240px	60cm - 600cm
ASUS Xtion	light	30fps, 16bit	
Occipital Structure Sensor			
Intel Realsense	Structured	640x480px	20cm - 120cm (F200)
	light	60fps, 16bit	50cm - 400cm (R200)
Microsoft Kinect v2	Time-of-flight	512x424px	50cm - 450cm
		30fps, 16bit	

*actual sensor range can be different

3.3 Datasets

Conforming to the research goals and applications scenarios presented above, the algorithms and methods developed in the current research aim at estimating the human pose from depth/3D data, in unconstrained realistic settings. Thus the datasets selected for experimentation, recognition model training and performance evaluation consist of depth images of humans in diverse poses recorded from various viewpoints, along with ground truth data for the human body parts and joints visible in each image.

The selection of the datasets used in this research is based on the following three criteria:

1. **Data availability:** The datasets should be publicly available and accessible online
2. **Data suitability:** The data should be suitable for the task in hand, including depth data with multiple poses of humans, with large diversity in human body shape and size,

as well as various capture viewpoints. Moreover, ground truth data regarding the 2D and 3D positions of the skeleton joints and body parts should be available for each frame

3. **Use in state-of-the-art methods:** The datasets should have been used in relevant state-of-the-art human pose estimation and tracking works, in order to allow for direct performance comparison of the proposed methods to other approaches

Based on the above criteria, five datasets with real data are selected for experimentation. Additionally, a new realistic dataset is captured, in order to provide a better insight to the performance of the proposed methods in unconstrained realistic environments. In the following, a description of each dataset is presented.

Stanford Time-of-Flight Motion Capture Dataset (SMMC-10)

The SMMC-10 dataset [Ganapathi et al., 2010a, Ganapathi et al., 2010b] has been recorded using a Mesa SwissRanger time-of-flight sensor. It includes 28 short recordings of a single person performing actions of gradually increasing complexity. The ToF sensor is placed approximately $\sim 2.5m$ away from the person. Joint ground truth data is captured using the Vicon motion capture system, providing the positions in 3D space of 14 joints: *head*, *torso*, *R/L shoulder*, *R/L elbow*, *R/L hand*, *R/L hip*, *R/L knee*, *R/L foot*. In total the SMMC-10 dataset contains approximately 8,000 3D point clouds, each one containing 25,344 points (corresponding to a 176x144px depth image).

EVAL Dataset (EVAL)

The EVAL dataset [Ganapathi et al., 2012b, Ganapathi et al., 2012a] has been recorded using a Microsoft Kinect v1 RGB-D camera. It includes 24 recordings of 3 different people performing actions of gradually increasing complexity. The Kinect camera is placed approximately $\sim 3m$ away from the person. Joint ground truth data is captured using the Vicon motion capture system, providing the positions in 3D space of 14 joints: *head*, *torso*, *R/L shoulder*, *R/L elbow*,

R/L hand, R/L hip, R/L knee, R/L foot. In total the EVAL dataset contains approximately 10,000 3D point clouds, each one containing 76,800 points (corresponding to a 320x240px depth image).

Personalized Depth Tracker Dataset (PDT)

The PDT dataset [Helten et al., 2013a, Helten et al., 2013b] has been recorded using a Microsoft Kinect v1 RGB-D camera. It includes 20 recordings of 5 different people (3 male, 2 female) performing actions of gradually increasing complexity. The Kinect camera is placed approximately $\sim 3.5m$ away from the person. Joint ground truth data is captured using the Phasespace active marker-based motion capture system, providing the positions in 3D space and the corresponding 2D projections on the image plane of 15 joints: *head, torso, neck, R/L shoulder, R/L elbow, R/L hand, R/L hip, R/L knee, R/L foot*. Additionally, the “ground-truth shape” is provided for each actor, obtained by fitting a synthetic human model to full-body laser-scans of the actors. In total the PDT dataset contains approximately 27,000 depth images, sized 640x480px.

Invariant Top View Dataset (ITOP)

The ITOP dataset [Haque et al., 2016b, Haque et al., 2016a] has been recorded using two Asus Xtion PRO RGB-D cameras. It includes recordings of 20 people performing 15 action sequences each. One camera is recording from a traditional front-facing viewpoint, from an average distance of $\sim 3m$, while the second camera is placed on the ceiling facing down, offering an extremely challenging viewpoint. Joint ground truth data is captured through a three-step semi-automatic procedure. First, 3D joints are estimated using [Shotton et al., 2013] from the front-facing camera. Second, an iterative ground truth error correction technique is used, based on per-pixel labeling using k-nearest neighbors and center of mass convergence. Finally, humans manually validate, correct and discard any noisy frames. The real world 3D and image plane 2D ground truth positions for 15 joints, from both viewpoints, are provided: *head, torso, neck,*

R/L shoulder, R/L elbow, R/L hand, R/L hip, R/L knee, R/L foot. In total the ITOP dataset contains approximately 100,000 depth images, sized 320x240px.

CMU PanopticStudio 3D PointCloud Dataset (CMU PtCloud)

The CMU PanopticStudio 3D PointCloud Dataset [Joo et al., 2019b, Joo et al., 2019a] includes approximately 6 hours of recordings split into 54 single and multi-person sequences. Each sequence is captured through a network of 10 overlapping Microsoft Kinect v2 RGB-D cameras, uniformly distributed around a geodesic dome with a radius $r = 5m$, providing the full 3D point cloud of the scene. A novel multi-view full body motion reconstruction algorithm [Joo et al., 2018] is used to provide ground truth data for 15 joints: *head, neck, R/L shoulder, R/L elbow, R/L hand, hip center, R/L hip, R/L knee, R/L foot*. In total the CMU PtCloud dataset contains approximately 540,000 3D point clouds, each one containing 2,170,880 points (corresponding to 10 512x424px depth images).

Realistic Human Motion Dataset

In order to evaluate the overall performance and robustness of the proposed framework in real-life monitoring conditions, the Realistic Human Motion dataset [Vasileiadis et al., 2017b, Vasileiadis et al., 2017a] is captured, since the public datasets described above were recorded in controlled laboratory environments and lack in challenges encountered in unrestrained real-life settings.

The dataset is captured using a Microsoft Kinect v1 RGB-D camera positioned on-board of a service robot at height $h \approx 1.2m$, simulating scenarios where the subject's domestic activities are monitored by an assistive service robot. In total 11 subjects (10 male, 1 female) perform a series of everyday actions in a single 1.5-minute long sequence including: interaction with large (i.e cupboard door) and small (i.e. pillbox) objects, sitting on a chair, eating while occluded behind a table, moving around the room near the camera's FOV limits or extremely close/far from the camera etc. This resulting human motion dataset incorporates, in contrast

to currently available public datasets, many of the challenges encountered in real life scenarios, such as: occlusions due to the presence of obstacles, out-of-FOV clipping, noisy observations due to object handling, decreased sensor accuracy due to long camera/subject distance, long-duration side-view tracking, non-frontal complex initialization poses, large variety in human body shape and more, creating a realistic benchmark for the evaluation of human pose estimation techniques in real-life settings.

Ground truth annotation is performed manually on the recorded data, by selecting and 3D projecting 9 skeleton joints: *head*, *R/L elbow*, *R/L hand*, *R/L knee*, *R/L ankle*. The selection of the joints is based on ensuring consistency among consecutive frames (i.e. a *torso* joint would be hard to track consistently), while annotation is performed every 10 frames and includes only joints visible in the specific frame. In total the Realistic Human Motion Dataset contains approximately 23,000 depth images, sized 640x480px. Figure 3.2 shows some sample frames from the realistic human motion dataset.



Figure 3.2: Sample frames from the realistic human motion dataset, with manually annotated ground truth data.

3.4 Computational Complexity Metrics

In order to evaluate the computational complexity of a specific HPE method it is necessary to establish an absolute metric that will describe the computational cost of an architecture and provide an estimation about its runtime. While actual runtime measurements could be used

as such a metric, they cannot be considered an absolute metric as their values depend on the platform upon which the algorithm is benchmarked, presenting significant variations between different hardware and software configurations.

3.4.1 Framework-Agnostic Metrics

The task in hand requires framework agnostic metrics. Towards this end, the following metrics are introduced:

Floating Point Operations (FLOPs): FLOPs¹ describe the total number of floating-point operations (additions, multiplications etc.) required to perform a function. Many recent works, especially in the scope of efficient CNN design, have resorted to assigning a single FLOP to every arithmetic operation. However, in practise complex arithmetic operations, such as exponential and sine, can be much slower than addition or multiplication [Dawson,]. While this simplification does not affect classification-based approaches (random trees, CNNs, SVMs etc.) where the bulk of the operations are multiplications and additions², it can significantly distort FLOP count for generative pose estimation techniques (see Section 2.1.4), where estimating the optimal pose vector for an articulated template requires a great amount of slower trigonometric operations.

Based on these observations and in order to use the metric meaningfully, FLOPs are assigned to individual operations according to Table 3.2 below [Huss and Pennline, 1987, Chimmalgi et al., 2019]. It should be noted that these values are only rules of thumb and can vary widely across programming languages and architectures.

¹Not to be confused with FLOPS as in Floating point operations per second

²It is common in literature to consider a float multiplication followed by addition as a single float operation, called MAC (multiply-accumulate operation). In this case $1MAC \approx 2FLOPs$

Table 3.2: FLOPs per operation [Huss and Pennline, 1987, Chimmalgi et al., 2019]

Operation	FLOPs
add, sub, mult, max	1
div, sqrt	4
exp, log, sin, cos, tan, arctan	8

Memory Access Operations (MEMs): MEMs [Zhang et al., 2018] describe the amount of memory access operations required in order to read all the data necessary for the computation and write its result. Following, for simplicity, a naïve approach, where each value has to be read and written every time (disregarding advanced techniques such as caching, memory sharing etc.), there are three memory access operation groups: a) read the input data, b) read the model parameters (i.e. pose-vector parameters or filter weights), c) write the output. Throughout the document, MEMs are counted without taking into consideration the actual size in bytes of the accessed data³, since a 32bit float data representation is adopted across all proposed methods, and the majority of the state-of-the-art.

Stored Parameters: The total storage requirements of an architecture are estimated by counting the amount of data necessary to save and restore its state. In the case of discriminative HPE methods this accounts for the number of trained variables of the classifier, while for generative trackers it may consider the data of the human template. The number of stored parameters not only dictates the minimum amount of storage required to save an architecture, but can also affect its computational complexity (more parameters usually lead to more FLOPs and MEMs). Similar to the MEM metric described above, the parameters are counted without taking into consideration the actual size in bytes.

³Reading an 16bit integer value and a 32bit float value, both count as a single MEM

It should be noted that the above metrics, while estimating the theoretical computational cost, can only provide an indication about the actual computational cost and how it translates into real life performance. Various issues such as memory bandwidth, parallel processing capabilities, software optimisation etc. can affect the overall runtime of an HPE method. They do provide, however, a framework agnostic comparison metric, which can direct the design process towards more efficient implementations; an architecture with a smaller theoretical computational cost should be faster (or at least as fast) than one with a higher cost, when benchmarked on the same device.

3.4.2 Metrics Calculation in CNNs

The exact mathematical formulas to calculate the metrics described above, can vary greatly depending on the machine learning frameworks used in each architecture. This is even more evident in the case of generative pose trackers, where a large variety of custom model representation and optimisation approaches are proposed.

On more recent Deep Learning approaches, on the other hand, it has become easier to define these formulas, as CNNs are composed of individual layers (fully connected, convolutional, pooling, activation etc.) shared among different architectures. The exact formulas for the calculation of computational complexity metrics for the most common layers in CNNs are presented in Table 3.3 below [Suzuo, 2018].

Table 3.3: Calculation of the computational complexity metrics for common CNN layers. D is the input's spatial dimension per axis, n is the number of fully connected nodes, C_{in}, C_{out} are the input and output channels, k is the kernel size, g is the number of convolution groups, s is the stride and d is the input dimensionality (1 for 1D, 2 for 2D, 3 for 3D). For simplicity we consider that the input's spatial dimensions, kernel size and stride are equal across all axes.

Layer	FLOPs	MEMs	Parameters
Fully Connected	$2 \cdot D \cdot n$	$D + n \cdot (D + 1) + n$	$n \cdot (D + 1)$
Convolution	$2 \cdot \frac{D^d \cdot C_{in} \cdot k^d \cdot C_{out}}{g \cdot s^d}$	$D^d \cdot C_{in} +$ $C_{out} \cdot (\frac{C_{in} \cdot k^d}{g} + 1) +$ $D^d \cdot C_{out} / s^d$	$C_{out} \cdot (\frac{C_{in} \cdot k^d}{g} + 1)$
Max Pooling	$\frac{D^d \cdot C_{in} \cdot (k^d - 1)}{s^d}$	$D^d \cdot C_{in} +$ $D^d \cdot C_{in} / s^d$	n/a
Avg Pooling	$\frac{D^d \cdot C_{in} \cdot (k^d + 3)}{s^d}$	$D^d \cdot C_{in} +$ $D^d \cdot C_{in} / s^d$	n/a
ReLU	$D^d \cdot C_{in}$	$2 \cdot D^d \cdot C_{in}$	n/a
Leaky ReLU	$2 \cdot D^d \cdot C_{in}$	$2 \cdot D^d \cdot C_{in}$	n/a
Sigmoid	$14 \cdot D^d \cdot C_{in}$	$2 \cdot D^d \cdot C_{in}$	n/a
Batch Norm	$2 \cdot D^d \cdot C_{in}$	$D^d \cdot C_{in} + 2 \cdot C_{in} + D^d \cdot C_{in}$	$2 \cdot C_{in}$

3.5 Summary

In this section, some preliminary concepts and notations, used throughout the Thesis, have been presented. Initially some application scenarios, within the context of which the results of this research are employed, are provided. Next the depth image format is described, along with the relevant HPE datasets. Finally, the computational complexity metrics used to evaluate the research results, are defined.

Chapter 4

Template-Based Human Pose Estimation and Tracking

This chapter presents a template-based real-time human pose estimation and tracking framework. The framework is based on the articulated 3D-SDF data representation model, and is extended with complementary mechanisms for addressing issues encountered in real-life unconstrained settings. Extensive evaluation on public datasets demonstrates the framework's state-of-the-art performance, while experimental results on a challenging realistic human motion dataset exhibit its robustness in real life scenarios.

*Parts of the contents of this chapter have appeared in the following peer-reviewed papers:
[Vasileiadis et al., 2017b, Kostavelis et al., 2019].*

The purpose of this PhD research is the development of a software/hardware co-design scheme for human pose estimation from depth/3D data. In this context, a human-template based pose estimation and tracking framework has been developed to serve as a starting point for further experimentation.

The proposed framework employs a human-template based tracker utilising the articulated 3D-SDF data representation model, originally proposed by [Schmidt et al., 2015]. The base tracker

is extended with complementary mechanisms, introducing novel methods for handling tracking initialisation, human model customisation, partial-view body-part tracking and large object handling, as well as adapting earlier works to address the issues of free space violation, body part intersection [Taylor et al., 2012] and tracking failure [Wei et al., 2012]. The incorporation of all these individual modules into a single HPE framework, achieves robust performance when employed in both controlled laboratory environments and unconstrained real-life settings, leading to a novel complete standalone body tracking implementation suitable for deployment in real life applications that require human motion analysis.

Through various code optimisations, including partial implementation on CUDA, the framework achieves close to real-time performance (~ 20 fps) on a mid-range Nvidia GTX970 GPU, using an ASUS Xtion depth camera as a capturing device. Additionally, extensive evaluation on public datasets demonstrates the framework's state-of-the-art performance, while experimental results on a novel challenging realistic human motion dataset, specifically captured for the task in hand, exhibit its robustness in real life scenarios. The proposed framework's efficiency is further demonstrated through its implementation within the context of daily activities recognition with a robotic agent.

The developed human-template based articulated pose tracker, along with all the introduced customisations and enhancements, is described in detail in the following Sections. Sections 4.1 and 4.2 present the input data and the articulated human template that the framework utilises. In Section 4.3 the main tracking algorithm is described, while Section 4.7 presents the complementary tracking features. Section 4.8 describes the tracker initialisation and tracking failure procedures, while in Section 4.10 the issue of large object handling is tackled. In Section 4.11 the technical details of the developed implementation are presented, and finally Sections 4.12 and 4.13 include the extensive experimental evaluation of the proposed framework and details regarding its experimental deployment on an autonomous robotic platform.

4.1 Data Input and Output

The framework’s human pose tracker employs an articulated skinned human template to track the human pose in sequences of depth images, extracting the 3D positions of the skeletal joints.

When working offline on a dataset or a prerecorded sequence, the input is parsed from the data, while an initial pose, required for the initialisation of the human template, is recreated using the ground truth data for the first frame. During online live operation, the depth map of the scene is acquired using a depth camera. The corresponding 3D point cloud is then created by back projecting the depth map to real world using the predefined projection coefficients of the sensor.

The human/background segmentation and the estimation of the initial pose are currently handled by the Primesense NiTE middleware library [PrimeSense, 2013] which uses a pose estimation algorithm based on [Shotton et al., 2013]. However, in future versions it may be replaced by a custom-developed implementation, as the NiTE library is not open-source and since it is no longer actively developed, it may present compatibility issues when trying to implement it on different architectures.

4.2 Articulated Human Template

The articulated human template used in the proposed approach is created using the MakeHuman open source tool [The MakeHuman team, 2000] and includes a skeleton composed of 15 rigid body parts connected to each other in a kinematic tree through 10 joints, along with 3000 vertices which represent the shape of the human body. The vertices are rigged on the skeleton, each controlled by up to 4 body parts, in order to realistically deform the skin mesh based on the skeleton movement.

The template pose is described through a pose vector $\theta = [t_0, q_0, q_1, \dots, q_{10}]$ which includes the rotation of each body part $q_{1..10}$ relative to its parent, along with the global rotation q_0 and translation t_0 of the template with reference to the camera frame. For efficiency reasons

and to avoid the problem of gimbal lock [Alan and Mark, 1992], the relative rotations of each body part are represented in the pose vector using unit quaternions. Given a pose θ_0 , the transformation matrix $T_{i,0}$ of a body part i in relation to the camera frame can be defined recursively by composing the transforms in a chain:

$$T_{i,0}(\theta_0) = T_{prt(i),0}(\theta_0) T_i(\theta_0) \quad (4.1)$$

where T_i the transformation matrix of body part i in relation to its parent $prt(i)$.

Two shape factors are also introduced to allow the dynamic manipulation of the size and shape of the template, in order to easily customise it based on the body shape of the tracked human. Specifically, a global scale factor g which uniformly resizes the skeleton, to account for variations in height and limb length, and a width factor s which increases or decreases the distance between the skin vertices and the skeleton bones in order to fit on silhouettes of humans with variable body fat (Figure 4.1).

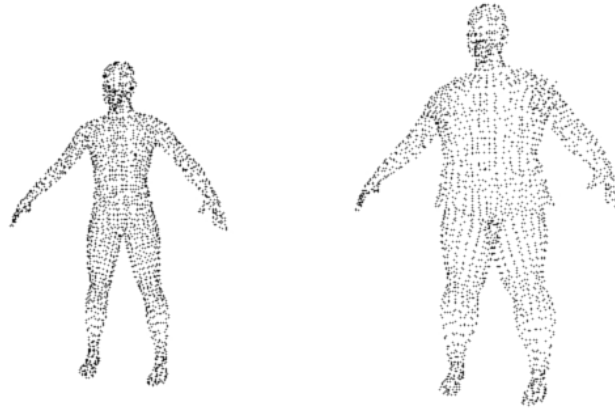


Figure 4.1: Human template customisation. Left: $[g=1, s=1]$, Right $[g=1.2, s=1.5]$

4.3 SDF-Based Tracker

The SDF-based human pose tracker attempts to minimise the sum of the distances between the 3D points of an observed point cloud and the 3D points of the mesh of the human template. In traditional ICP algorithms, this requires the explicit nearest point computation for each of the

input cloud points, which can become computationally expensive when using a detailed human template. However, based on the work of [Fitzgibbon, 2003], it is possible to remove the need for this taxing process by using implicit surface representations in the form of precomputed signed distance functions, which replaces nearest point computation with a much faster lookup in the SDF. This approach is further extended in [Schmidt et al., 2015] with the introduction of an articulated model signed distance function $SDF_{mod}(\mathbf{x}, \boldsymbol{\theta}) : R^3 \rightarrow R$, which defines for all $\mathbf{x} \in R^3$ the distance to the surface of a model when articulated according to the pose vector $\boldsymbol{\theta}$. Thus the pose estimation process is achieved by estimating the pose vector $\hat{\boldsymbol{\theta}}$ that minimises the tracker energy function:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} SDF_{model}(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{\mathbf{x} \in \Omega} SDF_{mod}(\mathbf{x}, \boldsymbol{\theta}) \quad (4.2)$$

where Ω the input 3D point cloud.

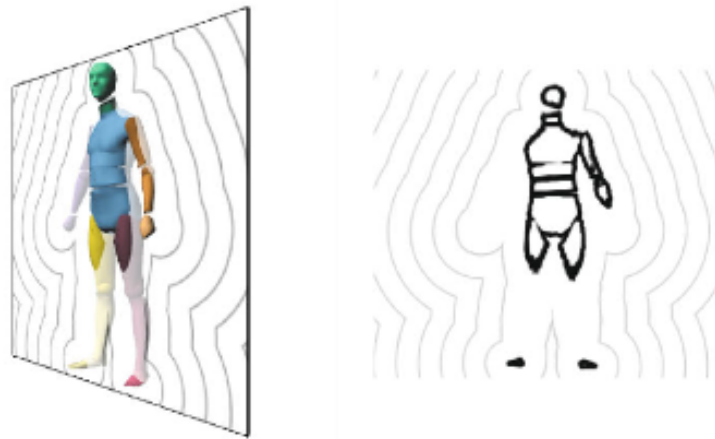


Figure 4.2: Representation of the human model in the tracker. Left: a planar slice through the human model with the isocontours of the SDF superimposed on the plane, Right: isocontours of the SDF, with the zero level set shown in bold (image taken from [Schmidt et al., 2015])

4.4 Model Representation

For each rigid body part i of the human template, the 3D mesh points rigged to it form a geometry defined implicitly by a signed distance function $SDF^i(\mathbf{x}, \boldsymbol{\theta}) : R^3 \rightarrow R$ [Felzenszwalb and Huttenlocher, 2004], which takes on negative values inside the geometry,

positive values outside, and has a zero value at the surface interface. This allows to approximate the global signed distance function $SDF_{mod}(\mathbf{x}, \boldsymbol{\theta})$, as a composition of the precomputed local signed distance functions $SDF^i(\mathbf{x}, \boldsymbol{\theta})$, which are not affected by the pose alterations, since the geometry they are defined from remains unaltered (Figure 4.3). As a result this alleviates the need to recompute the full global SDF every time the pose is updated.

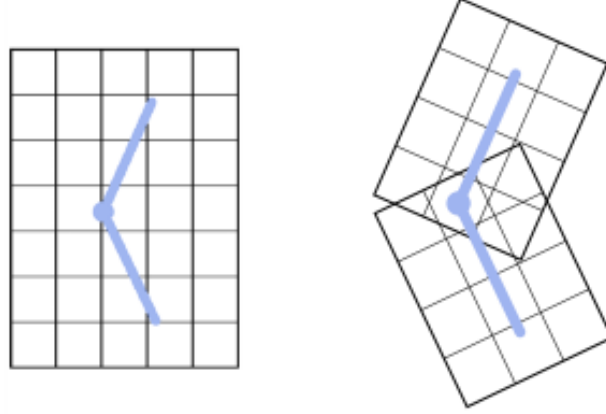


Figure 4.3: Left: global SDF, Right: composition of local SDFs (image taken from [Schmidt et al., 2015])

4.5 Data Association

In order to calculate the value of the energy function $SDF_{model}(\boldsymbol{\theta})$ for a given pose $\boldsymbol{\theta}_0$ the following steps are followed, similarly to [Schmidt et al., 2015]:

1. *Transformation of the observed point cloud to body part space:* All the points of the input point cloud are transformed to the local coordinates system of each rigid body part
2. *SDF lookup:* For each point of the input point cloud, the distance of the closest point on the surface of each body part is estimated by trilinear interpolation of the precomputed SDF of each body part
3. *Body part assignment:* Based on the distances estimated in the previous step, each point of the input cloud is assigned to a body part. If the distance is larger than d_{thresh} , the point

is discarded as an outlier. The sum of the distances of each point to its corresponding body part is the value of the energy function.

One of the main advantages of the described approach is that it is highly parallelisable, since every 3D point of the input point cloud can be processed independently, thus making it suitable for implementation on modern GPUs.

4.6 Optimisation

For the estimation of the optimal pose, a Quasi-Newton method is employed, using the BFGS algorithm [Nocedal and Wright, 2006], which in contrast to the Gauss-Newton algorithm used in [Schmidt et al., 2015], can minimise any general real-valued function $f(x)$ instead of only nonlinear least-squares problems.

A Quasi-Newton optimisation method uses an iterative approach to arrive at a minimal function value, similar to gradient descent or Newton’s method for optimisation. The Newton’s method, when used for optimisation, requires the Hessian of the function that is to be optimised, in order to estimate the direction of the linesearch algorithm. However, the Hessian of such a function can sometimes be extremely difficult to compute and may consist of unreasonably long expressions that require computationally intensive operations to calculate explicitly. A Quasi-Newton method, on the other hand, attempts to approximate the Hessian, allowing essentially to use Newton’s method without requiring the user to explicitly supply the Hessian.

As a result, the Hessian matrix of second derivatives is not evaluated directly, but is approximated using rank-one updates specified by gradient evaluations. The gradient values are calculated by estimating the data association error for each point, as described in Section 4.5 above, and explicitly computing the first derivatives according to the kinematic model structure and current pose estimate. The resulting pose step $\Delta\theta$ then updates the current pose estimate, and the algorithm iterates as needed until it meets a convergence criterion ($|\Delta\theta| \approx 0$), which means that the optimisation process has converged to a final pose vector, or reaches a

predefined iteration limit.

The use of the BFGS algorithm, or any similar optimisation technique in general, presents the risk of the optimisation process converging to a minimal function value that does not correspond to the actual human pose, due to the fact that the objective function is not strictly convex; besides the global minimum which corresponds to the correct pose, it also presents local minima corresponding to wrong poses. As a result, it becomes essential to initialise the optimisation process with a pose similar enough to the actual human pose, so that the minimal value “closest” to the starting position is actually the global minimum of the function.

4.7 Complementary Tracking Features

Besides the main tracking algorithm described in Section 4.3 above, a series of complementary tracking features are also employed herein, in order to increase the overall robustness and accuracy of the framework’s main tracking algorithm, as well as tackle specific problems that are typically encountered during online operation under real life recording conditions (Figure 4.4).

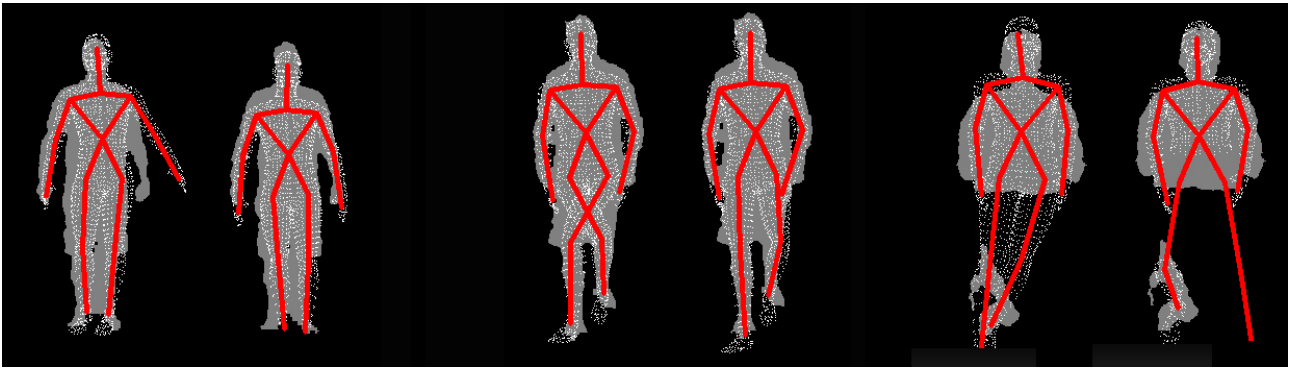


Figure 4.4: Pose correction through the complementary tracking features. Left: free space violation, Center: leg intersection, Right: body part visibility in occluded view; without the visibility feature both legs converge to the visible right leg, while with the feature enabled, the left occluded leg remains at the default position and is not rendered

4.7.1 Free Space Violation

Free space violation occurs when a part of the human template does not correspond to any of the input data and ends up “floating” in free space (Figure 4.4). To avoid this issue a 2D free space constraint is employed. Specifically, the 2D-SDF of the input depth image is computed [Felzenszwalb and Huttenlocher, 2004]. Next, the deformed template is projected on the image, with each vertex of the template contributing towards a free-space error $SDF_{fs}(\theta)$, based on the value of the corresponding pixel on the 2D-SDF image. As a result, the energy minimisation function (4.2) takes the form:

$$\begin{aligned} SDF_{overall}(\theta) &= SDF_{model}(\theta) + \lambda SDF_{fs}(\theta) \\ \hat{\theta} &= \underset{\theta}{\operatorname{argmin}} SDF_{overall}(\theta) \end{aligned} \tag{4.3}$$

where $\lambda \leq 1$ a weighing factor, helping the optimisation algorithm converge faster to a pose and reducing the number of necessary iterations.

4.7.2 Body Part Visibility

Commonly encountered in realistic environments is the occlusion of body parts due to the presence of obstacles and constraints of the camera’s FOV. In such cases only the visible body parts should be used in the optimisation process, as the non-visible body parts may incorrectly converge to the position of the closest visible body part (Figure 4.4).

To this end, a preprocessing step is introduced, which calculates the visibility of each body part before moving to the optimisation step. Specifically, the last tracked skeleton pose is projected on the camera’s image plane and any body parts that are outside the camera’s FOV or do not have valid depth observations in their surroundings pixels, are considered non-visible. For body parts without any children (lower arm, shin, head), visibility is determined by the position of that part’s endpoint, while for the rest of the body parts the midpoint is used. The non-visible body parts i are not taken into consideration during the optimisation process, by setting to zero the contribution of their corresponding 3D points to the energy function $SDF_{overall}(\theta)$:

$$SDF^i(\mathbf{x}, \boldsymbol{\theta}) = 0 \quad \mathbf{x} \in \Omega_i, \quad \Delta \mathbf{q}_i = 0 \quad (4.4)$$

where Ω_i the input points that correspond to part i and $\mathbf{q}_i$ the quaternion that describes its relative rotation in the pose vector $\boldsymbol{\theta}$. This results in keeping non-visible body parts in the last tracked pose, until a new valid depth observation appears.

4.7.3 Leg Intersection

During preliminary testing of the articulated tracker, it was noticed that there were instances where the tracker would misplace the legs of the human (Figure 4.4), mixing the left and right hips or shins, creating an unnatural pose, with the same behaviour noticed in the Kinect v1 pose estimator [Shotton et al., 2013] as well. This mix-up is mainly caused by two factors: (a) noisy observations between the two legs (due to long distance from the camera, or baggy clothes) making it hard to separate the legs and (b) quick turn-arounds of the human which may create a local minimum which “traps” the optimisation algorithm.

To counteract this issue, a leg intersection fix is employed. Using a body part representation similar to [Taylor et al., 2012], each of the four leg body parts (hip R/L and shin R/L) is approximated by 7 spheres s , equally spaced along the body part, with centres $\mathbf{c}_s$ on the body part’s bone and radii r_s small enough so that the spheres lie within the corresponding body part geometry. For a given pose $\boldsymbol{\theta}$, intersection between spheres s, t occurs when $\|\mathbf{c}_s(\boldsymbol{\theta}) - \mathbf{c}_t(\boldsymbol{\theta})\| < r_s + r_t$, thus the intersection error is defined as:

$$E_{intr}(\boldsymbol{\theta}) = \sum_{(s,t) \in P} \frac{1}{1 + e^{-(r_s + r_t - \|\mathbf{c}_s(\boldsymbol{\theta}) - \mathbf{c}_t(\boldsymbol{\theta})\|)\gamma}} \quad (4.5)$$

where P is the set of pairs of spheres and γ a normalisation factor.

In contrast to [Taylor et al., 2012], where the intersection error is incorporated within the optimisation process, the leg intersection correction is executed at post-processing, taking advantage of the fact that only two subcases need to be taken into consideration (mix up between R/L

hip or R/L shin). Specifically, if the error for the current pose is found larger than a threshold E_I , it is recalculated for two more poses: a) R/L knees interchanged and b) R/L ankles interchanged, with the pose that produces the minimum error retained as the optimal pose. Moving the intersection fix at post-processing ensures that any potential leg intersections will be fixed and removes the need to recalculate the error in each optimisation iteration.

4.8 Framework Initialisation

As noted in Section 4.6, the use of the BFGS algorithm, as with any iterative optimisation technique, presents the risk of the optimisation process converging to a minimal function value that does not correspond to the actual human pose, due to the fact that the objective function is not convex and can present local minima. As a result, it becomes essential that the tracking process begins from a pose close enough to the actual pose of the human. To this end, a two-step initialisation process is introduced herein, using as input the human pose estimation provided by the Kinect v1 built-in skeleton tracker [Shotton et al., 2013] (Figure 4.5), in order to initialise the human template in the first frame of a tracking sequence.

The first step concerns the calculation of the scale, global rotation and translation of the human template related to the camera frame. This is achieved by estimating the rigid transformation between the chest area of the template (formed by the chest and shoulders joints of the human template) and the corresponding area of the observed human, estimated from the Kinect v1 skeleton tracker.

The second step concerns the initialisation of each of the human template's body parts. If the actual human pose is close to the default template pose (standing, with the hands falling naturally by his side), then this step can be omitted. However, in more complex initial poses, it is essential to correctly initialise each body part, so as to allow the articulated tracker to converge to the correct pose, and not get trapped to a local minimum. Similarly to the main optimisation process of the pose tracker (Section 4.6), the initial pose θ_{init} is estimated by minimising, using the BFGS solver, the initialisation energy function which is defined as the

sum of the distances between the corresponding joints of the template and the target skeleton provided by the Kinect v1 skeleton tracker:

$$\theta_{init} = \underset{\theta}{\operatorname{argmin}} \sum_i \| \mathbf{j}t_i(\theta) - \mathbf{j}t_{i,init} \| \quad (4.6)$$

where $\mathbf{j}t_i(\theta)$ and $\mathbf{j}t_{i,init}$ are the 3D positions of the joint i of the template and the initialisation skeleton respectively.

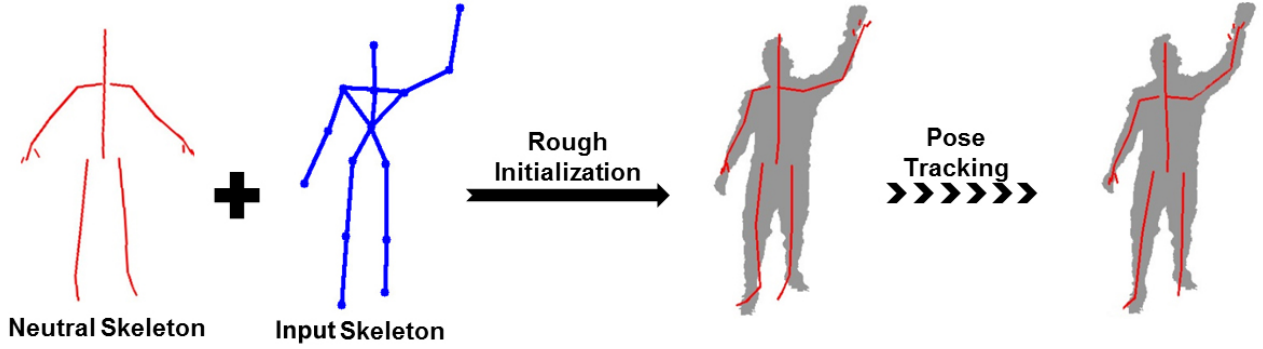


Figure 4.5: The initialisation pipeline of the human pose tracker

4.9 Tracking Failure Detection

After the conclusion of the initialisation process, the pose tracker functions independently, using as an initial pose the pose estimated in the last frame. In order to ensure that the last tracked pose is indeed correct and the optimisation process did not converge to a wrong human pose, a metric is employed, inspired by [Wei et al., 2012], to monitor the quality of the tracking process. The tracking confidence is estimated by projecting the deformed human template on the camera's image plane, taking into consideration only the visible body parts, and calculating its overlay ratio in respect to the input image. If the confidence score falls below a threshold, the estimated pose is considered incorrect and the tracker is re-initialised following the initialisation procedure described above. An example of successful and failed pose estimation is presented in Figure 4.6.

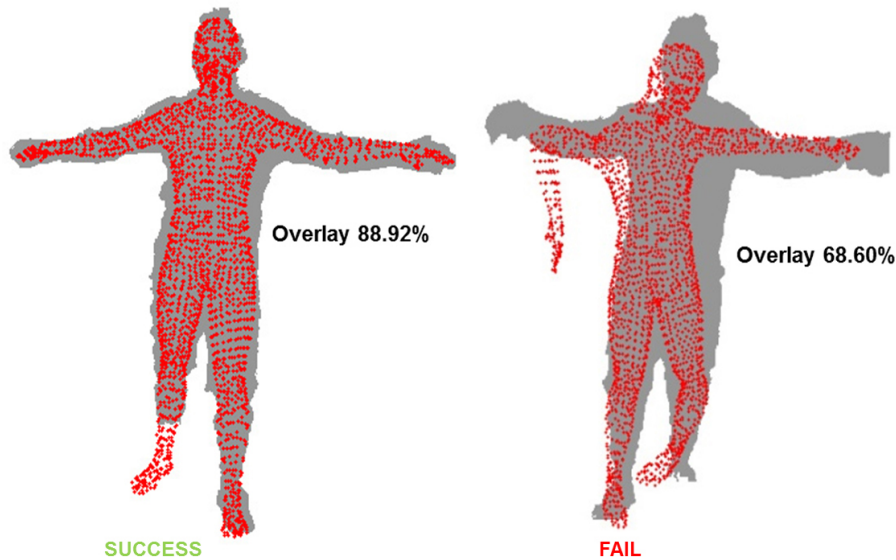


Figure 4.6: High (left) and low (right) confidence pose tracking. Deformed human template in red, input human silhouette in grey

4.10 Large Object Handling

Handling of objects by the monitored human is a very common scenario in realistic environments. In the case of small handheld objects, such as a cup, it does not significantly impact the accuracy of the pose tracking algorithm. However, when large objects, such as a cupboard door, are incorporated inside the foreground data that is passed to the algorithm, the optimisation process can lead to erroneous results, as the optimiser tries to match the human template to the input human and the large object simultaneously, thus making it necessary to remove any non-human data from the input.

Towards this end, a data preprocessing step is introduced. Specifically, the last successfully tracked human silhouette, along with a small buffer zone, is projected on the new input depth image. Any large areas that do not overlap with the human silhouette are considered as candidate large objects. Taking advantage of the fact that large objects tend to be smooth surfaces (i.e. doors, tables etc), the flood-fill algorithm is used on the depth image, seeded from the centre point of each candidate area, in order to remove any large objects in the scene. The threshold value for the flood-fill algorithm is set to a relatively low value to ensure that no

parts of the actual human will be removed, resulting in small parts of the object being left in the scene as well, without, however, significantly impacting the algorithm's accuracy anymore. Figure 4.7 presents an example of the large object removal process.



Figure 4.7: Data preprocessing for large object removal. The input image is split in three regions: blue - overlap with the last tracked human silhouette, red - buffer zone, green - candidate large object area which provides the seeds for the flood-fill algorithm. The image on the right is the processed image passed to the optimiser

4.11 Implementation Details

The developed articulated pose tracker implementation uses C++ and CUDA code. Computationally inexpensive processes (i.e. cloud subsampling, 2D-to-3D projection, bone transformation estimation, joint visibility estimation etc.) are executed by the CPU utilising open source libraries (OpenCV, Eigen etc.), while time consuming processes, mainly related to the optimisation process, are executed by custom-made CUDA kernels. The hardware/software splitting scheme is further elaborated in the following sections, while the block diagram of the developed pose tracking algorithm is presented in Figure 4.8.

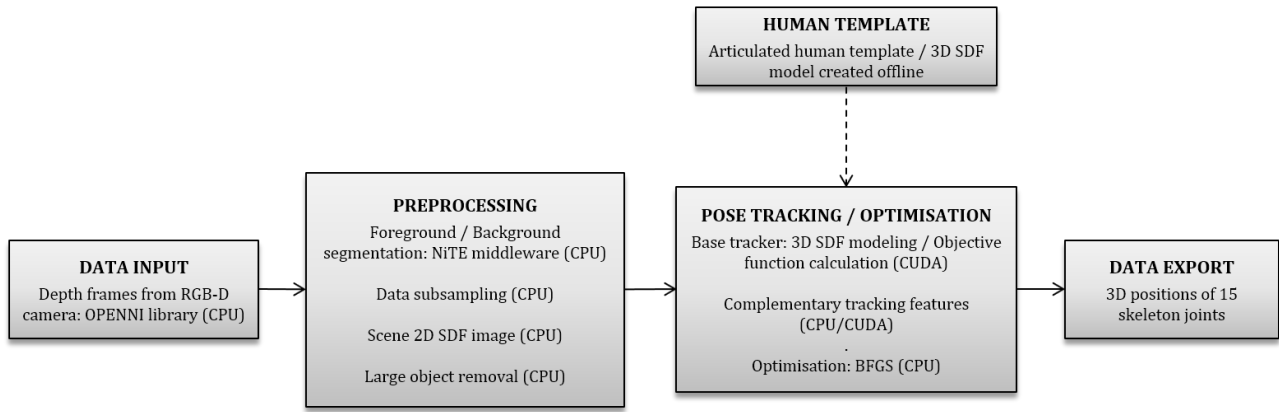


Figure 4.8: Block diagram of the developed human pose tracking algorithm

The specific design choices for the various components of the pose tracking framework described below were made through small-scale preliminary testing on subsets of the PDT and EVAL datasets. The goal of this experimental approach was to achieve the best possible pose estimation accuracy, while maintaining a high enough framerate, so that the framework would be able to track the pose both “offline”, where the frames would be fed to the system sequentially one after the other, and in real-time, with frames arriving as soon as they are available.

As discussed in Chapter 2.1.1, if there is a long delay between frames, the large body movements in consecutive observations can potentially trap the optimisation process in local minima and fail to recover properly, thus a fast enough processing rate is necessary. This framerate threshold was experimentally found to be 10fps, although in some scenes at least 15fps were needed in order for tracking not to fail.

4.11.1 Input Data Preprocessing

During online operation, input data is provided from an ASUS Xtion structured-light depth camera, which captures a 640x480px depth map of the scene at 30fps. After foreground / background segmentation, the resulting human 3D point cloud is produced; for a typical full body frontal view at a distance of $\sim 2.5m$ the point cloud contains around 20000 3D points. Next, the point cloud is uniformly subsampled, using a regular grid in 3-dimensional space with voxel size $d = 0.025m$ (a voxel is the 3D equivalent of a 2D pixel, representing a volume

in 3D space). This specific value was selected through preliminary experimental testing, as it was noticed that there wasn't any consistent increase in accuracy for finer resolutions (Figure 8), while for lower resolutions, although the accuracy achieved in laboratory recorded datasets was similar, under challenging realistic conditions the tracking robustness did degrade. The resulting 3D point cloud that is inputted in the pose tracker contains 1500-3000 3D points depending on the point of view of the camera and the human pose. Moreover, the 2D SDF image of the input human silhouette is estimated, using a parallelised implementation of the algorithm described in [Felzenszwalb and Huttenlocher, 2004], in order to be used for the free-space violation error estimation.

Both operations are performed by the CPU, as they are not computationally expensive and are executed once for every input frame. The 3D point cloud and 2D SDF data are then serialised into two float vectors and are copied to the GPU memory.

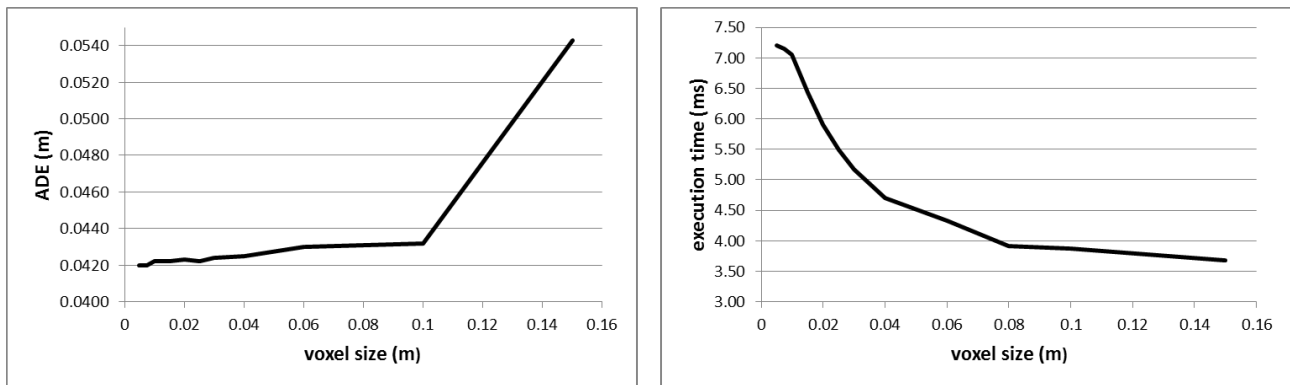


Figure 4.9: Tracking performance in relationship to the input cloud subsampling, for sample sequences from the EVAL dataset. Left: average distance error (10 iterations, 20 linesearch steps), Right: optimization execution time for a single iteration (20 linesearch steps)

4.11.2 Articulated-SDF Human Model

The pose tracker uses an articulated-SDF human body template which is produced offline from a detailed skinned human template, described in Section 4.2. When the tracker begins its operation, the SDF model is loaded and copied to the GPU's read-only texture memory, where it remains until the termination of the process.

In order to create the SDF human model, the 3D SDF transformation for each of the 15 rigid

body parts is estimated, using a two-step sampling process. First, the 3D mesh that defines the shape of the body part is sampled to 400 points using Poisson Disc sampling. Next, a $1m \times 1m \times 1m$ volume is defined around the body part, and is uniformly subsampled using a $128 \times 128 \times 128$ 3D grid. For each voxel center, the distance to the closest point of the sampled mesh is estimated, along with the normal vector and the gradient values along the 3 axis at this point. If the voxel lies within the mesh, the distance is assigned a negative sign. Finally, for each voxel point, the estimated distance and gradient vector are saved to a 4 point float array.

The produced articulated-SDF model takes up around 480MB ($128 \times 128 \times 128$ voxels $\times$ 4 float values $\times$ 15 body parts), which allowing it to fit in even relatively older GPUs. Newer devices do offer more available GPU memory, allowing the use of a larger 3D grid, however the resolution of the current grid is much higher than that of the subsampled input cloud ($0.008m$ vs $0.025m$), which ensures that the pose estimation accuracy is not hindered by it.

Further testing on a subset of the PDT dataset showed no significant increase in tracking accuracy for resolutions finer than $0.015m$ (corresponding to a $64 \times 64 \times 64$ 3D grid). On the other hand, there is a notable slow down in performance when the model becomes extremely large, for grid sizes over $192 \times 192 \times 192$. Moreover, selecting a grid size that is a power of 2 results in a small gain in speed.

4.11.3 Optimisation Process

Optimisation algorithm The implementation of the BFGS algorithm, used for the estimation of the optimal pose vector, is based on the LAPACK implementation. The direction and step of the optimisation process are computed on the CPU, while the functions that estimate the value and the derivatives of the objective function are passed to the BFGS as pointers, and thus are executed on the GPU.

Through preliminary testing, the maximum iterations and linesearch steps of the optimisation algorithm were set to $it_{max} = 10$ and $ls = 20$ respectively. While, it is possible to use a

smaller number of iterations ($it_{max} = 5$, $ls = 10$) for simpler motions, thus achieving a higher framerate, in the case of complex full body motions (e.g. sitting to the ground) the tracking would fail to converge. On the other hand, increasing the iterations ($it_{max} > 15$, $ls > 25$) did not result in any consistent gains in accuracy (Figure 4.10).

It should be noted that the optimal values for the it_{max} and ls parameters, as well as for the cloud subsampling resolution, described in Section 4.11.1, in order to achieve the optimal accuracy/speed combination, greatly vary from GPU to GPU, depending on each card's technical specifications.

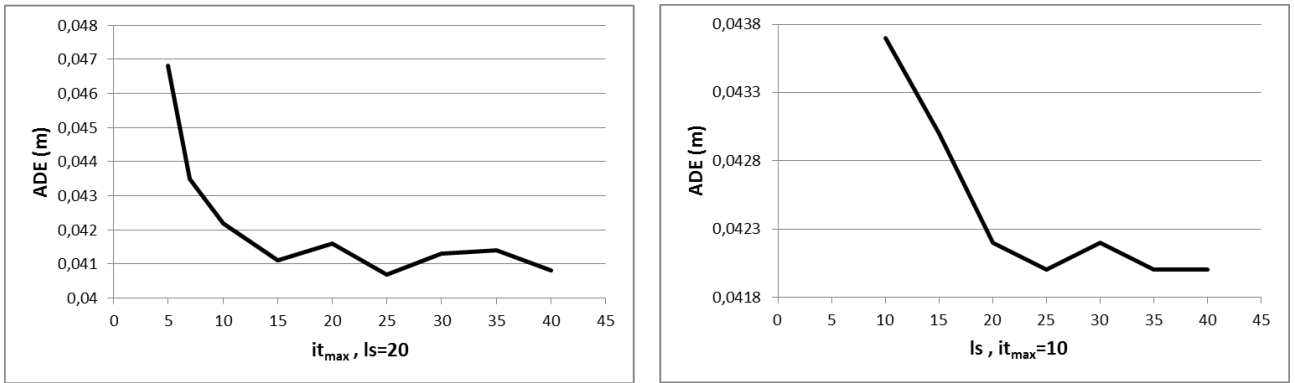


Figure 4.10: Tracking accuracy in relationship to the optimisation parameters, for sample sequences from the EVAL dataset. Left: maximum iterations (20 linesearch steps), Right: linesearch steps (10 maximum iterations)

CUDA kernels

For the approximation of the Hessian Matrix through the estimation of the value and derivatives of the objective function, as described in Section 4.3, four custom CUDA Kernels are employed:

```
transformPointsKernel <<< dim3(N_blocks , N_parts) , BLOCK_SIZE , smem >>>
samplePointsSDFKernel <<< dim3(N_blocks , N_parts) , BLOCK_SIZE , smem >>>
minDistanceKernel <<< N_blocks , BLOCK_SIZE , smem >>>
gradientKernel <<< dim3(N_blocks , N_vars) , BLOCK_SIZE , smem >>>
```

where $N_parts = 15$ the number of body parts, $BLOCK_SIZE = 192$ the experimentally selected optimal CUDA block size, $N_vars = 48$ the size of the pose vector (global translation,

global rotation in quaternions, 10x joint rotation in quaternions), `N_blocks = (N_points + BLOCK_SIZE - 1) / BLOCK_SIZE`, `N_points` the number of 3D points of the input point cloud and `smem` the corresponding memory.

In similar fashion, three more CUDA Kernels are employed for the free space violation error and the leg intersection error estimation, described in Section 4.7:

```
fsEnergyKernel  <<< N_blocks , BLOCK_SIZE , smem >>>
fsGradientKernel <<< dim3(N_blocks , N_vars) , BLOCK_SIZE , smem >>>
intersectionEnergyKernel <<< N_blocks_col , BLOCK_SIZE , smem >>>
```

where `N_blocks_col = (N_col_pairs + BLOCK_SIZE - 1) / BLOCK_SIZE` and `N_col_pairs = 96` the number of the sphere-pairs that are checked for intersection.

4.11.4 Tracker Initialisation

The tracker initialisation process is handled by the CPU, using the BFGS solver. Since initialisation is executed only once at the beginning of the tracking process, and the objective function value and gradient estimation is relatively cheap (15 joint 3D positions are used at most) the iteration limit and linesearch steps are set to high values ($it_{max}=300$, $ls=30$) in order to ensure that the algorithm successfully convergences to the correct pose.

4.11.5 Data Export

After the completion of the tracking process, the algorithm returns the 3D position of 15 joints (Figure 4.11), along with the pose vector and a tracking failure flag based on the tracking failure detection described in Section 4.9. If any of the joints is defined as non-visible, it is marked with an “invalid” flag, so as not to be taken into consideration.

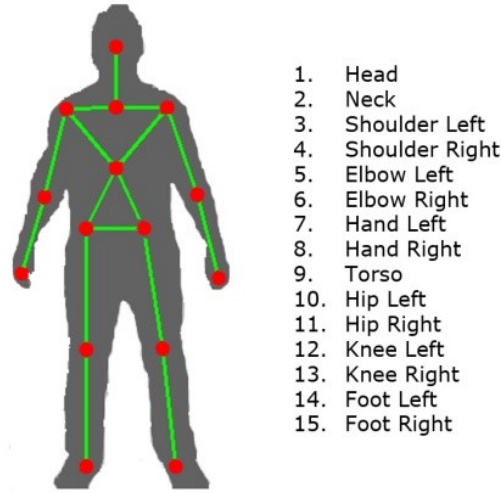


Figure 4.11: The skeleton joints extracted by the developed human tracking implementation

4.12 Experimental Evaluation

In order to quantitatively evaluate the pose detection accuracy of the proposed framework, a series of experiments are carried out. Following the evaluation process employed in the relevant literature, the measures used for the experimental evaluation of the proposed framework are:

- *Average Distance Error (ADE)*
- *Mean Average Precision at 0.1m (mAP)*

In all the experimental sequences a generic human model is used, which is initialised in the first frame using the ground truth data, while the proposed framework performs at an average framerate of 20fps on a modern CUDA-enabled GPU. Detailed experimental results are available in Appendix A, while a short demonstration video can also be found at <https://www.youtube.com/watch?v=ONv2JroqCnU> and in the supplementary data

4.12.1 Individual Components Evaluation

Two small-scale experiments are conducted, using subsets of the publicly available PDT [Helten et al., 2013a] dataset, in order to evaluate the effectiveness of the body part visibility and leg intersection features. Detailed results from these ablation studies can be found in Appendix A.

Body Part Visibility

The *body part visibility* component is tested in sequences of moderate lower body motion complexity, with the lower body parts being artificially occluded, by removing all the input data below the subjects' hips. Enabling the visibility component does not affect significantly the upper body parts' accuracy (disabled: [ADE 0.030, mAP 0.983] - enabled: [ADE 0.032, mAP 0.974]), which is expectable as all the upper body parts are visible throughout the testing sequences. For the occluded lower body parts, on the other hand, there is a clear increase in accuracy when enabling the body part visibility component (disabled: [ADE 0.283, mAP 0.366] - enabled: [ADE 0.204, mAP 0.389]). While the improvement in mAP is not significant, which is to be expected since it is hard to reach the 0.1m threshold without any information about the body parts, there is major improvement in the ADE of almost 8cm, which indicates that the body part visibility component keeps the occluded body parts closer to their actual position, increasing the probability of the tracker correctly tracking the occluded body parts once they reappear.

Table 4.1: Experimental evaluation of the body part visibility component, on artificially occluded sequences of the PDT dataset. Enabling the body part visibility component significantly decreases the ADE for the occluded lower body parts, increasing the probability of the tracker correctly tracking them once they reappear

	DISABLED		ENABLED	
	ADE(m)	mAP(0.1m)	ADE(m)	mAP(0.1m)
Upper body	0.030	0.983	0.032	0.974
Lower body	0.283	0.366	0.204	0.389

Leg Intersection

The *leg intersection* feature is tested in sequences with relatively complex leg movement. With the intersection feature disabled, the tracker sometimes mixes the lower body parts and keeps them mixed throughout the sequence, resulting in low accuracy for the four leg joints (R/L knee, R/L foot): *ADE* 0.094, *mAP* 0.814. However, enabling the leg intersection feature limits the leg mix-up to a few frames, significantly increasing the lower body accuracy to: *ADE* 0.041, *mAP* 0.941.

4.12.2 Public Datasets

The overall developed pose tracking method is evaluated using three publicly available human pose tracking datasets: SMMC-10 [Ganapathi et al., 2010a], EVAL [Ganapathi et al., 2012b] and PDT [Helten et al., 2013a]. Figure 4.12 presents the performance comparison between the proposed framework and state-of-the-art methods (as reported by their authors)¹, along with the per-joint accuracy for each of the three datasets. From the experimental results it becomes evident that the proposed framework performs comparably to other state-of-the-art methods, approaching the accuracy of the top performers in the SMMC-10 and EVAL datasets, while slightly outperforming them in the PDT dataset.

¹More comparative results against more recent DL-based methods are presented in section 5.8

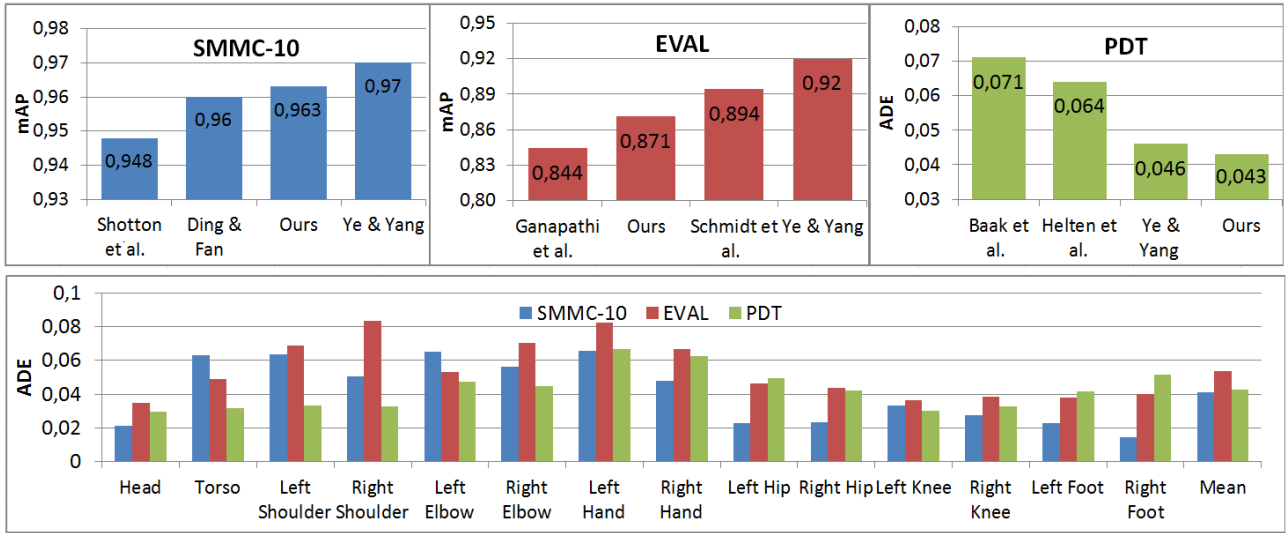


Figure 4.12: Framework performance on three public datasets. Top - comparison to state-of-the-art methods (results as reported by their authors), Bottom - ADE per-joint

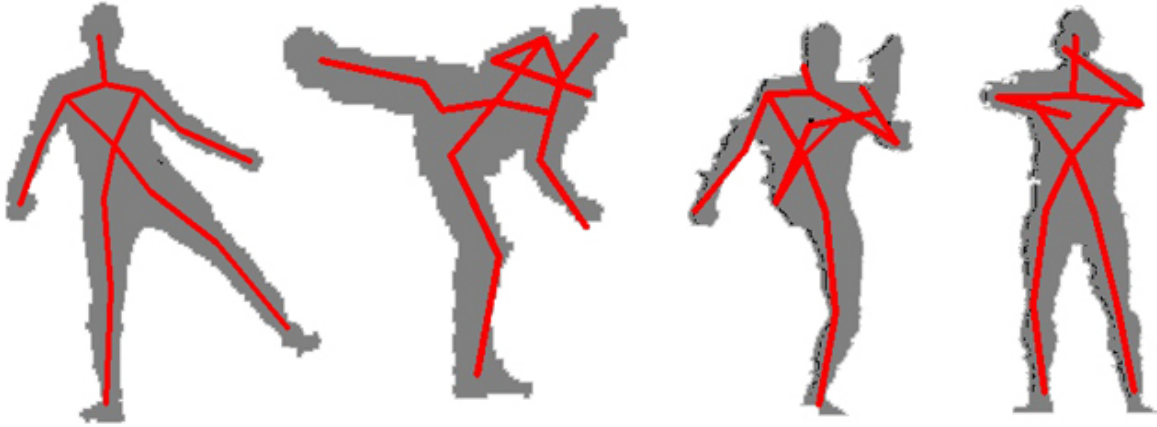


Figure 4.13: Sample frames from the evaluation on public datasets.

4.12.3 Realistic Human Motion Dataset

In order to evaluate the overall performance and robustness of the proposed framework in unconstrained real-life settings, it is tested on the captured Realistic Human Motion Dataset [Vasileiadis et al., 2017b], achieving promising results: ADE 0.075, mAP 0.825, and clearly outperforming the Kinect built-in pose estimator (NITE 2.2 middleware [PrimeSense, 2013]²). Figure 4.14 presents the detailed results, while Figure 4.15 shows some sample frames from the evaluation process.

²While the NITE algorithm has not been published, its functionality and performance approach the ones of Shotton et al. [Shotton et al., 2013, Calderita et al., 2013]

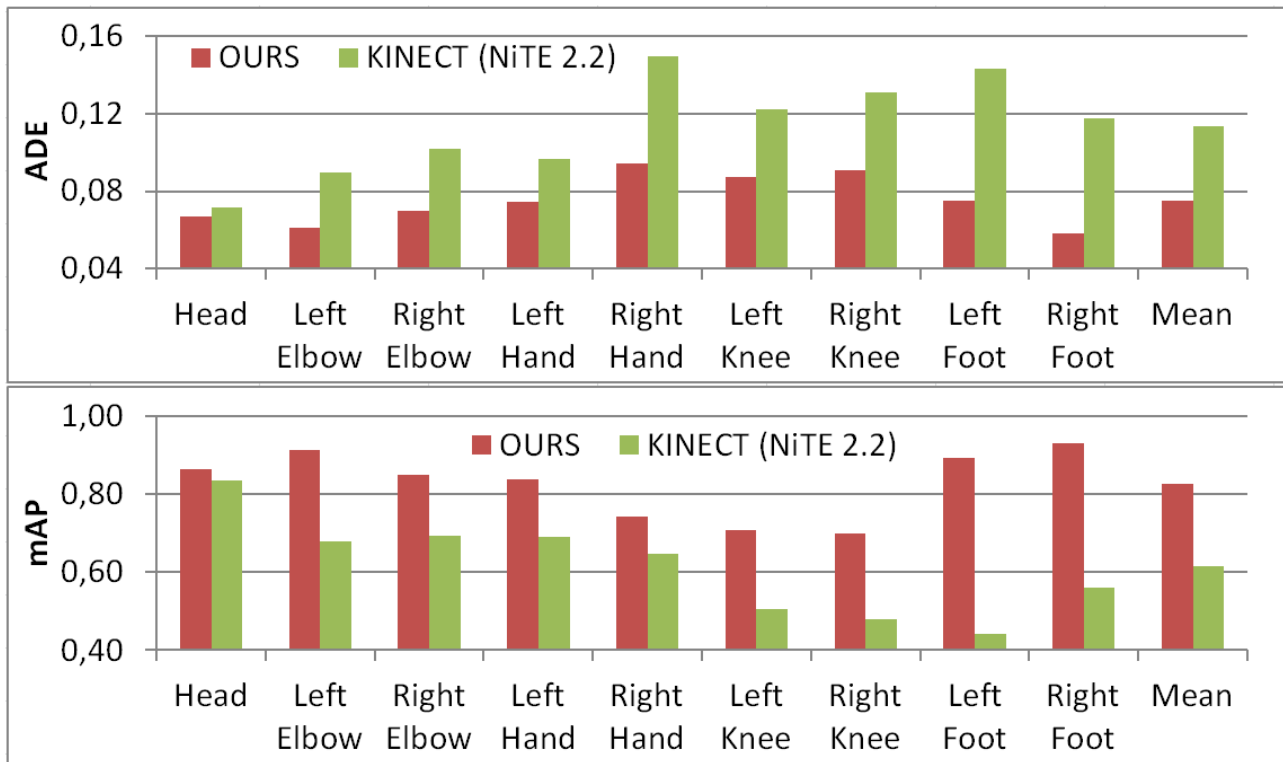


Figure 4.14: Comparison of per-joint ADE and mAP with the Kinect built-in pose estimator (NITE 2.2 middleware [PrimeSense, 2013]) on the Realistic Human Motion Dataset.

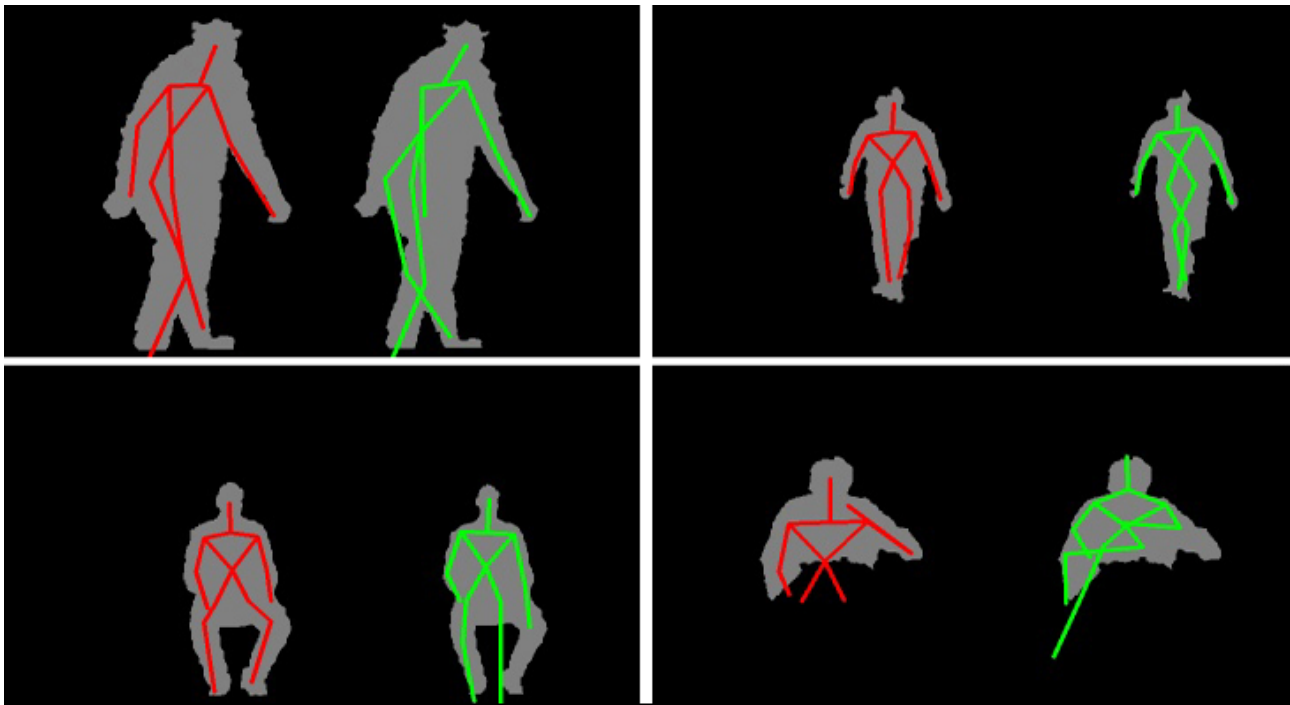


Figure 4.15: Sample frames from the evaluation on the Realistic Human Motion Dataset: Red - proposed framework, Green - Kinect built-in pose estimator (NITE 2.2 middleware [PrimeSense, 2013])

4.13 Implementation on Autonomous Robotic Platform

The developed method's robustness and efficiency under realistic conditions is further demonstrated by its implementation on the autonomous RAMCIP robotic platform [Kostavelis et al., 2018], which is a Robotic Assistant for patients with Mild Cognitive Impairments (MCI), suitable to provide its services in domestic environments.



Figure 4.16: The first and second iterations of the RAMCIP autonomous robotic platform

The template-based human pose tracking architecture is employed within the scope of human monitoring, which involves the continuous detection and tracking of the human in the environment. The estimated human skeleton is used for safe robot navigation, short term human action recognition [Yang and Tian, 2014] and object manipulation.



Figure 4.17: Sample frames of the template-based pose tracker used for action recognition in realistic settings

Moreover, it serves as a low level sensory input for a novel model for probabilistic Modelling of

Human Behaviour through analysis of activities of daily living [Kostavelis et al., 2019]. More specifically, a set of 4 target activities (meal preparation, cooking, eating, medication intake) are decomposed, using Interaction Unit Analysis [Ryu and Monk, 2009], into sequences of action units, each one associated with a behavioural factor. The modelling of human behaviour is then addressed with a Dynamic Bayesian Network that operates on top of the Interaction Units, offering quantification of the behavioural factors and the formulation of the human’s behavioural model.

The RAMCIP robotic platform is experimentally deployed in two test sites (Medical University of Lublin, Poland and 12 residences in Barcelona, Spain³) with the participation of 30 subjects in total. The proposed activity modelling method, utilising the developed template-based pose tracker, achieves extremely high recognition rates of over 93% for the target activities, and over 86% for the individual action units.

4.14 Summary

In this chapter, human pose tracking was handled through the introduction of a human-template based pose estimation and tracking framework from depth data. The framework’s human pose tracker employs an articulated skinned human template, composed of 15 rigid body parts and 3000 vertices, to track the human pose in sequences of depth images, extracting the 3D positions of 15 skeletal joints.

The tracker employs the articulated signed distance function representation, replacing the computationally taxing explicit nearest point computation with a much faster lookup in the SDF, thus allowing for the utilisation of a much denser data representation. The geometry of each rigid body part of the human template is defined implicitly by an SDF, leading to the approximation of the global signed distance function of the template, as a composition of the precomputed local signed distance functions, alleviating the need to recompute it every time

³The experiments at the Barcelona test site were performed under the supervision of Fundacio ACE Barcelona Alzheimer Institute and Research Centre

the template pose is updated. The optimal pose for each input is estimated through the minimisation of the objective function, using the Quasi-Newton BFGS algorithm. BFGS employs an iterative approach to arrive at a minimal function value, and in contrast to classic Newton methods, approximates the Hessian matrix of second derivatives using much simpler to calculate rank-one updates, specified by gradient evaluations

In addition to the main pose tracker, a series of complementary tracking features are also introduced, in order to increase the tracker's overall robustness and accuracy. The features tackle the issues of free space violation, body part visibility and leg intersection, based on problems typically encountered during operation under real life recording conditions. Moreover, a framework initialisation and tracking failure scheme is introduced, using the Kinect 1 built-in pose estimator, while a preprocessing step is employed, for the removal of large objects with which the tracked human is interacting, based on the floodfill algorithm.

The algorithm is implemented using C++ and CUDA code, utilising open source libraries, along with custom-made CUDA kernels, mainly related to functions of the optimisation process. Through extended performance analysis, a set of implementation parameters is selected, towards the optimal performance/accuracy trade-off, achieving an average framerate of 20fps on a modern CUDA-enabled GPU.

Experiments are performed to evaluate the effectiveness of the proposed approach. In the first series of experiments, the individual complementary tracking components were evaluated, significantly increasing the accuracy of the overall tracker. In the second phase the overall framework was compared to state-of-the-art methods on three publicly available human motion depth datasets, approaching the accuracy of the top performers in two of the datasets, while slightly outperforming them in the third. Finally, testing on the captured Realistic Human Motion Dataset, achieves promising results, clearly outperforming the Kinect built-in pose estimator.

The experimental results clearly demonstrate the effectiveness of the proposed algorithm. The combination of the dense SDF model representation and the BFGS optimizer, achieves high accuracy and real-time performance in sequences recorded in controlled environments. More-

over, the introduction of the complementary tracking features, further enhance the algorithm's robustness when deployed in unconstrained conditions, tackling, specifically, issues that are commonly encountered in such scenarios.

On the other hand, the main limitation of this approach lies on the framework's reliance on the Kinect's built-in estimator for initialisation and background/human segmentation, as the implementation used herein acts as a standalone module and is susceptible to failure, especially in the case of camera movement. Moreover the closed-source nature of the Kinect's estimation algorithm limits the framework's optimisation potential, especially in the case of implementation in non-PC architectures. In an attempt to overcome these limitations, preliminary work towards a novel human pose estimation algorithm is presented in the next section, based on deep convolutional neural networks.

Chapter 5

Deep Learning-Based Human Pose Estimation

This chapter presents a novel 3D-CNN architecture for 3D human pose estimation from 3D data. The architecture generates per-voxel likelihood maps for each human joint and body part, from a 3D voxel-grid input, and is able to simultaneously estimate multiple human poses, without its runtime complexity being affected by the number of people within the scene. Comparative evaluation on three single person and one multi-person public datasets, demonstrates state-of-the-art performance.

Parts of the contents of this chapter have been published in [Vasileiadis et al., 2019b].

In chapter 4, a novel human-template based pose tracking framework was proposed within the context of this PhD research. The developed method achieved state-of-the-art accuracy and real-time performance, targeting the application scenarios defined in Section 3.1. However, the design of the algorithm presents two major drawbacks: (a) the tracking algorithm requires human/background segmentation to function properly, which can hinder its performance on dynamic moving platforms (i.e. handheld devices) and (b) in each sequence the human template is initialised through an out-of-the-box pose estimator [Shotton et al., 2013], which can be susceptible to failure, especially in challenging poses, while simultaneously limiting the algorithm's

optimisation potential within the context of low power devices. Additionally, a performance issue also emerges in the case of multi-person tracking, as the algorithm has to be executed independently for each person in the scene, significantly affecting its ability to perform in real time.

Attempting to overcome these obstacles, an alternative human pose estimation technique is investigated, targeting the development of a human pose estimation framework capable of detecting, in an accurate and robust manner, the 3D real world coordinates of the skeleton joints of humans in depth images. The algorithm should not present the limiting factors of the template-based approach described in chapter 4, meaning it should perform efficiently without requiring any type of prior knowledge (i.e. scene labelling, last detected human pose etc.), while operating in a timely manner without being significantly affected by the number of people in the scene.

Towards this end, a Deep Learning-based multi-person human pose estimation approach is investigated, based on relevant state-of-the-art work for RGB data [Cao et al., 2017]. However, preliminary testing of the vanilla OpenPose architecture [Cao et al., 2017], utilising only depth images without taking into consideration any corresponding 3D information, fails to outperform current pure 3D data-based methods [Moon et al., 2018]. As a result, a novel 3D-CNN architecture for 3D human pose estimation from 3D data is developed, following the sequential network architecture paradigm [Ramakrishna et al., 2014], achieving state-of-the-art performance on single and multi-person public datasets.

Section 5.1 provides the experimental results from the preliminary testing of OpenPose on depth data, while section 5.2 introduces the novel 3D-CNN approach. Sections 5.3 and 5.4 describe the input data representation model and the network architecture respectively. Section 5.5 provides an overview of the multi-pose composition process and sections 5.6 and 5.7 present the training and implementation details. Finally, the results from the comparative experimental evaluation are reported in section 5.8, while section 5.9 discusses the potential advantages of the proposed design over other SoA methods, including the template-based tracker presented in chapter 4.

5.1 Preliminary Testing on the 2D Depth Image Plane

To better investigate the potential of CNNs for 3D HPE from depth data and establish a clear performance baseline, the multi-person 2D pose estimation architecture “OpenPose” [Cao et al., 2017] is slightly modified in order to operate with depth image data. “OpenPose” is selected due to its state-of-the-art accuracy and efficient implementation (e.g. while [Papandreou et al., 2018] is slightly more accurate, it is significantly slower). Moreover, its bottom-up approach (see Section 2.1.5) allows more consistent computational complexity evaluation, as it is not affected from the number of humans in the scene. Additionally, the sequential multi-stage architecture can be easily parameterised by changing the number of stages, providing a straightforward method for adjusting the accuracy/complexity trade-off.

5.1.1 The OpenPose Architecture

The architecture combines confidence maps for human joint prediction, along with Part Affinity Fields (PAFs) for body part association; 2D vector fields that represent the location and orientation, on the image plane, of body parts between neighbouring joints.

Specifically, a two-branch multi-stage CNN is employed (Figure 5.1), which simultaneously produces the 2D confidence maps $\mathbf{S} = (\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_{N+1})$, $\mathbf{S}_n \in R^{w \times h}$ for N skeleton joints and the non-human background data, and the 2D vector fields $\mathbf{L} = (\mathbf{L}_1, \mathbf{L}_2, \dots, \mathbf{L}_K)$, $\mathbf{L}_k \in R^{w \times h \times 2}$ for K body parts, each defined as the area between two neighbouring joints. Next, the joints’ confidence maps and part affinity fields are parsed through greedy inference and assembled into full body poses for all the people in the image. The predicted 2D joint positions are then transformed to 3D real world coordinates by back-projecting the corresponding pixels to the depth camera’s 3D coordinate system.

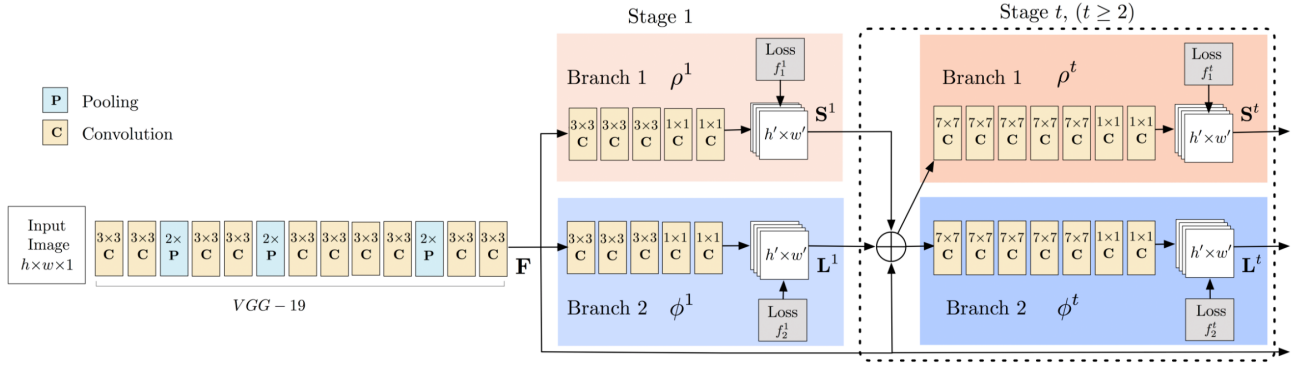


Figure 5.1: Architecture of the two-branch multi-stage CNN. First, the image is analysed through a VGG-19 [Simonyan and Zisserman, 2015] based network structure. Next, the exported feature maps F are passed to the two branch network structure. At each stage t the top branch produces the joint confidence maps S_t , while the lower branch produces the Part Affinity Fields L_t . After each stage, the predictions from the two branches, along with the original image features, are concatenated for the next stage (image taken from [Cao et al., 2017])

5.1.2 Experimental Evaluation

In order to quantitatively evaluate the pose detection accuracy of the 2D CNN-based framework, a series of experiments are carried out on the EVAL [Ganapathi et al., 2012b] and ITOP (front) [Haque et al., 2016b] datasets, using the 2D depth maps as input. Following the evaluation process employed in the relevant literature, two measures are used for the experimental evaluation of the proposed framework in 2D and 3D space:

- *Probability of Correct Keypoints, normalised towards the head (PCKh)*
- *Mean Average Precision at 0.1m (mAP)*

For each dataset, the data are separated into mutually exclusive train and test sets, with a small randomly sampled subset of the test set also used as a validation set during training. The number of confidence maps and PAFs is defined by the number of available ground truth annotations in each dataset, while each depth frame is evaluated independently without utilising any type of temporal information. Detailed experimental results are available in Appendix B.

For the ITOP dataset the front subset is used. Additionally, only the manually validated frames are taken into consideration, as per the authors’ instructions, which results in discarding approximately half of the available frames. The dataset uses a 80%/20% train-test split and is trained for 50K iterations, with ground truth annotations provided for 15 body joints: *head*, *neck*, *R/L shoulder*, *R/L elbow*, *R/L hand*, *torso*, *R/L hip*, *R/L knee*, *R/L foot*. Since head segmentation is not provided, the distance threshold for the PCKh measure is estimated by manually measuring the head segment length at the first frame of each sequence.

The algorithm achieves high accuracy rates on the 2D image plane (PCKh=95.1%). All the joint predictions surpass the 90% threshold, except for the hands, which due to their small size and noisy nature of real sensor readings present less details thus increasing the prediction error (PCKh=81.2%). In 3D space, however, there is a noticeable drop-off (mAP=85%), as the much noisier real sensor readings increase the projection error, especially in the case of smaller body parts, such as the hands (mAP=59%) (Figure 5.2).

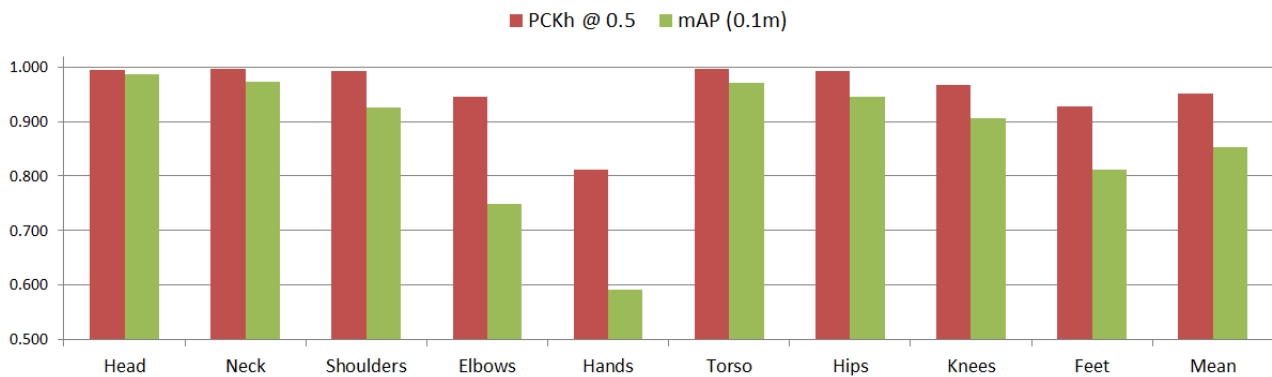


Figure 5.2: Per-joint accuracy from the experimental evaluation of OpenPose on the ITOP depth dataset (front)

For the EVAL dataset, since the authors do not define a train/test split, the leave-one-out strategy is followed similar to [Haque et al., 2016b, Moon et al., 2017]. The sequences from a person are used as the test set and the others are used as the training set. This process is performed once for each subject and the mean average precision is calculated to measure the overall performance. The network is trained 3 times for 30K iterations, using ground truth annotations for 12 body joints: *head*, *R/L shoulder*, *R/L elbow*, *R/L hand*, *torso*, *R/L knee*,

R/L foot. The distance threshold for the PCKh measure is estimated manually at the beginning of each sequence, as in the case of the ITOP dataset.

The experimental results follow the same pattern as the ITOP dataset; extremely high accuracy on the 2D plane (PCKh=94.7%), and significantly lower accuracy in the 3D space (mAP=79.8%), especially in the case of the hands (mAP=63.2%). It should be noted that the knee and elbow joints present unusually low accuracy in the 3D space (mAP_{knees}=75.5%, mAP_{elbows}=70.5%) as well. These results however are not consistent throughout the three subjects of the dataset. Specifically, the elbow joint predictions are very inaccurate only in the sequences of subject A (mAP_{elbows,A}=52.5%), while the knee joint predictions only in the sequences of subject C (mAP_{knees,C}=48.7%). Similar results are also reported in [Moon et al., 2017], indicating that the unusually low knee and elbow accuracy is caused by limitations in the dataset, rather than the algorithm design.

In the case of the elbow joints, the large discrepancies between sequences can be partially attributed to non-negligible deviations between the motion capture system (used for capturing the ground truth data) and the Kinect point cloud, which have been reported in other similar works as well [Schmidt et al., 2015]. Since the elbow joint lies on a relatively narrow area of the body, often the projected 2D ground truth positions end up outside of the human silhouette, incorrectly training the detection model. As for the knee joints, the accuracy levels are further affected by the relatively loose pants of the third actor, which decrease the amount of detail along the leg. Moreover, since the dataset does not provide robust ground truth data for hip joints, two PAFs are defined between the knee joints and the torso joint. However, defining such large PAFs, running along almost half of the body, over non-rigid areas, increases the chances of selecting a false positive knee detection over the actual knee, since a large portion of the image includes high confidence PAFs. This assumption is also validated from manually inspecting such frames, as it is noticed that the knee joints tend to be detected at higher positions, closer to the torso joint. Figure 5.3 presents the per-joint accuracy achieved on all the frames, while some qualitative results can be found in Figure 5.4.

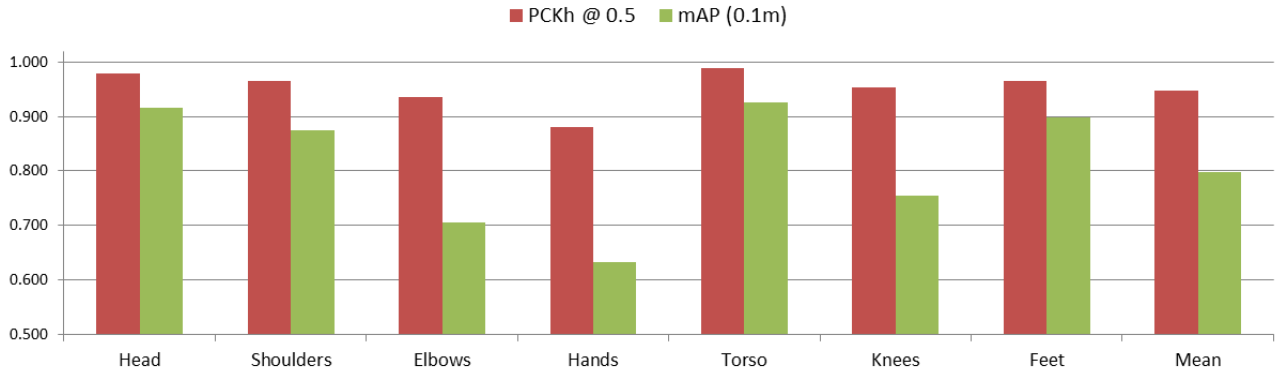


Figure 5.3: Per-joint accuracy from the experimental evaluation of OpenPose on the EVAL depth dataset

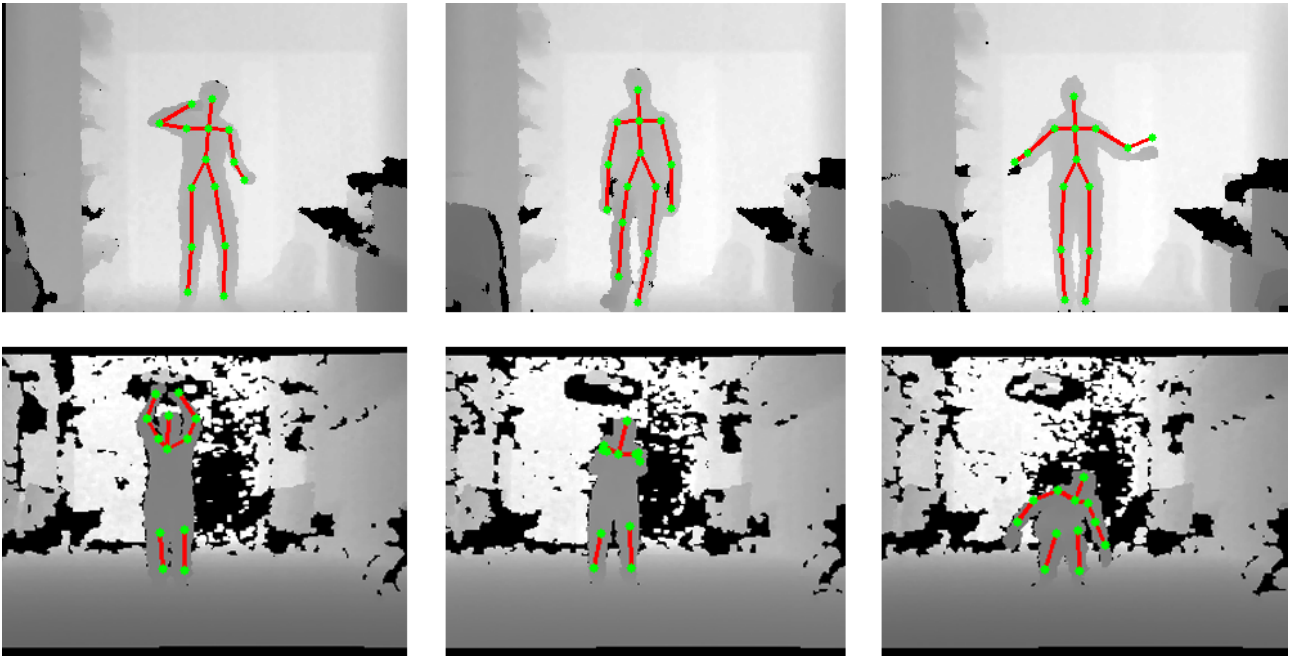


Figure 5.4: Sample frames from the experimental evaluation of OpenPose on the ITOP-front (top) and EVAL (bottom) depth datasets

Analysing the experimental evaluation results reveals various issues of the 2D CNN-based approach for 3D HPE. The comparison of the 2D CNN-based approach to state-of-the-art algorithms (Table 5.1), demonstrates that while the algorithm outperforms the methods that utilise only the 2D image information, it clearly falls behind approaches that operate in 3D space when it comes to 3D pose estimation accuracy, including the template-based method presented in chapter 4, which outperforms it by almost 8% on the EVAL dataset. Thus, it becomes apparent

that operating in 3D space provides a more natural representation of the input data, disengages the detection process from the limitations of the camera projection model, eliminating potential projection errors and improving overall accuracy and robustness.

Table 5.1: Comparison (mAP@0.1m) of the OpenPose [Cao et al., 2017] architecture to state-of-the-art-methods on the ITOP (front) and EVAL datasets

Method	Data type	Architecture	EVAL	ITOP (front)
[Haque et al., 2016b]	Depth map	CNN	74.1%	77.4%
[Jung et al., 2015]	Depth map	RF	74.0%	80.5%
[Moon et al., 2017]	Depth map + 3D PC	CNN	79.8%	83.4%
[Moon et al., 2018]	Depth map + 3D PC	CNN	-	88.7%
[Vasileiadis et al., 2017b]	3D point cloud	Template tracker	87.1%	-
[Ye and Yang, 2014]	3D point cloud	Template tracker	91.0%	-
OpenPose [Cao et al., 2017]	Depth map	CNN	79.8%	85.0%

5.2 3D-CNNs for 3D Human Pose Estimation from 3D Data

One of the main take-aways from the analysis of the experimental results presented in the previous section, is the potential improvement in performance that can be achieved by utilising the 3D data representation of the depth images, instead of being limited on the 2D image plane. Moving to 3D space has yielded noticeable improvements in various computer vision tasks, including human pose estimation and tracking [Ye and Yang, 2014, Moon et al., 2018], 3D object classification [Maturana and Scherer, 2015] and 3D hand pose estimation [Yuan et al., 2018a] from depth/3D data, making it a potentially decisive step towards the development of an accurate and robust CNN-based human pose estimation algorithm.

Towards this end a novel 3D-CNN architecture for 3D human pose estimation from depth/3D data is proposed. The network takes as input a 3D voxel grid and outputs 3D likelihood

maps for each joint. The proposed method does not segment the input into smaller regions of interest, as in the case of similar 3D-CNN architectures [Moon et al., 2017, Moon et al., 2018], but instead covers the majority of the effective Field-of-View of the depth sensor, allowing for the more efficient utilisation of global context information and the direct extension of the method towards multi-person pose estimation, in bottom-up fashion.

5.3 Input Data Representation

Input 3D point clouds are modelled in the form of 3D occupancy grids [Thrun, 2003] (also known as voxel grids). Occupancy grids provide a representation of the state of the environment as a 3D lattice of random variables (each corresponding to a voxel) and maintain a probabilistic estimate of their occupancy as a function of the input data. The finite resolution of these grids, as they quantize the 3D space into fixed-sized volumes, allows for the storage of unstructured 3D point clouds into dense data structures, suitable for CNN processing by GPUs.

In [Maturana and Scherer, 2015] the authors present 3 occupancy models, using 3D ray tracing [Amanatides et al., 1987] to calculate the number of hits and pass-throughs for each voxel:

- **Binary occupancy grid** Each voxel is assumed to be either occupied or unoccupied. The probabilistic estimate of occupancy for each voxel is computed with the log odds of the grid cell being occupied or free given that the measurement hit or missed the cell, respectively.
- **Density grid** Each voxel is assumed to have a continuous density, corresponding to the probability of the voxel blocking a sensor beam.
- **Hit grid** Each voxel is assigned a fixed value (usually 1) if at least one 3D point is included within its volume. Generating the hit grid does not require ray tracing, making it the most computationally inexpensive occupancy model.

For the proposed architecture, the computationally inexpensive hit grid model is utilised, as it

has been proven to provide similar accuracy results, at a fraction of the computational cost of more complex data representation models [Maturana and Scherer, 2015].

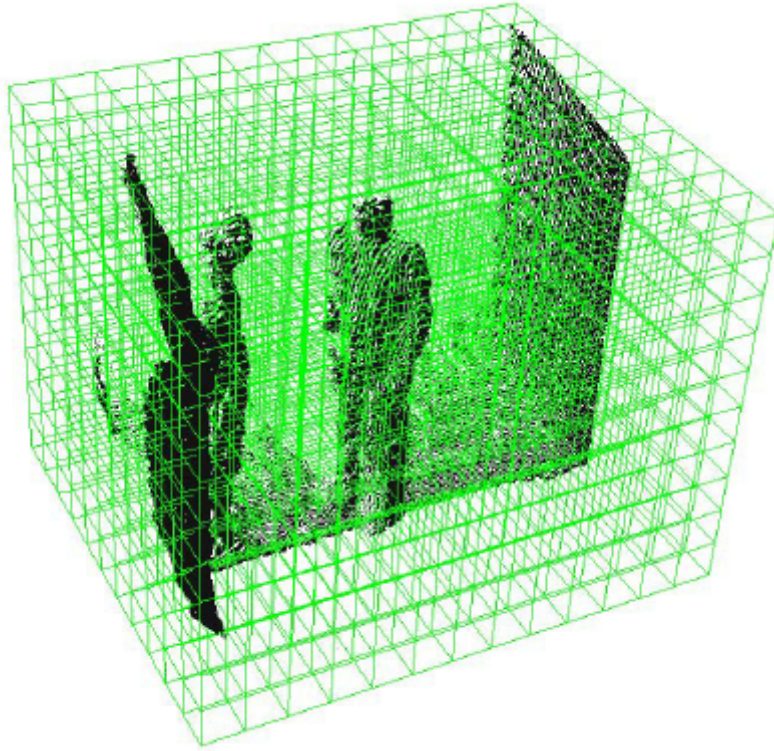


Figure 5.5: Generating a hit grid from a 3D point cloud. The 3D space is quantised using a fixed-size occupancy grid. Voxels that include at least one data point within their volume are assigned a fixed non-zero value.

5.4 Network Architecture

The proposed architecture is a fully-convolutional, detection-based estimator. It is split in two parts (Figure 5.6): an initial feature extraction stage followed by a sequence of repetitive prediction architectures, which refines the predictions over multiple $T \geq 1$ successive stages, with intermediate supervision at each stage [Ramakrishna et al., 2014, Wei et al., 2016]. Moreover, inspired by the work of [Cao et al., 2017], the architecture is extended towards multi-person 3D pose estimation, introducing a body part detector within the network and utilising an efficient joint association algorithm to compose full human poses from joint pairs.

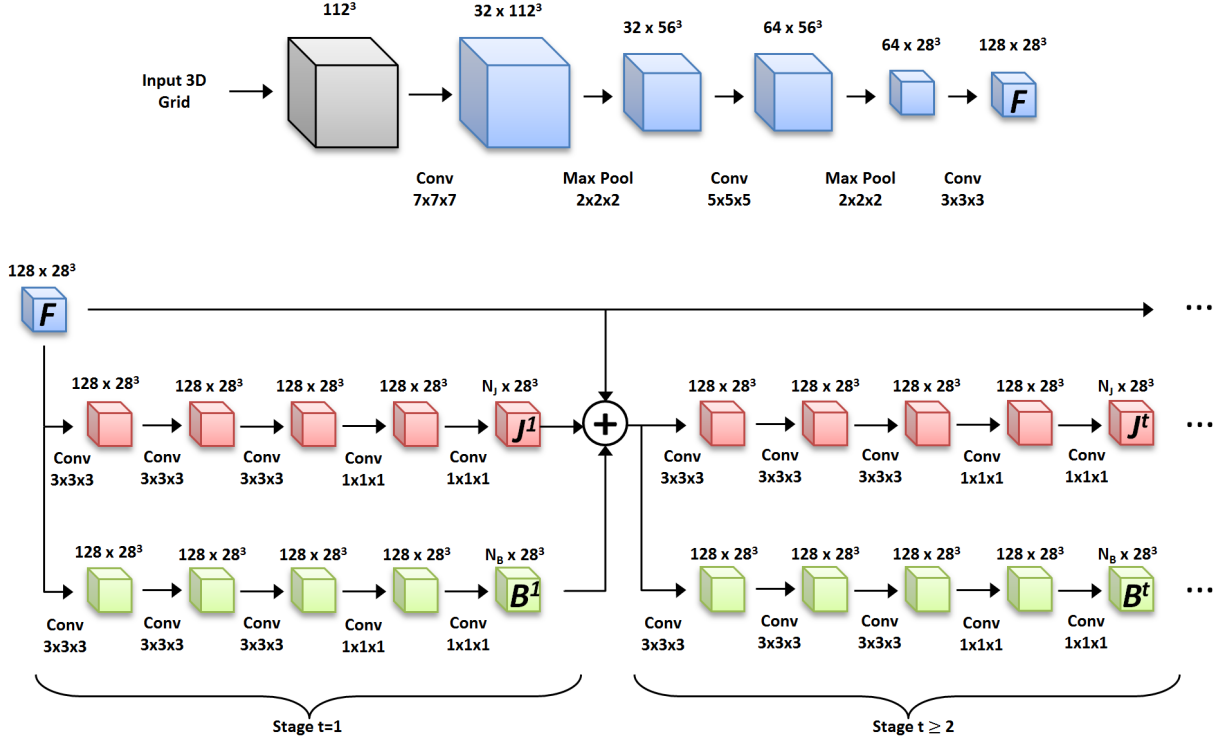


Figure 5.6: The proposed network architecture. Top: The feature extraction part (light blue), takes as input a $112 \times 112 \times 112$ 3D voxel grid (grey) and produces a set of features maps F . Bottom: The sequential multi-stage structure produces the per-voxel likelihood maps J^t , B^t for N_J joints (light red) and N_B body parts (light green) respectively. At the first stage $t = 1$ the likelihood maps are inferred only from the features F , while in subsequent stages the predictions from the previous stage, along with the original features F are concatenated in order to refine the predictions

The general layout of the network, including the single feature extractor and the multistage detectors, follows OpenPose [Cao et al., 2017], while the specific architecture and design parameters of each layer are extended to 3D and customised to better fit the task in hand. The 3-layer architecture of the feature extractor is based on the Voxnet 3D Object Recognition network by [Maturana and Scherer, 2015], which has been employed in many recent 3D CNN architectures for 3D object detection and classification [Zhi et al., 2017, Sedaghat et al., 2017]. Meanwhile, the 5-layer architecture of the individual stages, is an intuitive combination of the corresponding 2D layers in OpenPose and typical design choices commonly used in SOA 3D CNN architectures which attempt to “transfer” established 2D CNN architectures to 3D space [Moon et al., 2018].

5.4.1 Feature Extraction

The first part of the CNN-structure includes the feature extraction stage, which receives as input a $Width \times Height \times Depth$ 3D occupancy grid, and extracts a set of feature maps $\mathbf{F}$. While in typical 2D-CNN architectures, deep network layouts, such as VGG-19 [Simonyan and Zisserman, 2015], are commonly used for the initial feature extraction, recent works on similar 3D-oriented detection tasks [Ge et al., 2017, Huang and You, 2016, Maturana and Scherer, 2015] have demonstrated that high accuracy on 3D data can be achieved by using much shallower network layouts. Thus the feature extraction stage is defined as follows: three 3D convolutional layers with filter sizes $7 \times 7 \times 7$, $5 \times 5 \times 5$ and $3 \times 3 \times 3$ and two 3D max pooling layers in between, both with filter sizes $2 \times 2 \times 2$, with the ReLu [Nair and Hinton, 2010] activation function used at each convolutional layer.

5.4.2 Joints Detection

Following the feature extraction stage, the input is downsampled passing through the two 3D max pooling layers. This allows the use of a deeper architecture for the sequential multi-stage structure that produces the per-voxel likelihood maps for the joints. These maps provide a representation of the probability of a specific joint occurring at a 3D voxel, presenting peak values at the voxel coordinates where the corresponding joint is visible.

Each prediction stage follows the same architecture: three 3D convolutional layers with filter sizes $3 \times 3 \times 3$, followed by two layers with filter sizes $1 \times 1 \times 1$, with the ReLu activation function utilised at each layer except for the last one. At the first stage the joint likelihood maps $\mathbf{J}^1$ are inferred only from the features $\mathbf{F}$, while in subsequent stages the predictions from the previous stage, along with the original features $\mathbf{F}$ are concatenated in order to refine the predictions:

$$\mathbf{J}^t = \begin{cases} \rho^t(\mathbf{F}) & t = 1 \\ \rho^t(\mathbf{F}, \mathbf{J}^{t-1}) & t > 1 \end{cases} \quad (5.1)$$

where ρ^t denotes the convolution architecture at stage t .

The final joint likelihood maps are generated at a lower dimension than the original input, due to the use of the max pooling layers at the feature extraction stage, and thus are upsampled through trilinear interpolation, in order to provide more refined 3D positions of the joints.

Initially the upsampling stage was included within the network architecture, in the form of a stack of 3D deconvolutional layers, which along with an activation function can even learn nonlinear upsampling [Long et al., 2015]. However, through experimental evaluation it became evident that it did not offer any significant accuracy gain in comparison to fixed trilinear interpolation, and thus was removed from the architecture for efficiency reasons.

5.4.3 Extension Towards Multi-Person Pose Estimation

In order to extend the proposed methodology towards 3D human pose estimation for multiple people in the scene, the architecture is modified, inspired by the work of [Cao et al., 2017], by including a human body part detection structure (Figure 5.6). The body part detector runs in parallel with the sequential multi-stage joint prediction architecture described in section 5.4.2, allowing for the simultaneous prediction of joints and body parts, in the form of body part likelihood maps.

The body part likelihood maps provide information about the location of body parts in the 3D voxel grid, modelled as cylinders of radius r_k running along the line segment between two neighbouring joints, defined in a kinematic tree. While [Cao et al., 2017] utilise 2D vector fields, called “PAFs”, to model both the location and orientation of body limbs, herein a simplified binary map is used. When employed in 3D space the binary map actually outperformed the 3D “PAFs”, while being more computationally efficient, as it requires the estimation of just 1 voxel grid for each body part, instead of 3 for the 3D “PAFs” (one for each axis). The presence of a third dimension provides a natural representation of the empty space between entities in the scene, minimising body part overlapping between different people and making it easier, in comparison to 2D images, to define the corresponding body parts. Moreover, the more noisy

nature of depth data, compared to RGB images, increases the probability of a low confidence limb orientation score, thus negatively impacting the accuracy of 3D “PAFs”.

The body part detector follows the exact same sequential convolutional network design with the joint estimator, producing at each stage t the per-voxel body part likelihood maps $\mathbf{B}^t$, from the concatenation of the features $\mathbf{F}$ and the predictions from the previous stage, with eq. (5.1) being redefined as:

$$\begin{aligned} \mathbf{J}^t &= \begin{cases} \rho^t(\mathbf{F}) & t = 1 \\ \rho^t(\mathbf{F}, \mathbf{J}^{t-1}, \mathbf{B}^{t-1}) & t > 1 \end{cases} \\ \mathbf{B}^t &= \begin{cases} \varphi^t(\mathbf{F}) & t = 1 \\ \varphi^t(\mathbf{F}, \mathbf{J}^{t-1}, \mathbf{B}^{t-1}) & t > 1 \end{cases} \end{aligned} \quad (5.2)$$

where ρ^t , φ^t denote the joint and body part convolution architectures at stage t respectively.

5.5 Multi-Person Pose Composition

The joint prediction likelihood maps produce, through non-maximum suppression, multiple candidates for each joint, as areas presenting local maxima in each generated joint likelihood map are selected as potential joint positions. These candidate predictions correspond to multiple people in the scene or false positive joint detections and are composed into full human poses by estimating the optimal two-joint combinations that produce skeleton limbs which maximise the total body part overlap values.

This task can be approached as a graph matching problem, where the predicted joints are the nodes of the graph and the line segments between them, which represent the potential limbs, are the graph edges. Considering K types of joints, with every joint type J_k having N_k detections, a complete K-partite graph $K_{N_1, N_2, \dots, N_K}$ is defined, where $J_{1,2,\dots,K}$ are the K sets and $N_{1,2,\dots,K}$ the vertices per set. Finding the optimal combination of edges requires searching through $\sum_{i=1}^{K-1} (N_i \sum_{j=i+1}^K N_j)$ edges. Taking into consideration the fact that the human joints

are connected in a kinematic tree, meaning that each limb is defined by only two specific types of joints, the complete K -partite graph can be decomposed into $K - 1$ complete bipartite graphs. Each graph is composed by the adjacent joints in the kinematic tree, thus reducing the search space to $\sum_{i=1}^{K-1} N_i N_{i+1}$ edges. Moreover, since edges should not share nodes, as a joint cannot be shared by two similar body parts, a greedy parsing approach is employed, further reducing the search space to $\sum_{i=1}^{K-1} (N_i N_{i+1} - \frac{N_i(N_i-1)}{2})$ edges.

More specifically, the predicted body parts are compared to the corresponding body parts created from the predicted joints. The overlap between them is measured, defined as the sum of the body part likelihood map values at the voxel coordinates that lie on the predicted joint line segments. Next, the potential joint combinations are sorted in descending order based on their body part overlap score. Starting from the highest scoring candidate limb line segment, the candidate joint combinations are selected in order, ensuring that each joint is assigned only to single limb of the same type (i.e. two different upper hand limbs cannot share the same elbow joint), until no candidate joints are left to be selected. Finally, all the limbs that share common candidate joint detections are assembled into full human poses.

The adopted parsing approach produces high-quality results in an computationally efficient manner, while also providing efficient handling of false positive joint detections, as the lack of adjacent high level body part values leads to them being discarded. However, it also present the risk of discarding a legitimate joint detection, if its neighbouring joint is not detected (i.e. failing to detect the elbow joint, will result in discarding the hand joint as well, since the corresponding lower arm body part will not be composed). Similarly, a whole body part can be assessed as a separate human pose, if its parent joint is not detected (i.e. if the shoulder joint is not detected, the lower arm body part, composed from the elbow and hand joints, will not be connected to the rest of the pose). Thus, post processing is performed on the composed poses, by matching adjacent poses without corresponding body parts / joints (i.e. a pose without a left leg and a pose including only a lower left leg.)

Algorithm 1 Multi-person pose composition

```

1: procedure COMPOSE(predicted joints  $\mathbf{j}$ , predicted limbs  $\mathbf{l}$ , skeleton joint pairs  $\mathbf{p}$ )
2:   for all  $\mathbf{p}_{a,b}$  do
3:     for all  $\mathbf{j}_a$  do
4:       for all  $\mathbf{j}_b$  do
5:          $\mathbf{l}'_{ab} = \mathbf{j}_a - \mathbf{j}_b$  ▷ Line segment between  $\mathbf{j}_a, \mathbf{j}_b$ 
6:          $\text{score}(\mathbf{l}'_{ab}) = \text{overlap}(\mathbf{l}'_{ab}, \mathbf{l}_{ab})$ 
7:         if  $\text{score}(\mathbf{l}'_{ab}) < \text{threshold}$  then
8:           delete  $\mathbf{l}'_{ab}$ 
9:         end if
10:      end for
11:    end for
12:  end for
13:  sort  $\mathbf{l}'$  by score
14:  initialise  $\mathbf{l}''$  ▷ Final highest scoring line segments
15:  for all  $\mathbf{l}'_{a,b}$  do
16:    if  $\mathbf{j}_b \notin \mathbf{l}''$  then ▷ End joint  $\mathbf{b}$  hasn't been visited yet
17:       $\mathbf{l}''.\text{append}(\mathbf{l}'_{a,b})$ 
18:    end if
19:  end for
20:  initialise  $\mathbf{s}$  ▷ Final individual poses
21:  for all  $\mathbf{p}_{a,b}$  do
22:    for all  $\mathbf{l}''_{a,b}$  do
23:      for all  $\mathbf{s}_i$  do
24:        if  $\mathbf{j}_a \in \mathbf{s}_i$  then ▷ Parent limb already assigned to a pose
25:           $\mathbf{s}_i.\text{append}(\mathbf{l}''_{a,b})$ 
26:        else
27:          assign  $\mathbf{l}''_{a,b}$  to new  $\mathbf{s}$ 
28:        end if
29:      end for
30:    end for
31:  end for
32:  return  $\mathbf{s}$ 
33: end procedure

```

5.6 Network Training

5.6.1 Ground Truth Generation

Ground truth joint likelihood map generation During the training phase, the ground truth joint likelihood maps are generated from the ground truth 3D coordinates of the visible joints. Specifically, for a joint n the ground truth likelihood map $\mathbf{J}_n^{gt}$ is defined as:

$$\mathbf{J}_n^{gt}(\mathbf{v}) = \exp\left(-\frac{\|\mathbf{v} - \mathbf{j}\mathbf{t}_n^{gt}\|^2}{\sigma^2}\right) \quad (5.3)$$

where $\mathbf{v} \in R^3$ the voxel coordinates, $\mathbf{j}\mathbf{t}^{gt} \in R^3$ the ground truth joint coordinates and σ the coefficient that adjusts the spread of the Gaussian (Figure 5.7 - Left).

Ground truth body part likelihood map generation The ground truth body part likelihood maps $\mathbf{B}_k^{gt}$ are generated using the ground truth joint coordinates of each person in the scene (Figure 5.7 - Right). Considering a body part k created by two joints with ground truth 3D coordinates $\mathbf{j}\mathbf{t}_{n_1}^{gt}, \mathbf{j}\mathbf{t}_{n_2}^{gt} \in R^3$, the corresponding ground truth body part likelihood map is defined as:

$$\mathbf{B}_k^{gt}(\mathbf{v}) = \begin{cases} 1 & d(\mathbf{v}) \leq r_k \\ 0 & d(\mathbf{v}) > r_k \end{cases} \quad (5.4)$$

where $\mathbf{v} \in R^3$ the 3D voxel coordinates, $\mathbf{u} = \frac{\mathbf{j}\mathbf{t}_{n_2}^{gt} - \mathbf{j}\mathbf{t}_{n_1}^{gt}}{\|\mathbf{j}\mathbf{t}_{n_2}^{gt} - \mathbf{j}\mathbf{t}_{n_1}^{gt}\|}$ the line segment unit vector, $\mathbf{u}_\perp$ its perpendicular vector, $d(\mathbf{v}) = \|\mathbf{u}_\perp \cdot (\mathbf{v} - \mathbf{j}\mathbf{t}_{n_1}^{gt})\|$ the distance to the line segment and r_k the body part cylinder radius. Points that lie within a distance r_k of the body part line segment (points that “belong” to the body part cylinder) are equal to one, while points that do not meet the maximum distance threshold are equal to zero.

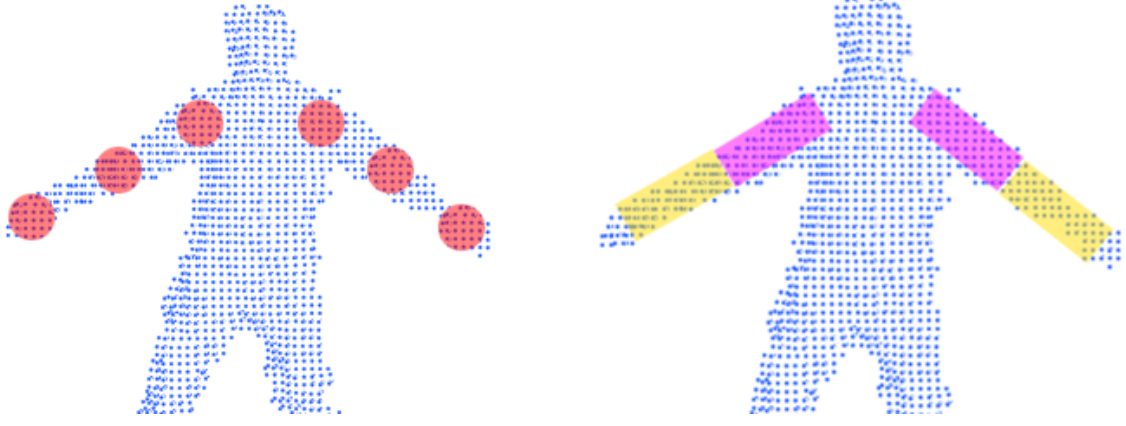


Figure 5.7: **Left:** Ground truth joint likelihood map for the shoulder, elbow and hand joints. Each joint is modelled by a 3D Gaussian, centred at the ground truth 3D coordinates of the joint. **Right:** Ground truth body part likelihood map for the upper and lower arm body parts. Each body part is modelled by a cylinder of a predefined radius, lying along the line segment generated from the corresponding ground truth joint positions

5.6.2 Training Loss Function

At each stage t , L_2 loss functions $f_{\mathbf{J}}^t, f_{\mathbf{B}}^t$ are applied between the ground truth likelihood maps $\mathbf{J}^{gt}, \mathbf{B}^{gt}$ and the output of the joint and body part prediction branches $\mathbf{J}^t, \mathbf{B}^t$ respectively. The intermediate supervision at each stage addresses the vanishing gradient problem by replenishing the gradient periodically [Wei et al., 2016]. The overall objective function f is defined as the sum of the functions across all stages and branches:

$$\begin{aligned}
 f_{\mathbf{J}}^t &= \sum_{i=1}^{N_J} \|\mathbf{J}_i^t - \mathbf{J}_i^{gt}\|_2^2 \\
 f_{\mathbf{B}}^t &= \sum_{i=1}^{N_B} \|\mathbf{B}_i^t - \mathbf{B}_i^{gt}\|_2^2 \\
 f &= \sum_{t=1}^T (f_{\mathbf{J}}^t + f_{\mathbf{B}}^t)
 \end{aligned} \tag{5.5}$$

5.6.3 Optimisation

For the optimisation of the objective function, the Adam [Kingma and Ba, 2015] gradient-based optimisation algorithm is employed, as it has been demonstrated to be well suited for problems

that are large in terms of data and parameters, such as the research problem at hand.

5.6.4 Data augmentation

The training data are augmented in 3D space through random rotation, translation and uniform scaling around all three axes, as well as flipping along the perpendicular axis. Including the augmented data in the training dataset yields an average increase in recognition accuracy of approximately $\sim 2\%$.

5.7 Implementation Details

A series of small scale experiments are carried out to fine-tune design aspects of the architecture, in order to achieve state-of-the-art accuracy.

5.7.1 Input Voxel Grid

Selecting the voxel resolution and dimensions of the input 3D grid comes down to finding the optimal computation/accuracy trade-off. Opting for a finer grid resolution provides a more detailed quantisation of 3D space, preserving the rich geometrical information of the input point cloud and leading to higher accuracy results (Figure 5.8 - Top), with overall accuracy reaching a plateau at approximately $\sim 2cm$. Using, however, a very fine resolution while simultaneously attempting to increase the effective area of the grid, requires the usage of large grids with multiple layers across each axis, with the run-time complexity increasing exponentially with the input grid dimensions (Figure 5.8 - Bottom).

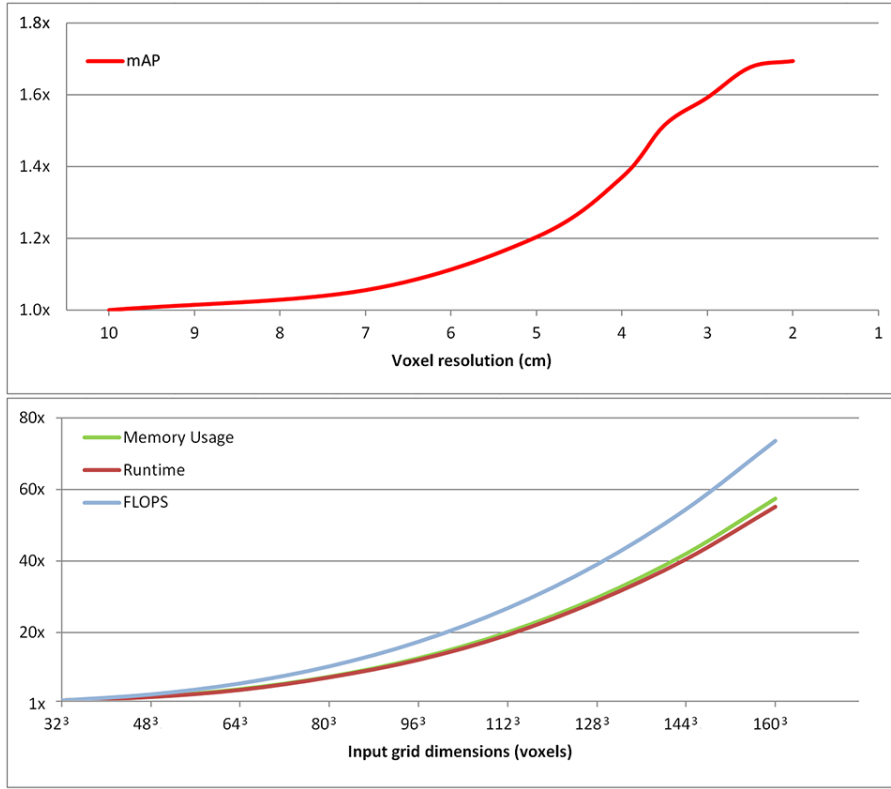


Figure 5.8: **Top:** Relative increase in accuracy when decreasing the voxel dimension of the input 3D voxel grid. **Bottom:** Relative increase in total memory usage, runtime and FLOPS when increasing the dimensions of the input 3D voxel grid

Taking into consideration these factors, as well as the effective field of view of typical RGB-D cameras [Gonzalez-Jorge et al., 2015], a $112 \times 112 \times 112$ grid is selected, spanning across a $3.2 \times 3.2 \times 3.2m$ volume around the approximate center of the cameras' effective FOV $\mathbf{c} = [0, 0, 2.8]m$. This quantisation produces a $0.028m$ voxel resolution, while the grid covers the majority of the cameras' effective field of view (Figure 5.9).

Furthermore, occupied voxels are assigned to $+1$, while unoccupied voxels are assigned to -1 , creating a naturally symmetric representation.

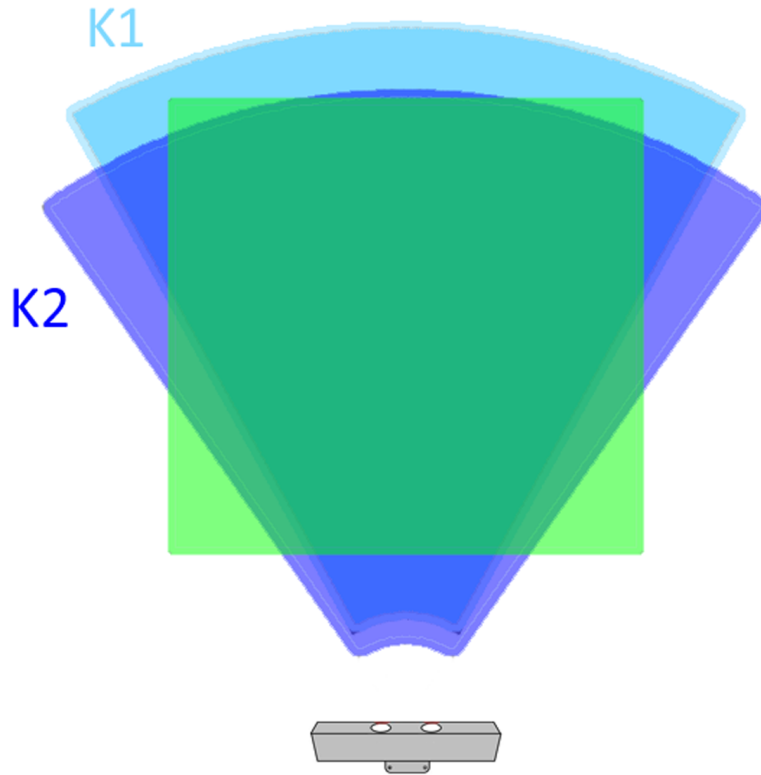


Figure 5.9: Top-down view of the input grid area (green) and the effective field of view of the Kinect 1 (light blue) and Kinect 2 (blue) depth cameras

Larger 3D scenes

Increasing the detection area of the network, in order to accommodate multiple people in distances larger than $3.2m$, simply comes down to increasing the dimensions of the input voxel grid; the pose estimation accuracy is not affected, as long as the $2cm$ voxel size requirement is met. Obviously, increasing the dimensions of the input grid, for a set voxel size value, requires more computational resources (Figure 5.8 - Right). However, due to the design of the algorithm, it is possible to share the load on multiple processing units (i.e. GPUs), with each one estimating the human poses in non-overlapping sub-grids of the input grid. As the algorithm does not require a fully visible human in order to detect the joints and body parts, it can estimate the poses even for persons who may be split by the boundaries of these sub-grids (i.e. the right arm appearing in one sub-grid, while the rest of the body on another). Any estimated partial poses are composed to full poses in the multi-person pose composition step described in section 5.5.

5.7.2 Network Implementation

Regarding the number of repetitive prediction stages in the sequential multi-stage structure, initial experimental evaluation reveals that each subsequent stage incrementally increases the overall prediction accuracy (Figure 5.10), with smaller body parts towards the edges of the kinematic tree (i.e hands, feet) benefiting from a deeper architecture. Moreover, while each additional stage incrementally increases the run-time and memory requirements, the initial feature extraction stage, which deals with the full-sized input grid, remains the most computationally taxing part of the architecture. Based on the above, along with the observation that there is marginal increase in overall accuracy after the 4th stage, the number of the stages in the sequential multi-stage convolution architecture is set to $T = 5$.

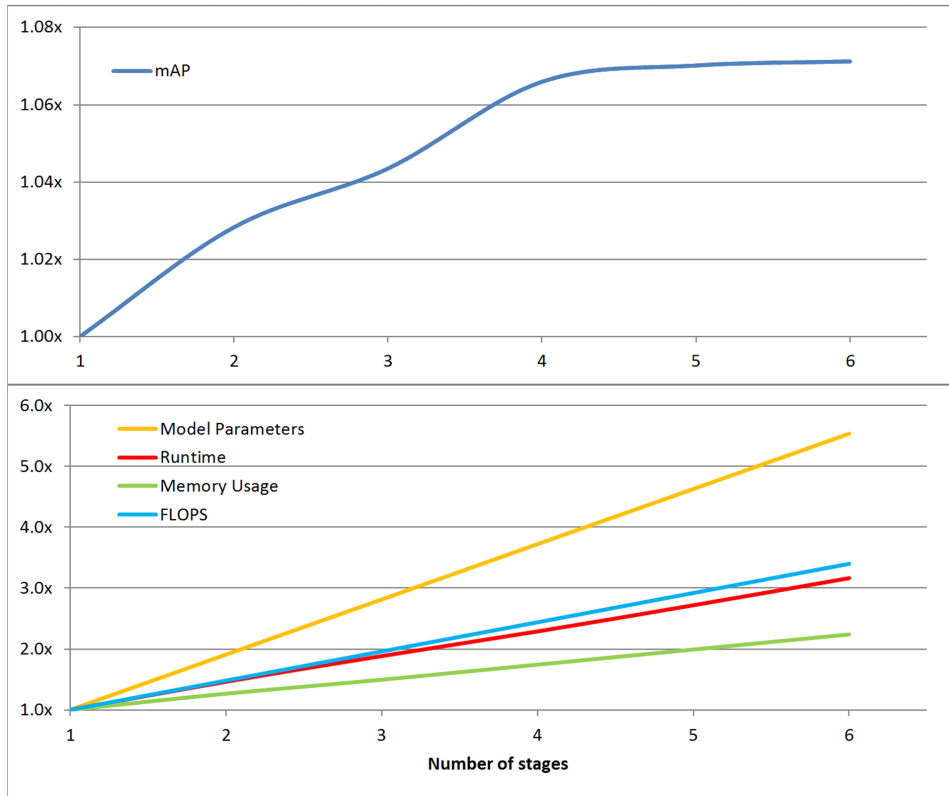


Figure 5.10: Relative increase in accuracy, number of model parameters, runtime, total memory usage and FLOPS when increasing the number of stages in the sequential multi-stage structure

5.7.3 Training

During the training phase, the default parameters are used for the Adam optimisation algorithm: $[\alpha = 10^{-4}, \beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}]$. The network is trained from scratch, initialised from a Gaussian distribution [LeCun et al., 1998b], for 20 epochs, using a batch size of 8 samples. For the ground truth joint likelihood maps generation the spread coefficient is set to $\sigma = 3$ (Eq. 5.3), while the maximum distance for the body part generation is set to $r_k = 0.07m$ (Eq. 5.4).

5.8 Experimental Evaluation

5.8.1 Single person 3D pose estimation

The 3D pose estimation architecture is trained and evaluated on three publicly available single person datasets, ITOP, EVAL and PDT. In order to ensure the consistency of the results, in comparison to other single person pose estimation techniques evaluated on the same datasets, for the single person experiments only the joint prediction structure is taken into consideration, with the pose at each frame composed from the global maxima of each generated joint likelihood map. The detailed results of the experimental evaluation on the three single person pose estimation datasets are presented in Table 5.2 below, while some qualitative results can be found in Figure 5.11. More detailed results are available in Appendix B, while a short demonstration video can also be found at <https://www.youtube.com/watch?v=0faB67BnwQE> and in the supplementary data.

Table 5.2: Experimental evaluation ($mAP@0.1m$) of the single-person 3D pose estimation architecture on the ITOP - front, ITOP - top, EVAL and PDT datasets

	ITOP - front	ITOP - top	EVAL	PDT
Head	0.985	0.984	0.917	0.962
Neck	0.990	0.987	0.955	0.986
Shoulders	0.967	0.965	0.919	0.959
Elbows	0.825	0.802	0.763	0.888
Hands	0.718	0.654	0.839	0.830
Torso	0.984	0.979	n/a	0.970
Hips	0.918	0.903	n/a	0.891
Knees	0.926	0.828	0.852	0.939
Feet	0.860	0.711	0.927	0.888
MEAN	0.893	0.845	0.873	0.914

For the ITOP dataset both the front and top subsets are used. Additionally, only the manually validated frames are taken into consideration, as per the authors’ instructions, which results in discarding approximately half of the available frames. The dataset uses a 80%/20% train-test split and is trained for 15 joints. For the ITOP - front subset the proposed framework achieves an overall mAP of 89.2%, with most joint predictions passing the 90% threshold. However, joints towards the edges of the kinematic tree (i.e. hands) present a lower accuracy, mainly attributed to the small size and noisy depth readings of the corresponding body parts, as well as the common self-occlusion of these parts. For the more challenging ITOP - top subset the proposed framework achieves an overall mAP of 84.5%, with the accuracy dropping as we move further down along the kinematic tree.

For the EVAL dataset, since the authors do not define a train/test split, the leave-one-out strategy is followed similar to [Haque et al., 2016b, Moon et al., 2017]. The sequences from a person are used as the test set and the others are used as the training set. This process is performed once for each subject and the mean average precision is calculated to measure the

overall performance over 12 joints (the dataset does not provide consistent ground truth for the hips and torso joints). The experimental results follow the same pattern as the ITOP dataset. An overall mAP of 87.3% is achieved, with high accuracy reported for joints on larger body parts.

Similar to the EVAL dataset, we follow the leave-one-out strategy for the PDT dataset as well. The dataset is divided in three subsets, ensuring that each one includes at least one sequence from every one of the 5 subjects, with two of the subsets used as a training set and one as a test set. The process is repeated three times, once for each subset, with other overall accuracy evaluated over 15 joints. The overall mAP surpasses the 90% threshold, achieving an accuracy score of 91.4%. Moreover, since the ground truth data are provided through a motion capture system, regardless if the joint is visible to the depth camera or not, along with the large number of training data, allows the model to efficiently learn to infer the position of occluded joints, providing a relatively good estimation for self-occluded joints when they are close to the visible body.

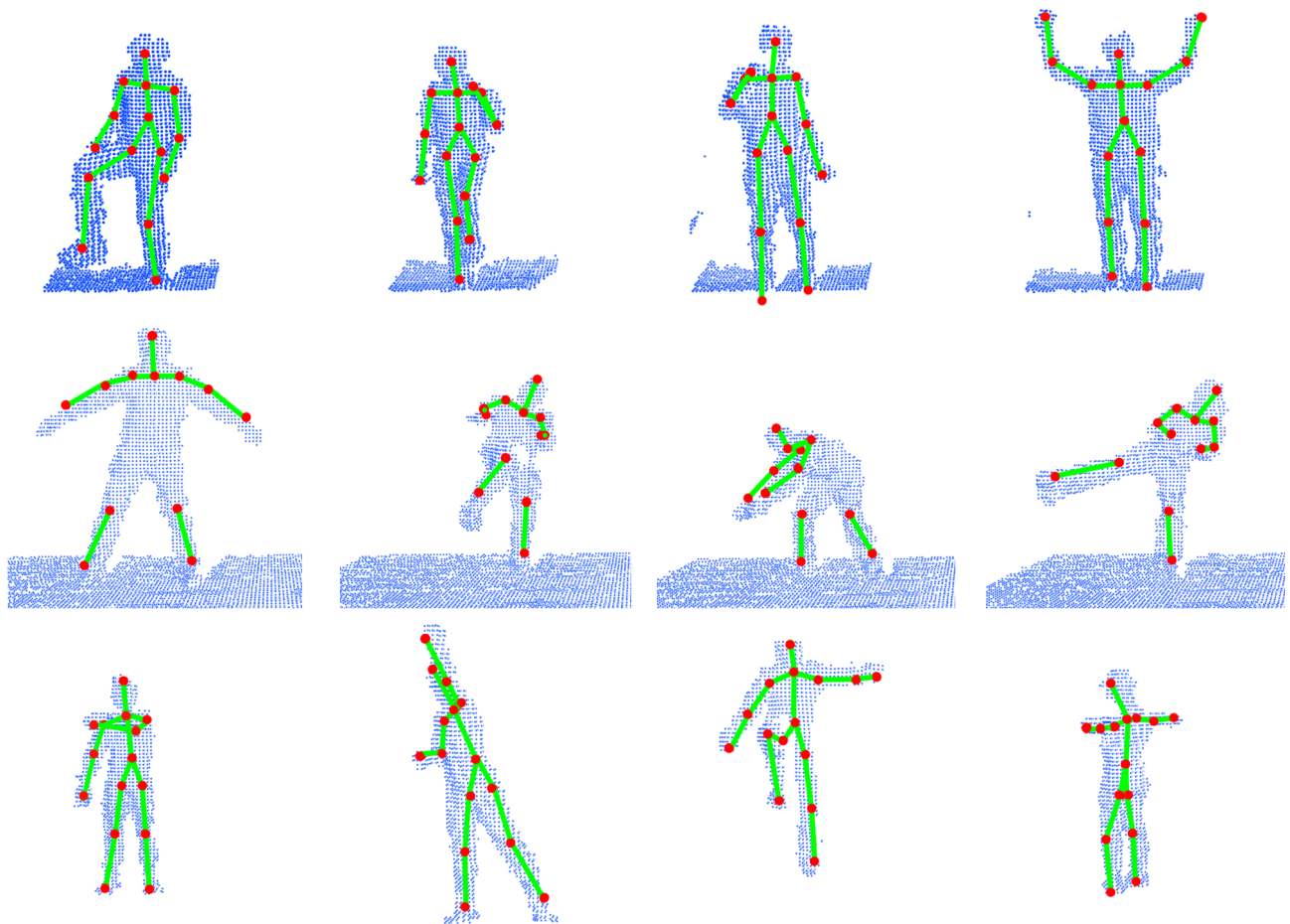


Figure 5.11: Qualitative results from experimental evaluation of the single person 3D pose estimation architecture on the ITOP - front (top), EVAL (middle) and PDT (bottom) datasets. Background data have been removed for better visualization

Comparison to state of the art

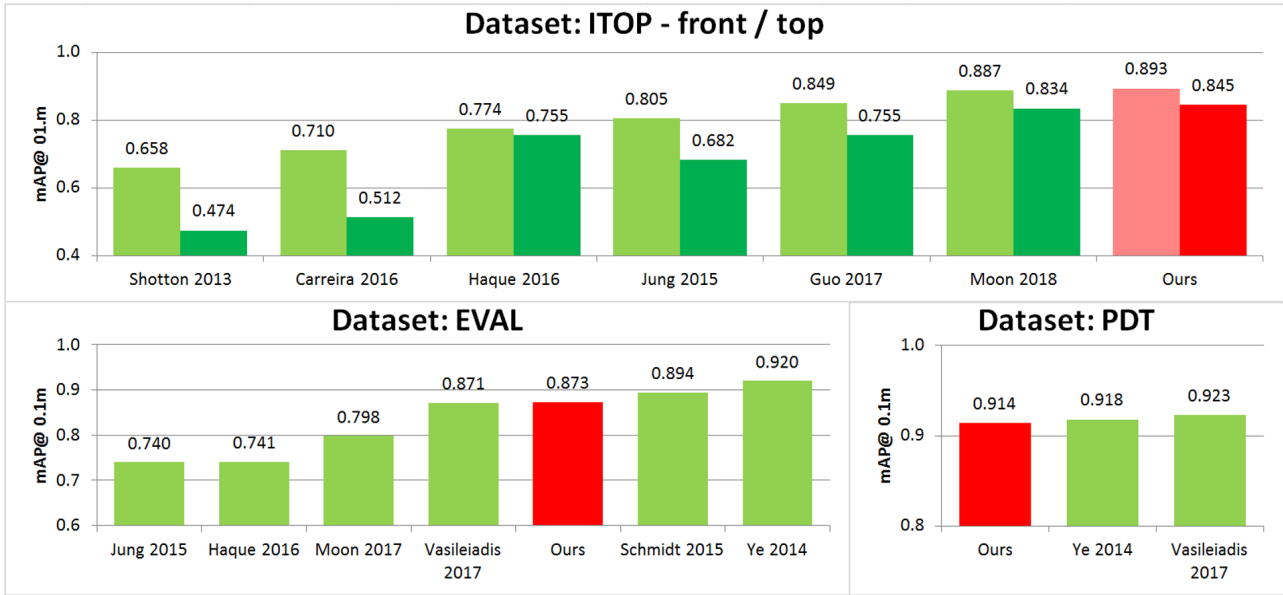


Figure 5.12: Comparison ($mAP@0.1m$) of the single person 3D pose estimation architecture to-state-of-the-art-methods on the ITOP - front, ITOP - top, EVAL and PDT datasets (results taken from the corresponding papers)

Figure 5.12 shows the recognition accuracy comparison between the proposed architecture and current state-of-the-art methods (as reported by their authors). From the comparative experimental results it becomes evident that the proposed framework performs comparably to other state-of-the-art methods on all three datasets.

On the ITOP dataset it outperforms the 2D-CNN [Carreira et al., 2016, Guo et al., 2017, Haque et al., 2016b] and random forest [Jung et al., 2015, Shotton et al., 2013] approaches on both the front and top subsets, showcasing the advantages of 3D representation over 2D-based approaches. Moreover, it outperforms, in both cases, the 3D-CNN based V2V-PoseNet [Moon et al., 2018] without requiring the segmentation of the input cloud around the human.

On the EVAL dataset the proposed method outperforms all single shot estimators [Haque et al., 2016b, Jung et al., 2015, Moon et al., 2017], but fails to reach the recognition accuracy levels of state-of-the-art articulated trackers [Schmidt et al., 2015, Ye and Yang, 2014], which however require skeleton initialization and background/foreground segmentation to function properly, thus hindering their applicability in realistic conditions. This can be attributed

to the small size and diversity of the training set (5K frames compared to the 20K frames of the ITOP dataset) which does not seem sufficient to adequately train the model. This notion is further justified by the fact that there is a large discrepancy in the overall accuracy among the actors (82.92% for actor A, 91.51% for actor C) indicating a significant gap in motion complexity between the sequences.

Similar results are observed on the PDT dataset, with the proposed method matching the two state-of-the-art articulated trackers [Vasileiadis et al., 2017b, Ye and Yang, 2014]. In contrast to the EVAL dataset, the training scheme used in this case, mixing sequences of all the actors in each training subset, along with the larger amount of available data, allows for the generation of more diverse training subsets leading to higher joint prediction accuracy.

5.8.2 Multi-Person 3D Pose Estimation

The multi person 3D pose estimation architecture is trained and evaluated on two multi-person sequences of the CMU PanopticStudio 3D PointCloud dataset: *Social Games: Hagglng* and *Musical Instruments: Band*. For the *Hagglng* sequences, *160422_hagglng1*, *160226_hagglng1* are used for training and *160224_hagglng1* for testing. Similarly for the *Band* sequences the training set is composed of *160906_band2*, *160906_band3* while *160906_band1* is used for evaluation. Table 5.3 below presents the detailed results from the experimental evaluation on the dataset, while some qualitative results can be found in Figure 5.13.

The algorithm surpasses the 98% and 96% mAP thresholds for the *Hagglng* and *Band* sequences respectively, clearly showcasing the proposed architecture’s potential towards high accuracy multi-person 3D pose estimation. The large increase in accuracy, compared to the single person 3D pose estimation experiments, can be partially attributed to some of the features of the CMU dataset, such as the large number of available data, the very high quality and consistency of the ground truth annotations, the lack of self-occlusions due to the overlapping setup of the capturing system, and the relatively lower motion complexity, as the actors simulate more natural social interactions, instead of the high complexity motions (i.e. backflips, handstands etc.) observed in the single person datasets.

Table 5.3: Experimental evaluation ($mAP@0.1m$) of the multi-person 3D pose estimation architecture on the *Haggling* and *Band* sequences of the CMU PanopticStudio 3D PointCloud dataset

	HAGGLING	BAND
Head	0.990	0.992
Neck	0.996	0.989
Shoulders	0.994	0.991
Elbows	0.991	0.985
Hands	0.975	0.968
Hips	0.986	0.835
Knees	0.990	0.982
Feet	0.984	0.969
MEAN	0.988	0.960

Directly comparing these results to other state-of-the-art methods is not applicable, as this is the first and only, at the time of writing, published method for multi-person 3D person pose estimation from 3D data. Moreover, since the dataset used herein is a multi-view 3D dataset, without ground truth per-view visibility masks, it is not possible to modify any of the existing single person 3D pose estimation methods, in order to extract any meaningful comparative data. However, these results can be used as reference for future works of similar nature.

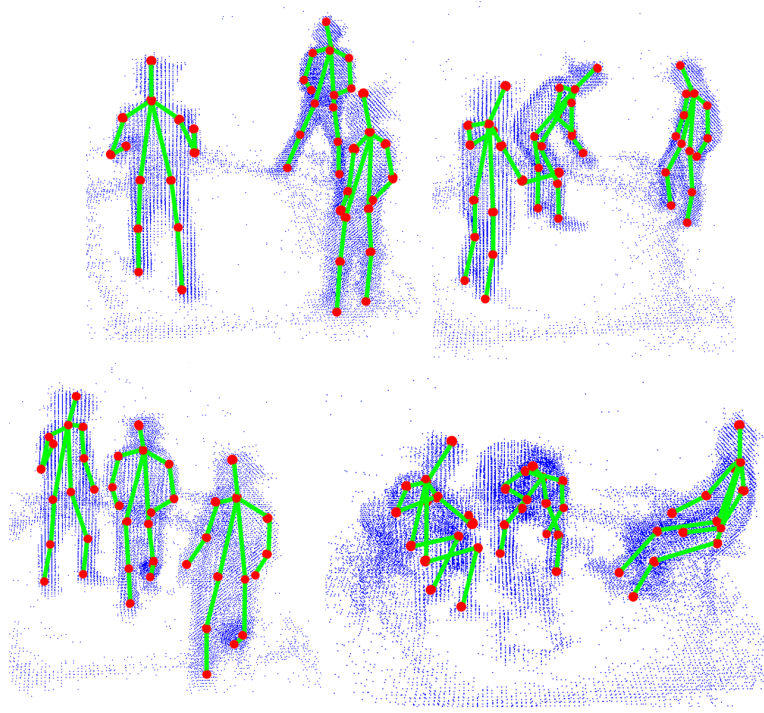


Figure 5.13: Qualitative results from experimental evaluation of the multi person 3D pose estimation architecture on the *Haggling* and *Band* sequences of the CMU PanopticStudio 3D PointCloud dataset. Background data have been removed for better visualisation

5.8.3 Inference

The proposed algorithm is implemented in Python, utilising the Chainer deep learning framework [Tokui et al., 2015]. Depth-to-point-cloud transformation and occupancy grid generation are performed on the CPU through a Cython [Behnel et al., 2011] compiled C module, while post-processing, including interpolation, non-maximum suppression and multi-pose composition, are executed on the GPU using CuPy [Okuta et al., 2017].

The final model includes $14.7M$ parameters, while the overall algorithm requires approximately $0.36s$ and $0.13s$ per frame for multi-person pose estimation, benchmarked on a mid-range NVIDIA GTX 970 (CUDA 7) and a high-end NVIDIA GTX 1080Ti (CUDA 10) respectively. A breakdown of the time complexity of the proposed method is presented in Table 5.4 below.

Table 5.4: Runtime analysis of all the individual components of the proposed methodology, benchmarked on the NVIDIA GTX 970 (CUDA 7) and 1080Ti (CUDA 10) GPUs.

	NVIDIA GTX	970	1080Ti
Grid Generation		5ms	5ms
Forward Pass		320ms	103ms
Feature Extraction		78ms	24ms
Stage $t=1$		46ms	15ms
Stage $t \geq 2$		49ms	16ms
Post Processing*		35ms	22ms
Total Runtime		360ms	130ms

*average for 5 people

5.9 Discussion

Besides the SoA accuracy reported in the previous section, the proposed architecture presents several advantages over methods that have preceded it, stemming from its DL-based, bottom-up, sequential design.

In terms of HPE robustness in realistic settings, some of the main advantages of the proposed method are:

- The bottom-up design does not require an initial detector to segment the data around potential RoIs (i.e. humans in the scene) [Moon et al., 2017, Moon et al., 2018]. This eclipses the risk of early commitment, where in the case of missed detections, when the initial detector fails, there is no way of recovery. Such issues can be observed in cases where only a small part of a human is visible, due to occlusions or out-of-FOV clipping (Fig. 5.14 - Left); cases that are frequently encountered in realistic multi-person scenarios. The proposed approach, on the other hand, can independently detect joints and body parts, even if they are not attached to a full skeleton.

- In the case of people in extremely close proximity (i.e. people hugging), it becomes possible for more than one corresponding joints to appear within a person's segmented volume (Fig. 5.14 - Right), which can lead to false joints association. Meanwhile, the proposed design is capable of dealing with multiple instances of the same joint within a grid, and along with the body part detector, successfully deals with these scenarios.
- Taking advantage of the increased learning capacity of the deep CNN architecture, it is able to provide accurate HPE from a single frame, without any type of data segmentation, in contrast to template trackers [Ye and Yang, 2014, Schmidt et al., 2015] which require relatively clean, segmented foreground data, as well as a rough estimation of the last detected human pose.

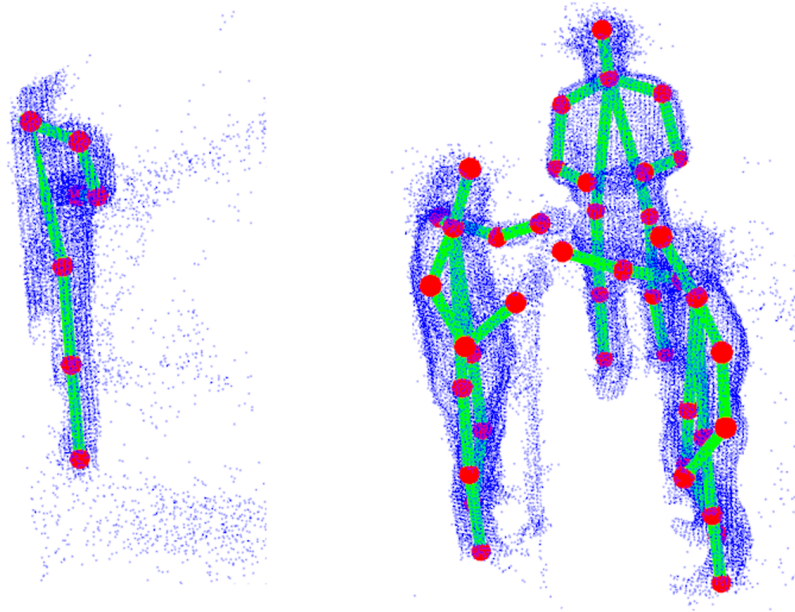


Figure 5.14: **Left:** Out-of-FOV clipping: the partial skeleton pose is estimated, even though the whole human is not visible and may not be detected. **Right:** Limbs in close proximity: the algorithm can handle multiple instances of the same joint (right hand) in close proximity, correctly assigning each detected joint to the corresponding skeleton

Moreover, multiple advantages can also be observed towards the potential optimisation of the architecture in terms of computational complexity and efficiency:

- Single person pose estimators and trackers [Moon et al., 2018] require the algorithm to be executed once for each detected person, thus linearly increasing the overall compu-

tational cost with each person added in the scene. On the other hand, the proposed bottom-up design remains practically unaffected by the number of people in the scene; the forward pass is not affected by the contents of the input grid, while the multi-person pose composition step that follows it is computationally insignificant, even for multiple people (Section 5.8.3)

- The joint and body part detection branches run independently of each other at every sequential stage, providing a highly parallelisable implementation, which can take advantage of the increased number of cores in modern GPUs. Moreover, this design aspect allows the straightforward transformation of the network for more limited single person cases (when desirable) by simply discarding the body part detection branch.
- The sequential architecture is easily parametrisable, as it allows the modification of the accuracy/complexity trade-off even after the definition and training of the network. During inference it is possible to utilise only few of the sequential stages, greatly reducing the computational complexity at the cost of accuracy, efficiently deploying the model on systems with different computational resources.

5.9.1 Comparison to the Template-Based Tracker

One of the initial motivations for the development of a single-shot pose estimator, was its implementation complementary to the template-based tracker, in place of the Kinect v1 built-in skeleton tracker (section 4.8), as earlier discriminative HPE approaches failed to reach the accuracy of generative trackers on their own. The high accuracy results, however, achieved by the proposed 3D CNN architecture, along with the design advantages presented above, demonstrate that it can be deployed on its own, providing robust performance under unconstrained realistic conditions.

Pose Estimation Accuracy

Both approaches are experimentally evaluated on the EVAL and PDT single person datasets. In terms of absolute average accuracy, the template-based tracker outperforms the 3D CNN by approximately 2cm *ADE*, on both datasets. This difference can be mainly attributed to the lower resolution of the generated 3D heatmaps, which adds a constant error to the final predicted positions. Moreover the presence of outlier detections further increases the average accuracy error, while in the case of the tracker, the use of the human template limits the maximum deviation of the estimated locations of the joints.

When using the *mAP* measure however, which provides a better indication about the method’s robustness, both approaches report similar accuracy, which along with the advantages described earlier, indicate that the proposed 3D-CNN architecture is the superior solution for the task in hand.

Table 5.5: Accuracy comparison of the Template-based tracker and the 3D-CNN estimator on the EVAL and PDT datasets.

	EVAL		PDT	
	ADE	mAP (0.1m)	ADE	mAP (0.1m)
Template-based Tracker	0.054	0.871	0.043	0.923
3D-CNN Estimator	0.077	0.873	0.062	0.914

Computational Complexity

The two proposed methods employ radically different architectures towards the same goal. The template-based tracker utilises very large pre-computed lookup tables for the individual SDFs of the rigid body parts, reducing the amount of necessary compute during the pose vector optimisation, in favor of multiple accesses to these tables. Taking advantage of the GPU’s extremely fast, read-only Texture Memory, which is especially efficient for spatially-localised accesses in 2D arrays, it achieves very high framerates despite the large size of the SDF Tables.

Table 5.6: Computational complexity estimation of the template-based human pose tracker, for 3000 3D input points, optimisation iterations $it_{max} = 10$ and linesearch steps $ls = 20$.

	FLOPs	MEMs	Params
Main SDF Tracker	655.0 M	319.2 M	125 M
Free Space Violation	157.0 M	165.0 M	
Body Part Visibility	0.1 M	0.3 M	
Leg Intersection	9.5 M	5.5 M	
Tracking Failure Detection	0.6 M	0.8 M	
TOTAL	822.2 M	490.8 M	125 M

The deep 3D-CNN architecture, on the other hand, requires significantly more floating point operations, despite the fact that the actual network parameters are less than half of the values in the SDF lookup tables. The convolution operations on the large 3D feature maps at each layer require a large number of multiplications and additions, along with memory access operations to read and write the data.

This large discrepancy in computational requirements becomes evident in practise as well, with the template-based tracker performing estimation almost 7x faster than the 3D-CNN estimator. Similar results are also observed when comparing against a simplified “single person” configuration which uses only the joints detection branch and a smaller 88^3 input grid.

Table 5.7: Computational complexity comparison between the template-based human pose tracker and 3D CNN estimator.

	FLOPs	MEMs	Params	Runtime GTX 970
Template-based Tracker	0.82 G	490 M	125.0 M	0.05 s
3D-CNN	755.12 G	406 M	14.7 M	0.32 s
3D-CNN Single	211.42 G	129 M	7.6 M	0.13 s

Potential Optimisation for Low Power Devices

Analysing the comparative results presented above, it may seem that the template-based tracker is a more attractive approach for implementation on low power devices, as it achieves similar accuracy to the 3D CNN architecture, at a significantly lower computational cost. However, the design presents two major limitations which significantly impact its optimisation potential:

- The template-based tracker achieves a relatively low FLOP count, by substituting computationally expensive operations with multiple table lookups to the precomputed SDFs. Fully exploiting the speed of the GPU's specialised texture memory, it achieves close-to-real-time performance. This design feature, however, is hardware specific, limited to discrete GPUs, and cannot be easily generalised towards devices with different architectures, such as modern mobile or integrated GPUs, which do not have dedicated memory, but rather share the much slower system memory. This limitation can lead to performance bottlenecks, negatively impacting the estimation inference.
- As discussed in section 4.6, it is essential that the tracking process begins from a pose close enough to the actual pose of the human, since the objective function is not convex and may otherwise converge to a minimal function value that does not correspond to the actual human pose. In order to achieve successful convergence two requirements must be met: a) the last tracked pose must be accurate and b) the delay between the last pose and the current frame must be small enough, to ensure that the current pose has not drastically changed compared to the last tracked pose.

Quantitatively estimating the latency and accuracy thresholds for convergence to the true minima is not a straightforward process, as it is affected by various factors, including but not limited to the specific position of the joints, the speed of the movement, the size of the body parts and more. However, it becomes evident that the tracking process must be both accurate and fast¹, which can be extremely challenging to achieve in systems with low computational capacity. Achieving the optimal performance/accuracy trade-off

¹Preliminary experimentation has shown that a framerate of at least 15fps seems to be necessary for consistently tracking relatively fast-paced movements

means maintaining one at the cost of the other; an accurate tracker will be potentially slow, leading to loss of accuracy, while a fast tracker will be potentially less accurate, accumulating the joint localisation error across subsequent frames.

The proposed deep-learning based approach, on the other hand, while computationally intensive in its current form, has the potential for optimisation in terms of efficiency, without presenting any hard accuracy or speed thresholds, since each frame is independently processed.

The 3D-CNN architecture was developed mainly with pose estimation accuracy in mind, and as such is relatively over-parameterised in order to outperform SoA methods. Employing the efficient network design techniques, presented in section 2.2.2, can largely reduce the computational complexity with a minimal drop-off in accuracy. Additionally, since these technique are hardware-agnostic, the optimised network design can be easily deployed in a variety of system architectures, while the multitude of available hardware-optimised deep-learning acceleration frameworks [Chetlur et al., 2014, Abadi et al., 2016, Venieris and Bouganis, 2016] can also contribute towards this goal.

5.10 Summary

In this chapter, deep-learning based approaches were investigated towards human pose estimation from depth/3D data, in order to overcome some of the inherent limitations of the template-based tracker presented in chapter 4.

First, a preliminary experimental evaluation of the SoA RGB-based OpenPose [Cao et al., 2017] multi-person pose estimator is performed, in order to explore its applicability on depth data. The first layer of the network is modified, to take as input single channel 2D depth maps, with the predicted 2D joint positions projected to 3D space to provide the 3D locations. The experimental results on the EVAL [Ganapathi et al., 2012b] and ITOP [Haque et al., 2016b] single person datasets, reveal that OpenPose achieves very high accuracy scores on the 2D image plane, with a noticeable drop-off in 3D localisation accuracy, which can be partially

attributed to the limitations of the depth-to-3D projection model. Compared to SoA methods, it outperforms approaches that utilise only the 2D information of depth images, however in terms of accuracy in 3D space, while the results are promising, it falls behind pure 3D-based methods [Moon et al., 2018, Ye and Yang, 2014], indicating that 3D point clouds provide a more natural scene representation, richer in geometrical features.

Following these preliminary results, and taking into consideration recent works that have utilised 3D data representation, yielding noticeable improvements in various computer vision tasks [Maturana and Scherer, 2015, Moon et al., 2018, Yuan et al., 2018a], a novel 3D-CNN architecture for 3D human pose estimation from depth/3D data is proposed. The network takes as input a 3D voxel grid and outputs 3D likelihood maps for each joint. The proposed method does not segment the input into smaller regions of interest, as in the case of similar 3D-CNN architectures [Moon et al., 2017, Moon et al., 2018], but instead covers the majority of the effective Field-of-View of the depth sensor, allowing the more efficient utilisation of global context information and the direct extension of the method towards multi-person pose estimation.

More specifically, the network uses a shallow feature extraction stage [Maturana and Scherer, 2015], followed by a series of repetitive joint prediction stages [Wei et al., 2016] which sequentially refine the predictions. Additionally, inspired by the work of [Cao et al., 2017], the architecture is extended towards multi-person 3D pose estimation, by including a body part detection structure, which runs in parallel with the sequential multistage joint prediction architecture, allowing the simultaneous prediction of joints and body parts, in the form of body part likelihood maps. Finally, an efficient multi-person composition algorithm is employed to combine the individual detected body parts into full human poses. This bottom-up approach allows the algorithm to simultaneously estimate multiple human poses, without its runtime complexity being affected by the number of people within the scene.

The 3D CNN estimator is experimentally evaluated on 3 single person and one multiperson dataset. In the former, it outperforms all single shot estimators [Haque et al., 2016b, Jung et al., 2015, Moon et al., 2018], while performing comparably to state-of-the-art articulated trackers [Ye and Yang, 2014, Schmidt et al., 2015], which however require skeleton ini-

tialisation and background/foreground segmentation to function properly, thus hindering their applicability in realistic conditions. In the multi person dataset, meanwhile, the algorithm surpasses the 98% and 96% mAP thresholds for two selected multi person subsets, clearly showcasing the proposed architecture’s potential towards high accuracy multi-person 3D pose estimation.

Besides the SoA accuracy, the proposed design presents several advantages over methods that have preceded it, both in terms of pose estimation robustness and potential computational efficiency. A detailed comparison to the earlier developed template-based tracker (chapter 4), indicates that both methods achieve similar accuracy, with the 3D CNN estimator, however, not presenting any of the practical algorithmic limitations of the tracker. Furthermore, while in terms of computational complexity the tracker is more efficient, an in depth investigation of the two architectures reveals that the 3D CNN design presents more potential for optimisation in terms of total computational cost. This aspect is explored in the next chapter, as the main limitation of the proposed approach is its high computational cost, which makes its direct deployment in low power devices troublesome.

Chapter 6

3D CNN Architecture Optimisation for HPE on Low Power Devices

This chapter presents the optimisation process, in terms of computational complexity, of the design of the 3D CNN estimator, in order to reduce its total computational cost and allow its implementation on low power mobile devices. A series of efficient network design guidelines are explored, leading to significant speed-up at a very small accuracy cost, while the efficiency of the proposed optimised design is further demonstrated through extensive on-device benchmarking.

Parts of the contents of this chapter have appeared in [Vasileiadis et al., 2019a].

In the previous chapters, two different approaches were developed towards robust and accurate 3D human pose estimation and tracking from depth/3D data, including a template-based pose tracker and a 3D-CNN pose estimator. Both proposed methods achieved competitive accuracy results, in publicly available HPE datasets, successfully accomplishing the first half of the twofold goal of the current research, as defined in section 1.2. These high accuracy scores, however, have been achieved at the cost of increased computational complexity, since the methods' design was mainly driven by high pose estimation accuracy, with computational efficiency being often an afterthought. Since HPE on systems with limited computational resources constitutes

the other half of the research topic, the following chapter investigates design guidelines, which will reduce the computational cost of the developed methods, while maintaining competitive accuracy, in order support the implementation of a SoA human pose estimation algorithm on low power devices.

Following the architecture analysis in section 5.9, in which the superiority, in terms of accuracy, robustness and optimisation potential, of the 3D-CNN estimator over the template-based tracker was discussed, the work herein undertakes the investigation of efficient CNN design techniques, within the scope of 3D-CNNs, in order to produce guidelines for 3D-CNN design, that can be applied to the already established network and lead to a computationally efficient 3D-CNN architecture for human pose estimation from 3D data. Various design aspects are explored, such as efficient convolutional blocks, small kernel sizes and alternative data representation models.

The final design guidelines for improved accuracy/complexity trade-off are employed on the developed 3D CNN estimator, leading to a novel optimised 3D-CNN architecture for multi-person 3D human pose estimation from 3D data, which performs comparably to the baseline model and other state-of-the-art methods, while requiring significantly fewer computational resources. Moreover, extensive benchmarking on high and low-end hardware experimentally validates the proposed guidelines, with the optimised architecture achieving almost 100x speed-up over the baseline model, on two modern mobile devices.

Section 6.1 investigates how different efficient network design techniques affect the computational complexity and accuracy of the 3D CNN architecture and Section 6.2 explores the potential parameterisation of the overall architecture. Meanwhile, in section 6.3, the proposed optimised design is comparatively evaluated against the baseline model and other SoA methods, in terms of accuracy, computational cost and on-device latency. Moreover, the the design guidelines are further validated, through their utilisation within the scope of 3D object classification, achieving similar results.

6.1 Network Design Optimisation

The network design optimisation process involves the revision of the proposed 3D-CNN architecture for human pose estimation, presented in Chapter 5. Exploring computationally efficient alternatives for every individual layer of the baseline network, the final goal is to reduce its overall computational complexity, while maintaining its accuracy, thus achieving an optimal complexity/accuracy trade-off.

The baseline network employs a fully-convolutional 3D-CNN architecture, that uses as input a 3D voxel grid and produces per-voxel likelihood maps of human joints and body parts, for multi-person 3D human pose estimation from 3D point cloud data. The network layout is partially inspired from earlier relevant works for both 2D and 3D computer vision tasks [Maturana and Scherer, 2015, Cao et al., 2017]. Close inspection the individual layers' architecture (Figures 5.6) makes it evident that many of the local layer optimisation methods described in section 2.2.2 are not employed e.g. the baseline architecture lacks bottleneck units and residual connections, while convolution factorisation is also absent. Implementing such techniques should improve the efficiency of the 3D CNN, by maintaining a similar level of accuracy at a lower computational cost. A breakdown of the computational complexity of the baseline 3D CNN architecture is given in Table 6.1 below.

Table 6.1: Computational complexity breakdown of the baseline 3D CNN estimator

Dim	Layer	FLOPs	Params	MEMs
Feature Extraction				
112^3	conv 1x7x7x7x32	30.84 G	11 K	46.4 M
112^3	max pool 2x2x2 / s=2	0.05 G	-	50.6 M
56^3	conv 32x5x5x5x64	89.92 G	256 K	17.1 M
56^3	max pool 2x2x2 / s=2	0.01 G	-	12.6 M
28^3	conv 64x3x3x3x128	9.71 G	221 K	4.4 M
	Total	130.52 G	488 K	131.1 M
Stage t=1 (x2 streams)				
	conv 128x3x3x3x128	19.42 G	442 K	6.0 M
	conv 128x3x3x3x128	19.42 G	442 K	6.0 M
28^3	conv 128x3x3x3x128	19.42 G	442 K	6.0 M
	conv 128x1x1x1x128	0.719 G	16 K	5.6 M
	conv 128x1x1x1x14	0.079 G	2 K	3.1 M
	Total	118.12 G	2688 K	53.9 M
Stage t=2...5 (x2 streams)				
	conv 156x3x3x3x128	23.67 G	539 K	6.8 M
	conv 128x3x3x3x128	19.42 G	442 K	6.0 M
28^3	conv 128x3x3x3x128	19.42 G	442 K	6.0 M
	conv 128x1x1x1x128	0.719 G	16 K	5.6 M
	conv 128x1x1x1x14	0.079 G	2 K	3.1 M
	Total	506.48 G	11528 K	221.2 M
	OVERALL	755.12 G	14704 K	406.2 M

6.1.1 Optimisation Protocol

A protocol is established to consistently evaluate the effect in accuracy and computational complexity of each potential network design. Specifically, every alternative network architecture is trained from scratch and evaluated on the ITOP-front dataset [Haque et al., 2016b], following the same training guidelines as the baseline model (section 5.6). Moreover, each individual training process is manually supervised to ensure that it converges and the final loss is of the same order of magnitude for all architectures¹. Similarly, the *Mean Average Precision at 0.1m* (*mAP*) measure is used for evaluation, while all the three complexity metrics are also estimated for each architecture.

Overall, the goal at each optimisation step is to reduce the network’s total computational complexity, while maintaining a similar estimation accuracy compared to the previous step. In general, between the three complexity metrics, more weight is given to the reduction of the network FLOPs and number of parameters, since the number of memory access operations is mainly dictated by the size of the 3D feature volumes, rather than the specific layer layout.

In the following subsections, at every optimisation step, the alternative layer layout that achieves the optimal accuracy/complexity trade-off (highlighted in bold in the corresponding table), is carried to the next step.

6.1.2 Efficient Convolutional Building Blocks

The first step in the design optimisation process is to find efficient alternatives to the standard convolutions employed in the baseline 3D CNN architecture. Recent research work has proposed extremely small and fast architectures for various 2D vision tasks, based on depthwise separable convolutions [Sifre and Mallat, 2014] and their variants. These architectures combine four efficient network design techniques: small 3×3 kernels, bottlenecks, residual connections and factorised convolutions, achieving high accuracy scores at an extremely small computational

¹In practise the only actual change from the training of the baseline model was setting the learning rate to $\alpha = 0.045$ and linearly decaying it over the last 8 epochs

cost, targeting real-time performance on modern low power mobile devices.

Inspired by these research efforts, five computationally efficient convolutional blocks are investigated, following the Mobilenets [Howard et al., 2017, Sandler et al., 2018], Shufflenet [Zhang et al., 2018, Ma et al., 2018] and Bottleneck [He et al., 2016] models (Figure 6.1).

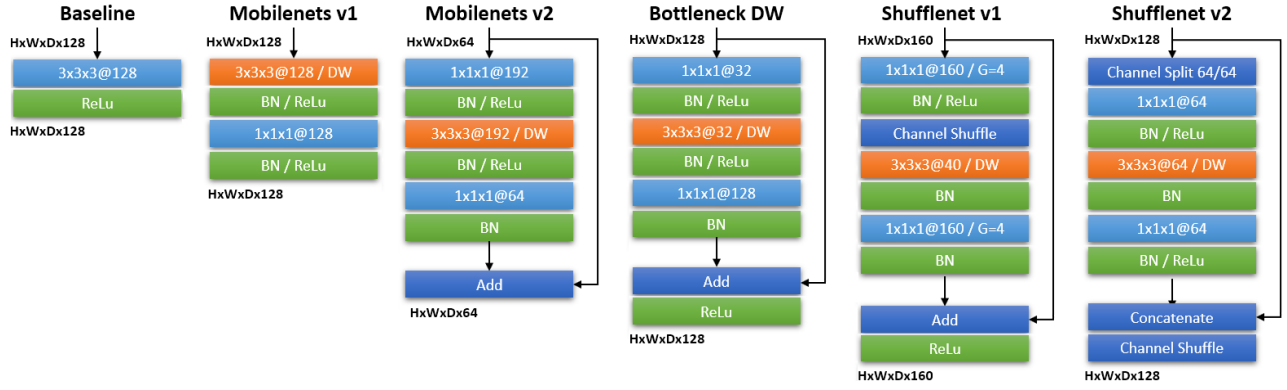


Figure 6.1: The basic convolutional building blocks investigated, based on the Mobilenets [Howard et al., 2017, Sandler et al., 2018], Shufflenet [Zhang et al., 2018, Ma et al., 2018] and Residual Bottleneck [He et al., 2016] architectures.

While in the corresponding papers, the authors propose specific backbone architectures based on these blocks, research on their direct implementation in 3D CNNs has been limited, without any optimal configuration available in literature. However, direct substitution of large blocks by smaller, more efficient ones is investigated by [Zhang et al., 2019], who propose the replacement of large stacks of hourglass [Newell et al., 2016] structures, by smaller ones both in terms of feature channels and number of structures, within the scope of knowledge distillation (KD) [Hinton et al., 2015]. Similarly, [Crowley et al., 2018] explore the use of cheaper grouped convolutions in place of standard ones, again within the KD framework, while [Ye et al., 2019] skip the KD process and directly replace standard 3D convolutions with basic 3D depthwise separable convolutional blocks and factorised Pseudo-3D convolutions [Qiu et al., 2017].

Building upon these research efforts, herein standard convolutions are directly substituted by the corresponding efficient blocks, maintaining the kernel sizes and channels widths of the baseline model. For the Mobilenets v2 and Shufflenet v1 blocks, the channel width is slightly altered in order to produce an architecture with a similar computational cost to the other three (Figure 6.1, Table 6.2). In cases where a layer is followed by strided Max Pooling, the

corresponding strided versions of the blocks are used, to maintain the spatial resolution of the baseline model. Moreover, for the blocks that include residual connections at the beginning of each joint prediction stage, a $1 \times 1 \times 1$ convolutional layer is added, since the input and output do not have the same number of channels. Finally, the first input layer, and the two final pointwise layers at the end of each stage, are not substituted, as they are investigated on their own in the following sections.

From the evaluation of the building blocks (Table 6.2), it becomes evident that all potential architectures achieve massive reductions, over 90%, in terms of FLOPs and parameters, while approaching the accuracy of the baseline model, with an average decrease in mAP of just 1.6%. Of the five investigated architectures, the Shufflenet v2 [Ma et al., 2018] building block not only achieves the highest accuracy score, but also presents the lowest complexity across all three metrics.

Table 6.2: mAP (0.1m) on the ITOP-front dataset and overall computational complexity for different efficient convolutional building blocks.

	mAP	FLOPs	Params	MEMs
Baseline	0.8923	755.12 G	14704 K	406 M
Mobilenets v1 [Howard et al., 2017]	0.8772	67.48 G	840 K	583 M
Mobilenets v2 [Sandler et al., 2018]	0.8769	67.76 G	1184 K	871 M
Bottleneck DW [He et al., 2016]	0.8765	61.82 G	707 K	662 M
Shufflenet v1 [Zhang et al., 2018]	0.8744	68.40 G	685 K	793 M
Shufflenet v2 [Ma et al., 2018]	0.8782	60.34 G	674 K	570 M

6.1.3 Input Layer

The first layer in the baseline model is comprised of a $7 \times 7 \times 7 \times 32$ convolution followed by strided Max pooling. Since it deals with the full resolution input grid, it accounts for approximately half of the total FLOPs of the model, after the previous optimisation step. Thus reducing its

complexity can significantly impact the overall network cost.

Towards this end, two potential blocks are investigated: a standard strided convolution, commonly used as a more efficient alternative to downsampling with convolution + pooling [He et al., 2016], and a depthwise separable convolutional layer (Mobilenets v1 strided variant), where the stride is employed on the depthwise convolution. Since the input data has only a single channel, the Mobilenets block practically decomposes the $7 \times 7 \times 7 \times 32$ layer into a single channel $7 \times 7 \times 7$ layer, followed by a pointwise convolution which projects the output onto the larger channel space. Meanwhile, the other four blocks from Section 6.1.2 are not employed, as they would require an extra expensive pointwise convolution at the full resolution of the input.

The strided convolutional layer, manages to slightly outperform the baseline layer, while reducing the total FLOPs in half. These results are in line with [Springenberg et al., 2014], where the authors argue that since pooling is a fixed operation, a strided convolution can learn it efficiently. Moreover, the Mobilenet v1 block achieves even greater complexity gains, while maintaining the same accuracy score, indicating that the decomposition of the first convolutional layer does not impact its learning capacity. Additionally, the potential use of a different data representation model, with more than one channels (see section 6.1.7), can further increase the Mobilenet v1 block’s computational gains.

Table 6.3: mAP (0.1m) on the ITOP-front dataset and overall computational complexity for different input layer layouts. s is the stride size.

	mAP	FLOPs	Params	MEMs
Convolution + Max Pooling / $s = 2$	0.8782	60.34 G	674 K	570 M
Convolution / $s = 2$	0.8830	33.28 G	674 K	479 M
Mobilenets v1 / $s = 2$	0.8841	29.56 G	663 K	480 M

6.1.4 Kernel Size

In the previous steps the kernel dimensions of the baseline layers are maintained in the depthwise convolutions. Meanwhile, in the original Mobilenets and Shufflenet architectures, the authors opt-out to use only kernel size $k = 3$ for the depthwise convolutions, along with the pointwise convolutions, as stacking 3×3 layers has been a staple of efficient network design [Simonyan and Zisserman, 2015] and most deep learning frameworks have custom, highly optimised implementations for these kernel types². The same principle is applied to the optimised architecture, specifically to the first two blocks, since the rest of the network only uses kernel sizes $k = \{1, 3\}$. Additionally, the use of dilated convolutions [Yu and Koltun, 2015] is investigated, attempting to maintain the receptive field of the original kernels.

The $3 \times 3 \times 3$ kernels achieve the same accuracy as the baseline kernels, slightly reducing the FLOPs and network parameters (Table 6.4). The dilated convolutional layers, on the other hand, result in a significant drop in accuracy, indicating that combining dilation with striding and depthwise convolutions degrades the quality of the extracted features.

Table 6.4: mAP (0.1m) on the ITOP-front dataset and overall computational complexity for different kernel size

	mAP	FLOPs	Params	MEMs
7x / 5x / 3x	0.8841	29.56 G	663 K	480 M
3x / 3x / 3x	0.8841	29.16 G	656 K	480 M
3x / 3x / 3x dilated	0.8454	29.16 G	656 K	480 M

6.1.5 End-of-Stage Pointwise Layers

At the end of each prediction stage two pointwise convolutional layers are employed, following [Long et al., 2015] where fully connected layers are substituted with pointwise convolutions in order to generate fully convolutional architectures. While pointwise convolutions are generally

²At runtime a 3×3 convolution is not going to be 9x slower than a 1×1 convolution

considered computationally efficient, in the current optimised architecture the second to last end-of-stage pointwise layers account for approximately a quarter of the total FLOPs and parameters, as they are applied on the full channel width (128 channels). Removing those layers results in a small decrease in accuracy of less than 0.7%, leading, however, to a significant 25% reduction in FLOPS and 13% reduction in MEMs, thus justifying this decision (Table 6.5)

Table 6.5: mAP (0.1m) on the ITOP-front dataset and overall computational complexity with and without the second-to-last pointwise layer at each detection stage.

	mAP	FLOPs	Params	MEMs
With Layer	0.8841	29.16 G	656 K	480 M
Without Layer	0.8782	21.94 G	491 K	419 M

6.1.6 Squeeze-and-Excitation Blocks

Moving in the opposite direction to the previous steps, the utilisation of *Squeeze - and - Excitation (SE) blocks* [Hu et al., 2018] is investigated, as a means of increasing the accuracy at a minor computational cost. SE blocks adaptively recalibrate the feature responses of each channel by modelling interdependencies between them, and have consistently demonstrated significant improvements in performance for various existing state-of-the-art CNN architectures.

Following [Ma et al., 2018], SE blocks with squeeze ratio $r = 4$ are added after the last BN/ReLU block in every Shufflenet v2 block (Figure 6.2), leading to an impressive 1.3% increase in accuracy (Table 6.6).



Figure 6.2: Integration of a Squeeze - and - Excitation block [Hu et al., 2018] inside a Shufflenet v2 basic block

Table 6.6: mAP (0.1m) on the ITOP-front dataset and overall computational complexity with and without the utilisation of Squeeze - and - Excitation blocks (SE) [Hu et al., 2018]. r is the SE block squeeze rate.

	mAP	FLOPs	Params	MEMs
No SE blocks	0.8782	21.94 G	491 K	419 M
SE blocks / $r = 4$	0.8913	22.22 G	558 K	514 M

6.1.7 Input Data Representation

The baseline model employs the computationally inexpensive Hitgrid volumetric representation to model the input data to a dense, fixed size 3D occupancy grid. This design decision was taken following preliminary research which indicated that it can achieve similar accuracy with more complex, ray-tracing based data representation models [Maturana and Scherer, 2015]. Recently, however, fast approximations of more complex models have been proposed in literature [Song and Xiao, 2016], which could potentially improve the overall accuracy at the cost of a relatively small increase in computational complexity, leading to a better accuracy/complexity trade-off.

Two low cost 3D data modelling approaches are investigated:

- **Projective Directional Truncated Signed Distance Function**

[Song and Xiao, 2016]: The projective directional TSDF model offers a fast ap-

proximation of the accurate TSDF model [Newcombe et al., 2011]. The directional TSDF stores a three-dimensional vector $[dx, dy, dz]$ in each voxel to record the distance in three directions to the closest surface point. It is further sped up in the projective variant, where the nearest point is found only on the line of sight from the camera. The truncation limit is set to $l = 5$ voxels, and voxel values are normalized to $[-1, 1]$, with positive values assigned to voxels in front of the surface, negative values to voxels behind the surface and zero values to voxels on the surface.

- **PointGrid** [Le and Duan, 2018]: An alternative to dense volumetric occupancy grids for 3D data modelling has been proposed recently in the PointNet [Qi et al., 2017] architecture, which consumes raw point clouds in the form of 3D coordinates without requiring voxelisation or rendering. While this approach has achieved competitive results in multiple 3D tasks, directly applying it on the develop 3D CNN would require the total overhaul of its architecture. Instead, the PointGrid model is employed, as it combines the dense volumetric representation with raw point cloud features. Specifically, the point cloud is normalised to the unit box $[-1, 1]^3$ and K 3D point coordinates are stacked as features at each voxel. For voxels with more than K points, random sampling is performed, while if the points are less, K of them are sampled with replacement. Meanwhile, empty voxels are padded with zeroes. The fixed number of points is set to $K = 6$, leading to a total of 18 values per voxel.

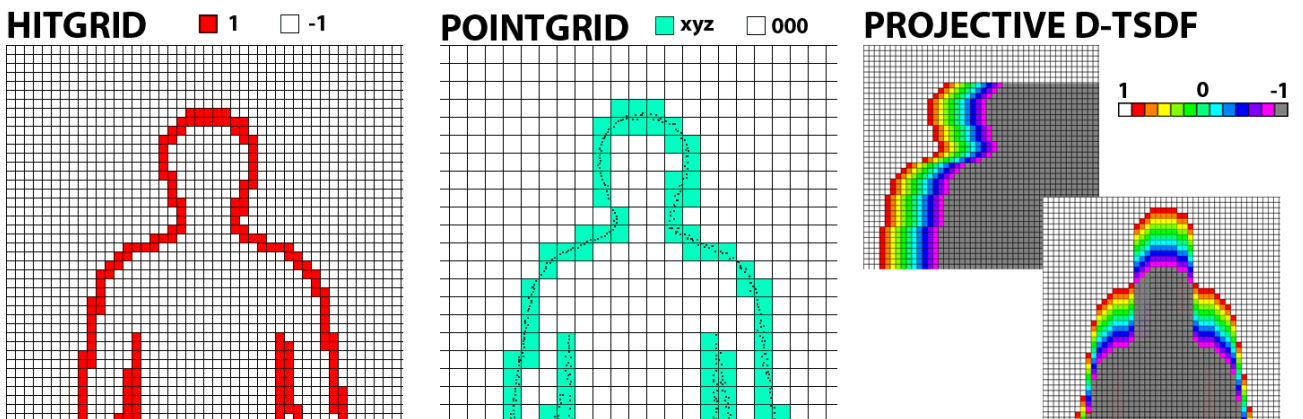


Figure 6.3: 2D illustrations of the *Hitgrid* [Maturana and Scherer, 2015], *PointGrid* [Le and Duan, 2018] and *projective D-TSDF* [Song and Xiao, 2016] 3D data representation models.

The experimental evaluation of the data representation models (Table 6.7) indicates that neither of the alternative options provides any meaningful improvements over the baseline HitGrid model. In the case of the projective D-TSDF, it can be partially attributed to the presence of obstacles that may block the projection beam. The pD-TSDF has been successfully utilised mainly in hand pose estimation tasks [Ge et al., 2018] where the data are clearly segmented.

For the PointGrid, on the other hand, the low number of 3D points around smaller body parts (hands, feet) results in randomly duplicating data which can make it harder for the convolution to learn new features. Reducing the number of points per pixel to $K = 4$ marginally improves the accuracy, practically reaching the performance of the baseline model, at a slightly higher, however, computational cost. Decreasing K may potentially achieve further improvement, however in this case the PointGrid would simply degenerate into a simple HitGrid. Meanwhile, reducing the spatial resolution of the occupancy grid results in a significant accuracy drop-off.

Table 6.7: mAP (0.1m) on the ITOP-front dataset and overall computational complexity for different data representation models. K is the fixed number of points per voxel, l is the truncation limit and D is the voxel grid spatial resolution .

	mAP	FLOPs	Params	MEMs
HitGrid / $D = 112$ [Maturana and Scherer, 2015]	0.8913	22.22 G	558 K	514 M
PointGrid / $K = 6, D = 112$ [Le and Duan, 2018]	0.8867	22.56 G	558 K	545 M
PointGrid / $K = 4, D = 112$	0.8896	22.44 G	558 K	530 M
PointGrid / $K = 6, D = 56^*$	0.8801	22.56 G	558 K	515 M
pD-TSDF / $l = 5, D = 112$ [Song and Xiao, 2016]	0.8853	22.26 G	558 K	514 M

*the stride on the first layer is set to $s = 1$

6.1.8 Design Guidelines Overview

Based on the study presented above, the design guidelines for improved accuracy/complexity trade-off in 3D-CNN architectures, are summarised as follows:

1. Hitgrid [Maturana and Scherer, 2015] volumetric data representation model
2. Mobilenets v1 [Howard et al., 2017] convolutional block for the first layer
3. Shufflenet v2 [Ma et al., 2018] convolutional blocks for the rest non-pointwise layers
4. Kernel size $k = 3$ for all non-pointwise layers
5. Strided convolution instead of pooling for downsampling
6. Removal of extra pointwise layers, besides last one
7. Squeeze and Excitation blocks [Hu et al., 2018]

While all of the above guidelines contribute towards achieving the optimal accuracy/complexity trade-off, it becomes evident from the experimental analysis described in the sections above, that two of these are the most effective:

- Replacing standard 3D convolutional layers with corresponding 3D Shufflenet v2 blocks, which reduce the FLOPs and parameters by over 90% while maintaining a comparable accuracy to the baseline model
- Using SE blocks, which practically restore the accuracy loss from the previous steps, at a fraction of the computational complexity cost. This also indicates that adding SE blocks to the baseline model could potentially further increase its accuracy.

6.2 Network Dimensionality Parameterisation

Following the optimisation of the layout of the individual layers, the overall accuracy/complexity trade-off can be further optimised through the uniform configuration of the networks's dimensionality. Specifically two dimensionality hyperparameters are introduced:

- **Width multiplier a** , which uniformly increases or decreases the number of channels at every layer of the network. The hyperparameter can take any positive float value $a \in R^+$,

- **Sequential stages t** , which defines the number of repetitive prediction stages in the sequential multi-stage structure. The hyperparameter can take any positive integer value $t \in \mathbb{Z}^+$

The baseline configuration described above corresponds to $\{a = 1, t = 5\}$.

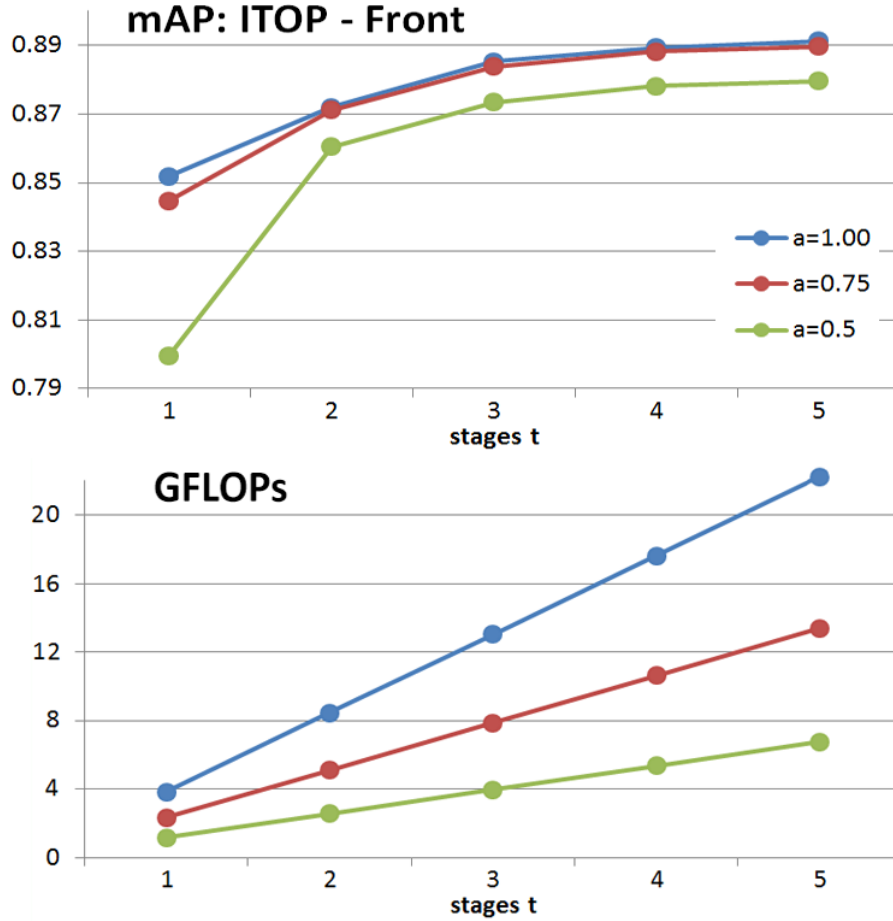


Figure 6.4: mAP (0.1m) on the ITOP-front dataset and computational complexity in gigaflops, for different width multiplier values a and number of sequential stages t

By evaluating all potential combinations for the a and t hyperparameters (Figure 6.4), the $\{a = 0.75, t = 4\}$ configuration achieves the best accuracy/complexity trade-off. It presents a minimal drop in accuracy of just 0.3% mAP compared to the baseline $\{a = 1, t = 5\}$ configuration, while requiring approximately half of the computational resources, in regards to all three computational complexity metrics. The detailed architecture of the final optimised version is presented in Figure 6.5 below.

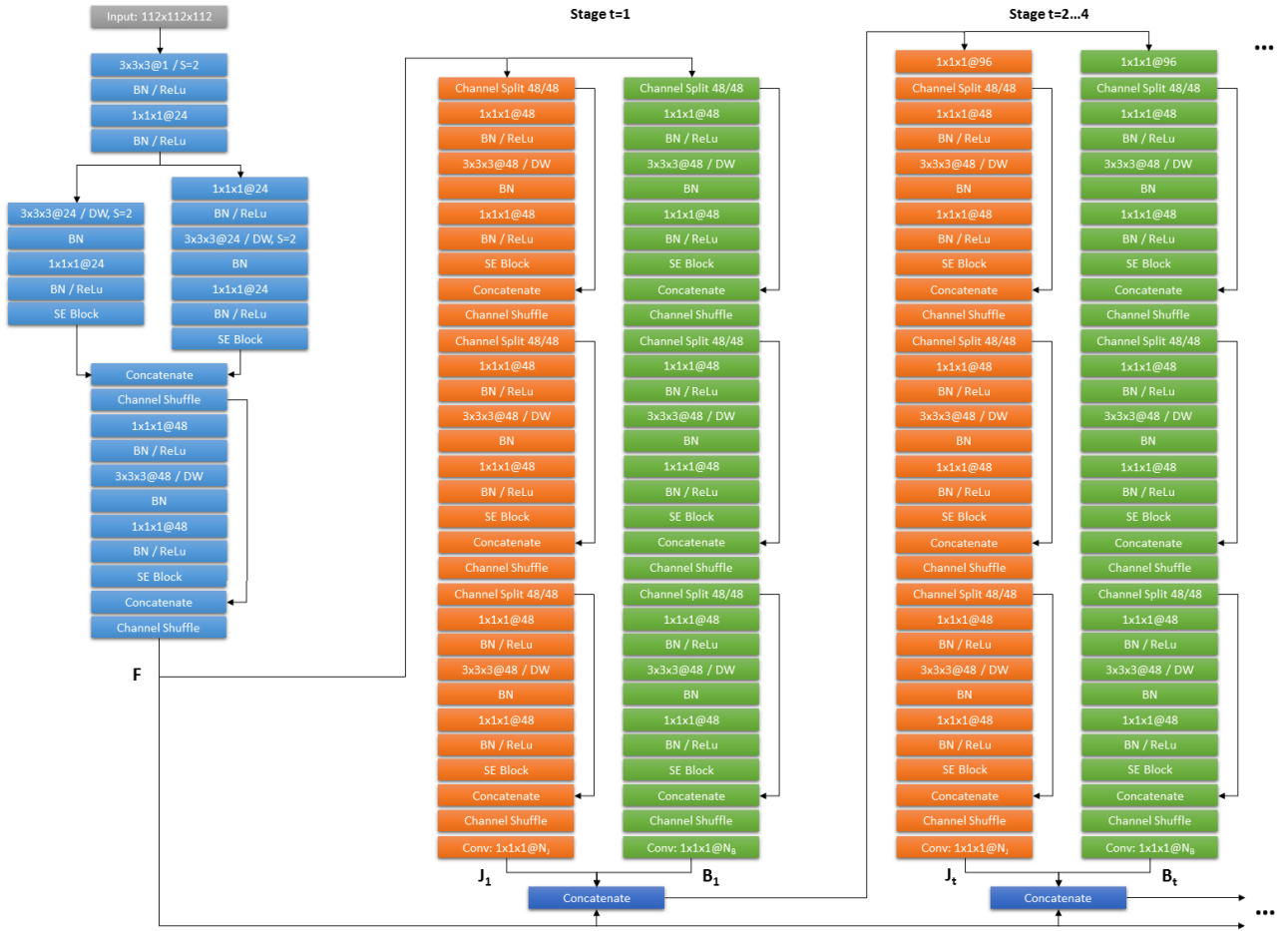


Figure 6.5: The final optimised version of the 3D CNN human pose estimation architecture

6.3 Experimental Evaluation

6.3.1 3D Human Pose Estimation

Following the same evaluation protocol used for the baseline 3D CNN estimator (see section 5.8), the final optimised architecture is evaluated on the single-person ITOP [Haque et al., 2016b] and multi-person CMU PanopticStudio (Band and Haggling sequences) [Joo et al., 2019b] datasets. The experimental results are presented in Table 6.8 below, while more detailed results are available in Appendix C.

Table 6.8: Comparison (mAP@0.1m) of the proposed optimised architecture to the baseline 3D CNN and state-of-the-art on the ITOP and CMU PanopticStudio datasets

	ITOP - front	ITOP - top	CMU Haggling	CMU Band
Head	0.983	0.981	0.983	0.981
Neck	0.986	0.983	0.995	0.981
Shoulders	0.967	0.957	0.991	0.975
Elbows	0.819	0.772	0.982	0.967
Hands	0.700	0.627	0.965	0.955
Torso	0.983	0.973	n/a	n/a
Hips	0.955	0.844	0.961	0.827
Knees	0.910	0.776	0.980	0.963
Feet	0.870	0.708	0.978	0.947
MEAN	0.888	0.820	0.978	0.945
3D CNN Baseline	0.893	0.845	0.988	0.960
[Xiong et al., 2019]	0.880	0.805	n/a	n/a
[Moon et al., 2018]	0.887	0.834	n/a	n/a
[Guo et al., 2017]	0.849	0.755	n/a	n/a

On the ITOP front and CMU Haggling subsets, which are of relatively lower motion complexity, the drop in accuracy, compared to the baseline architecture, is minimal (0.5% and 1% respectively), with the proposed optimised architecture actually outperforming the rest of the SoA CNN-based estimators [Xiong et al., 2019, Moon et al., 2018, Guo et al., 2017]. On the other hand, a relatively larger drop in performance is observed on the more challenging ITOP top and CMU Band subsets (2.5% and 1.5% respectively). However, the overall accuracy of the optimised architecture is comparable to the state-of-the-art, approaching the accuracy of [Moon et al., 2018] and outperforming [Guo et al., 2017] and [Xiong et al., 2019]. These results further demonstrate the efficiency of the proposed network design, validating the study carried out in sections 6.1 and 6.2 above.

6.3.2 3D Object Classification

In order to explore the generalisation potential of the proposed design guidelines in architectures other than the proposed 3D CNN estimator, their effectiveness is investigated within the scope of 3D object classification. Towards this end, a naive 3D extension of the VGG13 network [Simonyan and Zisserman, 2015] is employed as a baseline classifier and optimised following the proposed guidelines. Both the baseline and optimised architectures are then evaluated on the ModelNet10 subset of the ShapeNet dataset [Wu et al., 2015b], which includes 55k 3D CAD models, sampled as $32 \times 32 \times 32$ grids, split into 10 object categories.



Figure 6.6: The baseline VGG13 3D architecture. It takes as input a $32 \times 32 \times 32$ voxel grid, and predicts the label of the object depicted in the voxel grid

The optimised VGG13 3D architecture approaches the accuracy of the baseline model (Table 6.9), with the accuracy drop-off limited to 0.5%. Moreover, when compared to other state-of-the-art methods [Maturana and Scherer, 2015, Zhi et al., 2017, Sedaghat et al., 2017, Kumawat and Raman, 2019], it achieves competitive results. These results indicate that the proposed guidelines, even though developed targeting specifically the baseline 3D CNN HPE model, can be employed in other generic 3D tasks, improving the accuracy/complexity trade-off and providing a starting point towards finding the optimal architecture. The comparative evaluation is presented in Table 6.9 below.

VGG13 3D Optimised	VGG13 3D	[Maturana and Scherer, 2015]	[Zhi et al., 2017]	[Sedaghat et al., 2017]	[Kumawat et al., 2019]
0.918	0.923	0.920	0.934	0.938	0.944

Table 6.9: Accuracy comparison of the baseline and optimised VGG13 3D architectures, to state-of-the-art volumetric 3D-CNN models on the ModelNet10 dataset.

6.3.3 Computational Complexity Evaluation

The computational complexity of the optimised architecture is estimated and compared against the baseline 3D CNN estimator, as well as the state-of-the-art V2V Posenet architecture [Moon et al., 2018] which also employs 3D convolutional layers. The simplified “single person” configuration, described in section 5.9, is also included.

The reduction in FLOPs and network parameters, compared to the baseline architecture, is extremely large: over 70x and 55x respectively for the default multi-person configuration, while similar efficiency gains are also observed compared to [Moon et al., 2018]. The gains in terms of total memory access operations, on the other hand, are more limited, since the optimisation process does not significantly affect the spatial dimensions of the feature maps, which contribute to the bulk of MEMs. A moderate reduction of approximately 25% is achieved.

Table 6.10: Computational complexity comparison of the proposed optimised architecture, the baseline 3D CNN estimator and the state-of-the-art V2V Posenet by Moon et al. [Moon et al., 2018]

	FLOPs	Params	MEMs
3D CNN Optimised	10.64 G	264 K	315 M
3D CNN Baseline	755.12 G	14704 K	406 M
3D CNN Optimised Single	2.68 G	135 K	86 M
3D CNN Baseline Single	211.42 G	7596 K	129 M
[Moon et al., 2018]	72.78 G	3398 K	172 M

6.3.4 On-Device Benchmarking

Following the estimation of the theoretical cost of the proposed architectures, they are benchmarked on commercially available devices, in order to investigate how the gains in computational complexity, reported in the previous section, translate into real life performance. Five different hardware configurations are utilised covering a broad spectrum of high, mid and low power devices:

- Two discrete GPUs, a mid-range Nvidia GTX 970 and a high-end Nvidia GTX 1080Ti
- A high-end Intel Core i7-7700K x64 processor
- Two ARM based mobile chipsets, a mid-range Qualcomm Snapdragon 636 and a high-end Exynos 8895

To ensure consistent results across all devices, the Tensorflow r1.15 [Abadi et al., 2016] deep learning framework is employed³ and latency is measured as the average of 50 forward passes with a batch size $b = 1$. Moreover, for the multicore CPUs, both single core and multi core runtimes are reported. The technical details of the benchmarking configuration are presented in Table 6.11 below.

³While Tensorflow Mobile has been deprecated in favor of the mobile-optimised Tensorflow-lite, at the time of writing it was the only one that fully supported 3D convolutions

Table 6.11: Technical specifications of the benchmarking configuration

Device	Nvidia GTX 970	Nvidia GTX 1080Ti	Intel Core i7-7700K	Qualcomm SDN 636	Exynos 8895
Architecture	Maxwell	Pascal	x86-64	ARMv8-A	ARMv8-A
Cores	1664	3584	4 / 8	8	8
Clock (GHz)	1.050	1.481	4.2	1.8	2.3
Memory	4GB GDDR5	11GB GDDR5X	64GB DDR4	4GB LPDDR4	6GB LPDDR4X
OS	Ubuntu 18.04	Ubuntu 18.04	Ubuntu 18.04	Android 9	Android 9
DL Framework	TF r1.15	TF r1.15	TF r1.15	TF r1.15	TF r1.15
Backend	CUDA 7	CUDA 10		NDK r18b	NDK r18b

The detailed results from the runtime analysis on the desktop and mobile platforms are presented in Tables 6.12 and 6.13 respectively. A clear increase in speed is observed across all tested hardware configurations. Starting with moderate acceleration on the high-end GPUs, performance is dramatically increased as the computational resources of the platforms are reduced, with the observed relative improvement in speed even surpassing the theoretical improvement in FLOPs (Table 6.10). A similar decrease is also achieved in regards to the size of the final models, with the required storage space limited to approximately 1MB and 0.5MB for the multi and single person variants respectively. An overview of the average increase in runtime performance across all benchmark configurations is presented in Figure 6.7.

Table 6.12: Runtime in seconds, for a single frame inference, benchmarked on a desktop configuration. x1 and x8 correspond to single and multi core performance

	GPU		CPU x86-64		Model
	970	1080Ti	i7 x1	i7 x8	Size
3D CNN Baseline	0.3201	0.1032	9.7012	2.6163	56.09 MB
3D CNN Baseline Single	0.1303	0.0323	3.1633	0.8458	28.98 MB
[Moon et al., 2018]	0.1100	0.0216	2.1670	0.5418	12.96 MB
3D CNN Optimised	0.0297	0.0141	0.0871	0.0300	1.01 MB
3D CNN Optimised Single	0.0144	0.0079	0.0326	0.0146	0.51 MB

Table 6.13: Runtime in seconds, for a single frame inference, benchmarked on two mobile devices (Xiaomi Redmi Note 5 Pro and Samsung Galaxy Note 8). x1 and x8 correspond to single and multi core performance

	CPU ARMv8-A			
	SDN x1	SDN x8	EXN x1	EXN x8
3D CNN	111.620	26.320	85.730	19.260
3D CNN Single	37.760	8.971	28.680	6.325
[Moon et al., 2018]	19.460	3.745	14.680	3.261
3D CNN Optimised	0.655	0.246	0.479	0.168
3D CNN Single Optimised	0.241	0.124	0.181	0.081

On the high-end GTX 1080Ti GPU, the relative increase in inference speed is a moderate 7x and 4x for the multi and single person configurations, with the optimised 3D CNN estimator reaching a processing framerate of 70fps. The large number of available processing cores, along with the highly parallelizable nature of convolutions, enables the GPU to efficiently cope with high computational cost, as even the unoptimised version achieves a framerate of 10fps. On the other hand, the relatively small reduction in total memory access operations, mainly due to the extreme size of the 3D feature maps, limits any further gains in processing speed, indicating that memory access presents a performance bottleneck on high performance hardware, when

the amount of compute becomes extremely small.

The results, however, are dramatically different on the desktop CPU implementation. The limited amount of available processing cores, compared to a GPU, means that most of the floating point operations have to be performed in sequential fashion, making the FLOP metric the main indicator of real on-device performance. This assumption is reflected on the actual comparative runtimes, with the optimised versions outperforming the corresponding baseline ones by almost 100x in terms of speed, a performance gain ratio similar to the theoretical gain in terms of FLOPs. The high base clock along with the 8 working threads, result in the optimised model reaching the 30fps threshold when utilising all the available cores, with the framerate dropping to 12fps when deployed on a single thread.

Similar behaviour is observed in the low power ARM platforms, with the overall increase in speed being even larger, as the lower CPU clock struggles to keep up with the large amount of operations, increasing even more the effect of FLOP count on real time performance. The most extreme case is observed for single core operation on the mid-range Snapdragon 636 chipset, with the runtime for a single forward pass dropping from a massive 111s for the baseline 3D CNN to just 0.65s for the corresponding optimised version, an impressive 178x improvement. Overall, the high-end Exynos 8895 chipset provides an approximate 30% improvement over its mid-range counterpart, with the multicore implementation running at 6fps and 12fps for the multi and single person architecture respectively.

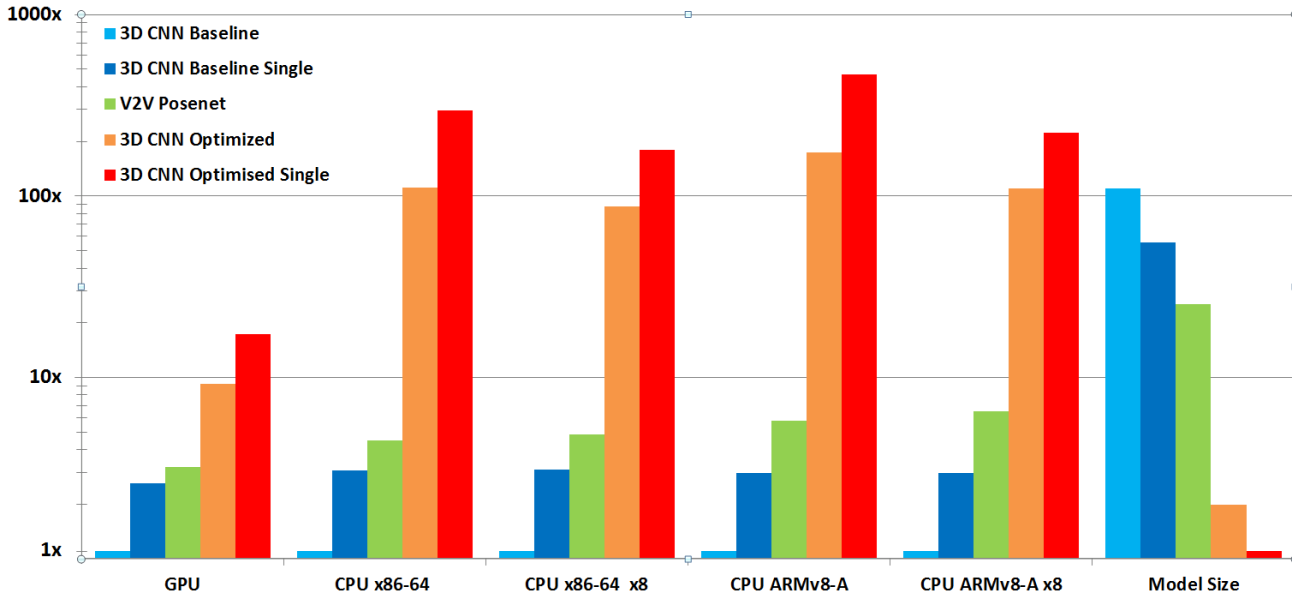


Figure 6.7: Average inference speed gain and model size reduction of the optimised architectures, relative to the baseline 3D CNN architecture, across all the benchmark devices

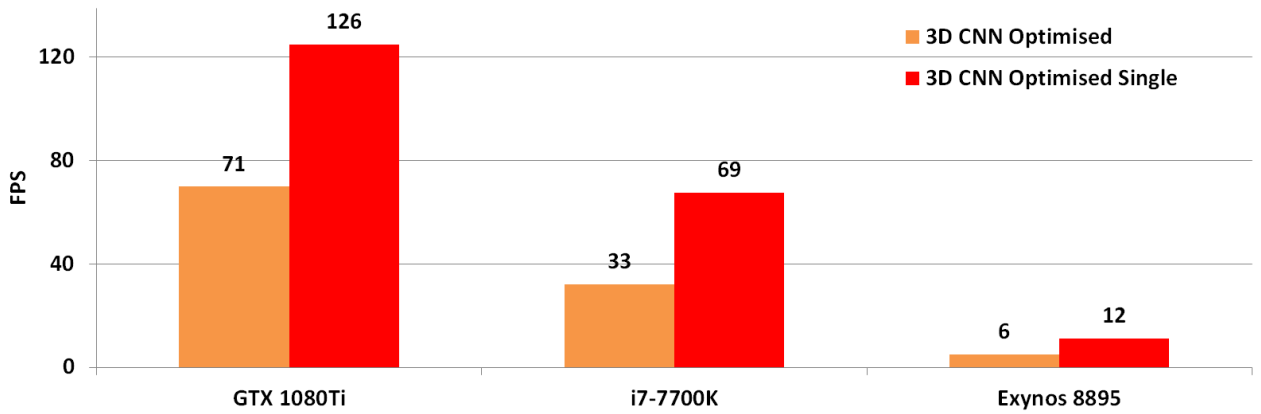


Figure 6.8: Maximum framerate of the optimised architectures on the GPU, CPU x86-64 and ARMv8-A benchmark platforms

The results from the benchmarking process presented herein, along with the high accuracy scores reported in section 6.3.1, experimentally validate the effectiveness of the proposed design guidelines in achieving the optimal accuracy/complexity trade-off. The optimised 3D CNN estimator presents minimal accuracy loss compared to the baseline design and other state-of-the-art methods, while simultaneously achieving massive gains in terms of runtime latency, which become even more evident when deployed in low power mobile devices. The observed framerates across all the devices (Figure 6.8) demonstrate the applicability of the proposed

HPE architecture, contributing towards the research goal of the Thesis for accurate human pose estimation in systems with limited computational resources.

6.4 Summary

This chapter presented the optimisation process of the developed 3D CNN estimator (see chapter 5), in order to reduce the architecture’s computational complexity, enabling its efficient deployment on low power devices. Aspiring to achieve the optimal accuracy/complexity trade-off, a series of efficient network design techniques are investigated and employed on the baseline 3D CNN model, with the goal of minimising the overall computational cost, while maintaining a comparable level of human pose estimation accuracy.

Initially, a protocol is established to consistently evaluate the effect in accuracy and computational complexity of each potential network design, according to which all alternative network layouts are trained from scratch on the ITOP-front dataset [Haque et al., 2016b], and are comparatively evaluated in terms of accuracy and complexity to the previous design. The first design aspect to be investigated is the replacement of the standard convolutional layers with alternative computationally efficient convolutional blocks. The Shufflenet v2 block [Ma et al., 2018] is found to provide the best performance. Its residual, depthwise, bottleneck architecture reduces the total number of FLOPs by over 90%, at the cost of an accuracy drop of just 1.5% mAP. Next, the first input layer is visited, where decomposing the convolution+pooling combo into a strided single channel convolution followed by a pointwise channel expansion layer, further cuts the FLOPs in half without any loss in accuracy. Small changes in terms of kernel size and layout of pointwise convolutions provide some incremental gains, while a significant boost in accuracy is achieved by the integration of Squeeze-and-Excitation Blocks [Hu et al., 2018] into the shufflenet structure. Finally, fast alternatives to the hitgrid occupancy grid are investigated, without however providing any meaningful improvements.

The novel optimised 3D CNN architecture is further configured by the introduction of two hyperparameters, which can uniformly increase or decrease the networks’s dimensionality: a

width multiplier a which controls the number of channels across all layers, and a sequential stages counter t , which defines the number of repetitive prediction stages.

With the final optimised architecture established, it is experimentally evaluated on the single person ITOP and multiperson CMU [Joo et al., 2019b] datasets. The overall accuracy scores are comparable the baseline model, as well as two other state-of-the-art DL-based human pose estimation methods. Moreover, in order to explore the generalisation potential of the proposed guidelines in other architectures, the task of 3D object classification is undertaken. The VGG13 [Simonyan and Zisserman, 2015] network is naively extended to 3D and employed as a baseline classifier. Next, it is optimised following the proposed guidelines, and evaluated on the ModelNet10 dataset [Wu et al., 2015b], with the classification accuracy drop limited to just 0.5%.

Following the experimental evaluation, the proposed architecture is benchmarked on various hardware configurations in order to explore how the theoretical improvements in computational complexity translate into real life performance. Two discrete GPUs, a high end desktop CPU and two mobile ARM-based chipsets are employed, covering a broad performance spectrum. A clear increase in speed is observed across all hardware configurations, with the high performance GPUs achieving a relatively moderate speed-up of 7x. However, in the case of the desktop and mobile CPUs, there is significant gain in inference latency. The low-core-count processors struggle to cope with the excessive amount of arithmetic operations required by the baseline model, and thus benefit from the significantly reduced number of FLOPs in the optimised 3D CNN. The relative increase in speed, compared to the baseline model, is over 87x and 114x for the desktop and mobile CPUs respectively, a performance gain ratio similar to the theoretical gain in terms of FLOPs.

Overall, the final optimised multi-person 3D CNN architecture achieves a framerate of 6fps on a modern mobile device, along with a small model size of just 1MB. These benchmark results, along with the high accuracy scores, experimentally validate the study that was performed, demonstrating a high accuracy human pose estimation architecture, efficiently implemented on a system with limited computational resources.

Chapter 7

Conclusions and Future Work

7.1 Thesis Summary and Achievements

This thesis investigates the task of human pose estimation (HPE) from 3D data on low power devices. It explores novel approaches for robust and accurate human pose estimation, along with computationally efficient algorithm design techniques that will enable the seamless deployment of these approaches on devices with limited computational resources. In terms of high accuracy HPE, the research focuses on two distinct methodologies for the task in hand; a template based tracker, which tries to fit an articulated human template to the input 3D data, through an iterative optimisation process, and a deep-learning based estimator, which utilises a deep 3D CNN architecture to generate 3D heatmaps corresponding to locations of the human joints and body parts within the scene. Regarding the algorithm design optimisation for low power devices, a series of hardware-agnostic efficient CNN design guidelines are explored, significantly reducing the complexity of the proposed 3D CNN estimator, while maintaining a comparable level of pose estimation accuracy.

Chapter 4 introduces a human-template based pose estimation and tracking algorithm. Employing an articulated skinned human template, composed of 15 rigid body parts and 3000 vertices, it tracks the human pose in sequences of 3D point clouds, attempting to find the optimal pose that best matches the template to the input data. The articulated signed distance function

representation is utilised, replacing explicit nearest point computation with faster lookup in the SDF tables. With each rigid body part of the human template defined implicitly by an SDF, the global SDF is approximated as a composition of the body parts' local SDFs, which need to be computed only once and do not have to be updated with the pose. The optimal pose is estimated by minimising the objective function using the BFGS algorithm, which iteratively arrives at the minimal function value by approximating the Hessian matrix of second derivatives with less computationally expensive rank-one updates.

Furthermore, novel complementary tracking mechanisms are introduced, contributing towards the method's accuracy and robustness, by tackling issues often encountered during recording under real life conditions, such as body part visibility, free space violation and leg intersection. Moreover, tracking initialisation and failure detection are handled through the integration in the tracking framework of the Kinect v1 built-in pose estimator, while additional preprocessing of the input data improves the human/background segmentation through a floodfill-based algorithm which removes large objects the target human may interact with.

The experimental evaluation of the proposed approach clearly demonstrates its effectiveness, as it performs comparably to other state-of-the-art methods on two publicly available datasets, while slightly outperforming them in a third one. Moreover, in order to better observe the framework's performance under realistic settings, a novel realistic human motion dataset is captured, incorporating, in contrast to most public datasets, challenges often encountered in real life scenarios, such as large occlusions, out-of-FOV clipping, large diversity in human body shape and more. Evaluated against the Kinect built-in pose estimator on this data, the template-based tracker achieves significantly superior results.

The main limitations of this approach are common across all generative pose trackers and come down to the need for well segmented human/background data, to fit the template to, as well as an approximate initial pose to ensure that the optimisation process will not converge to a local minimum, but rather to the global one which represents the actual target pose. Both of these issues can significantly impact real life performance, especially in the case of fast camera movement, making the tracker overly dependent on the complementary estimator used for

initialisation.

Attempting to overcome these shortcomings, in chapter 5 deep learning approaches for HPE from 3D data are investigated. First, the state-of-the-art OpenPose network for 2D HPE is experimentally evaluated on depth images. Its failure, however, to outperform contemporary pure 3D data-based methods makes it apparent that operating in 3D space provides a more natural representation of the input data, rich in geometrical features and with potential for improved accuracy and robustness. Following these observations, a novel 3D-CNN architecture for 3D human pose estimation from 3D data is proposed. The model employs a sequential network design, with successive prediction architectures refining the predictions over multiple stages. The input 3D data are modelled as a dense binary occupancy grid, while the predicted joint locations are represented in the form 3D likelihood maps.

The 3D CNN architecture is further extended towards multi-person 3D pose estimation, by incorporating a binary body part detection structure, parallel to the sequential multistage joint prediction architecture. This enables the simultaneous prediction of both joints and body parts, which are then composed into full human poses. This single stage bottom-up design presents several advantages over typical two stage approaches that combine a single person pose estimator with an initial human detector. It allows the simultaneous estimation of multiple poses without its complexity impacted from the number of people, it does not propagate human detection errors to pose estimation and can handle densely packed senses with overlapping human poses.

The 3D CNN estimator is experimentally evaluated on three single person datasets, outperforming all state-of-the-art discriminative estimators, while also approaching the accuracy levels of generative trackers, without requiring any kind of human/background segmentation or pose initialisation. Moreover, employed on a recently published multi-person 3D point cloud dataset, it achieves extremely high accuracy scores.

These state-of-the-art results, however, come at high computational cost. Covering the majority of the effective area of typical depth cameras, the proposed architecture is limited to 10fps on a modern high-end GPU, with a multi-processor CPU-only implementation requiring over 2.5s

to process a single frame, rendering the proposed architecture suitable only for specialised high-performance hardware. Moreover, attempting to cover larger areas, in the form multiple non-overlapping volumes, linearly increases the inference latency.

Having established an accurate but computationally expensive 3D pose estimator design, chapter 6 undertakes the optimisation, in terms of computational complexity, of the baseline architecture in order to achieve an optimal accuracy/complexity trade-off. Following a hardware-agnostic approach, efficient CNN design techniques are investigated, towards the definition of guidelines for 3D-CNN design, which when applied to the baseline HPE network will lead to a computationally efficient 3D-CNN architecture that can be seamlessly deployed on low power devices.

Using the theoretical computational complexity metrics defined in chapter 3, each design optimisation step replaces blocks of the baseline architecture with computationally efficient alternatives, while maintaining a comparable level of accuracy. The final optimised architecture, composed mainly of depthwise residual bottleneck units, SE blocks and a decomposed strided input layer, achieves massive computational gains over the baseline model, with FLOPs and network parameters reduced by almost 98%, at a very small overall drop in accuracy of just 1.3%.

Extensive on-device benchmarking experimentally validates the theoretical complexity improvements. Implemented on a broad spectrum of hardware configurations, ranging from high end GPUs to a mid-range low-power ARM chipsets, a clear increase in speed is observed across all devices. Starting with a moderate acceleration of 7x on the high-end 1080Ti GPU, performance is dramatically increased as the computational resources of the platforms are reduced, with the most extreme case observed in the single core implementation on the high-end Exynos 8895 ARM processor, where the proposed optimised architecture performs 170x faster from the baseline, and over 30x faster than the SoA V2V PoseNet single human pose estimator. Overall, a framerate of 6fps is achieved on the low-power Exynos 8895 platform, along with a compact model size of just 1MB, contributing towards the twofold research goal of the Thesis for accurate 3D human pose estimation in systems with limited computational resources.

7.2 Research Conclusions

The overall outcomes of the research efforts carried out for this Thesis can be summarised as follows:

- 3D data representation, in the form of a sparse 3D point cloud, is superior to 2D Depth maps for 3D human pose estimation. Both of the proposed HPE architectures, the template-based tracker and the 3D CNN estimators, clearly outperform current state-of-the-art RF [Shotton et al., 2013, Jung et al., 2015] and DL [Haque et al., 2016b, Carreira et al., 2016, Guo et al., 2017] methods which rely only on 2D depth images for HPE.
- Utilising a highly detailed articulated human template leads to higher accuracy scores, especially for more complex scenes. The proposed approach outperforms both [Ganapathi et al., 2012b] and [Ding and Fan, 2015], who approximate the human body as connected 3D capsules and anisotropic gaussian kernels respectively,
- Enhancing the main tracking algorithm with complementary mechanisms, each one tackling a discrete issue, increases the HPE accuracy and robustness in unconstrained settings. The introduced leg intersection and body part visibility features improve the prediction accuracy in the relevant joints by 12.7% and 2.3% mAP respectively. Moreover, the initialisation, failure detection and large object handling mechanisms allow the deployment of the proposed framework in unconstrained settings, addressing many of the general limitations of pose tracking algorithms.
- Deep Learning approaches, utilising 3D CNNs for 3D input data, reach the high accuracy levels of template-based trackers when provided with sufficient training data. The proposed 3D CNN estimator performs similarly to the template-based tracker on the EVAL [Ganapathi et al., 2012b] and PDT [Helten et al., 2013a] datasets (approximately 87% and 92% mAP respectively) without requiring any prior knowledge or segmented input data. This makes it more suitable for deployment in realistic conditions, as it can handle cluttered, non-segmented scenes and moving cameras.

- In contrast to 2D multi-person HPE, where top-down approaches [Xiao et al., 2018, Chen et al., 2018] achieve higher accuracy scores than bottom-up architectures [Cao et al., 2017, Papandreou et al., 2018], in 3D space the proposed bottom-up 3D CNN Estimator outperforms the state-of-the-art V2V Posenet model [Moon et al., 2018] by 1.7% mAP on the ITOP dataset [Haque et al., 2016b], despite not using a tightly segmented volume around the target person.
- The 3D CNN estimator architecture is not excessively wide or deep (the maximum number of channels is 156 and the total number of layers is 53), however it presents large computational complexity. The addition of an extra dimension in the generated feature maps and kernels, significantly increases the total amount of floating point and memory access operations, compared to a corresponding 2D architecture. As a result only high-end GPUs can utilise it effectively, as a desktop CPU requires approximately 2.5s for a single frame inference, while when tested on a low power ARM architecture, the inference latency surpasses 20s.
- Directly substituting standard 3D convolutions with corresponding factorised efficient blocks, significantly reduces the total computational cost for a minimal loss in accuracy. In the optimised 3D CNN estimator, replacing 3D convolutions with 3D Shufflenet v2 [Ma et al., 2018] blocks, results in a drop of 12x in FLOPs and 21x in network parameters, while reducing accuracy by only 1.4% mAP.
- The effectiveness and computational efficiency of SE [Hu et al., 2018] blocks, expand to 3D CNNs as well. Integrating SE blocks inside the 3D Shufflenet units, improves the accuracy by 1.3% mAP, while increasing the FLOPs of the optimised estimator by only 1.2%.
- The computationally inexpensive Hitgrid volumetric representation, outperforms fast approximations of more complex models, such as the projective D-TSDF model [Song and Xiao, 2016], and hybrid volumetric/3D point models [Le and Duan, 2018].
- The final optimised 3D CNN HPE architecture, achieves a framerate of 6fps on a modern ARM-powered mobile device, more than 110x faster than the baseline model, while still

achieving a comparable accuracy (average drop of 1.4% in mAP). This result showcases the applicability of 3D CNN architectures on modern low power devices, even for relatively large occupancy grids (i.e. 112^3). Moreover, it demonstrates, in practise, the effectiveness of using the framework agnostic complexity metrics to guide the network optimisation process. While in the case of the high-performance GPUs the actual speed-up is far lower than the theoretical one, for the low power devices which follow a more “traditional” architecture, the actual performance gains are of the same magnitude as the theoretical gains in FLOPs; a 71x reduction in FLOPs translates to 87x and 114x increase in inference speed on a desktop and mobile CPU respectively.

7.3 Future Work

The potential directions for future research follow the two main components of the research task: improving the human pose estimation accuracy and decreasing the overall computational complexity. Regarding the former, the following concepts can be explored:

- Combining the deep learning based estimator with the generative pose tracker, in a hybrid pose estimation approach, which can further increase the pose estimation accuracy. The pose estimator can either be used in a complementary way only for initialisation and tracking failure detection [Wei et al., 2012], or be incorporated inside the loss function [Ganapathi et al., 2010a] in the form of distance error between the estimated human pose and the current pose of the articulated template.
- Utilising RGB information along with the depth data, which can help to better detect individual body parts, especially in cases of overlapping poses, where the sampled 3D point cloud fails to provide distinct 3D edges. RGB images can either be used independently in a two stage system, proposing detections on the 2D image plane [Zimmermann et al., 2018], or in conjunction with the depth/3D data. For the template-based tracker, colour information can be encoded on each rigid body part, contributing towards a colour consistency

loss, while in the case of the 3D CNN architecture, normalised RGB values can be stacked along with voxel values in the volumetric occupancy grid.

- Employing a temporal coherence mechanism can result in more accurate and smooth results, reducing the impact of outlier pose estimations. In the case of the template tracker, while some temporal coherence is internally enforced since at each frame the human template is initialised from the previous pose estimation, there is no explicit temporal constraint and as a result there tends to be jittering of the joints' positions across consecutive frames. This issue can be addressed by incorporating a simple filtering approach, such as weighted averaging, on the temporal axis, or employing more complex motion models [Ganapathi et al., 2012b]. In the case of the 3D-CNN, on the other hand, where no temporal constraints are present at all, temporal coherence can be encouraged by stacking consecutive frames as input and training a temporal pooler which learns how to weigh the generated heatmaps from each frame [Pfister et al., 2015].

Towards further reducing the computational complexity and improving the accuracy/complexity trade-off, potential approaches include:

- Employing model compression techniques on the baseline and optimised 3D CNN architectures. Knowledge distillation [Hinton et al., 2015] can potentially reduce the loss in accuracy for network configurations with smaller values for the a and t hyperparameters, described in Chapter 6, resulting in an even shallower, faster architecture. Moreover, fine-tuning the trained models through quantisation-aware training [Jacob et al., 2018] to produce lower bit-depth networks, can significantly reduce inference latency on ARM-based mobile processors.
- Investigating the use of the ReLVP [Kumawat and Raman, 2019] block, an efficient alternative for 3D convolutions, which has recently achieved state-of-the-art accuracy in 3D object classification.
- Utilising the unstructured data representation of the PointNet model [Qi et al., 2017], which can potentially provide a more efficient solution for larger scenes, since increasing

the size of dense volumetric grids quickly becomes very computational intensive.

- Employing automated platform-aware neural architecture search on device (MnasNET [Tan et al., 2018]), which explicitly incorporates model latency, instead of the platform-agnostic FLOP metric, into the main objective so that the search can identify a model that achieves a good trade-off between accuracy and latency on the specific device.

Bibliography

- [Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.
- [Abba et al., 2010] Abba, A., Caponio, F., Geraci, A., and Ripamonti, G. (2010). Implementation of high efficiency non-linear least-squares in fpga devices for digital spectroscopy. In *IEEE Nuclear Science Symposium Conference Record*, pages 1371–1376. IEEE.
- [Abba et al., 2009] Abba, A., Manenti, A., Suardi, A., Geraci, A., and Ripamonti, G. (2009). Non-linear least squares fitting in fpga devices for digital spectroscopy. In *IEEE Nuclear Science Symposium Conference Record*, pages 563–568. IEEE.
- [Alan and Mark, 1992] Alan, W. and Mark, W. (1992). Advanced animation and rendering techniques. *Addison-Wesley*.
- [Amanatides et al., 1987] Amanatides, J., Woo, A., et al. (1987). A fast voxel traversal algorithm for ray tracing. In *Eurographics*, volume 87, pages 3–10.
- [Andriluka et al., 2014] Andriluka, M., Pishchulin, L., Gehler, P., and Schiele, B. (2014). 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on computer Vision and Pattern Recognition*, pages 3686–3693.
- [Andriluka et al., 2009] Andriluka, M., Roth, S., and Schiele, B. (2009). Pictorial structures revisited: People detection and articulated pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1014–1021. IEEE.

- [Athanasopoulos et al., 2011] Athanasopoulos, A., Dimou, A., Mezaris, V., and Kompatsiaris, I. (2011). Gpu acceleration for support vector machines. In *Workshop on Image Analysis for Multimedia Interactive Services*.
- [Baak et al., 2013] Baak, A., Müller, M., Bharaj, G., Seidel, H.-P., and Theobalt, C. (2013). A data-driven approach for real-time full body pose reconstruction from a depth camera. In *Consumer Depth Cameras for Computer Vision*, pages 71–98. Springer.
- [Balan et al., 2007] Balan, A. O., Sigal, L., Black, M. J., Davis, J. E., and Haussecker, H. W. (2007). Detailed human shape and pose from images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.
- [Barbareschi et al., 2015] Barbareschi, M., Del Prete, S., Gargiulo, F., Mazzeo, A., and Sansone, C. (2015). Decision tree-based multiple classifier systems: An fpga perspective. In *International Workshop on Multiple Classifier Systems*, pages 194–205. Springer.
- [Bauer et al., 2010] Bauer, S., Köhler, S., Doll, K., and Brunsmann, U. (2010). Fpga-gpu architecture for kernel svm pedestrian detection. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 61–68. IEEE.
- [Behnel et al., 2011] Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., and Smith, K. (2011). Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31.
- [Belagiannis and Zisserman, 2017] Belagiannis, V. and Zisserman, A. (2017). Recurrent human pose estimation. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 468–475. IEEE.
- [Belshaw and Greenspan, 2009] Belshaw, M. S. and Greenspan, M. A. (2009). A high speed iterative closest point tracker on an fpga platform. In *IEEE International Conference on Computer Vision Workshops*, pages 1449–1456. IEEE.
- [Bergstra et al., 2010] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: A cpu and gpu math compiler in python. In *Python in Science Conference*, pages 1–7.

- [Besl et al., 1992] Besl, P. J., McKay, N. D., et al. (1992). A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256.
- [Bilbeisi and Malik, 2019] Bilbeisi, G. and Malik, J. (2019). 3D Hand Pose Estimation using Depth RGB and RGBD Images. Technical report, Technische Universität Kaiserslautern.
- [Blake and Shiffrar, 2007] Blake, R. and Shiffrar, M. (2007). Perception of human motion. *Annual review of psychology*, 58.
- [Bogo et al., 2016] Bogo, F., Kanazawa, A., Lassner, C., Gehler, P., Romero, J., and Black, M. J. (2016). Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *European Conference on Computer Vision*, pages 561–578. Springer.
- [Bottou, 1991] Bottou, L. (1991). Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nimes*, 91(8).
- [Bourdev and Malik, 2009] Bourdev, L. and Malik, J. (2009). Poselets: Body part detectors trained using 3d human pose annotations. In *IEEE International Conference on Computer Vision*, pages 1365–1372. IEEE.
- [Bulat and Tzimiropoulos, 2016] Bulat, A. and Tzimiropoulos, G. (2016). Human pose estimation via convolutional part heatmap regression. In *European Conference on Computer Vision*, pages 717–732. Springer.
- [Bulat and Tzimiropoulos, 2017] Bulat, A. and Tzimiropoulos, G. (2017). Binarized convolutional landmark localizers for human pose estimation and face alignment with limited resources. *IEEE International Conference on Computer Vision*.
- [Bulat and Tzimiropoulos, 2019] Bulat, A. and Tzimiropoulos, G. (2019). Xnor-net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*.
- [Cadambi et al., 2009] Cadambi, S., Durdanovic, I., Jakkula, V., Sankaradass, M., Cosatto, E., Chakradhar, S., and Graf, H. P. (2009). A massively parallel fpga-based coprocessor for support vector machines. In *IEEE Symposium on Field Programmable Custom Computing Machines*, pages 115–122. IEEE.

- [Calderita et al., 2013] Calderita, L. V., Bandera, J. P., Bustos, P., and Skiadopoulos, A. (2013). Model-based reinforcement of kinect depth data for human motion capture applications. *Sensors*, 13(7):8835–8855.
- [Cao et al., 2017] Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Conference on Computer Vision and Pattern Recognition*.
- [Carpenter, 2009] Carpenter, A. (2009). Cusvm: A cuda implementation of support vector classification and regression. *patternsonscreen. net/cuSVMDesc. pdf*.
- [Carreira et al., 2016] Carreira, J., Agrawal, P., Fragkiadaki, K., and Malik, J. (2016). Human pose estimation with iterative error feedback. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4733–4742.
- [Catanzaro et al., 2008] Catanzaro, B., Sundaram, N., and Keutzer, K. (2008). Fast support vector machine training and classification on graphics processors. In *International Conference on Machine learning*, pages 104–111. ACM.
- [Charles et al., 2016] Charles, J., Pfister, T., Magee, D., Hogg, D., and Zisserman, A. (2016). Personalizing human video pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3063–3072.
- [Chen and Ramanan, 2017] Chen, C.-H. and Ramanan, D. (2017). 3d human pose estimation= 2d pose estimation+ matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7035–7043.
- [Chen et al., 2015] Chen, W., Wilson, J., Tyree, S., Weinberger, K., and Chen, Y. (2015). Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pages 2285–2294.
- [Chen and Yuille, 2014] Chen, X. and Yuille, A. L. (2014). Articulated pose estimation by a graphical model with image dependent pairwise relations. In *Advances in Neural Information Processing Systems*, pages 1736–1744.

- [Chen and Medioni, 1992] Chen, Y. and Medioni, G. (1992). Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155.
- [Chen et al., 2018] Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., and Sun, J. (2018). Cascaded pyramid network for multi-person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7103–7112.
- [Cheng and Bouganis, 2013] Cheng, C. and Bouganis, C.-S. (2013). Accelerating random forest training process using fpga. In *International Conference on Field Programmable Logic and Applications*, pages 1–7. IEEE.
- [Cheng et al., 2017] Cheng, Y., Wang, D., Zhou, P., and Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*.
- [Chetlur et al., 2014] Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., and Shelhamer, E. (2014). cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*.
- [Chimmalgi et al., 2019] Chimmalgi, S., Prins, P. J., and Wahls, S. (2019). Fast nonlinear fourier transform algorithms using higher order exponential integrators. *IEEE Access*, 7:145161–145176.
- [Chollet, 2017] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258.
- [Chu et al., 2017] Chu, X., Yang, W., Ouyang, W., Ma, C., Yuille, A. L., and Wang, X. (2017). Multi-context attention for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1831–1840.
- [Collobert et al., 2002] Collobert, R., Bengio, S., and Mariéthoz, J. (2002). Torch: a modular machine learning software library. Technical report, Idiap.

- [Comaniciu and Meer, 2002] Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619.
- [Courbariaux et al., 2014] Courbariaux, M., Bengio, Y., and David, J.-P. (2014). Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*.
- [Courbariaux et al., 2016] Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*.
- [Crowley et al., 2018] Crowley, E. J., Gray, G., and Storkey, A. J. (2018). Moonshine: Distilling with cheap convolutions. In *Advances in Neural Information Processing Systems*, pages 2888–2898.
- [Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893. IEEE.
- [D’Amore et al., 2015] D’Amore, L., Laccetti, G., Romano, D., Scotti, G., and Murli, A. (2015). Towards a parallel component in a gpu-cuda environment: a case study with the l-bfgs harwell routine. *International Journal of Computer Mathematics*, 92(1):59–76.
- [Darwin, 1872] Darwin, C. (1872). The expression of the emotions in man and animal.
- [Dawson,] Dawson, B. A simple benchmark of various math operations. <https://latkin.org/blog/2014/11/09/a-simple-benchmark-of-various-math-operations/>. Accessed: 2020-01-01.
- [De Sa et al., 2017] De Sa, C., Feldman, M., Ré, C., and Olukotun, K. (2017). Understanding and optimizing asynchronous low-precision stochastic gradient descent. In *Annual International Symposium on Computer Architecture*, pages 561–574. ACM.

- [Dean et al., 2012] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. (2012). Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231.
- [Demirdjian et al., 2003] Demirdjian, D., Ko, T., and Darrell, T. (2003). Constraining human body tracking. In *IEEE International Conference on Computer Vision*, pages 1071–1078 vol.2.
- [Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.
- [Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.
- [Ding and Fan, 2015] Ding, M. and Fan, G. (2015). Articulated gaussian kernel correlation for human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 57–64. IEEE.
- [Drozda and Sopyła, 2016] Drozda, P. and Sopyła, K. (2016). Accelerating svm with gpu: The state of the art. In *International Conference on Artificial Intelligence and Soft Computing*, pages 624–634. Springer.
- [Du et al., 2016] Du, Y., Wong, Y., Liu, Y., Han, F., Gui, Y., Wang, Z., Kankanhalli, M., and Geng, W. (2016). Marker-less 3d human motion capture with monocular image sequence and height-maps. In *European Conference on Computer Vision*, pages 20–36. Springer.
- [Everingham et al., 2007] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>.
- [Fang et al., 2017] Fang, H.-S., Xie, S., Tai, Y.-W., and Lu, C. (2017). Rmpe: Regional multi-person pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2334–2343.

- [Fang et al., 2018] Fang, H.-S., Xu, Y., Wang, W., Liu, X., and Zhu, S.-C. (2018). Learning pose grammar to encode human body configuration for 3d pose estimation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- [Farabet et al., 2009] Farabet, C., Poulet, C., Han, J. Y., and LeCun, Y. (2009). Cnp: An fpga-based processor for convolutional networks. In *International Conference on Field Programmable Logic and Applications*, pages 32–37. IEEE.
- [Fastovets et al., 2013] Fastovets, M., Guillemaut, J.-Y., and Hilton, A. (2013). Athlete pose estimation from monocular tv sports footage. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1048–1054.
- [Fei et al., 2014] Fei, Y., Rong, G., Wang, B., and Wang, W. (2014). Parallel l-bfgs-b algorithm on gpu. *Computers & Graphics*, 40:1–9.
- [Felzenszwalb and Huttenlocher, 2004] Felzenszwalb, P. and Huttenlocher, D. (2004). Distance transforms of sampled functions. Technical report, Cornell University.
- [Felzenszwalb and Huttenlocher, 2005] Felzenszwalb, P. F. and Huttenlocher, D. P. (2005). Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79.
- [Ferrari et al., 2008] Ferrari, V., Marin-Jimenez, M., and Zisserman, A. (2008). Progressive search space reduction for human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.
- [Fitzgibbon, 2003] Fitzgibbon, A. W. (2003). Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21(13):1145–1153.
- [Fletcher, 1976] Fletcher, R. (1976). Conjugate gradient methods for indefinite systems. *Numerical analysis*, pages 73–89.
- [Freund and Schapire, 1995] Freund, Y. and Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37. Springer.

- [Ganapathi et al., 2010a] Ganapathi, V., Plagemann, C., Koller, D., and Thrun, S. (2010a). Real time motion capture using a single time-of-flight camera. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 755–762. IEEE.
- [Ganapathi et al., 2010b] Ganapathi, V., Plagemann, C., Koller, D., and Thrun, S. (2010b). Stanford Time-of-Flight Motion Capture Dataset (SMMC-10). <http://ai.stanford.edu/~varung/cvpr10>. Accessed: 2020-01-01.
- [Ganapathi et al., 2012a] Ganapathi, V., Plagemann, C., Koller, D., and Thrun, S. (2012a). EVAL Dataset. <http://ai.stanford.edu/~varung/eccv12>. Accessed: 2020-01-01.
- [Ganapathi et al., 2012b] Ganapathi, V., Plagemann, C., Koller, D., and Thrun, S. (2012b). Real-time human pose tracking from range data. In Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., and Schmid, C., editors, *Computer Vision -- ECCV 2012. ECCV 2012. Lecture Notes in Computer Science, vol 7577*, pages 738–751. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [Garcia, 2010] Garcia, N. (2010). Parallel power flow solutions using a biconjugate gradient algorithm and a newton method: A gpu-based approach. In *IEEE Power and Energy Society General Meeting*, pages 1–4. IEEE.
- [Ge et al., 2017] Ge, L., Liang, H., Yuan, J., and Thalmann, D. (2017). 3d convolutional neural networks for efficient and robust hand pose estimation from single depth images. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 5.
- [Ge et al., 2018] Ge, L., Liang, H., Yuan, J., and Thalmann, D. (2018). Real-time 3d hand pose estimation with 3d convolutional neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):956–970.
- [Gkioxari et al., 2016] Gkioxari, G., Toshev, A., and Jaitly, N. (2016). Chained predictions using convolutional neural networks. In *European Conference on Computer Vision*, pages 728–743. Springer.
- [Gong et al., 2014] Gong, Y., Liu, L., Yang, M., and Bourdev, L. (2014). Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

- [Gonzalez-Jorge et al., 2015] Gonzalez-Jorge, H., Rodríguez-Gonzálvez, P., Martínez-Sánchez, J., González-Aguilera, D., Arias, P., Gesto, M., and Díaz-Vilariño, L. (2015). Metrological comparison between kinect i and kinect ii sensors. *Measurement*, 70:21–26.
- [Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.
- [Guo et al., 2012] Guo, C., Fu, H., and Luk, W. (2012). A fully-pipelined expectation-maximization engine for gaussian mixture models. In *International Conference on Field-Programmable Technology*, pages 182–189. IEEE.
- [Guo et al., 2017] Guo, H., Wang, G., Chen, X., and Zhang, C. (2017). Towards good practices for deep 3d hand pose estimation. *arXiv preprint arXiv:1707.07248*.
- [Gupta et al., 2015] Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. (2015). Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pages 1737–1746.
- [Han and Zhu, 2008] Han, F. and Zhu, S.-C. (2008). Bottom-up/top-down image parsing with attribute grammar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(1):59–73.
- [Han et al., 2015a] Han, S., Mao, H., and Dally, W. J. (2015a). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.
- [Han et al., 2015b] Han, S., Pool, J., Tran, J., and Dally, W. (2015b). Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143.
- [Han et al., 2015c] Han, X., Leung, T., Jia, Y., Sukthankar, R., and Berg, A. C. (2015c). Matchnet: Unifying feature and metric learning for patch-based matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3279–3286.

- [Hanson and Pratt, 1989] Hanson, S. J. and Pratt, L. Y. (1989). Comparing biases for minimal network construction with back-propagation. In *Advances in neural information processing systems*, pages 177–185.
- [Haque et al., 2016a] Haque, A., Peng, B., Luo, Z., Alahi, A., Yeung, S., and Fei-Fei, L. (2016a). Invariant-Top View Dataset (ITOP). https://www.alberthaque.com/projects/viewpoint_3d_pose/. Accessed: 2020-01-01.
- [Haque et al., 2016b] Haque, A., Peng, B., Luo, Z., Alahi, A., Yeung, S., and Fei-Fei, L. (2016b). Towards viewpoint invariant 3d human pose estimation. In *European Conference on Computer Vision*, pages 160–177. Springer.
- [He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, pages 770–778.
- [He et al., 2017] He, Y., Zhang, X., and Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397.
- [Helten et al., 2013a] Helten, T., Baak, A., Bharaj, G., Muller, M., Seidel, H.-P., and Theobalt, C. (2013a). Personalization and evaluation of a real-time depth-based full body tracker. In *International Conference on 3D Vision*, pages 279–286, Washington, DC, USA. IEEE Computer Society.
- [Helten et al., 2013b] Helten, T., Baak, A., Bharaj, G., Muller, M., Seidel, H.-P., and Theobalt, C. (2013b). Personalized Depth Tracker Dataset (PDT). <http://gvpvperfcapeva.mpi-inf.mpg.de/public/PersonalizedDepthTracker/index.php>. Accessed: 2020-01-01.
- [Herrero-Lopez et al., 2010] Herrero-Lopez, S., Williams, J. R., and Sanchez, A. (2010). Parallel multiclass classification using svms on gpus. In *Workshop on General-Purpose Computation on Graphics Processing Units*, pages 2–11. ACM.
- [Hinton et al., 2015] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

- [Hiromoto and Miyamoto, 2009] Hiromoto, M. and Miyamoto, R. (2009). Hardware architecture for high-accuracy real-time pedestrian detection with cohog features. In *IEEE International Conference on Computer Vision Workshops*, pages 894–899. IEEE.
- [Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- [Howard et al., 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
- [Hu et al., 2018] Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141.
- [Huang and You, 2016] Huang, J. and You, S. (2016). Point cloud labeling using 3d convolutional neural network. In *IEEE International Conference on Pattern Recognition*, pages 2670–2675. IEEE.
- [Hubara et al., 2016] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016). Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115.
- [Hubara et al., 2017] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. (2017). Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898.
- [Huss and Pennline, 1987] Huss, J. E. and Pennline, J. A. (1987). A comparison of five benchmarks. *NASA Tech. Memorandum 88956*.
- [Huttenlocher et al., 1993] Huttenlocher, D. P., Klanderman, G. A., and Rucklidge, W. J. (1993). Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863.

- [Iandola et al., 2016] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.
- [Insafutdinov et al., 2017] Insafutdinov, E., Andriluka, M., Pishchulin, L., Tang, S., Levinkov, E., Andres, B., and Schiele, B. (2017). Arttrack: Articulated multi-person tracking in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6457–6465.
- [Insafutdinov et al., 2016] Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M., and Schiele, B. (2016). Deeppercut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*, pages 34–50. Springer.
- [Ionescu et al., 2014] Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2014). 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.
- [Iqbal and Gall, 2016] Iqbal, U. and Gall, J. (2016). Multi-person pose estimation with local joint-to-person associations. In *European Conference on Computer Vision*, pages 627–642. Springer.
- [Iqbal et al., 2017] Iqbal, U., Garbade, M., and Gall, J. (2017). Pose for action-action for pose. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 438–445. IEEE.
- [Jacob et al., 2018] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713.
- [Jain et al., 2014] Jain, A., Tompson, J., LeCun, Y., and Bregler, C. (2014). Modeep: A deep learning framework using motion features for human pose estimation. In *Asian conference on computer vision*, pages 302–315. Springer.

- [Jia et al., 2014] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *ACM international conference on Multimedia*, pages 675–678. ACM.
- [Jin et al., 2014] Jin, J., Dundar, A., and Culurciello, E. (2014). Flattened convolutional neural networks for feedforward acceleration. *arXiv preprint arXiv:1412.5474*.
- [Johnson and Everingham, 2010] Johnson, S. and Everingham, M. (2010). Clustered pose and nonlinear appearance models for human pose estimation. In *British Machine Vision Conference*.
- [Joo et al., 2019a] Joo, H., Liu, H., Tan, L., Gui, L., Nabbe, B., Matthews, I., Kanade, T., Nobuhara, S., and Sheikh, Y. (2019a). CMU Panoptic Dataset. <http://domedb.perception.cs.cmu.edu>. Accessed: 2020-01-01.
- [Joo et al., 2019b] Joo, H., Simon, T., Li, X., Liu, H., Tan, L., Gui, L., Banerjee, S., Godisart, T., Nabbe, B., Matthews, I., et al. (2019b). Panoptic studio: A massively multiview system for social interaction capture. *IEEE transactions on pattern analysis and machine intelligence*, 41(1):190–204.
- [Joo et al., 2018] Joo, H., Simon, T., and Sheikh, Y. (2018). Total capture: A 3d deformation model for tracking faces, hands, and bodies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8320–8329.
- [Jung, 2017] Jung, H. Y. (2017). Retraction for “random tree walk toward instantaneous 3d human pose estimation” cvpr 2015 and “a sequential approach to 3d human pose estimation: separation of localization and identification of body joints” eccv 2016.
- [Jung et al., 2015] Jung, H. Y., Lee, S., Heo, Y. S., and Yun, I. D. (2015). Random tree walk toward instantaneous 3d human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2467–2474.
- [Justus et al., 2018] Justus, D., Brennan, J., Bonner, S., and McGough, A. S. (2018). Predicting the computational cost of deep learning models. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3873–3882. IEEE.

- [Kadota et al., 2009] Kadota, R., Sugano, H., Hiromoto, M., Ochi, H., Miyamoto, R., and Nakamura, Y. (2009). Hardware architecture for hog feature extraction. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 1330–1333. IEEE.
- [Kaleem et al., 2015] Kaleem, R., Pai, S., and Pingali, K. (2015). Stochastic gradient descent on gpus. In *Workshop on General Purpose Processing using GPUs*, pages 81–89. ACM.
- [Kim and Smaragdis, 2016] Kim, M. and Smaragdis, P. (2016). Bitwise neural networks. *arXiv preprint arXiv:1601.06071*.
- [Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.
- [Knoll and Keyes, 2004] Knoll, D. A. and Keyes, D. E. (2004). Jacobian-free newton–krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397.
- [Kocabas et al., 2018] Kocabas, M., Karagoz, S., and Akbas, E. (2018). Multiposenet: Fast multi-person pose estimation using pose residual network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 417–433.
- [Kostavelis et al., 2016] Kostavelis, I., Giakoumis, D., Malasiotis, S., and Tzovaras, D. (2016). Ramcip: Towards a robotic assistant to support elderly with mild cognitive impairments at home. In Serino, S., Matic, A., Giakoumis, D., Lopez, G., and Cipresso, P., editors, *Pervasive Computing Paradigms for Mental Health. MindCare 2015. Communications in Computer and Information Science, vol 604*, pages 186–195. Springer International Publishing, Cham.
- [Kostavelis et al., 2018] Kostavelis, I., Giakoumis, D., Peleka, G., Kargakos, A., Skartados, E., Vasileiadis, M., and Tzovaras, D. (2018). Ramcip robot: A personal robotic assistant; demonstration of a complete framework. In *European Conference on Computer Vision Workshops*.
- [Kostavelis et al., 2019] Kostavelis, I., Vasileiadis, M., Skartados, E., Kargakos, A., Giakoumis, D., Bouganis, C.-S., and Tzovaras, D. (2019). Understanding of human behavior with

- a robotic agent through daily activity analysis. *International Journal of Social Robotics*, 11(3):437–462.
- [Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- [Kumar et al., 2009] Kumar, N. P., Satoor, S., and Buck, I. (2009). Fast parallel expectation maximization for gaussian mixture models on gpus using cuda. In *IEEE International Conference on High Performance Computing and Communications*, pages 103–109. IEEE.
- [Kumawat and Raman, 2019] Kumawat, S. and Raman, S. (2019). LP-3DCNN: Unveiling local phase in 3D convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [Le and Duan, 2018] Le, T. and Duan, Y. (2018). Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214.
- [LeCun et al., 1998a] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- [LeCun et al., 1998b] LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (1998b). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer.
- [Leo et al., 2017] Leo, M., Medioni, G., Trivedi, M., Kanade, T., and Farinella, G. M. (2017). Computer vision for assistive technologies. *Computer Vision and Image Understanding*, 154:1–15.
- [Lepetit et al., 2005] Lepetit, V., Lagger, P., and Fua, P. (2005). Randomized trees for real-time keypoint recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 775–781. IEEE.
- [Li et al., 2016] Li, F., Zhang, B., and Liu, B. (2016). Ternary weight networks. *arXiv preprint arXiv:1605.04711*.

- [Li and Chan, 2014] Li, S. and Chan, A. B. (2014). 3d human pose estimation from monocular images with deep convolutional neural network. In *Asian Conference on Computer Vision*, pages 332–347. Springer.
- [Lin et al., 2017a] Lin, M., Lin, L., Liang, X., Wang, K., and Cheng, H. (2017a). Recurrent 3d pose sequence machines. In *IEEE Conference on Computer Vision and Pattern Recognition*.
- [Lin et al., 2017b] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017b). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.
- [Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer.
- [Liu and Nocedal, 1989] Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528.
- [Liu et al., 2017] Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744.
- [Liu et al., 2015] Liu, Z., Zhu, J., Bu, J., and Chen, C. (2015). A survey of human pose estimation: the body parts parsing based methods. *Journal of Visual Communication and Image Representation*, 32:10–19.
- [Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.
- [Loper et al., 2015] Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M. J. (2015). Smpl: A skinned multi-person linear model. *ACM Transactions on Graphics*, 34(6):248.

- [Luo et al., 2017] Luo, J.-H., Wu, J., and Lin, W. (2017). Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066.
- [Ma et al., 2018] Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131.
- [MacLean, 2005] MacLean, W. J. (2005). An evaluation of the suitability of fpgas for embedded vision systems. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 131–131. IEEE.
- [Marron et al., 2014] Marron, D., Bifet, A., and Morales, G. D. F. (2014). Random forests of very fast decision trees on gpu for mining evolving big data streams. In *European Conference on Artificial Intelligence*, pages 615–620. IOS Press.
- [Martin, 2005] Martin, S. (2005). Training support vector machines using gilbert’s algorithm. In *IEEE International Conference on Data Mining*, pages 8–pp. IEEE.
- [Maturana and Scherer, 2015] Maturana, D. and Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 922–928. IEEE.
- [McColl et al., 2011] McColl, D., Zhang, Z., and Nejat, G. (2011). Human body pose interpretation and classification for social human-robot interaction. *International Journal of Social Robotics*, 3(3):313.
- [Mehta et al., 2017] Mehta, D., Rhodin, H., Casas, D., Fua, P., Sotnychenko, O., Xu, W., and Theobalt, C. (2017). Monocular 3d human pose estimation in the wild using improved cnn supervision. In *2017 International Conference on 3D Vision (3DV)*, pages 506–516. IEEE.
- [Mikolajczyk and Schmid, 2005] Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630.

- [Mishina et al., 2015] Mishina, Y., Murata, R., Yamauchi, Y., Yamashita, T., and Fujiyoshi, H. (2015). Boosted random forest. *IEICE Transactions on Information and systems*, 98(9):1630–1636.
- [Moeslund et al., 2006] Moeslund, T. B., Hilton, A., and Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2–3):90 – 126.
- [Moon et al., 2017] Moon, G., Chang, J. Y., Suh, Y., and Lee, K. M. (2017). Holistic planimetric prediction to local volumetric prediction for 3d human pose estimation. *arXiv preprint arXiv:1706.04758*.
- [Moon et al., 2018] Moon, G., Yong Chang, J., and Mu Lee, K. (2018). V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5079–5088.
- [Moré, 1978] Moré, J. J. (1978). The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer.
- [Morris and Prasanna, 2005] Morris, G. R. and Prasanna, V. K. (2005). An fpga-based floating-point jacobi iterative solver. In *International Symposium on Parallel Architectures, Algorithms and Networks*, pages 8–pp. IEEE.
- [Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *International conference on machine learning*, pages 807–814.
- [Narayanan et al., 2007] Narayanan, R., Honbo, D., Memik, G., Choudhary, A., and Zambreno, J. (2007). Interactive presentation: An fpga implementation of decision tree classification. In *Conference on Design, automation and test in Europe*, pages 189–194. EDA Consortium.
- [Negi et al., 2011] Negi, K., Dohi, K., Shibata, Y., and Oguri, K. (2011). Deep pipelined one-chip fpga implementation of a real-time image-based human detection algorithm. In *International Conference on Field-Programmable Technology*, pages 1–8. IEEE.

- [Newcombe et al., 2011] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011). Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE international symposium on mixed and augmented reality*, pages 127–136. IEEE.
- [Newell et al., 2017] Newell, A., Huang, Z., and Deng, J. (2017). Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*, pages 2277–2287.
- [Newell et al., 2016] Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer.
- [Nishi and Miura, 2017] Nishi, K. and Miura, J. (2017). Generation of human depth images with body part labels for complex human pose recognition. *Pattern Recognition*.
- [Nocedal and Wright, 2006] Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.
- [Nvidia, 2007] Nvidia (2007). CUDA - compute unified device architecture programming guide.
- [Obdrzalek et al., 2012] Obdrzalek, S., Kurillo, G., Ofli, F., Bajcsy, R., Seto, E., Jimison, H., and Pavel, M. (2012). Accuracy and robustness of kinect pose estimation in the context of coaching of elderly population. In *International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1188–1193.
- [Oberg et al., 2012] Oberg, J., Eguro, K., Bittner, R., and Forin, A. (2012). Random decision tree body part recognition using fpgas. In *International Conference on Field Programmable Logic and Applications*, pages 330–337. IEEE.
- [Okuta et al., 2017] Okuta, R., Unno, Y., Nishino, D., Hido, S., and Loomis, C. (2017). Cupy: A numpy-compatible library for nvidia gpu calculations. In *of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*.

- [Oliveira et al., 2016] Oliveira, G. L., Valada, A., Bollen, C., Burgard, W., and Brox, T. (2016). Deep learning for human part discovery in images. In *IEEE International Conference on Robotics and Automation*, pages 1634–1641. IEEE.
- [Paine et al., 2013] Paine, T., Jin, H., Yang, J., Lin, Z., and Huang, T. (2013). Gpu asynchronous stochastic gradient descent to speed up neural network training. *arXiv preprint arXiv:1312.6186*.
- [Palacios et al., 2013] Palacios, J., Sagiüés, C., Montijano, E., and Llorente, S. (2013). Human-computer interaction based on hand gestures using rgb-d sensors. *Sensors*, 13(9):11842–11860.
- [Papadonikolakis and Bouganis, 2010] Papadonikolakis, M. and Bouganis, C.-S. (2010). A novel fpga-based svm classifier. In *International Conference on Field-Programmable Technology*, pages 283–286. IEEE.
- [Papadonikolakis et al., 2009] Papadonikolakis, M., Bouganis, C.-S., and Constantinides, G. (2009). Performance comparison of gpu and fpga architectures for the svm training problem. In *International Conference on Field-Programmable Technology*, pages 388–391. IEEE.
- [Papandreou et al., 2018] Papandreou, G., Zhu, T., Chen, L.-C., Gidaris, S., Tompson, J., and Murphy, K. (2018). Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–286.
- [Papandreou et al., 2017] Papandreou, G., Zhu, T., Kanazawa, N., Toshev, A., Tompson, J., Bregler, C., and Murphy, K. (2017). Towards accurate multi-person pose estimation in the wild. *IEEE conference on computer vision and pattern recognition*.
- [Park and Hodgins, 2006] Park, S. I. and Hodgins, J. K. (2006). Capturing and animating skin deformation in human motion. In *ACM Transactions on Graphics*, volume 25, pages 881–889. ACM.
- [Park et al., 2011] Park, S.-Y., Choi, S.-I., Kim, J., and Chae, J. S. (2011). Real-time 3d registration using gpu. *Machine Vision and Applications*, 22(5):837–850.

- [Pavlakos et al., 2017] Pavlakos, G., Zhou, X., Derpanis, K. G., and Daniilidis, K. (2017). Coarse-to-fine volumetric prediction for single-image 3d human pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7025–7034.
- [Peemen et al., 2013] Peemen, M., Setio, A. A., Mesman, B., and Corporaal, H. (2013). Memory-centric accelerator design for convolutional neural networks. In *IEEE International Conference on Computer Design*, pages 13–19. IEEE.
- [Pethiyagoda et al., 2014] Pethiyagoda, R., McCue, S. W., Moroney, T. J., and Back, J. M. (2014). Jacobian-free newton–krylov methods with gpu acceleration for computing nonlinear ship wave patterns. *Journal of Computational Physics*, 269:297–313.
- [Pfister et al., 2015] Pfister, T., Charles, J., and Zisserman, A. (2015). Flowing convnets for human pose estimation in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1913–1921.
- [Pfister et al., 2014] Pfister, T., Simonyan, K., Charles, J., and Zisserman, A. (2014). Deep convolutional neural networks for efficient pose estimation in gesture videos. In *Asian Conference on Computer Vision*, pages 538–552. Springer.
- [Pishchulin et al., 2013a] Pishchulin, L., Andriluka, M., Gehler, P., and Schiele, B. (2013a). Poselet conditioned pictorial structures. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 588–595.
- [Pishchulin et al., 2013b] Pishchulin, L., Andriluka, M., Gehler, P., and Schiele, B. (2013b). Strong appearance and expressive spatial models for human pose estimation. In *IEEE International Conference on Computer Vision*, pages 3487–3494.
- [Pishchulin et al., 2016] Pishchulin, L., Insafutdinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P. V., and Schiele, B. (2016). Deepcut: Joint subset partition and labeling for multi person pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4929–4937.

- [Pittman et al., 2013] Pittman, N., Forin, A., Criminisi, A., Shotton, J., and Mahram, A. (2013). Image segmentation using hardware forest classifiers. In *IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 73–80. IEEE.
- [Plagemann et al., 2010] Plagemann, C., Ganapathi, V., Koller, D., and Thrun, S. (2010). Real-time identification and localization of body parts from depth images. In *IEEE International Conference on Robotics and Automation*, pages 3108–3113. IEEE.
- [Plantard et al., 2015] Plantard, P., Auvinet, E., Pierres, A.-S. L., and Multon, F. (2015). Pose estimation with a kinect for ergonomic studies: Evaluation of the accuracy using a virtual mannequin. *Sensors*, 15(1):1785–1803.
- [Polino et al., 2018] Polino, A., Pascanu, R., and Alistarh, D. (2018). Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*.
- [Pons-Moll et al., 2015] Pons-Moll, G., Taylor, J., Shotton, J., Hertzmann, A., and Fitzgibbon, A. (2015). Metric regression forests for correspondence estimation. *International Journal of Computer Vision*, 113(3):163–175.
- [PrimeSense, 2013] PrimeSense (2013). NiTE: Natural Interaction Middleware. <http://openni.ru/files/nite>. Accessed: 2018-02-01.
- [Pugliese et al., 2015] Pugliese, R., Förger, K., and Takala, T. (2015). Game experience when controlling a weak avatar in full-body enactment. In *International Conference on Intelligent Virtual Agents*, pages 418–431. Springer.
- [Qi et al., 2017] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660.
- [Qiu et al., 2009] Qiu, D., May, S., and Nüchter, A. (2009). Gpu-accelerated nearest neighbor search for 3d registration. In *International Conference on Computer Vision Systems*, pages 194–203. Springer.

- [Qiu et al., 2017] Qiu, Z., Yao, T., and Mei, T. (2017). Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541.
- [Rabieah and Bouganis, 2015] Rabieah, M. B. and Bouganis, C.-S. (2015). Fpga based nonlinear support vector machine training using an ensemble learning. In *International Conference on Field Programmable Logic and Applications*, pages 1–4. IEEE.
- [Rabieah and Bouganis, 2016] Rabieah, M. B. and Bouganis, C.-S. (2016). Fpgasvm: A framework for accelerating kernelized support vector machine. In *Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, pages 68–84.
- [Ramakrishna et al., 2014] Ramakrishna, V., Munoz, D., Hebert, M., Bagnell, J. A., and Sheikh, Y. (2014). Pose machines: Articulated pose estimation via inference machines. In *European Conference on Computer Vision*, pages 33–47. Springer.
- [Ramanan, 2007] Ramanan, D. (2007). Learning to parse images of articulated bodies. In *Advances in neural information processing systems*, pages 1129–1136.
- [Rastegari et al., 2016] Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer.
- [Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.
- [Roldao and Constantinides, 2010] Roldao, A. and Constantinides, G. A. (2010). A high throughput fpga-based floating point conjugate gradient implementation for dense matrices. *ACM Transactions on Reconfigurable Technology and Systems*, 3(1):1.
- [Romero et al., 2014] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2014). Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

- [Ryu and Monk, 2009] Ryu, H. and Monk, A. (2009). Interaction unit analysis: A new interaction design framework. *Human-Computer Interaction*, 24(4):367–407.
- [Sandler et al., 2018] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520.
- [Sapp and Taskar, 2013] Sapp, B. and Taskar, B. (2013). Modec: Multimodal decomposable models for human pose estimation. In *IEEE conference on computer vision and pattern recognition*.
- [Saqib et al., 2015] Saqib, F., Dutta, A., Plusquellic, J., Ortiz, P., and Pattichis, M. S. (2015). Pipelined decision tree classification accelerator implementation in fpga (dt-caif). *IEEE Transactions on Computers*, 64(1):280–285.
- [Sarafianos et al., 2016] Sarafianos, N., Boteanu, B., Ionescu, B., and Kakadiaris, I. A. (2016). 3d human pose estimation: A review of the literature and analysis of covariates. *Computer Vision and Image Understanding*, 152:1 – 20.
- [Schmidt et al., 2015] Schmidt, T., Newcombe, R., and Fox, D. (2015). Dart: dense articulated real-time tracking with consumer depth cameras. *Autonomous Robots*, pages 1–20.
- [Sedaghat et al., 2017] Sedaghat, N., Zolfaghari, M., Amiri, E., and Brox, T. (2017). Orientation-boosted voxel nets for 3D object recognition. In *British Machine Vision Conference (BMVC)*.
- [Sedai et al., 2009] Sedai, S., Bennamoun, M., and Huynh, D. (2009). Context-based appearance descriptor for 3d human pose estimation from monocular images. In *2009 Digital Image Computing: Techniques and Applications*, pages 484–491. IEEE.
- [Serban et al., 2015] Serban, R., Melanz, D., Li, A., Stanciulescu, I., Jayakumar, P., and Ne-grut, D. (2015). A gpu-based preconditioned newton-krylov solver for flexible multibody dynamics. *International Journal for Numerical Methods in Engineering*, 102(9):1585–1604.

- [Sharp, 2008] Sharp, T. (2008). Implementing decision trees and forests on a gpu. In *European conference on computer vision*, pages 595–608. Springer.
- [Shotton et al., 2013] Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., and Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124.
- [Sifre and Mallat, 2014] Sifre, L. and Mallat, S. (2014). Rigid-motion scattering for image classification. *Ph. D. dissertation*.
- [Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.
- [Song et al., 2017] Song, J., Wang, L., Van Gool, L., and Hilliges, O. (2017). Thin-slicing network: A deep structured model for pose estimation in videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4220–4229.
- [Song and Xiao, 2016] Song, S. and Xiao, J. (2016). Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816.
- [Song and Davis, 2015] Song, Y. and Davis, R. (2015). Continuous body and hand gesture recognition for natural human-computer interaction. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.
- [Spring and Shrivastava, 2017] Spring, R. and Shrivastava, A. (2017). Scalable and sustainable deep learning via randomized hashing. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 445–454. ACM.
- [Springenberg et al., 2014] Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

- [Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.
- [Suzuo, 2018] Suzuo, D. (2018). Chainer computational cost. https://github.com/belltailjp/chainer_computational_cost. Accessed: 2020-01-01.
- [Szegedy et al., 2017] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.
- [Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.
- [Szegedy et al., 2016] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.
- [Tan et al., 2018] Tan, M., Chen, B., Pang, R., Vasudevan, V., and Le, Q. V. (2018). Platform-aware neural architecture search for mobile.
- [Taylor et al., 2012] Taylor, J., Shotton, J., Sharp, T., and Fitzgibbon, A. (2012). The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 103–110. IEEE.
- [Tekin et al., 2016] Tekin, B., Rozantsev, A., Lepetit, V., and Fua, P. (2016). Direct prediction of 3d body poses from motion compensated sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 991–1000.
- [The MakeHuman team, 2000] The MakeHuman team (2000). MakeHuman: Open source tool for making 3D characters. <http://www.makehumancommunity.org/>. Accessed: 2020-01-01.
- [Thrun, 2003] Thrun, S. (2003). Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15(2):111–127.

- [Tokui et al., 2015] Tokui, S., Oono, K., Hido, S., and Clayton, J. (2015). Chainer: a next-generation open source framework for deep learning. In *Conference on neural information processing systems workshops*, volume 5.
- [Tompson et al., 2014] Tompson, J. J., Jain, A., LeCun, Y., and Bregler, C. (2014). Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807.
- [Toshev and Szegedy, 2014] Toshev, A. and Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1653–1660.
- [Tran and Forsyth, 2010] Tran, D. and Forsyth, D. (2010). Improved human parsing with a full relational model. *European Conference on Computer Vision*, pages 227–240.
- [Unzueta et al., 2014] Unzueta, L., Goenetxea, J., Rodriguez, M., and Linaza, M. T. (2014). dependent 3d human body posing for sports legacy recovery from images and video. In *2014 22nd European Signal Processing Conference (EUSIPCO)*, pages 361–365. IEEE.
- [Vanhoucke et al., 2011] Vanhoucke, V., Senior, A., and Mao, M. Z. (2011). Improving the speed of neural networks on cpus.
- [Varol et al., 2017] Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., and Schmid, C. (2017). Learning from Synthetic Humans. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 109–117.
- [Vasileiadis et al., 2019a] Vasileiadis, M., Bouganis, C.-S., Stavropoulos, G., and Tzovaras, D. (2019a). Optimising 3d-cnn design towards human pose estimation on low power devices. In *British Machine Vision Conference*. BMVA.
- [Vasileiadis et al., 2019b] Vasileiadis, M., Bouganis, C.-S., and Tzovaras, D. (2019b). Multi-person 3d pose estimation from 3d cloud data using 3d convolutional neural networks. *Computer Vision and Image Understanding*, 185:12 – 23.

- [Vasileiadis et al., 2016] Vasileiadis, M., Giakoumis, D., Votis, K., and Tzovaras, D. (2016). A living lab infrastructure for investigating activity monitoring needs in service robot applications. In *Proceedings of the 9th ACM International Conference on Pervasive Technologies Related to Assistive Environments*, pages 1–7.
- [Vasileiadis et al., 2017a] Vasileiadis, M., Malassiotis, S., Giakoumis, D., Bouganis, C. S., and Tzovaras, D. (2017a). Realistic Human Motion Dataset. <http://ramcip-project.eu/ramcip-data-mng/#/dataManagement>. Accessed: 2020-01-01.
- [Vasileiadis et al., 2017b] Vasileiadis, M., Malassiotis, S., Giakoumis, D., Bouganis, C. S., and Tzovaras, D. (2017b). Robust human pose tracking for realistic service robot applications. In *IEEE International Conference on Computer Vision Workshops*, pages 1363–1372. IEEE.
- [Venieris and Bouganis, 2016] Venieris, S. I. and Bouganis, C.-S. (2016). fpgaconvnet: A framework for mapping convolutional neural networks on fpgas. In *IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 40–47. IEEE.
- [Wang et al., 2016] Wang, K., Zhai, S., Cheng, H., Liang, X., and Lin, L. (2016). Human pose estimation from depth images via inference embedded multi-task learning. In *ACM Conference on Multimedia*, pages 1227–1236. ACM.
- [Wang et al., 2017] Wang, M., Liu, B., and Foroosh, H. (2017). Factorized convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 545–553.
- [Wang et al., 2018] Wang, X., Zhang, R., Sun, Y., and Qi, J. (2018). Kdgan: knowledge distillation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 775–786.
- [Wang et al., 2011] Wang, Y., Tran, D., and Liao, Z. (2011). Learning hierarchical poselets for human parsing. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1705–1712. IEEE.

- [Watanabe et al., 2009] Watanabe, T., Ito, S., and Yokoi, K. (2009). Co-occurrence histograms of oriented gradients for pedestrian detection. *Advances in Image and Video Technology*, pages 37–47.
- [Wedderburn, 1974] Wedderburn, R. W. (1974). Quasi-likelihood functions, generalized linear models, and the gauss—newton method. *Biometrika*, 61(3):439–447.
- [Wei et al., 2016] Wei, S.-E., Ramakrishna, V., Kanade, T., and Sheikh, Y. (2016). Convolutional pose machines. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4724–4732.
- [Wei et al., 2012] Wei, X., Zhang, P., and Chai, J. (2012). Accurate realtime full-body motion capture using a single depth camera. *ACM Transactions on Graphics*, 31(6):188.
- [Williams et al., 2009] Williams, S., Waterman, A., and Patterson, D. (2009). Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76.
- [Wu et al., 2015a] Wu, F., Wang, F., Jiang, P., Zhao, C., and Cheng, J. (2015a). A nearest neighbor searches (nns) algorithm for fast registration of 3d point clouds based on gpgpu. In *International Conference on Intelligent Systems Research and Mechatronics Engineering*. Atlantis Press.
- [Wu et al., 2016] Wu, J., Leng, C., Wang, Y., Hu, Q., and Cheng, J. (2016). Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828.
- [Wu et al., 2015b] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015b). 3d shapenets: A deep representation for volumetric shapes. In *IEEE conference on computer vision and pattern recognition*, pages 1912–1920.
- [Xiao et al., 2018] Xiao, B., Wu, H., and Wei, Y. (2018). Simple baselines for human pose estimation and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 466–481.

- [Xiaohan Nie et al., 2015] Xiaohan Nie, B., Xiong, C., and Zhu, S.-C. (2015). Joint action recognition and pose estimation from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1293–1301.
- [Xie et al., 2017] Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.
- [Xiong et al., 2019] Xiong, F., Zhang, B., Xiao, Y., Cao, Z., Yu, T., Zhou, J. T., and Yuan, J. (2019). A2j: Anchor-to-joint regression network for 3d articulated pose estimation from a single depth image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 793–802.
- [Xiong and De la Torre, 2013] Xiong, X. and De la Torre, F. (2013). Supervised descent method and its applications to face alignment. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 532–539.
- [Xu and Ziavras, 2003] Xu, X. and Ziavras, S. G. (2003). Iterative methods for solving linear systems of equations on fpga-based machines. In *Computers and Their Applications*, pages 472–475.
- [Yang et al., 2017] Yang, W., Li, S., Ouyang, W., Li, H., and Wang, X. (2017). Learning feature pyramids for human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1281–1290.
- [Yang et al., 2018] Yang, W., Ouyang, W., Wang, X., Ren, J., Li, H., and Wang, X. (2018). 3d human pose estimation in the wild by adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5255–5264.
- [Yang and Tian, 2014] Yang, X. and Tian, Y. (2014). Effective 3d action recognition using eigenjoints. *Journal of Visual Communication and Image Representation*, 25(1):2–11.
- [Yang and Ramanan, 2013] Yang, Y. and Ramanan, D. (2013). Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2878–2890.

- [Ye et al., 2011] Ye, M., Wang, X., Yang, R., Ren, L., and Pollefeys, M. (2011). Accurate 3d pose estimation from a single depth image. In *IEEE International Conference on Computer Vision*, pages 731–738. IEEE.
- [Ye and Yang, 2014] Ye, M. and Yang, R. (2014). Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2345–2352. IEEE.
- [Ye et al., 2019] Ye, R., Liu, F., and Zhang, L. (2019). 3d depthwise convolution: Reducing model parameters in 3d vision tasks. In *Canadian Conference on Artificial Intelligence*, pages 186–199. Springer.
- [Yim et al., 2017] Yim, J., Joo, D., Bae, J., and Kim, J. (2017). A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4133–4141.
- [Yu and Koltun, 2015] Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.
- [Yuan et al., 2018a] Yuan, S., Garcia-Hernando, G., Stenger, B., Moon, G., Chang, J. Y., Lee, K. M., Molchanov, P., Kautz, J., Honari, S., Ge, L., et al. (2018a). Depth-based 3d hand pose estimation: From current achievements to future goals. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2636–2645. IEEE.
- [Yuan et al., 2018b] Yuan, S., Stenger, B., and Kim, T.-K. (2018b). Rgb-based 3d hand pose estimation via privileged learning with depth images. *arXiv preprint arXiv:1811.07376*.
- [Zanuttigh et al., 2016] Zanuttigh, P., Marin, G., Dal Mutto, C., Dominio, F., Minto, L., and Cortelazzo, G. M. (2016). *Time-of-Flight and Structured Light Depth Cameras*. Springer.
- [Zeiler et al., 2011] Zeiler, M. D., Taylor, G. W., and Fergus, R. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *IEEE International Conference on Computer Vision*, pages 2018–2025. IEEE.

- [Zhang et al., 2015] Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., and Cong, J. (2015). Optimizing fpga-based accelerator design for deep convolutional neural networks. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 161–170. ACM.
- [Zhang et al., 2019] Zhang, F., Zhu, X., and Ye, M. (2019). Fast human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3517–3526.
- [Zhang et al., 2013a] Zhang, S., Zhang, C., You, Z., Zheng, R., and Xu, B. (2013a). Asynchronous stochastic gradient descent for dnn training. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6660–6663. IEEE.
- [Zhang et al., 2013b] Zhang, W., Zhu, M., and Derpanis, K. G. (2013b). From actemes to action: A strongly-supervised representation for detailed action understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2248–2255.
- [Zhang et al., 2018] Zhang, X., Zhou, X., Lin, M., and Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856.
- [Zhang, 2012] Zhang, Z. (2012). Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10.
- [Zhi et al., 2017] Zhi, S., Liu, Y., Li, X., and Guo, Y. (2017). Lightnet: A lightweight 3D convolutional neural network for real-time 3D object recognition. In *3DOR*.
- [Zhou et al., 2017] Zhou, A., Yao, A., Guo, Y., Xu, L., and Chen, Y. (2017). Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*.
- [Zhou et al., 2016a] Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., and Zou, Y. (2016a). Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*.

- [Zhou et al., 2016b] Zhou, X., Sun, X., Zhang, W., Liang, S., and Wei, Y. (2016b). Deep kinematic pose regression. In *Computer Vision–ECCV 2016 Workshops*, pages 186–201. Springer.
- [Zhou et al., 2016c] Zhou, X., Zhu, M., Leonardos, S., Derpanis, K. G., and Daniilidis, K. (2016c). Sparseness meets deepness: 3d human pose estimation from monocular video. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4966–4975.
- [Zimmermann et al., 2018] Zimmermann, C., Welschhold, T., Dornhege, C., Burgard, W., and Brox, T. (2018). 3d human pose estimation in rgb-d images for robotic task learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1986–1992. IEEE.

Appendix A

Experimental Evaluation of the Template Based Pose Tracking Framework

Public Datasets

Table A.1: Experimental evaluation of the proposed template-based pose tracking algorithm on three publicly available datasets.

	SMMC-10		EVAL		PDT	
	ADE	mAP(0.1m)	ADE	mAP(0.1m)	ADE	mAP(0.1m)
Head	0.021	0.997	0.035	0.980	0.030	0.956
Torso	0.063	0.993	0.049	0.978	0.032	0.946
Left Shoulder	0.064	0.971	0.069	0.824	0.033	0.952
Right Shoulder	0.051	0.972	0.083	0.737	0.033	0.951
Left Elbow	0.065	0.855	0.053	0.831	0.047	0.905
Right Elbow	0.056	0.885	0.071	0.754	0.045	0.906
Left Hand	0.066	0.887	0.083	0.693	0.067	0.828
Right Hand	0.048	0.935	0.067	0.808	0.063	0.838
Left Hip	0.023	1.000	0.046	0.927	0.050	0.952
Right Hip	0.023	0.997	0.044	0.943	0.042	0.965
Left Knee	0.033	0.997	0.036	0.955	0.030	0.967
Right Knee	0.027	1.000	0.038	0.928	0.033	0.959
Left Foot	0.023	0.995	0.038	0.899	0.042	0.924
Right Foot	0.015	0.998	0.040	0.932	0.052	0.867
MEAN	0.041	0.963	0.054	0.871	0.043	0.923
<i>Articulated ICP</i>		0.909		0.804		
[Ganapathi et al., 2010a]	0.073	0.912				
[Ganapathi et al., 2012b]		0.973		0.844		
[Helten et al., 2013a]	0.061				0.064	
[Baak et al., 2013]	0.062				0.071	
[Shotton et al., 2013]		0.948			0.080	
[Wei et al., 2012]		0.966				
[Ye et al., 2011]	0.038					
KinectSDK					0.087	
[Ye and Yang, 2014]	0.034	0.970		0.920	0.046	0.918
[Schmidt et al., 2015]				0.894		
[Ding and Fan, 2015]	0.039	0.960				
[Jung et al., 2015]		0.960		0.740		

Realistic Human Motion Dataset

Table A.2: Experimental evaluation of the proposed template-based pose tracking algorithm on the novel realistic human motion dataset and comparison to the Kinect 1 built-in pose estimator (NiTE 2.2). To ensure consistency between the two systems, the distance between the ground truth positions and the skeleton joints was used, instead of the corresponding mesh point, which may have resulted in a slightly higher Average Distance Error

	OURS		KINECT (NiTE 2.2)	
	ADE	mAP(0.1m)	ADE	mAP(0.1m)
Head	0.067	0.862	0.072	0.834
Left Elbow	0.061	0.913	0.090	0.676
Right Elbow	0.070	0.848	0.102	0.693
Left Hand	0.074	0.838	0.096	0.689
Right Hand	0.094	0.740	0.150	0.644
Left Knee	0.087	0.707	0.122	0.504
Right Knee	0.091	0.697	0.131	0.478
Left Foot	0.075	0.891	0.143	0.441
Right Foot	0.058	0.928	0.117	0.559
MEAN	0.075	0.825	0.114	0.613

Complementary Tracking Features

Table A.3: Experimental evaluation of the leg intersection fix. Tested on a subset of the PDT dataset, for sequences with complex leg movement

	Without leg intersection fix		With leg intersection fix	
	ADE	mAP(0.1m)	ADE	mAP(0.1m)
Left Knee	0.078	0.849	0.030	0.982
Right Knee	0.066	0.849	0.035	0.969
Left Foot	0.127	0.786	0.044	0.925
Right Foot	0.104	0.771	0.056	0.890
MEAN	0.094	0.814	0.041	0.941

Table A.4: Experimental evaluation of the body-part visibility feature. Tested on a subset of the PDT dataset, where lower body motion was of moderate complexity. The lower body parts were artificially occluded, in order to simulate a “acting behind a table” scenario

	Without body part visibility		With body part visibility	
	ADE	mAP(0.1m)	ADE	mAP(0.1m)
Upper visible limbs				
Head	0.024	0.993	0.027	0.983
Torso	0.030	0.994	0.034	0.966
Left Shoulder	0.028	0.997	0.029	0.992
Right Shoulder	0.026	0.998	0.030	0.974
Left Elbow	0.026	0.995	0.025	0.995
Right Elbow	0.026	0.994	0.028	0.994
Left Hand	0.042	0.930	0.045	0.936
Right Hand	0.035	0.960	0.035	0.951
MEAN	0.030	0.983	0.032	0.974
Lower occluded limbs				
Left Knee	0.141	0.568	0.118	0.568
Right Knee	0.164	0.569	0.097	0.630
Left Foot	0.388	0.150	0.313	0.198
Right Foot	0.440	0.175	0.288	0.160
MEAN	0.283	0.366	0.204	0.389
TOTAL MEAN	0.114	0.777	0.089	0.779

Appendix B

Experimental Evaluation of Deep Learning Based Pose Estimation

OpenPose

Table B.1: Experimental evaluation of the 2D-CNN based OpenPose architecture on two publicly available datasets.

	EVAL		ITOP (FRONT)	
	PCKh	mAP	PCKh	mAP
Head	0.979	0.915	0.995	0.987
Neck			0.996	0.973
Left Shoulder	0.971	0.888	0.992	0.919
Right Shoulder	0.959	0.862	0.992	0.934
Left Elbow	0.941	0.719	0.942	0.733
Right Elbow	0.929	0.691	0.949	0.763
Left Hand	0.888	0.622	0.806	0.576
Right Hand	0.875	0.641	0.817	0.605
Torso	0.989	0.927	0.997	0.971
Left Hip			0.993	0.947
Right Hip			0.994	0.943
Left Knee	0.964	0.791	0.966	0.903
Right Knee	0.942	0.719	0.971	0.911
Left Foot	0.968	0.890	0.934	0.816
Right Foot	0.960	0.907	0.923	0.806
MEAN	0.947	0.798	0.951	0.852

Proposed 3D-CNN Architecture

Table B.2: Experimental evaluation of the proposed 3D-CNN based pose estimation algorithm on three publicly available single person datasets.

	ITOP - front		ITOP - top		EVAL		PDT	
	ADE	mAP	ADE	mAP	ADE	mAP	ADE	mAP
Head	0.034	0.985	0.039	0.984	0.065	0.917	0.050	0.962
Neck	0.036	0.990	0.040	0.987	0.052	0.955	0.043	0.986
Torso	0.043	0.984	0.048	0.979			0.051	0.970
Left Shoulder	0.045	0.962	0.046	0.967	0.063	0.931	0.053	0.956
Right Shoulder	0.045	0.971	0.051	0.962	0.071	0.906	0.051	0.963
Left Elbow	0.069	0.817	0.074	0.800	0.081	0.806	0.067	0.874
Right Elbow	0.067	0.832	0.075	0.804	0.095	0.720	0.062	0.903
Left Hand	0.095	0.708	0.103	0.660	0.085	0.856	0.074	0.846
Right Hand	0.091	0.727	0.104	0.648	0.091	0.821	0.078	0.814
Left Hip	0.057	0.926	0.064	0.899			0.060	0.888
Right Hip	0.060	0.910	0.062	0.907			0.058	0.893
Left Knee	0.053	0.926	0.074	0.827	0.072	0.900	0.096	0.934
Right Knee	0.052	0.926	0.073	0.829	0.090	0.804	0.055	0.956
Left Foot	0.072	0.863	0.094	0.751	0.065	0.954	0.065	0.910
Right Foot	0.066	0.858	0.099	0.670	0.088	0.899	0.069	0.866
MEAN	0.059	0.893	0.070	0.845	0.077	0.873	0.062	0.914
[Ganapathi et al., 2012b]						0.844		
[Shotton et al., 2013]		0.658		0.474			0.080	
[Ye and Yang, 2014]						0.920	0.046	0.918
[Schmidt et al., 2015]						0.894		
[Jung et al., 2015]		0.805		0.682		0.740		
[Haque et al., 2016b]		0.774		0.755		0.741		
[Carreira et al., 2016]		0.714		0.512				
[Cao et al., 2017]		0.852				0.798		
[Vasileiadis et al., 2017b]					0.054	0.871	0.043	0.923
[Moon et al., 2017]		0.834				0.798		
[Guo et al., 2017]		0.849		0.755				
[Moon et al., 2018]		0.887		0.834		0.844		
[Xiong et al., 2019]		0.880		0.805				

Table B.3: Experimental evaluation of the proposed 3D-CNN based pose estimation algorithm on the CMU PanopticStudio 3D PointCloud multi person dataset.

	HAGGLING		BAND	
	ADE	mAP	ADE	mAP
Head	0.040	0.990	0.029	0.992
Neck	0.039	0.996	0.031	0.989
Left Shoulder	0.040	0.993	0.029	0.990
Right Shoulder	0.038	0.996	0.033	0.992
Left Elbow	0.044	0.992	0.036	0.982
Right Elbow	0.040	0.991	0.040	0.988
Left Hand	0.040	0.977	0.038	0.961
Right Hand	0.040	0.973	0.043	0.976
Left Hip	0.046	0.988	0.059	0.967
Right Hip	0.050	0.984	0.085	0.704
Left Knee	0.047	0.990	0.030	0.989
Right Knee	0.050	0.989	0.040	0.974
Left Foot	0.049	0.981	0.052	0.966
Right Foot	0.050	0.986	0.050	0.973
MEAN	0.044	0.988	0.043	0.960

Appendix C

Experimental Evaluation of the Optimised 3D CNN Architecture

Table C.1: Experimental evaluation of the proposed optimised 3D-CNN pose estimation architecture on the single person ITOP and multi-person CMU Panoptic PtCloud datasets

	ITOP - front		ITOP - top		CMU - Haggling		CMU - Band	
	ADE	mAP	ADE	mAP	ADE	mAP	ADE	mAP
Head	0.036	0.983	0.040	0.981	0.044	0.983	0.036	0.981
Neck	0.038	0.986	0.043	0.983	0.041	0.995	0.031	0.981
Torso	0.044	0.983	0.053	0.973				
Left Shoulder	0.046	0.965	0.047	0.964	0.042	0.991	0.032	0.973
Right Shoulder	0.048	0.968	0.053	0.950	0.040	0.992	0.044	0.976
Left Elbow	0.070	0.818	0.081	0.773	0.046	0.985	0.033	0.965
Right Elbow	0.070	0.821	0.080	0.771	0.044	0.979	0.046	0.969
Left Hand	0.099	0.696	0.111	0.630	0.042	0.969	0.039	0.961
Right Hand	0.103	0.704	0.111	0.623	0.046	0.960	0.045	0.950
Left Hip	0.061	0.909	0.070	0.854	0.057	0.965	0.062	0.962
Right Hip	0.057	0.927	0.070	0.833	0.057	0.957	0.085	0.693
Left Knee	0.057	0.914	0.086	0.768	0.051	0.982	0.036	0.971
Right Knee	0.055	0.911	0.082	0.785	0.053	0.978	0.034	0.955
Left Foot	0.072	0.865	0.105	0.689	0.053	0.978	0.052	0.951
Right Foot	0.067	0.875	0.095	0.726	0.053	0.978	0.055	0.944
MEAN	0.062	0.888	0.075	0.820	0.048	0.978	0.045	0.945
<i>Baseline 3D CNN Estimator</i>	0.059	0.893	0.070	0.845	0.044	0.988	0.043	0.960