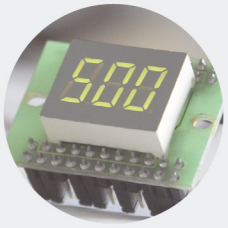
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Estimação da pose humana com método baseado em visão para reabilitação do membro superior

FRANCISCO JOSÉ PRETO PIRES
outubro de 2024

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Human pose estimation vision-based method for upper limb rehabilitation

Francisco José Preto Pires

Master in Electrical and Computer Engineering
Specialization Area of Automation and Systems

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

October, 2024

This dissertation partially satisfies the requirements of the Thesis/Dissertation course of the program Master in Electrical and Computer Engineering, Specialization Area of Automation and Systems.

Candidate: Francisco José Preto Pires, No. 1181808, 1181808@isep.ipp.pt

Scientific Guidance: Manuel Santos Silva, mss@isep.ipp.pt

Company: INESC TEC - Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência

Advisor: Hélder Filipe Pinto de Oliveira helder.f.oliveira@inesctec.pt / Cláudia Daniela Costa Rocha, claudia.d.rocha@inesctec.pt

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

October, 2024

Acknowledgements

I would like to express my sincere gratitude to everyone who has supported me throughout the journey of completing this thesis.

First and foremost, I am deeply indebted to my thesis supervisor at ISEP, Professor Manuel Santos, for his invaluable guidance, patience, and encouragement. He always made time for me, providing insightful comments and feedback that helped pave the way for this work.

I would also like to extend my heartfelt thanks to my advisors at INESC TEC, Professor Hélder Oliveira and Cláudia Rocha, for their constructive suggestions and critical evaluations, which greatly improved the quality of this research. They were always available when I needed assistance, and I am grateful for their efforts to support me while managing their own responsibilities.

I am profoundly grateful to my family for their unwavering support and encouragement. Their belief in me has been a constant source of motivation, especially during difficult times. Without them I wouldn't have the strengths needed to embark on such a arduous journey. I would also like to thank my friends and colleagues for providing a healthy balance of academic discussions and light-hearted distractions during lunchtime, vacations, and evenings filled with tabletop and party games.

Thank you all, you are the best.

A special thanks goes to INESC TEC for providing the resources and facilities necessary to carry out this research, as well as the researchers and collaborators who contributed to making the environment in their lab a pleasant place to stay.

I gratefully acknowledge the support provided by the AI-Care4U project at INESC TEC. This work was co-financed by Component 5 - Capitalization and Business Innovation of the core funding for Technology and Innovation Centres (CTI), integrated in the Resilience Dimension of the Recovery and Resilience Plan under the Recovery and Resilience Mechanism (MRR) of the European Union (EU), framed within the Next Generation EU, for the period 2021 - 2026.

Thank you all for your invaluable support.

Abstract

In recent years, the need for real-time and efficient human pose estimation has grown significantly across various fields, from sports analytics, virtual reality, and healthcare, especially in the context of rehabilitation, as one-third of stroke survivors become dependent on others for daily life. This work addresses this challenge by developing lightweight and resource-efficient deep learning models optimized for markerless, vision-based real-time upper-limb stroke rehabilitation, as an alternative to cumbersome physical sensors and markers. Traditional pose estimation models, while accurate, are often too computationally expensive for real-time use, particularly in environments with limited hardware resources. Three different models were implemented and tested: ResNet-18, U-Net, and Keypoint Region-based Convolutional Neural Network (R-CNN), each offering distinct approaches to 2D keypoint estimation. The models were evaluated using multiple datasets, with their performance varying based on the complexity of the data and estimation methods used.

A key aspect of this research involved the integration of temporal smoothing techniques using a Long Short-Term Memory (LSTM) model, which aimed to enhance pose tracking by learning from past keypoint positions. While this method improved smoothness in stable keypoints, challenges remained in dynamic scenarios, particularly with more mobile body parts.

The findings showed that while some models achieved higher accuracy (Keypoint R-CNN), they required more processing time, making them less suited for real-time scenarios, where speed is crucial. Additionally, the thesis includes efforts to improve system performance through camera calibration techniques, improving the overall alignment between Red Green Blue (RGB) and depth data of the Red Green Blue and Depth (RGB-D) camera system.

In summary, this work highlights the trade-offs between model complexity, accuracy, and speed, presenting a solution that advances human pose estimation while identifying areas for future improvement in both real-time processing and system optimization.

Keywords: Human pose estimation, Real-time, Upper-limb rehabilitation, Stroke, Temporal smoothing, Neural networks

Resumo

Nos últimos anos, a necessidade de estimativa da pose humana em tempo real e de forma eficiente cresceu significativamente em vários campos, desde a análise desportiva à realidade virtual e aos cuidados de saúde, especialmente no contexto da reabilitação, onde um terço dos sobreviventes de Acidente vascular cerebral (AVC) se tornam dependentes de outros para o dia-a-dia. Este trabalho aborda esse desafio ao desenvolver modelos de *deep learning* leves e eficientes, otimizados para reabilitação de AVC dos membros superiores sem marcadores e baseada na visão em tempo real, como uma alternativa aos sensores e marcadores físicos incómodos. Os modelos tradicionais de estimativa de pose, embora precisos, costumam ser muito exigentes em termos computacionais para uso em tempo real, especialmente em ambientes com recursos limitados. Foram testados três modelos distintos, cada um oferecendo abordagens diferentes para a estimativa de pontos-chave em 2D. Os modelos foram avaliados utilizando vários conjuntos de dados, com o seu desempenho a variar com base na complexidade dos dados e nos métodos de estimativa utilizados.

Um aspecto fundamental desta pesquisa é a integração de técnicas de suavização temporal usando um modelo LSTM, que visa melhorar o acompanhamento da pose ao aprender com posições passadas dos pontos-chave. Embora este método tenha melhorado a suavidade em pontos-chave estáveis, continuaram a existir desafios em cenários dinâmicos, particularmente com partes do corpo mais móveis.

Os resultados mostram que, embora alguns modelos tenham alcançado maior precisão (Keypoint R-CNN), eles exigem mais tempo de processamento, tornando-os menos adequados para cenários em tempo real, onde a velocidade é crucial. Além disso, a tese inclui esforços para melhorar o desempenho do sistema por meio de técnicas de calibração de câmeras, melhorando o alinhamento geral entre os dados de RGB e profundidade do sistema de câmara RGB-D.

Em resumo, este trabalho destaca os compromissos entre a complexidade do modelo, a precisão e a velocidade, apresentando uma solução que avança a estimativa de pose humana, ao mesmo tempo que identifica áreas para melhorias futuras tanto no processamento em tempo real quanto na otimização do sistema.

Palavras-Chave: Estimativa da pose humana, Tempo real, Reabilitação do membro superior, AVC, Suavização temporal, Redes neurais

Contents

List of Figures **xi**

List of Tables **xvii**

List of Acronyms **xix**

1 Introduction **1**
1.1 Contextualization . . . 1
1.2 Motivation . . . 2
1.3 Problem description . . . 2
1.3.1 Objectives . . . 3
1.3.2 Achieved results . . . 3
1.4 Work plan . . . 3
1.5 Dissertation structure . . . 6

2 Stroke rehabilitation **7**
2.1 What is a stroke . . . 7
2.2 Post-stroke rehabilitation . . . 8
2.2.1 Outcome assessment methods . . . 9
2.2.2 Upper limb intervention . . . 9
2.3 Employed exercises in upper limb rehabilitation . . . 13
2.4 Summary . . . 15

3 Computer vision and pose estimation **17**
3.1 Sensor technologies for human pose capture . . . 17
3.2 Challenges . . . 21
3.3 Pose estimation approaches . . . 21
3.3.1 Human body models . . . 22
3.3.2 2D/3D pose estimation . . . 23
Single-person pose estimation . . . 24
Multi-person pose estimation . . . 25
3.4 Backbones . . . 26
3.4.1 AlexNet . . . 26
3.4.2 VGG . . . 28

3.4.3 ResNet . . . 28
3.4.4 Faster R-CNN . . . 29
3.4.5 Mask R-CNN . . . 30
3.4.6 Long Short-Term Memory . . . 30
3.4.7 U-Net . . . 31
3.4.8 Stacked Hourglass . . . 31
3.4.9 Cascaded Pyramid Network . . . 32
3.4.10 HRNet . . . 32
3.4.11 SelecSLS . . . 33
3.4.12 Key findings . . . 34
3.5 Common evaluation metrics and datasets . . . 34
3.5.1 Evaluation metrics . . . 35
3.5.2 Datasets for human pose estimation . . . 37
3.6 Deep learning based approaches . . . 39
3.6.1 DeepPose . . . 39
3.6.2 XNect . . . 40
3.6.3 AlphaPose . . . 41
3.6.4 OpenPose . . . 41
3.6.5 IEF . . . 41
3.6.6 HRNet . . . 42
3.6.7 Key findings . . . 42
3.7 Summary . . . 44

4 Proposed system architecture 47
4.1 Overview of system requirements . . . 47
4.1.1 Key considerations for the scenario . . . 48
4.2 Key components . . . 50
4.2.1 RGB-D input . . . 51
4.2.2 Chosen architecture for the HPE model . . . 52
4.2.3 Choice for temporal smoothing network . . . 53
4.2.4 3D joint annotations computation . . . 54
4.3 Technologies and frameworks harnessed for development . . . 54
4.3.1 JupyterLab . . . 55
4.3.2 PyTorch . . . 55
4.3.3 Windows Subsystem for Linux . . . 55
4.4 Summary . . . 56

5 Development work for the human pose estimation method 57
5.1 Analysis and processing of selected datasets . . . 57
5.1.1 Analysis of dataset content quality . . . 57
5.1.2 Dataset initialization and preparation for training . . . 58

5.1.3 Keypoint overlap on image . . . 61
5.1.4 Heatmap representation of keypoints . . . 62
5.1.5 3D keypoint visualization . . . 64
5.2 Training the neural network . . . 64
5.2.1 Dataset initialization for processing during learning . . . 65
5.2.2 Network preparation . . . 66
5.2.3 Training setup . . . 67
5.2.4 Training optimizations . . . 69
5.3 Heatmaps . . . 70
5.4 Inference with trained models . . . 72
5.4.1 Types of visualization . . . 72
5.4.2 Comparison and benchmarking metrics . . . 75
5.5 Temporal smoothing . . . 75
5.5.1 Preliminary testing with LSTM . . . 75
Prediction of next value in a sequence . . . 75
Prediction of next value in sine wave . . . 76
Prediction of next value in sum of random numbers . . . 76
Prediction of next coordinates of multiple points . . . 77
5.5.2 Application of LSTM temporal smoothing in HPE . . . 78
5.6 Camera calibration and device configuration . . . 80
5.7 Summary . . . 82

6 Results and tests **85**
6.1 Experiments with human pose estimation methods . . . 85
6.1.1 Training results and iterations . . . 86
Direct regression (Model A) . . . 86
Heatmap detection (Model B) . . . 88
6.1.2 Benchmark between models . . . 90
6.2 Temporal smoothing with LSTM . . . 92
6.2.1 Training results . . . 93
6.2.2 Inference analysis and prediction visualization . . . 93
6.3 Camera system calibration results . . . 95
6.4 Summary . . . 95

7 Conclusions **99**
7.1 Future work . . . 100

References **101**

Appendix A Benchmark results **123**

A.1 Cross-validation results of the ResNet18 architecture on the MPII dataset 123

A.2 Cross-validation results of the ResNet18 architecture on the Pandora dataset 124

List of Figures

2.1 Causes and mechanisms that lead to stroke. 8
2.2 Task specific training example where a patient pours a drink into a cup. 10
2.3 Task-oriented training with unaffected limb restricted. Tasks include: (A) cutting bread, (B) pouring water, (C) picking up and placing back money, and (D) playing a game. 11
2.4 Therapy with a BATRAC device. 11
2.5 Setup used for mirror therapy. 12
2.6 Neuro-rehabilitation session with MindMotion™ PRO being used for early bedside therapy. 13
2.7 Setup used to measure arm stiffness during therapy. 13

3.1 Representation of object triangulation with stereo-vision. 19
3.2 Inference examples using the Kinect SDK. The left column shows the ground truth neutral pose. The following columns display sequences of depth images, inferred body parts, and joint proposals from front, side, and top views. Top rows use synthetic data, the middle show real-person examples, and the bottom illustrate failure modes. 20
3.3 Commonly used human body models in HPE: (a) skeleton-based model, (b) contour-based models, (c) volume-based models. 22
3.4 Example of multiple possibilities of 3D pose reconstructions from similar 2D joint locations. 23
3.5 Heatmap based method. Given the ground-truth 2D pose (right), each joint has a heatmap generated by a Gaussian kernel. Then, prediction of each joint heatmap is done with a model (left), and loss is calculated between the two. 25
3.6 Comparison between top-down and bottom-up approach. Both consist of a two stage process in order to identify and estimate human poses in a multi-person scenario. 26

3.7 Different pose representations for multiple person in original image (a); (b) shows the conventional joint representation by absolute coordinates; (c) showcases the SPR with joints relative to a central person-identifying joint. Finally, (d) adopts an hierarchical SPR where joints are represented by vectors relative to the attached limb, starting from their person-identifying joint. 27
3.8 Architecture of AlexNet neural network backbone. 27
3.9 VGG neural network architecture. 28
3.10 Residual learning block with identity feed-forwarding. 29
3.11 Faster R-CNN unified network composed of two stages. 29
3.12 Mask R-CNN framework for parallel segmentation of instances. . . . 30
3.13 Architecture of a memory cell and gate units inside a block used in LSTM networks. 30
3.14 U-Net architecture composed of various convolutional operations at different resolution sizes, achieving the physical aspect of the letter 'U'. 31
3.15 Stacked hourglass architecture and single hourglass module architecture. 32
3.16 Cascaded pyramid network architecture. 32
3.17 Architecture of a HRNet with four stages. 33
3.18 SelecSLS module design with concatenation-skip connections. 33
3.19 Samples illustrations of dataset contents: a) UNIPD-BPE, b) UI-PRMD, c) FLIC, d) Human3.6M, e) Occlusion-Person, f) IntelliRehabDS, g) MPII, h) Extended BBC Pose. 37
3.20 Architecture of DeepPose method. 40
3.21 AlphaPose system architecture. 41

4.1 Occupancy of a person's upper body centered and fully captured by the camera system. 48
4.2 Occupancy of a person's affected limb centered in relation to the camera system. 49
4.3 Occupancy of a person's affected limb centered and body rotated by 15° in relation to the camera system. The central pose with the arm up makes the hand go out-of-view. 49
4.4 Views of the rehabilitation scenario: (a) top view, and (b) back view. 50
4.5 Proposed architecture diagram. 51
4.6 Training methods using ImageNet. The 2D pose estimator of the proposed architecture would be considered the classifier of the network, which defines the output number, while the backbone is the feature extractor. 54

5.1 Disposition and nomenclatures of the keypoints from the COCO dataset. 59
5.2 Sample images from the COCO dataset overlapped with keypoints. . 60

5.3 Disposition and nomenclatures of the keypoints from the MPII dataset. 60
5.4 Sample images from the MPII dataset overlapped with keypoints. . . 61
5.5 Disposition and nomenclatures of the keypoints from the Pandora dataset. 61
5.6 Sequence of RGB frames and corresponding keypoints from the Pandora dataset. 62
5.7 Sequence of depth frames and corresponding keypoints from the Pandora dataset. 62
5.8 Individual heatmaps created from ground truth keypoint coordinates of the Pandora dataset. 63
5.9 Image overlapped with heatmaps converted from ground-truth keypoints from the Pandora Dataset. 63
5.10 Visualization of 3D keypoints from the Pandora dataset. 64
5.11 Training and validation dataset splitting from the original MPII dataset. 65
5.12 KFold division representation of a dataset with shuffled indices and $K = 5$. 66
5.13 ResNet-34 architecture diagram with final fully connected (dense) layer modified for HPE. 67
5.14 Influence of different learning rates in the convergence to an optimal solution. 68
5.15 2D Gaussian function centered in $x = 1$ and $y = 1$ with $s = 1$ and $a = 1$. 72
5.16 Comparison between keypoint structure of inferred skeleton and original dataset due to differences in connection lists. 73
5.17 Output image comparison between the developed and the `draw_keypoints` function from PyTorch for keypoint visualization: a) developed function, and b) built-in function from PyTorch. 74
5.18 Multiple images from the Pandora, COCO and MPII datasets overlapped simultaneously by different methods. 74
5.19 LSTM network prediction of next value in sine wave. 76
5.20 LSTM network prediction of next value in a random sum of numbers. 77
5.21 Comparison between real values and LSTM predictions for multiple points before hyper-parameter tuning. 78
5.22 Comparison between real values and LSTM predictions for multiple points after hyper-parameter tuning. 78
5.23 Past joint coordinates from the Pandora dataset. 79
5.24 Video feed of the Microsoft Kinect camera system, composed by the infrared receptor, RGB sensor and the processed feed of depth pixels colored with the corresponding RGB pixels and a grayscale representation of the depth feed. 80

5.25 Kinect v2 calibration with a checkerboard being performed. 81

6.1 Training behaviour of a ResNet18 network on two different datasets: Pandora and MPII. 87
6.2 Inference results of the ResNet18 model trained and tested on the same MPII dataset. The red dots are inferred and the blue are ground-truth. 88
6.3 Inference results of the ResNet18 model trained and tested on the same Pandora dataset. The red dots are inferred and the blue are ground-truth. 88
6.4 U-Net training and validation loss during learning exhibiting a different scale compared to direct regression. This discrepancy arises because the loss is calculated between heatmaps that predominantly contain zeros on most pixels. 89
6.5 Inference results of images from the MPII dataset. Each row represents a new iteration of U-Net weights being updated during training. 90
6.6 Inference results on a sample image from the COCO dataset. The position of the heatmap peaks is mostly correct, but it can sometimes have problems distinguishing sides. 91
6.7 Inference results on a sample image from the Pandora dataset. The left hand is out of the image, which is noticed by the model's generated heatmaps for the keypoints of that region, which contain a different tone of blue background (low confidence). 91
6.8 Inference with keypoint representation of heatmaps with the U-Net on the Pandora dataset. (a) Heatmap generated keypoints with no threshold and (b) Heatmap generated keypoints with 80% threshold. 91
6.9 Validation losses during cross validation using a ResNet18 architecture on the MPII dataset. 92
6.10 Training and validation losses for the LSTM model at the end of the training loop. 93
6.11 Frames from the Pandora dataset demonstrating the difficulties of the LSTM model in estimating the current keypoint position. 95
6.12 Normal distribution of distances from LSTM previsions to ground truth keypoints on the Pandora dataset. 96
6.13 Visual comparison between: (a) default Kinect v2 calibration settings, and (b) manual calibration results. 96

A.1 Average Percentage of Correct Keypoints (head) (PCKh)@0.5 on the Max Planck Institute for Informatics (MPII) dataset across five ResNet18 folds. 123

A.2 Average Object Keypoint Similarity (OKS) on the MPII dataset across five ResNet18 folds. 124
A.3 Average Percent of Detected Joints (PDJ) on the MPII dataset across five ResNet18 folds. 124
A.4 Average PCKh@0.5 on the Pandora dataset across five ResNet18 folds. 125
A.5 Average OKS on the Pandora dataset across five ResNet18 folds. . . 125
A.6 Average PDJ on the Pandora dataset across five ResNet18 folds. . . 125

List of Tables

1.1 Thesis work plan. 5

2.1 Upper extremity outcome measures. 9
2.2 Compilation of exercises present in literature for upper limb rehabilitation. 14

3.1 Backbones commonly found for feature extraction in HPE literature. 34
3.2 Popular datasets for 2D and 3D pose estimation. 38
3.3 Deep-learning based methods used for HPE. R stands for regression, H for heatmap, BU for bottom-up and TD for top-down. All models use a kinematic body model for representations. 43

4.1 Technical specifications of the Microsoft Kinect v2 RGB-D camera system 52
4.2 Comparison of different ResNet architectures. The inference times are measured from a system running Ubuntu 22.04 with a Nvidia RTX 4060 Mobile GPU with 8 GB of VRAM. 53

5.1 Comparison of the execution speed between the developed and the `draw_keypoints` function from PyTorch for keypoint visualization. . 74

6.1 Description of the models tested for the HPE method. 86
6.2 Attempts in order to reach a different outcome during the training with the MPII dataset. 87
6.3 Inference times of the tested models. The test considers the time it takes to get from the image (as input) to keypoints (output) of a single image. The inference is performed using CUDA as the computational device, with each model running 10 times across 10 iterations 92
6.4 Prediction results when performing inference on Pandora data. All values are in pixels. 94

List of Acronyms

3DPCK	3D Percentage of Correct Keypoints
ADL	Activities of Daily Living
AP	Average Precision
APIs	Application Programming Interfaces
APK	Average Precision of Keypoints
ASMs	Active Shape Models
AUC	Area Under the Curve
AVC	Acidente vascular cerebral
BATRAC	Bilateral Arm Trainer with Rhythmic Auditory Cueing
CIMT	Constraint-Induced Movement Therapy
CLI	Command Line Interface
CNN	Convolutional Neural Networks
COCO	Common Objects in Context
CPN	Cascaded Pyramid Network
CPSR	Canadian Partnership for Stroke Recovery
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DALYs	Disability-Adjusted Life-Years
DNN	Deep Neural Network
EBRSR	Evidence-Based Review of Stroke Rehabilitation
FMA	Fugl-Meyer Assessment
FMA-UE	Fugl-Meyer Assessment for Upper Extremity

FNT	Finger to Nose Test
FPS	Frames Per Second
GFLOPS	GigaFLoating point Operations Per Second
GPU	Graphics Processing Unit
HOG	Histograms of Oriented Gradient
HPE	Human Pose Estimation
HRNet	High-Resolution Net
IEF	Iterative Error Feedback
IMUs	Inertial Measurement Units
ISBI	International Symposium on Biomedical Imaging
KVIQ	Kinesthetic Visual Imagery Questionnaire
LSTM	Long Short-Term Memory
LSVRC	Large Scale Visual Recognition Challenge
mAP	mean Average Precision
MEEAR	Maximal Elbow Extension Angle during Reach
MMST	Manual Muscle Strength Test
MoCap	Motion Capture
MOTA	Multiple Object Tracker Accuracy
MOTP	Part Affinity Fields
MPII	Max Planck Institute for Informatics
MPJPE	Mean Per Joint Position Error
NFDS	Neurological Function Deficit Scale
OKS	Object Keypoint Similarity
PA-MPJPE	Procrustes Analysis Mean Per Joint Position Error
PAF	Part Affinity Fields
PCK	Percentage of Correct Keypoints

PCKh Percentage of Correct Keypoints (head)

PCP Percentage of Correct Parts

PDJ Percent of Detected Joints

PICH Primary IntraCerebral Hemorrhage

R-CNN Region-based Convolutional Neural Network

ReLUs Rectified Linear Units

REPAS REsistance to PAssive movement Scale

RGB Red Green Blue

RGB-D Red Green Blue and Depth

RMS Root Mean Square

ROS Robot Operating System

RPN Region Proposal Network

SCAPE Shape Completion and Animation for PEople

SDK Software Development Kit

sEMG surface ElectroMyoGraphy

SMPL Skinned Multi-Person Linear model

SPR Structured Pose Representation

SSD Solid-State Drive

TOF Time Of Flight

UI-PRMD University of Idaho - Physical Rehabilitation Movements Data Set

UPSET UPper limb Self-Efficacy Test

UTD-MHAD University of Texas at Dallas - Multimodal Human Action Dataset

VRAM Video Random Access Memory

WSL Windows Subsystem for Linux

WSO World Stroke Organization

Chapter 1

Introduction

Advancements in computer vision and machine learning have facilitated a new outlook on rehabilitation, enabling the search for different solutions for post-stroke rehabilitation, in this case, of the upper-body. Employing vision-based methods to evaluate the pose and motion of the human body, holds promise to enhance recovery outcomes, personalize therapy and improve the overall quality of life for individuals undergoing rehabilitation.

1.1 Contextualization

Human Pose Estimation (HPE) is a task where the position and orientation of a person's body joints are computed to be available as data for other applications. This sugests that it is a multidisciplinary subject, since the study of human movement has applications and increasingly interest in various fields such as robotics, human-computer interaction, healthcare, augmented and virtual reality [1]. The accurate translation of movements, from a real person to computer data can then be used to recognize gestures, help develop immersive gaming and improve rehabilitation and physical therapy.

The research in HPE has advanced rapidly, with some works making use of Inertial Measurement Units (IMUs) that can be attached to garments or be integrated in special clothes, but may cause inaccuracies if misaligned and suffer from heading drift [2].

Out of the different methods used to detect the human pose, vision sensors can be used without wearable visual markers, allowing the patient to accomplish a wide range of exercises without being hindered.

Taking the computer vision approach means that the most relevant features can be estimated with high accuracy, leading to more efficient rehabilitation.

1.2 Motivation

The application of machine learning and computer vision hold immense promise for stroke rehabilitation. By making use of these technologies, rehabilitation can be personalized and optimized, leading to faster recovery and improved functional outcomes for stroke survivors, as well as making these services more accessible and cost-effective worldwide. Machine learning and computer vision have broad range of use and great potential to expand as they are integrated in more specific areas and help in more tasks.

The scientific and technological advances on computer vision and machine learning should be harnessed for other areas, where current solutions are time-consuming and might have high implementation costs. That said, the use of computer vision in healthcare fields is one of the most prominent in developments, since it directly impacts the quality of life of everyone, thus, bringing better solutions is a big motivation factor.

1.3 Problem description

Strokes are on the rise around the world [3], gradually increasing since 1990, accounting for about 6.6 million deaths (ranking as the second leading cause of death) as well as the third leading cause of disability, contributing to 143 million Disability-Adjusted Life-Years (DALYs) and, is increasing in individuals under 55 years [4]. Disabilities linger with stroke survivors, which can mean living with motor, sensorial and other impairments that hinder daily activities [4]. Although recovery is dependent on various factors, like socioeconomic status, severity of stroke and age, rehabilitation aims to improve functional activities and extend life without disability. This is a cause for worry that accumulates on rehabilitation, where professionals spend precious time in directing the individuals on how to do the exercise and monitoring the exercises. This means that each individual might be getting less time with the professional. To solve this, in some situations, patients do these rehabilitation exercises from home or by having multiple individuals in the same clinic room, each one of them using a device that guides them and frees up some time for the professional [4].

The monitoring can be achievable with wearable devices which contain sensors that collect data about the motion carried out, and then be analyzed by the professional, allowing some degree of monitoring of the exercises [5]. However, these devices can be cumbersome and can provide wrong results, either because they were poorly placed or they hinder the movements, and get in the way of meaningful rehabilitation [6]. Markerless vision methods provide a way to perform these exercises without constrains, while also being precise and can provide additional data for subsequent analysis [7].

1.3.1 Objectives

The work described in this dissertation aims to:

- Study stroke rehabilitation exercises and motions.
- Analyse state of the art implementations of pose estimation in various domains and approaches, such as full body pose estimation and novel algorithms.
- Review computer vision and machine learning techniques and explore their applicability in the pose estimation context.
- Compare solutions and build the architecture for the pose estimation method.
- Develop a pose estimation method for upper limb rehabilitation.
- Assess the performance and precision of the method against other implementations.

1.3.2 Achieved results

The work described in this dissertation addresses the development of a human pose estimation method, conceived for upper limb rehabilitation, after a careful and attentive literature review of related topics such as stroke rehabilitation, computer vision and machine learning.

The HPE approaches are then evaluated during training, tracking the loss evolution, and utilizing metrics such as PCKh, OKS and PDJ to evaluate cross-validation inference results, as well as visual comparisons against ground-truth data. Performance for real-time usage is also important, and, as such, the inference time against a state-of-the-art implementation is also evaluated.

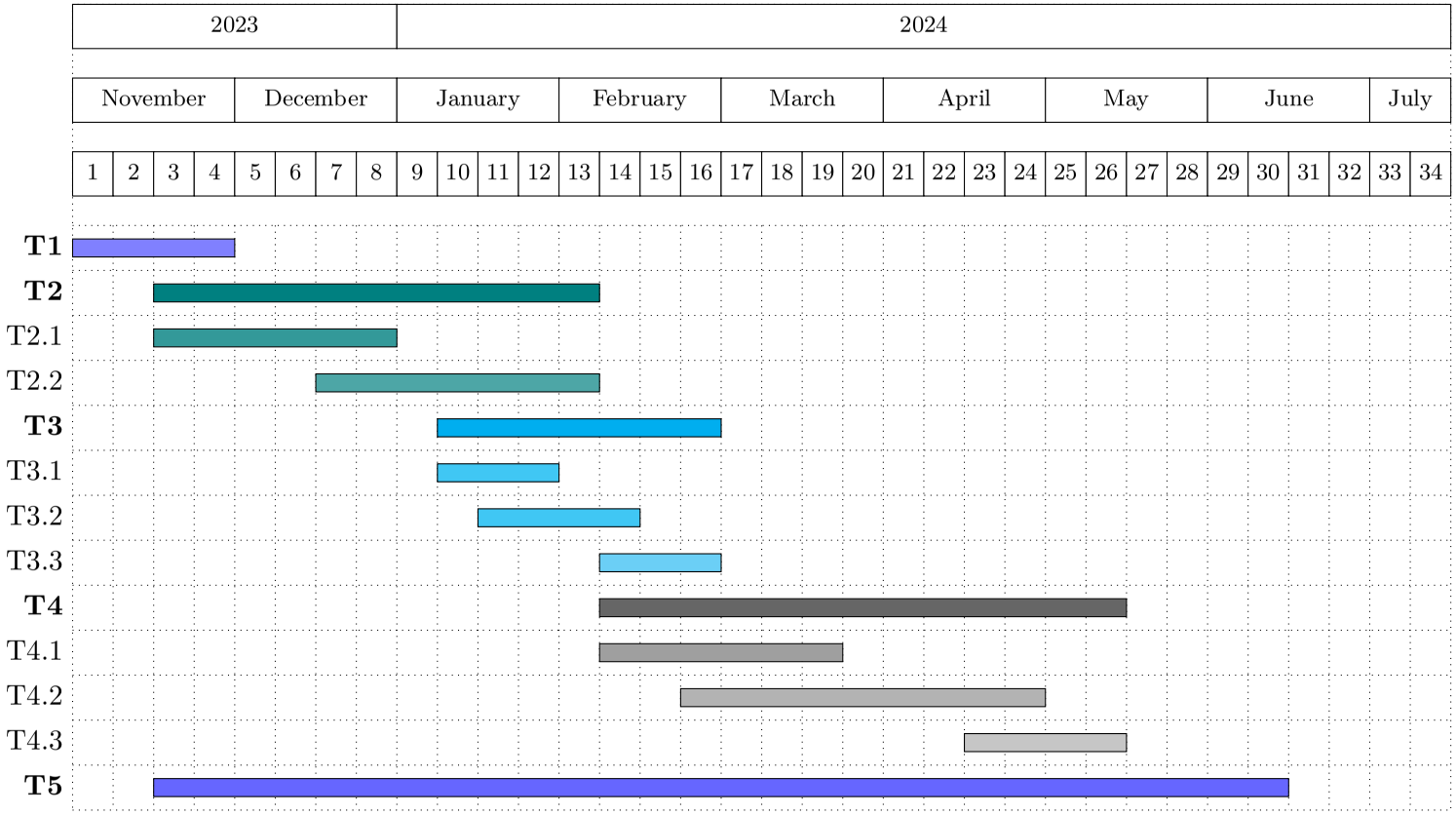
1.4 Work plan

For a better understanding of the work schedule, as well as a better time organization, a planning was made to fulfill the imposed goals. This plan can be seen in

Table 1.1. The plan, however, was not fulfilled due to setbacks during model training, which caused delays and prevented the achievement of desired results within the expected timeframe, extending the development phase and delaying results and the dissertation writing.

- **T1** - Problem formulation and objective definition
- **T2** - Fundamentals of stroke rehabilitation
  - T2.1 - Impact of stroke
  - T2.2 - Rehabilitation exercises and movements for recovery
- **T3** - Literature review
  - T3.1 - Sensor technology for pose estimation
  - T3.2 - Computer vision techniques
  - T3.3 - Datasets and trained models
- **T4** - Implementation of the pose estimation method
  - T4.1 - Architecture proposal
  - T4.2 - Implementation development
  - T4.3 - Performance testing and validation
- **T5** - Dissertation Writing

Table 1.1: Thesis work plan.



1.5 Dissertation structure

This document is organized in a way that allows for a clear flow of topics discussed, where many of them are related to each other, followed by the description of the project development and final conclusions.

Chapter 2 starts by clarifying stroke concepts, such as the different types of stroke, post-stroke rehabilitation and efficacy of rehabilitation methods, outcome measures for evaluating physical function evolution and patient recovery routines and movements.

Chapter 3 examines the pose estimation topic, beginning by exploring sensor technology currently in use and, afterwards, gives a general vision of different pose estimation methods, evaluation metrics and datasets, as well as different types of architecture backbone commonly employed in HPE.

Chapter 4, the first that illustrates the development of a method for human pose estimation, presents and details the architecture for the system, allowing a better comprehension of the path to follow. The system is described as a whole, explaining the architecture beginning with the video input to the pretended 2D pose estimation modules and ending with the depth registration needed to obtain 3D keypoints.

Chapter 5 details the implementation of the system, outlining the essential steps taken to develop an effective HPE method, detailing key steps from dataset preparation to model training and inference. It begins with an analysis of selected datasets, emphasizing data quality and visualization techniques. The chapter focuses on optimizing the neural network training process, discusses inference methods for interpreting results, and examines the use of Long Short-Term Memory (LSTM) networks to improve temporal consistency. Finally, it addresses practical aspects of camera calibration and device configuration for accurate pose estimation.

In Chapter 6, testing and evaluation using various models is performed on HPE models, including ResNet-18, U-Net, and Keypoint Region-based Convolutional Neural Network (R-CNN). The chapter detais the different approaches employed by each model and the datasets utilized for training and inference. It also discusses the implementation of temporal smoothing to enhance keypoint tracking over time. Additionally, the chapter addresses the camera calibration process, examining its impact on the alignment of Red Green Blue (RGB) and depth data.

Lastly, Chapter 7 summarizes the key findings of this dissertation, highlighting the potential of vision-based human pose estimation methods in enhancing upper limb rehabilitation for post-stroke patients. It addresses the challenges encountered during development, such as dataset limitations and model training times, while emphasizing the promise of integrating temporal smoothing for improved accuracy. The chapter also outlines future improvements that can be applied to the developed method.

Chapter 2

Stroke rehabilitation

Strokes are responsible for a large portion of disability in the world, leaving a significant percentage of individuals with different levels of limb paralysis, depression and language disturbances, even 14 years after stroke [8]. The recovery process is seen as necessary to the improvement of function of the body, but recovery must also make efforts in evaluating psychological and social condition and capturing Activities of Daily Living (ADL) measurements to better assess the recovery process [9].

2.1 What is a stroke

A stroke is a syndrome characterized by loss of cerebral function with a vascular origin of the central nervous system, being the second leading cause of death [10]. Strokes can be caused by various circumstances, and can differ in type and sub-type, but hypertension is the most significant malleable risk factor, as seen in Figure 2.1 [10].

The most common stroke type is ischaemic (85%) and, depending on the cause, can be further characterized as cerebral small vessel disease, cardioembolism or large artery atherosclerosis-related thromboembolism. Around 15% of strokes are caused by intracerebral haemorrhage, and out of those around 80% result from cerebral small vessel diseases [10]. Being so heterogeneous, just like other syndromes, the prognosis, treatment and preventive strategies are highly dependant on case to case [11].

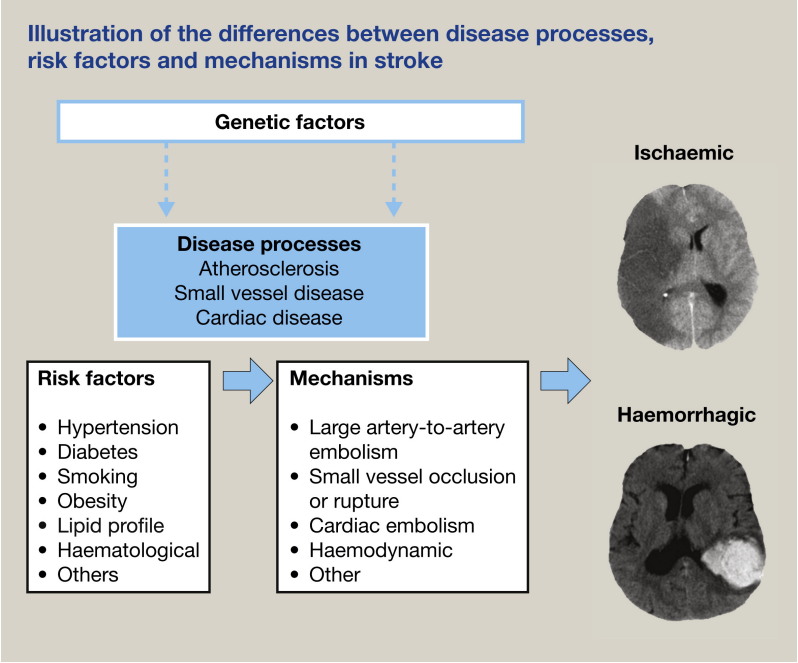


Figure 2.1: Causes and mechanisms that lead to stroke [10].

After a stroke, about 20% of individuals die in the first month (early case-fatality), and one third of those who reach 6 months become dependent on others for daily life [11].

While recording stroke incidence with accuracy is difficult, monitoring over time is even harder, since distinguishing different pathological types of stroke omitted in death certificates (ischaemic stroke and Primary IntraCerebral Hemorrhage (PICH)) might be hard [11]. Moreover, while stroke incidence based on community samples provide a better understanding on stroke burden, they require more resources and rigid methods, as well as accurate diagnostic [11].

2.2 Post-stroke rehabilitation

Stroke rehabilitation can assist in reducing disabilities and complications derived from a stroke, allowing individuals to reclaim their independence [12].

Stroke rehabilitation intends to facilitate neuroplasticity (how well the nervous system can change activity in response to stimuli) utilizing clearly defined tasks and goals, which involves repetitive, but yet challenging, training that allows for gains in capabilities [4].

According to the World Stroke Organization (WSO) [4], effective rehabilitation services must provide a multidisciplinary team, a personalized approach for the goal of the patient and equipment and facilities. The lack or unavailability of services can explain the difference in recovery between different income countries.

Telerehabilitation and telemedicine are seen as emerging solutions for the low availability of these services, but it offers challenges such as the equipment setup, possible scope of exercises or problems with evaluation, interface or time constrains of the professionals [4]. In 2018, Sarfo *et al.* analysed telerehabilitation efficacy, and found that only 8 out of 22 studies were in favor of telerehabilitation, as it could be essential for under-resourced areas, but, for global use of telerehabilitation, larger and longer-term studies are necessary [13].

High-quality rehabilitation is aided by the implementation of evidence-based care [4], such as those pointed by Evidence-Based Review of Stroke Rehabilitation (EBRSR), funded by the Canadian Partnership for Stroke Recovery (CPSR), which provide up-to-date reviews of evidence in stroke rehabilitation, delving into the effectiveness of both pharmacological and non-pharmacological interventions [14].

2.2.1 Outcome assessment methods

The assessment of post-stroke body function and its resulting outcomes can be categorized into various domains. Several approaches offer a baseline benchmark for evaluation, utilizing metrics like motor function, dexterity, and daily life activities [15]. Table 2.1 showcases diverse upper limb metrics alongside evaluation methods.

Table 2.1: Upper extremity outcome measures [15].

Measures	Description	Example
Motor Function	Functional motor movements	Fugl-Meyer Assessment (FMA) [16]
Dexterity	Fine motor and manual skills (particularly with hand)	Finger to Nose Test (FNT) [17]
Activities of daily living	Level of independence in everyday tasks	UPper limb Self-Efficacy Test (UPSET) [18]
Spasticity	Muscle tone, stiffness and contractures	REsistance to PAssive movement Scale (REPAS) [19]
Range of motion	Ability to move through flexion, abduction and subluxation	Maximal Elbow Extension Angle during Reach (MEEAR) [20]
Proprioception	Sensor awareness about body and location of limbs	Kinesthetic Visual Imagery Questionnaire (KVIQ) [21]
Stroke severity	Measure of stroke severity through assessment of deficits	Neurological Function Deficit Scale (NFDS) [22]
Muscle strength	Muscle power and strength during movement	Manual Muscle Strength Test (MMST) [23]

2.2.2 Upper limb intervention

Upper limb problems are very common after a stroke, which include difficulty when moving and positioning the arms, hands and fingers, as well as coordination problems, that hinder activities of the day-to-day life [24]. Since the majority of people

living with upper limb impairment after a stroke keeps having problems years after the stroke, rehabilitation is a mean to improving arm function. Rehabilitation is one of the possible interventions, involving the execution of different exercises and techniques that involve the effort of the individual, carers and rehabilitation team [24]. The effectiveness of interventions to improve arm function after stroke differ greatly, existing distinguishable methods with favorable results [15, 24].

Task-specific training comprises the practice of everyday tasks, like the one present in Figure 2.2, both full movement or partial [24]. It can be used in repetitive task training, and supplemented by assistive technologies, such as interactive games [24]. Due to the repetitive practice of new functional skills and intensity nature of it, it can reduce muscle weakness and cement the baseline for motor learning [25]. This training is well received and provides longer lasting cortical reorganization to traditional stroke rehabilitation [26].

Figure 2.2: Task specific training example where a patient pours a drink into a cup [27].

Rehabilitation with Constraint-Induced Movement Therapy (CIMT) is seen as beneficial on measures of upper limb function [24]. The methodology is usually to restrain the non-paretic upper limb in order to overcome the tendency of using it [15]. The method presents small to moderate benefits in short term [28], but the results fade after some months [29, 30], or does not register differential effects in comparison to traditional therapy [31, 32]. This method can be combined with another, such as task-specific training seen in Figure 2.3.

Bilateral arm training is a technique where both upper limbs perform the same movement simultaneously [15]. The therapy, shown in Figure 2.4, is being adopted recently since it is thought to have positive impact on neural plasticity [34], where the use of the unaffected limb, in theory, improves recovery via coupling effects

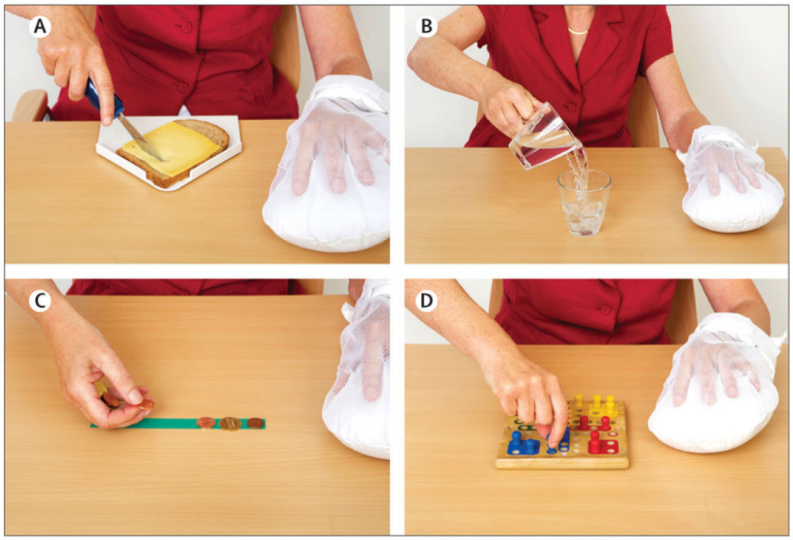


Figure 2.3: Task-oriented training with unaffected limb restricted. Tasks include: (A) cutting bread, (B) pouring water, (C) picking up and placing back money, and (D) playing a game [33].

between cerebral hemispheres [35]. The theory, however, is not backed by literature, since results often indicate no advantage compared to unilateral training [15, 24, 35].

Figure 2.4: Therapy with a Bilateral Arm Trainer with Rhythmic Auditory Cueing (BATRAC) device [36].

Mirror therapy is considered beneficial at all stages after stroke, consisting of a setup with a mirror positioned between both hands, mirroring the unaffected hand, as seen in Figure 2.5 [37]. This way, when manual exercises are executed, patients can stimulate the brain part involved in action imitation [38]. Unilateral hand movement of the unaffected limb can then have benefits for the functional recovery of the impaired limb [34]. According to a research note from Garry *et al.*, this might

also be suggested by bilateral movement therapy [34].

Figure 2.5: Setup used for mirror therapy [37].

Technology based interventions, such as rehabilitation programs making use of robotics or virtual reality, are recent candidates that show promising results. In the case of virtual reality, where research is increasing as technology becomes more accessible [15], the rehabilitation programs can be presented as games with haptic or visual feedback, where interaction with the simulated environment creates a sensory experience, aligning with the principles of rehabilitation: intensity, task-specific training, biofeedback and motivation [39]. The biggest impact from virtual reality was felt in motivation and engagement of patients [39], nevertheless, virtual reality did not provide better results compared to traditional therapy but might be a good option to increase therapy time [40, 41]. Some commercially available solutions, such as the ones comercialized by MindMaze like the one displayed in Figure 2.6, exist for home [1] and clinic[2], allowing for extended therapy time [42].

Electromechanical and robot assisted arm training may also be used to strengthen other methods, such as task-specific training, helping motor learning and increasing motor capabilities [24]. There are a variety of collaborative devices capable of assisting with rehabilitation, adapted to different situations [43]. According to Lam *et al.* [44], robot assisted therapy can help severely impaired patients, especially during the acute phase of recovery.

Piovesan *et al.* [45] used a technique to measure arm stiffness with robot therapy in a trial that involved a seated individual holding the handle of the robot and a restraint cast to limit wrist movement. The individual then moved the arm from and to different points highlighted on a computer screen, as seen in Figure 2.7. This

[1] `https://mindmaze.com/digital-therapies-for-neurorehabilitation/#mindmotion-go` (last accessed in 21/09/2024).

[2] `https://mindmaze.com/digital-therapies-for-neurorehabilitation/#mindmotion-pro` (last accessed in 21/09/2024).

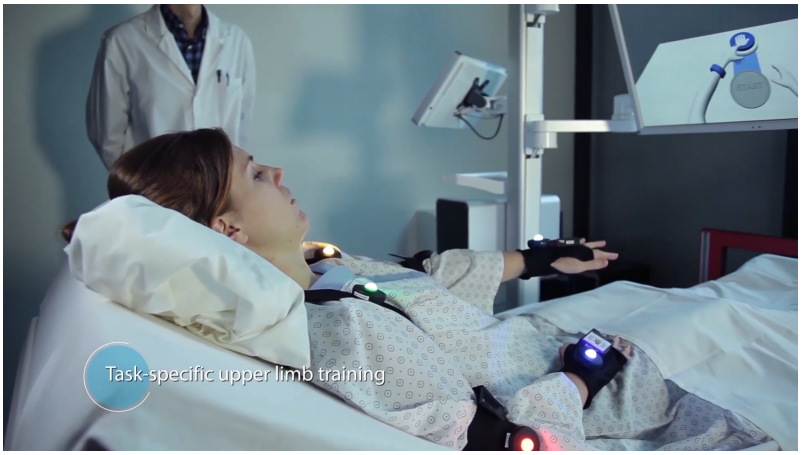


Figure 2.6: Neuro-rehabilitation session with MindMotion™ PRO being used for early bedside therapy [42].

allows for shoulder and elbow flexion/extension movements to be evaluated using the robot encoders [45].

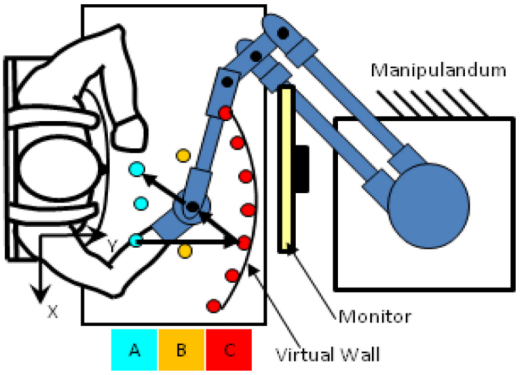


Figure 2.7: Setup used to measure arm stiffness during therapy [45].

2.3 Employed exercises in upper limb rehabilitation

Subsection 2.2.2 presented some of the interventions with evidence for favourable results. With base on those results, a subset of the exercises used in them is detailed in Table 2.2. This way, it is possible to have a better understanding of movements and poses, and pay special attention to the dataset used for the implementation.

Table 2.2: Compilation of exercises present in literature for upper limb rehabilitation.

Author	Gesture	Description
Wang *et al.* [46] (2024)	Hand touch mouth	Arm down with palm facing front, bend elbow to bring hand up until it can touch the lips.
	Hand touch back of neck	Arm down with palm facing front, shoulder adduction, bend elbow, forearm pronation.
	Hand touch opposite knee joint	Arm down with palm facing body, touch outer side of contralateral knee joint.
	Forearm supination	Arm down with palm facing body, forearm supination 80^{o} to 90^{o}.
	Forearm pronation	Arm down with palm facing body, forearm pronation 80^{o} to 90^{o}.
	Touch the waist	Arm down with palm facing body, bend elbow and place back of hand on same waist.
Wang *et al.* [46] (2024) & Sun *et al.* [47] (2022)	Shoulder flexion	Arm down with palm facing body, shoulder forward until 180^{o} and elbow extension.
	Shoulder adduction	Arm down with palm facing body, shoulder to side (adduction) until 180^{o} and elbow extension.
Sun *et al.* [47] (2022)	Touching the height	Arm down with a closed fist, lift hand above head height.
	Drinking	Arm down with a closed fist, elbow flexion to reach above the chest.
	Trouser lifting	Hand down pinching near side of waist, lift arm a little.
	Elbow extension	Arm near chest, extend it completely (horizontally).
	Elbow flexion	Arm extended in front of body, flexion into chest.
Mahmoud *et al.* [48] (2021)	Grabbing an object	Arm down while holding object, raise arm over head at more than 45^{o}, slowly put it down.
	Object on elevator	Arm down while holding object, lift up to a certain height, down, up and down, raise to above horizontal plane and slowly put it down.
	Object up and right	Arm down while holding object, lift to certain height, move horizontally to side, slowly put it down.

2.4 Summary

Chapter 2 introduced the context behind stroke rehabilitation and the different exercises during therapy, to have a firm base to complete posterior work. As mentioned in Subsection 2.1, stroke leaves increasingly more individuals with varying levels of limb paralysis and other problems. Because of this, strategies for recovery should be designed with the individual in mind.

Next, in Subsection 2.2, some outcome assessment methods are first explored in Sub-Subsection 2.2.1, since they allow for professionals to evaluate recovery and body function using well established measures, such as the Fugl-Meyer Assessment for Upper Extremity (FMA-UE) [49], comprised of the upper extremity evaluation tests. After, Sub-Subsection 2.2.2 introduces credited therapy methods, such as mirror therapy or CIMT, that have different levels of benefits and long term effect. While some therapy can be done independently, combining different therapies can have a positive impact on motivation and increase the overall therapy duration.

Finally, Subsection 2.3 presents some of the therapy exercises found in existing literature, offering a comprehensive exploration of the routines commonly employed in daily rehabilitation sessions. By examining these exercises, this section provides an overview of the collective muscle engagement patterns observed in group settings, providing valuable insights into the efficacy and practicality of various therapeutic interventions

Chapter 3

Computer vision and pose estimation

Human Pose Estimation (HPE) is a task that uses sensor inputs to locate the posture of the human body. For that task, there are various approaches, from the different type of sensors used to the algorithms adopted, which is highly dependant on the scenario and number of people which the algorithm is expected to extract and estimate the poses from [50].

3.1 Sensor technologies for human pose capture

Various challenges are presented, such as occlusion, which is considered a large obstacle for pose estimation when relying exclusively on camera systems [51]. Some implementations oppose the problem with wearable technology containing IMUs, but they can also be inherently problematic, due to misalignments and heading drift [2, 52]. One possible solution is to use a combination of both IMUs and camera system, which are found to increase reliability and improve tracking [2, 52].

The use of surface ElectroMyoGraphy (sEMG) signals is also commonly found in rehabilitation literature, notably because their signal contains valuable muscle excitation and contraction information [53] which can be used to classify movements and act as an input for control algorithms of robots [54, 55, 56] and emergency stop [57]. However, they are found to suffer from external environment perturbations and

have trouble acquiring the weak sEMG signals from the body, requiring excellent signal processing [58, 59].

Implementations utilizing IMUs for HPE or Motion Capture (MoCap) integrated in wearable sensors have been emerging, for their inherent capability of capturing acceleration, tilt, vibration and rotation [60]. Having non-invasive sensors on a wide workspace while achieving an accurate estimation is considered one of the biggest challenges [61]. The popularity comes from the fact that they are not affected by occlusions [62], and are considered less intrusive than sEMG [63], but still take considerable time to attach and cause discomfort [6], being also considered for stroke rehabilitation use and evaluation [5, 46, 64, 65, 66].

Video-based capture systems utilizing RGB cameras without markers are compelling because they are inexpensive and non-intrusive. However, as mentioned before, they suffer from occlusion and other aspects, such as partial observations and depth ambiguities which translate to worse accuracy and reliability compared to marker-based solutions [7]. Other implementations merge IMUs and cameras to achieve 3D pose estimation, that, when combined, offer better tracking performance and stability [7]. Kaichi *et al.* [52] tested this combination and found that, on one hand, the accuracy of position and orientation is largely affected by reducing the number of IMUs (from the original 14 IMUs), on the other hand, wearing many of them hinders range of motion and is time consuming.

Depth cameras have garnered attention due to their remarkable versatility, being used in tracking and image recognition [67]. There are different techniques implemented to measure depth, such as [67]:

- Triangulation: utilizing stereo-vision, analyse the ray-constructed triangles of two optical systems and trigonometry is used to find the distance to the surface point (Figure 3.1).
- Depth-from-focus: depth is determined by varying the focus of the camera.
- Time Of Flight (TOF): this method involves a modulated light source, an array of pixels and a optical system to focus the light onto the sensor. The distance is achieved by measuring the phase shift of the light since it leaves the source until it returns.

As an example, the original Kinect for Xbox 360 [69] camera system employs, among other sensors, a RGB camera and a depth sensor, consisting of an infrared projector and an infrared camera sensor. It emits infrared dots in a pattern and then match the dot pattern features with the infrared image (structured light approach), utilizing a normal cross correlation [69, 70]. The infrared image is then overlapped with the RGB image if needed. The Kinect allowed, with its high availability and low cost, the use of this technology outside the gaming industry, where it initially

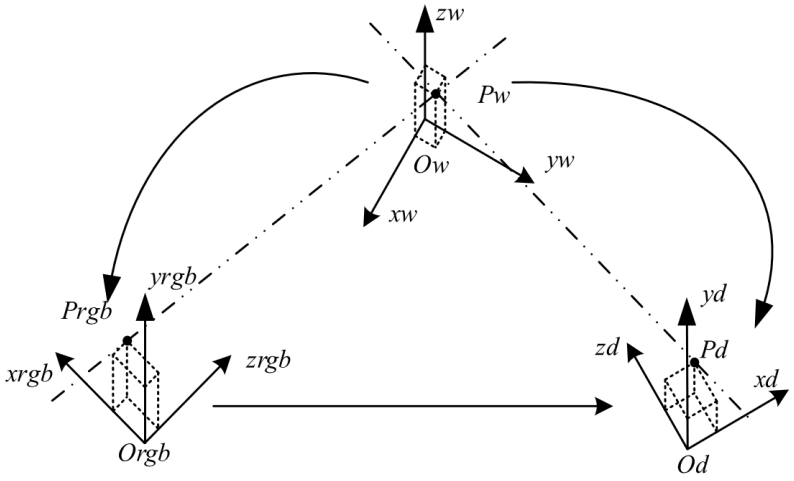


Figure 3.1: Representation of object triangulation with stereo-vision [68].

originated [69]. Future Kinect generations employed a TOF technique which is found to be more accurate in indoor scenarios with non-transparent materials in comparison to the structured light Kinect [70, 71]. Still, the Kinect family can suffer from motion blur originating from fast movements [72] due to the limited rate of capture.

An HPE algorithm is present in the Microsoft Kinect Software Development Kit (SDK) which is capable of producing a pose estimation based on a skeleton-based model. While the algorithm is considered as capable, the inferred noise from the depth sensor is pointed to be the main limitation along with the occlusion resulting from the monocular view camera. The Kinect SDK is capable of real-time pose estimation while keeping a consistent and real-world motion and physiology of the human body because it does many approximations. However, while these add robustness in gaming applications, the approximations add error to joint positions in relation to more expensive and invasive MoCap systems [73].

The Kinect SDK uses a single input depth image to perform the inference of one part of the body, accommodating for body size and shape, and can obtain direct 3D positions from joints. Furthermore, the algorithm does not use any temporal or kinematic constraint and results in video sequences show that it predicted most of the joints without major jitter [74]. The inference quality can be verified in Figure 3.2, where the algorithm is tested in both real and synthetic data.

Currently, Kinect sensors are often used in research environments, as a way to access the Red Green Blue and Depth (RGB-D) data needed for pose estimation and other computer vision tasks [70, 75]. Li *et al.* [68] utilized the Kinect V1.0 to estimate the pose of lying supine infants but found out that pose estimation using the Kinect's own algorithm is difficult due to the infants close distance to the background and algorithm being based for adults, so a different method was

Figure 3.2: Inference examples using the Kinect SDK. The left column shows the ground truth neutral pose. The following columns display sequences of depth images, inferred body parts, and joint proposals from front, side, and top views. Top rows use synthetic data, the middle show real-person examples, and the bottom illustrate failure modes [74].

developed for the task. A previous work by Wu *et al.* achieved better results when estimating the pose of lying infants by first utilizing a off-the-shelf 2D HPE and then obtain the 3D coordinates using the depth information [76]. Additionally, the Kinect is found to provide enough detail for torso, head and limbs, but not smaller body parts [77].

Over the years, the Kinect family has been recognized for its use in rehabilitation and body function literature and research [78]. Such cases include a system for fall risk assessment for elderly people, employing temporal and spacial gait parameters, which obtained good results compared to RGB camera systems, allowing for better background extraction and reducing computational requirements [79]. A system called KiReS, a telerehabilitation system developed for the Kinect for Windows v1, gives the user and physiotherapist feedback and provides customised exercise sessions adapted to the patient [80]. Schönauer and Jansen [81] developed a system for chronic pain rehabilitation comparing the Kinect v1 to iotracker [82], a motion capture system, more expensive and harder to operate, and results show that the Kinect v1 would be a remarkable alternative for the motion capture system [81].

RGB-D cameras allow for 3D HPE to acquire depth information without resorting to multi-camera systems, which require calibration [83], or software based depth estimation, which is the focus of many researchers [1]. Depth cameras present the opportunity to perform 2D to 3D pose lifting without relying on depth estimation and achieve favourable results eliminating the depth ambiguity [84].

Although the Kinect is widely used in research, other RGB-D camera systems exist on the market, and are found in rehabilitation [85]. Such camera systems include the ones available from Livox Tech[1], Intel RealSense[2], Stereolabs ZED[3], Structure Sensor Pro[4] or Luxonis[5], which employ different methods and hardware

[1] `https://www.livoxtech.com/mid-70` (last accessed in 21/09/2024).
[2] `https://www.intelrealsense.com/compare-depth-cameras/` (last accessed in 21/09/2024).
[3] `https://www.stereolabs.com/products/zed-2` (last accessed in 21/09/2024).
[4] `https://structure.io/structure-sensor-pro/specs/` (last accessed in 21/09/2024).
[5] `https://shop.luxonis.com/collections/oak-cameras-1` (last accessed in 21/09/2024).

to obtain depth and incorporate different features.

3.2 Challenges

Pose estimation methods face a multitude of challenges, namely their lack of accuracy, occlusion, the different capture scenarios, person positioning or limb association [86]. The different capture scenarios are especially troublesome since human visual appearance, light conditions and scene occlusion are diverse and manifest constantly during estimation [87]. Moreover, the human body can assume flexible body positions that might cause occlusion and be hard to identify, and can be further accentuated by the different viewing angles [50].

Video systems, where there is a need to track position of individual limbs and keep a consistent and realistic pose tracking, might need assistance when recovering from failures and manual initialization can be a problem [88]. Another important consideration for many HPE methods is the efficiency, and capability of being implemented for a real-time scenario, while maintaining temporal coherence [89]. This means that computational metrics such as the frame rate [90], number of weights, network design (in case of Convolutional Neural Networks (CNN)) or operations per second [91] are valuable for HPE [50]. The works by Ho *et al.* [92] and Ganapathi *et al.* [93] implement algorithms capable of pose estimation using depth image sequences at over 100 Frames Per Second (FPS) using a single Central Processing Unit (CPU) core.

3.3 Pose estimation approaches

The process of HPE can be encapsulated into two steps [86]:

1. Finding the location of body joints or keypoints.
2. Aggregating the joints into a valid configuration of a human pose.

In early days, pose estimation was considered a part based inference task that could be classified in two categories:

- Appearance models, in which features of the different body parts are first extracted by descriptors, like the Histograms of Oriented Gradient (HOG) descriptor [94], that separated the human body from background and foreground [95], and later combined together the different body parts [96].
- Structural models, which use articulated constraints to parse the different body parts, which helps estimation of the many positions where limbs are present [97].

3.3.1 Human body models

Human body modeling is considered one of the key components of HPE. The human body is characterized as a flexible and complex non-rigid object with a kinematic structure and various body shapes and textures [50]. A model should be such that it satisfies the requirements for specific tasks and correctly describes the human body pose [50]. There are mainly three types of adopted body models in HPE, shown in Figure 3.3, but some variations, such as the one by Belagiannis *et al.* [98] also exist. The main models are skeleton-based, countour-based and volume-based [50].

- Skeleton-based models: The skeleton-based model is also known as the stick-figure or kinematic model and represents the set of joints as vertices (typically around 10 to 30) and limbs as lines connecting them [50, 99]. This topology is widely utilized in 2D and 3D HPE [90, 100, 101, 102, 103] as well as in datasets [104] , being very simple and flexible compared to others [50].
- Countour-based models: This model, also known as planar model, which represents the human body parts as rectangles or silhouette boundaries, was used frequently in earlier HPE methods. The cardboard models [105] and Active Shape Models (ASMs) [106] are examples of popular contour models.
- Volume-based models: Represented by geometric shapes [107] or meshes in modern models used in 3D HPE. The latter is usually captured with 3D scans and include the Shape Completion and Animation for PEople (SCAPE) [108] and Skinned Multi-Person Linear model (SMPL) [109] models [50]. These models are considerably more accurate and realistic when representing human shape but have trouble dealing with unnatural deformations around joints from different bodies and poses [109].

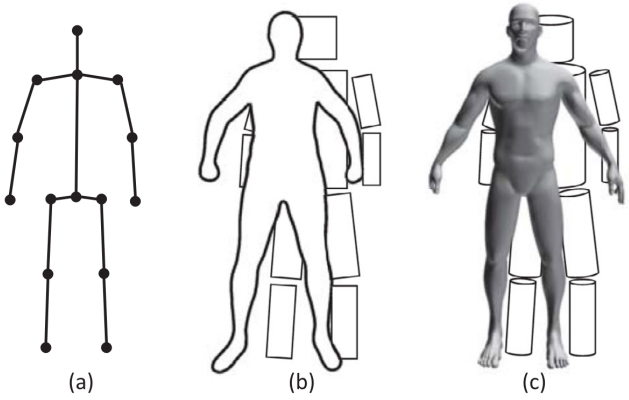


Figure 3.3: Commonly used human body models in HPE: (a) skeleton-based model, (b) contour-based models, (c) volume-based models [50].

3.3.2 2D/3D pose estimation

When considering an image as an array of 2 dimensions, 2D pose estimation utilizes monocular images or video as the input for 2D human pose estimation, calculating the location of human joints in the image as pixel coordinates and placing key points in the 2D space [50, 99].

3D HPE research is growing in popularity due to the wide application range. However, on top of the challenges of 2D estimation, 3D HPE finds a lack of in-the-wild datasets, depth ambiguities and a demand for rich posture information [110]. A common approach is to calculate 3D pose directly from the input images. An end-to-end network directly maps input images to 3D body joints and can benefit from having more supervision signals, such as depth input channels [110]. Multi-view images are also being explored for 3D HPE in some works [111, 112], having better performance than single-view methods, but require a large number of cameras and coordinated studio environments [110].

Another common approach is to recuperate 3D poses from 2D joints or heatmaps, in a two-stage approach known as 2D to 3D lifting [113, 114]. This is described as a difficult task when dealing with RGB cameras, since the algorithm should present unchanging results when facing variations like textures, imperfections, occlusions and others [99]. A single 2D skeleton might match many different 3D poses, as seen in Figure 3.4. The use of RGB-D sensors like the Kinect allows these methods to take advantage of an off-the-shelf 2D HPE model, and then estimate the 3D pose with aid from the depth data [1, 84, 115, 116], which also means that this methodology relies heavily on the 2D detection and identification, which can affect 3D performance [76]. Alignment between RGB and depth cameras must also be carried to improve the accuracy of 3D keypoints [116].

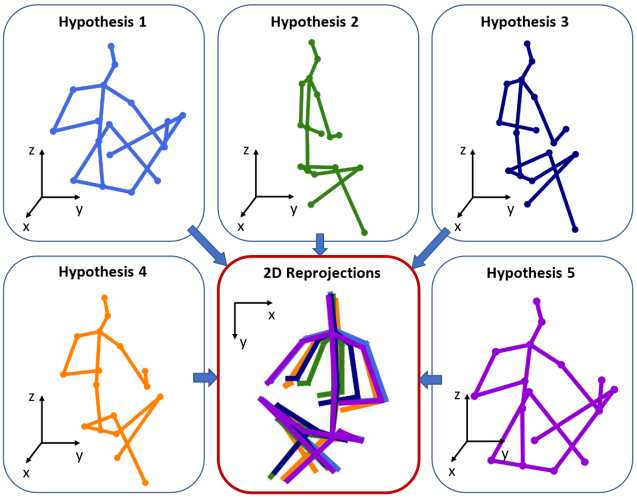


Figure 3.4: Example of multiple possibilities of 3D pose reconstructions from similar 2D joint locations [117].

Single-person pose estimation

Depending on the number of individuals to be computed, pose estimation methods are divided into single-person and multi-person. Single-person pose estimation is considered significantly easier than multi-person pose estimation, since it only involves the pose estimation of an individual in a given image, even if the image contains more than a single individual [118]. By contrast, multi-person pose estimation is particularly more complex since it must estimate the poses of all individuals in an image, which requires better image segmentation to separate overlapping individuals and correctly organize respective body parts [118].

With the accelerated use of deep CNN, performance of single-person pose estimation models improved, and even multi-person pose estimation has reached satisfactory levels [95].

Traditional 2D HPE methods adopt various feature extractor techniques to obtain global pose structures [1]. Early work introducing deep learning into pose estimation only complemented traditional HPE methods by replacing some components of their frameworks for neural networks [50], such as in the work presented by Ouyang *et. al* [119], where a multi-source deep model was used to collect nonlinear information about visual appearance, mixture type and deformation to aid a traditional pose estimation method.

Recently, CNN are widely used due to significantly improved results [1]. Single-person pipelines employing CNN are classified as one of two categories: keypoint regression-based methods or detection-based methods, like heatmaps [50].

- Regression-based methods

  A keypoint regression-based model directly regresses the keypoints from the feature maps. In a 2D HPE method, this can mean outputting a 17 by 2 vector with x and y pixel coordinates of each joint prediction, if the model representation is composed of 17 joints [99]. The representation is dependant on the output format usually associated with datasets, since the COCO output format contains 17 keypoints to represent different joints and other parts of the body, while the MPII format utilizes only 15 keypoints [86].

  The regression-based methods, also known as direct regression methods, are being constantly improved upon in order to increase detection of the human joints. Such is the case of Carreira *et al.*, who proposed a process called Iterative Error Feedback (IEF), to aid output prediction, utilizing top-down feedback which progressively changes the initial solution with the error predictions, achieving comparable results with other methods without needing ground truth annotations [120].

- Heatmap detection-based methods

Heatmap-based methods (or detection-based methods) estimate the location of body joints by adding 2D Gausian kernels on every joint location. The pixel intensity in each heatmap indicates the probability for the keypoint to lie in its position, as shown in Figure 3.5. Then, training focus on reducing difference between target heatmaps and predicted heatmaps [1].

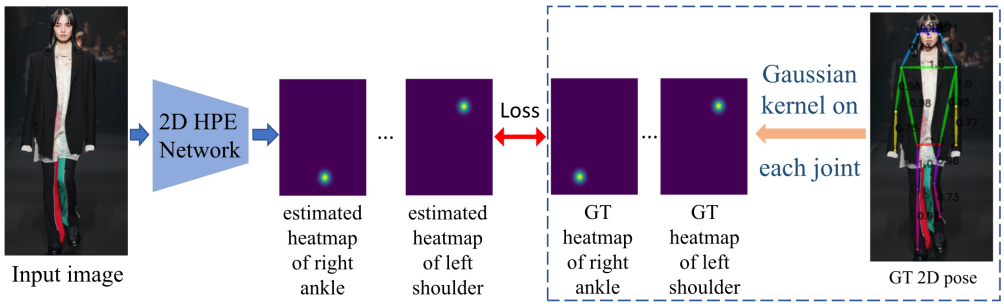


Figure 3.5: Heatmap based method. Given the ground-truth 2D pose (right), each joint has a heatmap generated by a Gaussian kernel. Then, prediction of each joint heatmap is done with a model (left), and loss is calculated between the two [1].

Multi-person pose estimation

Unlike single-person pose estimation, in a multi-person pose estimation method it is necessary to make associations between estimated joints and the individuals [95]. This can generally be done using one of the following methods, top-down and bottom-up, visible in Figure 3.6 [95]:

- Top-down approach, in which every person is first found, and then each pose is calculated separately.
- Bottom-up approach, where every human body part is first estimated and then associations to different persons are made.

This classification allows for multi-person pose estimation methods to be subdivided according to the starting point of the prediction. A significant difference between the two is the computational cost for pose estimation when the number of persons increases: while top-down methods increase in computational cost with the number of individuals, requirements for bottom-up methods remain stable [50, 86]. Bottom-up approaches, however, have difficulties when grouping body parts when there is significant overlapping between people and complex backgrounds [50, 86].

These strategies adopt two-stage solutions which achieve good results but can be inefficient due to the required two-stage process that can cause redundancy [121]. Nie *et al.* introduced a single-stage pose estimation model aimed at overcoming the constraints of the two-stage approach, which were partly due to the conventional

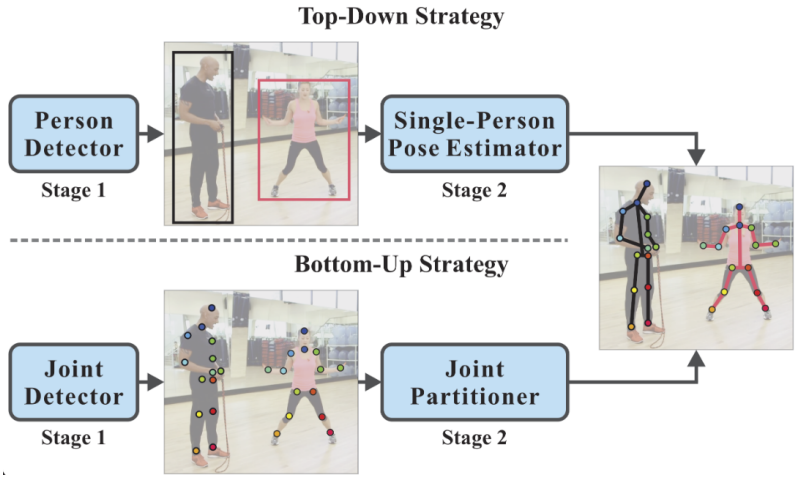


Figure 3.6: Comparison between top-down and bottom-up approach. Both consist of a two stage process in order to identify and estimate human poses in a multi-person scenario [121].

pose representation used [121]. The proposed model makes use of a Structured Pose Representation (SPR), using a unique central joint to identify the position of each person in an image and vectors pointing to joint displacements in relation to the central joint, representing the pose of a person together with its location. The method then employs a hierarchical Structured Pose Representation (SPR), capable of displaying body joints as well as relations between them more clearly, as seen in comparison in Figure 3.7 [121].

3.4 Backbones

The following neural network backbones are present in HPE, and are an important aspect of every method, since they deal with the underlying features that are used to identify joint locations in images and can compensate for occlusion in some cases by using temporal and spacial context [110].

3.4.1 AlexNet

The AlexNet (2012) is a deep CNN initially created for the ImageNet Large Scale Visual Recognition Challenge (LSVRC)-2010 contest and then used to participate in the ImageNet LSVRC-2012, where it achieved better results that other state-of-the-art [122]. Its use as backbone, while being a generic Deep Neural Network (DNN) architecture, is due to the outstanding results in image classification and location problems, negating the need to develop a domain-specific pose model, since the model and features originate from the training [123].

AlexNet's architecture, represented in Figure 3.8, is formed by 8 layers, namely 5 convolutional layers, and 3 fully connected layers. The novel features incorporated

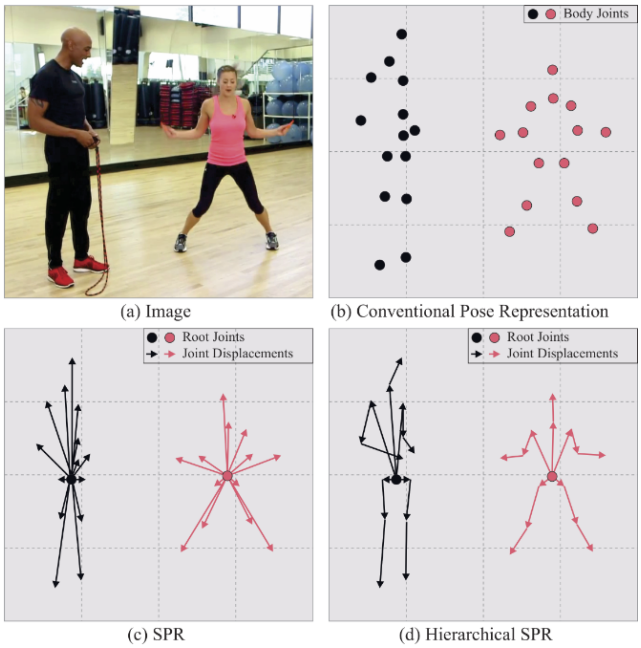


Figure 3.7: Different pose representations for multiple person in original image (a); (b) shows the conventional joint representation by absolute coordinates; (c) showcases the SPR with joints relative to a central person-identifying joint. Finally, (d) adopts an hierarchical SPR where joints are represented by vectors relative to the attached limb, starting from their person-identifying joint [121].

in its architecture at the time were the use of Rectified Linear Units (ReLUs) as activation function for neurons, in detriment of hyperbolic tangent activation functions. This made possible for the authors to use a large neural network, since ReLUs are much faster when training [122].

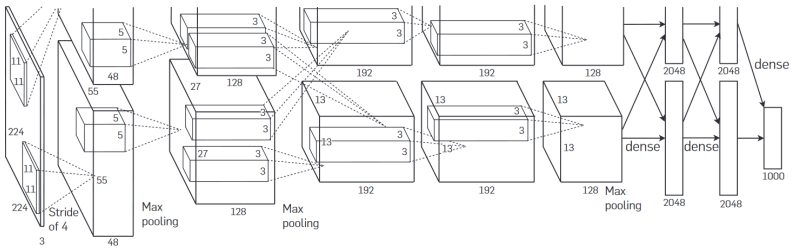


Figure 3.8: Architecture of AlexNet neural network backbone [122].

Other important considerations were the use of an efficient implementation of the convolution operation using Graphics Processing Unit (GPU), as well as the dropout technique that helps reducing overfitting in fully connected layers by randomly disconnecting a percentage of the connections between layers [122].

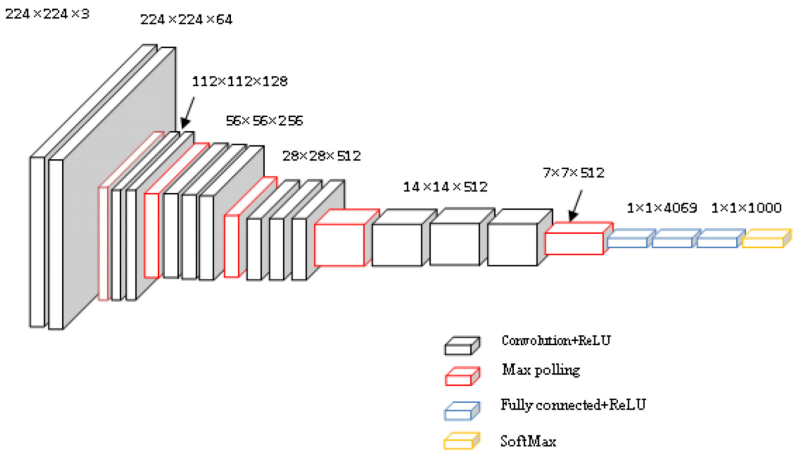


Figure 3.9: VGG neural network architecture [125].

3.4.2 VGG

The VGG network was released in 2015, with the purpose of finding the effect of depth in model accuracy in large scale image recognition exercises. It achieved a deeper network by using *3*x*3* convolution filters, which allowed to push depth to 16 and 19 weight layers and get first and second place in localisation and classification respectively in the ImageNet LSVRC-2014 [124].

This network utilizes a fixed-size *224*x*224* RGB image, doing minimal pre-processing, and then uses filters with small receptive fields of *3*x*3* after the convolution layers, as seen in Figure 3.9. Using three *3*x*3* convolutional layers, instead of a single *7*x*7* receptive field, a more distinction decision function and reduces the number of parameters [124].

3.4.3 ResNet

The ResNet architecture (2016) stands for Residual Network and it was designed for image recognition tasks, creating a viable way of expanding deep networks and winning the ImageNet LSVRC-2015 [126]. Deep CNN integrate low to high-level features, and can be further enhanced by increasing the number of layers (depth) of the network [126].

The advantage, compared to shallow architectures that expand by increasing the number of layers, is that degradation of training accuracy is reduced by using identity mapping. This is achieved with shortcut connections that feedforward their output by one or more layers, represented in Figure 3.10. This residual mapping, according to the author, allows for an easier optimization compared to the original unreferenced mapping optimization [126].

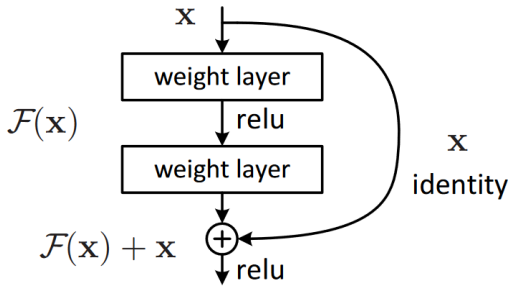


Figure 3.10: Residual learning block with identity feed-forwarding [126].

In terms of complexity, even though both the 101-layer and 152-layer ResNets are significantly deeper, they still achieve lower complexity than VGG-16/19 networks, requiring less GigaFLoating point Operations Per Second (GFLOPS). Moreover, comparisons between 50/101/152-layer ResNets showed that the degradation problem was nonexistent and accuracy rises further with the increased depth [126].

3.4.4 Faster R-CNN

Introduced in 2015, Faster R-CNN is a two stage detector composed of a Region Proposal Network (RPN), that proposes bounding boxes, and a second stage called Fast R-CNN [127] that extracts features from each box and performs classification and regression [128].

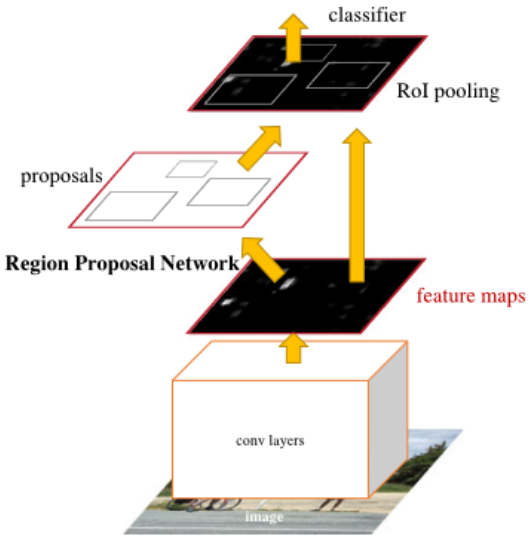


Figure 3.11: Faster R-CNN unified network composed of two stages [128].

The framework also uses the attention mechanisms that help finding global and local context in an image. Moreover, the RPN modeled with a fully convolutional network generates 2 for global and local features [128].

3.4.5 Mask R-CNN

The Mask R-CNN (2017) is a framework proposal for segmentation of different instances. It is based on Faster R-CNN [128], extending it by adding a parallel branch for object mask, at the cost of a small overhead, as seen in Figure 3.12. As a general framework, it is suitable for tasks such as human pose estimation or other tasks, since it outperforms every winner from COCO 2016 challenge [129].

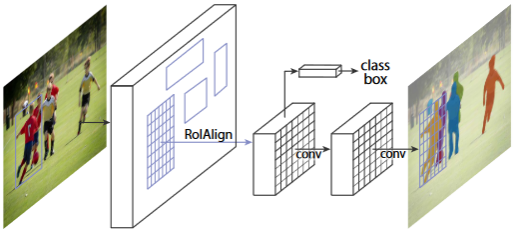


Figure 3.12: Mask R-CNN framework for parallel segmentation of instances [129].

The main distinctive features include pixel-to-pixel alignment and bilinear resampling when quantization is performed in the 7x7 feature maps. While fast, this network was not designed with performance as a priority, but, as a unified model that can predict boxes, segments and keypoints, it was tested running at 5 FPS. When using as a HPE framework, each mask can be used for calculation on each one of the K keypoints [129].

3.4.6 Long Short-Term Memory

The LSTM was first introduced in 1997 and it brought a novel and efficient method using gradients to minimise crescent error. When it was announced it solved complex long-time-lags tasks that couldn't be solved by algorithms at the time. This architecture achieves that by using an gradient-based algorithm that enforces constant error flow when circulating inside special circuits, composed of memory cells and gate units, visible in Figure 3.13 [130].

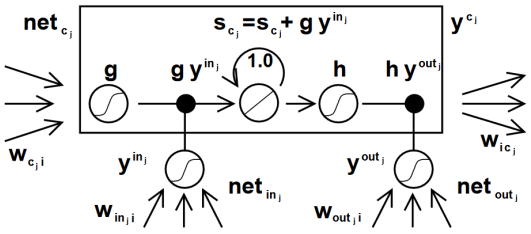


Figure 3.13: Architecture of a memory cell and gate units inside a block used in LSTM networks [130].

3.4.7 U-Net

The U-net architecture (2015) is applied in image segmentation, where precision and performance is required. This deep convolutional network has won two challenges at International Symposium on Biomedical Imaging (ISBI), outperforming the second best by a large margin at dental x-ray segmentation as well as ranking among the winners at segmentation and tracking on a set of different microscopic time series of living cells [131].

The network architecture, represented in Figure 3.14, consists of a contracting dimensional path, which is used to capture context, and an expanding path, symmetric to the previous, which can facilitate spatial accuracy and detail. The main difference compared to usual convolutional networks is the use of upsampling operators instead of pooling operating, increasing output resolution, as well as the connection of high resolution features from the expanding path to the upsampled output [132, 133].

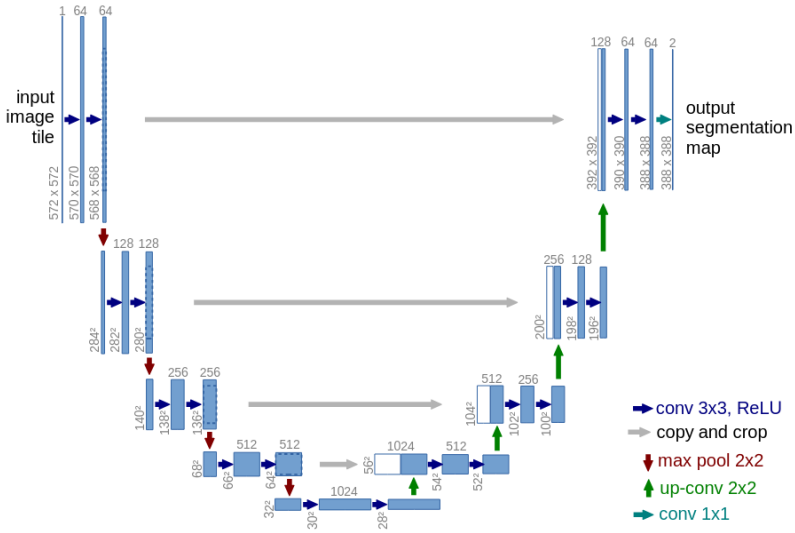
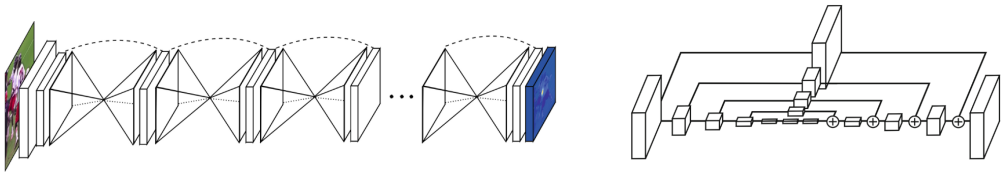


Figure 3.14: U-Net architecture composed of various convolutional operations at different resolution sizes, achieving the physical aspect of the letter 'U' [132].

3.4.8 Stacked Hourglass

In the stacked hourglass network, introduced in 2016, features are converted into very low resolution, by pooling, and then upsampled, to combine features from across various resolutions, like in many convolutional approaches [133, 134]. The difference in design lays in its symmetric topology, consisting of multiple hourglass modules put together, as seen in Figure 3.15a. The individual hourglass modules, represented in Figure 3.15b, are chained together to perform repeated bottom-up, top-down inference across different resolution scales. The performance of the network is due to

intermediate supervision and bidirectional inference, which allowed the architecture to achieve state-of-the-art results in two of the standard pose estimation benchmarks [134].



(a) Architecture of multiple stacked hourglass modules applied for HPE.

(b) Hourglass module architecture. Each box corresponds to a residual module.

Figure 3.15: Stacked hourglass architecture and single hourglass module architecture [134].

This architecture uses modules equipped with convolutional and max pooling layers to push features down low resolutions. After reaching the lowest resolution, a sequence of upsampling and feature combination across different scales are performed. The network also makes use of skip layers to preserve spatial information at every resolution [134].

3.4.9 Cascaded Pyramid Network

The Cascaded Pyramid Network (CPN) is composed of two subnetworks, GlobalNet and RefineNet, and was proposed in 2018 as an effective and efficient network for pose estimation. GlobalNet is based on ResNet, which can be used to locate keypoints such as eyes, but fails in locating hips, due to the lack of context around, and not only appearance. To find harder keypoints, RefineNet is applied, which receives the information across levels from GlobalNet, and concatenates the pyramid features [135]. The general architecture can be found in Figure 3.16.

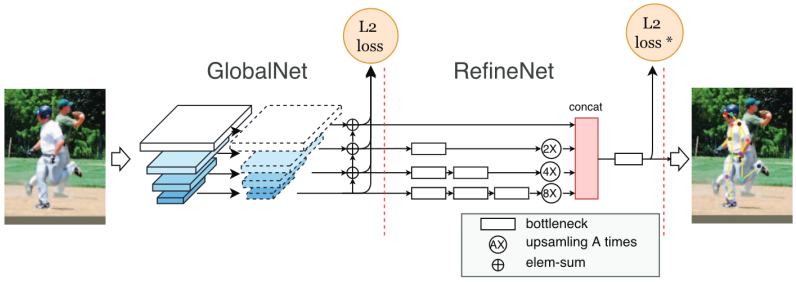


Figure 3.16: Cascaded pyramid network architecture [135].

3.4.10 HRNet

A backbone found in HPE methods is the High-Resolution Net (HRNet) from 2021, capable of maintaining high-resolution representations across the process. On

launch, it starts by adding high-to-low resolution convolution streams, and connecting the different resolution streams in parallel. The number of stages corresponds to the number of resolutions in the corresponding layer and, if using a 4 stage network like in the initial proposal, the forth stage will contain four different resolutions and streams, as represented in Figure 3.17 [136].

The parallel approach maintains high resolution instead of recovering it from low resolution, which makes it more precise spatially. Also, because of the parallel processing of different resolution streams, connections between high resolution and low resolution representations are stronger semantically [136].

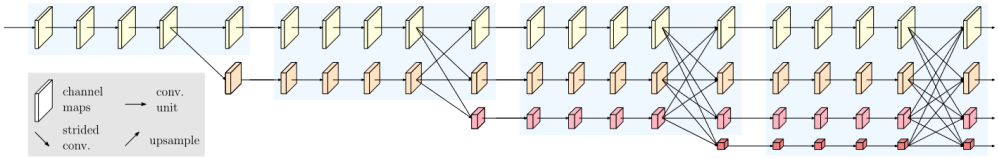


Figure 3.17: Architecture of a HRNet with four stages [136].

3.4.11 SelecSLS

The SelecSLS network was created to provide fast and accurate pose inference, as a first stage of a HPE method called XNect presented in 2020. This network was created with the ResNet-50 architecture as a baseline for performance, since a less computational intensive method was needed to reach real-time performance [137].

The SelecSLS uses both short and long range concatenation-skip connections, represented in Figure 3.18, which perform concatenation along the channel dimension, instead of additive-skip connections, that perform addiction only at the skip connection, as used in ResNet architectures. This allows for information flow along the network, without the memory and compute costs of the fully connected DenseNet, which achieves faster inference time compared to ResNet-50 while maintaining accuracy in both single and multi-person 2D and 3D HPE benchmarks. Moreover, the smaller memory usage allows for mini-batches twice as big of the ResNet-50 for training and batch inference [137].

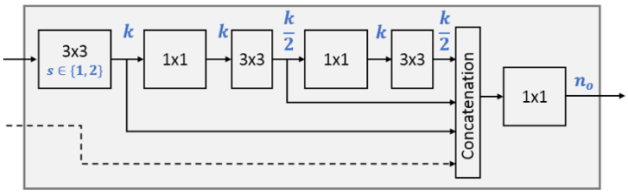


Figure 3.18: SelecSLS module design with concatenation-skip connections [137].

3.4.12 Key findings

Advances made in DNN backbones benefit HPE whether they were developed with it in mind or not [138]. Previous backbones mentioned were mostly used for detection and classification tasks, such as the ImageNet LSVRC, but can easily be used in benefit for HPE as most low-level features are in common and tested backbones are a good way of progressing by taking advantage of prior research [139]. Table 3.1 presents these network architectures that are found in HPE works.

Table 3.1: Backbones commonly found for feature extraction in HPE literature.

Year	Backbone	Layers	Features
2021	HRNet [136]	—	Parallel low to high resolution stages.
2020	SelecSLS [137]	—	Short and long range concatenation-skip connections.
2018	Cascaded Pyramid [135]	—	Proposed for HPE.
2017	Mask R-CNN [129]	Over 100	Parallel branch for object mask.
2016	ResNet [126]	Up to 110	Residual mapping.
2016	Stacked Hourglass [134]	—	Inference across resolutions.
2015	U-Net [132]	23	Contracting and expanding path
2015	VGG [124]	19	*3*x*3* filters.
2015	Faster R-CNN [128]	Over 100	High-quality region proposals.
2012	AlexNet [122]	8	ReLUs.
1997	LSTM [130]	—	Minimize vanishing gradient problem.

3.5 Common evaluation metrics and datasets

Given the diversity of tasks where human pose estimation is applied, multiple datasets and evaluation metrics exist for both 2D and 3D pose estimation [50]. These evaluation metrics are useful to precisely evaluate the performance of HPE methods, since there are many features and requirements considered for their implementation [1].

The existence of different datasets fulfills various necessities of developing approaches, since an universal dataset would be difficult due to the variation in scenarios [50]. By providing diverse data and annotated notations that differ in quality, these datasets offer researchers a way to evaluate and compare their proposed techniques but can also complicate a fair comparison of different algorithms due to shortcomings in datasets that do not align with the pretended scenario [1, 140].

3.5.1 Evaluation metrics

Metrics allow for performance of pose estimation methods to be assessed, granting valuable insight into what factors must be taken into consideration for further improvement [99].

- Percentage of correct parts

  A metric used in early works is the Percentage of Correct Parts (PCP), used for 2D and 3D HPE to evaluate the predicted stick accuracy for limbs [50, 141, 142]. The limb is considered correctly localized when the distance between predicted joints that compose the limb and ground truth limb are lower than a fraction of the limb length (usually between 10 and 50%), referred to as PCP@0.5 if using a 50% threshold [50].

  $$PCP = \frac{No.\ of\ correct\ parts\ for\ the\ entire\ dataset}{No.\ of\ total\ parts\ for\ the\ entire\ dataset}$$

  Recent works tend to not use this metric since it penalizes short limbs, which are harder to detect [104], and instead use a different criterion called PDJ [123].

- Percent of detected joints

  The PDJ metric was introduced in DeepPose [1, 123], as an alternative to PCP, with a fixed distance threshold for the detection criteria of every joint [123].

  $$PDJ = \frac{No.\ of\ correct\ joints\ for\ the\ entire\ dataset}{No.\ of\ total\ joints\ for\ the\ entire\ dataset}$$

- Percentage of correct keypoints

  A highly used metric in human pose estimation is the Percentage of Correct Keypoints (PCK) and its variant, the PCKh [141]. These metrics evaluate whether a predicted keypoint falls within a certain margin from the ground-truth keypoint. The margin is defined by a percentage of the bounding box for PCK, or by the length of the head segment in the case of PCKh. Specifically, PCKh@0.5 evaluates if the predicted keypoint falls within 50% of the head segment length h from the ground-truth keypoint.

  In this context, the total number of keypoints being evaluated is N, and for each keypoint, the Euclidean distance between the predicted and ground-truth keypoints is d_i. The threshold for acceptance is $\alpha \cdot h$, where α is typically 0.5. The indicator function ($d_i \leq \alpha \cdot h$) returns 1 if the predicted keypoint is within the acceptable distance and 0 otherwise. Thus, PCKh calculates the

average accuracy of keypoint predictions based on this threshold, providing a normalized measure of accuracy across different images and subjects [104].

$$\text{PCKh} = \frac{1}{N}\sum_{i=1}^{N}(d_i \leq \alpha \cdot h)$$

- Area Under the Curve

  Introduced for 3D pose estimation in VNect [100], the Area Under the Curve (AUC) is a metric related to the PCK, since it measures the range of PCK threshold. The threshold is usually stipulated as half the head size [50].

- Average precision

  The Average Precision (AP) and variants are widely used in the MPII [104] and PoseTrack [143] evaluation datasets, needing only joint locations for evaluation [50]. The AP metric was first called Average Precision of Keypoints (APK) [144] and results positive if the predicted joint falls within a threshold of a ground-truth joint location [50]. The mean Average Precision (mAP) is a variant calculated with the mean of the AP of each joint [145].

- Mean per joint position error

  In 3D HPE the Mean Per Joint Position Error (MPJPE) is commonly used for evaluating quality of joint predictions [50]. The metric is calculated using the Euclidean distance from the estimated joints to the ground truth joints, in millimeters, after being averaged over all joints and, in case of video, the metric MPJPE averages over all frames [50].

  Different datasets adopt different joint post-processing before computing the MPJPE [50]. The MPJPE in evaluation datasets such as HumanEva-I and Human3.6M is calculated after predicted and ground truth joints are aligned using the procrustes analysis [146], also known as reconstruction error [50, 140] or Procrustes Analysis Mean Per Joint Position Error (PA-MPJPE) [147].

- Frame rate, number of weights and giga floating-point operations per second

  While not entirely related with HPE, computational requirements are essential for any application, as they influence the devices capable of integrating them [50]. The FPS or seconds per image is a recurrent concern of real-time HPE methods, since this mean having to carefully select implementation methods early on [90].

  As an example, in a real time multi-person method, Cao *et al.* [90] considered that, while the initial bottom-up has the advantage of decoupled runtime complexity from the number of people in the image in comparison with the

top-down method, final global inference was costly [90]. This is also comparable with the case of the DeepPose [123] and DeeperPose [148], two known HPE methods. While the latter presented a significant run-time reduction while increasing accuracy performance, it still falls short of being capable for real-time (achieving 270 seconds per image) [148].

Another crucial aspect to consider is leveraging hardware-specific optimizations that can substantially decrease runtime, such as utilizing a device compatible with Compute Unified Device Architecture (CUDA) [90]. The presentation of number of weights and GFLOPS is found recurrently in result tables, and gives valuable information about efficiency of each HPE method [1, 91].

3.5.2 Datasets for human pose estimation

Be it for testing or model training, datasets establish a baseline when comparing with methods developed by other researchers and feature different characteristics [50]. Table 3.2 presents a overview of datasets created over the years with different purposes and Figure 3.19 showcases some of contents of these datasets.

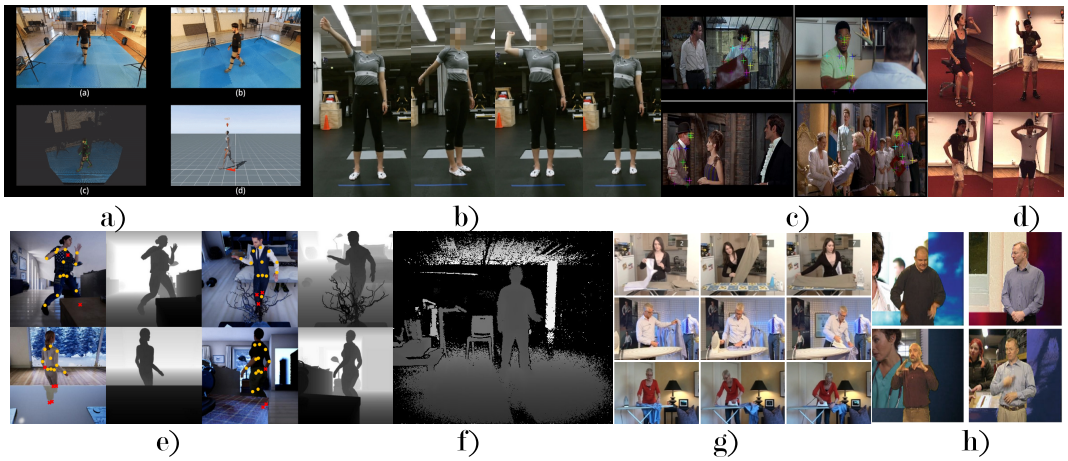


Figure 3.19: Samples illustrations of dataset contents: a) UNIPD-BPE [149], b) UI-PRMD [150], c) FLIC [151], d) Human3.6M [152], e) Occlusion-Person [51], f) IntelliRehabDS [153], g) MPII [154] and h) Extended BBC Pose [155].

Table 3.2: Popular datasets for 2D and 3D pose estimation.

Year	Dataset	Rehab	Anno-tations	Nº Joints	Single or Multiple	Video or Image based	Sensor	Characteristics
2022	UNIPD-BPE [149]	✗	3D	32	Both	Video	Kinect Azure and Xsens MVN Awinda inertial suits	Activities performed by 15 individuals. There are 12 sequences of single person sequences, 4 of them relevant as upper-limb activities.
2021	IntelliRehabDS [153]	✓	3D	25	Single	Video	Kinect v2	8 rehabilitation movements performed by 15 patients and 14 healthy individuals. 7 of the 9 exercises are upper-limb related.
2020	Occlusion-Person [51]	✗	2D and 3D	15	Single	Image	Synthetic	Synthetic dataset composed of 13 different human models with variations in pose, clothing and spaces. Occluded joints are labeled together with the remaining joints.
2018	UI-PRMD [150]	✓	3D	22	Single	Video	Kinect and Vicon optical tracker	10 rehabilitation movements performed by 10 healthy individuals. 4 of the exercises are upper-limb related.
2017	Pandora [156]	✗	2D and 3D	17	Single	Video	Kinect	Total of 110 driving like sequences performed by 22 individuals.
2015	Extended BBC Pose [155]	✗	2D	7	Single	Video	RGB cameras	92 videos from BBC containing 9 different overlaid sign language interpreter. Manual and semi-automatic annotations of upper body joint locations.
2014	Human3.6M [152]	✗	3D	32	Single	Video	Motion capture system (name not disclosed) and RGB cameras	11 actor data across different scenarios (smoking, talking to the phone) totaling 3.6 million images and pose data files.
2014	MPII [154]	✗	2D	15	Mixed	Image	RGB cameras	Human activities from Youtube performed by over 40k people with annotated body joints. The dataset is intended for evaluation and comparison between different methods.
2013	FLIC [151]	✗	2D	10	Single	Image	RGB cameras	Around 20k images from popular films. The images possess upper body joint annotations and individuals are mostly frontal and not occluded.

The presented datasets show a mixture of video and image data that represent the range of data available for HPE tasks. While some datasets possess 2D joint annotations over entertainment shows (FLIC [151] and Extended BBC Pose [155]), others were intentionally created for the task of HPE in physical rehabilitation, such as UI-PRMD [150] and IntelliRehabDS [153], or scenarios that share a lot of similarities, such as the driving like dataset Pandora [156]. For RGB-D datasets, some authors have extensive research material that contains various scenarios, such as Lopes *et al.*[157].

3.6 Deep learning based approaches

There are different implementations arising from research with effective pose estimation networks. It will be given focus to the understanding some of these methods, diving into architecture backbones adopted, which datasets were used and what metrics where applied for testing their performance.

In 2D single-person HPE, both regression-based and heatmap-based approaches exist, as referred previously in Subsection 3.3.2. Each of them has inherent limitations: while regression based methods can perform non-linear mapping from images to keypoints, the solutions offered are sub-optimal, due to the non-linear problem [1].

Heatmap-based methods are adopted more often for 2D HPE, and outperform most regression-based approaches [1]. The adoption is due to the probabilistic nature of each keypoint prediction and better supervision information that preserves spacial information [1]. Some aspects to consider, however, are the larger memory footprint and increased computational cost when using high-resolution heatmaps [1, 91].

3.6.1 DeepPose

DeepPose (2014) is considered the first CNN based HPE model and it uses a 2D skeleton body model to represent the estimated poses of a single person in images. This method uses the AlexNet [122] as backbone, previously explained in Subsection 3.4, and uses a series of cascades of regressors that outputs the coordinates (x,y) of each joint [86]. The architecture can be seen in Figure 3.20.

DeepPose shifted the research from classic approaches to deep learning by using a cascaded DNN regressor [1]. In DeepPose, each joint is regressed using as input the full image and a generic deep CNN with 7 layers. According to the authors, this allows the network to capture the full context of each body joint, since every regressor uses the full image as signal. Moreover, the approach is simpler than methods based on graphical models, not needing explicit joint interaction or model topology design [123].

Figure 3.20: Architecture of DeepPose method [123].

DeepPose is evaluated in FLIC and LSP datasets using the PCK and PCP metrics, and it outperformed other works in most cases [86, 123].

3.6.2 XNect

XNect was presented in 2020 as the first real-time multi-person 3D motion capture implementation that works with a single RGB camera. It uses a skeleton-based body model, and its system was designed from the ground up to enable real-time performance, which included the pose representations, network architectures and pose fitting model. The CNN SelecSLS, previously presented in Subsection 3.4.11, allows real-time performance while keeping accuracy when comparing to the ResNet-50 backbone used as a baseline [137].

The method is composed of three stages. The SelecSLS network is the first stage, being used for 2D pose prediction and 3D pose encoding. It predicts joints as 2D heatpoints and then uses Part Affinity Fields (PAF) to correlate joints with the belonging person, performing part association [137].

Second stage uses a fully connected network with 5 layers to predict 3D positions of each individual from the 2D joint positions relative to the neck and can use full body context to fill occluded joints. It was trained from the MuCo-3DHP dataset [158] after passing the first stage, extracting 2D pose and 3D pose encodings [137].

The last stage is used for reconstruction and tracking of poses and people across frames, even after occlusion, done through a color histogram of the upper body section. The stage also computes height and finally fuses 2D and 3D predictions with temporal smoothing and joint constraints to finally achieve the space-time kinematic pose method.

It can achieve above 30 FPS in real-time, in which the first stage is the biggest contributor for the total time. The quality of the estimated pose is comparable with the depth sensing method from the Kinect v2 SDK, with XNect showing superior performance in cluttered scenarios [100].

3.6.3 AlphaPose

AlphaPose (2017) is described as the first open-source system capable of achieving 70+ mAP on the COCO dataset. It was designed for estimation of the entire body of multiple people using a top-down approach, and is constituted by a multistage concurrent process that allows it to be ran in real-time. The network used for the first detection is the YOLOV3-SPP [159], which can achieve good results while keeping high efficiency, and after detection and bounding-box of every person in the image, a single-person HPE method is performed. This method uses a new gradient design and keypoint regression method to provide the body poses [160]. The entire multistage process can be seen in Figure 3.21.

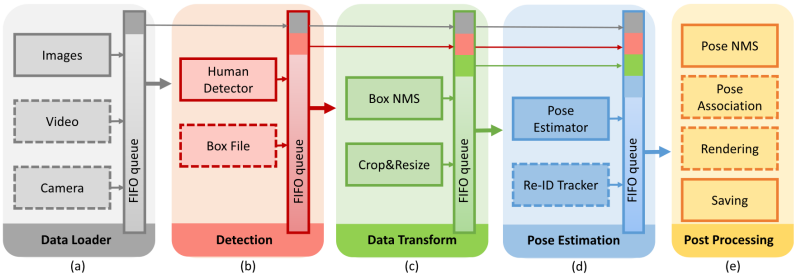


Figure 3.21: AlphaPose system architecture [160].

This implementation was trained on the Halpe-FullBody and COCO datasets, and evaluated using OKS and AP and mAP for accuracy between frames. For tracking performance, the model used the Multiple Object Tracker Accuracy (MOTA) and Part Affinity Fields (MOTP) on the posetrack validation dataset [160].

3.6.4 OpenPose

OpenPose (2017) is a real-time multi-person 2D HPE method using PAF. This was the first system with these characteristics to detect body, foot, hand and facial keypoints, making a total of 135 keypoints via a single image. The method possesses an efficient pipeline which makes use of a bottom-up approach, but simultaneously performs detection and association over every location with sufficient context to achieve good results without being so affected by the runtime complexity [90].

The method is tested using the MPII and COCO validation set, where it achieves nearly top results in the MPII dataset but only comparable results with other methods in COCO. The MPII used the mAP metric and in COCO it was used both the mAP and the PCKh.

3.6.5 IEF

The IEF was first used in 2016 for HPE with distinguished performance, due to use of hierarchical feature extractors. It was used in 2D human pose estimation, locating

and showcasing the 17 keypoints in the skeleton-based model [120]. The input from a single RGB camera is processed by a network pre-trained in ImageNet, the googlenet, which is comparable to the VGG-16 backbone, but requires less memory [161].

This method was tested in the MPII benchmark dataset, using the PCKh metric, which surpassed the previous best result by a significant margin, but achieving less efficiency [120].

3.6.6 HRNet

The HRNet uses high-resolution representations to perform HPE. The following method from Sun *et al.*, is named deep-high-resolution-net (2019) and it achieved superior results in 2D pose estimation in multi-person scenarios [91].

This method uses the skeleton model with 17 keypoints to represent individual poses, as it was trained using the COCO dataset. It then used the different variations of the AP metric for comparison with other implementations for HPE, using the COCO validation test, where it achieves, with the HRNet-W48 backbone, the best results, while being one of the heaviest models in terms of GFLOPS and number of parameters [91].

3.6.7 Key findings

Neural networks (DNN) have addressed various problems in pose estimation tasks. These architectures excel in modeling sequential data, enabling them to predict human poses over time with enhanced precision or dealing with occlusion gracefully and with better robustness [110]. Table 3.3 the previous mentioned methods that showed advancements on HPE were analysed and compared based on them characteristics.

While there is a continued effort in development, since much work is still being done specially in 3D multi-person HPE [110], there are methods that specialize in real-time operation (XNect [137], AlphaPose [160] and OpenPose [90]) or temporal coherency (XNect [137]), while others focus their effort in improving their accuracy and dealing with object and person occlusion (IEF [120] and pose_hrnet_w32 [136]).

Table 3.3: Deep-learning based methods used for HPE. R stands for regression, H for heatmap, BU for bottom-up and TD for top-down. All models use a kinematic body model for representations.

Year	Method	Real-Time	Temporal info.	Approach	Backbone	Input resolution (pixels)	Input type	Output	Characteristics	Eval. metrics
2020	XNect [137]	✓	✓	BU, H	SelecSLS	512×320	RGB	3D	Reconstruction of occluded joints.	3D Percentage of Correct Keypoints (3DPCK), AUC, MPJPE
2019	pose_hrnet_w32 [91]	✗	✗	TD, H	HRNet-W32	384×288	RGB	2D	Accurate high resolution heatmaps.	AP, GFLOPS
2017	AlphaPose [160]	✓	✗	TD, R	ResNet and FastPose	256×192	RGB	2D	Real-time pose estimation and tracking.	mAP, PCKh
2017	OpenPose [90]	✓	✗	BU, H	VGG-19	368x368	RGB	2D	Included in OpenCV library.	mAP, PCKh
2016	IEF [120]	✗	✗	R	GoogleNet	224×224	RGB	2D	Keypoints outside images can be predicted.	PCKh, PCP
2014	DeepPose [123]	✗	✗	R	AlexNet	220x220	RGB	2D	First CNN based HPE method.	PCK, PCP

3.7 Summary

Over the course of Chapter 3, topics related to HPE were discussed in order to build a solid foundation of knowledge about the development of this area. The sensors commonly used for HPE, with a special focus on previous works that address stroke rehabilitation and other rehabilitation work, are sEMG, IMUs and both RGB and RGB-D camera systems. While sEMG benefit from the valuable muscle information [53], they are found intrusive, suffer from external perturbations and need excellent signal processing [58, 59]. Systems utilizing IMUs capture acceleration, tilt, vibration and rotation [60] and are not affected by occlusion [62], but require time to attach and cause discomfort [6]. Systems utilizing markers with vision-based approaches can also cause discomfort and time to attach [7].

In the vision-based approaches to the design of a method, such as which body model to use (skeleton, contour or volume-based), which have their vantages and disadvantages. 3D HPE usually use skeleton-based models, for its simplicity and flexibility [50, 99], but the volume-based models like the SMPL [109] are also considered for the accuracy and realistic shape [50]. A body model can be represented with a varied number of keypoints (representing major body joints, face features, hands) but the number is related to the output format of datasets like COCO (containing 17 keypoints) and MPII (15 keypoints) [86].

Aside from representation used, methods are divided in 2D and 3D, with the main distinction being the third coordinate associated with depth, which allows 3D HPE to be used for more applications [50, 110]. Methods can also be implemented for single-person or multi-person pose estimation depending on the desired number of people to be processed. Single-person HPE is significantly easier, since it does not require the level of image segmentation or processing of a multi-person method [118]. Even so, with the recent advancements of CNN, even multi-person pose estimation has reached good levels of performance [95]. Single-person HPE is further divided in regression-based and detection-based methods. While the former directly regresses the joints, the latter builds heatmaps of probability for every joint [1, 86].

Video systems based on RGB cameras without markers are inexpensive and non-intrusive, but occlusion and depth ambiguities compromise the quality of detection of body joints, which is why implementations merging IMUs and cameras exist [2]. This combination, however, requires a significant number of IMUs to be accurate enough [52]. Depth cameras have the advantage of depth capture, giving them versatility [67] over multi-camera systems, which require calibration [83]. They also alleviate computational requirements by not having to estimate depth like monocular RGB cameras, which is the focus of many researchers [1, 79] but the former are more expensive and many have limited rates of depth capture that might cause blurriness [72]. RGB-D systems are often utilized in research environments, with the Microsoft

Kinect family being the most recognized for this [68, 77, 78, 81].

The Kinect possesses an SDK with HPE method capable of inferring 3D positions from joints in real-time, but are often inaccurate [73] and has difficulties when individuals are close to the background [68]. Many HPE are not designed for inference speed, and instead focus on precision or new approaches [120, 123, 136, 160], with recent methods being capable of real-time operation while determining joints for multiple individuals in both 2D [90] and 3D [137].

These methods are often built using foundations, called backbones, which are trained networks for feature extraction in visual tasks [110]. Backbones possess different structures, but usually increase the number of layers of the network, becoming DNN since it allows them to capture features of different scales [122, 125, 126, 134]. For that, they employ different techniques that help optimizing the architecture for viable inference and training speed [122, 125, 128]. Some of the backbones are more specific and are developed with HPE in consideration [135, 137] or simply feature memory cells that help storing temporal features [130]

Methods are evaluated and trained using datasets [50]. 2D datasets [151, 154] might use metrics such as PCP, PDJ and PCK, and in datasets employing 3D the metrics like AUC, 3DPCK, MPJPE and AP. A common method of evaluation for 2D and 3D is based on computational requirements [50]. These datasets are created in diverse ways, since some are based on third-party images and videos [151, 155, 154] or synthetic data [51], with some specifically created for rehabilitation of upper and lower limbs [149, 150, 153] with joint data captured by visual or inertial systems.

Chapter 4

Proposed system architecture

This chapter details the development choices for the HPE method used in upper-limb rehabilitation. The presented architecture aligns with the objectives initially proposed for this work and is built on the solid foundations established by the research discussed in earlier Chapters.

4.1 Overview of system requirements

The system must meet several critical requirements to achieve the established goals. First and foremost, it should operate effectively with a single RGB-D camera. This camera will provide both RGB image sequences and depth maps, which are essential for accurately positioning 3D keypoints. The integration of RGB and depth data allows for a more comprehensive analysis of the upper limbs' spatial configuration.

Additionally, the system is designed for real-time operation. This means it must process data quickly enough to provide immediate feedback, which is crucial for immersive rehabilitation exercises. Ensuring fluid keypoint tracking is a significant aspect of this requirement. The system must maintain consistent and smooth tracking of keypoints, even during rapid or complex movements.

Handling occlusion is another critical requirement. Occlusions can occur when the individual being tracked is blocked by an object or experiences self-occlusion (when one part of the body blocks another). The system must be robust enough to account for these scenarios, ensuring that the tracking remains accurate and reliable.

Moreover, the method will output skeleton-based keypoints for the upper-limb joints. This output is vital for analyzing the movement and positioning of the limbs. Finally, the system should be designed for easy integration into a broader rehabilitation project. This includes compatibility with existing software and hardware, user-friendly interfaces for therapists and patients, and the ability to adapt to various rehabilitation protocols.

By fulfilling these requirements, the system will effectively support upper-limb rehabilitation, providing accurate and real-time tracking of limb movements and contributing to the overall effectiveness of rehabilitation programs.

4.1.1 Key considerations for the scenario

A consideration to have is the room setup for the system and how it might be used by a variety of individuals. Pose detection on a wide variety of individuals and clothing without losing consistency should be reviewed when choosing a scenario, as well as the training dataset. One key aspect is the positioning of the camera responsible for the capture of input data for the HPE method. The captured image should contain the entire upper body of a person. Cases such as when a person is leaving the scenario or their body is blocked by other person or object are exceptions, but, during the procedure of a rehabilitation exercise, the upper body should be visible, as it is shown in Figure 4.1.

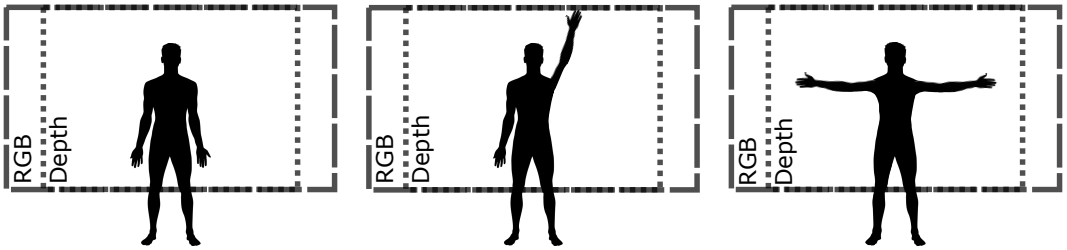


Figure 4.1: Occupancy of a person's upper body centered and fully captured by the camera system.

A body has a wingspan (arms extended horizontally at shoulder height) of about the same size as the person's height, which means that the camera should have sufficient periphery vision to cover about 2 meters of width if the person was facing the camera at the center of the frame [162]. For rehabilitation, the focus is set mostly on the affected limb side, which should be kept inside the limits of the camera, and, by this logic, can be considered a fraction of the space needed for the wingspan of a person. Figure 4.2 shows a diagram illustrating the advantage of prioritizing the capturing of the affected limb by putting it closer to the center of the frame, and, if necessary, decrease the distance to the camera system.

Another change that could be useful is a rotation of about 15° of the orientation of the individual in relation to the camera system, so that the affected arm depth

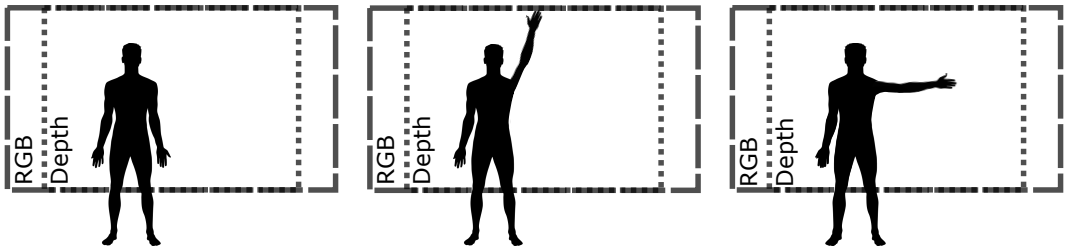


Figure 4.2: Occupancy of a person's affected limb centered in relation to the camera system.

is easier to calculate (by increasing the range between minimum and maximum distance of the body to the camera), which provides better clarity on the movement tracking and reduces self inflicted occlusion, as shown in Figure 4.3. However, the modification might put extremities of the upper body, such as the hand, out-of-view of the camera system, which can be fixed by the patient standing further away from the camera system, where it covers less of the captured area and decreases body resolution, which might not be ideal.

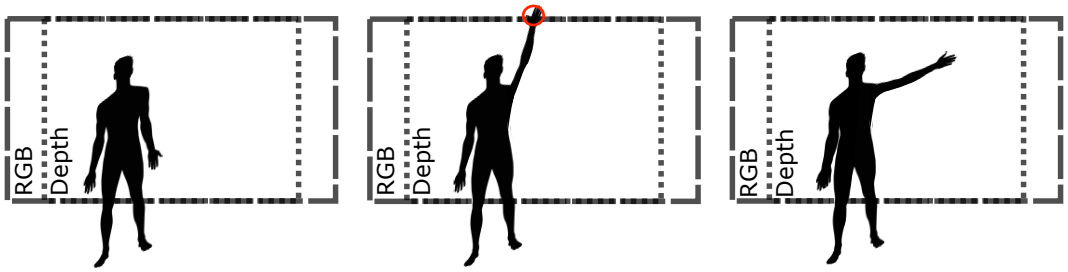
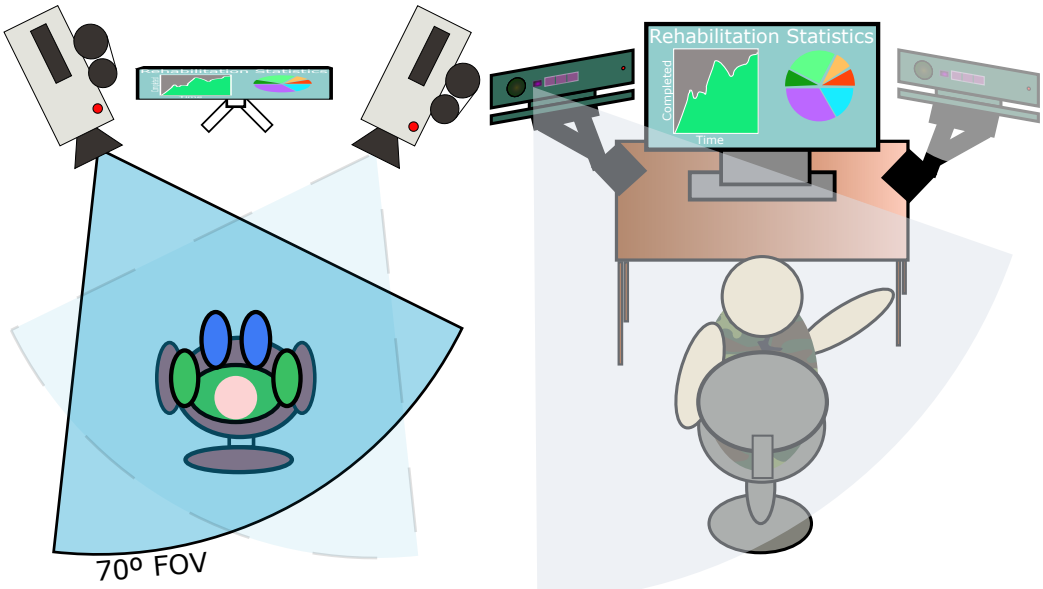


Figure 4.3: Occupancy of a person's affected limb centered and body rotated by 15° in relation to the camera system. The central pose with the arm up makes the hand go out-of-view.

A body rotation also involves a more complex scenario to keep the process compatible with different impairments, such as left and right side paralysis, which affects the limb being tracked. The impaired limb should be kept at the center of the capture area and also be the further body part from the camera system, while the person is facing the screen. This allows the person to fully extend the limb without self occlusion occurring.

Figure 4.4 allows a clearer understanding of the scenario and supports the development of a HPE that comprehends the intended working space. Subfigure 4.4a demonstrates what the scenario might look like when seen from above. In this, the patient is sitting in a chair, facing a screen that is guiding the rehabilitation exercises and two cameras, at head level, which are chosen based on rehabilitation needs. For compatibility with different person heights, a table with height adjustment is required, since it holds both cameras and the screen, making it possible for standardized testing, as seen in the back view from Subfigure 4.4b.



(a) (b)

Figure 4.4: Views of the rehabilitation scenario: (a) top view, and (b) back view.

4.2 Key components

The requirements of the system ask for a composition of highly specialised and capable modules that perform each function with accuracy and efficiency. Most processes work as complementary work for the main 2D HPE module, which needs an adequate feature collection, RGB joint smoothing, and posterior depth alignment for output.

The architecture diagram can be seen in Figure 4.5 and contains the various modules necessary for this application. Starting from the top, the input in this system is composed of the RGB-D video feed from a camera system in real-time, which is then passed to a preprocessing module that prepares the individual frames for one of the two HPE methods. On the left, the heatmap detection module utilizes an U-Net to output multiple heatmaps containing the keypoint predictions which are later converted to keypoints.

The right-sided method is the direct regression module using a ResNet-based architecture, which directly maps input images to keypoint coordinates and it can be used in alternative to the heatmap detection. Then, temporal smoothing is applied on the predictions, making sure that the values correspond with what would be expected based on past data samples, which, together with the correct camera configuration, allows the use of the definitive 2D keypoints to be used in the RGB-D registration to find the corresponding 3D coordinates, as the output of the system.

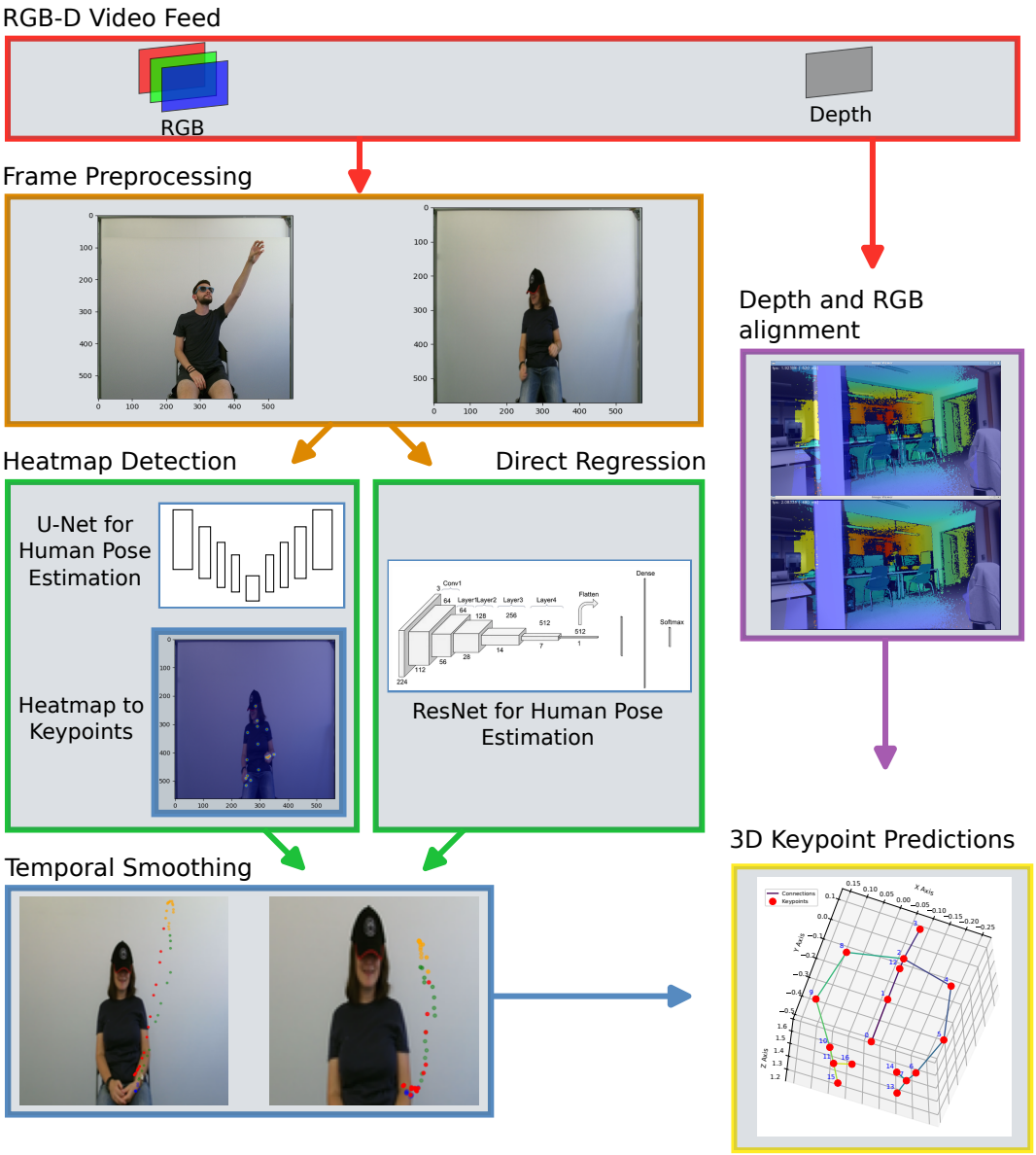


Figure 4.5: Proposed architecture diagram.

4.2.1 Selected RGB-D input for the system

A RGB-D input camera system allows access to the three colour channels that compose most digital images (RGB) as well as a depth channel that can also be found in image format in grey-scale, with proximity objects having a lighter shade of grey than those further away.

The depth sensor has advantages, especially for 3D HPE, being used in the Microsoft Kinect SDK to obtain 3D positions from joints, without the need for the auxiliary RGB channels [74]. While the RGB feed has the advantage of providing information on texture and helps distinguishing objects grouped close to each other, using it together with the depth channel might increase complexity of the task,

since it requires the three RGB input channels in a neural network as well as depth registration work in order to map RGB pixels and depth.

The Microsoft Kinect v2 is chosen as the input source of this system architecture, since it can be used to access RGB-D data at capture rates compatible with a real-time application and is often used for these applications [80]. The technical details can be consulted in Table 4.1.

Table 4.1: Technical specifications of the Microsoft Kinect v2 RGB-D camera system [69, 70, 163].

Microsoft Kinect v2		
RGB	**Resolution (pixels)**	1920x1080
	Frame rate (frames per second)	30 frames per second
Depth	**Resolution (pixels)**	512x424
	Frame rate (frames per second)	30 frames per second
Field of View	**Horizontal (degrees)**	70
	Vertical (degrees)	60
Measuring range	0.5 m to 4.5 m	

The advantage of using the Microsoft Kinect v2 is the high availability and modest cost of the hardware. The official SDK[1] contains the drivers for the Windows operating system and included examples with the Application Programming Interfaces (APIs). For Linux systems, an alternative is provided by OpenKinect[2], which includes libfreenect2[3], an open-source driver developed for the Microsoft Kinect v2. This driver offers access to the Kinect for Windows devices in Linux, enabling the retrieval of RGB and depth images, as well as auxiliary functions.

4.2.2 Chosen architecture for the HPE model

For the feature extractor, a backbone commonly used in computer vision tasks, such as those trained on the ImageNet LSVRC, is preferred. This is because these models are known to generalize well to various tasks due to their ability to capture essential visual features. Given this, ResNet was chosen as the backbone for the direct regression method, being used in other HPE methods with good results [1, 164]. ResNet offers different configurations based on the number of layers, which determines the complexity of the network, as shown in Table 4.2. These ResNet models are easy to implement and readily available in PyTorch libraries [165].

[1] `https://www.microsoft.com/en-us/download/details.aspx?id=44561` (last accessed in 21/09/2024).

[2] `https://openkinect.org/wiki/Main_Page` (last accessed in 21/09/2024).

[3] `https://github.com/OpenKinect/libfreenect2` (last accessed in 21/09/2024).

Table 4.2: Comparison of different ResNet architectures. The inference times are measured from a system running Ubuntu 22.04 and a Nvidia RTX 4060 Mobile GPU with 8 GB of Video Random Access Memory (VRAM).

Model	Parameters (M)	Inference Time (ms)	Time Relative to ResNet18 (%)
ResNet18	11.69	90.055	100.00
ResNet34	21.80	155.034	172.15
ResNet50	25.56	209.402	232.53
ResNet101	44.55	328.533	364.81
ResNet152	60.19	476.084	528.66

The ResNet is short for Residual Network and applies short skip connections so that features from early layers reduce the degradation of training accuracy, as introduced in Section 3.4 [126]. While this allows for better accuracy when training, a deeper network will also be more computationally expensive, which might be a limiting factor for real time performance.

The architecture is also composed with a custom classifier, identical as what is seen in Figure 4.6, since it receives the processed features from the backbone and outputs a set of joints based on current input. The final layers of the network must be trained from scratch, in order to get the desired number of keypoints and dimension, in accordance to the training dataset. Taking advantage of a pretrained network allows the possibility for the use of low-level features to be extracted without requiring much data or processing.

Alternatively, the U-Net architecture can be adapted for 2D pose estimation. Originally designed for image segmentation, U-Net can be trained to map keypoints to corresponding heatmaps [132]. By leveraging both ResNet and U-Net, it is possible to test two different backbones for single-person HPE: regression-based and heatmap detection approaches, introduced in Subsubsection 3.3.2.

4.2.3 Choice for temporal smoothing network

According to Luo *et al.*, image-based methods produce sub-optimal results in video sequences. This can be result of their inability to capture temporal dependency and keep object consistency, suffering from motion blur and occlusion, which generate different results for a sequence of frames [167]. Temporal smoothing is found beneficial in HPE, as some studies incorporate different temporal cues directly into pose estimation, such as using color and motion features during training of a convolutional network [168].

The temporal smoothing will be achieved using a LSTM network [130], which excels at capturing short-term temporal dependencies and producing recurrent outputs, but does not benefit from longer sequence lengths [169]. This limitation is

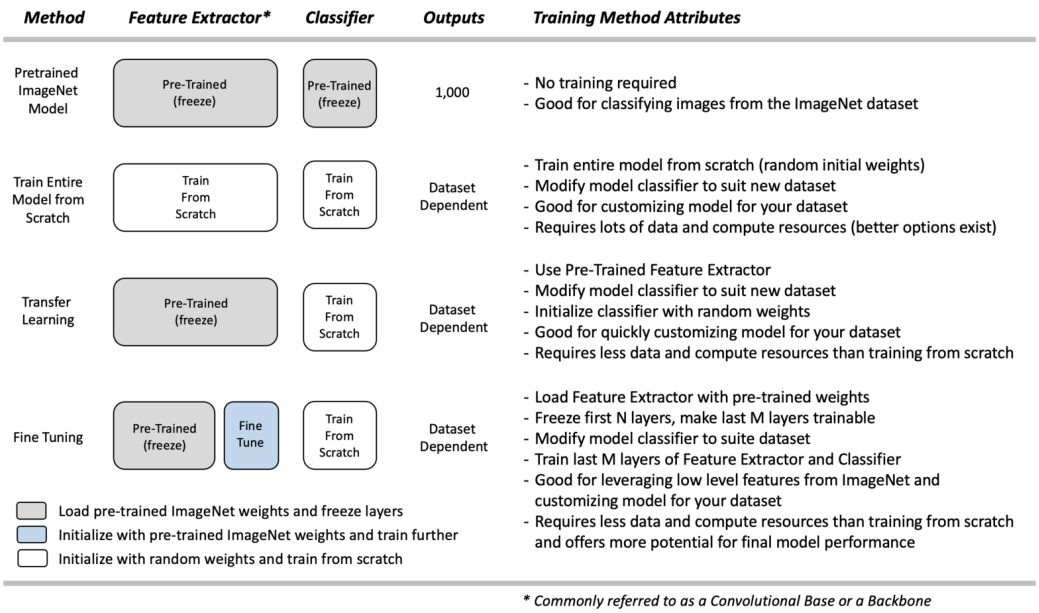


Figure 4.6: Training methods using ImageNet. The 2D pose estimator of the proposed architecture would be considered the classifier of the network, which defines the output number, while the backbone is the feature extractor [166].

not expected to be an issue in this case, as the model will focus on short temporal windows. The LSTM receives previous keypoints samples and predicts the next keypoint locations. A function then evaluates whether this prediction aligns with the current estimation from the HPE module. If the predicted keypoint significantly deviates from the current estimate or is missing, the model preserves the previous position and smooths the current one based on motion.

4.2.4 3D joint annotations computation

The output of the system will provide the 3D joint annotations of the captured RGB-D image, which can be overlapped with the original image to provide visual feedback. While this can be performed, the image from the RGB camera feed and depth sensor are misaligned and image registration is needed for alignment, which requires the camera calibration parameters to be well suited for the particular device used. Then, by providing the 2D joint estimations, the camera software can find the correlations between dimensions and return the 3D joints for each point.

4.3 Technologies and frameworks harnessed for development

The development of this method utilizes various tools and software. During implementation, tools such as Jupyter notebooks, Windows Subsystem for Linux (WSL)

and PyTorch are indispensable. For the deployment and standard use of this HPE method, to achieve a fully containerized for portability system, Docker is seen as a viable option, since it assures reproducible results in different systems. Other tools, present in the whole development, include the libfreenect2[4] driver capable of communicating with the RGB-D camera system providing the video feed and the ffmpeg software[5] which allows visualization of the feed via standard Linux devices. The playback of the inferred upper-limb pose overlapped with the initial input data is performed with OpenCV[6] and the software used to calibrate the RGB-D camera is named IAI Kinect2 [170], which also takes advantage of a Robot Operating System (ROS) environment[7].

4.3.1 JupyterLab

JupyterLab[8] provides an interface to interact with dynamic Python code cells embedded together with Markdown for document formatting. Its advantages lie in providing a modular and interative interface which allows for easier reading and even supports exporting to different document formats like *.pdf* or *.html* for documentation or reports. Online platforms such as Google Collaboratory[9] and Kaggle[10] adopt Jupyter-based notebooks for development and testing of Python scripts and provide access to cloud resources like GPUs and other accelerator units for more complex tasks.

4.3.2 PyTorch

PyTorch[11] is a popular tensor library for deep learning compatible with the Python and C++ programming languages that can take advantage of GPU and CPU. A Tensor can be thought of as an array or matrix, depending on dimension, that can be directly executed in a GPU for accelerated computing. This allows for fast development of a HPE method utilizing popular pre-trained models, loss functions and optimization techniques.

4.3.3 Windows Subsystem for Linux

WSL[12] is a tool that allows running a Linux environment on a Windows computer, without needing a virtual machine or dual boot. It allows access to the Linux file

[4] `https://github.com/OpenKinect/libfreenect2` (last accessed in 21/09/2024).
[5] `https://ffmpeg.org/` (last accessed in 21/09/2024).
[6] `https://opencv.org/` (last accessed in 21/09/2024).
[7] `https://www.ros.org/` (last accessed in 21/09/2024).
[8] `https://jupyter.org/` (last accessed in 21/09/2024).
[9] `https://colab.research.google.com/` (last accessed in 21/09/2024).
[10] `https://www.kaggle.com/` (last accessed in 21/09/2024).
[11] `https://pytorch.org/` (last accessed in 21/09/2024).
[12] `https://learn.microsoft.com/en-us/windows/wsl/` (last accessed in 21/09/2024).

system from the Windows host system and vice-versa, which facilitates interaction between systems. After installation, the user can choose to install a Linux distribution such as Ubuntu and access a Linux Command Line Interface (CLI) with access to GPU and other devices making it possible to install a program like JupyterLab. Then, the program can launch an instance that can be connected by accessing the local address on a browser.

4.4 Summary

Chapter 4 outlined the architecture for the HPE method built for upper-limb rehabilitation. The system is designed to operate with a single RGB-D camera, which captures RGB image sequences and depth maps to accurately position 3D keypoints. With real-time processing in mind, the system also establishes the need for smooth keypoint tracking over time, by considering the use of a LSTM network. Key considerations include the camera setup, ensuring full visibility of the upper body during rehabilitation exercises, and accommodating different user needs. The architecture incorporates specialized modules for heatmap detection and regression-based approaches, utilizing ResNet and U-Net as backbone models. Overall, this architecture aims to provide accurate, real-time monitoring of upper-limb movements, contributing significantly to rehabilitation practices.

Chapter 5

Development work for the human pose estimation method

The development of the HPE method is a multi-phase process which involves different aspects for complete functioning of the system, from auxiliary tasks to main priorities aiming to accomplish the objectives. This Chapter will elaborate on the tasks performed on the project to build a complete and functional HPE method that can be applied for upper-limb HPE directed for stroke rehabilitation.

5.1 Analysis and processing of selected datasets

For the development of neural networks with high performance, quality datasets are essential as they provide real or synthetic data and accurate ground-truth annotations that guide the model's predictions during training. In this work, the analyzed datasets are selected from the ones shown previously in Table 3.2. This ensures that the chosen datasets are suitable in terms of features relevant to the project's context.

5.1.1 Analysis of dataset content quality

Datasets for HPE are created by a variety of people and institutions, which makes it difficult to streamline the process of analysis and processing of datasets. While most datasets include both images or videos and corresponding annotations, some, like the

University of Idaho - Physical Rehabilitation Movements Data Set (UI-PRMD)[150], provide only text-based annotations without image or video data. This is inadequate for HPE systems based on computer vision, which require input images for training.

Additionally, annotation and depth data can come in various formats, complicating analysis. For example, datasets like University of Texas at Dallas - Multimodal Human Action Dataset (UTD-MHAD)[171] and MSR DailyActivity3D[172] use formats such as `.mat` and `.bin` for depth or 3D joint data. While .mat files can be processed with Matlab or Octave, `.bin` files require specific metadata—such as resolution and frame count—which is often missing, making them difficult to use effectively.

Since the focus was on datasets containing both RGB and depth data, most 2D datasets were briefly reviewed due to their limitation to RGB data and 2D annotations. Exceptions were made for the Common Objects in Context (COCO)[173] and MPII[154] datasets, as they are widely used for evaluation and provide a useful basis for comparing methods.

The 3D datasets selected for training or testing, containing both RGB and depth data with 3D annotations, were Pandora[156] and Human3.6M[152]. Pandora offers easily processed image sequences in `.png` and 3D keypoint annotations in `.json` format. However, access to Human3.6M was not possible due to account restrictions, which was a notable setback, as it is regarded as one of the most popular and comprehensive benchmark for 3D HPE [139].

5.1.2 Dataset initialization and preparation for training

Once the selected datasets were obtained, the first objective is visualizing their content by analyzing the annotation files and displaying sample images. To achieve this, separate Jupyter notebooks are created and prepared for each dataset, as the annotations are structured differently. Initialization is identical between datasets despite their differences, since it follows the same sequence of steps:

1. Open the annotation file and image folder.
2. Map the images to their corresponding annotation data.
3. Define the necessary transformations for images and annotations.
4. Prepare functions to retrieve data by index.

For this, since the models will be developed using PyTorch, the use of its Dataset and DataLoader data primitives allow easy access to the data. A custom dataset needs three functions, which mostly align with the required steps and allow access to it [174]:

- `__init__`: used to initialize both the folder and annotation file and define transforms.
- `__len__`: returns the number of data samples.
- `__get_item__`: given an index number, it loads an image as tensor and applies transformations, gets its corresponding label and finally returns both in a tuple.

After defining the paths for the image folder and `.json` file, the dataset can be created by using `dataset = ImageJointDataset(image_folder, json_path, target_size=(224, 224))` where it is also possible to define extra arguments, such as the image target size after transforms, which in this case is set to 224 pixels in both dimensions. This allows the dataset to be utilized by model architectures that only accept certain input sizes.

From this point on any operation which involves the access to the dataset is performed using this class, simplifying and providing decoupling from future operations such as model training.

The first dataset presented is the COCO dataset, containing 2D keypoints and the corresponding RGB images. The annotations are separated into training and validation files and contain the number of visible keypoints in the image and positions of every keypoint (with coordinates $(0, 0)$ if not visible) as well as bounding boxes and segmentation masks. Figure 5.1 contains the keypoint structure used in the dataset.

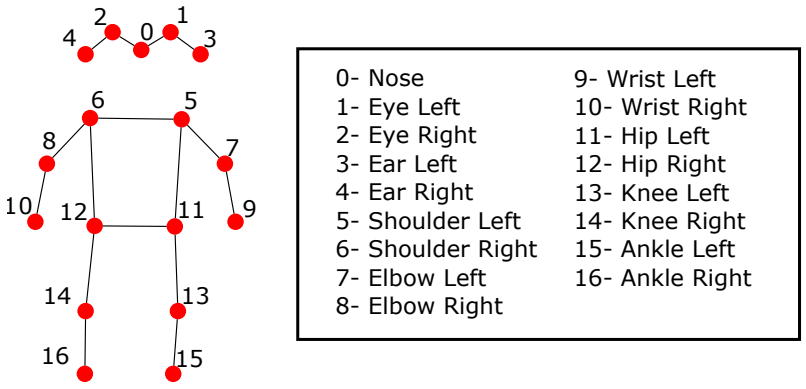


Figure 5.1: Disposition and nomenclatures of the keypoints from the COCO dataset.

An additional feature about the COCO dataset is the fact that it contains multiple keypoint annotations if there are multiple people in the image. This can be beneficial for multi-person pose estimation methods and could even be used in single person estimation if correctly parsed. Sample images from the dataset with overlapped keypoint annotations are shown in Figure 5.2.

The MPII dataset contains RGB images with full-body annotations. The annotation file includes, on top of the joint positions in the 2D space, the visibility

Figure 5.2: Sample images from the COCO dataset overlapped with keypoints.

of each joint, which can be helpful when improving the performance of methods with occluded joints. Its keypoint structure is represented in Figure 5.3 and shows similarities in terms of structure to the Pandora dataset.

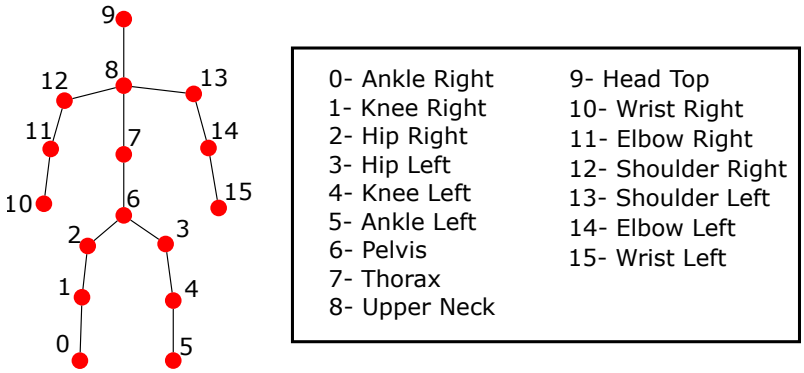


Figure 5.3: Disposition and nomenclatures of the keypoints from the MPII dataset.

Contrary to the COCO dataset, the MPII dataset only contains annotations for one person per image, but it has the advantage of containing human beings in every image, something that the COCO dataset does not. Figure 5.4 contains visualization samples with keypoints overlapped.

Finally, the Pandora dataset provides both RGB and depth images that correspond to the same time instant. These images contain the upper body of a single seated individual, and, as such, only contain upper-body keypoints. While the annotations are stored in the same file, the depth and RGB image sequences are located in separate folders. The Pandora dataset keypoint structure can be seen in Figure 5.5. Sample RGB images from the dataset with overlapped keypoints are shown in Figure 5.6, while Figure 5.7 contains depth images with overlapped keypoints.

Figure 5.4: Sample images from the MPII dataset overlapped with keypoints.

Figure 5.5: Disposition and nomenclatures of the keypoints from the Pandora dataset.

5.1.3 Keypoint overlap on image

MatPlotLib[1] is a Python library can be utilized to visualize the resulting images and keypoints by overlaying them in a plot. The process involves the following steps:

1. Retrieve an item from the dataset.
2. Convert the image to a NumPy array.
3. Display the image in a plot.
4. Overlay the keypoints using a scatter plot.
5. Show the final plot.

Converting the image to a NumPy array is necessary since the `getitem` function returns the image from the dataset after applying transformations, including a `.toTensor()` conversion to a `torch.FloatTensor`, which cannot be visualized directly with MatPlotLib [175]. The keypoints are scattered by using their corresponding x and y coordinates, which have also been resized to match the target size of the image defined by the dataset. By utilizing a loop, multiple images can be visualized simultaneously with subplots, as illustrated in Figure 5.6.

[1] `https://matplotlib.org/` (last accessed in 21/09/2024).

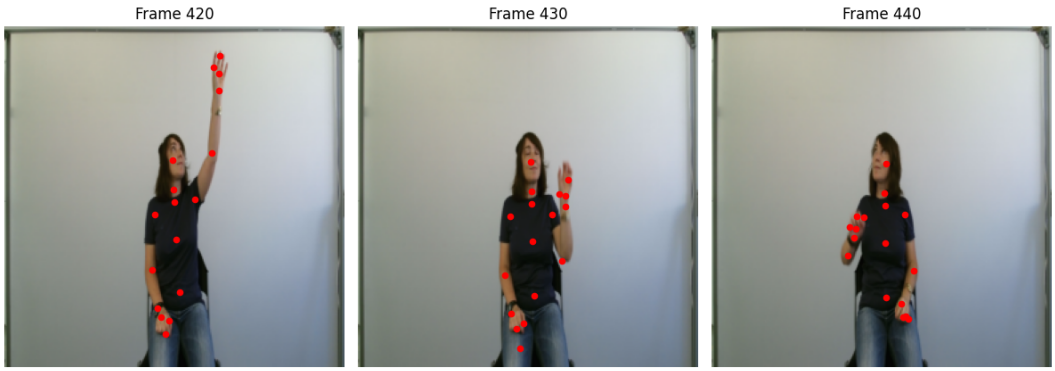


Figure 5.6: Sequence of RGB frames and corresponding keypoints from the Pandora dataset.

Since the Pandora dataset also contains the depth data in image format with the corresponding joint annotations, outputting the image sequence is as simple as changing the image folder, as well as modifying the frame data being accessed inside the same `.json` file. This way, since the dataset class implements the same functions as before, the dataset remains easily accessible. In Figure 5.7 the resulting sequence of depth images and annotations can be found.

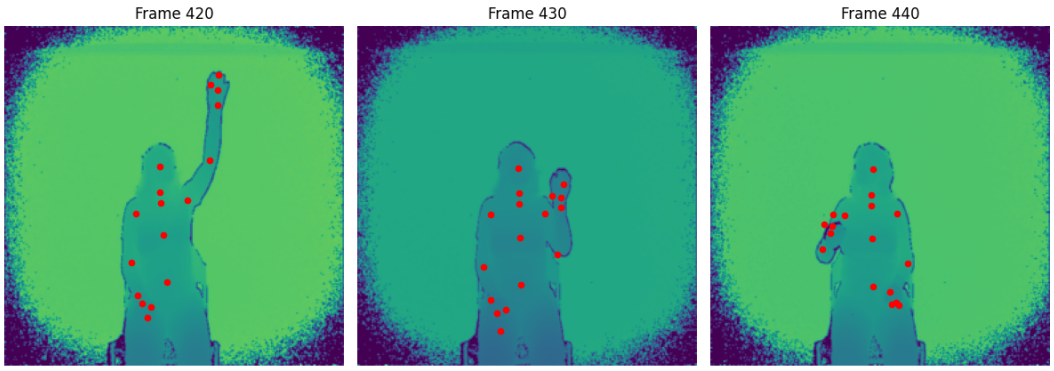


Figure 5.7: Sequence of depth frames and corresponding keypoints from the Pandora dataset.

5.1.4 Heatmap representation of keypoints

Heatmaps are visual representations that use colours to display different levels of relevance. They are employed to indicate probability and frequency of metrics such as clicks in websites [176] and, in case of HPE and the computer vision field, it is used to signal relevant areas in an image and can result from the inference of a neural network which uses heatmap detection-based methods [1].

For model training, it is important to understand and implement an automatic way of converting keypoints into heatmaps and vice-versa, since the datasets only provide keypoint annotations and heatmaps are indispensable for training. For this

case, a new function was developed to generate heatmaps when provided both image resolution and keypoint coordinates, so that each heatmap has the same resolution as the image.

The output heatmaps can be generated and then processed for visualization, as seen in Figure 5.8, where each of the 16 heatmaps are represented with a local maximum indicating each keypoint location of one image from the dataset.

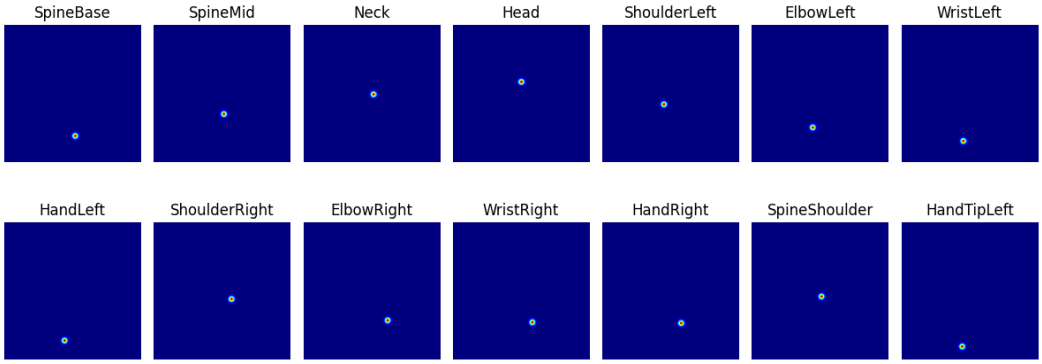


Figure 5.8: Individual heatmaps created from ground truth keypoint coordinates of the Pandora dataset.

Another possibility for the visualization of the heatmaps is to overlap each heatmap on top of the RGB image which allows for easier identification of each of the keypoints in contrast to the individual heatmaps images, as seen in Figure 5.9. For this, the heatmaps are accumulated over a single heatmap, originally totally covered by minimal values, and then the larger values of every heatmap are copied to the single heatmap if smaller values are found for each corresponding pixel. A more detailed explanation about the heatmaps is given in Section 5.3.

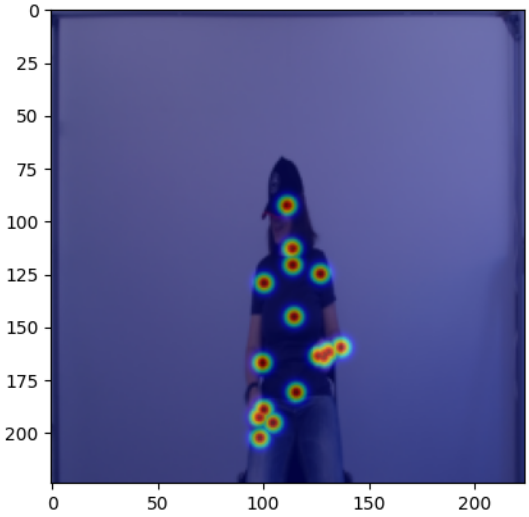


Figure 5.9: Image overlapped with heatmaps converted from ground-truth keypoints from the Pandora Dataset.

5.1.5 3D keypoint visualization

Being a 3D compatible dataset, the Pandora dataset also provides the 3D joints inside the annotation files. Visualizing these requires the extraction of three spacial dimensions and the plotting must be done on a 3D plot, an identical process to the previous 2D visualization. The visualization for the ground truth keypoints from one sample of the Pandora dataset is shown in Figure 5.10.

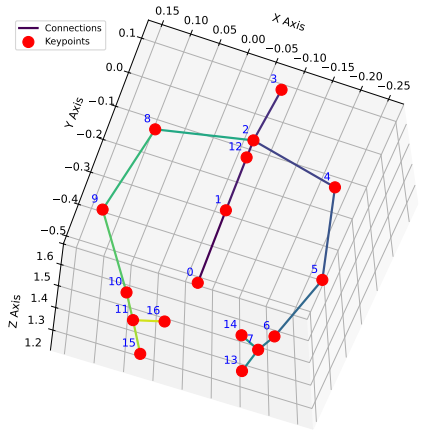


Figure 5.10: Visualization of 3D keypoints from the Pandora dataset.

The view of the keypoints in 3D helps understanding the spatial arrangement and relationships between different joints. By examining these 3D visualizations, one can better interpret the variety of poses, occlusions, and depth variations present in the dataset.

5.2 Training the neural network

In order to obtain a model capable of mapping input images to output keypoints that represent the general position and connections of human body joints it is necessary to train its parameters and make changes in its structure if needed. The learning routine usually involves a training and a validation phase, in which the images and annotations from the dataset are passed through the model.

Based on loss between the ground-truth and the inferred output, adjustments are performed on the values of the weights and bias present throughout the network, with the objective of reducing the loss in the next epoch. Since images are employed as input for the network, they are computed and transformed to be adequate for intake of the network, and the same can be said for the keypoints contained in the

annotation files, which must be in the same format as the output of the network for loss computation.

5.2.1 Dataset initialization for processing during learning

For training, the dataset must be divided to accomplish different tasks. The larger division is called training data, and it is used to train the model. It typically comprises about 80% of the total dataset, since the remaining 20% of the dataset is used for validation, which is important during the learning process to prevent the network from over fitting to the test data. Figure 5.11 shows this division with the MPII dataset [154].

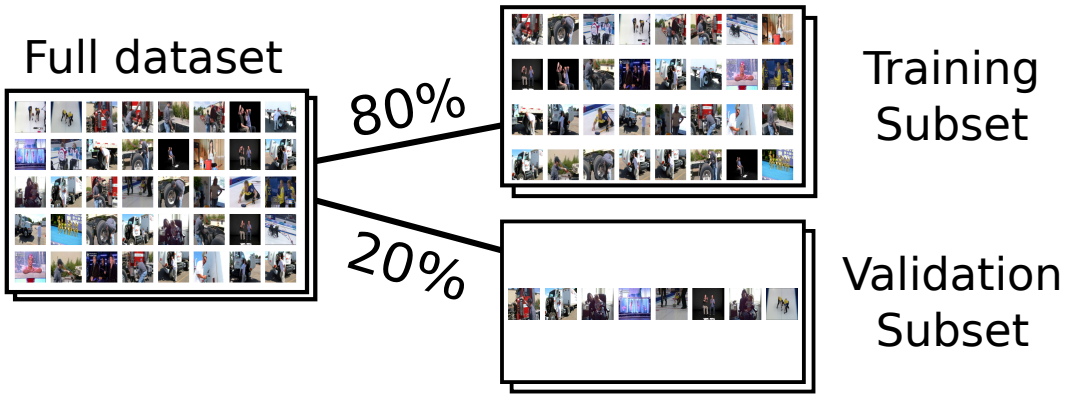


Figure 5.11: Training and validation dataset splitting from the original MPII dataset.

The split is done through the separation of the dataset indexes by using the `train_test_split` utility from SciKit-Learn [177]. Then, since the datasets can be several gigabytes in size when stored, they are loaded in batches for training, reducing system memory requirements. This, however, can impact training speed since the data must be repeatedly pulled from storage, which is slower than system memory and might leave the GPU starving for new data and not be fully utilized during the training.

To pass samples in batches, the Subset class from `torch.utils.data` divides the initial dataset into training and validation datasets by returning the indexes of each image-annotation pair, instead of loading all the image-annotation pairs to memory. The Dataloader class, also from the `torch.utils.data`, then pulls from the data from the Dataset class at each epoch utilizing the indexes to iterate through the subsets [174]. Here, it is possible to configure multiple workers that allow multi-process data loading for the dataset, and enable faster training by feeding the GPU with data more frequently [178].

Alternatively, by using one of the other splitters from SciKit-Learn [179], such as the KFold, it is possible to test various 80/20 divisions (in case of $K = 5$ folds),

evaluate how well the model generalizes and get an idea of performance variability, implementing a procedure known as cross-validation [180]. Each fold is used once as validation while the $K-1$ remaining folds form the training set, as seen in Figure 5.12 [181], where the total of the blue bars per fold corresponds to the 20% of the total data used for validation.

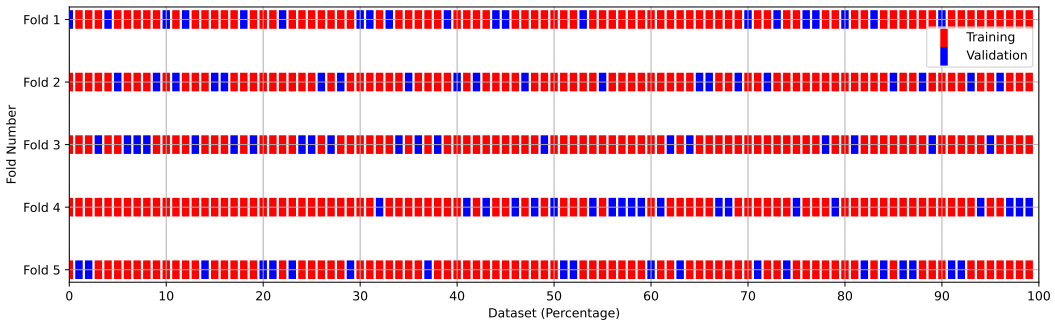


Figure 5.12: KFold division representation of a dataset with shuffled indices and $K = 5$.

This way, the risk of overfitting or even underfitting is reduced, since the model can be evaluated by averaging the performance metrics across all folds, offering a better performance assessment accounting for the variations in training and validation datasets.

5.2.2 Network preparation

In accordance with Section 3.4, the backbones are indispensable since they provide network architectures proven to be effective at image classification and segmentation, computer vision tasks that are in constant development. By harnessing the power of these networks it is possible to use them for HPE tasks while retaining many of the characteristics of these networks.

For this, the `torchvision.models` sub-package can be used to access some models defined for computer vision tasks, as well as pre-trained weights [165]. While this reduces complexity while setting up and testing different backbone architectures and respective weights, only a selected few backbones are available. Specifically, for person keypoint detection there is only one model available that can be loaded directly for regression named Keypoint R-CNN [182], based on the Mask R-CNN architecture, already presented in Subsection 3.4.5.

Other backbones can also be used for HPE after some modifications, such as changes to the dimensions of the output layer. By doing this, a backbone used to classify 1000 objects, like the ResNet family, can retain both the low-level and high-level feature capturing layers and pipeline them into regressing $num\_keypoint \times 2$ values for the x and y that compose the different $num\_keypoints$ used to represent the body of a single person in 2D. Modifying the last layer of a backbone requires

loading the model and accessing the previous final layer. It is necessary to obtain the number of input features entering that layer and then connect a new fully connected (often called dense) layer with the desired output number, to obtain a network such as the one in Figure 5.13.

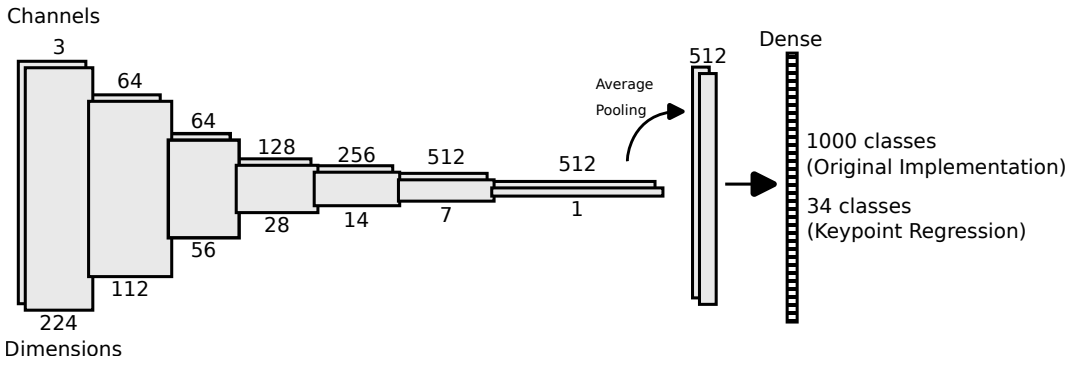


Figure 5.13: ResNet-34 architecture diagram with final fully connected (dense) layer modified for HPE. Adapted from Ruiz [183]

When applying these models for HPE it is fundamental to reflect about their performance in their native applications, and consider that they might not present the best results by simply utilizing them outside their intended field. Moreover, some in-the-wild datasets as the COCO and MPII datasets [154, 173] possess a higher degree of variability and present bigger challenges should not affect the intended rehabilitation scenario. In a more controlled dataset, such as the Pandora [156], is possible that even a simple network can correctly predict the keypoints, but this might also present overfitting during training and not be usable for other scenarios.

Training such models is a long process due to the sheer quantity of parameters that neural networks contain. The use of hardware components such as GPUs allows faster training and inference compared to a system using CPU. In PyTorch the selection of the desired device for computation can be selected with `torch.device`. In this case, the option `"cuda" if torch.cuda.is_available() else "cpu"` is selected, which will use a CUDA compatible device if present on the system or simply resort to the CPU if not found [184]. By selecting the GPU, all parts for training must be allocated to it, meaning model, inputs and other. For the model part, the command `model.to(device);` is applied to pass it to GPU memory (VRAM).

5.2.3 Training setup

Apart from model and dataset, there are other elements of the training that should also be defined before the procedure. The global parameters, or hyper-parameters, must be defined to set aspects such as the amount of data allocated for each loop, also known as epoch, how many epochs will be executed and how aggressive should

the network change its parameters to try reducing the error from the previous loop. These are described bellow:

- Number of epochs: An epoch refers to one cycle through the entire training dataset. Usually, each epoch can be further divided into steps called iterations, which correspond to the number of batches necessary to traverse the dataset. During training, the number of epochs sets how many times the network will perform the full training and validation steps, allowing it to learn from the same data multiple times [185].

- Batch size: This variable controls the sample size of the training and validation data used in each iteration. Since datasets are often large, their content is divided and loaded in batches to alleviate system requirements such as VRAM and system memory [186].

- Learning rate: This parameter changes how quickly a model can learn, by varying how much a model's parameters change. This has influence on the convergence rate of the model in reaching an optimal solution, as seen in Figure 5.14 [187, 188].

- Dataset train/test splitting: Although it is not directly an hyper-parameter, this is usually defined as a value between 0 and 1 representing the percentage of the dataset that is used for training and validation. A common value to divide the datasets is 0.2 which corresponds to 20% of the dataset for validation and 80% for training [177].

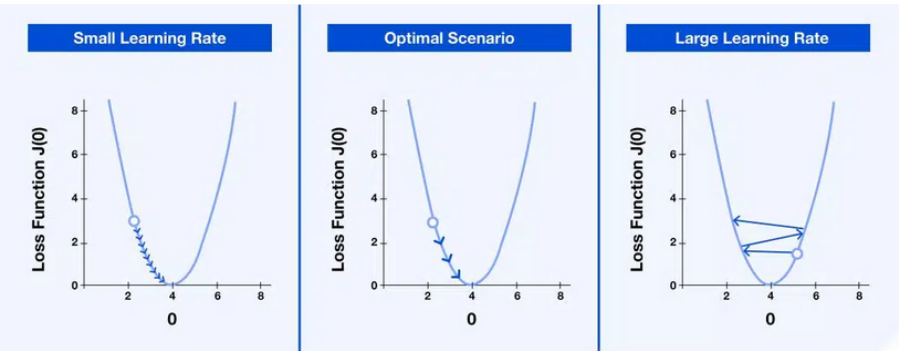


Figure 5.14: Influence of different learning rates in the convergence to an optimal solution [187].

Related to the model training there are two important functions to compute loss and improve model performance. These are:

- Optimizers: The optimizers used are part of the `torch.optim` package and it receives from the model all the parameters to optimize and learning rate, as

well as some specific options, like weight decay and momentum [189]. Some of the algorithms are the following [190]:

  - Adam: Computes adaptive gradient-based optimizations efficiently and with little memory requirements [191].
  - AdamW: Introduced L2 and weight decay regularization on top of Adam which improved on Adam's generalization capabilities [192].
  - RMSprop: Keeps a moving average of the squared gradient for each weight which facilitates escapes from plateaus of tiny gradients [193]. However, Adam reaches faster convergence [194].

- Criterion: The following criterion provided by the `torch.nn` package are used to calculate loss for each forward pass, which is then used for backpropagation to update the model's parameter values. Such criterion are:

  - MSELoss: Calculates the mean squared error (squared L2 norm) between each input and target [195].
  - GaussianNLLLoss: Utilizes a Gaussian negative log likelihood loss that employs expectations and variances from the network [196].

  The criterion is followed by the `.to(device)` method, since it has to be on the same computing device as the model and input data.

These concepts greatly influence the training of neural networks and should also be initialized before the learning routine, since both the loss calculated by the criterion as well as the optimizer are dynamic and their effects indeed change over the course of training. The learning loop consists of 2 sub-processes:

1. Training phase: Where a forward pass is first done on the train subset and then the output is utilized together with the ground-truth to calculate loss and update weights.

2. Validation phase: Which only consists of a forward pass on the validation subset and loss calculation via the criterion. This is done with input samples outside of those used in the training phase, since the objective is to see how well the networks applies on unseen data.

5.2.4 Training optimizations

Although learning occurs during the training phase, the final model is selected based on the best results from the validation phase. This ensures that the model generalizes well to unseen data and performs optimally in real-world scenarios. Improving the performance of the training process is crucial for achieving better results without changing the data or network structure. Some optimization techniques are:

- Early stopping mechanism: Capable of stopping the training loop after a given number of epochs with no improvement. This helps prevent potential overfitting of the model. In general this can be accomplished by the Algorithm 1. Setting a patience value, such as eight, with a total of 200 epochs allows training to stop when there is reasonable certainty that a local or global minimum has been reached.

- Weight decay: Which is used as a parameter of the AdamW optimizer [192], can help the models generalizing better [197].

- Use of a learning rate scheduler: Such as the `ReduceLROnPlateau()`, from the `torch.optim.lr_scheduler` library, that allows dynamic reduction of the learning rate based on metrics if no improvement is seen, such as when the model performance is hitting a plateau [198].

  In this case, the learning rate will update after x epochs without improvements (no new decrease of validation loss) in accordance with the following formula: $newlr = lr * factor$. The metric is initialized before the learning loop and updated after each validation step.

- CUDA as the device for training and inference, and the option of using smaller batches of samples from the original dataset can also be included, since they allow faster development and allow the use of large datasets and mostly disregard system specifications like system memory, which contribute to easily exploring different training strategies.

Following both phases, the average train and validation errors for the current epoch are calculated, which are then used to change, if necessary, parameters such as the learning rate scheduler or saving the newest model if the current epoch has achieved the lowest loss during validation. Finally, the function `torch.cuda.empty_cache()` is used to free unused memory from the GPU so that other applications may use it, such as inference for testing [199].

5.3 Heatmaps

Although this work started experimenting with direct regression methods, the alternative way of using heatmap detection for pose estimation is also touched upon. Heatmaps, due to their nature, need a different processing comparatively to direct regression, since each keypoint outputs a map where each coordinate indicates the probability of containing the predicted joint. This affects the way that the dataset class is built, since now it must return a pair of image and heatmaps, since training the model will require the ground truth heatmap for loss computation against the predicted heatmap.

Algorithm 1 Early stopping algorithm

```
best_loss ← ∞
no_improve_counter ← 0
patience ← 8
while epoch ≠ num_epochs do                          ▷ Start of learning loop
    train_loss ← 0
    val_loss ← 0
    for each training batch do                       ▷ Training loop
        Perform training step and update train_loss
    end for
    for each validation batch do                     ▷ Validation loop
        Perform validation step and update val_loss
    end for
    train_loss ← train_loss/train dataloader length
    val_loss ← val_loss/validation dataloader length
    if val_loss < best_loss then
        best_loss ← val_loss
        Save Torch model
        no_improve_counter ← 0
    else
        no_improve_counter ← no_improve_counter + 1
        if no_improve_counter ≥ patience then
            Message "Early stopping!"
            break                                    ▷ Quit learning loop
        end if
    end if
end while
```

In order to convert the keypoint coordinates to heatmaps, a function that allows the dataset class to transform each keypoint coordinate into a heatmap was created. The function receives parameters, such as the desired heatmap resolution (which is set to the same as the transformed image), the pair of coordinates defining the position of the keypoint in which the heatmap is centered, and a variable that defines the width of the peak.

Equation 5.1 implements a 2D Gaussian function, illustrated in Figure 5.15. In this equation, a defines the maximum height of the peak and is set to 1, while x and y specify the position of the peak's center. The parameter s corresponds to the standard deviation, also known as the Gaussian Root Mean Square (RMS) width, and X and Y represent a point on the plane.

$$heatmap = a \cdot \exp\left(-\frac{(X-x)^2 + (Y-y)^2}{2s^2}\right) \tag{5.1}$$

The backbone architecture used for the heatmap method is the U-Net, previously presented in Section 3.4, since it can extract features with precision due to

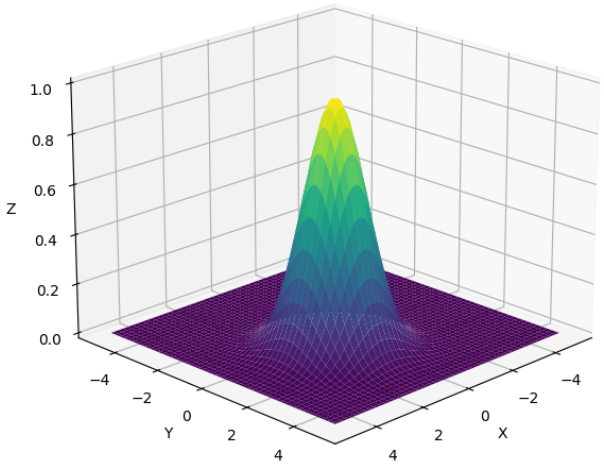


Figure 5.15: 2D Gaussian function centered in $x = 1$ and $y = 1$ with $s = 1$ and $a = 1$.

the encoder-decoder architecture and, although it is mostly used in image segmentation, it can be easily adapted to work in different areas, such as heatmap detection. This PyTorch implementation was taken from a repository in Github [200] and it is applied in semantic segmentation. This can be modified for this use case by instantiating the model with the necessary input channels (three for a RGB image) and enough classes to match the number of keypoints: `model = UNet(n_channels=3, n_classes=17, bilinear=True)`.

Training a heatmap detection method differs slightly from direct regression. The learning loop remains the same structure, only this time both the input image, ground truth heatmaps and model must be in float mode to keep a consistent data type. A 32-bit floating point represents a `torch.FloatTensor` or `torch.cuda.FloatTensor` [201]. The results from the heatmap detection method are shown in Subsection 6.1.1.

5.4 Inference with trained models

A quick way of comparing and verifying the performance of trained models is to visualize the results of inference overlapped on top of an image. While not extensive, since testing is only performed on a few sample images, this can sometimes show performance differences between methods.

5.4.1 Types of visualization

Visualization can be done individually, with one method applied at a time or multiple. Overlapping the keypoints on top of images can also be used to visually compare with the ground-truth keypoint annotation or other methods or trained models.

Observing the results of inference on a single image is the first step to visually understand the results of training and evaluate the quality of the predictions. For this, it is possible to use MatPlotLib to plot the image and then the keypoints, as performed in Subsection 5.1.3, where first the image is placed and then the keypoints are overlapped. This time, the procedure adds a new step, after placing the keypoints, which plots the connections between keypoints for better understanding the human anatomy and the overall pose of the person.

While developing and testing the various models, this single file visualization is the foundation for other visualization scripts, such as the various multi-image and multi-model visualization scripts shown in this document. For this, the function receives the image, keypoints and a list containing the connection pairs. This list is unique for each dataset and must match the keypoint structure from the dataset used to train the model that produces the keypoints. This means that while the connection list depends on the network used, the image for inference does not need to be from that dataset, but might receive different keypoints from the one of its dataset ground truth, as seen in Figure 5.16.

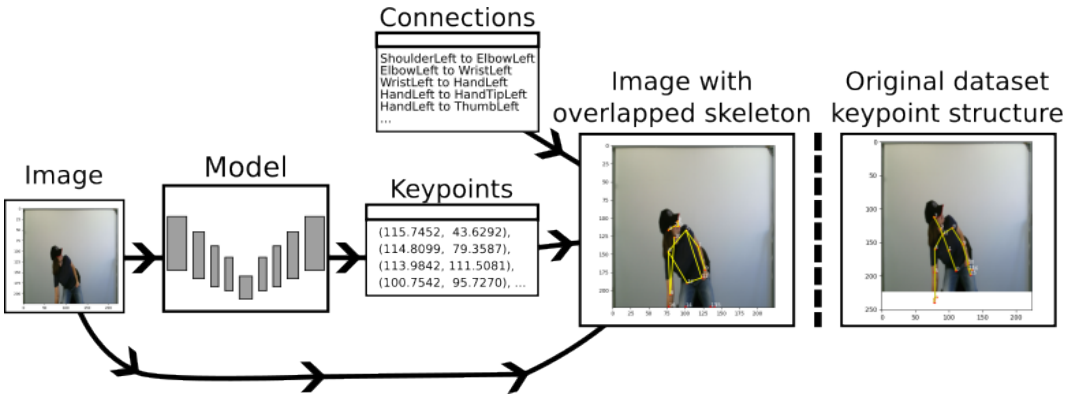


Figure 5.16: Comparison between keypoint structure of inferred skeleton and original dataset due to differences in connection lists.

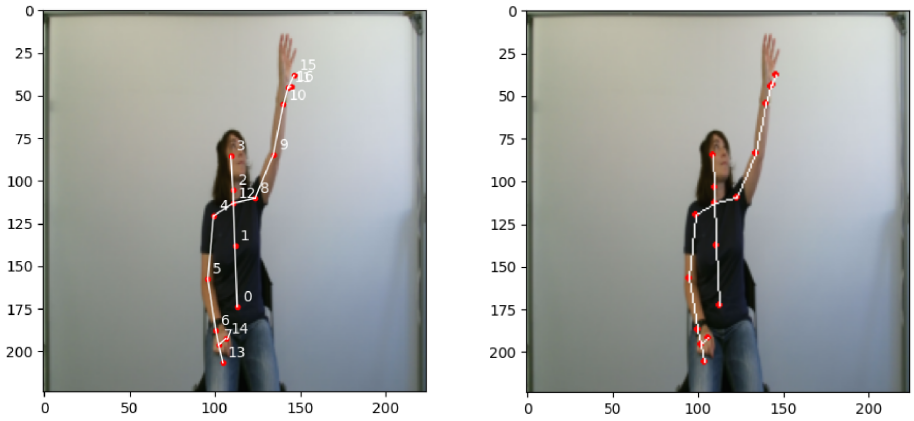
During the development of this work, it was found that a similar function, named `draw_keypoints()`, existed and could be applied for this type of visualization [202]. This function has parameters such as image, keypoints, connectivity and others, used to control colour, radius and width or visibility of the drawn skeleton.

Due to the similarities between this built-in function and the developed function, some testing was performed to compare both in terms of flexibility, appearance and speed. The built-in method offers some customizability, but it doesn't present any major options that couldn't be achieved by using the styling options from MatPlotLib, available on the developed function. The built-in function distinguishes itself by providing a considerably faster execution when displaying an identical image, as seen in Table 5.1, which can be useful in some scenarios, and by allowing keypoint removal if their visibility flag is not set.

Table 5.1: Comparison of the execution speed between the developed and the `draw_keypoints` function from PyTorch for keypoint visualization.

	Developed function	Built-in PyTorch
Inference speed	114 ms ± 1.03 ms	45.2 ms ± 419 µs

The developed function allows for extra features that the built-in function doesn't allow, such as multiple keypoints and skeletons overlapped on the same image or having the index number of each keypoint printed throughout the image, as seen in Figure 5.17, where both were configured to identical looks, but the connections from the built-in function appear as having lower resolution.



(a) (b)

Figure 5.17: Output image comparison between the developed and the `draw_keypoints` function from PyTorch for keypoint visualization: a) developed function, and b) built-in function from PyTorch [202].

This visualization script can then be expanded for multiple methods on the same image or multiple side-by-side images, as seen in Figure 5.18. This way, it is possible to visually compare results from various models.

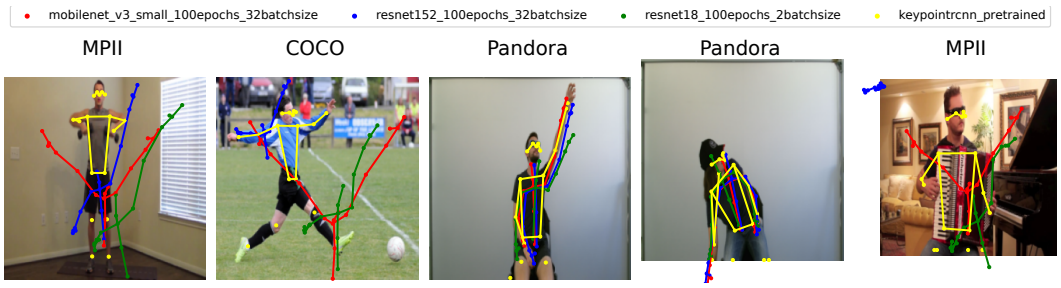


Figure 5.18: Multiple images from the Pandora, COCO and MPII datasets overlapped simultaneously by different methods.

5.4.2 Comparison and benchmarking metrics

Evaluating the performance of the HPE methods should not rely solely on visual inspection of inferred images but also on established metrics. To achieve this, a benchmarking script was developed to assess entire datasets, such as MPII and Pandora, using three key metrics: PCKh, OKS, and PDJ. These metrics, along with their corresponding equations, were introduced in Subsection 3.5.1 and have been implemented in Python.

For these benchmarks, the evaluation focuses exclusively on upper limb joints, which are central to the scope of the work. This includes keypoints such as the head, shoulder, elbow, and wrist, which are commonly present in most datasets and models[2].

An important consideration to make is that keypoints can differ in index, depending on the dataset used for training, as shown in Figure 5.16. Because of this, indices must be manually configured for the models that use different numbering for the corresponding keypoints. The shoulder, elbow, and wrist metrics are calculated for both limbs, rather than using separate metrics for the left and right sides of the body, as done with the PCKh and OKS metrics. The PDJ utilizes the full body annotations whenever possible.

5.5 Temporal smoothing

In the context of HPE, temporal smoothing helps in creating smoother motion trajectories by addressing the common problem of sudden change in keypoint prediction. The main goal is to ensure that the temporal sequence behaves more consistently and alleviate some limitations of the HPE method, such as sudden out-of-bounds keypoints, and brief occlusion occurrences.

5.5.1 Preliminary testing with LSTM

Before applying a LSTM network to keypoint data from a sequential dataset, such as Pandora dataset, the architecture is tested in some scenarios to know what to expect from it. For these tests, a LSTM implementation written in PyTorch is adopted, in which it is possible to change input and output dimensions, as well as hidden dimension and number of hidden layers [203].

Prediction of next value in a sequence

The first test involves the prediction of the next number in a sequence. The chosen sequence is simple, consisting of ten consecutive integers from 1 to 10. The model

[2]In some cases, the head keypoint may be absent. If so, the nose or head top keypoints can serve as replacements.

receives a single feature (input) and outputs one prediction, and due to the simplicity of the task, an hidden size of 50 neurons placed on 1 hidden layer was chosen.

Using a sequence length of 3 time steps to predict the next one, the LSTM model was first trained during 200 epochs, with a learning rate of 0.01, employing a MSELoss criterion and Adam optimizer. At the end of training, a loss of 0.0002 was achieved, and, using a sequence of 8, 9, 10, the model inferred the value 10.403 as the next number in the sequence, when the real number would have been 11.

Prediction of next value in sine wave

Predicting the next value of a sine wave can be more challenging due to the variations in speed and direction of a wave. By employing the same LSTM network and hyper-parameters as before, but modifying sequence length to 10, it was possible to closely predict the next value of the sine wave, as seen in Figure 5.19.

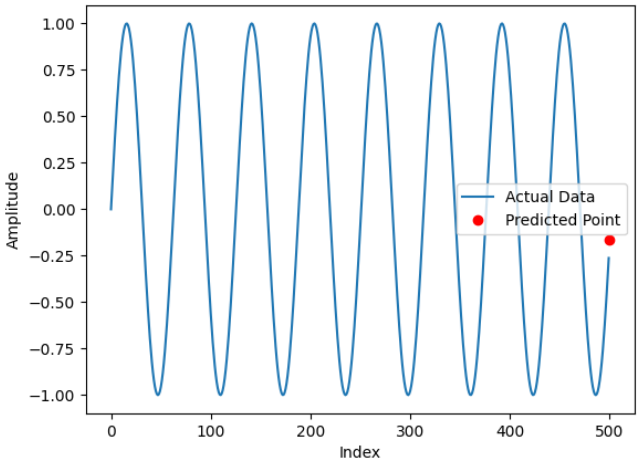


Figure 5.19: LSTM network prediction of next value in sine wave.

The predicted next value is approximately -0.1630, while the actual next value in the sine wave sequence is -0.1645. This close match indicates that the model has effectively captured the underlying pattern of the sine wave.

Prediction of next value in sum of random numbers

A sum of random numbers is a challenging task that involves predicting spontaneous changes in direction and magnitude, dealing with non-repetitive data. This task was performed by the same network and hyper-parameters as in the sine wave prediction, as they were considered enough for the task. The random numbers are selected with a set random seed to ensure replicability of results and to test different parameters for the network and training. The resulting sum of values and prediction can be seen in Figure 5.20.

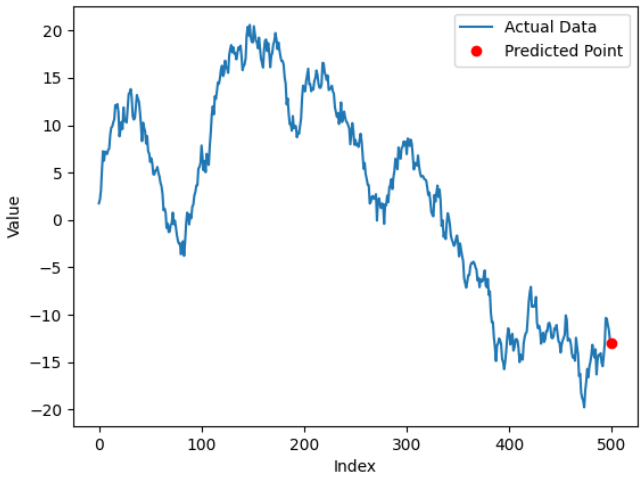


Figure 5.20: LSTM network prediction of next value in a random sum of numbers.

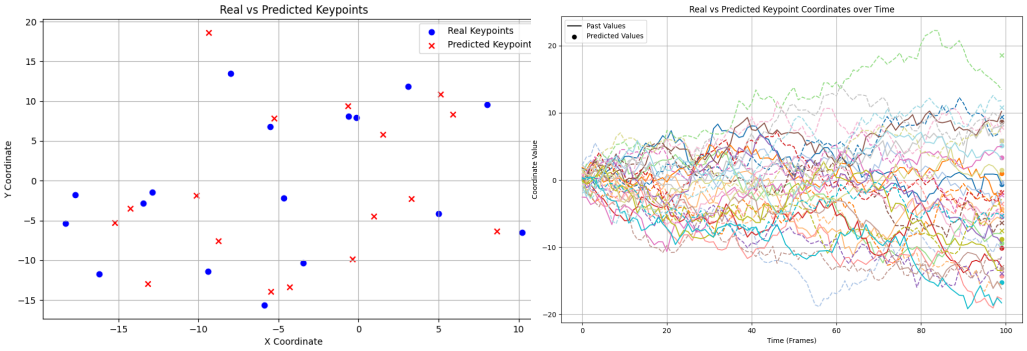
Knowing that the next value is -12.2945, a prediction of -12.9368 demonstrates that the LSTM network has learned to approximate the values reasonably well, capturing the underlying trends and patterns in the data.

Prediction of next coordinates of multiple points

The final test involves the inference of subsequent values for multiple points. This can be considered the same as the prediction of the next value in a sum of random numbers but now considering multiple random sequences, in which each pair constitutes a 2D coordinate. For this exercise, a fist test was performed by the LSTM model with modified input and output layers, containing larger input and output sizes, since the objective consists in obtaining multiple coordinate values for a similar number of coordinates present in a HPE exercise. For this, the input and output sizes are set to $17 * 2$, representing 17 different keypoints in a 2D space.

The model is trained during 200 epochs, finishing with a training loss of 1.7524. Performing inference with this model configuration revealed visually poor performance, as shown in Figure 5.21. Subfigure 5.21a showcases the coordinates set in a 2D plot, which highlights the difference between predicted and real coordinates, while Subfigure 5.21b allows the analysis of the different paths taken by each coordinate value as well as the misalignment between predicted and the final segment of the paths.

Due to the higher complexity of the data, different adjusted models were trained and a final configuration was settled at. This time, the training would occur during 2000 epochs, and the network was designed containing an hidden size of 1000, twenty times as many as the previous model. There was also a change to sequence length, from 10 to 20, to help the model have enough temporal information about past samples. This improved significantly the predictions, as seen in Figure 5.22,

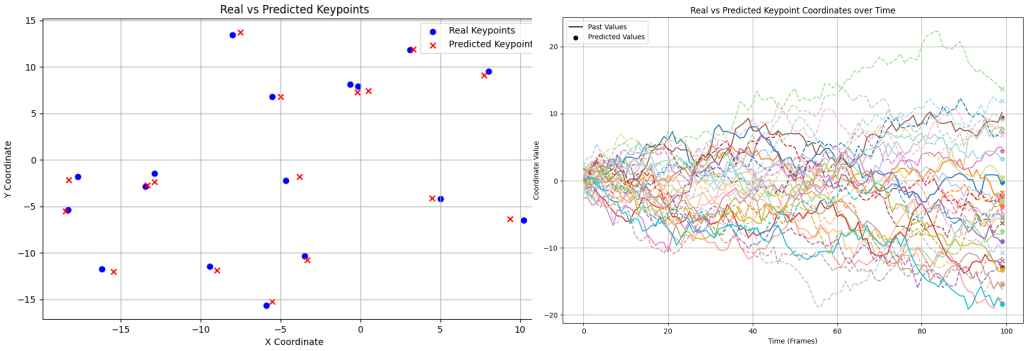


(a) Spacial coordinates of predicted and real points.

(b) Evolution of coordinates over time and prediction of next value.

Figure 5.21: Comparison between real values and LSTM predictions for multiple points before hyper-parameter tuning.

where is possible to see how the predictions slightly match, or even overlap, the real coordinates, visible in Subfigure 5.22a, or how they keep up better with the past samples in Subfigure 5.22b



(a) Spacial coordinates of predicted points and real coordinates.

(b) Evolution of coordinates over time and prediction of next value.

Figure 5.22: Comparison between real values and LSTM predictions for multiple points after hyper-parameter tuning.

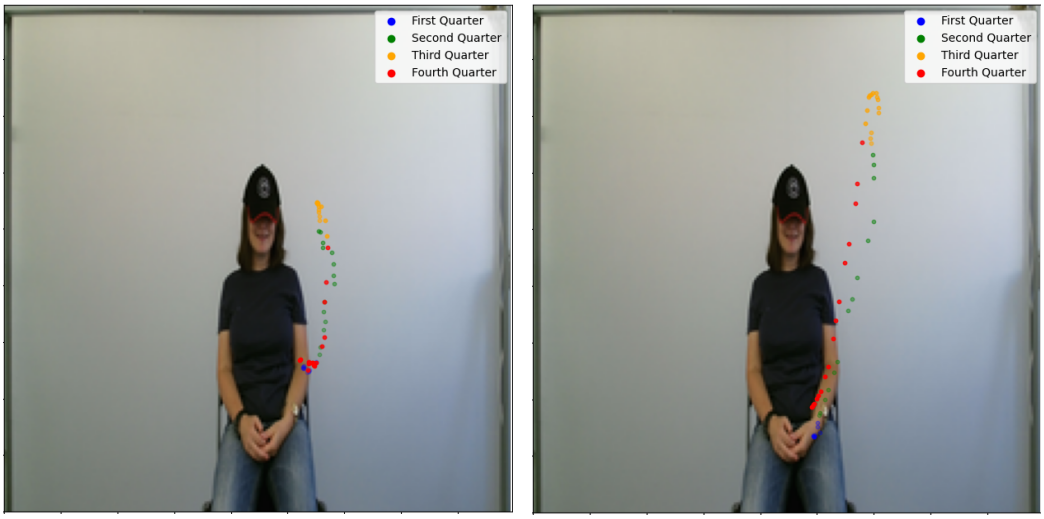
5.5.2 Application of LSTM temporal smoothing in HPE

Applying temporal smoothing into a HPE method involves basically the same as the previous test. This time, since the data comes from a sequential HPE dataset such as Pandora [156], it is necessary to preprocess the dataset before using the data to train the LSTM network.

Since the dataset is already used for training the HPE method, some preparation work is already done. However, the Dataset class still requires modifications, as the goal this time is not to provide an image/annotation pair, but rather a sequence of past annotation samples leading up to a current sample.

By processing the dataset this way, it is possible to receive one object via the indices, gain access to its past and present samples for all keypoints or a single keypoint. For example, if the objective is to represent the past coordinates of the right elbow, the dataset can be accessed for that specific articulation and a temporal window can also be chosen.

Obtaining a representation of these keypoints, such as the right elbow represented in Figure 5.23, involves opening the image, and plotting the keypoint coordinates on top of the image, with a variable color and alpha value for better clarity. This way, the trajectory of the movement performed by any joint can be visualized for a given time frame, such as the elbow visible in Subfigure 5.23a and the hand keypoint in Subfigure 5.23b.



(a) (b)

Figure 5.23: Past joint coordinates visualized from the Pandora dataset [156]: a) Right elbow trajectory, and b) right hand trajectory.

The visualization in 5.23 uses a window size of 60, which is more than necessary for LSTM training and inference. However, this larger window works well for visualization, as it allows to see the positions of the joints over previous time frames. For context, the Microsoft Kinect v2 captures at a rate of 30 frames per second, as mentioned in Table 4.1, thus, a window size of 60 represents two seconds of footage, showing the joint positions throughout that period.

The training results of the LSTM network applied for temporal smoothing are shown in Subsection 6.2, where predicted keypoints are compared with the real ones for the same time instant, and allow for better understanding of its accuracy.

5.6 Camera calibration and device configuration

The choice of the Microsoft Kinect v2 as the RGB-D camera for the implementation of this work lies on its capabilities to capture both RGB and depth data, while being affordable and widely used in research. Out of the box, the camera system already performs well during tests with libfreenect2, an open-source driver for the Microsoft Kinect v2 device [204].

After compiling the drivers[3], running the executable file with the Kinect v2 connected to the system correctly opens up a window with the camera live video feed, as seen in Figure 5.24. Out of the four video feeds in the image, the ones on the right are the most relevant, since both feeds seem to contain overlapped images with few regions without depth data, producing uniform contours around objects, which indicates that the default calibration settings are already adequate.

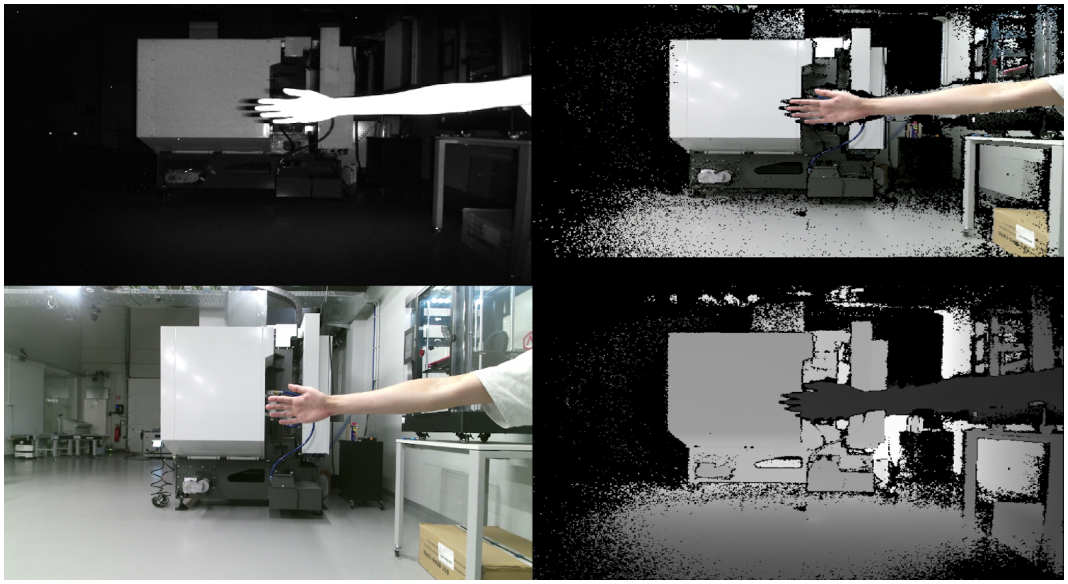


Figure 5.24: Video feed of the Microsoft Kinect v2 camera system, composed by the infrared receptor, RGB sensor and the processed feed of depth pixels colored with the corresponding RGB pixels and a grayscale representation of the depth feed.

In spite of the satisfactory results for the default settings, it was chosen to perform a calibration routine for learning purposes and completeness of this work. For this task, a toolkit with the purpose of allowing the Kinect v2 device in ROS, for robotics, was used, which also depends on the previous libfreenect2 named *iai_kinect2* [170]. More importantly, the toolkit contains:

- A calibration tool for the infrared and RGB and depth measurements.
- An image and point-cloud viewer.

[3]The drivers have been compiled without CUDA support due to an error mentioned in a Github issue [205].

- A bridge between ROS and libfreenect2.

The installation environment of these tools is dependent on the version of ROS and, therefore, a computer with Ubuntu 20.04 is used to install ROS Noetic. Once the environment of ROS has been configured and compilation of the toolkit tools is finalized, the tools can be used. Figure 5.25 presents the calibration tool being used to capture images of the checkerboard. The calibration results and its comparison with pre-calibration are shown in Section 6.3.

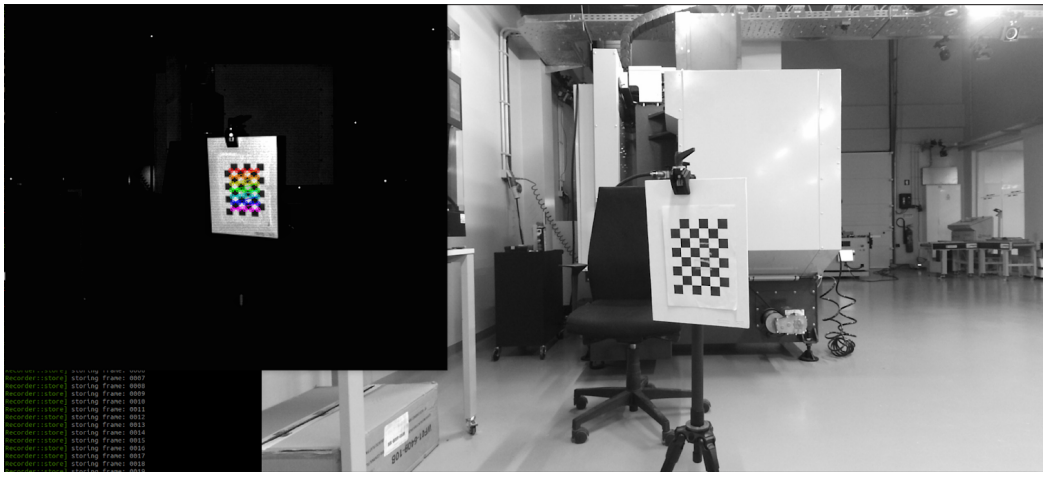


Figure 5.25: Kinect v2 calibration with a checkerboard being performed.

The video feed of the camera system is sent in real-time (at a lower framerate) from the `kinect_bridge` ROS topic over to any listening node, such as `kinect_calibration` and `kinect_viewer`, where they can be used. The calibration program opens a window with viewable RGB and depth streams, and then, if the configured checkerboard is detected with OpenCV's library findChessboardCorners [206], the image can be saved for posterior calibration.

The calibration is performed by utilizing a black and white checkerboard pattern[4] of 5 rows by 7 columns, with 30 centimeters of side per square, that was printed into a paper sheet and then attached to a piece of hard plastic. Then, the pattern was mounted on a tripod, which allowed to position it steadily while the Kinect v2 captured the images. In total, 34 instants were captured, containing the pattern in various positions and angles in relation to the camera.

The captured instants can then be used to perform calibration using the provided tool, which processes the frame data and calculates the intrinsic and extrinsic matrices. To load these calibration parameters it is necessary to create a folder with

[4]Other patterns exist, such as ArUco and ChArUco boards [207].

the Kinect v2 serial number[5] in which the configuration files are stored. Then, by restarting the `kinect_bridge` topic, the new parameters are loaded.

The matrices are stored in a `.yaml` file for each camera (RGB and infrared) and are the following [208, 209]:

- The camera matrix **A**: Composed of the focal lengths f_x and f_y (in pixel units), and the principal point (c_x, c_y), which is typically near the image center. This matrix represents the intrinsic parameters of the camera and is used to project 3D world points into 2D image coordinates.

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

- The distortion coefficients matrix **D**: Composed of five parameters that describe the standard radial lens distortion model. These parameters include radial distortion coefficients k_1, k_2, k_3 and tangential distortion coefficients p_1, p_2. Radial distortion causes straight lines to appear curved, especially toward the edges of an image, while tangential distortion is caused by slight misalignments in the lens.

$$\mathbf{D} = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix}$$

- The rotation matrix **R**: Describes the orientation of the camera in 3D space. It is a 3x3 matrix that represents rotations around the three principal axes (x, y, and z).

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

- The projection matrix **P**: Combines the camera's intrinsic parameters with its extrinsic parameters (rotation and translation) to map 3D world coordinates into 2D image coordinates. It can be represented as:

$$\mathbf{P} = \mathbf{A} \cdot [\mathbf{R} \mid \mathbf{t}] = \begin{bmatrix} f_x & 0 & c_x & t_x \\ 0 & f_y & c_y & t_y \\ 0 & 0 & 1 & t_z \end{bmatrix}$$

5.7 Summary

Chapter 5 details the development process of a HPE method, highlighting the key stages from dataset preparation to model training and testing. The chapter begins

[5]The device serial number can be found when launching `kinect_bridge`, `Protonect` or in `dmesg` when connecting the device.

with an in-depth analysis of selected datasets in Section 5.1, emphasizing the importance of content quality in training effective models. It covers the procedures for dataset initialization and preparation, ensuring that data is suitably formatted and annotated for model learning. Key aspects of dataset analysis include evaluating the quality of the keypoint overlap on images and visualizing these keypoints through heatmaps and 3D representations.

Training the neural network is a significant focus in Section 5.2, detailing the steps necessary to set up and optimize the training process. This includes initializing datasets for training, preparing the network architecture, and configuring training parameters. Training optimizations are discussed to improve efficiency and model performance, ensuring that the network can effectively learn from the data provided. The discussion on heatmaps done in Section 5.3 further reinforces their role in translating keypoint data into a format that enhances model accuracy and performance.

Section 5.4 explains inference with trained models. The various types of visualization techniques are explored, demonstrating how results from pose estimation can be interpreted and analyzed. Additionally, it highlights the comparison and benchmarking metrics used to evaluate the effectiveness of different models, providing a basis for assessing their performance in real-world scenarios.

Temporal smoothing, particularly through the use of LSTM networks, is examined in Section 5.5 to address the challenges of sequential data in pose estimation. It presents preliminary testing results and discusses the application of LSTM for improving the temporal consistency of pose predictions. This section highlights how LSTM can enhance the accuracy and reliability of pose estimation over time.

Finally, the Chapter 5 addresses practical aspects of camera calibration and device configuration in Section 5.6 which ensures accurate pose estimation without depending on default configuration settings, as well as presenting a more standard way of accessing the camera device on the system.

Chapter 6

Results and tests

The current chapter presents the results of the work described in the previous ones. It covers the testing performed on the HPE method, outlining the progress and differences between various network configurations and parameters. A comparison of key HPE metric results is provided, along with an analysis of the different configurations of the LSTM model for temporal smoothing, to determine the optimal setup.

Then, the calibration results are discussed, comparing the default settings with the manually adjusted ones obtained from the calibration script, which used a checkerboard pattern in the captured images. Additionally, the results from programs that provide access to Kinect video through standard Linux devices are examined, alongside the program that calculates depth for coordinates in the RGB image using Kinect depth registration. Finally, the system is tested as a whole, with a focus on visual evaluations to assess its overall performance.

6.1 Experiments with human pose estimation methods

The training resulting from iterative experimentation is shown for three distinct models: ResNet-18 (Model A), U-Net (Model B), and Keypoint R-CNN (Model C). The aim is to evaluate the effectiveness in achieving the objectives outlined in Subsection 1.3.1 and reach a satisfactory level of performance for the given task of upper limb estimation. While model B is fully trained from scratch, model A is built on top of a trained model (which is aiming for a different vision task), applying a

technique called fine tuning, first presented in Subsection 4.2.2 and model C is a pretrained model. The models characteristics can be seen in Table 6.1.

Table 6.1: Description of the models tested for the HPE method.

Model	Backbone	Approach	Dataset
A	ResNet-18 [126]	Direct regression	MPII/Pandora
B	U-Net [200]	Heatmap detection	MPII/Pandora
C	Keypoint R-CNN [182]	Direct regression	COCO (Pre-trained)

The models are tested and executed on a system equipped with an AMD Ryzen 7735HS processor (8 cores, 16 threads), 16 GB of system memory, and a PCIe 4.0 Solid-State Drive (SSD) for storage. The GPU used is an NVIDIA RTX 4060 Mobile with 8 GB of VRAM. The system is tested on Linux Mint 21.3 Cinnamon, with the `6.8.0-40-generic` kernel.

6.1.1 Training results and iterations

For the training, as previously mentioned in Section 5.2.3, the learning process involves the prior initialization and configuration of some aspects. The hyper-parameters can differ from test to test, and, because of this, their value is revealed in each training iteration and is associated with the resulting performance.

Other aspects of the training are kept the same, even when training the LSTM models, such as the criterion and loss function. Every training routines utilizes the AdamW and MSELoss as the optimizer and loss function. The same can be declared for the optimizations that are seen as indispensable, such as the early-stopping mechanism presented in Subsection 5.2.4 which is utilized in every model.

Direct regression (Model A)

The direct regression approach is tested with the ResNet architecture, as mentioned earlier in Subsection 5.2.2, where the model is loaded with its default weights from image classification and then is modified at its final layer to be used in HPE, and able to regress the necessary coordinates of the various joints.

By utilizing different networks, it is possible to test the level of complexity needed for some of the datasets associated with training, since complexity of the network must grow in response to bigger and more diverse datasets. On the other hand, higher complexity of the network also requires a longer period of training and, in some cases, better hardware. As an illustrative example, Figure 6.1 presents the loss curves for the ResNet18 model on both the Pandora and MPII datasets. While the model successfully maps input images to output keypoints on the Pandora dataset, it struggles to achieve the same on the MPII dataset.

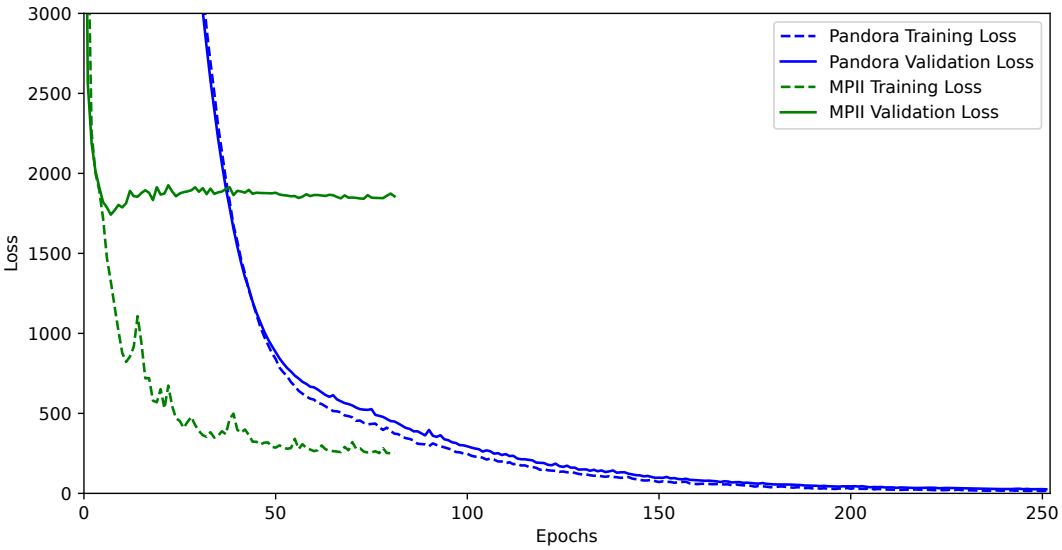


Figure 6.1: Training behaviour of a ResNet18 network on two different datasets: Pandora and MPII.

Since this behavior was seen with the MPII dataset from the start, various attempts have been done to obtain a different outcome, as seen in Table 6.2. However, the dataset's high complexity appears to be incompatible with this particular approach or network architecture.

Table 6.2: Attempts in order to reach a different outcome during the training with the MPII dataset.

Approach	Outcome	Comentary
Smaller subset of the MPII dataset	Similar final losses	Slower loss reduction during initial epochs
Capable ResNet model (ResNet101/151)		Same loss behavior between ResNet models
Partially/totally freezing weights		Increased training speed
Different optimizer (RMSProp)		Losses fluctuated more between epochs

Having the training results in mind, it should not be unexpected that the inference results of this model trained on the MPII dataset would perform poorly, even on the same dataset that it was trained in. These results can be seen in Figure 6.2, which show a lack of adaptability to the different images, only positioning the keypoints near the ground-truth.

In contrast to the inference results on the MPII dataset, the Pandora dataset yields better outcomes when inferring on its own data, as shown in Figure 6.3. While this is encouraging, as it indicates the model has successfully learned to map input images to keypoint coordinates, it also reveals an overfitting issue. Despite the matching validation and training losses, the model struggles to generalize to other datasets, likely due to the similarity between the training and validation data.

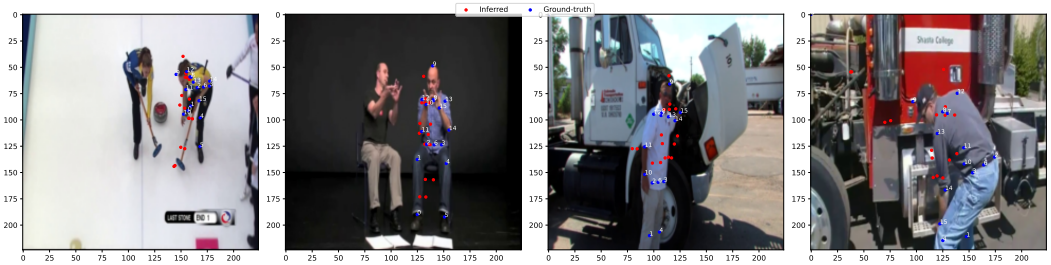


Figure 6.2: Inference results of the ResNet18 model trained and tested on the same MPII dataset. The red dots are inferred and the blue are ground-truth.

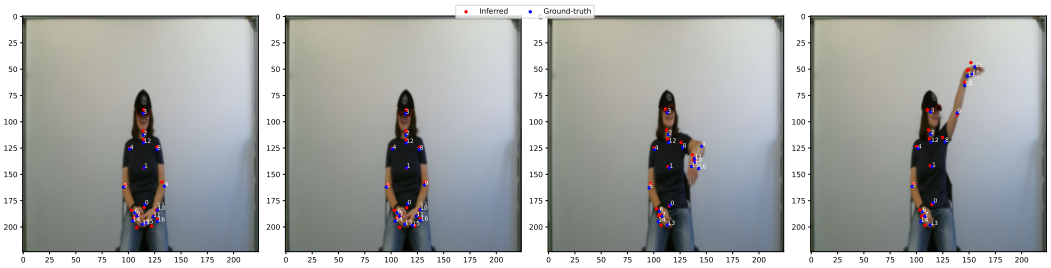


Figure 6.3: Inference results of the ResNet18 model trained and tested on the same Pandora dataset. The red dots are inferred and the blue are ground-truth.

Heatmap detection (Model B)

For the heatmap detection based method, the previously mentioned U-net from Section 5.3 is trained, using the same MPII dataset. This method is more computer intensive, as most heatmap detection methods are, but can learn how to perform more complex mappings in comparison to direct regression [141].

During the learning routine, it was observed training and validation losses with different error scales comparatively to direct regression, as seen in Figure 6.4. While regression losses shows around the hundreds to thousands, during heatmap training the peak loss value is smaller than 1 from the very first epoch. This can be expected, since heatmaps mostly contain values close to zero and only a few regions with error would be located when computing loss.

Examining the loss graph from Figure 6.4, it is evident that the validation loss closely tracked the training loss during the first half of the training, indicating that the model was improving effectively. In the latter half, however, the validation loss began to plateau, suggesting that the model may have reached a local minimum.

The initial attempt with the U-Net architecture in the MPII dataset produced interesting results, more so if the evolution during learning is visualized with the model inferences, as shown in Figure 6.5. For this, every time an updated model resulted from a better epoch (lower validation loss), its inference performance would be tested in a set of five sample images also from the MPII dataset.

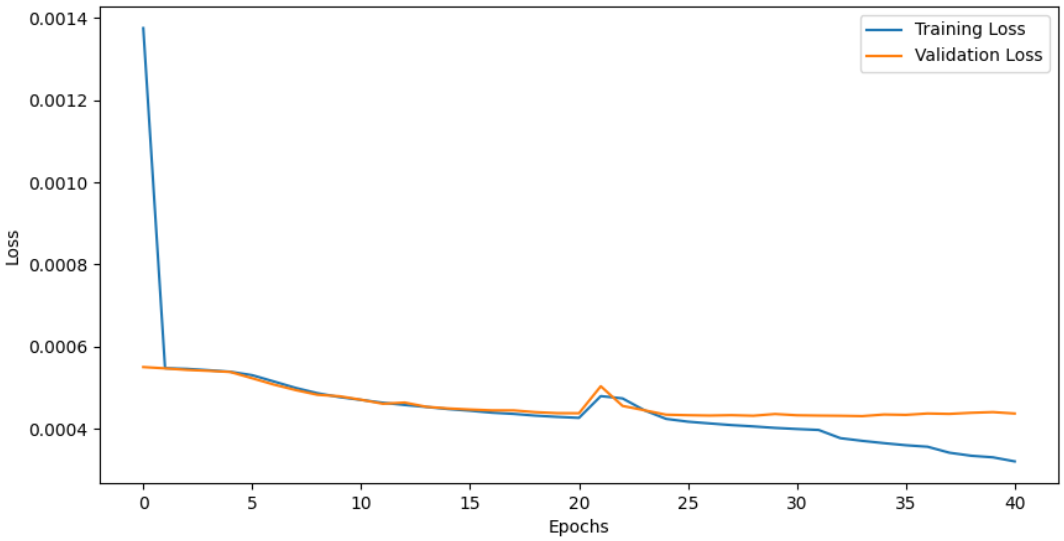


Figure 6.4: U-Net training and validation loss during learning exhibiting a different scale compared to direct regression. This discrepancy arises because the loss is calculated between heatmaps that predominantly contain zeros on most pixels.

The first row of images in Figure 6.5 shows the results from the initial saved model. Although the U-Net architecture was loaded without pre-trained weights, the heatmaps exhibit similarities to the output of a segmentation network. In subsequent model checkpoints, keypoints start to become more distinguishable. In the second row, some keypoints show higher confidence, particularly for the top of the head and neck, where peak values are indicated by dark red colors. From the third row onward, most keypoints are detected with increasing confidence, as shown by the intensification of warm colors, such as yellow, orange, and red. By the fifth and final row, while the model improves overall, it still fails to consistently detect all keypoints in some instances. This is especially noticeable in cases of self-occlusion or when only the upper body is visible, as seen in the third and fifth columns of Figure 6.5.

Further improvements to the performance of the heatmap detection method are achieved by doing a data augmentation technique which involves using random image transformations to get variation on the images of each dataset batch. This way, it can ensure a better performance on new data, as verified in the inference test on a random sample from the COCO dataset, in Figure 6.6, which contains the individual heatmap predictions from the model. Additionally, the heatmaps can provide more information about model performance than direct regression, as seen in Figure 6.7, where the left hand occlusion is visually apparent due to the colour of the heatmaps for the respective keypoints.

For keypoints with low confidence, it is possible to discard any keypoint with a confidence score below a certain threshold, such as 80%, when converting them to

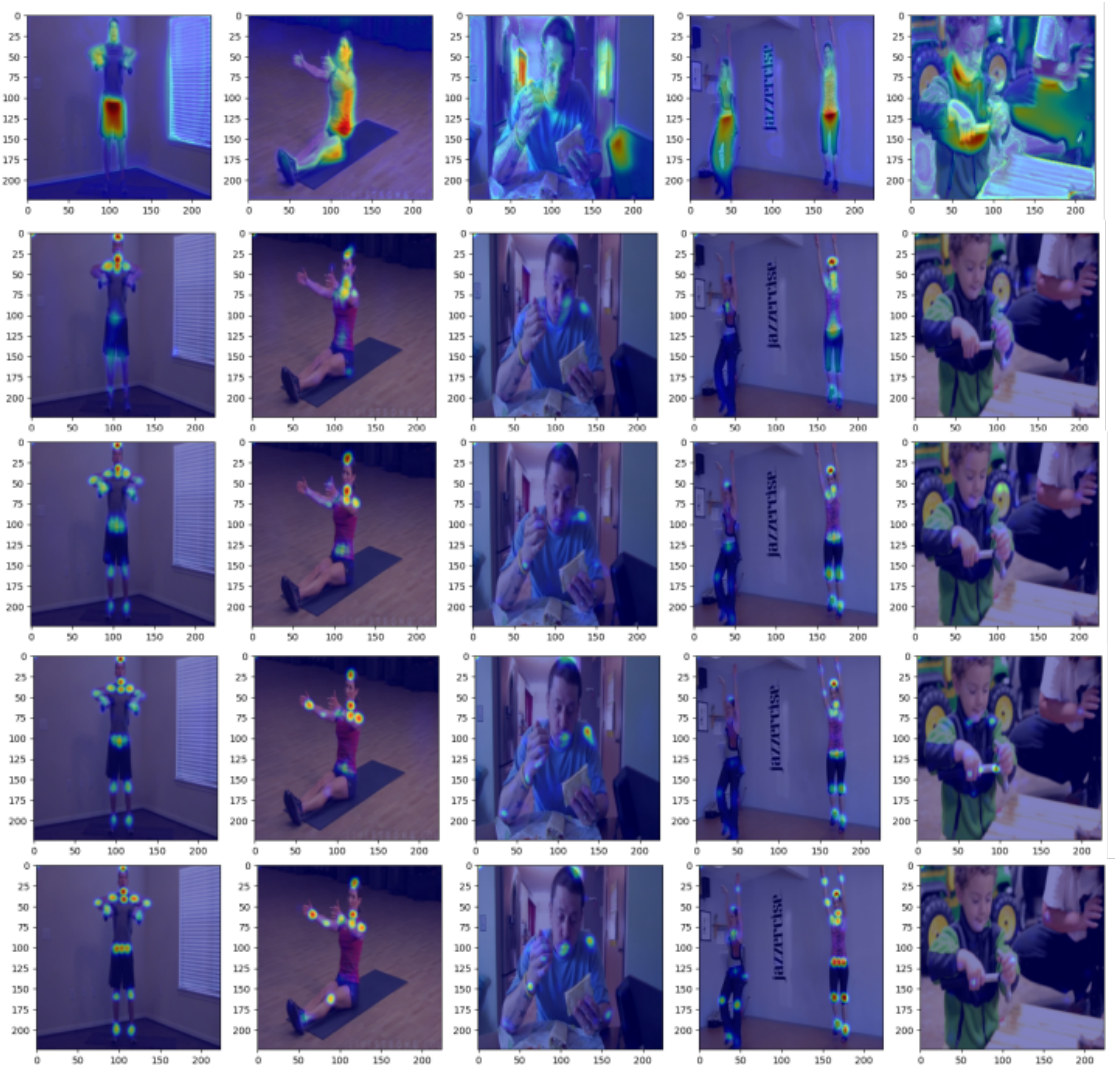


Figure 6.5: Inference results of images from the MPII dataset. Each row represents a new iteration of U-Net weights being updated during training.

coordinates. This results in omitting those keypoints in the final visualization. In Figure 6.8 the difference can be seen between the Subfigure 6.8a which shows all keypoints, including those with low confidence, while Subfigure 6.8b demonstrates the effect of applying an 80% confidence threshold, effectively filtering out outliers.

6.1.2 Benchmark between models

The developed models were also evaluated against each other by utilizing some of the metrics mentioned in Section 3.5.1 on entire datasets, such as the Pandora and MPII. This method can also help comparing cross-validation results, guaranteeing a uniform dataset which does not suffer with different training/validation splits.

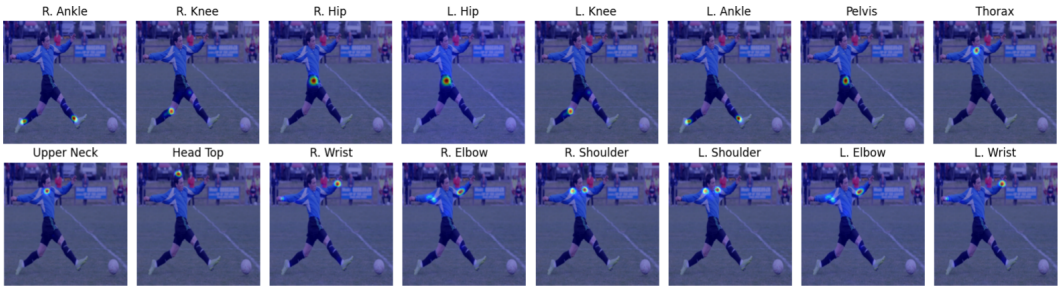
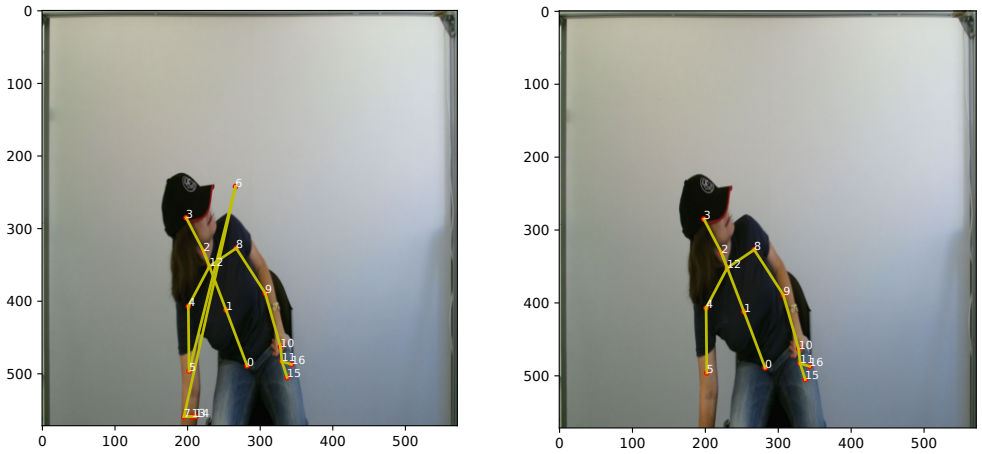


Figure 6.6: Inference results on a sample image from the COCO dataset. The position of the heatmap peaks is mostly correct, but can have problems distinguishing sides of the body.

Figure 6.7: Inference results on a sample image from the Pandora dataset. The left hand is out of the image, which is noticed by the model's generated heatmaps for the keypoints of that region, which contain a different tone of blue background (low confidence).



(a) (b)

Figure 6.8: Inference with keypoint representation of heatmaps with the U-Net on the Pandora dataset. (a) Heatmap generated keypoints with no threshold and (b) Heatmap generated keypoints with 80% threshold.

For the MPII dataset, the ResNet18 was tested using cross-validation on $K = 5$ folds, where it achieved an average validation loss of 1828, as seen in Figure 6.9. The benchmark results of these folds can be consulted in Appendix A, and contains mostly a even result between the different folds, resulting from a balanced dataset division.

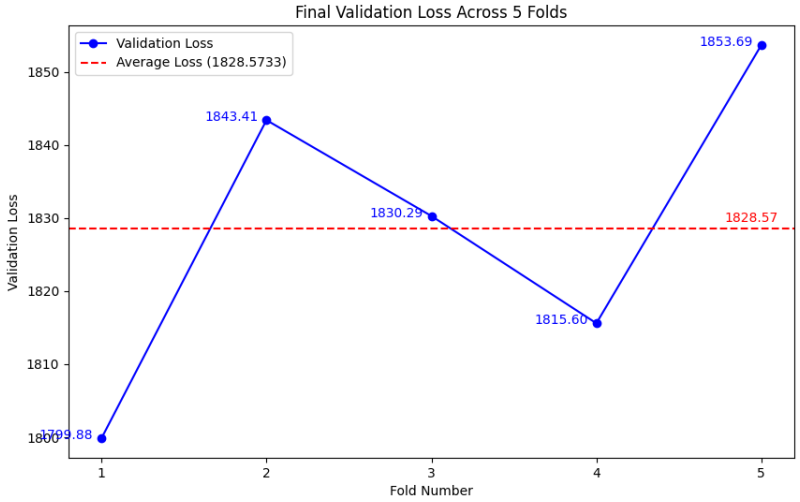


Figure 6.9: Validation losses during cross validation using a ResNet18 architecture on the MPII dataset.

In terms of inference speed, the models are evaluated based on the time required to perform inference and, in the case of the heatmap detection model (Model B), the time to convert heatmaps to keypoints is also considered. The results can be found in Table 6.3. The test is performed with the `%timeit -n 10 -r 10` command, which performs the run 10 times per each 10 loops.

As seen in Table 6.3, the developed models are less resource intensive and complete their tasks significantly faster. This presents a major advantage for these methods, as they are designed for real-time applications, allowing ample time to process additional modules. Based on inference time alone, both methods are capable of keeping pace with a 60 FPS video source, which has an interval of 16.67 milliseconds between frames.

Table 6.3: Inference times of the tested models. The test considers the time it takes to get from the image (as input) to keypoints (output) of a single image. The inference is performed using CUDA as the computational device, with each model running 10 times across 10 iterations

Model	Mean	Deviation
A	1.3 ms	24.1 µs
B	9.2 ms	786 µs
C	54 ms	15.6 ms

6.2 Temporal smoothing with LSTM

The temporal smoothing module is built using an LSTM architecture, as presented in Section 5.5. This Section will showcase the results of the network in a keypoint

estimation scenario, in which the model needs to output the forecast of the next keypoint coordinates while receiving a window of past keypoint positions. The model is trained on the Pandora dataset since it contains labeled frame-by-frame sequences of common upper limb actions.

6.2.1 Training results

The LSTM architecture is similar to the ones used for the preliminary tests. This time, the dataset is divided using a common 80/20 split, which then is sent to the Dataloader object of each subset. Aside from the batch size, this model was tested with different window sizes, where predictions visually close to real keypoints were found at 10, 30 and 60 past samples, and a hidden size of 4800 on a single hidden layer, which helps with the complexity of the task. The training and validation losses evolution over the trained epochs can be seen in Figure 6.10.

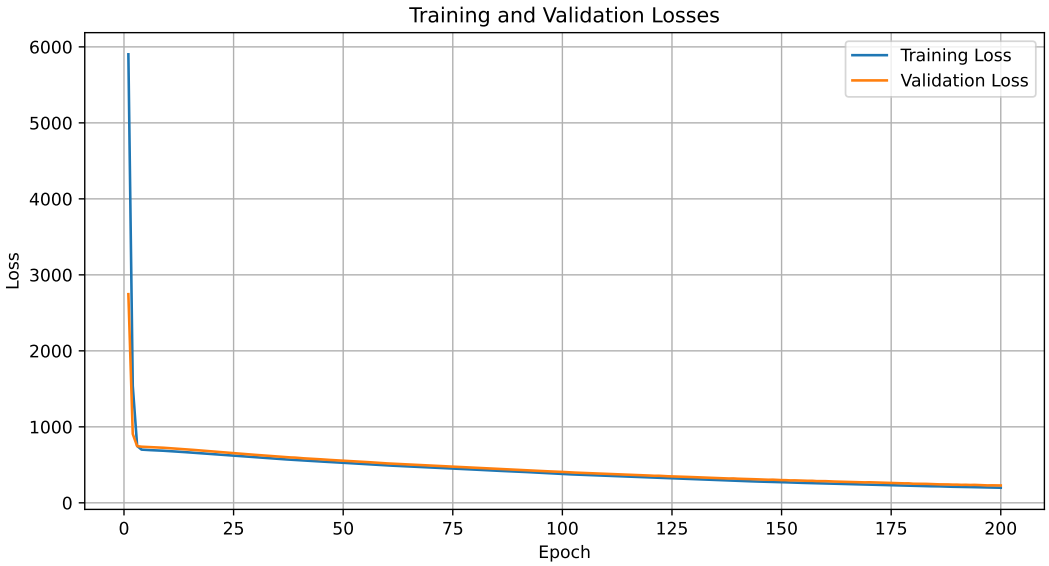


Figure 6.10: Training and validation losses for the LSTM model at the end of the training loop.

6.2.2 Inference analysis and prediction visualization

Predicting keypoints based on past samples with the LSTM model is straightforward, since the model only requires the past samples as an input to generate a prediction for the current time frame. The model was tested on data from the fiftieth index, having access to 9 past samples. The results can be seen in Table 6.4, where both the true (current) coordinates from the tenth sample and the predicted keypoints resulting from the model inference are compared, as well as a column containing the euclidean distance between both coordinates.

Table 6.4: Prediction results when performing inference on Pandora data. All values are in pixels.

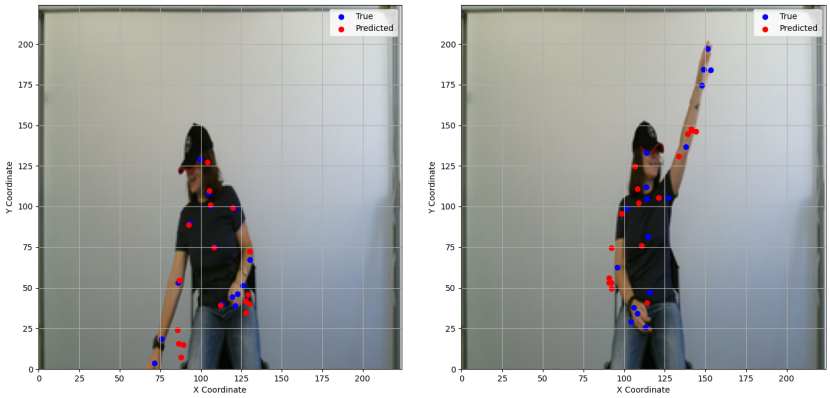
Keypoint	**True**		**Predicted**		**Euclidean Distance (pixels)**
	x	**y**	**x**	**y**	
Spine Base	113.3064	40.0575	112.0500	39.1805	1.5322
Spine Mid	109.0828	75.3992	108.1972	74.8247	1.0556
Neck	105.0474	108.3233	105.4164	109.7055	1.4306
Head	99.4454	128.9203	104.2162	127.2716	5.0477
Shoulder Left	93.1183	89.7694	92.5009	88.5487	1.3679
Elbow Left	85.8202	53.2572	86.5529	54.6730	1.5941
Wrist Left	75.9905	18.3746	85.5173	23.8767	11.0015
Hand Left	71.2885	3.4616	86.3163	15.5852	19.3084
Shoulder Right	120.7960	98.8302	119.8697	99.0921	0.9626
Elbow Right	130.3890	67.1309	130.2364	72.4656	5.3369
Wrist Right	125.8736	51.4187	129.3996	45.9875	6.4754
Hand Right	122.7738	46.3394	127.9460	41.8120	6.8738
Spine Shoulder	106.0320	100.2994	106.0419	101.1291	0.8298
Hand Tip Left	66.3328	-9.2078	87.7778	7.0975	26.9398
Thumb Left	73.2571	-2.4852	89.2004	15.0089	23.6692
Hand Tip Right	119.5535	44.3497	127.7193	34.6691	12.6647
Thumb Right	121.3687	39.0424	130.1098	39.9431	8.7874

The results suggest that certain keypoints, such as the spine, neck, and shoulders, are predicted accurately. However, as the prediction moves towards more mobile keypoints, significant errors become apparent, particularly in areas like the hands, wrists, and even the head.

Furthermore, visual representation of these results, such as the ones seen in Figure 6.11 can help identifying the problems. As an example, in Subfigure 6.11a the hand further down causes problems with keypoint tracking, most likely because the hand is moving down and the LSTM model is not able to keep up with the movement. In Subfigure 6.11b, the movement with the upper hand reaches the peak of its trajectory, slowing down, however, this also proves challenging, since every hand keypoint is far from its true location. The scenarios where both hands are closer to the body have better tracking, which also happens when slower movements are being made.

Finally, the trained model is also tested on the entire dataset, to assess the mean distance from each ground-truth as well as the variance of that distance depending on the movement. To answer this question, in Figure 6.12 it can be consulted a normal distribution plot containing the individual distributions for each keypoint.

The graph shows how some of the keypoints (spine, neck and shoulders) have less variance while achieving small mean errors between the predicted and real keypoint, showing a clear difference between the harder to track keypoints and the ones that



(a) Frame showing a person reaching their hand downward.

(b) Frame showing a person reaching their hand upward.

Figure 6.11: Frames from the Pandora dataset demonstrating the difficulties of the LSTM model in estimating the current keypoint position.

mostly stay in the same position.

6.3 Camera system calibration results

The camera calibration was performed using the setup mentioned previously, in Section 5.6. Utilizing all the images captured with the chessboard pattern, the script calculated the matrices for intrinsics and extrinsics, which were then used in place of the default calibration.

For comparison, Figure 6.13 displays two different calibration settings being used with the Kinect v2. Subfigure 6.13a showcases the use of the default settings, which accomplishes good results, but fails to completely align objects between both video feeds. In contrast, Subfigure 6.13b shows the results after applying calibrated settings. While it should have performed better, since the calibration routine was performed for the device in specific, a identical problem of not achieving complete overlapping was observed.

With this, calibration can be performed in a case-by-case scenario, if needed before using the HPE method. The default calibration settings show visually consistent results, which should not present many problems when performing depth registration in the RGB feed.

6.4 Summary

This chapter presents the results of the experiments and tests conducted on the HPE system. The experiments involved three different models: ResNet-18 (Model A), U-Net (Model B), and Keypoint R-CNN (Model C). Each model employed a

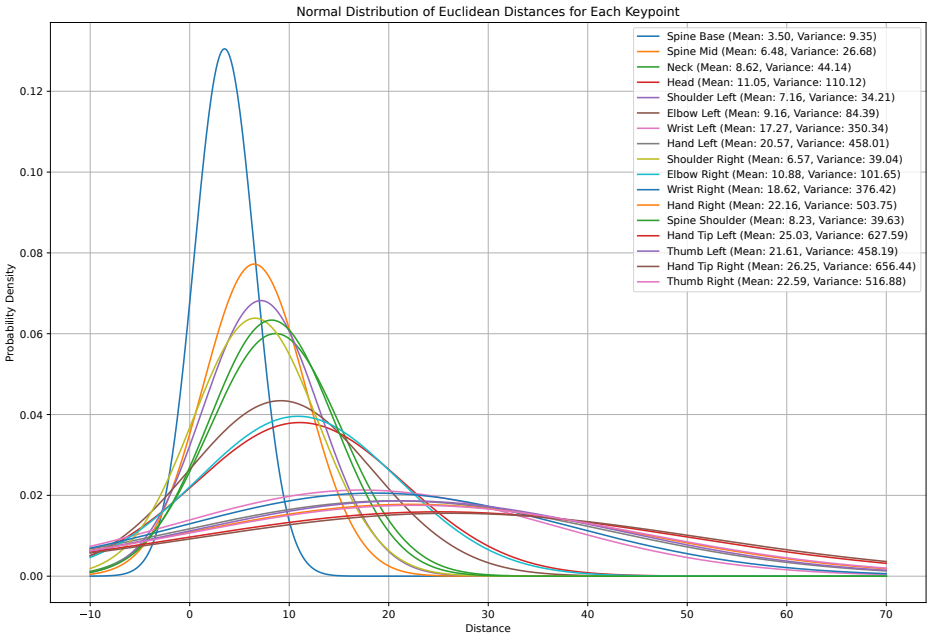
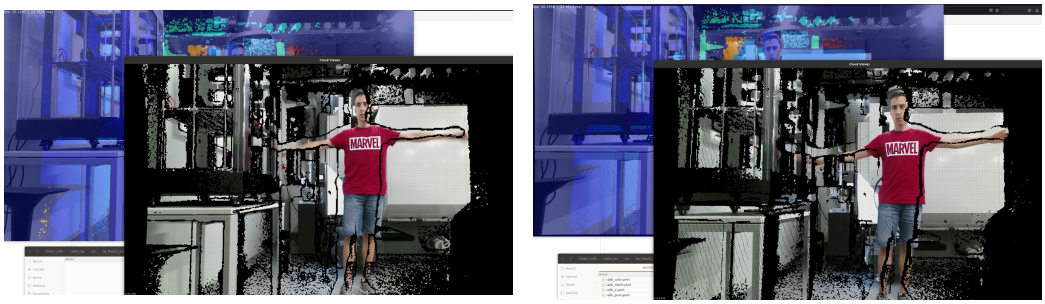


Figure 6.12: Normal distribution of distances from LSTM previsions to ground truth keypoints on the Pandora dataset.



(a) (b)

Figure 6.13: Visual comparison between: (a) default Kinect v2 calibration settings, and (b) manual calibration results.

different approach, ranging from direct regression to heatmap detection, and used a variety of datasets, including Pandora, MPII, and COCO. Model A, using direct regression, performed adequately on simpler datasets like Pandora but struggled when trained on more complex datasets such as MPII, showing limited generalization. On the other hand, Model B's heatmap detection method showed more robust results, especially when enhanced with data augmentation, although it still faced challenges in scenarios with occlusion or complex motions and poses.

The chapter also discusses the role of temporal smoothing using an LSTM architecture. The LSTM model was designed to predict keypoints over time, smoothing the motion by learning from past keypoint positions. It was trained on the Pandora dataset and tested with various window sizes to assess performance. While the LSTM performed well in tracking stable keypoints, such as the spine, neck, and

shoulders, it struggled with more mobile parts, like the hands, where the predictions exhibited greater errors. These findings highlighted the LSTM's ability to maintain smooth transitions in static movements but emphasized the need for improvement in dynamic tracking scenarios.

In addition to the model comparisons and temporal smoothing tests, the system's overall performance was evaluated through benchmarks. Cross-validation on the MPII and Pandora datasets revealed that the models achieved varying levels of success, with the models showing poor validation loss due to dataset complexity on MPII, while displaying more consistent performance with Pandora. Inference speed and accuracy varied between models, with U-Net's heatmap detection offering more detailed output but at the cost of slower processing times compared to direct regression.

Finally, the chapter concludes with an analysis of the camera calibration process. Calibration was performed using a checkerboard pattern, and results were compared between default settings and manually adjusted parameters. Although calibration improved the alignment between RGB and depth data from the Kinect v2 sensor, perfect alignment was not achieved, suggesting that additional case-by-case calibration may be necessary for optimal performance.

Overall, the results emphasize trade-offs between model complexity, accuracy, and computational demands, with room for further refinement in both pose estimation and system calibration.

Chapter 7

Conclusions

This dissertation set out to explore the potential of HPE methods, particularly focusing on upper limb rehabilitation for post-stroke patients. By leveraging advancements in computer vision and machine learning, the aim was to develop a vision-based method that could assist in rehabilitation without the need for intrusive wearable devices. The motivation behind this work stemmed from the increasing prevalence of strokes globally, and the necessity for accessible, personalized rehabilitation methods that can improve the quality of life for individuals recovering from stroke-induced disabilities. However, the development process was not without challenges.

One key limitation was the datasets itselves, as the precision of pose estimation methods heavily relies on the quality and variety of data used for training. Although efforts were made to utilize high-quality datasets, further improvements could be made by incorporating more specialized and larger datasets, that better capture the variety of movements seen in rehabilitation scenarios. Another setback was caused by the long periods of training models, since the architectures used for the models have several parameters, slowing progress. Aside from that, the use of LSTM networks, which were investigated to address the problem of jittery or noisy keypoint predictions, present preliminary results that indicate that integrating temporal data can improve the smoothness and accuracy of pose estimation over time.

In conclusion, this dissertation demonstrates that vision-based pose estimation methods have the potential to significantly enhance stroke rehabilitation practices

by providing non-intrusive monitoring of upper limb movements. The method developed offers a promising step toward more personalized and accessible rehabilitation services.

7.1 Future work

Future work should focus on utilizing custom datasets which could be used to refine the developed models accuracy or be used to finetune an already existing HPE method that already possesses great accuracy. For meaningful application, this system would need to be part of a larger rehabilitation ecosystem, as it has minimal standalone usability, and so, exploring the integration of this system in a real-time environment would be best.

As a standalone system, having a user interface or application for real-time visualization and even calibration would be desirable, as to make the system more interactive and user-friendly.

References

[1] C. Zheng, W. Wu, Chen, *et al.*, "Deep learning-based human pose estimation: A survey," *ACM Comput. Surv.*, vol. 56, Aug. 2023.

[2] T. von Marcard, R. Henschel, M. J. Black, B. Rosenhahn, and G. Pons-Moll, "Recovering accurate 3D human pose in the wild using IMUs and a moving camera," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 614–631, Springer International Publishing, Sept. 2018.

[3] V. L. Feigin, B. A. Stark, C. O. Johnson, *et al.*, "Global, regional, and national burden of stroke and its risk factors, 1990–2019: a systematic analysis for the global burden of disease study 2019," *The Lancet Neurology*, vol. 20, pp. 795–820, Oct. 2021.

[4] V. L. Feigin, M. O. Owolabi, *et al.*, "Pragmatic solutions to reduce the global burden of stroke: a world stroke organization–lancet neurology commission," *The Lancet Neurology*, vol. 22, pp. 1160–1206, Dec. 2023.

[5] S. I. Lee, C. P. Adans-Dester, M. Grimaldi, A. V. Dowling, P. C. Horak, R. M. Black-Schaffer, P. Bonato, and J. T. Gwin, "Enabling stroke rehabilitation in home and community settings: A wearable sensor-based approach for upper-limb motor training," *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 6, pp. 1–11, May 2018.

[6] S. Lee, Y.-S. Lee, and J. Kim, "Automated evaluation of upper-limb motor function impairment using fugl-meyer assessment," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, pp. 125–134, Jan. 2018.

[7] T. v. Marcard, G. Pons-Moll, and B. Rosenhahn, "Human pose estimation from video and IMUs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 1533–1547, Aug. 2016.

[8] J. Tuomilehto, T. Nuottimäki, K. Salmi, K. Aho, M. Kotila, C. Sarti, and D. Rastenytè, "Psychosocial and health status in stroke survivors after 14 years," *Stroke*, vol. 26, pp. 971–975, June 1995.

[9] M. and Grypdonck, "Being a stroke patient: a review of the literature," *Journal of Advanced Nursing*, vol. 26, pp. 580–588, Sept. 1997.

[10] S. J. Murphy and D. J. Werring, "Stroke: causes and clinical features," *Medicine*, vol. 48, pp. 561–566, Sept. 2020.

[11] C. Warlow, "Epidemiology of stroke," *The Lancet*, vol. 352, pp. S1–S4, Oct. 1998.

[12] A. Cotoi and R. Teasell, "Evidence-based review of stroke rehabilitation: Introduction and methods." Available at `https://www.ebrsr.com/sites/default/files/Ch.10UpperExtremityMotorInterventions_v20.pdf`, 2018. (Last accessed in 18/01/2024).

[13] F. S. Sarfo, U. Ulasavets, O. K. Opare-Sem, and B. Ovbiagele, "Tele-rehabilitation after stroke: An updated systematic review of the literature," *Journal of Stroke and Cerebrovascular Diseases*, vol. 27, pp. 2306–2318, Sept. 2018.

[14] Canadian Partnership for Stroke Recovery, "EBRSR Main Page." Available at `https://www.ebrsr.com/`, 2022. (Last accessed in 18/01/2024).

[15] Canadian Partnership for Stroke Recovery, "EBRSR Upper Extremity Motor Rehabilitation." Available at `https://www.ebrsr.com/sites/default/files/Ch.10UpperExtremityMotorInterventions_v20.pdf`, 2018. (Last accessed in 18/01/2024).

[16] A. R. Fugl-Meyer, L. Jääskö, I. Leyman, S. Olsson, and S. Steglind, "The post-stroke hemiplegic patient. 1. a method for evaluation of physical performance.," *Scandinavian journal of rehabilitation medicine*, vol. 7, pp. 13–31, July 1975.

[17] M. R. M. Rodrigues, M. Slimovitch, G. Chilingaryan, and M. F. Levin, "Does the finger-to-nose test measure upper limb coordination in chronic stroke?," *Journal of NeuroEngineering and Rehabilitation*, vol. 14, pp. 6–16, Jan. 2017.

[18] A. Abdullahi, "Upper Limb Self-efficacy Test (UPSET): A Measure of Confidence in the Use of the Upper Limb After Stroke.," *Advances of Science for Medicine*, vol. 1, pp. 10–18, July 2016.

[19] T. Platz, P. Vuadens, C. Eickhof, P. Arnold, S. Van Kaick, and K. Heise, "REPAS, a summary rating scale for resistance to passive movement: Item selection, reliability and validity," *Disability and Rehabilitation*, vol. 30, pp. 44–53, Jan. 2008. PMID: 17852258.

[20] M. A. Murphy, C. Willén, and K. S. Sunnerhagen, "Kinematic variables quantifying upper-extremity performance after stroke during reaching and drinking from a glass," *Neurorehabilitation and Neural Repair*, vol. 25, pp. 71–80, Jan. 2011. PMID: 20829411.

[21] F. Malouin, C. L. Richards, P. L. Jackson, M. F. Lafleur, A. Durand, and J. Doyon, "The Kinesthetic and Visual Imagery Questionnaire (KVIQ) for Assessing Motor Imagery in Persons with Physical Disabilities: A Reliability and Construct Validity Study," *Journal of Neurologic Physical Therapy*, vol. 31, Mar. 2007.

[22] W.-j. Yao and B.-s. Ouyang, "Effect of relaxing needling plus rehabilitation training on post-stroke upper limb dysfunction," *Journal of Acupuncture and Tuina Science*, vol. 12, pp. 146–149, June 2014.

[23] O. H. Kristensen, E. Stenager, and U. Dalgas, "Muscle strength and post-stroke hemiplegia: A systematic review of muscle strength assessment and muscle strength impairment," *Archives of Physical Medicine and Rehabilitation*, vol. 98, pp. 368–380, Feb. 2017.

[24] A. Pollock, S. Farmer, M. Brady, P. Langhorne, G. Mead, J. Mehrholz, and F. van Wijck, "Interventions for improving upper limb function after stroke," *Cochrane Database of Systematic Reviews*, vol. 11, Nov. 2014.

[25] C. Bütefisch, H. Hummelsheim, P. Denzler, and K.-H. Mauritz, "Repetitive training of isolated movements improves the outcome of motor rehabilitation of the centrally paretic hand," *Journal of the Neurological Sciences*, vol. 130, pp. 59–68, May 1995.

[26] J. Classen, J. Liepert, S. P. Wise, M. Hallett, and L. G. Cohen, "Rapid plasticity of human cortical movement representation induced by practice," *Journal of Neurophysiology*, vol. 79, pp. 1117–1123, Feb. 1998. PMID: 9463469.

[27] St. Louis Post-Dispatch - Harry Jackson Jr., "Custom-made rehab helps victims of stroke." Available at `https://www.stltoday.com/life-entertainment/nation-world/wellness/custom-made-rehab-helps-victims-of-stroke/article_06eb5759-3291-5730-930f-725c0d436450.html`, Nov. 2012. (Last accessed in 21/01/2024).

[28] L. S. O. Rocha, G. C. B. Gama, R. S. B. Rocha, L. de Barros Rocha, C. P. Dias, L. L. S. Santos, M. C. de Souza Santos, M. I. de Lima Montebelo, and R. M. Teodori, "Constraint induced movement therapy increases functionality and quality of life after stroke," *Journal of Stroke and Cerebrovascular Diseases*, vol. 30, p. 105774, June 2021.

[29] V. Sirtori, D. Corbetta, L. Moja, and R. Gatti, "Constraint-induced movement therapy for upper extremities in patients with stroke," *Stroke*, vol. 41, pp. e57–e58, Jan. 2010.

[30] D. Corbetta, V. Sirtori, L. Moja, and R. Gatti, "Constraint-induced movement therapy in stroke patients: Systematic review and meta-analysis," *European journal of physical and rehabilitation medicine*, vol. 46, pp. 537–44, Dec. 2010.

[31] R. P. V. Peppen, G. Kwakkel, S. Wood-Dauphinee, H. J. Hendriks, P. J. V. der Wees, and J. Dekker, "The impact of physical therapy on functional outcomes after stroke: what's the evidence?," *Clinical Rehabilitation*, vol. 18, pp. 833–862, Dec. 2004. PMID: 15609840.

[32] D. Corbetta, V. Sirtori, G. Castellini, L. Moja, and R. Gatti, "Constraint-induced movement therapy for upper extremities in people with stroke," *Cochrane Database of Systematic Reviews*, vol. 11, Oct. 2015.

[33] G. Kwakkel, J. M. Veerbeek, E. E. H. van Wegen, and S. L. Wolf, "Constraint-induced movement therapy after stroke," *The Lancet Neurology*, vol. 14, pp. 224–234, Feb. 2015.

[34] M. Garry, A. Loftus, and J. Summers, "Mirror, mirror on the wall: Viewing a mirror reflection of unilateral hand movements facilitates ipsilateral M1 excitability," *Experimental brain research. Experimentelle Hirnforschung. Expérimentation cérébrale*, vol. 163, pp. 118–122, May 2005.

[35] J. H. Morris, F. van Wijck, S. Joice, S. A. Ogston, I. Cole, and R. S. MacWalter, "A comparison of bilateral and unilateral upper-limb task training in early poststroke rehabilitation: A randomized controlled trial," *Archives of Physical Medicine and Rehabilitation*, vol. 89, pp. 1237–1245, July 2008.

[36] National Science Foundation - Raima Larter, "Rewiring the brain to regain control after stroke." Available at `https://medicalxpress.com/news/2011-09-rewiring-brain-regain.html`, Sept. 2011. (Last accessed in 21/01/2024).

[37] M. E. Michielsen, R. W. Selles, J. N. van der Geest, *et al.*, "Motor recovery and cortical reorganization after mirror therapy in chronic stroke patients: A phase II randomized controlled trial," *Neurorehabilitation and Neural Repair*, vol. 25, pp. 223–233, Mar. 2011. PMID: 21051765.

[38] B. Johansson, "Multisensory stimulation in stroke rehabilitation," *Frontiers in Human Neuroscience*, vol. 6, pp. 1–11, Apr. 2012. Article 60.

[39] P. Dias, R. Silva, P. Amorim, J. Laíns, E. Roque, I. Serôdio, F. Pereira, and B. S. Santos, "Using virtual reality to increase motivation in poststroke rehabilitation," *IEEE Computer Graphics and Applications*, vol. 39, pp. 64–70, Jan. 2019.

[40] K. Laver, B. Lange, S. George, J. Deutsch, G. Saposnik, and M. Crotty, "Virtual reality for stroke rehabilitation," *Cochrane Database of Systematic Reviews*, vol. 11, Nov. 2017.

[41] K. E. Laver, B. Lange, S. George, J. E. Deutsch, *et al.*, "Virtual reality for stroke rehabilitation," *Stroke*, vol. 49, pp. e160–e161, Apr. 2018.

[42] MindMaze, "Mindmotion's game-based digital therapies." Available at `https://mindmaze.com/digital-therapies-for-neurorehabilitation/`, Sept. 2011. (Last accessed in 01/03/2024).

[43] Z. Yue, X. Zhang, and J. Wang, "Hand rehabilitation robotics on poststroke motor recovery," *Behavioural Neurology*, vol. 2017, pp. 1–20, Nov. 2017.

[44] P. S. Lum, C. G. Burgar, P. C. Shor, M. Majmundar, and M. Van der Loos, "Robot-assisted movement training compared with conventional therapy techniques for the rehabilitation of upper-limb motor function after stroke," *Archives of Physical Medicine and Rehabilitation*, vol. 83, pp. 952–959, July 2002.

[45] D. Piovesan, M. Casadio, F. A. Mussa-Ivaldi, and P. G. Morasso, "Multijoint arm stiffness during movements following stroke: Implications for robot therapy," in *2011 IEEE International Conference on Rehabilitation Robotics*, pp. 1–7, July 2011.

[46] S. Wang, J. Liu, S. Chen, S. Wang, Y. Peng, C. Liao, and L. Liu, "Recognizing wearable upper-limb rehabilitation gestures by a hybrid multi-feature neural network," *Engineering Applications of Artificial Intelligence*, vol. 127, p. 107424, Jan. 2024.

[47] Y. Sun, X. Li, J. Wang, D. Li, and Y. Zhu, "Research on upper limb rehabilitation action recognition method of unsupervised contrast learning based on time-domain multi-scale feature fusion," in *2022 5th International Conference on Intelligent Robotics and Control Engineering (IRCE)*, pp. 103–107, Sept. 2022.

[48] S. S. Mahmoud, Z. Cao, J. Fu, X. Gu, and Q. Fang, "Occupational therapy assessment for upper limb rehabilitation: A multisensor-based approach," *Frontiers in Digital Health*, vol. 3, pp. 1–15, Dec. 2021.

[49] M. A. Murphy, "About the fugl-meyer assessment." Available at `https://www.gu.se/en/neuroscience-physiology/fugl-meyer-assessment`, July 2023. (Last accessed in 12/01/2024).

[50] Y. Chen, Y. Tian, and M. He, "Monocular human pose estimation: A survey of deep learning-based methods," *Computer Vision and Image Understanding*, vol. 192, p. 102897, Mar. 2020.

[51] Z. Zhang, C. Wang, W. Qiu, W. Qin, and W. Zeng, "Adafuse: Adaptive multiview fusion for accurate human pose estimation in the wild," *International Journal of Computer Vision*, vol. 129, p. 703–718, Nov. 2020.

[52] T. Kaichi, T. Maruyama, M. Tada, and H. Saito, "Resolving position ambiguity of IMU-based human pose with a single RGB camera," *Sensors*, vol. 20, Sept. 2020.

[53] S. Guo, M. Pang, B. Gao, H. Hirata, and H. Ishihara, "Comparison of sEMG-based feature extraction and motion classification methods for upper-limb movement," *Sensors*, vol. 15, pp. 9022–9038, Apr. 2015.

[54] L. Peternel, T. Noda, T. Petrič, A. Ude, J. Morimoto, and J. Babič, "Adaptive control of exoskeleton robots for periodic assistive behaviours based on EMG feedback minimisation," *PLOS ONE*, vol. 11, pp. 1–26, Feb. 2016.

[55] X.-F. Zhang, X. Li, J.-T. Dai, G.-X. Pan, N. Zhang, H.-Q. Fu, J.-G. Xu, Z.-C. Zhong, T. Liu, and Y. Inoue, "The design of a hemiplegic upper limb rehabilitation training system based on surface EMG signals," *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, vol. 12, pp. JAMDSM0031–JAMDSM0031, Mar. 2018.

[56] Z. Lei, "An upper limb movement estimation from electromyography by using BP neural network," *Biomedical Signal Processing and Control*, vol. 49, pp. 434–439, Mar. 2019.

[57] S. Cai, Y. Chen, S. Huang, Y. Wu, H. Zheng, X. Li, and L. Xie, "SVM-based classification of sEMG signals for upper-limb self-rehabilitation training," *Frontiers in Neurorobotics*, vol. 13, June 2019.

[58] J. Shao, Y. Niu, C. Xue, Q. Wu, X. Zhou, Y. Xie, and X. Zhao, "Single-channel SEMG using wavelet deep belief networks for upper limb motion recognition," *International Journal of Industrial Ergonomics*, vol. 76, p. 102905, Mar. 2020.

[59] Y. Wang, Q. Wu, N. Dey, S. Fong, and A. S. Ashour, "Deep back propagation–long short-term memory network based upper-limb sEMG signal classification for automated rehabilitation," *Biocybernetics and Biomedical Engineering*, vol. 40, pp. 987–1001, July 2020.

[60] C. Shen, X. Ning, Q. Zhu, S. Miao, and H. Lv, "Application and comparison of deep learning approaches for upper limb functionality evaluation based on

multi-modal inertial data," *Sustainable Computing: Informatics and Systems*, vol. 33, p. 100624, Jan. 2022.

[61] A. Filippeschi, N. Schmitz, M. Miezal, G. Bleser, E. Ruffaldi, and D. Stricker, "Survey of motion tracking methods based on inertial sensors: A focus on upper limb human motion," *Sensors*, vol. 17, June 2017.

[62] X. Yi, Y. Zhou, and F. Xu, "Transpose: real-time 3D human translation and pose estimation with six inertial sensors," *ACM Trans. Graph.*, vol. 40, Aug. 2021.

[63] L. A. Contreras Rodríguez, E. Cardiel, A. L. Soto, J. Antonio Barraza Madrigal, and P. R. Hernández Rodríguez, "Human upper limb motion recognition using IMU sensors and artificial neural networks," in *2022 19th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pp. 1–4, Nov. 2022.

[64] S. Datta, C. K. Karmakar, A. S. Rao, B. Yan, and M. Palaniswami, "Automated scoring of hemiparesis in acute stroke from measures of upper limb co-ordination using wearable accelerometry," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, pp. 805–816, Apr. 2020.

[65] A. Hua, P. Chaudhari, N. Johnson, J. Quinton, B. Schatz, D. Buchner, and M. E. Hernandez, "Evaluation of machine learning models for classifying upper extremity exercises using inertial measurement unit-based kinematic data," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, pp. 2452–2460, Sept. 2020.

[66] B. Oubre, J.-F. Daneault, H.-T. Jung, K. Whritenour, J. G. V. Miranda, J. Park, T. Ryu, Y. Kim, and S. I. Lee, "Estimating upper-limb impairment level in stroke survivors using wearable inertial sensors and a minimally-burdensome motor task," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, pp. 601–611, Mar. 2020.

[67] S. Gokturk, H. Yalcin, and C. Bamji, "A Time-Of-Flight depth sensor - system description, issues and solutions," in *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pp. 35–35, Jan. 2004.

[68] M. Li, F. Wei, Y. Li, S. Zhang, and G. Xu, "Three-dimensional pose estimation of infants lying supine using data from a kinect sensor with low training cost," *IEEE Sensors Journal*, vol. 21, pp. 6904–6913, Mar. 2021.

[69] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, pp. 4–10, Feb. 2012.

[70] Q. Wang, G. Kurillo, F. Ofli, and R. Bajcsy, "Evaluation of pose tracking accuracy in the first and second generations of microsoft kinect," in *2015 International Conference on Healthcare Informatics*, pp. 380–389, Oct. 2015.

[71] H. Sarbolandi, D. Lefloch, and A. Kolb, "Kinect range sensing: Structured-light versus time-of-flight kinect," *Computer Vision and Image Understanding*, vol. 139, pp. 1–20, Oct. 2015.

[72] H. P. H. Shum, E. S. L. Ho, Y. Jiang, and S. Takagi, "Real-time posture reconstruction for microsoft kinect," *IEEE Transactions on Cybernetics*, vol. 43, pp. 1357–1369, Oct. 2013.

[73] I. Theodorakopoulos, D. Kastaniotis, G. Economou, and S. Fotopoulos, "Pose-based human action recognition via sparse representation in dissimilarity space," *Journal of Visual Communication and Image Representation*, vol. 25, pp. 12–23, Jan. 2014. Visual Understanding and Applications with RGB-D Cameras.

[74] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *CVPR 2011*, pp. 1297–1304, June 2011.

[75] P. McIlroy, S. Izadi, and A. Fitzgibbon, "Kinectrack: 3D pose estimation using a projected dense dot pattern," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, pp. 839–851, June 2014.

[76] Q. Wu, G. Xu, S. Zhang, Y. Li, and F. Wei, "Human 3d pose estimation in a lying position by rgb-d images for medical diagnosis and rehabilitation," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 5802–5805, July 2020.

[77] M. Stommel, M. Beetz, and W. Xu, "Model-free detection, encoding, retrieval, and visualization of human poses from kinect data," *IEEE/ASME Transactions on Mechatronics*, vol. 20, pp. 865–875, Apr. 2015.

[78] Š. Obdržálek, G. Kurillo, F. Ofli, R. Bajcsy, E. Seto, H. Jimison, and M. Pavel, "Accuracy and robustness of kinect pose estimation in the context of coaching of elderly population," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1188–1193, Aug. 2012.

[79] E. E. Stone and M. Skubic, "Evaluation of an inexpensive depth camera for passive in-home fall risk assessment," in *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pp. 71–77, May 2011.

[80] D. Antón, A. Goñi, A. Illarramendi, J. J. Torres-Unda, and J. Seco, "Kires: A kinect-based telerehabilitation system," in *2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013)*, pp. 444–448, Oct. 2013.

[81] C. Schönauer, T. Pintaric, H. Kaufmann, S. Jansen Kosterink, and M. Vollenbroek-Hutten, "Chronic pain rehabilitation with a serious game using multimodal input," in *2011 International Conference on Virtual Rehabilitation*, pp. 1–8, June 2011.

[82] T. Pintaric and H. Kaufmann, "Affordable infrared-optical pose-tracking for virtual and augmented reality," in *Proceedings of Trends and Issues in Tracking for Virtual Environments Workshop, IEEE VR*, pp. 44–51, Mar. 2007.

[83] W. Kim, J. Sung, D. Saakes, C. Huang, and S. Xiong, "Ergonomic postural assessment using a new open-source human pose estimation technology (openpose)," *International Journal of Industrial Ergonomics*, vol. 84, p. 103164, July 2021.

[84] H. Tang, Q. Wang, and H. Chen, "Research on 3d human pose estimation using rgbd camera," in *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pp. 538–541, July 2019.

[85] F. Uccheddu, R. Furferi, L. Governi, and M. Carfagni, "Rgb-d-based method for measuring the angular range of hip and knee joints during home care rehabilitation," *Sensors*, vol. 22, Dec. 2022.

[86] T. L. Munea, Y. Z. Jembre, H. T. Weldegebriel, L. Chen, C. Huang, and C. Yang, "The progress of human pose estimation: A survey and taxonomy of models applied in 2D human pose estimation," *IEEE Access*, vol. 8, pp. 133330–133348, July 2020.

[87] A. Rohan, M. Rabah, T. Hosny, and S.-H. Kim, "Human pose estimation-based real-time gait analysis using convolutional neural network," *IEEE Access*, vol. 8, pp. 191542–191550, Oct. 2020.

[88] X. Wei, P. Zhang, and J. Chai, "Accurate realtime full-body motion capture using a single depth camera," *ACM Trans. Graph.*, vol. 31, Nov. 2012.

[89] D. Weiss and B. Taskar, "Learning adaptive value of information for structured prediction," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, (Red Hook, NY, USA), p. 953–961, Curran Associates Inc., Dec. 2013.

[90] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1302–1310, July 2017.

[91] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5686–5696, June 2019.

[92] H. Y. Jung, S. Lee, Y. S. Heo, and I. D. Yun, "Random tree walk toward instantaneous 3d human pose estimation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2467–2474, June 2015.

[93] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real-time human pose tracking from range data," in *Computer Vision – ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), (Berlin, Heidelberg), pp. 738–751, Springer Berlin Heidelberg, Oct. 2012.

[94] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, June 2005.

[95] L. Song, G. Yu, J. Yuan, and Z. Liu, "Human pose estimation and its application to action recognition: A survey," *Journal of Visual Communication and Image Representation*, vol. 76, p. 103055, Apr. 2021.

[96] L. Bourdev, S. Maji, T. Brox, and J. Malik, "Detecting people using mutually consistent poselet activations," in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), (Berlin, Heidelberg), pp. 168–181, Springer Berlin Heidelberg, Sept. 2010.

[97] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *CVPR 2011*, pp. 1385–1392, June 2011.

[98] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic, "3D pictorial structures for multiple human pose estimation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1669–1676, June 2014.

[99] S. Dubey and M. Dixit, "A comprehensive survey on human pose estimation approaches," *Multimedia Systems*, vol. 29, pp. 167–195, Feb. 2023.

[100] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "Vnect: real-time 3D human pose estimation with a single RGB camera," *ACM Transactions on Graphics*, vol. 36, p. 1–14, July 2017.

[101] G. Rogez, P. Weinzaepfel, and C. Schmid, "LCR-Net: Localization-Classification-Regression for human pose," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1216–1224, July 2017.

[102] W. Tang and Y. Wu, "Does learning specific features for related parts help human pose estimation?," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1107–1116, June 2019.

[103] S. Johnson and M. Everingham, "Clustered pose and nonlinear appearance models for human pose estimation," in *Proceedings of the British Machine Vision Conference*, pp. 12.1–12.11, BMVA Press, Sept. 2010. doi:10.5244/C.24.12.

[104] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2D human pose estimation: New benchmark and state of the art analysis," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3686–3693, June 2014.

[105] S. Ju, M. Black, and Y. Yacoob, "Cardboard people: a parameterized model of articulated image motion," in *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pp. 38–44, Oct. 1996.

[106] T. Cootes, C. Taylor, D. Cooper, and J. Graham, "Active shape models-their training and application," *Computer Vision and Image Understanding*, vol. 61, pp. 38–59, Jan. 1995.

[107] H. Sidenbladh, F. De la Torre, and M. Black, "A framework for modeling the appearance of 3D articulated figures," in *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pp. 368–375, Mar. 2000.

[108] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "SCAPE: shape completion and animation of people," *ACM Trans. Graph.*, vol. 24, p. 408–416, July 2005.

[109] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: a skinned multi-person linear model," *ACM Trans. Graph.*, vol. 34, Oct. 2015.

[110] J. Wang, S. Tan, X. Zhen, S. Xu, F. Zheng, Z. He, and L. Shao, "Deep 3d human pose estimation: A review," *Computer Vision and Image Understanding*, vol. 210, p. 103225, Sept. 2021.

[111] Y. Huang, F. Bogo, C. Lassner, A. Kanazawa, P. V. Gehler, J. Romero, I. Akhter, and M. J. Black, "Towards accurate marker-less human shape and pose estimation over time," in *2017 International Conference on 3D Vision (3DV)*, pp. 421–430, Oct. 2017.

[112] M. Trumble, A. Gilbert, A. Hilton, and J. Collomosse, "Deep autoencoder for combined human pose estimation and body model upscaling," in *Computer Vision – ECCV 2018* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), (Cham), pp. 800–816, Springer International Publishing, Oct. 2018.

[113] I. Akhter and M. J. Black, "Pose-conditioned joint angle limits for 3d human pose reconstruction," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1446–1455, June 2015.

[114] A. Nibali, Z. He, S. Morgan, and L. Prendergast, "3d human pose estimation with 2d marginal heatmaps," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1477–1485, Jan. 2019.

[115] P.-L. Liu and C.-C. Chang, "Simple method integrating openpose and rgb-d camera for identifying 3d body landmark locations in various postures," *International Journal of Industrial Ergonomics*, vol. 91, p. 103354, Sept. 2022.

[116] Z. Fang, A. Wang, C. Bu, and C. Liu, "3d human pose estimation using rgbd camera," in *2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI)*, pp. 582–587, Sept. 2021.

[117] C. Li and G. H. Lee, "Generating multiple hypotheses for 3d human pose estimation with mixture density network," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9879–9887, June 2019.

[118] M. Fieraru, A. Khoreva, L. Pishchulin, and B. Schiele, "Learning to refine human pose estimation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 318–31809, Dec. 2018.

[119] W. Ouyang, X. Chu, and X. Wang, "Multi-source deep learning for human pose estimation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2337–2344, June 2014.

[120] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4733–4742, June 2016.

[121] X. Nie, J. Feng, J. Zhang, and S. Yan, "Single-stage multi-person pose machines," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6950–6959, Oct. 2019.

[122] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, p. 84–90, May 2017.

[123] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1653–1660, June 2014.

[124] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, May 2015.

[125] N. N. Prakash, V. Rajesh, D. L. Namakhwa, S. Dwarkanath Pande, and S. H. Ahammad, "A densenet cnn-based liver lesion prediction and classification for future medical diagnosis," *Scientific African*, vol. 20, p. e01629, July 2023.

[126] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.

[127] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, Dec. 2015.

[128] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, (Cambridge, MA, USA), p. 91–99, MIT Press, Dec. 2015.

[129] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, Oct. 2017.

[130] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.

[131] R. B. Olaf Ronneberger, Philipp Fischer and T. Brox, "Our U-net wins two Challenges at ISBI 2015." Available at `https://lmb.informatik.uni-freiburg.de/people/ronneber/isbi2015/`, Apr. 2015. (Last accessed in 16/09/2024).

[132] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), (Cham), pp. 234–241, Springer International Publishing, Nov. 2015.

[133] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 640–651, May 2017.

[134] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 483–499, Springer International Publishing, Sept. 2016.

[135] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun, "Cascaded pyramid network for multi-person pose estimation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7103–7112, Dec. 2018.

[136] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep high-resolution representation learning for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, pp. 3349–3364, Oct. 2021.

[137] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, M. Elgharib, P. Fua, H.-P. Seidel, H. Rhodin, G. Pons-Moll, and C. Theobalt, "Xnect: real-time multi-person 3d motion capture with a single rgb camera," *ACM Trans. Graph.*, vol. 39, Aug. 2020.

[138] P. Kumar, S. Chauhan, and L. K. Awasthi, "Human pose estimation using deep learning: review, methodologies, progress and future research directions," *International Journal of Multimedia Information Retrieval*, vol. 11, pp. 489–521, Dec 2022.

[139] A. El Kaid and K. Baïna, "A systematic review of recent deep learning approaches for 3d human pose estimation," *Journal of Imaging*, vol. 9, Dec. 2023.

[140] Z. Liu, J. Zhu, J. Bu, and C. Chen, "A survey of human pose estimation: The body parts parsing based methods," *Journal of Visual Communication and Image Representation*, vol. 32, pp. 10–19, Oct. 2015.

[141] Q. Dang, J. Yin, B. Wang, and W. Zheng, "Deep learning based 2D human pose estimation: A survey," *Tsinghua Science and Technology*, vol. 24, pp. 663–676, Dec. 2019.

[142] M. Eichner, M. Marin-Jimenez, A. Zisserman, and V. Ferrari, "2d articulated human pose estimation and retrieval in (almost) unconstrained still images," *International Journal of Computer Vision*, vol. 99, pp. 190–214, Sept. 2012.

[143] M. Andriluka, U. Iqbal, E. Insafutdinov, L. Pishchulin, A. Milan, J. Gall, and B. Schiele, "PoseTrack: A benchmark for human pose estimation and tracking," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5167–5176, June 2018.

[144] Y. Yang and D. Ramanan, "Articulated human detection with flexible mixtures of parts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 2878–2890, Dec. 2013.

[145] A. Haque, B. Peng, Z. Luo, A. Alahi, S. Yeung, and L. Fei-Fei, "Towards viewpoint invariant 3D human pose estimation," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 160–177, Springer International Publishing, Sept. 2016.

[146] J. C. Gower, "Generalized procrustes analysis," *Psychometrika*, vol. 40, pp. 33–51, Mar. 1975.

[147] X. Sun, B. Xiao, F. Wei, S. Liang, and Y. Wei, "Integral human pose regression," in *Computer Vision – ECCV 2018* (V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds.), (Cham), pp. 536–553, Springer International Publishing, Oct. 2018.

[148] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "DeeperCut: A deeper, stronger, and faster multi-person pose estimation model," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 34–50, Springer International Publishing, Sept. 2016.

[149] M. Guidolin, E. Menegatti, and M. Reggiani, "Unipd-bpe: Synchronized rgb-d and inertial data for multimodal body pose estimation and tracking," *Data*, vol. 7, June 2022.

[150] A. Vakanski, H.-p. Jun, D. Paul, and R. Baker, "A data set of human body movements for physical rehabilitation exercises," *Data*, vol. 3, Jan. 2018.

[151] B. Sapp and B. Taskar, "MODEC: Multimodal decomposable models for human pose estimation," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3674–3681, June 2013.

[152] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 1325–1339, Dec. 2014.

[153] A. Miron, N. Sadawi, W. Ismail, H. Hussain, and C. Grosan, "IntelliRehabDS (IRDS)—a dataset of physical rehabilitation movements," *Data*, vol. 6, Apr. 2021.

[154] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2D human pose estimation: New benchmark and state of the art analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[155] T. Pfister, K. Simonyan, J. Charles, and A. Zisserman, "Deep convolutional neural networks for efficient pose estimation in gesture videos," in *Computer Vision – ACCV 2014* (D. Cremers, I. Reid, H. Saito, and M.-H. Yang, eds.), (Cham), pp. 538–552, Springer International Publishing, Apr. 2015.

[156] G. Borghi, M. Venturelli, R. Vezzani, and R. Cucchiara, "Poseidon: Face-from-depth for driver pose estimation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5494–5503, IEEE, July 2017.

[157] A. Lopes, R. Souza, and H. Pedrini, "A survey on rgb-d datasets," *Computer Vision and Image Understanding*, vol. 222, p. 103489, Sept. 2022.

[158] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, S. Sridhar, G. Pons-Moll, and C. Theobalt, "Single-shot multi-person 3d pose estimation from monocular rgb," in *3D Vision (3DV), 2018 Sixth International Conference on*, IEEE, Sept. 2018.

[159] Y. Z. Xinliang Zhang, Wanru Wang and H. Xie, "An improved yolov3 model based on skipping connections and spatial pyramid pooling," *Systems Science & Control Engineering*, vol. 9, pp. 142–149, Sept. 2021.

[160] H.-S. Fang, J. Li, H. Tang, C. Xu, H. Zhu, Y. Xiu, Y.-L. Li, and C. Lu, "Alphapose: Whole-body regional multi-person pose estimation and tracking in real-time," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, pp. 7157–7173, June 2023.

[161] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015.

[162] P. H. Quanjer, A. Capderou, M. M. Mazicioglu, A. N. Aggarwal, S. D. Banik, S. Popovic, F. A. Tayie, M. Golshan, M. S. Ip, and M. Zelter, "All-age relationship between arm span and height in different ethnic groups," *European Respiratory Journal*, vol. 44, pp. 905–912, Oct. 2014.

[163] J. Jiao, L. Yuan, W. Tang, Z. Deng, and Q. Wu, "A post-rectification approach of depth images of kinect v2 for 3d reconstruction of indoor scenes," *ISPRS International Journal of Geo-Information*, vol. 6, Nov. 2017.

[164] Z. Tian, H. Chen, and C. Shen, "Directpose: Direct end-to-end multi-person pose estimation," 2019.

[165] PyTorch Website, "Pytorch Models and pre-trained weights." Available at `https://pytorch.org/vision/stable/models.html#models-and-pre-trained-weights`, 2024. (Last accessed in 02/09/2024).

[166] OpenCV University, "TensorFlow Bootcamp." Available at `https://opencv.org/university/free-tensorflow-keras-course/`, 2018. (Last accessed in 27/05/2024).

[167] Y. Luo, J. Ren, Z. Wang, W. Sun, J. Pan, J. Liu, J. Pang, and L. Lin, "Lstm pose machines," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5207–5215, June 2018.

[168] A. Jain, J. Tompson, Y. LeCun, and C. Bregler, "Modeep: A deep learning framework using motion features for human pose estimation," in *Computer Vision – ACCV 2014* (D. Cremers, I. Reid, H. Saito, and M.-H. Yang, eds.), (Cham), pp. 302–315, Springer International Publishing, Jan. 2015.

[169] Y. Zhao and P. Krähenbühl, "Real-time online video detection with temporal smoothing transformers," in *Computer Vision – ECCV 2022* (S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, eds.), (Cham), pp. 485–502, Springer Nature Switzerland, Oct. 2022.

[170] T. Wiedemeyer, "IAI Kinect2." Available at `https://github.com/code-iai/iai_kinect2`, 2014 – 2015. (Last accessed in 07/09/2024).

[171] C. Chen, R. Jafari, and N. Kehtarnavaz, "Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor," in *2015 IEEE International Conference on Image Processing (ICIP)*, pp. 168–172, Sept. 2015.

[172] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1290–1297, June 2012.

[173] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 740–755, Springer International Publishing, May 2014.

[174] PyTorch Website, "Datasets & DataLoaders." Available at `https://pytorch.org/tutorials/beginner/basics/data_tutorial.html`, 2018. (Last accessed in 09/06/2024).

[175] PyTorch Forums, "How to visualize/display a data image in 'torch.FloatTensor' type." Available at `https://discuss.pytorch.org/t/how-to-visualize-display-a-data-image-in-torch-floattensor-type/7770/2`, 2024. (Last accessed in 11/06/2024).

[176] Heap Website, "What Are Heatmaps? How They Work and Ways You Can Use Them." Available at https://www.heap.io/topics/what-are-heatmaps, 2024. (Last accessed in 11/07/2024).

[177] Scikit Learn Website, "Train-test-split API Reference." Available at https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html, 2024. (Last accessed in 11/06/2024).

[178] PyTorch Website, "Pytorch Multi-process data loading." Available at https://pytorch.org/docs/stable/data.html#multi-process-data-loading, 2024. (Last accessed in 02/09/2024).

[179] Scikit Learn Website, "Splitters API Reference." Available at https://scikit-learn.org/stable/api/sklearn.model_selection.html#splitters, 2024. (Last accessed in 02/09/2024).

[180] M. Alhamid, "What is Cross-Validation?." Available at https://towardsdatascience.com/what-is-cross-validation-60c01f9d9e75, 2020. (Last accessed in 09/09/2024).

[181] Scikit Learn Website, "KFold API Reference." Available at https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html#kfold, 2024. (Last accessed in 02/09/2024).

[182] PyTorch Website, "Keypoint R-CNN Model from Pytorch Vision." Available at https://pytorch.org/vision/stable/models/keypoint_rcnn.html, 2024. (Last accessed in 17/06/2024).

[183] P. Ruiz, "Understanding and visualizing ResNets." Available at https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8, 2018. (Last accessed in 15/09/2024).

[184] PyTorch Website, "Torch Devices." Available at https://pytorch.org/docs/stable/tensor_attributes.html#torch-device, 2024. (Last accessed in 23/06/2024).

[185] M. A. baeldung, "Epoch in Neural Networks." Available at https://www.baeldung.com/cs/epoch-neural-networks#epoch-in-neural-networks, 2024. (Last accessed in 09/09/2024).

[186] G. P. baeldung, "The Difference Between Epoch and Iteration in Neural Networks." Available at https://www.baeldung.com/cs/neural-networks-epoch-vs-iteration#3-batch, 2024. (Last accessed in 09/09/2024).

[187] T. Keary and N. Medleva, "Learning Rate." Available at `https://www.techopedia.com/definition/learning-rate`, 2024. (Last accessed in 09/09/2024).

[188] J. Brownlee, "Understand the Impact of Learning Rate on Neural Network Performance." Available at `https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/`, 2020. (Last accessed in 14/09/2024).

[189] PyTorch Website, "Torch Optimizers." Available at `https://pytorch.org/docs/stable/optim.html`, 2024. (Last accessed in 26/06/2024).

[190] S. Doshi, "Various Optimization Algorithms For Training Neural Network." Available at `https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6`, 2019. (Last accessed in 14/09/2024).

[191] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Jan. 2017.

[192] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," Jan. 2019.

[193] Hilton, Geoffrey, "Lecture 6a: Overview of mini-batch gradient descent." Available at `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`, 2012. (Last accessed in 26/06/2024).

[194] Jason Huang - Cornell University Computational Optimization Open Textbook, "RMSProp." Available at `https://optimization.cbe.cornell.edu/index.php?title=RMSProp#RMSProp`, 2020. (Last accessed in 26/06/2024).

[195] PyTorch Website, "Torch MSELoss Criterion." Available at `https://pytorch.org/docs/stable/generated/torch.nn.MSELoss.html#torch.nn.MSELoss`, 2024. (Last accessed in 26/06/2024).

[196] PyTorch Website, "Torch GaussianNLLLoss Criterion." Available at `https://pytorch.org/docs/stable/generated/torch.nn.GaussianNLLLoss.html`, 2024. (Last accessed in 26/06/2024).

[197] Dipam Vasani - Towards Data Science Website, "This thing called Weight Decay." Available at `https://towardsdatascience.com/this-thing-called-weight-decay-a7cd4bcfccab`, 2024. (Last accessed in 09/07/2024).

[198] PyTorch Website, "Pytorch ReduceLROnPlateau Optimizer." Available at `https://pytorch.org/docs/stable/generated/torch.optim.`

lr_scheduler.ReduceLROnPlateau.html#torch.optim.lr_scheduler.ReduceLROnPlateau, 2024. (Last accessed in 08/07/2024).

[199] PyTorch Website, "Torch Empty Cache." Available at https://pytorch.org/docs/stable/generated/torch.cuda.empty_cache.html, 2024. (Last accessed in 08/07/2024).

[200] Milesi Alexandre, "PyTorch implementation of the U-Net for image semantic segmentation with high quality images." Available at https://github.com/milesial/Pytorch-UNet, 2024. (Last accessed in 22/07/2024).

[201] PyTorch Website, "Pytorch Torch Tensor." Available at https://pytorch.org/docs/stable/tensors.html, 2024. (Last accessed in 22/07/2024).

[202] PyTorch Website, "Pytorch Draw Keypoints Utils Function." Available at https://pytorch.org/vision/stable/generated/torchvision.utils.draw_keypoints.html#torchvision.utils.draw_keypoints, 2024. (Last accessed in 23/07/2024).

[203] R. Ng, "ritchieng/deep-learning-wizard: LLM Section Release," Feb. 2024.

[204] L. Xiang, F. Echtler, C. Kerl, T. Wiedemeyer, Lars, hanyazou, R. Gordon, F. Facioni, laborer2008, R. Wareham, M. Goldhoorn, alberth, gaborpapp, S. Fuchs, jmtatsch, J. Blake, Federico, H. Jungkurth, Y. Mingze, vinouz, D. Coleman, B. Burns, R. Rawat, S. Mokhov, P. Reynolds, P. Viau, M. Fraissinet-Tachet, Ludique, J. Billingham, and Alistair, "libfreenect2: Release 0.2," Apr. 2016.

[205] Y. Yu, "libfreenect2 Github: Missing helper_math.h." Available at https://github.com/OpenKinect/libfreenect2/issues/777/#issuecomment-527648078, 2016. (Last accessed in 07/09/2024).

[206] OpenCV Documentation, "Camera Calibration and 3D Reconstruction: findChessboardCorners function." Available at https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a, 2024. (Last accessed in 14/09/2024).

[207] OpenCV Documentation, "Detection of ChArUco Boards." Available at https://docs.opencv.org/3.4/df/d4a/tutorial_charuco_detection.html, 2024. (Last accessed in 15/09/2024).

[208] OpenCV Documentation, "Camera Calibration and 3D Reconstruction." Available at https://docs.opencv.org/4.10.0/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a, 2024. (Last accessed in 15/09/2024).

[209] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1330–1334, Nov. 2000.

Appendix A

Benchmark results

The benchmark results are displayed in this Appendix. These consist of three metrics (PCKh, OKS and PDJ) which are evaluated using the MPII and Pandora datasets.

A.1 Cross-validation results of the ResNet18 architecture on the MPII dataset

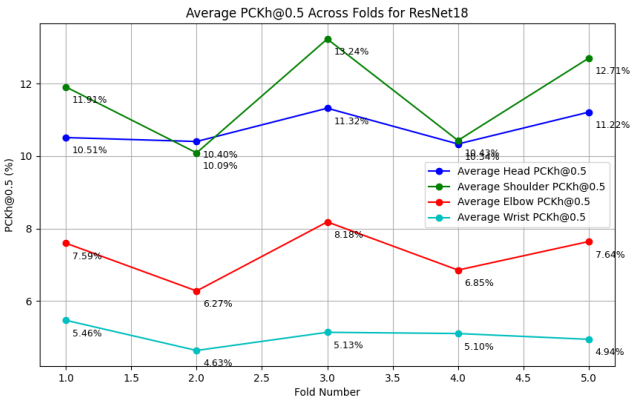


Figure A.1: Average PCKh@0.5 on the MPII dataset across five ResNet18 folds.

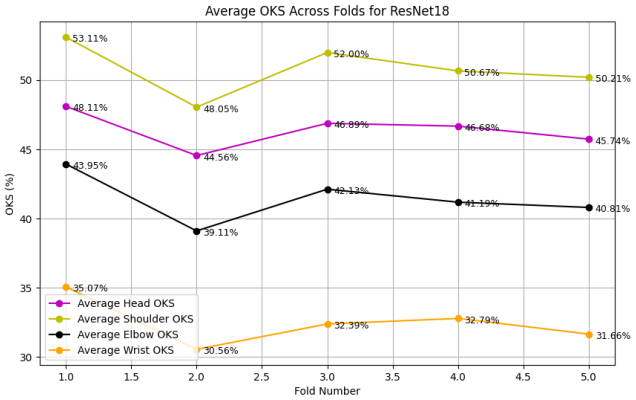


Figure A.2: Average OKS on the MPII dataset across five ResNet18 folds.

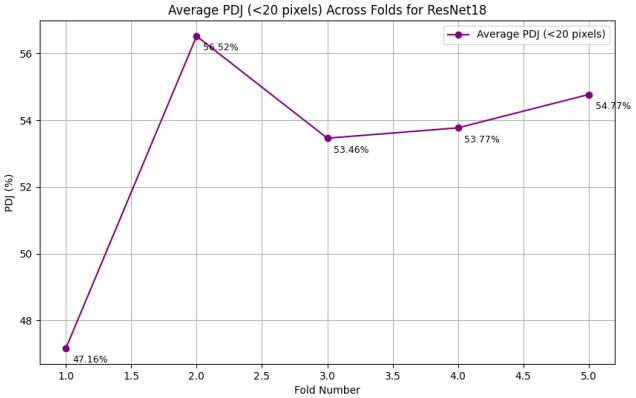


Figure A.3: Average PDJ on the MPII dataset across five ResNet18 folds.

A.2 Cross-validation results of the ResNet18 architecture on the Pandora dataset

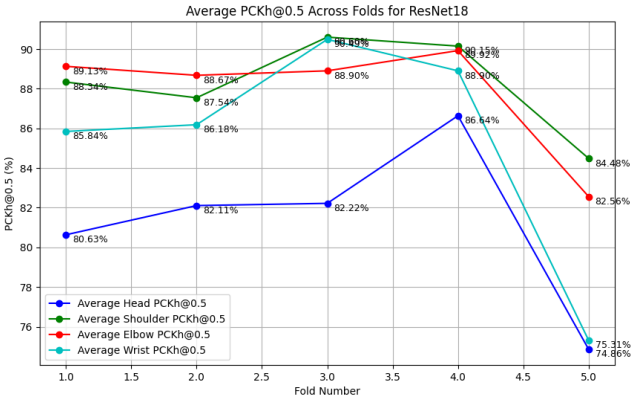


Figure A.4: Average PCKh@0.5 on the Pandora dataset across five ResNet18 folds.

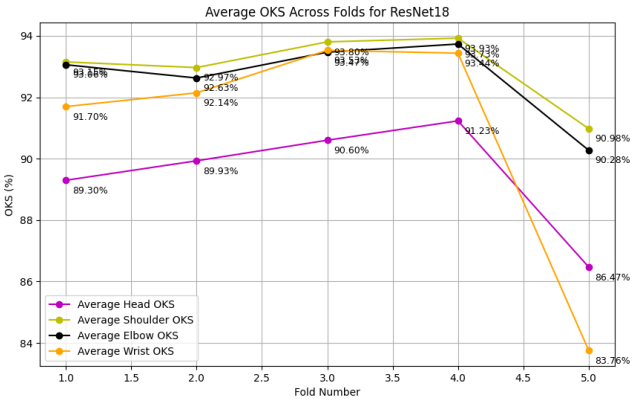


Figure A.5: Average OKS on the Pandora dataset across five ResNet18 folds.

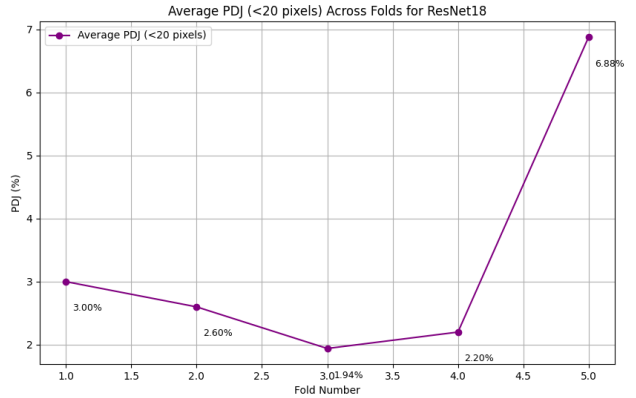


Figure A.6: Average PDJ on the Pandora dataset across five ResNet18 folds.