Deep Learning based 3D Multi-person Human pose estimation from Monocular Vision

Amal El Kaid

Cotutelle PhD THESIS
CIFRE France - Maroc

To obtain the degree of Doctor delivered by

The University of Clermont Auvergne

Doctoral School of Engineering Sciences (EPSI)

And

The University Mohammed V in Rabat

Doctoral School in Information Technology and Engineering Sciences (ST2I)

Defended by

Amal EL KAID

Deep Learning based 3D Multi-person Human pose estimation from Monocular Vision

Defended on 22/12/2023, after the reviewers' opinions, in front of the examination jury:

Pr. Vincent Barra, University of Clermont Auvergne	**Director**
Pr. Karim Baïna, University Mohammed V in Rabat	**Director**
Dr. Denis Brazey, Pryntec Company	**Supervisor**
Dr. Violaine Antoine, University of Clermont Auvergne	**President**
Pr. Renaud Péteri, La Rochelle University	**Reviewer**
Pr. Mounir Ghogho, International University of Rabat	**Reviewer**
Pr. Rachid Oulad Haj Thami, University Mohammed V in Rabat	**Reviewer**

Acknowledgement

I would like to express my deepest gratitude to all those who helped me in the completion of this thesis.

First and foremost, I am deeply indebted to my thesis advisors, Professor Karim Baïna and Professor Vincent Barra. Their exceptional expertise, unwavering commitment, and continuous guidance have been the cornerstone of this work. Beyond academic advice, their administrative support was crucial in making my journey easier and more rewarding. Their encouragement and dedication have been invaluable.

My sincere appreciation also goes to my supervisor at Pryntec company, Dr. Denis Brazey. His encouragement greatly contributed to my professional development and the realization of this thesis. Additionally, I am thankful for the directions and opportunity provided by Jérôme Brossais, the R&D team director.

I am grateful to my colleagues for their valuable insights, suggestions, and support throughout this academic adventure. Their diverse perspectives, constructive feedback, and companionship greatly enriched my thesis and experience.

My gratitude extends to the LIMOS laboratory, as well as Pryntec company, for their exemplary resources and facilities. The dedicated office space, in particular, was essential in facilitating my research and writing.

I sincerely thank the reviewers for their time, effort, and insightful feedback. Their constructive criticism was crucial in enriching the depth and quality of this thesis.

This research was made possible thanks to the generous funding of a CIFRE France/Morocco scholarship (2018/1635), in collaboration with the TEB/Pryntec company. This collaboration is part of a joint initiative between Clermont-Auvergne University in France and Mohammed V University in Rabat, Morocco. I am deeply grateful for the financial support provided by the ANRT (National Agency for Research and Technology) and the CNRST (National Center for Scientific and Technical Research), which significantly contributed to the advancement of knowledge in our field.
I am also deeply grateful to the Mesocentre of Clermont Auvergne University for providing access to their supercomputer facilities, an essential resource for my research.

A heartfelt thank you goes to my family, especially my parents, Mohammed and Khadija, and my sisters, Rajae and Salma, whose love, wisdom, and unwavering encouragement have been my pillars of strength and inspiration.
A special note of gratitude to my husband, Mohamed Alimoussa, for his unwavering support, particularly in creating figures for my publications and manuscript. His presence in my life is a profound blessing.
Furthermore, I extend a heartfelt thank you to my friends Nabila, Narjisse, Chaima, Afaf, and Kawtar for their unwavering support. I sincerely wish them success in all their endeavors.
Lastly, I want to express my profound appreciation to my esteemed colleagues in the lab: Caroline, Trang, José, Aurélien, Soufiane, Vsevolod, Achref, and Ala. Their collaborative spirit, valuable insights, and camaraderie have significantly enhanced my research journey.

Contents

Acknowledgement **2**

Avant-propos **3**

Abstract **11**

Abstract **11**

Résumé **12**

Résumé **13**

1 General Introduction **16**

1.1 Context and Motivation 17

1.2 Company's Requirements 19

1.3 Challenge and Work plan 20

1.4 General Introduction 21

1.5 Organisation of the Thesis 22

1.6 List of publications 23

I Literature Review **26**

2 Deep learning Concepts **27**

2.1 Introduction . . . 28
2.2 Overview of Deep learning . . . 30
2.3 Basic concepts and principles . . . 31
2.4 Deep learning algorithms . . . 34
2.4.1 Learning Methods . . . 40
2.4.2 Acceleration methods for neural network training . . . 40
2.4.3 Deep Learning Model Implementation . . . 42
2.5 Limitations . . . 44
2.6 Deep learning-based human pose estimation . . . 45
2.7 Conclusion . . . 45

3 State of the art of human pose estimation **47**

3.1 Introduction . . . 48
3.2 Applications based on human pose estimation . . . 48
3.2.1 Animation & Gaming . . . 49
3.2.2 Training Robots . . . 50
3.2.3 Intelligent surveillance and security systems . . . 51
3.2.4 Sports analysis . . . 52
3.2.5 Healthcare . . . 53
3.3 Human body modeling . . . 54
3.4 Categories of Posture Estimation Methods . . . 55
3.4.1 Generative approaches . . . 57
3.4.2 Discriminative approaches . . . 58
3.4.3 Hybrid approaches: . . . 60
3.5 Human pose estimation by deep-learning . . . 60
3.5.1 Overview of 2D human pose estimation approaches . . . 61
3.5.2 Single-person pipeline . . . 62

3.5.2.1 Direct Keypoint regression 62
3.5.2.2 Keypoint heatmaps estimation 63
3.5.3 Multiple person pose estimation 65
3.5.3.1 Top-down approaches 66
3.5.3.2 Bottom-up approaches 67
3.5.4 Comparative Analysis of Top-Down and Bottom-Up Approaches in Multi-Person 2D Pose Estimation 68
3.6 Overview of 3D human pose estimation approaches 69
3.6.1 Supervised learning 72
3.6.1.1 Direct regression 73
3.6.1.1.1 Direct regression using only 3D Data . 73
3.6.1.1.2 Fully supervised learning for 3D pose in the wild 74
3.6.1.2 3D poses from 2D joints 75
3.6.1.2.1 Exemplar-based approaches 75
3.6.1.2.2 Deep neural mapping 78
3.6.1.3 3D pose tracking in video 79
3.6.2 Weakly supervised learning 82
3.6.2.1 Generative Adversarial Networks (GAN) 82
3.6.2.2 Multi-view supervision 83
3.6.3 Self-Supervised/Unsupervised Learning 84
3.6.4 Comparative Analysis of Methods Based on Learning Paradigms 85
3.7 Overview of Databases and Evaluation Metrics for Human Pose Estimation 87
3.7.1 Common databases and evaluation metrics for 2D human pose estimation 87
3.7.2 Common databases and evaluation metrics for 3D human pose estimation 90

3.8 Conclusion 94

II 3D Real-time Multi-person pose estimation : Software system design and developement 95

4 3D root-relative person pose estimation 97

4.1 Introduction 98
4.2 Human detection 98
4.2.1 Existing Human detection methods 98
4.2.1.1 Two-stage object detectors 99
4.2.1.2 One-stage object detectors 101
4.2.1.3 Conclusion 103
4.2.2 The human detection method adopted in our system 103
4.3 Multi-person tracking 104
4.3.1 Existing tracking methods 105
4.3.2 The tracking method adopted in our system 106
4.4 2D human pose estimation 107
4.4.1 Existing 2D pose estimation networks 108
4.4.2 The 2D pose estimation method adopted in our system . . 110
4.5 3D pose estimation from 2D joints 112
4.5.1 Existing 3D pose estimation methods 112
4.5.2 The 3D pose estimation method adopted in our system . . 116
4.6 Conclusion 119

5 Proposed approach for 3D absolute pose estimation 120

5.1 Introduction 121
5.2 Existing techniques 123
5.3 Monocular Root depth estimation 126
5.3.1 RootNet network architecture 128

5.3.2 Camera-intrinsic parameters . . . 129
5.4 Two-stage approach 3D absolute pose estimation . . . 130
5.4.1 Approach structure . . . 130
5.4.2 Validation . . . 132
5.4.3 Conclusion . . . 134
5.5 One-Stage approach for 3D absolute pose estimation . . . 134
5.5.1 Approach structure . . . 134
5.5.2 Validation . . . 136
5.5.3 Conclusion . . . 137
5.6 Hybrid approach for 3D absolute pose estimation . . . 137
5.6.1 Conclusion . . . 138
5.7 Geometric method for absolute root keypoint . . . 138
5.8 Conclusion . . . 146

6 Software system implementation pipeline and experimental results 147
6.1 Introduction . . . 148
6.2 Implementing the framework's training and Inference . . . 149
6.2.1 GAST-Net$_{ABS}$ training . . . 149
6.2.2 Pre-processing phase . . . 149
6.2.3 2D human pose estimation . . . 150
6.2.4 3D human pose estimation . . . 152
6.2.5 Determining the focal length of a camera . . . 157
6.2.6 Pose Visualization . . . 159
6.2.7 Taxonomy of the Framework . . . 161
6.2.8 Posture analysis and fall detection . . . 164
6.3 Experiments results . . . 165
6.3.1 Performance of Sequence-wise on the MuPoTS-3D . . . 165
6.3.2 Performance on the Human3.6m . . . 167
6.3.3 End-to-End Real-time system responsiveness . . . 169

6.3.4 Qualitative results . . . 170
6.4 Conclusion . . . 170

III Software delivery **174**

7 Software system industrialization **175**
7.1 Introduction . . . 176
7.2 Main challenges of the system . . . 177
7.2.1 Challenge 1: Simultaneously launch the AI models with other processes . . . 178
7.2.2 Challenge 2: Reuse existing material as much as possible . 180
7.2.3 Challenge 3: Error accumulation in a chain of algorithms . 182
7.3 System hardware and software . . . 184
7.3.1 Hardware . . . 184
7.3.2 Software libraries . . . 189
7.3.2.1 Deep learning libraries for training . . . 190
7.3.2.2 Deep learning libraries for inference . . . 194
7.3.2.2.1 PyTorch to TensorRT conversion . . . 200
7.3.2.2.2 Integration with existing systems . . . 201
7.3.2.2.3 Pre-processing images phase . . . 202
7.3.2.3 Deep learning visualization . . . 203
7.4 Industrial Software system delivery . . . 204
7.4.1 Software delivery . . . 204
7.4.2 DevOps delivery pipeline . . . 205
7.4.3 MLOps delivery pipelines . . . 208
7.5 Conclusion . . . 212
7.6 General Conclusion and Perspectives . . . 214

List of figures **253**

List of tables **255**

Publications **256**

Abstract

Human pose estimation (HPE) has gained significant attention in various scientific and industrial domains, including security and surveillance systems. This thesis focuses on advancing the field of human pose estimation, specifically in the context of 3D pose estimation using deep learning techniques.

The first part of the thesis involves a comprehensive review of recent methods for human pose estimation from monocular images and videos. The review encompasses both 2D and 3D scenarios, with a particular emphasis on deep learning-based approaches. A systematic literature review methodology is employed, where relevant publications are selected based on inclusion/exclusion/quality criteria. These selected papers are analyzed and summarized using a tree-structured taxonomy, grouping them based on pose space representation (2D/3D), number of individuals to be estimated, and learning strategies employed. The review covers approximately 200 eligible papers, providing an overview of different approaches, highlighting their limitations, and comparing their strengths, weaknesses, and computational complexities. Additionally, popular datasets and evaluation metrics used in the field are presented, and current limitations and unsolved problems are identified, laying the groundwork for future research.

The need for accurate 3D poses in real-life applications, particularly in surveillance systems, is emphasized. The challenges of preserving distance and capturing interactions between people are highlighted, as these factors play a crucial role in scenarios such as fall detection systems. The limitations of existing research in

maintaining distance and interactions are acknowledged, underscoring the significance of further investigation in these areas, which serves as the main challenge addressed in this work.

In the second part of the thesis, a novel framework named Root-GAST-Net is proposed to address the challenge of recovering 3D absolute poses using a monocular RGB camera. The framework integrates a human detector, a 2D pose estimator, a 3D root-relative pose reconstructor, and a root depth estimator in a top-down manner, enabling real-time processing. Modifications to the GAST-Net and RootNet networks are made to achieve efficient 3D absolute pose estimation. The proposed Root-GAST-Net system is evaluated on benchmark datasets, including Human3.6M and MuPoTS-3D, demonstrating its effectiveness by outperforming state-of-the-art methods on various metrics. Moreover, the system operates in real-time at 15 frames per second on an Nvidia GeForce GTX 1080 GPU, showcasing its practical applicability.

In the third part of this thesis, the proposed framework is developed into a real system at Pryntec Company. The delivery of the software and development details of the proposal are explained, providing insights into the practical implementation and deployment of the Root-GAST-Net system. The collaboration with Pryntec Company contributes to the real-world application of the research findings and demonstrates the industry relevance of the proposed framework.

This thesis contributes to the progress of 3D human pose estimation by offering a thorough analysis of current methods, proposing a new real-time framework with improved performance, and demonstrating its application in a practical context. The findings of this research provide opportunities for further investigation and advancement in deep learning-based human pose estimation, tackling important issues and creating a path for future research pursuits.

Keywords: 3D pose estimation from monocular camera; multi-person pose estimation; absolute poses; camera-centric coordinates; deep-learning models.

Résumé

L'estimation de la posture humaine (HPE) a suscité une attention considérable dans divers domaines scientifiques et industriels, notamment les systèmes de sécurité et de surveillance. Cette thèse se concentre sur l'avancement du domaine de l'estimation de la posture humaine, plus précisément dans le contexte de l'estimation de la posture en 3D à l'aide de techniques d'apprentissage profond.

La première partie de la thèse comprend une revue complète des méthodes récentes d'estimation de la posture humaine à partir d'images et de vidéos monoculaires. La revue englobe à la fois les scénarios 2D et 3D, en mettant particulièrement l'accent sur les approches basées sur l'apprentissage profond. Une méthodologie de revue de la littérature systématique est utilisée, où les publications pertinentes sont sélectionnées en fonction de critères d'inclusion/exclusion/qualité. Ces articles sélectionnés sont analysés et résumés à l'aide d'une taxonomie structurée en arbre, les regroupant en fonction de la représentation de l'espace de pose (2D/3D), du nombre d'individus à estimer et des stratégies d'apprentissage utilisées. La revue couvre environ 200 articles admissibles, offrant un aperçu des différentes approches, mettant en évidence leurs limitations et comparant leurs forces, faiblesses et complexités computationnelles. De plus, les ensembles de données populaires et les mesures d'évaluation utilisées dans le domaine sont présentés, et les limites actuelles et les problèmes non résolus sont identifiés, jetant les bases de recherches futures.

La nécessité de poses 3D précises dans les applications réelles, en particulier

dans les systèmes de surveillance, est soulignée. Les défis liés à la préservation de la distance et à la capture des interactions entre les personnes sont mis en évidence, car ces facteurs jouent un rôle crucial dans des scénarios tels que les systèmes de détection des chutes. Les limites des recherches existantes concernant la préservation de la distance et des interactions sont reconnues, soulignant ainsi l'importance de poursuivre les investigations dans ces domaines, qui constituent le principal défi abordé dans ce travail.

Dans la deuxième partie de la thèse, un nouveau cadre appelé Root-GAST-Net est proposé pour relever le défi de la récupération des poses absolues en 3D à l'aide d'une caméra RGB monoculaire. Le cadre intègre un détecteur humain, un estimateur de pose 2D, un reconstructeur de pose relative à la racine en 3D et un estimateur de profondeur de la racine de manière descendante, permettant un traitement en temps réel. Des modifications sont apportées aux réseaux GAST-Net et RootNet pour obtenir une estimation efficace de la pose absolue en 3D. Le système proposé, Root-GAST-Net, est évalué sur des ensembles de données de référence, notamment Human3.6M et MuPoTS-3D, démontrant son efficacité en surpassant les méthodes de pointe sur différentes mesures. De plus, le système fonctionne en temps réel à 15 images par seconde sur une carte graphique Nvidia GeForce GTX 1080, démontrant ainsi son applicabilité pratique.

Dans la troisième partie de cette thèse, le cadre proposé est développé en un système réel chez Pryntec Company. La livraison du logiciel et les détails de développement de la proposition sont expliqués, offrant un aperçu de la mise en œuvre pratique et du déploiement du système Root-GAST-Net. La collaboration avec Pryntec Company contribue à l'application concrète des résultats de la recherche et démontre la pertinence industrielle du cadre proposé.

Cette thèse contribue à l'avancement de l'estimation de la posture humaine en 3D en offrant une analyse approfondie des méthodes actuelles, en proposant un nouveau cadre en temps réel avec des performances améliorées et en démontrant son application dans un contexte pratique. Les résultats de cette recherche ouvrent

des opportunités pour des investigations et des avancées supplémentaires dans l'estimation de la posture humaine basée sur l'apprentissage profond, en abordant des problématiques importantes et en ouvrant la voie à de futures recherches.

mots-clés : estimation de pose 3D à partir d'une caméra monoculaire; estimation de pose multi-personnes; poses absolues; coordonnées centrées sur la caméra; modèles d'apprentissage profond.

CHAPTER

General Introduction

1.1 Context and Motivation

Recently, the use of cameras in various fields has greatly transformed our capability to capture and process visual information, leading to the emergence of computer vision as a multidisciplinary field that aims to analyze, interpret, and comprehend images and videos, enabling machines to perceive the visual world around them.

Computer vision has a wide range of applications in several fields. In the area of surveillance, cameras are used for intrusion detection, face recognition, and monitoring of critical environments. In the automotive industry, computer vision is vital for driver assistance systems, pedestrian detection, and autonomous navigation. Additionally, industries such as robotics, medicine, manufacturing, and virtual reality and augmented reality also benefit from the advancements in computer vision.

However, analyzing and interpreting visual data is complex, and 2D vision alone is insufficient in addressing certain problems such as occlusion, viewpoint variability, and three-dimensional scene understanding. Deep learning has played a crucial role in addressing these challenges and advancing the field of computer vision.

Deep learning, particularly deep neural networks, has revolutionized how computer vision systems can learn and extract complex features from visual data. With vast datasets, deep neural networks can learn patterns, allowing them to automatically recognize and understand objects, people, and actions. This approach has achieved remarkable performance on tasks such as object detection, facial recognition, and semantic segmentation.

Thanks to deep learning, computer vision has made significant strides, solving intricate problems that previously seemed insurmountable.

In many scenarios, accurate human pose estimation is of paramount importance. Understanding and interpreting human body movements in a three-dimensional environment holds great value for many applications. 2D human pose estimation was one of the early research goals, representing the positions and movements of

people in a two-dimensional space. However, this approach has significant limitations, particularly concerning occlusions and ambiguities that can occur when multiple people overlap or adopt similar 2D poses.

However, in real-world applications, 2D estimation proved insufficient to solve problems such as occlusions, where certain body parts can be hidden or partially visible. Additionally, 2D poses can be deceptive, as different poses may appear identical. This complicates activity recognition and limits systems' ability to understand and interpret human movements accurately.

To overcome these limitations, research has shifted towards 3D human pose estimation. By incorporating the depth dimension, 3D pose estimation provides more comprehensive and precise information about the position and movements of the human body. It allows for accurate differentiation of similar poses in 2D, facilitating the recognition and interpretation of human activities.

By developing advanced methods based on deep learning, researchers have been able to achieve more accurate and robust 3D human pose estimations. Such estimations are essential in numerous fields of application.

However, estimating 3D pose for multiple individuals in real-world scenarios remains a major challenge. In many applications, such as surveillance and security systems, knowing the distances between individuals is essential for analyzing and recognizing their interactions. This is where absolute pose estimation comes into. The goal of absolute pose estimation is to locate the root joint (the key central point of the person) and estimate its distance from the camera.

Currently, estimating 3D pose for multiple individuals is a complex challenge. Whenever possible, stereo vision is used to calibrate the cameras and determine the precise position of each person from images taken from different viewpoints. However, the use of stereo vision is not always feasible, and it significantly increases the overall costs of the procedures. Moreover, acquiring such data is impractical in real-time systems requiring optimization of the amount of data to capture and process. The other common option is to use RGB-D (Red Green Blue - Depth)

cameras. Similarly, these cameras are not usable in practice because they generally rely on infrared light, which makes them sensitive to unregulated ambient light. Therefore, they are mainly designed for indoor environments. This highlights the gap between scientific literature and real-world requirements. Researchers need to find efficient and practical solutions to estimate 3D pose for multiple individuals respecting real-world constraints.Monocular images or videos may serve as an alternative input for applications based on 3D human pose estimation, offering a simpler hardware setup and reduced costs. Another challenge faced is the need for real-time estimation. Real-time 3D human pose estimation is essential in applications that require fast and precise responses.

It is in this context that the motivation for this thesis arises. The thesis's purpose is to design and develop a deep learning-based method for real-time estimation of multi-human absolute 3D pose from a monocular camera.

1.2 Company's Requirements

TEB Group is a Burgundy-based company established by Louis Bidault in 1978 that specializes in producing and installing video protection solutions. The company has evolved with technical and technological advancements and has over 40 years of experience in the industry. TEB Group's product offerings have expanded to include computerized video recording solutions that cater to the retail, industrial, logistics, and urban security sectors. TEB Group has established technical agencies across France and distributes its solutions through its subsidiary companies and partner network, which has a presence in Europe, the United States, and Brazil. One of TEB Group's companies is Pryntec, founded in May 2001.

Pryntec is a simplified joint stock company, founded in May 2001 and is a subsidiary of TEB. It is responsible for researching and developing electronic and computer systems for video surveillance. Pryntec consists of a team of about 15 employees with two teams, one focused on electronic solutions and the other on

software solutions. The company specializes in developing both hardware and software products.

As part of the CIFRE France/Morocco thesis program, I had the opportunity to conduct an applied thesis with a company that prioritizes security and surveillance initiatives. The CIFRE program (Convention de formation industrielle par la recherche (Industrial Training through Research Convention)) is a French initiative that enables companies to sponsor PhD students to work on research projects that benefit both the company and the student. The company's focus areas include monitoring isolated workers for potential accidents and controlling access through door positions. To achieve more accurate posture analysis, the company aims to analyze 3D poses instead of 2D poses, which can be unreliable. My thesis research is centered around the estimation of 3D poses to help the company achieve its objectives in this area.

1.3 Challenge and Work plan

In a thesis that integrates academic and corporate aspects, the main difficulty is finding a balance between the scientific rigor required for a successful thesis and the practical considerations and objectives of the company. This project was carried out in a company that specializes in surveillance cameras and image processing systems, which necessitated close collaboration and regular monthly meetings with both academic and industrial supervisors. The goal was to ensure that decisions were made and tasks were executed in a way that met both academic and corporate objectives. The following sections outline the work plan in the form of work packages:

- WP 0: Conducting a state-of-the-art review.
- WP 1: Acquisition and management of data.
- WP 2: Developing and implementing the network and training code.

- WP 3: Conducting experiments.
- WP 4: Comparing the results with existing literature.
- WP 5: Preparing and publishing the findings.

1.4 General Introduction

This thesis delves into the field of human pose estimation, a subject that has garnered significant attention and research in recent years. The advancements in this area have led to the emergence of various methods for accurately determining human body postures, opening up new possibilities for image processing and video analysis applications.

The primary objective of this thesis was to tackle the challenge of absolute estimation of multi-person 3D poses using RGB monocular images. To achieve this, we developed a comprehensive software pipeline that addresses both application-specific and industrial requirements, ensuring its practicality and usability in real-world scenarios.

Our approach begins by detecting all the individuals present in the image. Each person is assigned a unique identifier and tracked throughout the sequence as long as they are present. For each frame and for each person, we first predict the 2D coordinates of all the joints and then utilize a grouping process to predict 3D skeletons by leveraging temporal information. Our proposed approach relies on absolute estimation of the postures, accurately determining the true position of each person in the room relative to the camera. This necessitates training a network to estimate the depth of one of the joint points.

Throughout this thesis, we have enhanced several components of the pipeline to improve the precision of our estimates, which will be discussed in detail. Our approach has demonstrated favorable results when compared to existing literature. The primary contribution of this thesis is the development of a method for 3D

skeleton estimation that can be applied to various scenarios with limited prior knowledge about the scene and the individuals within it. Notably, we have created a system that does not rely on body markers or similar equipment, instead utilizing only a sequence of RGB images from a standard camera as input. This system has potential applications in domains such as surveillance and fall detection, particularly in recognizing specific actions.

1.5 Organisation of the Thesis

The thesis is structured as shown in Figure 1.1. It consists of a general introduction followed by three main parts: the literature review, the software system design and development work, and software delivery.

The first chapter is a general introduction provides a brief overview of the research study, its context, and motivation from both academic and industrial perspectives. This chapter also outlines the structure of this thesis organization as well as the list of publications.

The first part is divided into two main chapters:

Chapter 2: provides an overview of the fundamental principles and concepts of deep learning.

Chapter 3: presents an up-to-date analysis of the latest methods and approaches used in 3D human pose estimation.

The second part of this thesis focuses on the design and development of a real-time multi-person 3D pose estimation software system. It is divided into three chapters:

Chapter 4: concentrates on the conceptual design of the software framework for real-time multi-person 3D pose estimation. It provides an in-depth discussion of the models utilized at various stages of the process to predict 3D root-relative person pose estimation.

Chapters 5: delves into the three proposed architectures for obtaining 3D absolute pose estimation. Each architecture is presented and analyzed in detail, highlighting their respective strengths and limitations.

Chapter 6: focuses on the implementation pipeline of the software system and presents the experimental results that validate our approach, comparing it to existing literature. The results demonstrate the effectiveness and accuracy of our proposed method.

The last part of the thesis is dedicated to Software Delivery and consists of one chapter:

Chapter 7: discusses the industrialization of the software system. It addresses the practical aspects of deploying the system in real-world scenarios, considering factors such as scalability, performance optimization, and integration with existing software infrastructure.

This thesis covers the entire spectrum of studying and developing a real-time multi-person 3D pose estimation software system, from conceptual design to implementation and industrialization.

1.6 List of publications

- El Kaid, A., Baïna, K., Baina, J., Barra, V. Real-world case study of a deep learning enhanced Elderly Person Fall Video-Detection System. International Conference on Computer vision Theory and Apllications (VISAPP'23), 19 - 21 February, 2023, Lisbon, Portugal.
- El Kaid, A., Baïna, K., Baïna, J.. Improving Elderly person fall video-detection algorithm: False positive reduction technique. International Conference on Complexity Analysis of Industrial Systems and Advanced Modeling (CAISAM'19), 25- 27 April,2019, Benguerir, Morocco.

- El Kaid, A., Baïna, K., Baïna, J. Reduce false positive alerts for elderly person fall video-detection algorithm by convolutional neural network model, The Second International Conference on Intelligent Computing in Data Sciences (ICDS'18), 03- 05 October, 2018, Fez, Morocco.
- El Kaid, A., Brazey, D., Barra, V., Baïna, K. (2022). Top-down system for multi-person 3D absolute pose estimation from monocular videos. Sensors, 22(11), 4109.
- El Kaid, A. Baïna, K. (2023). A Systematic Review of Recent Deep Learning Approaches for 3D Human Pose. Journal of Imaging, 9(12), 275.

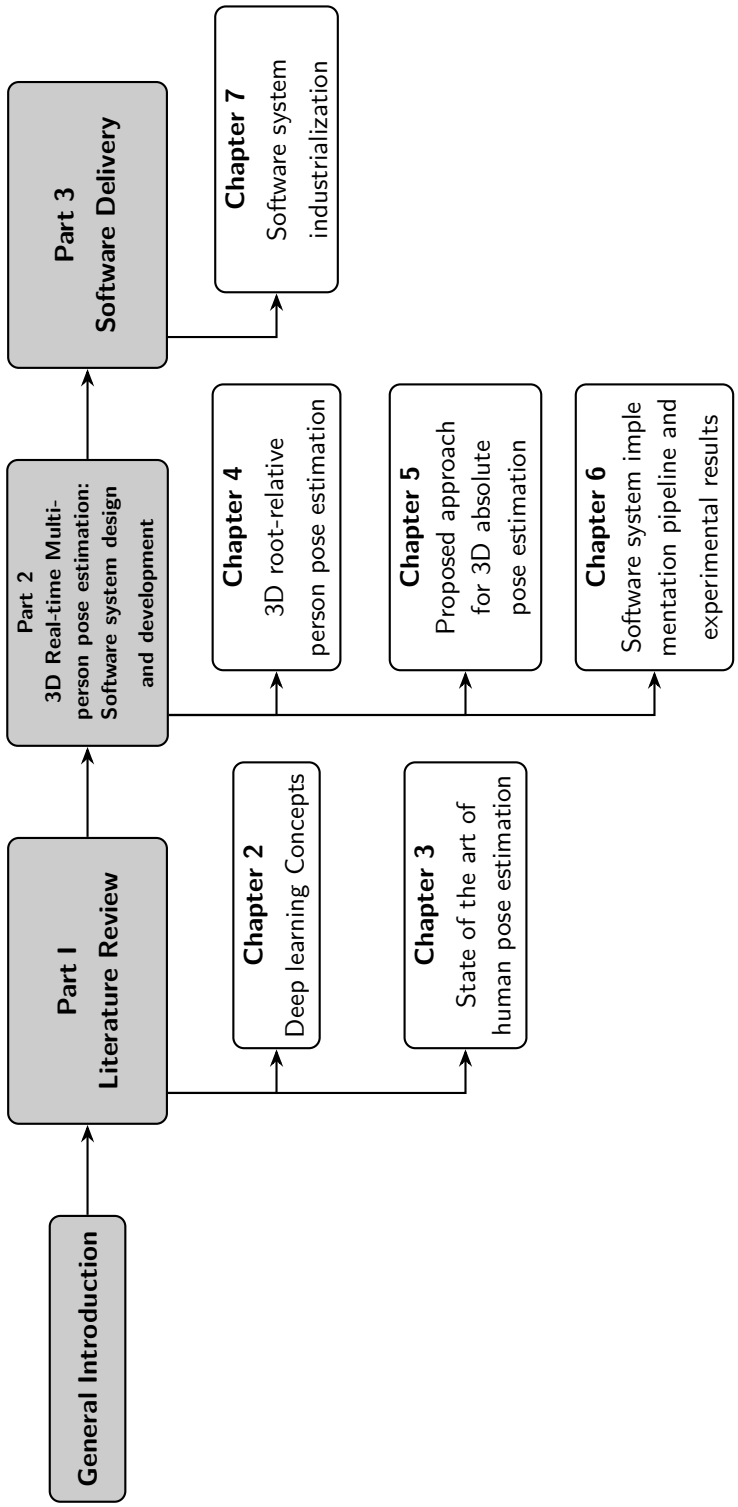

Figure 1.1: Thesis structure

Part I

Literature Review

CHAPTER

2

Deep learning Concepts

2.1 Introduction

Artificial intelligence has become an important factor in the industrial revolution. It aims to understand and reproduce the fundamental principles of human intelligence and to improve human life through the use of technology. Currently, Deep Learning Technology, which is very popular, plays a central role in the last achievements and research of artificial intelligence.

Indeed, Deep learning (DL) has become a prevalent technology in our daily life, often in ways we may not even be aware of. For example, Facebook uses deep learning algorithms to automatically recognize and tag people in photos. Digital assistants such as Siri, Cortana, and Alexa employ natural language processing and speech recognition created by deep learning algorithms to understand and respond to user requests. Skype also employs deep learning algorithms to offer real-time translations for conversations with international contacts. Additionally, email service providers utilize deep learning algorithms to recognize and eliminate spam emails. The uses of deep learning are numerous and diverse.

Deep learning has also led to the development of complex and advanced practical applications in various fields, as table 2.1 presents:

Additionally, deep learning has many other applications in various domains that are not listed in the table . The potential for deep learning to transform the way we process and analyze data has garnered significant interest from both academia and industry.

This thesis also presents research work based on deep learning approaches in an industrial setting. Before delving into the specifics of this work, it is essential to first understand the concept of deep learning. Deep learning is a sub-field of machine learning that involves the use of artificial neural networks with multiple layers, or "deep" neural networks, to learn hierarchical representations of data. These deep neural networks can adapt and improve their performance over time by adjusting the weights and biases of the connections between the nodes in each

Field	DL can be used to ...
Self-driving cars	enable self-driving cars to distinguish between red and green lights, identify pedestrians and other objects on the side of the road, and even measure the distance between two cars.
Personal assistants	create chatbots and service bots that can interact with customers and provide intelligent answers to increasingly complex voice and text queries.
Security and surveillance	enhance security measures, such as facial recognition systems, which are already widely implemented in airports to facilitate seamless, paperless checking.
Healthcare	analyze medical images, predict patient outcomes, and identify potential outbreaks of diseases.
Finance	analyze market trends, identify fraudulent transactions, and make investment recommendations.
Marketing	analyze customer data and make personalized recommendations for products or services.
Education	personalize learning experiences for students, adapting to their strengths and weaknesses.
Law enforcement	analyze video footage from security cameras to identify criminal activity and assist in investigations.
Transportation	optimize traffic flow and improve public transportation systems.
Agriculture	analyze aerial images of crops to identify pests, diseases, and areas of unhealthy growth.
Environmental monitoring	analyze satellite images to identify changes in land use and detect environmental disasters such as oil spills.
Manufacturing	improve the efficiency of manufacturing processes by predicting equipment failures and optimizing supply chain management.
Language generation	generate text based on a given prompt or context, such as writing summaries, providing information, or answering questions such as ChatGPT or Bard.

Table 2.1: Applications of Deep Learning in various industries

layer.

Machine learning is a subset of artificial intelligence that focuses on "learning" rather than computer programming. It is a broad field that deals with the study and construction of algorithms that can learn from and make predictions on data. Artificial neural networks are a type of machine learning algorithm inspired by the structure and function of the human brain. These algorithms consist of interconnected nodes, or "neurons," that can learn patterns and features in data by adjusting the weights and biases of the connections between the nodes

Artificial intelligence (AI) is the field of computer science and engineering focused on the creation of intelligent machines that work and act like humans. At its core, AI involves the development of algorithms and systems that can analyze, reason, and make decisions in a way that is similar to human cognition. The ultimate goal of AI research is to create systems that are able to perform tasks that would normally require human intelligence, such as learning, problem-solving, and decision-making.

2.2 Overview of Deep learning

Neural networks are a type of machine learning algorithm, and deep learning is a sub-field of machine learning that uses deep neural networks to learn hierarchical representations of data.

Deep learning is a sub-field of machine learning that has gained significant attention in recent years due to its successes in a wide range of applications, including image and speech recognition, natural language processing, and autonomous vehicles. The concept of artificial neural networks, which are at the heart of deep learning, was first proposed in 1943 by Warren McCulloch and Walter Pitts. In the following decades, training algorithms for neural networks were developed, including the perceptron learning rule and the back-propagation algorithm. However, the field of neural networks experienced a period of stagnation during the 1980s and 1990s

due to the limited computational power available at that time and the lack of large databases for learning.

A breakthrough in the field came in 2006 when a paper by Geoffrey Hinton and his team demonstrated the effectiveness of deep learning for image recognition. This sparked a resurgence of interest in deep learning, and in the 2010s it became a hot topic in the field of machine learning. Some of the key figures in the development and advancement of deep learning include Hinton, Yann LeCun, Andrew Ng, and Fei-Fei Li.

Deep learning has been successfully applied to a variety of tasks and has also led to the development of complex and advanced practical applications in various fields.

Additionally, deep learning has many other applications in various domains that are not listed above. The potential for deep learning to transform the way we process and analyze data has garnered significant interest from both academia and industry.

2.3 Basic concepts and principles

Deep learning (DL), a subset of machine learning, leverages deep artificial neural networks (ANNs), algorithms inspired by the structure and function of the human brain. In this section, we delve into the core ideas of machine learning and ANNs, presenting the fundamentals of DL, including the architecture and operation of ANNs.

Machine learning, a component of AI, focuses on developing algorithms that enable machines to learn from data, detect patterns, and make predictions. These algorithms can be categorized into supervised and unsupervised learning.

Supervised learning algorithms require labeled data for training, while unsupervised learning algorithms can learn from unlabeled data. Machine learning is a statistical technique that uses algorithms to train data-driven models and use

those models to make predictions and decisions. It requires carefully selected and well-prepared data. Commonly used models include linear regression, decision trees, random forest, support vector methods, and even shallow neural networks.
Deep learning is a machine learning technology that uses deep neural networks with many layers to process data. These networks can learn directly from raw data and independently extract hierarchical features for pattern recognition, classification, segmentation, and content generation tasks. Deep learning uses different types of neural networks, such as convolutional neural networks, recurrent neural networks, generative adversarial networks, and more recently, transformers.
The main difference between deep learning and machine learning is the complexity of the data and the processing power. Machine learning extracts features from data through manual tasks such as feature engineering and builds models from them. Deep learning, on the other hand, uses artificial neural networks with many hidden layers to automatically extract features from data and uses deep neural networks that can learn from large amounts of raw data, allowing them to identify more complex relationships between features. Deep learning approaches allow computers to learn and recognize patterns automatically by simulating the structure and operation of the neural networks in the human brain. These networks are composed of neurons, layers, and weights. The neuron is a fundamental component that takes in input and generates output. In a conventional neural network, each neuron is linked to other neurons, and they are grouped into layers. These layers are divided into three types: the input layer, the output layer, and the hidden layer. The input layer takes in external data, such as image pixel values or text character encoding, which is then used to process the input and extract useful features. The output layer produces the results of the learning task, such as object categories in images or sentiment classification of text. The hidden layer is responsible for processing the input data and extracting and combining the relevant features. Each neuron has one or more inputs, and each input weights to specify its level of influence on the neuron's output 2.1. All the inputs and their respective weights are multiplied

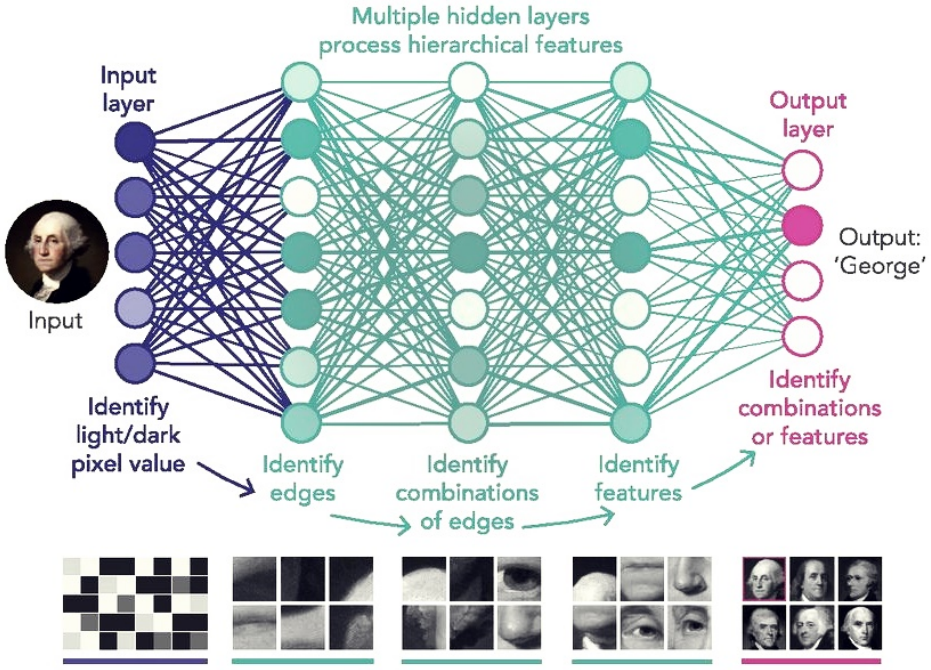

Figure 2.1: Typical Deep Neural Network architecture

together and then summed, before being used to compute the output through a nonlinear activation function, such as a sigmoid or ReLU function. Nonlinear activation functions are required for neural networks to learn more complex patterns. Training an artificial neural network involves learning it to accurately map input data to the corresponding output so that the network gradually adjusts its weights based on this feedback. Specifically, data is fed into the input layer and then processed by several hidden layers to produce an output prediction. The prediction is then compared to the correct response, and the difference is used to adjust the weights of the network. The goal is to reduce the error as much as possible and obtain a well-trained model. This process is facilitated by the back-propagation algorithm, which leverages gradient descent of the loss function to compute the discrepancy between the predicted and actual values for each weight. The gradient

represents the direction of the steepest increase in the loss function, so by moving in the opposite direction, the weights can be updated to decrease the loss. The learning rate determines the step size of the weight updates. The process is iteratively repeated until the network's predictions are sufficiently accurate, resulting in a successfully trained model.

Testing is crucial in controlling overfitting in machine learning. A separate test set is used to assess the model's ability to accurately generalize to new data, as it contains examples that the model has never encountered during training. By evaluating the model's performance on the test set after each training iteration or epoch, it is possible to determine if the model is overfitting or generalizing well to unseen data. Improvement in the model's performance on the test set indicates good generalization, while degradation of the performance on the test set while training continues to improve suggests overfitting, which requires appropriate steps to address it.

2.4 Deep learning algorithms

The objective of the DL task is to extract useful information from input data and convert it into meaningful output. This task is widely used in computer vision, natural language processing, and speech recognition.

The field of computer vision encompasses a wide variety of tasks, including but not limited to detection, classification, recognition, segmentation and regression.
- The detection task is to locate and mark the position of a particular object in an image or video. It aims to find a specific class of objects in an image or video and label their position and size. For example, a detection system might be able to identify and label all the cars in a parking lot (see Figure 2.2).
- The classification task aims to classify input data into predefined categories. A typical example is image classification. For example, if an image is required to be classified as either "Donald," "Goofy," or "Tweety," the classification task would

classify this image into the respective category (see Figure 2.2).
- The recognition task is to identify a particular object or face from a set of objects or faces. The recognition task is usually performed using a classifier. This means using a trained model to classify objects in an image or video. The classifier extracts features from the input image and compares them to known objects to assign the most likely class. For example, a recognition system might be able to identify a particular person from a crowd of people (see Figure 2.1).
- The segmentation is the task of dividing an image or video into numerous segments or regions for the purpose of identifying and isolating particular objects or boundaries. It is comprised of semantic segmentation, which assigns each pixel to a specific class or category, and instance segmentation, which differentiates between separate instances of the same class. For instance, in a street view, a semantic segmentation algorithm might label all pixels as belonging to roads, buildings, cars, or pedestrians, while an instance segmentation algorithm would distinguish between individual cars in that same scene (see Figure 2.2).
- The regression task, on the other hand, aims to estimate a continuous value or numerical output from the input data. A typical example is the estimation of prices. For example, given information about the year, mileage, body type, and color of a certain used car, the estimation task estimates the price of that car (see Figure 2.3).

Furthermore, Deep learning is a machine learning technique that uses multi-layer neural networks. Convolutional neural networks (CNN) and recurrent neural networks (RNN) are the two main types of deep learning algorithms.
Convolutional neural networks (CNN) (Figure 2.4) are primarily used for image processing tasks. A CNN consists of a set of convolutional layers, pooling layers, and flattened layers to extract features in an image. The algorithm process involves using compact filters on an image to conduct convolutional operations, which generate feature maps. The filters slide over the image's smaller regions and apply convolution operations in these regions. This generates a feature map

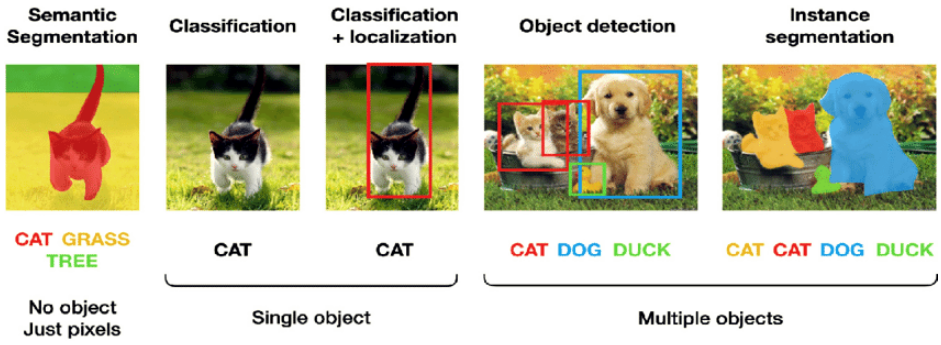

Figure 2.2: Comparison between semantic segmentation, classification, object detection and instance segmentation

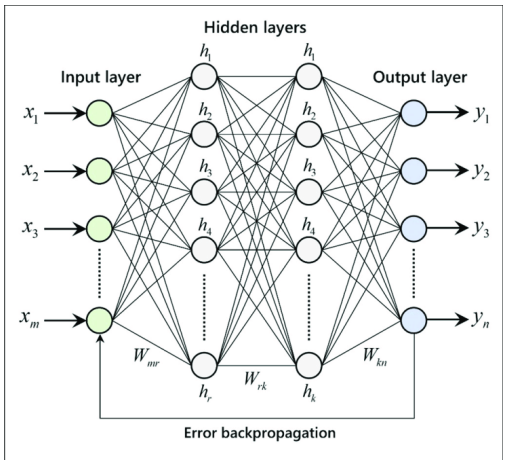

Figure 2.3: Regression Analysis Using Artificial Neural Networks

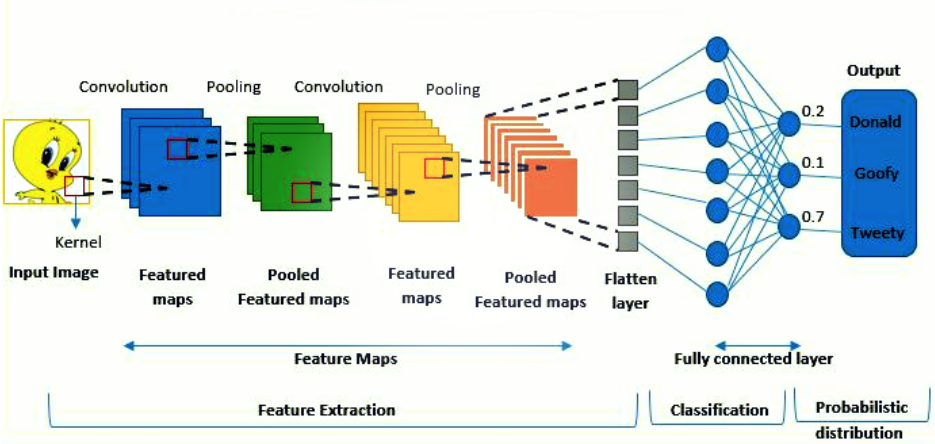

Figure 2.4: Convolutional Neural Network Architecture for Classification

that corresponds to the image features extracted by that filter. Then the pooling layer compresses and decreases the size of the feature map to reduce computational complexity and passes it on to the next layer. Finally, the feature map is converted into a flat vector to perform tasks such as class classification [1].

Recurrent Neural Networks (RNN) [2, 3] are a class of artificial neural networks used to process time series data and sequential data. These data structures have a specific arrangement where the previous output influences the next input, such as in natural language processing. The key feature of RNNs is their capacity to memorize past information and use it to make decisions about future data. As a result, RNNs are used for various tasks, including sequence prediction, text and audio signal processing, image sequence analysis, language translation, speech recognition, and sentiment analysis.

RNNs [4] (see Figure 2.5) are usually composed of recursive cells that receive the current input and the hidden state of the previous cell as input. The hidden state represents the information stored in the cell and is used to inform decisions made by the cell in the next step. Backpropagation techniques are commonly used in combination with RNNs to train the network using training data. LSTM and GRU

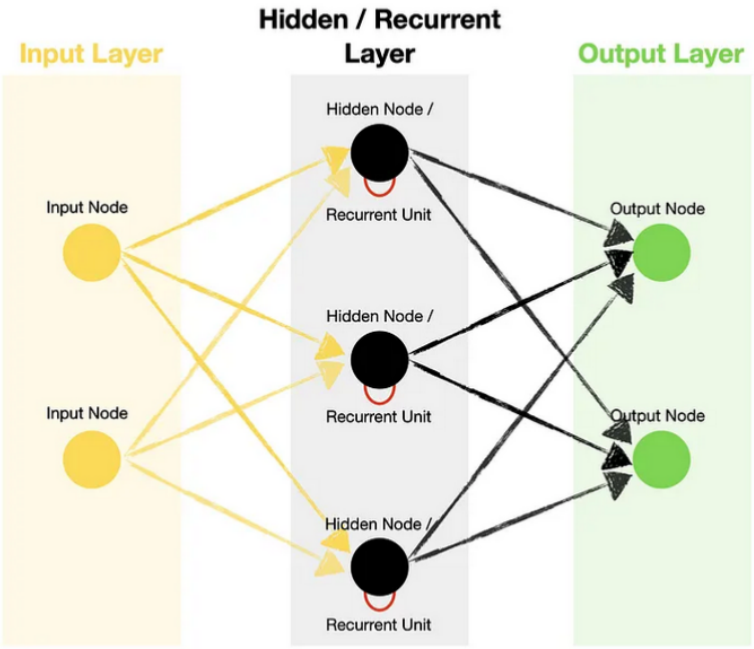

Figure 2.5: Recurrent Neural Network Architecture [5]

are more advanced types of RNNs, capable of learning long-term dependencies, addressing the gradient vanishing issue, and achieving excellent performance in speech recognition and natural language processing tasks.

Long-short-term memory (LSTM) networks [4, 6] is an extension of recurrent neural networks that allow to remember important information over a long period. LSTM networks were designed to address a common problem with RNNs, namely their tendency to forget important information about long sequences. The units of an LSTM are used to build layers of recurrent neural networks, which are often referred to as LSTM networks. LSTMs store information in gated cellular memory, which decides whether to store or discard information based on its importance, which is learned by the algorithm over time.

GRUs (Gated Recurrent Units) [7, 8] are the newer generation of recurrent neural networks and are pretty similar to an LSTM. However, one of the main differences

between LSTMs and GRUs is that GRUs use two gates (reset gate and update gate) to control the information that is stored and transmitted in the cell. Unlike LSTMs, GRUs do not have an explicit forget gate, which means that they can potentially have difficulty forgetting unnecessary information in certain situations. Another important difference is that GRUs have fewer parameters than LSTMs, making them computationally lighter and therefore faster to train. However, LSTMs tend to be more powerful than GRUs for modeling complex data sequences, as they can store information longer and more accurately due to their explicit forgetting gate. LSTMs have also been more widely studied and used in deep learning research than GRUs.

Similar to RNNs, Temporal Convolutional Networks (TCNs) [9] are another type of neural network used for processing sequential input. TCNs are created utilizing 1D convolutional layers that work with the input data's temporal dimension. TCNs can parallelize computation across the entire sequence and learn long-term dependencies using dilated convolutions, in contrast to typical RNNs, which process sequences sequentially and have vanishing gradient problems.

Auto-encoder [10] is another type of artificial neural network that learns to represent the input data using a hidden layer that reduces the dimensionality of the input. It consists of two main parts: the encoder and the decoder. The encoder takes the input data and transforms it into a reduced dimensional representation, called the latent code. The decoder then takes this latent code and transforms it into an output that must match the original input data. The objective of the auto-encoder is to minimize the reconstruction error between the decoder output and the original input data. Unlike CNNs and RNNs, auto-encoders can be used to learn latent representations of unstructured data, such as images or text, without requiring additional annotations.

Auto-encoders are widely used in data compression, dimensionality reduction, anomaly detection, and data generation applications. They are also used as a preprocessing step to improve the performance of classification and object recognition

models.

2.4.1 Learning Methods

Machine learning involves a range of algorithms that can be used for training purposes. One type is supervised learning, which requires labeled data for training. Another type is semi-supervised learning, which can work with both labeled and unlabeled data. Unsupervised learning algorithms, on the other hand, do not require labeled data for training. Finally, reinforcement learning algorithms enable agents to learn from trial and error through interactions with their environment. Table 2.2 provides detailed descriptions for each of the different types.

2.4.2 Acceleration methods for neural network training

Training the neural network often requires a lot of computing power. However, the training can be accelerated by various techniques.

Although a single GPU can handle a large number of computing tasks, when processing large models, multiple GPUs can provide more computing power and can significantly shorten the training time. Indeed, when training a neural network, large amounts of data need to be run through the network to optimize the weights, so it is recommended to use multiple GPUs in parallel to be able to divide the data between the GPUs and process them simultaneously. This results in faster data processing and faster weight optimization.

Another method to speed up neural network training is overlap pooling. In pooling, a matrix is divided into smaller parts to shrink and simplify the network. Overlapping pooling extends this method by dividing the matrix into overlapping regions. This introduces more information into the network as the individual areas overlap. This can help improve the accuracy of the neural network.

The use of these methods has become standard in many deep learning programs, allowing researchers and engineers to develop more complex and accurate neural

Learning Method	Description
Supervised Learning	Machine learning method that predicts new data output through the training data that is known as input and output to predict new data output. The labels of training data are known. The model is classified or returned to the analysis of how these labels learn how these unsigned data are classified. The following are some common methods in supervision learning: Linear regression, Logic regression, Linear discrimination analysis, and Decision-based (classification and regression tree).
Reinforcement Learning	Machine learning method learns the best strategy by taking action in an uncertain environment. In reinforcement learning, the model adjusts the strategy based on the feedback signal to maximize the expected reward. Q-Learning, SARSA, and Deep Reinforcement Learning are some common methods in reinforcement learning. They have applications in game AI, robot control, self-driving cars, and more.
Semi-supervised Learning	Algorithm that performs supervised learning in situations where only a portion of the input data is labeled. The algorithm uses both labeled and unlabeled data to train the model. Semi-supervised learning has the advantage of reducing the cost of data collection and labeling.
Unsupervised learning	Algorithm that learns from input variables without being given an output variable. A typical example is the clustering algorithm. A clustering algorithm is an algorithm that categorizes input data into similar groups.

Table 2.2: Learning Methods

network models.

On the other hand, deep learning models often require large amounts of data to train effectively, but obtaining sufficient and accurate labeled data can be a challenge. Data augmentation is one of the techniques used to deal with this issue. It involves generating additional training examples by performing various transformations on the existing data, thereby increasing the diversity and size of the dataset. This can help prevent over-fitting and improve the performance of the DL model, as well

as its ability to generalize to new, unseen examples, and to increase its robustness. There are different ways to perform data augmentation, depending on the nature of the data and the problem being addressed. This can involve:

- Geometric transformation: transformations such as rotating an image by a certain angle, zooming, flipping it horizontally or vertically, scaling it up or down, cropping out a portion of the image, and translating an image to generate pictures of different perspectives and sizes.
- Adding Noise: adding random noise such as Gaussian noise, salt and pepper noise, etc. to an image.
- Changing Color and Contrast: changing the color or contrast of an image.
- Color translation: adjusting the color including brightness, contrast, color, saturation, etc. of the picture to produce different color styles.

2.4.3 Deep Learning Model Implementation

To implement a deep learning model, several steps need to be followed. First, it is important to determine the problem that the model will address and gather relevant data for training and testing. The data needs to be prepared by cleaning it from noise and outliers, formatting it in a way that is appropriate for deep learning algorithms, preprocessing it to obtain meaningful features, and dividing it into training, validation, and test samples.

Next, the appropriate deep learning architecture must be built based on the type of problem being addressed and the characteristics of the data. This step requires selecting the model architecture, including the choice of the number of layers, layer types, and activation functions, as well as the parameters of each layer. This may require multiple experiments to determine the optimal model configuration.

Once the architecture is defined, the model needs to be trained using the training data. This involves passing the data through the model during training with a

specific batch size, calculating losses and gradients, and then updating the model weights according to the gradient using a specific optimization algorithm (e.g., stochastic gradient descent). Furthermore, it is essential to track the model's performance on validation data to prevent overfitting and adjust hyperparameters as needed during the training process.

To avoid overfitting, it is common practice to apply regularization techniques such as dropout. Dropout is another common method in the image processing field that solves the problem of overfitting and optimizes the training of a Deep Learning model. This method is considered a regularization technique [cite source]. The term "dropout" means the removal of certain neurons in the layers of a Deep Learning model, randomly during training, as well as all the incoming and outgoing connections of these neurons. The objective is to make the model less dependent on certain neurons and therefore, better generalize with new data. It is important to note that dropout is only active during the training of the model. During tests, all neurons remain active.

Finally, after the model is trained and validated, it needs to be evaluated using the test dataset. This evaluation provides an estimate of the model's performance on data that it has not used during training, to determine if the model can generalize to new data or scenarios using various metrics, as well as visualizing the results. If a model doesn't perform well enough on the test data, it may be necessary to fine-tune the model based on the evaluation results, incorporating new data or optimizing existing parameters. In fact, fine-tuning involves adjusting the upper layers or hyperparameters of a pre-trained model for a new dataset, while keeping the lower layers frozen to retain the previously learned knowledge. This allows the model to adapt to the new dataset while still benefitting from the pre-training. Fine-tuning is a specific type of transfer learning, which involves taking a pre-trained model and completing its training for a similar but more specific task. This can potentially reduce the time, cost, and data requirements. Retraining can involve changing hyperparameters, model architecture, or training on new data.

2.5 Limitations

Deep learning is a technology that shows great promise, having achieved significant performance in many machine learning problems, such as image recognition and natural language processing. However, caution and precautionary measures are required due to several limitations and challenges.

One of the main limitations of deep learning is over-fitting, where a model over-trains on training data and becomes less adaptable when presented with new data. This can result in high training accuracy but low generalizability on new, unseen data. To address this issue, techniques such as adjusting hyperparameters and cross-validation might be applied.

Deep learning also heavily relies on large amounts of high-quality data for training, which can be difficult and time-consuming to collect and create. Additionally, the dataset may be biased. Transition learning and unsupervised learning can be used to address this issue.

Another challenge is the need for significant computational resources. Large models and data require high-performance hardware, such as fast graphics cards or distributed processing systems.

Interpretability of deep learning results is also a challenge. While deep neural networks can produce very good predictions, it is difficult to explain how they arrived at these predictions. This is especially important in situations where decisions are made based on critical data, such as medical diagnoses. Furthermore, deep learning's large computational requirements can make it difficult to deploy in resource-constrained environments, such as mobile devices or embedded systems. Therefore, the use of deep learning must be carefully considered within the scope of its limitations and challenges in order to maximize its benefits and minimize potential risks.

2.6 Deep learning-based human pose estimation

Building on the successes of deep learning in diverse domains, human pose estimation using deep learning methods has gained significant attention. Researchers have exploited the power of convolutional neural networks (CNNs), recurrent neural networks (RNNs), and other specialized architectures to overcome the challenges of occlusion, scale variation, and complex articulations. These efforts have led to the ability to accurately infer body postures and movements from visual data, sparking innovation in sectors like sports science, surveillance, and augmented reality. We will delve deeper into the relevant literature and explore these developments in greater detail in the next chapter.

2.7 Conclusion

In this chapter, we have covered the successful applications of deep learning in various domains such as healthcare, finance, and security. One reason for its success is its ability to automatically learn relevant features from input data without manual feature engineering. We have also discussed different architectures like CNNs, RNNs, and auto-encoders that have been proposed for specific tasks. However, deep learning requires computationally intensive training and large amounts of data. We have also discussed how advancements in pre-trained models and hardware have sped up the training process, employing techniques like model parallelism, data parallelism, and distributed training. Also, some challenges have been identified such as overfitting, the need for large amounts of labeled data, and the lack of interpretability of results.

In the future, integrating deep learning with other technologies such as augmented reality and virtual reality is a potential growth area. Another area of growth is the development of more explainable deep learning models that can provide insights into decision-making. In addition, there is growing interest in developing

more efficient deep learning algorithms that require fewer data and computational resources. These advances can further expand the applications of deep learning in various domains and make it more accessible to a wider range of users.

CHAPTER

State of the art of human pose estimation

3.1 Introduction

Pose estimation (PE) is a popular task in Computer Vision (CV) with numerous applications in understanding human behavior and human-computer interaction. The task involves predicting and tracking the positions of key points of a person or object in 2D or 3D space from an image or sequences of images. It integrates knowledge from machine learning, deep learning, mathematics, and physics. While for objects, the key points could be corners or other significant features, for humans, they represent joints such as wrists, ankles, and shoulders. The prediction of the position of these objects is known as **rigid pose estimation**.

Pose estimation is a challenging task due to the high degree of variation in object and human poses, lighting conditions, occlusions, and image noise. To overcome these challenges, various approaches have been proposed, including traditional computer vision techniques such as template matching and feature-based methods, as well as deep learning-based methods such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). In recent years, there has been a growing interest in using 3D data for pose estimation, as it provides additional information about the object or human pose and can improve the accuracy and robustness of the estimation.

This chapter will provide an overview of the main categories of approaches and basic architectures used in this area, presenting the different methods available for this important process based on the taxonomy proposed in our survey paper.

3.2 Applications based on human pose estimation

Human pose estimation is a crucial problem in computer vision, and its impact can be seen in many rapidly evolving technology domains and industrial applications. These applications range from gaming and animation to medicine and healthcare,

from gesture control to autonomous driving and security and surveillance, etc. Moreover, as computer vision technology continues to evolve, we can expect to see even more innovative use cases for pose estimation in the future.

In this section, we will outline some real-world use cases of pose estimation and explore the impact of this computer vision technology on many industries.

3.2.1 Animation & Gaming

Motion capture (MoCap) is the most important technology used by production studios in the fields of animation, film, and video games to record authentic and complex movements of people or objects, digitize them, and reproduce the actual movement of a person or object in a virtual environment. However, motion capture devices are generally expensive, complicated to implement, and challenging to use, so their use is generally limited.

Recently, researchers have turned to the computer vision task of human pose estimation to replace this technology and extract realistic information on postures [11, 12]. The pose estimation technique is particularly useful for animation and gaming. In animation, it can be used to create natural movements for animated characters, while in gaming, it can be used to detect player behavior and affect game characters accordingly. This can lead to innovative gaming experiences.

Furthermore, pose estimation is also used in virtual reality (VR) and augmented reality (AR). VR refers to the creation of simulated environments, while AR involves the integration of virtual elements in the real world. In these fields, pose estimation technology can track the movement of users and translate it into corresponding movements of virtual or augmented objects in the environment. For example, Pokemon Go and ARkit use pose estimation to place virtual objects in the real world, allowing for more interactive experiences where users can move around freely and interact with objects in their immediate environment.

An example of this technology's use is demonstrated in [13], where a system for

Figure 3.1: An example of Augmented reality use case [15]

real-time music tracking has been developed based on pose estimation techniques. The system generates virtual musician behavior and motion. Additionally, [14] presents a deep learning approach for estimating the upper-body pose of a user in a virtual reality environment. The proposed approach uses a convolutional neural network to predict key points on the user's upper body, which are then used to animate an avatar in real-time.

3.2.2 Training Robots

Robotics is a rapidly expanding and changing sector that is increasingly adopted across diverse fields such as security, healthcare, logistics, and many other service-related industries. Deep learning approaches, particularly reinforcement techniques, are commonly used for robot programming. These approaches simulate environments to train robots and enable them to reach the level of precision required to

perform specific tasks successfully.
Traditionally, in industrial robotics, 2D vision systems have been used to enable robots to perform a variety of tasks by performing time-consuming and demanding calibrations. However, recently, pose estimation has allowed robots to learn and train to perform identical actions and gestures in a faster, more responsive, and more accurate manner, instead of being manually programmed to follow trajectories and predetermined paths with programming restrictions. While humans can perform a wide range of movements in a variety of environments, robots work within a limited range of tasks. By using pose estimation technology, robots can learn a variety of human movements and evolve to perform more diverse tasks. This learning process allows the robot to be more efficient and accurate in its tasks. As an example, Vasileiadis et al. [16] proposed a method for robust human pose tracking using a deep learning-based approach with a two-stage network to estimate human poses in RGB images. Their proposed method can be used for various service robot applications, including assistive robots, autonomous vehicles, and surveillance systems, and has been reported to provide accurate results.

3.2.3 Intelligent surveillance and security systems

Intelligent surveillance and security systems have become increasingly popular due to the rise in criminal activities and the need for public safety. In recent years, human activity estimation has been integrated into surveillance systems to detect unusual or suspicious behaviors [17].
Furthermore, pose estimation technology can also be utilized to detect whether someone is walking down the street, which can help improve traffic safety and prevent accidents. Real-time surveillance systems like W/sup 4/ [18] have been developed to detect and track multiple people and monitor their activities in an outdoor environment. This system employs shape analysis and tracking to locate people and their parts and create models of people's appearance so that they can

Figure 3.2: An example of autonomous driving application using pose estimation [20]

be tracked through interactions such as occlusions. It can also recognize events between people and objects, such as depositing an object, exchanging bags, or removing an object.

Another important application of intelligent surveillance and security systems is in detecting anomalous events. A student behavior recognition system was proposed for the classroom environment by FC lin et al. [19], which uses skeleton pose estimation and person detection to monitor student behavior in real-time.

In the field of autonomous driving, pose estimation can be used to detect and track pedestrians, cyclists, and other objects in the environment. This information can then be used to make decisions about how the vehicle should behave, such as slowing down or stopping to avoid collisions [20].

3.2.4 Sports analysis

Human pose estimation is a versatile technology that has numerous applications in sports analysis [21]. Coaches and athletes can utilize this technology to enhance

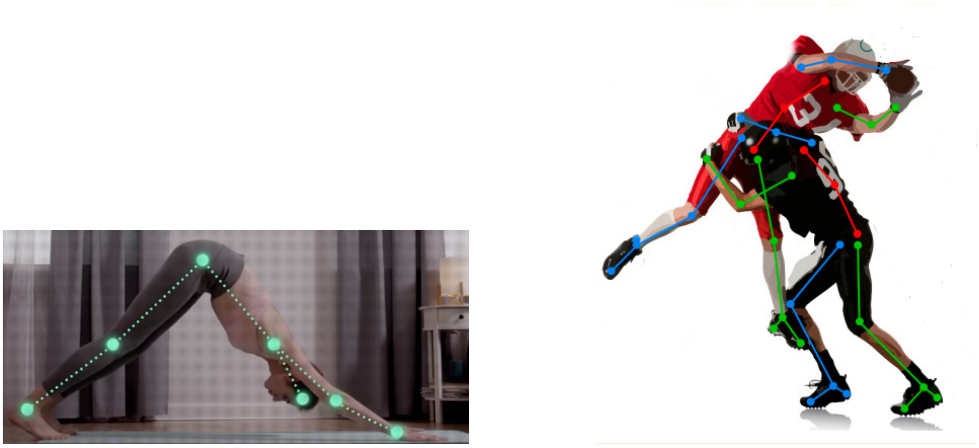

Figure 3.3: Examples in the sport analysis and fitness industry using pose estimation [21].

their performance or compare the movements of different players to develop effective game strategies. This technology can be used in various sports, including but not limited to gymnastics, weight training, dance, and team sports such as soccer.

Additionally, this technology also aids in tracking physical activities in fitness programs and for personal training. Individuals can record their physical activities and track their progress over time. They can also receive personalized cues and advice based on their fitness level and goals. There are also applications that utilize human pose estimation to analyze exercises and postures in general, providing cues and graphical analysis to assist users in improving their technique, correcting their posture, and preventing injury.

3.2.5 Healthcare

Pose estimation also has many applications in healthcare, such as biomechanical analysis, fall detection, and rehabilitation. For example, as poor posture can lead

to a variety of health problems. There are several healthcare applications that can be used to estimate human posture. These technologies can assist medical professionals track patient posture, analyze movement, and develop personalized treatment strategies to promote faster recovery and long-term health.

One application is the use of sensors to monitor posture during exercise or physical therapy. Sensors such as inertial sensors or kinetic/kinematic sensors can be placed on various parts of the body to measure movement and posture. This information can then be used to improve the effectiveness of exercises or to develop a customized treatment strategy.

Another application is the use of video technology to analyze posture. By using special cameras that have a high frame rate and resolution, movements and posture can be captured and analyzed in real time. For example, this technology can be used to diagnose and treat back problems by allowing physicians to analyze a patient's posture during movement to identify the cause of the problem.

Another application of posture analysis technology is monitoring patients during recovery from surgery or injury. By using sensors or cameras, doctors and nurses can monitor the patient's posture and ensure that they are moving properly and not causing additional damage.

3.3 Human body modeling

Human Pose Estimation (HPE) technology relies on accurate human body models that capture the complex and non-rigid nature of the human body. To build such models, several specific characteristics such as the kinematic structure, surface texture, and different parts or joints positions need to be taken into account. The ideal body models for HPE don't necessarily need to contain all the features of the human body but should fulfill the specific requirements of the particular task assigned. There are three major human model types (figure 3.4):

Skeleton-Based Models: are the most commonly used type of human body

model. They describe the structure of the human body by connecting its parts with a skeleton in a tree structure, consisting of joints such as elbows, wrists, ankles, knees, and shoulders. This model is used both in 2D and 3D human pose estimation techniques because of its flexibility.

Contour-based Models: are widely used in earlier HPE methods. They depict the shape of the human body using a collection of closed curves or edge contours, typically extracted from images or video frames through edge detection or segmentation methods. One advantage of this approach is that it is computationally efficient and can yield a reasonable approximation of the body's shape even in low-quality images. However, it may not capture fine-grained details of the body's structure and may be sensitive to variations in illumination and other contextual factors.

Volume-Based Models: are frequently utilized in medical imaging and virtual reality simulations, as they represent the human body with geometric shapes or meshes. This model provides a more comprehensive representation of the body's shape and structure, but it necessitates more computational resources and can be computationally expensive. Some examples of widely used volume-based models are Shape Completion and Animation of People (SCAPE), Skinned Multi-Person Linear model (SMPL), and a unified deformation model.

The choice of human model should also be determined by the specific task requirements.

3.4 Categories of Posture Estimation Methods

Over the years, researchers have been developing various methods to tackle the challenging task of estimating human pose from images. One way to categorize

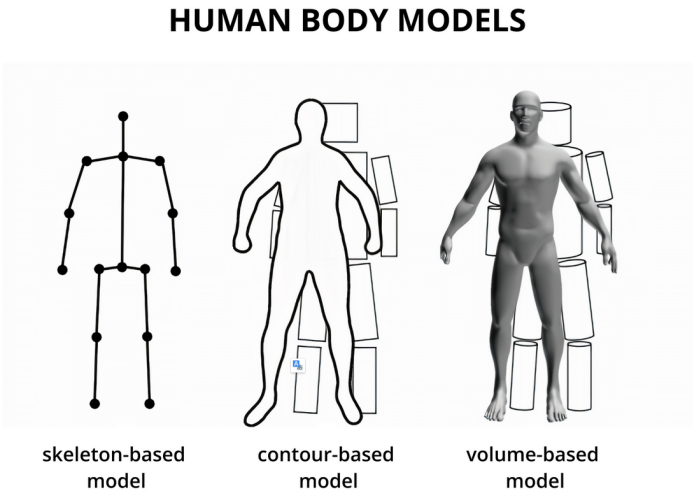

Figure 3.4: Commonly used human body models. (a) skeleton-based model; (b) contour-based models; (c) volume-based models. [22]

these methods is to divide them into three broad categories: generative, discriminative, and hybrid approaches, which have been the focus of many studies. These approaches aim to establish the underlying relationships between different body parts to deduce a body skeleton composed of linked joints or body parts. The major distinction between them is whether a method employs human body models. Generative methods rely on the diverse representations of human body models to generate possible poses that match the input data. Discriminative methods, on the other hand, learn a mechanism to model directly the relations between the input sources and human poses space based on training data. Hybrid methods combine both generative and discriminative approaches to leverage their advantages and overcome their limitations. The most of traditional methodologies based on image descriptors are investigated and examined in many surveys [23–31].

3.4.1 Generative approaches

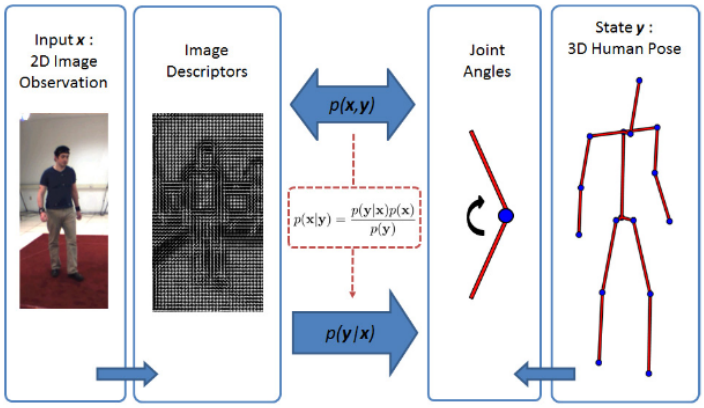

Figure 3.5: Generative Approach to Monocular 3D Human Pose Estimation [32].

The first category of approaches to modeling the human body is known as model-based approaches, in which the body is modeled as a collection of parts. Each part is indicated by a node that represents a joint, limb's length and orientation, as well as its vertices. One of the most popular model-based approaches is the Pictorial Structure Model (PSM), which was introduced by Fischler and Elschlage in the 1970s [33] and later used for pose estimation in various works [34, 35] and [36]. The PS model is a graphical tree mode that decomposes the human body into a set of parts with connections between pairs of parts, represented as springs (figure 3.6). The location of each body part contains information such as its position in the image, pixel orientation, and amount of foreshortening. The goal is to predict the location of the body joints using the observation and a priori body [34]. In most cases, the body prior has been a 2D Gaussian distribution that models the relationships between body parts. The cost function is a combination of an error term for joint location and a penalty for segment length deformations, which did not match the limb size. In 2012, Sigal et al. [37] adopted pictorial structures in 3D pose estimation.

However, these models have limitations as all parts of the person's limbs must be visible, which is not always the case in real-world scenarios with occlusions and complex lighting conditions. This limits their accuracy and poses a risk of errors. Part-based models usually model the parts before synthesizing the body pose, but they do not account for the complete anatomy of the human body. Consequently, fully occluded parts are typically not predicted accurately.

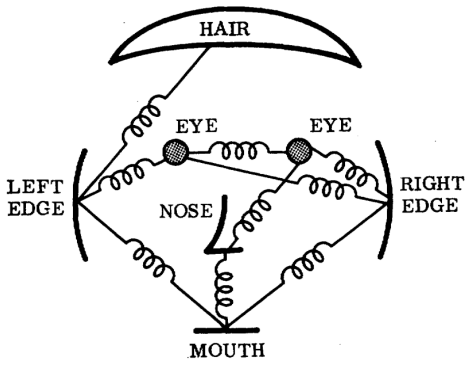

Figure 3.6: Pictorial structures model by Fischler and Elschlager. Objects (as, e.g., faces) are modeled as a set of parts that appear in some typical spatial relationship with some flexibility concerning the relative locations [33].

3.4.2 Discriminative approaches

Another approach is the use of discriminative methods (figure 3.7), which involves learning and modeling the correspondence between the extracted features and the human pose based on training data. Testing is typically faster than generative approaches because the model only requires a calculation or a limited search instead of optimizing a high-dimensional parametric space. Discriminative methods can be divided into two sub-categories based on their approach to modeling relations: learning-based methods and exemplar-based methods.
Learning-based methods are based on mapping mechanisms that can be achieved in

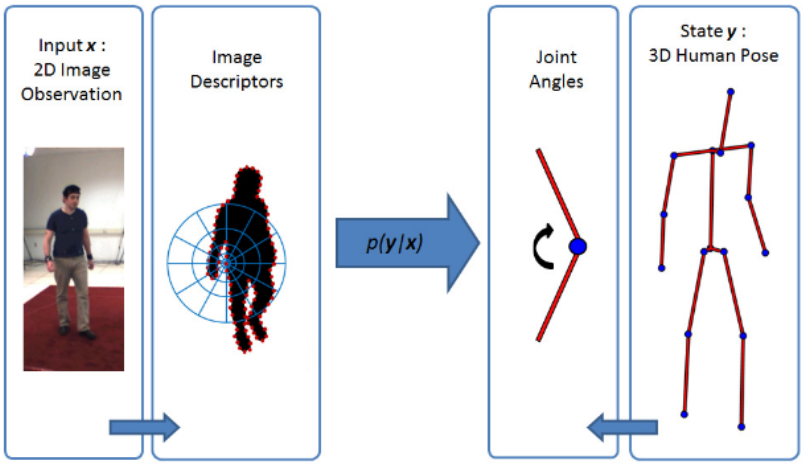

Figure 3.7: Discriminative Approach to Monocular 3D Human Pose Estimation [32].

different ways depending on the input provided. One way is to guide the algorithm along the learning path by providing it with labeled examples, known as supervised learning. Another way is to use a combination of a small amount of labeled data and a large amount of unlabeled data during training, known as semi-supervised learning. The third way, unsupervised learning, involves having only input data with no examples of expected output. In human pose estimation, the mapping can be a direct mapping from image features or a model that predicts 2D parts to 3D poses.

Several learning algorithms have been adopted, such as Support Vector Machines (SVMs) [38, 39], which train hyperplanes for discrimination between classes, and Relevance Vector Machines (RVMs) [40, 41], which are Bayesian sparse kernel techniques used in regression and classification. A discriminative bag-of-words approach has also been proposed by Ning et al. [42] for recovering 3D human pose by extracting the most representative features as vocabulary and statistically labeling each training data according to the image and vocabulary evidence. Indeed, this model only considers the occurrence of each word in the image, which results

in a histogram that represents the image. Deep learning frameworks are currently widely used in human pose estimation research and applications as discriminative approaches. We will discuss these approaches in the following section.
The exemplar-based approaches, which rely on a discrete set of specific poses with their corresponding representations, are used to estimate the pose of an unknown visual input image [43–47]. To handle this type of problem, fast and robust classification techniques such as randomized trees and random forests are often used [26].

3.4.3 Hybrid approaches:

Discriminative methods are generally faster than generative methods, but they may not be as effective for poses that were not present in the training data. While discriminative methods identify the parts of the human body to estimate the pose, generative methods project the human model into 2D picture space and measure the distance between them [42]. However, hybrid methods can be used to combine the strengths of both approaches and overcome their limitations. These methods typically use discriminative methods to obtain an initial pose estimation, and then generative methods to refine the pose within a local area [48, 49]. A good initialization is crucial for generative methods, which often rely on non-convex objective functions. The 3D human model's poses are adjusted through iterative optimization during the generative phase by comparing them with the image evidence acquired through the discriminative process [26].

3.5 Human pose estimation by deep-learning

Human pose estimation has been revolutionized by deep learning techniques, which have led to significant progress and remarkable breakthroughs in the field. There are various approaches to human pose estimation, which can be categorized

based on the number of people being detected, the representation plan (2D or 3D), and the overall strategy (top-down or bottom-up). In this overview, we will categorize the approaches based on the representation plan, and for each category, we will present the prevalent works for both single and multiple persons. This will provide a comprehensive understanding of the state-of-the-art techniques used in human pose estimation. Recent surveys have mainly focused on either 2D or 3D pose estimation, without a comprehensive perspective to explore the intrinsic similarities and connections between the two. Therefore, a more comprehensive survey covering the recent advances in pose estimation is of great need in this community [22, 27, 50, 51].

3.5.1 Overview of 2D human pose estimation approaches

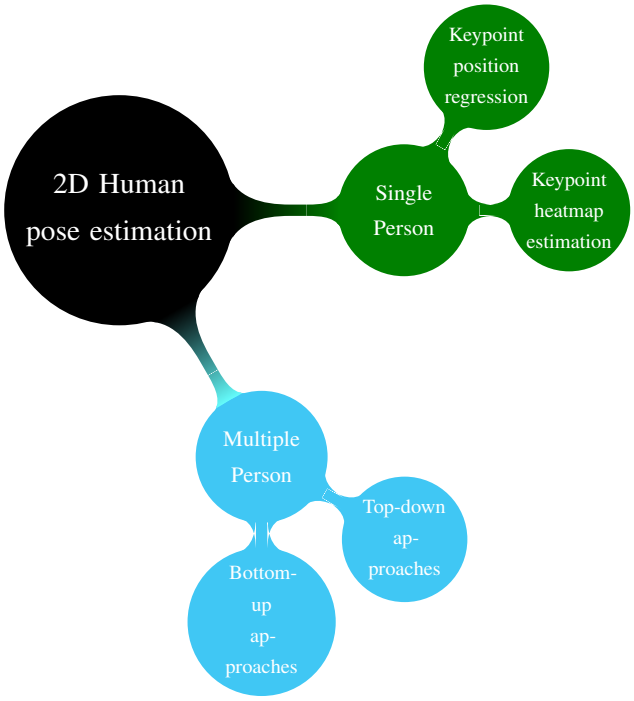

Figure 3.8: A taxonomy of 2D Human Pose Estimation approaches.

To provide a comprehensive overview of 2D human pose estimation, which involves determining the (x, y) coordinates of each joint from an RGB image, we will follow the taxonomy presented in Figure 3.8 and discuss the relevant literature on this topic. The fundamentals of 2D human pose estimation will be introduced, and we will present the prevalent works for both single and multiple persons.

3.5.2 Single-person pipeline

Single-person pose estimation methods assume that only one person is present in the image, which makes it easier to extract and predict the posture of the person [52–55]. This task is also used as an intermediate step for top-down methods of multiple people estimation, as described in [56–59], which will be discussed later.

The first neural network-based works on 2D pose estimation were presented in 2014, which showed the effectiveness of learning-rich features [52, 60, 61]. These approaches can be categorized into two types based on the way they predict the location of keypoints: direct keypoint regression [60, 62, 63] and keypoint heatmap estimation [55, 61, 64–68].

3.5.2.1 Direct Keypoint regression

Keypoint regression methods are a widely used approach for human pose estimation. These methods directly predict the coordinates of body keypoints, which are typically represented as joints. Toshev and Szegedy [60] proposed a cascaded deep neural network regressor named DeepPose to predict the coordinates of the keypoints directly from the image. Following DeepPose, several works have been proposed based on the direct regression framework. Carreira et al. [62] proposed an iterative error feedback-based human pose estimation system that uses multi-stage training and shared weights across iterations. The authors used a self-correcting model by feeding back error predictions to progressively refine

the pose. Another approach was proposed by Zhao et al. [63], which adopts a re-parameterized pose representation using bones instead of joints, and uses a compositional loss function to encode the long-range interactions in the pose. The method was proved to be either 2D or 3D compliant.

To improve the feature representation for regression-based methods, multi-task learning has been widely used. Li et al. [69] proposed a cascaded transformer-based model that can obtain the spatial relations from the joints for the keypoints prediction with regression inference. Li et al. [70] also proposed a heterogeneous multi-task framework that consists of two tasks: predicting joints coordinates from full images by a regressor and detecting body parts from image patches using a sliding window. Fan et al. [71] proposed a dual-source CNN for two tasks: joint detection and joint localization, leading to improved results. Luvizon et al. [72] learned a multi-task network to jointly handle 2D/3D pose estimation and action recognition from videos.

3.5.2.2 Keypoint heatmaps estimation

Keypoint heatmaps estimation is a widely used technique for coordinate representation in computer vision. The technique generates a heatmap for each keypoint to predict the probability of the joint occurring at each pixel. The confidence level of the keypoint position is encoded in each pixel value of the heatmap, and the highest activation of the heatmap predicts the location of the joint. Keypoint heatmap estimation was introduced by Tompson et al. in 2014 [61] to address the keypoint regression problem. They proposed to train a CNN and a graphical model jointly, where the CNN model predicts the Gaussian response heatmaps of keypoints, and the corresponding positions are determined by finding local maxima in heatmaps. The heatmaps are processed by a graphical model to learn the typical spatial relationships between joints. Subsequently, in [73], the authors used a multi-resolution CNN architecture in parallel with the existing convolution features to fuse the features at different scales to generate a coarse heatmap output,

which was refined using the convolution features in the pose refinement CNN. In contrast, Lifshitz et al. [74] proposed a method that uses information from the whole image, instead of just a sparse set of keypoint locations. They used a CNN for keypoint consensus voting, where each part of the image voted on the position of each keypoint, and then combined the keypoints votes and joint probabilities to predict the optimal pose. Another approach proposed by Wei et al. [65] was the Convolutional Pose Machines (CPM) based on the pose machine architectures [75], which consists of a sequence of CNN that repeatedly produce 2D belief maps for the location of each part. Bulat et al. [64] proposed a CNN cascade architecture for learning part relationships and spatial context that guides the regression part of the network to rely on contextual information to predict the location of occluded parts.

Newell et al. [66] introduced a novel CNN architecture called the stacked hourglass network, where features were processed repeatedly across all scales of pooling and up-sampling to produce the final predictions. They used both global context and local information to learn the predictions. Several approaches based on the original stacked hourglass network [57, 76] or modified versions [77–81] have been proposed, using, for example, a multi-scale feature pyramid (PyraNet) [78] on a Residual Module, a multi-stage refinement strategy [77] using the Deeply Learned Compositional Model (DLCM), or identity mapping between the same spatial extent feature maps and heatmaps across different stages [82].

The stacked hourglass network was also used as a generator to refine the pose, where Chou et al. [79] used it in a Generative Adversarial Network (GAN) [83] with self-adversarial training, and Cao et al. [81] improved it with a self-attention mechanism to learn long-range dependencies of body parts. Generative models have also been used to predict occluded parts of the body Chen et al. [55] designed a novel structure-aware convolutional network with a multi-task generator and two discriminators for assessing whether the body configuration is plausible based on the heatmaps and discriminating the high confidence predictions from the low-

confidence. The architecture proposed in [84] was evaluated on 2D facial landmark estimation and 3D human pose estimation datasets, using the 3D pose dataset for validation, with a network architecture based on [85].

A new method called High-Resolution Net (HRNet) was proposed by [86], which maintains high-resolution representation throughout the process by connecting multi-resolution sub-networks in parallel. This leads to rich high-resolution representations, achieved through repeated multi-scale fusions. On the other hand, [87] proposed EfficientPose, which is based on EfficientNet backbone pre-trained on ImageNet for feature extraction. EfficientPose generates low-level and high-level features from low-resolution and high-resolution images, respectively, which are concatenated to yield cross-resolution features. The desired keypoints are then localized using an iterative detection process with the Mobile DenseNet architecture. Both HRNet and EfficientPose are methods within the same area.

3.5.3 Multiple person pose estimation

Real-world applications of human pose estimation often require the estimation of multiple people in a single image, which is a more complex task than estimating the pose of a single person. The goal is to determine the pose for each individual present in the image, without prior knowledge of the number of people or their positions. This is made more challenging by occlusions and interactions between individuals, which make it difficult to locate joints accurately.

Current methods for multiple person pose estimation can be divided into two categories: top-down and bottom-up approaches. Top-down approaches first detect and localize each person in the image using bounding boxes, then apply a single-pose estimation algorithm to each box. In contrast, bottom-up approaches first identify keypoints or body parts, and then group them together to estimate the poses of each individual.

3.5.3.1 Top-down approaches

The initial step in top-down approaches involves object detection algorithms such as Fast-RCNN [88], Faster-RCNN [89] or R-FCN [90] to locate all human instances in an image or video. Subsequently, these methods estimate the keypoints of each person potentially present within the bounding boxes. For example, [56] proposed an approach that computes dense heatmaps and offsets using a dense ResNet, followed by an aggregation procedure to obtain highly localized keypoint predictions. Similarly, [58] proposed a method called simple baselines, which utilizes ResNet [91] as the backbone in the keypoint detection part and applies deconvolutional layers to generate the final heatmaps. Another example is [92] that introduced a coarse-fine network (CFN) that uses a fully convolutional Inception network, with supervisions at several levels, and three branches of detectors with different numbers of deduction stacking modules to compute receptive fields at various sizes. The feature maps generated by these coarse detectors are used in fine detector's training, providing accurate location.

In another work, [57] proposed the Regional Multi-Person Pose Estimation (RMPE) architecture, which has three stages for multi-person pose estimation. First, a symmetric spatial transformer network selects regions of interest in the image. Then, a parametric Pose Non-Maximum-Suppression algorithm eliminates the redundant detections, and finally, a data augmentation technique generates a large sample of training proposals for two-stage pose estimation. This method generates samples with the same distribution as the output of the human detector. Similarly, [93] proposed the Cascaded Pyramid Network (CPN), which addresses problems of invisible, occluded keypoints, and complex background. CPN comprises two networks: the first one (GlobalNet) localizes simple keypoints but fails to recognize occluded or invisible keypoints precisely, and the second one (RefineNet) tackles these issues by integrating all levels of feature representations from GlobalNet with an online hard keypoint mining loss.

In addition, [94] proposed a unified framework that can extract bounding boxes, masks, and keypoints simultaneously. In human pose estimation, this approach predicts K masks, one for each of K keypoint types.

3.5.3.2 Bottom-up approaches

In the literature, the first bottom-up approach was DeepCut [95], which aimed to solve detection and pose estimation tasks together. The approach first determined the number of people in the image and then computed the joint locations for each individual. The joint locations were pruned using non-maximum suppression, grouped into individuals using integer linear programming (ILP), and subsequently used to estimate body poses. An improved version of DeepCut, called Deepercut [96], was proposed to alleviate the ILP approximation and improve computational complexity. Deepercut used stronger body part detectors based on ResNet [91], image-dependent pairwise scores, and an incremental optimization approach that uses a branch-and-cut algorithm to incrementally solve several instances of ILP.

The first real-time multi-person system was proposed in [97]. The approach used keypoint heatmaps and part affinity fields (PAFs) to jointly determine the location and orientation of limbs of people over the image. A greedy algorithm was used to associate joints with a person, and a set of bipartite matching was used to maintain high accuracy while achieving real-time performance, irrespective of the number of people in the image. In [98], the same authors increased the network depth, removed the body part confidence maps refinement, and refined the PAF, which increased the speed by approximately 200% and accuracy by 7%.

Associative embedding was proposed in [76] to express output joint detection and grouping using a single-stage deep network trained end-to-end. For each detection, a vector embedding was introduced that serves as a "tag" to identify its group assignment. All detections associated with the same tag belong to the same group. The network was trained using a loss function that encouraged pairs of tags

to have similar values if the corresponding detections belong to the same group or dissimilar values otherwise. This approach combines detection and grouping steps and works with any network architecture that produces a pixel-wise prediction.

PersonLab [99] proposed a network model that detects individual keypoints and predicts the relative displacements between each pair, which allows a greedy decoding process to group keypoints into person pose instances. The authors also proposed a recurrent scheme to improve the accuracy of long-range predictions. PifPaf [100] used a Part Intensity Field (PIF) to localize body parts based on the confidence score, precise location, and size of a joint, and a Part Association Field (PAF) to associate body parts. A decoder was used to convert PIF and PAF fields into the pose estimates containing 17 joints.

Finally, Scale-Aware High-Resolution Network (HigherHRNet) [101] was introduced, based on the HRNet model [86]. HigherHRNet used deconvolution modules to generate scale-aware high-resolution heatmaps using high-resolution feature pyramids in the training stage and multi-resolution heatmap aggregation in the inference stage. The feature pyramid was composed of HRNet output and up-sampled higher-resolution outputs through a transposed convolution. The approach aimed to deal with the scale variation problem to be able to localize keypoints for both tall and small persons by adopting Pyramidal representation. Associative embedding [76] was used in the grouping stage.

3.5.4 Comparative Analysis of Top-Down and Bottom-Up Approaches in Multi-Person 2D Pose Estimation

The recent methods follow state-of-the-art handling of multi-person 2D pose estimation complexities, like occlusions, interactions between people, scales, and varying poses. Table 3.1 briefly summarizes the backbone architectures and the part association strategies of several popular models in these two categories, below, for reference, along with the human detector type for top-down approaches and AP

on the MS-COCO test-dev dataset. This comparison really points out the progress and effectiveness made by the different strategies in improving the accuracy of pose estimation.

Category	Approach	Backbone	Part Association/ Human Detector	AP (%)
Bottom-up	CMU-pose [97]	VGGNet	PAF	61.8
	Openpose [98]	CPM	PAF	64.2
	AE [76]	Stacked Hourglass	AE	65.5
	PifPaf [100]	ResNet-152	PIF + PAF	66.7
	PersonLab [99]	ResNet-152	AE	68.7
	HigherHRNet [101]	HRNet-W48	AE	70.5
Top-down	Mask-RCNN	ResNet-50	FPN	63.1
	G-RMI	ResNet-101	Faster-RCNN	64.9
	RMPE [57]	Stacked Hourglass	VGG based SSD-512	61.8
	RMPE++ [57]	PyraNet	Faster-RCNN	72.3
	CPN	ResNet-Inception	FPN	72.1
	HRNet [101]	HRNet-W48	Faster RCNN	75.5
	DarkPose [102]	HRNet-W48	Faster RCNN	77.4

Table 3.1: Comparison of Bottom-up and Top-down Approaches on MS-COCO test-dev

3.6 Overview of 3D human pose estimation approaches

In recent years, there has been significant progress in developing deep learning-based approaches for 3D human pose estimation, which is a critical task in various

fields such as computer vision, robotics, and human-computer interaction to recognize the actual posture in a scene because due to the ambiguity, this goal cannot be achieved only with 2D poses only. 3D human pose estimation is the process of inferring the 3D coordinates of human joints from 2D images or videos. This task has many applications in fields such as robotics, virtual reality, and human-computer interaction. 2D pose estimation methods has achieved high detection rates above 90% on all different human joints. There are two primary groups of techniques for 3D human pose estimation: direct methods and inverse kinematics-based methods. Direct methods obtain 3D poses from input data (such as images, videos, or depth maps) without explicitly modeling the human body's kinematic structure. In contrast, inverse kinematics-based methods use a pre-defined skeletal model to estimate 3D poses by optimizing a cost function that enforces kinematic constraints. These methods typically employ 2D estimator networks at an intermediate stage. This section focuses on methods based on deep neural networks for 3D pose estimation. We will review and compare several state-of-the-art approaches, including volumetric methods, multi-view methods, and hybrid methods that combine 2D and 3D information. However, we will discuss the methods used in the Multi-person case and the challenges and limitations of these methods later in chapter 5.5. The advancements in deep learning techniques for 3D human pose estimation have shown promising results and open up new avenues for research in this area.

Fully supervised learning is a popular learning strategy in the field of deep learning. However, creating a 3D annotated dataset is more challenging and time-consuming than creating a 2D annotated dataset, and manual annotations are not practical. Most existing 3D pose datasets are built from indoor environments using motion capture systems, which require an elaborate setup with multiple sensors and are not generalizable to natural settings. Therefore, fully supervised methods may not be suitable for 3D pose estimation, particularly for industrial or application cases. To address this, other learning strategies have been explored in the literature,

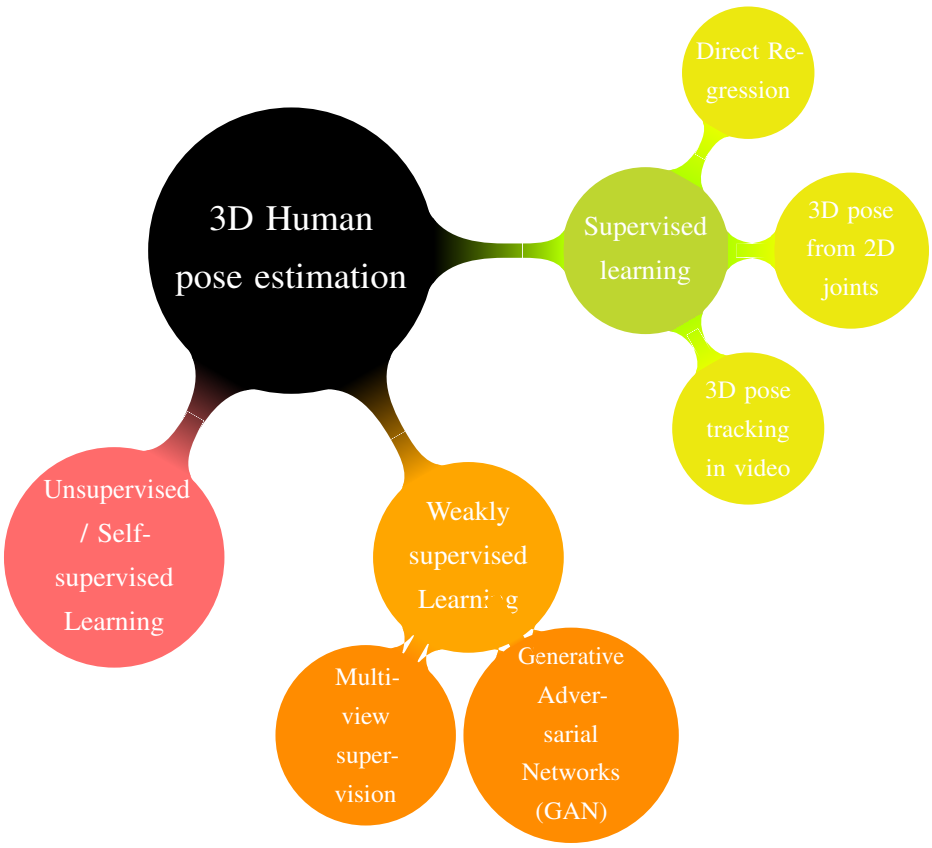

Figure 3.9: A taxonomy of 3D Human Pose Estimation approaches.

including semi-supervised learning, where a small amount of data is labeled by 3D poses, weakly supervised learning, where samples are labeled by 2D poses, and completely unsupervised or self-supervised learning. These strategies are illustrated in Figure 3.9. We adopt this organization scheme, presented in the same figure, to showcase the primary techniques proposed in each category.

3.6.1 Supervised learning

As in the 2D case, the earliest approaches for 3D human pose estimation from single images [103–108] were generally based on discriminative methods and viewed the pose estimation as a regression or a classification problem. A mapping function is learned from a space of image features, that are either extracted directly as shape context [44, 104], segmentation [107], silhouette [109], HOG [110, 111], SIFT [103], or computed as body part information [112] to the pose in 3D space. This mapping must be well generalized to accurately estimate a 3D pose from a test image that has never been seen before. The outstanding results of deep learning methods in computer vision have made them a general trend to use deep nets to automatically learn the key information in images. Some papers rely on supervision learning to directly regress joint locations and predict the 3D pose from 2D monocular images without intermediate supervision [63, 113–116]. In this case, models must be trained using a 3D pose annotated images dataset. For example, Human3.6M [106] and HumanEva-I [117] datasets contain images captured in controlled environments using MOCAP systems. Thus, they prevent the models to generalize well to images in different environments as in the wild. To address this issue, some approaches use both 2D datasets in the wild and 3D MOCAP datasets to guide and improve the training.

3.6.1.1 Direct regression

This kind of approach directly maps image features to a 3D pose, but it typically lacks constraints. While this method has shown promise in 3D human pose estimation, it may not be suitable for all applications.

3.6.1.1.1 Direct regression using only 3D Data The first attempt to predict 3D joint locations directly from single images using deep neural networks was made by [113]. Their approach involved training the network on bounding box images in a multi-task end-to-end learning framework, where pose regression and body parts detection tasks were jointly trained. The first task estimated keypoint locations, while the second task classified whether a local window contained the specific keypoint or not. Similarly, [115] introduced an end-to-end regression architecture that achieved structured prediction by incorporating a pre-trained auto-encoder at the top of traditional CNN networks, rather than directly regressing joint coordinates. With the auto-encoder, they were able to learn a high-dimensional latent pose representation and account for joint dependencies. The same authors proposed a method in [116] to learn a regression model for 3D pose mapping from videos using CNNs.

To ensure the validity of the predicted poses, [114] embedded a kinematic object model into a deep learning algorithm to regress joint angles of a skeleton. They defined a continuous and differentiable kinematic function based on bone lengths, bone connections, and a definition of joint rotations. This function was integrated into a neural network as a special layer, called kinematic layer, to map the motion parameters to joints.

Another approach to reduce data variance was proposed by [63], who used bones representation instead of joints in a structure-aware regression approach. They defined a compositional loss function that encoded long-range interactions between these bones, based on the joint connection structure.

3.6.1.1.2 Fully supervised learning for 3D pose in the wild Despite the success of 3D pose ConvNets regressors using 3D data, models cannot generalize well to images in the wild, due to the difficulty to generate ground-truth 3D body joint locations in an unconstrained environment. Variations in background, viewpoint and lighting are hence limited. To address this issue, some researchers have proposed end-to-end approaches trained with both paired 2D and 3D datasets, since there is an abundance of 2D pose images annotated in the wild. For example, some authors have proposed a single model that shared intermediate CNN features between 2D and 3D joint locations, such as the algorithm proposed by Park et al. [118], which learned the relationship between 2D pose and 3D pose using image features and 2D pose estimation results as inputs. Other approaches include a fine-tuning strategy to 2D data, such as the approach proposed by Pavlakos et al. [119], which trained a CNN to predict 3D joint locations by regressing the per voxel likelihood for each joint in this volume separately using the end-to-end learning paradigm. They also designed a coarse-to-fine supervision learning scheme to deal with the increased dimensionality of the volumetric representation and improve initial estimates.

Similarly, Tome et al. [120] proposed a multi-stage CNN architecture that can be trained end-to-end to jointly estimate 2D and 3D joint locations. They added a pre-trained layer to the convolutional pose machine (CPM) [65] based on a probabilistic 3D pose model to lift 2D landmark coordinates into 3D space and propagate 3D information about the skeletal structure to the 2D convolutional layers. [121] jointly predicted 2D and 3D poses in a coupled way. The method used camera viewpoint with 2D joint locations to incorporate depth information and to get global joint configuration information used in a 3D pose estimation network.

2D and 3D data can be combined to learn a pose regressor, such as the compositional pose proposed in [63], or the end-to-end trainable multitask deep model proposed in [72], jointly learning 2D and 3D poses as well as action recognition.

Intermediate volumetric representation is used for 3D poses (as in [119]) but with a lower resolution, since coordinates were computed from volumetric heatmaps using a soft-argmax operation instead of argmax. Li et al. [122] proposed another regression approach that estimated 3D pose with a nearest neighbor form between images and pose. They trained the network to learn a similarity score function between feature embedding of the input image and the 3D pose.

3.6.1.2 3D poses from 2D joints

Inspired by the rapid development of 2D human pose estimation algorithms, many works [63, 121, 123–131] have tried to utilize 2D pose estimation results for 3D human pose estimation to improve in-the-wild generalization performance. The first step of these approaches estimates the 2D pose from monocular images, while the second step lifts the 2D joint locations to 3D poses with a subsequent optimization step. This can be done by either training a deep neural network to learn a mapping between 2D and 3D poses [124] or by finding the best 3D pose that represents the 2D observations using a complete dictionary of 3D poses learned from large 3D pose databases using principal component analysis (PCA) [132] or another dictionary learning method. Lee et al. [125] were the first to infer 3D joint locations from their 2D projections using binary decision trees, in which each split corresponds to two possible states of a joint relative to its parent. Taylor [133] assumed that the ratios of limb lengths were known and estimated 3D pose from this information.

3.6.1.2.1 Exemplar-based approaches Exemplar-based approaches are a type of content-based image retrieval that use a reference image to search for similar images.

Rogez and Schmid [134] approached pose estimation as a classification problem, where each image is assigned to the class with the highest score. This ensures a valid pose prediction, but it is limited to the existing classes, resulting in approxi-

mate poses. The higher the number of classes, the more precise the classification becomes, but it also makes discrimination more challenging. To address this issue, Rogez et al. [135] proposed the LCR-Net architecture, an end-to-end system for the joint estimation of both 2D and 3D human poses in natural images that combines classification and regression for improved results. The LCR-Net architecture has three main stages, which share convolutional feature layers and are jointly trained. The localization stage generates a set of pose proposals in the image using a fixed set of anchor poses and bounding boxes obtained by using a Region Proposal Network (RPN). The classification stage scores the different pose proposals and predicts the closest anchor-pose for each bounding box of a set of K-poses obtained from a MOCAP data-set, similar to [136]. Finally, the regressor stage refines the coarse anchor-poses located in the region proposals in 2D and 3D for a precise pose. The overall loss is defined as the sum of three losses: Localization loss, Classification loss, and Regression loss. Du et al. [137] proposed a method that combines images and calculated height-maps to detect 2D joint landmarks using a dual-stream CNN. They formulated an objective function to estimate 3D pose from the detected 2D pose. The method uses a dictionary of 3D poses, and the objective function minimizes the loss over the coefficients of the dictionary and its pose-conditioned joint velocity. In the state of the art, some authors have created a comprehensive basis of 3D poses to simplify the estimation process. Such as Zhou et al. [138] who have created a dictionary of 3D shapes from training set, and use a convex relaxation approach to estimate the 3D pose from the basis shapes. The method begins by selecting a set of basis shapes that represent the 3D shape, and then solving a series of convex optimization problems to estimate the 3D shape. A convex approach is then proposed to jointly estimate the coefficients of the sparse representation to handle missing data and produce a sparse solution that is close to the true underlying shape.

The same authors [139] predicted the uncertainty heatmaps of the 2D joints location, then combined these maps with a sparse model of 3D human pose to

retrieve the 3D pose using an Expectation-Maximization algorithm. Chen and Ramanan [140] adopted a large library of 2D keypoints and their 3D representations to match the depth of the 2D poses estimated by the k-nearest neighbor algorithm. Similarly, Gupta et al. [141] and Jiang et al. [75] used a large database of poses to resolve ambiguities through nearest neighbor queries. The authors of [142] built a sparse representation of 3D human pose in an over-complete dictionary and then proposed a projected matching pursuit algorithm to estimate the sparse model from only 2D projections. Simo-Serra et al. [132] proposed a stochastic sampling method to propagate the noise from the image plane to the shape space, and then imposed kinematic constraints to disambiguate among the set of feasible 3D shapes. In contrast to [132, 142, 143], the authors of [144, 145] addressed the issue of 2D pose estimation errors. In [145], Simo-Serra et al. proposed a Bayesian framework that integrated a generative kinematic model and discriminative 2D part detectors based on HOGs to generate the set of 3D pose hypotheses. Yasin et al. [144] combined two independent training sources using a dual-source approach. During inference, they retrieved the nearest 3D poses using the estimated 2D pose. The final 3D pose was then reconstructed by minimizing the projection error under the constraint that the solution is close to the retrieved poses.

Another strategy is to integrate a generative 3D body shape model into the skinned multi-person linear model [146] to reconstruct 3D pose and shape. For example, Bogo et al. [147] proposed a method called SMPLify for estimating 3D human pose and shape. First, DeepCut [95] is used to generate 2D body joint locations, which are then fit to the predicted 2D joints using SMPL [146]. The fitting is driven by an objective function that matches the projected 3D model joints and detected 2D joints as a sum of five error terms: a joint-based data term, three pose priors, and a shape prior.

$$E(\beta,\theta) = E_J(\beta,\theta;K,J_{est}) + \lambda_\theta E_\theta(\theta) + \lambda_a E_a(\theta) + \lambda_{sp} E_{sp}(\theta;\beta) + \lambda_\beta E_\beta(\beta)$$

[147], where K are camera parameters and λ_θ,λ_a,$\lambda_{sp}$and λ_β are scalar weights.

This function is minimized to directly optimize the pose and shape. Tripathi et al. [148] outputs 3D SMPL body model parameters from 2D poses using the 3D joints predicted by another network as pseudo ground-truth in training, since the method is unsupervised by nature.
However, since these methods rely on optimization and data retrieval, they tend to have high computational time and require normalized input. To avoid these problems, some techniques have turned to the use of deep learning models, which can learn the underlying pose structures from the data during 3D pose estimation from the 2D pose.

3.6.1.2.2 Deep neural mapping Deep learning models have been used to learn the mapping from 2D to 3D poses, which can be implemented using fully connected, convolutional, or recurrent networks. Moreno-Noguer et al. [124] used a two-step pipeline, first locating 2D human joints using Convolutional Pose Machine [65], then inferring the 3D poses based on these observations using two Euclidean distance matrix regressions. Martinez et al. [85] proposed a simple multilayer perceptron, with various regularization techniques, which regresses 3D joint locations from 2D keypoints predicted by stacked hourglass network [66]. Mehta et al. [149] used transfer learning to transfer knowledge learned for 2D joints location to 3D pose estimation in a supervised manner. These methods are typically more efficient than optimization-based approaches and do not require normalized input data.
The VNect approach [150] employed a combination of a CNN, which was trained on both 3D and 2D human pose datasets, and a kinematic skeleton fitting technique to generate smooth, temporal 3D skeletal pose estimates in real-time.
To address the issue of different 3D poses corresponding to the same 2D location of joints, some researchers have proposed using recurrent neural networks to model temporal information, which is especially useful for video inputs. Additionally, these approaches often incorporate prior knowledge of 3D human pose into the

deep learning models so that they can learn the 2D to 3D pose mapping.

3.6.1.3 3D pose tracking in video

Understanding human activity and/ or pose estimation requires rather a processing on a video or a sequence of images. Most approaches have attempted to exploit temporal information [116, 140, 151–153] whatever the methodology followed for 3D pose estimation. [116] followed direct regression approaches, [139, 140, 152] used exemplar-based approaches and [150, 153] proposed deep neural mapping approaches.

This subsection summarizes papers which explored variants of a temporal search strategy.

[151] used tracking-by-detection to associate 2D poses detected in each frame individually over short image sequences (tracklets) then used them to recover 3D pose. [150] applied temporal filtering and smoothing across 2D and 3D poses from previous frames to obtain a temporally stable and robust result. [152, 153] used a sequence-to-sequence network to predict temporally consistent 3D poses.

Long Short Term Memory networks (LSTMs) [154] are widely used for pose estimation from monocular videos since they are able to learn long-term dependencies.

[155] presented a RNN approach which considers priors on body part based structural connectivity of joints. Authors proposed a LSTM-based deep learning architecture called propagating LSTM networks (p-LSTMs) to estimate depth information from 2D joints location obtained with a Stacked Hourglass Network [66]. Several LSTMs were connected in series in order to elaborate the 3D pose while transferring the depth information, called pose depth cues. AnimePose [156] used Scene-LSTM to estimate the temporal trajectory of the person and track the overlapping poses in the occluded frames. Authors predicted the missing poses in the previous frames and plotted the trajectory of each keypoint, then estimated the

position of joints in the next frames. [157] proposed a spatial-temporal convolutional long short-term memory model (ST-CLSTM) with two parts allowing to capture the spatial features and temporal consistency among the frames. Weights were updated using temporal consistency and spatial losses between the estimated and the ground-truth depths, computed by a 3D convolutional neural network (3D CNN). GANs were used as a temporal consistency loss to maintain the temporal consistency among the estimated depth frames. Authors applied a 3D CNN as a discriminator to output the temporal loss from the estimated and ground-truth depth sequences, and ST-CLSTM acted as the generator.

Inspired by convolutional pose machine [65], [158] presented a Recurrent 3D Pose Sequence Machine (RPSM) that integrates both convolutional and recurrent neural networks to exploit spatial and temporal constraints from a sequence of images. The network predicts the 3D poses for each frame, and then sequentially refines them with multi-stage recurrent learning. RPSM is composed of three consecutive modules: a 2D pose module extracting the image-dependent pose representations, a feature adaptation module to transform the pose representation from 2D to 3D domain, and a 3D pose recurrent module to predict the human joints in 3D coordinates. At each stage, the 2D pose module takes as inputs each frame and 2D feature maps produced in previous stages and progressively updates the 2D pose representations.

Given a sequence of 2D joint locations, [152] proposed a sequence-to-sequence learning model using LSTM units with Layer Normalization and Recurrent Dropout to regularize the network. The encoder part of the network transforms the information of a sequence of 2D poses into a fixed-size vector. This vector is then decoded by the decoder side, built with residual connections to help learn the perturbation and predict the 3D pose in each frame from the previous frames. Compared to [158], this method doesn't use multiple refinement stages which makes it simple and efficient.

In PoseNet3D [148], temporal dynamics are modeled using dilated convolu-

tions allowing feedback at every time-step and avoids common pitfalls in using LSTM/RNN.

Residual connections, as well as dilated temporal convolutions over 2D keypoints were also used in [159, 160] used Temporal convolutional Network to lift 2D keypoints to 3D joints. The TCN [9] architecture is a 1D fully convolutional plus causal dilated convolutions, where each hidden layer has the same length as the input layer. The dilated convolution operation F on element s of the sequence is defined as:

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i).x_{s-d.i}$$

where d is the dilation factor, k is the filter size, and $s - d.i$ accounts for the direction of the past. This architecture takes a sequence of any length and map it to an output sequence of the same length as with an RNN.

[160] started with predicted 2D keypoints for unlabeled video, then estimated 3D poses and finally used a semi-supervised training method to back-project 3D data to the input 2D keypoints. Likewise, authors in [161] proposed graph attention spatio-temporal convolutional network (GAST-Net), also composed of dilated temporal model to tackle long-term pattern to process poses in multi-frame estimation. The model also comprises graph attention blocks, including a local spatial attention network to model the hierarchical and symmetrical structure of the human skeleton, and a global spatial attention network to extract global semantic information, enabling better encoding of the human body's spatial characteristics. Moreover, the graph convolutional networks is used in others related researches [63, 162, 163]. As well as [164], the authors proposed a U-shaped spatial-temporal conditional directed graph convolution network to leverage varying non-local dependence for different poses by conditioning the graph topology on input poses and representing the human skeleton as a directed graph with the joints as nodes and bones as edges. They achieve the highest accuracy in the Human3.6m database when compared to other monocular view picture searches. Otherwise, to reconstruct

3D poses from a sequence of predicted 2D poses, Li et al. [165] used the Vanilla Transformer Encoder (VTE) to model long-range information, and the whole sequence scale to ensure temporal smoothness.then, a Strided Transformer Encoder (STE) is used to build one target posture representation.

3.6.2 Weakly supervised learning

Various weakly or self-supervised pose estimation methods have been proposed due to the lack of 3D data. Unlike two stage approaches that use 2D joints as input, weakly supervised techniques used paired 2D data plus unpaired 3D Data.

3.6.2.1 Generative Adversarial Networks (GAN)

If GANs [83] have successfully been applied for 2D pose estimation, several works also used generative networks for 3D human pose estimation, especially in weakly supervised learning.

AIGN (Adversarial Inverse Graphics network) is a weakly supervised neural network proposed by [166], using GANs to learn from unpaired 2D/3D datasets and including a 2D projection consistency term. Given an image $\mathbf{x}$, a set of generators $\mathcal{G}_i, i \in [\![1 \cdots K]\!]$ maps $\mathbf{x}$ to a set of predictions $\mathcal{G}_i(\mathbf{x})$. A task-specific differentiable renderer gives predictions $\mathcal{P}(\mathcal{G}_i(\mathbf{x}))$ back to the original input space. Discriminator networks $\mathcal{D}_i, i \in [\![1 \cdots K]\!]$ are then trained to discriminate between predictions $\mathcal{G}_i(\mathbf{x})$ and a collection of ground-truth factors M_i, using as loss a combination of a reconstruction loss and an adversarial loss:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \|\mathcal{P}(\mathcal{G}(\mathbf{x})) - \mathbf{x}\|_2 + \beta \sum_{i=1}^{K} log\mathcal{D}_i(M_i) + log\left(1 - \mathcal{D}_i(\mathcal{G}_i(\mathbf{x}))\right)$$

[131] introduced an adversarial re-projection network (RepNet), composed of three networks: a pose and camera estimation network, a critic network and a re-projection network. 2D observations are lifted to 3D poses by means of a distribution mapping approach with a weakly supervised adversarial training. The

2D human pose observations, used as input, are passed through the first network composed of a pose generator network, built by two consecutive residual blocks, to regress the 3D pose and camera estimation network. The output is a vector of camera parameters. The results of both branches are given to the re-projection layer to learn matching 2D and 3D poses by minimizing the re-projection loss, the gap between the initial 2D pose and 2D pose re-projection. The critic network, based on fully connected networks and taking advantage of kinematic chain space of [167] and Wasserstein loss function, develops a weakly supervised training procedure.

Using GANs also allows to generate multiple diverse 3D human pose hypotheses from 2D detection of joints, as in [168]. The model produces a pose by combining parts and every plausible pose can be generated since only anatomical constraints must be respected. Authors argue that generating multiple plausible poses is more reasonable, due to possible occlusions and uncertainty in depth estimation.

3.6.2.2 Multi-view supervision

Multi-view 3D human pose estimation studies generally aim to obtain ground truth annotations for the estimation of monocular 3D human pose. Weak supervision transfer learning method has been explored in [169] using training data with mixed 2D and 3D labels. The authors introduced a 3D geometric constraint to regularize the predicted 3D poses on images that only have 2D annotations. [170] proposed to replace most of the annotations by the use of multiple views to train the system to predict the same pose in all views. Also, [171] exploited different views to predict 3D keypoints of an object on unlabeled instances.

3.6.3 Self-Supervised/Unsupervised Learning

Unsupervised learning tries to output 3D skeletons by using only 2D poses in training and avoiding using either paired or unpaired 3D data.
PoseNet3D [148] is an end-to end framework jointly finetuning two networks to predict Skin Multi Person Linear Model (SMPL) parameters and 3D pose joints. The use of a student/teacher paradigm allows to avoid using 3D data. The first network (the teacher) is trained to output 3D skeletons, using only 2D poses for training. It distills its knowledge to the student network that predicts 3D pose in SMPL representation. Both networks are finally jointly finetuned in an end-to-end manner using temporal, self-consistency and adversarial losses, improving the accuracy of each individual network.

Self-supervised learning, generating its own labels from data, is also used for 3D pose estimation. [172] proposed a differentiable and modular self-supervised learning framework based on conventional encoder-decoder architecture. Specifically, the encoder network produces a set of local 3D vectors from an input RGB image, camera parameters and a foreground human appearance. Local limb vectors are processed by leveraging the prior knowledge on human skeleton and poses, which is in the form of a single part based 2D puppet model, through a series of 3D transformations to obtain the camera projected 2D pose. The encoded representations are used by the decoder to project onto 2D space and synthesize foreground human image and 2D part segmentation. This approach remains limited for multiple 3D pose estimation or partial occlusions. [173] presented EpipolarPose which does not need any 3D ground-truth data. It uses 2D pose estimation and epipolar geometry to obtain 3D poses which are subsequently processed to train a 3D pose estimator.

3.6.4 Comparative Analysis of Methods Based on Learning Paradigms

After presenting the 3D human pose estimation methods according to their learning paradigms, it is crucial to compare and contrast these approaches to identify their main advantages, inherent challenges, and ideal scenarios. The subsequent comparative analysis is presented in Table 3.2. This comparison wraps up our discussion and serves as a guide for selecting the most suitable paradigm for specific research questions or practical implementations in the field of 3D human pose estimation.

Learning Paradigm	**Advantages**	**Disadvantages**	**Suitable Applications**	**Examples Human P timation**
Supervised Learning	High accuracy with labeled data. Clear performance metrics.	Requires extensive labeled datasets. Time-consuming annotation.	Controlled environments. Specific pose estimation tasks where labeled data is available.	[63, 174–1
Semi-supervised Learning	Utilizes both labeled and unlabeled data. Improves generalization.	Still needs some labeled data. Complex model tuning.	Scenarios with limited labeled data. Enhancing model performance with additional unlabeled data.	[177–179]
Weakly-supervised Learning	Less precise data requirements. Flexible with available annotations.	Potentially lower accuracy. Difficult to quantify performance.	Scenarios lacking 3D pose annotations. Use of 2D joint locations for 3D pose estimation.	[131, 180–
Unsupervised Learning	No need for labeled data. Innovative methods for knowledge discovery.	May yield less accurate results. Challenging to design effective models.	Exploration of unknown poses. Learning from purely observational data.	[148, 185–
Self-supervised Learning	Generates its own training data. Can leverage temporal consistency.	Dependent on data quality and structure. May introduce bias in self-generated labels.	Continuous or semi-continuous monitoring scenarios. Tasks where temporal or structural data consistency can be exploited.	[172, 173 194]

Table 3.2: Advantages, Disadvantages, Suitable Applications, and Examples for Each Learning Paradigm

3.7 Overview of Databases and Evaluation Metrics for Human Pose Estimation

3.7.1 Common databases and evaluation metrics for 2D human pose estimation

Table 3.3 provides a list of commonly used datasets for 2D human pose estimation, while Table 3.4 details the specific evaluation metrics employed in these studies.

Database	Year	Characteristics	Evaluation Metrics
MPII Human Pose Dataset [195]	2014	Approximately 25,000 images paperswithcode_mpii-human-pose; Diverse human activities and poses; 2D joint annotations for 16 body joints; Full body	Percentage of Correct Keypoints (PCK); Percentage of Correct Keypoints with a given threshold (PCKh); Mean Average Precision (mAP); Percentage of Correct Parts (PCP)
COCO (Common Objects in Context) Keypoints Dataset [196]	2014	More than 200,000 images with diverse scenes github_COCO-Human-Pose; 2D annotations for 17 keypoints; Full body	Percentage of Correct Keypoints (PCK); Percentage of Correct Keypoints with a given threshold (PCKh); Percentage of Correct Parts (PCP)
LSP (Leeds Sports Pose) Dataset	2010	Contains 2,000 images paperswithcode_lsp; Sports-focused dataset; Pose annotations for 14 keypoints; Full body	Percentage of Correct Keypoints (PCK); Percentage of Correct Keypoints with a given threshold (PCKh)
PoseTrack Dataset [197]	2017	514 videos including 66,374 frames in total paperswithcode_PoseTrack; Large-scale dataset with videos captured in real-world scenarios; Annotations for 2D keypoints and instance-level tracking; Full body	Percentage of Correct Keypoints (PCK); Percentage of Correct Keypoints with a given threshold (PCKh)
FLIC (Frames Labeled In Cinema) Dataset [198]	2013	3,987 training images and 1,016 test images [199]; Images extracted from movies; Annotations for 10 body joints; Upper body	Percentage of Correct Keypoints (PCK); Percentage of Correct Keypoints with a given threshold (PCKh); Percentage of Correct Parts (PCP)

Table 3.3: Common Databases for 2D Human Pose Estimation

Evaluation Metric	**Characteristics**
Percentage of Correct Keypoints (PCK)	Measures the percentage of correctly predicted keypoints based on a specified threshold distance
Percentage of Correct Keypoints with a given threshold (PCKh)	Similar to PCK, but allows for different threshold distances for different body joints
Mean Average Precision (mAP)	Computes the average precision across different recall thresholds, often used for multi-person pose estimation tasks
Percentage of Correct Parts (PCP)	Measures the percentage of correctly predicted parts (e.g., upper arm, lower leg) based on a specified threshold
Mean Per Joint Position Error (MPJPE)	Computes the mean per joint position error between the estimated and ground truth 3D poses

Table 3.4: Characteristics of 2D Human Pose Estimation Evaluation Metrics

3.7.2 Common databases and evaluation metrics for 3D human pose estimation

Table 3.5 provides a list of commonly used datasets for 3D human pose estimation, while Table 3.6 details the specific evaluation metrics employed in these studies.

	Dataset	Description	Evaluation metrics
Single Person	HumanEva-I [117] 2010	7 calibrated video sequences using multiple RGB and gray-scale cameras, synchronized with 3D body poses obtained using marker-based motion capture system The database contains 4 subjects performing a 6 common actions	3D error metric
	Human3.6M [200] 2013	The most popular and biggest data-set and benchmark for 3D human pose estimation 3.6 million indoor video frames and corresponding poses of 11 professional actors captured by MoCap system from 4 camera viewpoints Subjects 9 and 11 are used for testing, as in prior studies	MPJPE Procrustes aligned MPJPE MRPE
	MPI-INF-3DHP [149] 2017	It consists of more than 1.3 million frame captured with marker-less motion capture using 14 RGB cameras, consisting of both constrained indoor and complex outdoor scenes It has 8 subjects performing 8 activity sets.	MPJPE 3D_PCK AUC_{rel}
Multiple Person	MuCo-3DHP [201] 2018	Training data-set which merges randomly sampled 3D poses from single-person 3D human pose data-set MPI-INF-3DHP to form realistic multi-person scenes.	MPJPE 3D-PCK AUC_{rel} $3DPCK_{abs}$
	MuPoTS-3D [201] 2018	A data-set used for testing real-world shot of a 3D human pose dataset containing 20 videos (8000 frames) captured in both indoor and outdoor scenes, with challenging occlusions and person-person interactions.	3D-PCK AUC_{rel} $3DPCK_{abs}$
	Muco-Temp [202] 2020	A data-set generated in the same way as MuCo-3DHP. It consists of videos instead of frames Usually used for temporal networks training.	3D-PCK AUC_{rel} $3DPCK_{abs}$ MPJPE MRPE

Table 3.5: Common databases for 3D human pose estimation

	Evaluation metric	Name	Description
Person-centric (relative 3D pose)	3D error metric	3D error metric	which measures the average squared distance between the predicted pose coordinates and the actual ones.
	MPJPE	Mean per joint position error	Mean Euclidean error averaged over all joints and all poses, calculated after aligning the human root of the estimated and ground truth 3D poses. $MPJPE = \frac{1}{T}\frac{1}{N}\sum_{t=1}^{T}\sum_{i=1}^{N}\left\|(J_i^{(t)} - {J_{root}}^{(t)}) - (\hat{J}_i^{(t)} - \hat{J}_{root}^{(t)})\right\|_2$ The first protocol **(P1)** uses five subjects for training and two for testing, while the second protocol **(P2)** uses six subjects for training and one for testing. The third protocol **(P3)** splits the dataset in the same way as P1 but evaluates only sequences captured by the frontal camera in trial 1 without sub-sampling the original video. The error is averaged over 14 joints in P1 and P2 and a subset of 14 joints in P3. All protocols use Procrustes analysis to calculate the pose error.
	3DPCK	3D Percentage of Correct Keypoints	measures the percentage of correctly estimated keypoints within a certain distance threshold. In studies, an estimated joint is considered correct if it is within a 150 mm distance from the corresponding ground truth joint.
	AUC_{rel}	Area under 3D-PCK curve	This performance metric is calculated by plotting the PCK values against different distance thresholds and integrating the area under the curve. A higher value of this metric indicates better performance of the algorithm.
	MRPE	Mean Root Position Error	The average error of the absolute root joint (the hip) localization. $MRPE = \frac{1}{T}\sum_{t=1}^{T}\left\|({J_{root}}^{(t)} - \hat{J}_{root}^{(t)})\right\|_2$

Camera-centric (absolute 3D pose)	AP_{25}^{root}	Average precision of the root	permit to measure the 3D human root location prediction error, which considers the prediction as correct when the Euclidean distance between the estimated and the groundtruth coordinates is smaller than 25cm.
	$3DPCK_{abs}$	3D Percentage of Correct absolute Keypoints	3DPCK without root alignment to evaluate the absolute poses In studies, the threshold distance used for an absolute joint to be estimated as correct is 250 mm.

Table 3.6: Evaluation metrics.

Note that T denotes the total number of test samples and N denotes the number of joints. Ground-truth joint and the predicted joint are indicated by J and $\hat{J}$, respectively. i represents each joint from all joints and *root* represents the root-joint.

3.8 Conclusion

This chapter provides an overview of the different approaches and techniques for human pose estimation from monocular images, categorized based on their approach to 2D and 3D, single and multiple person scenarios, learning algorithms, and body structure interpretation. The networks and architectures used in these methods were also presented, along with the datasets and evaluation metrics used in the field.

Recent years have seen significant attention to 3D pose estimation, with deep learning enabling significant improvements in performance compared to 2D pose estimation. These approaches leverage prior knowledge of 3D human poses such as kinematic models to control the mapping between 2D and 3D poses, resulting in better accuracy.

However, these methods are still not suitable for real-life applications due to various challenges. Most of the works only estimate the pose of one person in specific conditions, such as an indoor environment. Additionally, their accuracy decreases for smaller or more distant persons, or for fast movements.

The problem of 3D multi-person pose estimation from monocular images, particularly in real-time, remains an open area of research with numerous challenges yet to be tackled.

Part II

3D Real-time Multi-person pose estimation : Software system design and developement

General introduction

Multi-person pose estimation involves determining the 3D poses of multiple individuals in an image or video. This is an important area of research in computer vision because it has many practical applications and could impact a wide range of fields. It serves to revolutionize the way humans interact with computers and each other. Absolute pose is more useful in certain scenarios because it allows for the objects or people to be located and tracked in a consistent coordinate system, regardless of their initial positions or orientations. In contrast, relative pose is more useful in certain applications because it allows for the objects or people to be located and tracked relative to each other, which may be more relevant in certain scenarios. Ultimately, which type of pose is more important depends on the specific application and the needs of the user.

3D root-relative pose estimation and 3D absolute pose estimation are two methods determining the orientation and position of an object or body in 3D space. The main difference between the two is the reference frame used to determine the pose. In 3D root-relative pose estimation, the pose of an object or body is determined with respect to a reference frame or "root" position. This means that the pose is defined relatively to the position and orientation of the reference frame, rather than with respect to an external coordinate system.

On the other hand, 3D absolute pose estimation involves determining the pose of an object or body with respect to an external coordinate system. This means that the pose is defined in terms of the position and orientation of the object or body in relation to the coordinate system, rather than with respect to a reference frame.

3D root-relative pose estimation is often used in situations where the movement of an object or body is being tracked over time, such as in robotics or computer vision. 3D absolute pose estimation is used in situations where the position and orientation of an object or body need to be determined with respect to a fixed coordinate system, such as in augmented reality or robotics.

CHAPTER

3D root-relative person pose estimation

4.1 Introduction

This chapter presents the design of the software system, a multi-stage framework based on deep learning networks. The system consists of four stages: the human detector, the 2D human pose estimator, the 3D root-relative pose estimator, and the depth root estimator. The chapter provides a detailed overview of these components and their functions within the system.

4.2 Human detection

Human detection is a specific type of object detection, which is a sub-field of computer vision that focuses on identifying and locating objects in images or videos. Object detection has a wide range of practical applications, including autonomous and assisting driving, human-computer interaction, robotics, security and surveillance. In this section, we discuss the main categories of existing human detection methods to provide context for the selection of the detector used in our work.

4.2.1 Existing Human detection methods

Human detection specifically refers to the task of the recognition and detection of human behavior or characteristics through computer vision techniques. There are several existing methods for detecting humans in images and video streams, which can be broadly classified into two categories: video-based methods and image-based methods.

Video-based human detection methods typically use motion detection techniques to identify human behavior. These methods recognize humans by analyzing the trajectories of moving objects in videos and can be used to detect various human behaviors such as walking, running, cycling, etc. Image-based human detection

methods, on the other hand, typically use target detection techniques and analyze the information to recognize humans in a single image.
Both methods can be further divided into feature-based methods and deep learning-based methods. Feature-based methods rely on hand-crafted features extracted manually such as edge, scale-invariant feature transforms (SIFT), and histogram of oriented gradients to detect objects/humans in images. These methods are often faster and more efficient, but they may be prone to errors and may not perform as well on complex images. While deep learning-based methods use deep neural networks (such as convolutional neural networks) or other machine learning models to learn features directly from the data to recognize human silhouettes, faces, body features in images. In videos, deep learning models can be used to analyze human motion trajectories and detect various human behaviors. These deep learning-based methods are widely used in various human detection scenarios, such as video surveillance, face recognition, and autonomous driving, due to their high accuracy and efficiency. However, these methods require large amounts of data and computational resources for training.
In the following, we focus on existing deep learning techniques that use neural network architectures to detect objects and then identify them into classes, including the "person" class that we are interested in for the rest of our work. Object detection generally is categorized into Proposal-based object detectors (two-stage) and one-shot object detectors (one-stage).

4.2.1.1 Two-stage object detectors

Two-stage object detectors are a type of method that consists of two independent stages: a region proposal stage and a classification stage.
In the region proposal stage, the algorithm generates a set of potential regions where objects may be present. These regions are typically generated using either sliding windows or selective search techniques. The sliding window detector involves creating multiple windows at different locations to detect different objects,

while selective search involves grouping pixels into regions based on texture and color and then merging the regions to form candidate ROIs.
The objects are classified in the proposed regions by a classifier algorithm during the second phase. The most used classifier is convolutional neural network (CNN). The major strength of these methods is their capacity to handle a wide variety of shapes and locations in the region proposal stage, which may process objects with various aspect ratios and spatial locations. These methods, however, might be computationally demanding because they require the generation and classification of numerous region proposals.
Some examples of two-stage object detectors include the R-CNN family of methods (e.g., R-CNN, Fast R-CNN, and Faster R-CNN), and the Cascade R-CNN method. These methods are widely utilized in both academia and industry, since they have been shown to be efficient for a range of object detecting applications.

R-CNN (Region-based Convolutional Neural Network) [203] is the original method proposed in the R-CNN family. It generates region proposals using a selective search technique, and then classifies the objects in those regions using a convolutional neural network (CNN). Due to the computationally expensive nature of the selective search method and the need to train the CNN independently for each image, the main drawback of this approach is that it is relatively slow.
Fast R-CNN [88] enhances upon the original R-CNN approach by using a single CNN to classify the objects in all the region proposals. Compared to the original R-CNN method, this reduces the training time and computational cost. The region proposal stage is, however, still moving very slowly.
Faster R-CNN [89] enhances the Fast R-CNN method (figure 4.1) even further by using a region proposal network (RPN) to provide region proposals. The RPN is trained together with the rest of the network, which accelerates and improves the effectiveness of the entire process. The main drawback of this method is that it consumes more memory and computational resources compared to the R-CNN family

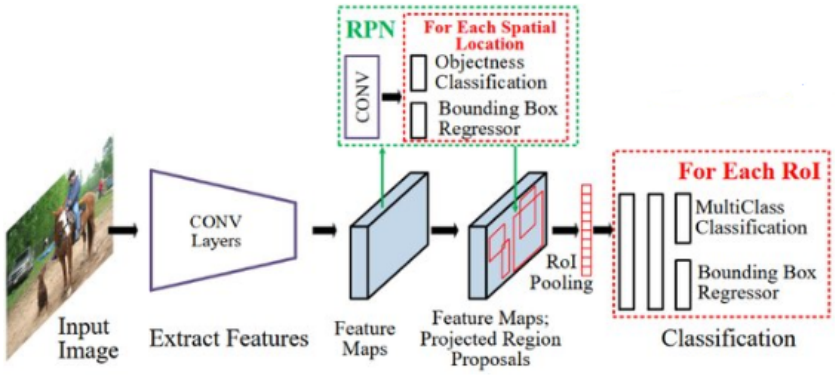

Figure 4.1: Faster R-CNN architecture [204].

algorithms. **Cascade R-CNN** [205] is an advanced version of Faster R-CNN that implements a cascaded network architecture to enhance object detection accuracy and robustness. This approach involves training multiple stages of detectors built sequentially, with each stage acting as a filter that refines the predictions of the previous stage and focuses on difficult examples. The first stage is trained on the initial dataset, and subsequent stages are trained on hard examples misclassified by previous detectors. The cascaded structure of the model allows it to become more discerning and better manage difficult detection cases.

4.2.1.2 One-stage object detectors

One-stage object detectors are a type of method that consists of a single stage for object detection. In these methods, the algorithm directly predicts the object classes and locations in the input image, without the need for a region proposal stage. Some examples of one-stage object detectors include the YOLO (You Only Look Once) and SSD (Single Shot Multibox Detector) methods. The one-stage methods are known for their fast inference times, as they do not require the genera-

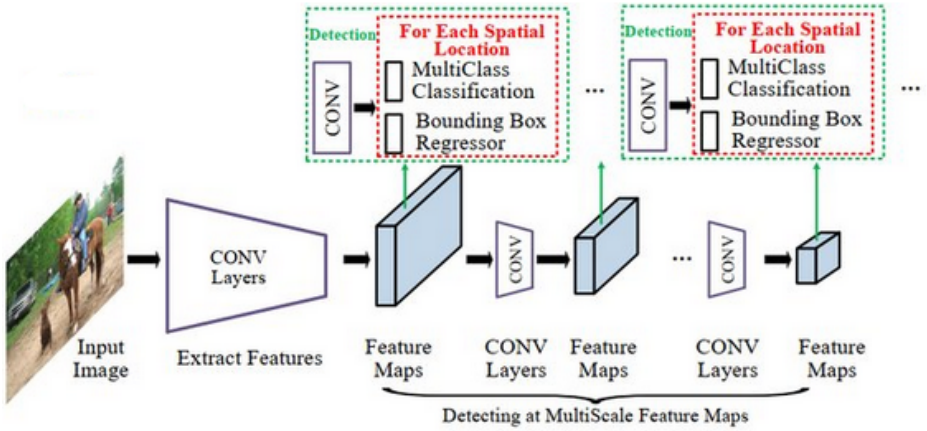

Figure 4.2: SSD architecture [204].

tion of multiple region proposals.

YOLO is known for being fast and accurate, while SSD is known for its ability to handle a wide range of object sizes and aspect ratios. One key difference between YOLO and SSD is the way they process the input image. YOLO divides the image into a grid of cells and predicts bounding boxes and class probabilities for each cell. SSD, on the other hand, uses a set of default boxes at different scales and aspect ratios, and predicts the offsets to these boxes and class probabilities for each default box. Another difference is the way they handle scale variations. YOLO is more sensitive to scale changes and may not perform as well when the objects in the image are significantly larger or smaller than the objects it was trained on. SSD, on the other hand, is able to handle a wider range of scales due to its use of default boxes at multiple scales.

One area of research for improving the accuracy of one-stage object detectors is the use of multi-scale feature maps, such as those used in the FPN (Feature Pyramid Network) [206] architecture. These feature maps can help to improve the detection of small objects, which are often poorly detected by single detectors.

Another area of research is the use of loss functions, such as the Focal loss and the RetinaNet [207], to address the issue of category imbalance during training. Category imbalance occurs when the number of samples in each class is not balanced, leading to poor performance in underrepresented classes. The Focal loss and RetinaNet address this issue by down-weighting the loss for well-classified examples and focusing on the difficult examples during training.

4.2.1.3 Conclusion

The choice between using a one-stage or two-stage object detection method will depend on the specific requirements and resources available for the task. Two-stage methods are generally more accurate, but also require more computational resources and are slower. On the other hand, one-stage methods typically offer faster real-time processing speed but may require a trade-off in terms of accuracy, especially when it comes to detecting small objects or objects that are too close. These methods are known for their fast inference times, as they do not require the generation of multiple region proposals, but they may be less accurate and prone to problems in certain situations.

4.2.2 The human detection method adopted in our system

In our work, we faced the challenge of deciding between using a one-stage or two-stage object detection method. On one hand, we wanted to accelerate the execution time, which led us to consider one-stage detectors. On the other hand, we wanted to use the most accurate method, as the accuracy of the detection would impact the subsequent results and estimations. Therefore, to address this challenge, we experimented with both one-stage and two-stage methods and ultimately chose the best model for our specific case. After reviewing the literature, we selected Faster R-CNN and YOLO as the top performing options in each category. We noticed

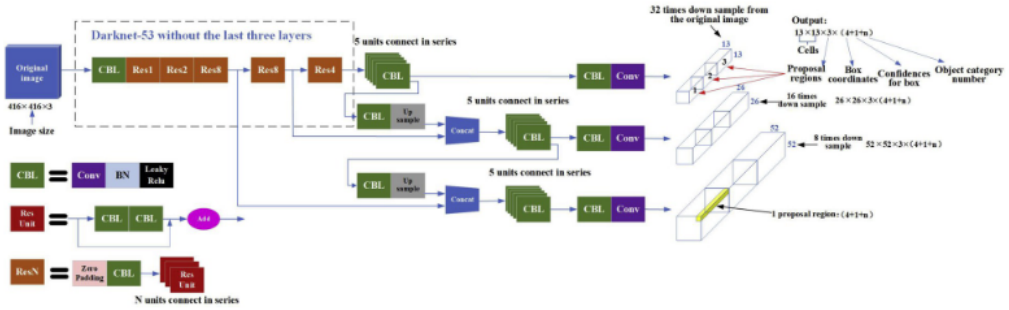

Figure 4.3: YOLO-v3 architecture [209].

by testing these models in our images that the bounding box shapes detected by Faster R-CNN varied significantly between frames. The estimated depth, which will be discussed in the next chapter (Chapter 5.5), exhibits fluctuations even when the person is standing still. In contrast, YOLO was able to provide more stable and accurate detection, making it the better choice for our purposes.

We used the third version of YOLO (YOLO-v3) [208], which was released in 2018 and has improved upon the previous versions in terms of accuracy and speed. It is a fast and accurate object detector that is widely used in various applications such as self-driving cars, video surveillance, and face recognition. The YOLO-v3 architecture predicts bounding boxes using dimension clusters as anchor boxes. The network predicts four coordinates for each bounding box (bbox): the 2D image coordinates of the top-left pixel of the bbox, the width and height of the bbox, and the confidence score. Darknet-53 was used for feature extraction

4.3 Multi-person tracking

Multi-person tracking is the task of tracking multiple people in a video sequence. It is a challenging task due to occlusion, appearance change, motion blur, and background clutter. However, it is an important task with a wide range of applications, such as video surveillance, human-computer interaction, and sports analysis.

There are a number of different approaches to multi-person tracking, including track-by-detection, track-by-association, and social force models. The choice of approach depends on the specific application and the available resources.

4.3.1 Existing tracking methods

Tracking people in images is a crucial aspect of video surveillance, as it allows for the analysis of human behavior and the identification of potential threats. There are very diverse approaches to tracking people in images, including traditional methods such as Kalman filters, particle filters, Mean-Shift, and optical flow, as well as more recent methods based on deep learning. These algorithms can be used to track different parts of the body (head, face, legs, etc.) or the entire person, and sometimes involve representing the person as a skeleton, contour, or other type of representation. The state-of-the-art methods for tracking objects in images and videos can be categorized into four classes:

- **Particle filters** are a type of probabilistic method that uses a set of samples to estimate the state of an object over time. Examples include the Kalman filter [210] used for predicting the position of an object in a continuous and reliable manner over time, and the particle filter [211], which is used to track the position and movement of an object by representing it as a set of discrete particles, each representing a possible state of the object. Particle filters are particularly useful for dealing with non-linearities and the non-Gaussian nature of noise.
- **Correlation-filtering-based tracking** are a type of filter that uses cross-correlation to track an object in an image or video. Examples include the Kernelized Correlation Filter (KCF) [212] which is based on the concept of correlation filters to track the position and movement of an object in a video, the discriminative scale space tracker (DSST) [213] uses a discriminative model to track an object, which outperforms KCF filter in terms of accuracy and robustness, and the spatio-temporal context (STC) [214] method that improves object tracking by taking into

account the relationship between the tracked object (the target) and its surroundings (context) in a Bayesian framework. STC can be used in combination with other tracking techniques, such as particle filters, to improve the accuracy of tracking results.

- **Optical flow techniques** measure the pixel-level motion between two images and can be used to track objects in a video as Lukas- Kanade algorithm in [215] by the intensity changes of frames, and the Horn-Schunck method [216] based on the minimization of an energy functional which is composed of two terms: a smoothness term and a brightness constancy term.

- **Deep learning-based methods** have become increasingly popular in the field of object tracking due to their ability to learn complex patterns and features in the data and adapt to changing conditions. These include DaSiamRPN [217], DiMP [218], MDNet [219], SiamFC [220], and SiamRPN++ [221].

4.3.2 The tracking method adopted in our system

In our situation, we already had multiple deep learning models integrated and needed to run the system in real-time. Therefore, due to their high computing needs and requirement for a large dataset of annotated images or videos, adopting deep learning-based algorithms for tracking was not a viable option for us. Instead, we had to consider alternative methods that were more effective and suitable for our real-time constraints. As an alternative, we considered traditional methods such as Optical flow techniques, Correlation-filtering based tracking, and Particle filters. However, these methods may be limited in terms of their ability to track multiple objects efficiently or accurately identify target objects in cluttered environments. They can also be challenging to set up and tune correctly due to their sensitivity to the motion and measurement models chosen, which may result in unsatisfactory results in some cases.

The Hungarian optimization algorithm [222] offers an effective solution that could

accurately detect targets even when they are partially occluded or moving at varying speeds, while also requiring minimal computational resources. The Hungarian optimization algorithm is preferred for object tracking due to its robustness and accuracy, and is capable of handling large-scale problems with a high degree of complexity, making it suitable for complex scenarios such as multi-object tracking. It is a method for solving the assignment problem, which involves finding the optimal way to assign a set of tasks to a set of resources in order to minimize the cost. In the context of people tracking, the Hungarian optimization algorithm is sometimes used to assign tracked objects to detected objects in order to reduce the number of tracking errors. It has the benefit of being comparatively fast and efficient, which is advantageous for real-time applications. It can also deal with scenarios when there is a mismatch between the number of tracked and detected objects, which can be problematic for other approaches.
This method proved successful for us since it allowed us to execute the system in real time without compromising on accuracy.
To ensure accurate tracking, every person is given a one-of-a-kind ID, which is randomly generated and never replicated, that must be associated with the tracking system which utilizes the Hungarian optimization algorithm for high accuracy in real time and safeguards the system from being deceived by two or more people with the same ID at the same time.

4.4 2D human pose estimation

The process of determining a person's body's location and orientation in an image or video is known as a 2D posture estimation. This is a crucial enabler for many applications, including activity analysis, action recognition, and human-computer interaction. There are various approaches to 2D pose estimation, from traditional methods based on hand-crafted features to more recent methods based on deep learning, such as Mask R-CNN and OpenPose.

A major benefit of modern deep learning techniques is their capacity to automatically learn feature representations from data. However, the quality of the features is closely linked to the network architecture, thus the area of network design warrants a thorough exploration. Consequently, network architecture design approaches attempt to extract robust features by exploring a variety of network designs to solve human pose estimation. In this section, we aim to provide a comprehensive overview of these approaches while focusing on their network architectures.

4.4.1 Existing 2D pose estimation networks

In chapter 3, we reviewed various approaches used for human pose estimation, including methods for single 2D human pose estimation, which can be categorized into keypoint regression and heatmap estimation methods. In their survey paper, Dang et al. [27] compared the techniques of direct keypoints regression and heatmap estimation. According to their findings, each approach has its own advantages and limitations, making it difficult to determine a definitive superior method. Direct regression offers the advantage of being fast and trained in an end-to-end manner. However, it can be challenging to learn the mapping accurately. On the other hand, heatmap-based methods can handle complex scenarios effectively. Although the use of high-resolution heatmaps can enhance precision, it requires a significant amount of memory capacity. Despite this requirement, heatmap-based strategies have gained increasing popularity in recent years.

To illustrate, in the LSP dataset [223], the DeepPose approach [60] employed direct regression and achieved an average Percentage of Correct Parts (PCP) of 61%. However, this result was lower compared to heatmap-based methods, even when cascaded regressors were used to refine predictions. Although the use of cascaded regressors increased learning complexity, it also diminished the model's generalization capability.

In deep learning-based pose estimation, ResNet [91], Convolutional Pose Machine

(CPM) [65] and Stacked Hourglass Network [66] are commonly utilized as backbone models for predicting the XY locations of joints based on heatmaps. ResNet is renowned for capturing comprehensive contextual information of each human body joint through residual mapping and skip connections. DeeperCut [96], which added intermediate supervision to ResNet-152, achieved a PCK score of 90.1% on the LSP dataset, while part heatmap regression attained 90.7% accuracy. The skip connections in the Stacked Hourglass Network enable precise estimation by preserving spatial information and merging functionalities at different scales within the same resolution layers.

Differently, CPM employs distinct pipelines operating independently at multiple resolutions before combining their functionality within the network. This approach also incorporates supervised intermediate heatmaps and a loss function to capture both global and local information through network learning, enhancing its performance

In the FLIC dataset [198], CPM network and Stacked Hourglass Network achieved high precision for wrist and elbow estimation. Specifically, the CPM network achieved 95.03% wrist estimation precision and 97.59% elbow estimation precision using the PCK@0.2 metric, while the Stacked Hourglass Network achieved 97.0% wrist estimation precision and 99.0 % elbow estimation precision.

By combining the Stacked Hourglass and Inception-Resnet architectures, Ning et al. [68] achieved significant results. They estimated 96.9% of the keypoints on the LSP dataset and 91.2% on the MPII dataset [195] using the PCKh evaluation metric. Notably, the Stacked Hourglass model outperformed DeepCut and DeeperCut in terms of PCK values.

On the MPII dataset, both DeeperCut and CPM achieved a PCK@0.5 score of 88.52% by incorporating intermediate supervision to the ResNet-152 model. Additionally, the Stacked Hourglass Network achieved a PCK@0.5 score of 90.9%. Furthermore, the integrated Stacked Hourglass and Inception-ResNet model [68] reached a PCK@0.5 score of 91.2

The High-Resolution Network (HRNet) [101] has shown impressive performance on both the MS-COCO and MPII datasets. Its ability to maintain high-resolution representations of features across the network eliminates the need for feature restoration in the output. On the MS-COCO dataset, HRNet achieved 77% using the Average Precision (AP) metric and 82% using the Average Recall (AR) metric. Moreover, the smaller version of HRNet, HRNet-w32, which offers improved efficiency in terms of model size (# Params) and computation complexity (GFLOPs), achieved a PCKh@0.5 score of 92.3% on the MPII dataset. Additionally, the HRNet model is not limited to multi-person pose estimation datasets but is also widely adopted in 3D human pose estimation methods as a 2D pose detector.

When it comes to multiple-person pose estimation, top-down approaches typically offer higher precision and accuracy compared to bottom-up methods. These top-down methods achieve superior results by applying a pose estimator individually to each person in the scene. However, this approach can be less effective and slower due to the need for separate processing for each individual. Additionally, top-down methods may struggle to capture spatial dependencies that exist across different people, requiring global inference to handle such scenarios.

On the other hand, bottom-up approaches demonstrate greater robustness in scenes with numerous individuals in close proximity. These methods excel in scenarios where multiple people are present, as they focus on detecting and associating body parts without explicitly identifying each individual beforehand. By considering the relationships between body parts and their spatial connections, bottom-up approaches can effectively handle complex multi-person poses.

4.4.2 The 2D pose estimation method adopted in our system

The popular choice for researchers on 2D human pose estimation is High-Resolution Network (HRNet) [86] due to its high accuracy, efficient training and inference, and flexibility. It has been demonstrated to achieve state-of-the-art results and

strong performance on several benchmarks for this task.

The HRNet architecture 4.4 starts from a high-resolution subnetwork and gradually adds lower resolution subnetworks, by decreasing the resolution to half and increasing the width to double in separate branches that are connected in parallel, which allows HRNet to maintain a high-resolution representation throughout the process. The input image size is 256×192 or 384×288, which produces 17 heatmaps (one per keypoint) of size 64×48 or 96×72, respectively. This multi-resolution architecture of HRNet allows learning features at different scales during training. This, along with its lightweight network architecture, also makes it suitable for real-time applications. In addition to its performance, HRNet is also versatile, as it can be used for both single-person and multi-person pose estimation. All of these factors make HRNet an attractive choice for researchers in this field, including us. The authors proposed two variations of HRNet: a small network containing 32 channels (HRNet-W32) and a large network with 48 channels (HRNet-W48).

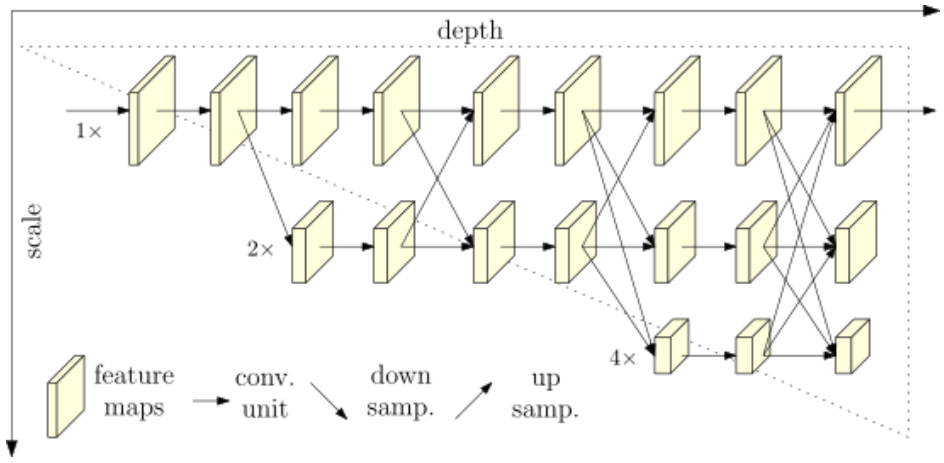

Figure 4.4: HRNet architecture [86].

4.5 3D pose estimation from 2D joints

3D pose estimation from 2D joints refers to the process of determining the three-dimensional position of a person's joints in an image, using two-dimensional joint locations as input. This can be achieved through various techniques, such as binary decision trees [125], deep learning [63, 123, 124, 127–131], and dictionary learning [121, 132]. Some approaches also involve a two-stage process, where the first stage estimates the 2D pose from monocular images, and the second stage lifts the 2D joint locations to 3D poses, followed by an optimization step [126, 133]. This section presents an overview of the current two-stage 3D pose estimation techniques based on deep learning that estimate the 3D pose of a person from 2D joints in an image.

4.5.1 Existing 3D pose estimation methods

Over the years, numerous approaches have been developed for estimating the 3D pose of a human body as seen in chapter 3, which can be also classified based on the input data they use. Specifically, two major types of methods have been proposed in the literature: monocular image-based methods, outlined in Table 4.1, and monocular video-based methods, outlined in Table 4.2.

In practice, it is generally observed that the two-stage approach of dividing 3D human pose estimation into two stages, including 2D pose estimation as an invariant feature and lifting to 3D poses, is more accurate and feasible than building an end-to-end network for direct 3D joint regression. Many state-of-the-art methods predict 3D poses from intermediate 2D coordinates obtained from monocular images. These methods include the ones proposed by [63, 159, 160, 164, 226–233] (source: [234]).

These methods use different techniques, such as concatenation of 2D pose estimation results and image features [118], multi-stage lifting by an unimodal Gaus-

Method	Description
Direct Regression [124, 150]	A deep learning approach that involves training a neural network to directly regress 3D joint locations from 2D image data. Typically requires a large amount of annotated data to train the network effectively.
Two-stage Approach [120, 224]	A two-stage approach that first estimates 2D joint locations and then lifts them to 3D using a 3D pose model. Often involves a convolutional neural network (CNN) to predict 2D joint locations and then solving an optimization problem to obtain the final 3D pose.
Pose-guided Joint Regression [57, 163, 225]	A deep learning approach that uses pose information to guide the regression of 3D joint locations. Typically involves first estimating the pose (e.g. body orientation) and then using that information to help predict the joint locations.
Generative Models [81]	A family of methods that involve learning a probabilistic model of the 3D pose distribution given 2D image data. Typically involves a generative adversarial network (GAN) or variational autoencoder (VAE) to learn the model.

Table 4.1: Monocular Image-Based Methods

sian 3D pose model [120], 2D to 3D distance matrix regression [124], feedback from 3D re-projection and use a discriminator to judge feasibility of the generated 3D pose [166], lifting 2D pose to 3D by a neural network [85], exploiting body part images to predict depth [235], fusing 3D image cues with 2D joint heatmaps [123], multi-source discriminator [128], ordinal depth as supervision [130], estimating pixel-wise 3D surface correspondence [236], part-centric heatmap triplet [237], predicting 3D poses from low-DOF to high-DOF [238], using different filters for feature extraction [239], deep conditional variational autoencoder [224], generating

multiple corresponding feasible 3D pose solutions [240], jointly understanding holistic scene and estimating 3D human pose [241], lifting by semantic graph convolutional network [63] unsupervised lifting with a discriminator [242], and learning a consistency measure between 2D observations and a proposed world model by a neural network [243]. The methods aim to reduce lifting ambiguity and generate realistic 3D pose solutions.

Keeping this in mind, we will now explore the use of video-based techniques that make use of multiple frames to enhance the inference of 3D pose, overcome ambiguities, and reduce errors caused by occlusion or other factors by leveraging motion cues. These methods differ in their ability to leverage temporal and motion information and the complexity of the models used to estimate poses.

Method	Description
Temporal Smoothing [202, 232]	A post-processing technique that involves applying temporal smoothing to the 2D joint locations over multiple frames to obtain a more accurate 3D pose estimate using filters such as a Kalman filter or a variant thereof.
3D Kinematic Model Fitting [150, 244, 245]	The approach involves an algorithm to match the 2D joint locations in every frame of a video with a 3D kinematic model of the human body. This usually requires optimizing the model to minimize the difference between the predicted 3D joint locations and the actual 2D joint locations detected in the video frames.
Temporally Consistent Pose Estimation [246, 247]	This technique entails estimating the motion and pose of the human body in 3D across various frames of a video. The procedure often involves constructing a continuous trajectory to represent the motion and ensuring that the estimated poses are consistent throughout the video.
Pose-guided Joint Regression [248]	Similar to the method used in monocular images, this method uses pose information to guide the regression of 3D joint locations. However, in the case of monocular videos, this may involve estimating the temporal evolution of the pose and using that information to help predict the joint locations.
Video-based Pose Refinement [245, 249]	The approach employs a pre-trained model for 3D pose estimation on a single frame of video, followed by enhancing the estimations across multiple frames by including temporal information. Commonly, the technique applies a fusion of optical flow and temporal consistency restrictions to improve the precision of the pose estimations.
Generative Models [250, 251]	Similar to the method used in monocular images, this method involves learning a probabilistic model of the 3D pose distribution given the video data. However, in the case of monocular videos, this may involve modeling the temporal evolution of the pose distribution over time.

Table 4.2: Monocular Video-Based Methods

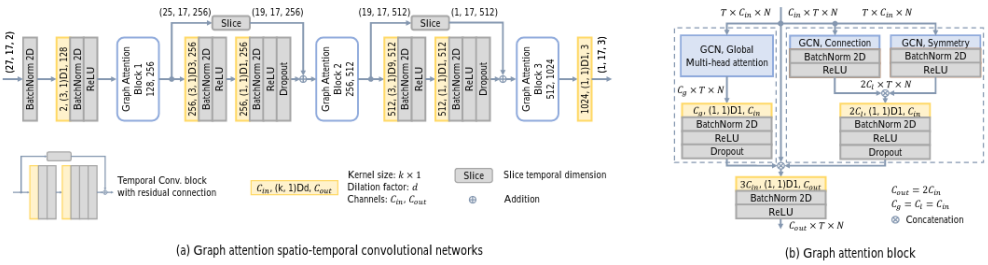

Figure 4.5: GAST-Net architecture [253].

4.5.2 The 3D pose estimation method adopted in our system

Recent research has shown that the two-stage approach is the most precise method for 3D human pose estimation. This approach utilizes deep neural networks to estimate 2D keypoints from an RGB image and subsequently map them to their corresponding 3D coordinates. The accuracy of this approach has been shown to exceed that of other methods, particularly those that use a deep neural network in the second stage.

To analyze and interpret video input, most models rely on Temporal Convolutional Networks (TCNs). These networks were initially introduced for action segmentation by Lea et al. [252]. GAST-Net (graph attention spatio-temporal network) [253] is an architecture that draws inspiration from VideoPose3D [160] and aims to predict 3D poses from 2D keypoints.This network comprises dilated temporal convolutional networks and a graph attention block. The TCNs can handle long-term patterns and have an extended memory, while the graph attention block includes two spatial attention networks: a local spatial attention network that models the hierarchical and symmetrical structures of the human skeleton and a global spatial attention network that extracts global semantic information in an adaptive manner. This allows GAST-Net to better encode the spatial characteristics of the human body.

GAST-Net was selected as the best option for 3D pose estimation from 2D keypoints in our framework, as it strikes a balance between the number of frames needed for processing and the precision of the estimation. In fact, the most accurate methods for monocular videos from Human3.6m, (the largest database of 3D human pose estimation) are temporal convolution [160] trained in semi-supervision learning, Attention 3D Human Pose [247], which identifies significant frames and tensor outputs from each layer using the attention mechanism, the RIE paper [159], which improves the accuracy through relative information encoding that yields positional and temporal-enhanced representations, and Anatomy3D [254], which estimates the 3D skeleton by predicting bone orientation and length. These methods reached the MPJPEs (defined in chapter 3) of 46.8 mm, 45.1, 44.3, and 44.1, respectively. However, these methods require a large number of input frames (243 frames). This is very costly in terms of memory and processing time, and it increases the delay between the image display and the result, making it unsuitable for real-time processing.

Furthermore, tracking multiple individuals over long periods of time is more complex and prone to errors.
On the other hand, approaches that employ fewer frames have higher errors. For example, models such as VIBE [255] and TP-Net [225] which rely on a limited number of frames, 16 and 20 respectively, resulted in substantial errors with MPJPE of 65.6 mm and 52.1 mm respectively Even the Trajectory space factorization method [232] which used 50 frames scored an error of 46.6 mm. GAST-Net achieved an MPJPE of 46.2 mm despite utilizing a limited number of frames (27 frames), making it a more practical solution for real-world contexts.

GAST-Net is a sophisticated machine learning algorithm that utilizes a combination of convolutional neural networks (CNNs) and graph attention layers (GATs) to accurately detect and map human body parts such as arms, legs, and torso in three-dimensional space using only two-dimensional images captured by cameras or other sensors. This approach enables more accurate tracking compared to tradi-

Method	Number of Frames	MPJPE (mm)
MotionBERT [256]	243	37.5
MixSTE [257]	243	39.8
P-STMO [192]	243	42.1
Anatomy3D [254]	243	44.1
RIE Methodpaper [159]	243	44.3
Attention 3D Human Pose [247]	243	45.1
VideoPose3D [160]	243	46.8
Spatio-Temporal Network [228]	128	40.1
GAST-Net [253]	27	46.2
Trajectory space factorization [232]	50	46.6
TP-Net [225]	20	52.1
VIBE [255]	16	65.6

Table 4.3: Summary of methods and their accuracy in terms of MPJPE on Human3.6m and number of frames used.

tional methods that only rely on two-dimensional data points.

In conclusion, GAST-Net has allowed us to accurately estimate the 3D poses of multiple individuals in real-world scenarios. It utilizes both temporal convolutional networks (TCNs) and graph attention layers (GATs) to effectively identify and map body parts onto 3D coordinates. Compared to state-of-the-art methods, GAST-Net provides a good balance between the number of frames required to process and the estimation precision, making it suitable for real-time applications.

4.6 Conclusion

In this chapter, we have thoroughly described the various steps involved in 3D human pose estimation and the reasoning behind our choice of network models. To carry out human detection, we utilized YOLO-v3; tracking was performed with the help of the Hungarian optimization algorithm; and 2D human pose estimation was performed using HRNet. These methods were combined with GAST-Net to obtain 3D pose estimation from 2D poses. However, it is essential to keep in mind that this approach provides 3D root-relative human pose estimation, which may not be as suitable for real-world applications as compared to absolute pose estimation. In the upcoming chapters, we aim to address this limitation by making substantial advancements and contributions in the field of 3D human pose estimation. Our focus will be on developing methods to provide absolute pose estimation, which will be more reliable and practical for a wider range of use cases.

CHAPTER

Proposed approach for 3D absolute pose estimation

5.1 Introduction

In the previous chapter, 3D root-relative human posture estimation approaches were covered. The estimation relative to the root body joint, also known as person-centered pose estimation, refers to the estimation that determines the distance between the root point (the person's pelvis) and other body points.
For single person 3D pose estimation, we only need to estimate the relative depth at the root key point, which was discussed in the previous chapter 4.However, when the scene includes multiple people, it becomes more important to also estimate the absolute position of each person relative to the camera. This requires estimating the depth of key points with respect to the camera, in addition to estimating the relative 3D pose of each person's body.
There are many commercially available devices that can provide depth information, such as ToF sensors. Time-of-flight (ToF) sensors are a type of device that can be used to measure the distance or depth of an object in 3D space. The Time-of-flight technology is based on the measurement of optical distance. The way they operate is by emitting a light pulse and timing how long it takes for the light to travel from the sensor (light source) to the object and back. The object's distance or depth can be determined using the time it takes for light to travel over this distance.
$DISTANCE\_TO\_OBJECT = SPEED\_OF\_LIGHT \times TIME\_OF\_FLIGHT$
There are many sensors available commercially that are used in various applications. For example, Kinect sensors use structured light ToF technology and are known for their high accuracy and reliability. They are commonly used in gaming and robotics applications, but are generally most suitable for short-range depth sensing in indoor environments, typically up to a range of around 2 meters. This is because Kinect sensors rely on the detection of reflected light, and the accuracy of the measurement can be affected by factors such as ambient light and surface reflections.
LIDAR sensors, on the other hand, use laser ToF technology and are commonly

used in autonomous vehicles and other applications. They are especially useful in outdoor environments or at longer ranges, as they are fast, accurate, and capable of measuring long distances and in a wider range of lighting conditions. LIDAR sensors are often preferred over other depth sensing technologies for these reasons. They, meanwhile, consume a lot of energy power, making them unsuitable for consumer products.

There are other technologies that can be used to provide depth information, such as structured light sensors, and laser scanning. These technologies work by using different techniques to measure the distance or depth of objects in 3D space, and they may be more or less suitable for different applications depending on the specific requirements and constraints. However, they are currently not as widely used doe to their higher cost and lower accuracy.

Stereo cameras are a technology that is widely used for depth sensing, particularly in the field of 3D human pose estimation. They consist of two or more cameras that are positioned such that they each capture a different view of the same scene. The images from the different cameras are then used to calculate the distance or depth of objects in the scene. This is also known as triangulating the scene. There are various methods proposed using this technology for measuring the distance or depth of people in 3D spaces. The best choice for a particular application will depend on the specific requirements and constraints of the application, as some technologies may be more or less suitable for different purposes.

In our thesis, we have chosen to use monocular cameras for depth estimation due to constraints in the application that will be explained in chapter 7. Monocular cameras are a type of camera that uses a single lens to capture images and are widely used in security surveillance and industrial security applications. However, estimating depth using monocular cameras can be a challenging task because of the limited information available from a single viewpoint. When a 2D image is captured, it loses the depth and scale information that is present in the original scene. In addition, the process of estimating depth from a 2D image is often

ambiguous and can be affected by a variety of factors, including the camera focal length, camera pose, person size and posture, and so on. As a result, much of the research in this area has focused on estimating the relative pose of each individual, rather than the absolute depth of the scene.
However, absolute pose estimation is an important problem that warrants further study because it is closely related to real-world scenes. It has been difficult to measure this distance accurately because the distance between two objects in a real-world scene is not necessarily constant. By determining these distances and relationships between individuals, it may be possible to develop more complex applications that help to better understand the interaction between people and analyze their behaviors. For example, such applications could be used in security surveillance to detect and track the movement of individuals in a crowded scene, or to identify patterns of behavior that may indicate suspicious activity. In social science research, depth sensing technology could be used to study group dynamics and communication patterns in order to better understand how people interact with one another. In conclusion, the ability to accurately estimate the absolute pose of people in a scene can provide valuable insights and enable the development of a wide range of applications.

In this chapter, we review current research on single-view, multi-person absolute position and pose estimation tasks, and then propose a solution that is expected to meet the initial requirements.

5.2 Existing techniques

There have been relatively few studies on estimating the 3D poses of multiple people from a single RGB image. These methods can generally be divided into two categories: top-down and bottom-up approaches.
Top-down 3D human pose estimation methods [119, 153, 258] are those that start

by detecting and cropping each person in a bounding box and then estimating person-centric 3D full-body joints [150, 160, 259]. These methods have shown promising performance, but they can be affected by inter-person occlusions and close interactions because they process each person independently. Rogez et al. introduced LCR-Net [135] and LCR-Net++ [153], which classified bounding boxes into a set of K-poses and refine them using a regressor. The Localization-Classification-Regression architecture consists of three components that share convolutional feature layers and are jointly trained: a pose proposal generator, a classifier, and a regressor. The proposal generator suggests candidate poses at different locations in the image, the classifier scores the different pose proposals, and the regressor refines pose proposals in both 2D and 3D. The final pose estimation is obtained by integrating neighboring pose hypotheses. Benzine et al. [260] proposed PandaNet, a single-shot anchor-based approach for 3D pose estimation. It performs bounding box detection and 2D and 3D pose regression in a single forward pass, without requiring any post-processing to regroup joints. To handle overlapping people, PandaNet uses a Pose-Aware Anchor Selection strategy and optimizes weights for different people sizes and joints to improve training efficiency. The proposed pipeline of Moon et al. [259] consists of human detection, absolute 3D human root localization, and root-relative 3D single-person pose estimation modules. The RootNet model in is used for absolute 3D human root localization, while the PoseNet [129] model is used for root-relative 3D single-person pose estimation. The end-to-end HDNet (Human Depth Estimation Network) [261] follows the same pipeline. It estimates the depth of a person in an image by using a combination of a Feature Pyramid Network [206] for general feature extraction and separated multi-scale feature extraction for pose and depth estimation, and a Graph Neural Network to propagate and aggregate features for the target person's depth estimation. The estimated depth is represented as a bin index and can be transformed into a continuous value using a soft-argmax operation. Similar to the above methods for depth estimation, HMOR (Hierarchical Multi-person Ordinal

Relations) [262] employs an integrated top-down model to estimate human bounding boxes, depths, and root-relative 3D poses simultaneously, with a coarse-to-fine architecture that, instead of using image features as the above methods for depth estimation, hierarchically estimates multi-person ordinal relations of depths and angles which captures body-part and joint-levels semantics while maintaining global consistency to improve the accuracy of depth estimation. The proposed framework for 3D multi-person pose estimation in [263] combines graph convolutional networks and temporal convolutional networks to estimate camera-centric poses without requiring camera parameters. It includes GCNs that estimate frame-wise 3D poses and TCNs that enforce temporal and human dynamics constraints to estimate person-centric with a joint-TCN and camera-centric 3D poses across frames with a root-TCN.

In addition to the methods mentioned above, there are also bottom-up approaches, such as [201, 264, 265] to 3D multi-person pose estimation. These approaches first generate all body joint locations and depth maps and then associate body parts to each person based on the root depth and the relative depth of the parts. Mehta et al. [201] proposed a single forward pass approach that is independent of the number of people in the scene. The method applies temporal and kinematic constraints in three steps to predict occlusion-robust pose-maps (ORPM) and part affinity fields [97]. This method produces multi-person 2D joint locations and 3D pose maps in a single shot. MubyNet is another bottom-up multi-stage framework proposed by Zanfir et al. [266], which first estimated the volumetric heatmaps to determine the 3D keypoint locations and limbs using the confidence scores of all possible connections, and then conducted skeleton grouping in order to assign limbs to various people. Likewise, Fabbri et al. [264] proposed estimating volumetric heatmaps in an encoder-decoder manner. They first produced compressed volumetric heatmaps, which are used as ground truth, and then decompressed them at test time to re-obtain the original representation. Zhen et al. [267] estimated 2.5D representations of body parts and reconstructed 3D human pose in a single-

shot bottom-up framework. Wang et al. [268] also proposed distribution-aware single-stage models to represent 3D poses with a 2.5D human center and 3D center-relative joint offsets in a one-pass solution.

On the other hand, to exploit top-down and bottom-up pipelines strengths, some works combine top-down and bottom-up approaches to address the challenges of inter-person occlusion and close interactions. Such as TDBU_Net framework [178]. Its top-down network estimates human joints for all persons in an image patch, making it robust to possible erroneous bounding boxes, while the bottom-up network incorporates human-detection based normalized heatmaps to handle scale variations. The estimated 3D poses from the top-down and bottom-up networks are then fed into an integration network for the final 3D poses. In addition, the method includes a two-person pose discriminator to enforce natural two-person interactions and applies a semi-supervised approach to overcome the lack of 3D ground-truth data. Similarly, in the approach proposed by [179], a top-down network is used to estimate the 3D joints of all people in an image patch, while a bottom-up network incorporates human detection-based normalized heatmaps to improve robustness in handling scale variations. The 3D pose estimates from both networks are then combined in an integration network to produce the final 3D poses.

5.3 Monocular Root depth estimation

Depth estimation is a computer vision task that involves estimating the depth of objects in a scene using a single camera. It has been intensively researched in the past decade and has seen significant progress using deep learning (DL) approaches. Monocular depth estimation has a wide range of applications, including augmented reality [269–271], target tracking [272, 273], 3D reconstruction [274] and medical imaging [275], among others. There are several ways to acquire depth maps for monocular depth estimation, including using specialized hardware devices such as the Microsoft Kinect, or by estimating the depth using one or more RGB views.

In our research, we focus on monocular depth estimation using a single camera, which is known as monocular depth estimation. Monocular depth estimation is the process of estimating the distance or depth of objects in 3D space using a single camera. There are several methods that can be used for this purpose, including stereopsis [276], structure from motion, and depth from focus. These methods work by analyzing the differences between views, the motion of the camera or objects, or the focus of the camera to infer the depth of the objects in the scene. These methods can be used individually or in combination, depending on the specific requirements and constraints of the application.

Monocular root depth estimation involves inferring information about the distance of the person's central key point (such as the pelvis or lower back) from the camera viewpoint using a single monocular images. While using monocular images to capture depth information can potentially solve memory issues, it can be computationally challenging to capture global scene features such as ground variation or defocus data. This can affect the accuracy of the depth estimation. One of the main challenges of using these monocular depth estimation methods is that it can be difficult to capture enough features in the image to match when the scene has less or no texture. This can make it difficult to accurately estimate the depth of objects in the scene, particularly in situations where there are few or no distinctive features in the image that can be used for matching. To overcome this challenge, researchers have developed various techniques to improve the robustness and accuracy of monocular depth estimation in such scenarios, such as using machine learning or deep learning approaches to learn to recognize and match features in the image [277].

We are particularly interested in the RootNet model, proposed by Moon et al. [259], for the purpose of localizing the absolute 3D human root in images or videos. We detail this model in the next subsection.

5.3.1 RootNet network architecture

RootNet [259] is a model that estimates the 3D coordinates of the human root (the pelvis or lower back) in a camera-centered coordinate space from a cropped image of a person. It does this by separately estimating the 2D image coordinates (x, y) and the depth value (z) of the human root. The 2D image coordinates are then back-projected to the camera-centered coordinate space using the estimated depth value, resulting in the final output of the model. Estimating the depth from a cropped image of a person is difficult because the input does not contain information about the relative position of the camera and the person. To address this issue, RootNet introduces a new distance measure called k defined in equation 5.1, which approximates the absolute depth from the camera to the object using the ratio of the actual area of the human A_{real} (assumed to be constant at 2000mm x 2000mm) and the imaged area of the human A_{img} (calculated by extending the bounding box of the human in the image to a fixed aspect ratio (height:width = 1:1)), given the camera's intrinsic parameters (fx, fy), which are the focal lengths divided by the per-pixel distance factors (pixel) of x- and y-axe. This distance measure is derived from a pinhole camera projection model.

$$k = \sqrt{fx \times fy \times \frac{A_{real}}{A_{img}}} \tag{5.1}$$

However, this value can be incorrect in some cases because it assumes that A_{img} is equal to A_{img} when the distance between the person and the camera is k. To address this issue, RootNet uses image features to correct the value of A_{img} and improve the accuracy of the distance estimate. The model outputs a correction factor (γ) based on the image features, which is used to modify the value of A_{img} . The modified value of A_{img} (A_{img}^{γ}) is then used to calculate the final depth value (k), which represents the distance between the camera and the person.

RootNet is composed of three main components: a backbone network that extracts global features from the input image using a ResNet architecture, a 2D image

coordinates estimation part that uses deconvolutional layers to upsample the feature map and output a 2D heatmap of the root, and a depth estimation part that uses global average pooling and a $1x1$ convolution to output a scalar value (γ) that represents a correction factor for the depth estimate. The 2D image coordinates (x_R, y_R) are obtained by applying a soft-argmax function to the 2D heatmap, and the final absolute depth value (Z_R) is calculated by multiplying k (a value calculated using the ratio of A_{real} to A_{img}) with $1/\gamma$. In practice, RootNet outputs $\gamma' = 1/\sqrt{\gamma}$ directly, and Z_R is calculated by multiplying γ' and k.

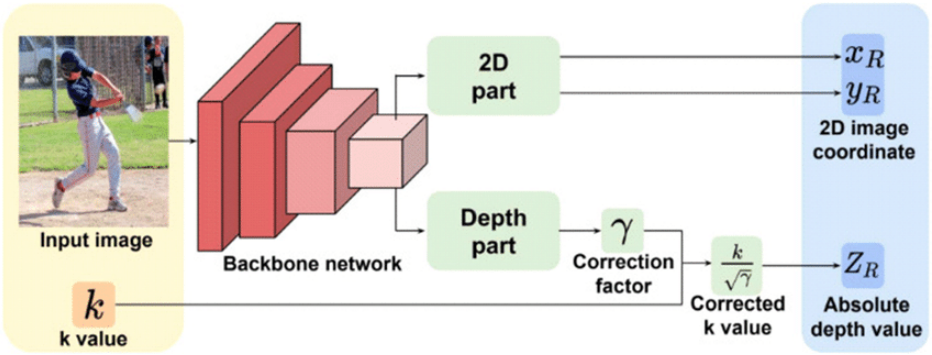

Figure 5.1: Network architecture of the RootNet [259].

5.3.2 Camera-intrinsic parameters

Previously, we examined the original structure of RootNet, which provides us with the depth of the root point (Z_R, referred to as Z^{abs}_{root} in the thesis) and the root-image coordinates (x_R and y_R, referred to u and v in the thesis). In order to obtain the camera-centric coordinates (X^{abs}_{root} and Y^{abs}_{root}), we use projection with the camera intrinsic parameters, such as the focal length and the center of the camera.

The focal length of a camera $((fx, fy))$ is a measure of the ability of the lens to focus light onto the image sensor, expressed in millimeters (mm). A longer focal length corresponds to a more powerful zoom, while a shorter focal length allows a wider field of view. The center of the camera $((cx, cy))$ is the point in the image

where the camera's optics are focused. It is usually located at the center of the image sensor and is used as a reference point to calculate the position of objects in the scene. In 3D pose estimation, the center of the camera is often used as the origin of the coordinate system, with the X, Y and Z axes corresponding to the horizontal, vertical and depth dimensions of the scene, respectively.
The projection in the majority of videos can be modeled as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X^{abs} \\ Y^{abs} \\ Z^{abs} \end{bmatrix} \tag{5.2}$$

where u and v are the image coordinates, X^{abs}, Y^{abs} and Z^{abs} are the camera coordinates.
f, c_x, c_y stands for the focal length and camera centers, respectively. The camera centers are considered as image centers, which is applicable to most cameras.
Thus if the depth estimate of the root Z^{abs}_{root} is known, we can determine the coordinates X^{abs}_{root} and Y^{abs}_{root} of the camera by:

$$X^{abs}_{root} = \frac{Z^{abs}_{root}}{f}(u - c_x) \qquad Y^{abs}_{root} = \frac{Z^{abs}_{root}}{f}(v - c_y)$$

In the following sections, we present the four approaches developed in this thesis for determining the absolute position of a root keypoint, as illustrated in Figure 5.2.

5.4 Two-stage approach 3D absolute pose estimation

5.4.1 Approach structure

It can be difficult to accurately estimate the absolute depth of key points from a video due to variations in person height and the mismatch in scale between the depth point and the full body. To address this challenge, we rely on a commonly

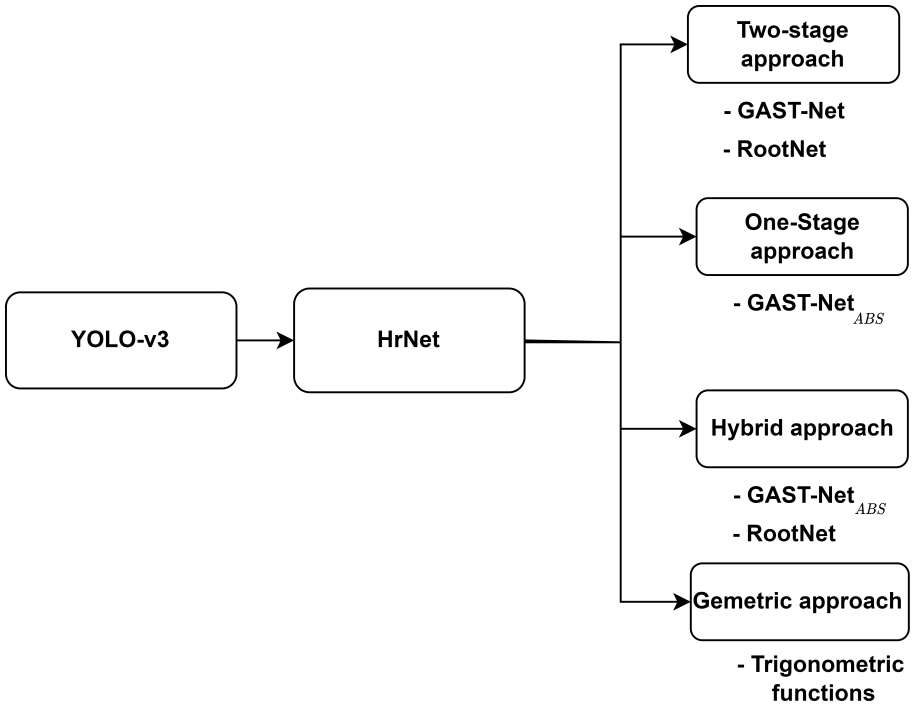

Figure 5.2: Summary Diagram of the Networks Used in the System.

used strategy that uses deep learning techniques to estimate the absolute depth of key-points from a video stream. We referred to this strategy as a two-stage approach, which consists of two stages: one for relative pose estimation and the other for depth determination of one of the key points, the root key point, typically the pelvis or lower back. The first stage of our approach builds upon the pipeline developed in the previous chapter for 3D root-relative human pose estimation, while the second stage involves adding an additional step to estimate the absolute root depth. At the beginning, we use the RootNet model for this. By combining the results of these two stages, we can compute and estimate the full absolute 3D pose of the person in the scene.
As mentioned in the previous section, RootNet is a model that is used to estimate the absolute depth of the root key point in a 3D pose estimation pipeline. It does this by estimating the Z-coordinate of the root key point depth, which is the distance between the point and the camera.

5.4.2 Validation

To validate the effectiveness of our method, we conducted experiments on MuPoTS-3D benchmark dataset. Our method was compared to several state-of-the-art approaches, including HMOR, HDNet, and TDBU. The results show that our method outperformed all other approaches, We use the standard evaluation protocol, following [259] such as $3D-PCK_{abs}$, with a significant improvement of 6.7% over the second best method *TDBU_Net [178]* as shown in the table below 5.1.

It is important to note that the accuracy of the full absolute 3D pose depends on the accuracy of the relative-root key points coordinates and the accuracy of the absolute root coordinates in the 3D pose estimation pipeline. Thus, in addition to the percentage of a correct 3D absolute keypoints metric, we also measure the percentage of a correct 3D keypoint *PCK* and AUC_{rel} to evaluate the root-relative keypoints and AP_{25}^{root} to measure absolute root. These experimental results on the

Method	**Year**	$3D-PCK_{abs}$
3D MPPE PoseNet [259]	2019	31.5
HDNet [261]	2020	35.2
SMAP [267]	2020	38.7
HMOR [262]	2020	43.8
GnTCN [263]	2021	45.7
TDBU_Net [178]	2021	<u>48.0</u>
DAS [268]	2022	39.2
Root-GAST (Two-Stage approach) [278]	2022	**54.7**

Table 5.1: Camera-centric evaluations on the MuPoTS-3D dataset. The best is in bold, the second best is underlined.

MuPoTS-3D dataset are shown in Table 5.2.

Method	**Year**	*PCK*	AUC_{rel}	AP_{25}^{root}
3D MPPE PoseNet [259]	2019	81.8	39.8	31.0
HDNet [261]	2020	83.7	-	39.4
SMAP [267]	2020	80.5	45.5	45.5
HMOR [262]	2020	82.0	43.5	-
GnTCN [263]	2021	<u>87.5</u>	<u>48.9</u>	45.2
TDBU_Net [178]	2021	**89.6**	**50.6**	<u>46.3</u>
DAS [268]	2022	82.7	-	-
Root-GAST (Two-Stage approach) [278]	2022	63.8	30.6	**58.4**

Table 5.2: Person-centric and camera-centric evaluations on the MuPoTS-3D dataset. The best is in bold, the second best is underlined.

As shown in the table, our method outperforms all the comparison methods in terms of the average precision of the root, AP_{25}^{root}, which evaluates the absolute coordinates of the root. However, our root-relative performances, as measured by *PCK* and AUC_{rel}, are not satisfactory and are worse than the best accuracy by 25.8% and 20%, respectively.

5.4.3 Conclusion

The first proposition is a two-stage approach for 3D absolute pose estimation, which combines the Gast-Net to estimate root-relative key-points and RootNet to estimate the absolute root. In 3D pose estimation, the root keypoint is typically the pelvis or lower back, and the absolute depth of this key-point can be used to determine the entire pose of the person in 3D space.
On the MuPoTS-3D dataset, the system adopting the two-stage method outperformed previous methods by more than 12.1 percentage points on AP25^{root}, contributing to a more than 6.7 percentage point improvement on 3D-PCK*abs*. However, we observed that the root-relative keypoints were less accurate by 25.8 percentage points on PCK, which sparked the idea to upgrade the GAST-Net.

5.5 One-Stage approach for 3D absolute pose estimation

5.5.1 Approach structure

The previous section has shown that the two-stage approach for estimating absolute coordinates results in improved performance compared to previous methods. Specifically, it was found that this approach outperformed previous methods by more than 12.1 percentage points on AP_{25}^{root} and contributed to more than 6.7 percentage points on $3D-PCK_{abs}$, when tested on the MuPoTS-3D dataset.

However, we also observed that the root-relative keypoints were less accurate, with a decrease of 25.8 percentage points on *PCK*. This led to the idea of upgrading the GAST-Net model. The original GAST-Net was trained on single-person databases [106], but in order to improve the root-relative keypoints, we decided to retrain the model on both a single-person video database (MPII-3DHP [149]) and a multi-person video database (MuCo-Temp [202]) with the required processing, following the method of [202], in order to produce direct absolute keypoint coordinates. It is worth noting that in the following, we will name the upgraded GAST-Net by *GAST-NET*$_{ABS}$, and refer to this approach by one-Stage approach.
In the literature, TCN-based approaches for 3D human pose estimation that were evaluated on the MuPoTS-3D dataset were trained on the MPII-3DHP and/or MuCo-3DHP databases. The MPII-3DHP database contains videos of a single person recorded in a green-screen studio and captures the characteristics of movement over time. The MuCo-3DHP database, on the other hand, is composed of MPII-3DHP frames containing multiple poses copied into a single frame, and captures the characteristics of multiple people in a single image, as well as the occlusion between them. This provides a diverse training dataset which may improve the model's capacity to be used in real-world applications. However, it would be more efficient to have a database containing videos of multiple people. This would help the model to generalize better to real-world scenarios, where multiple people are often present in a single image or video. for that, [202] proposed MuCo-Temp dataset, a temporal extension of MuCo-3DHP. This extension was generated using the same method as MuCo-3DHP, but it is composed of videos instead of individual frames. This makes it useful for training temporal networks, such as GAST-NET$_{ABS}$, which can take advantage of this dataset to improve its accuracy.

5.5.2 Validation

We evaluate the performance of Gast-Net$_{ABS}$ to determine if we have made improvements to the two-stage approach. The results are shown in Table 5.3.

Method	**PCK**	**AP$^{root}_{25}$**	**3D-PCK$_{abs}$**
Root-GAST (Two-Stage approach) [278]	63.8	**58.4**	54.7
Root-GAST (One-stage approach) [278]	**82.5**	56.8	**56.1**

Table 5.3: Evaluating the Performance of Root-Relative keypoints, absolute root and Absolute Keypoints on the MuPoTS-3D Database. The best is in bold.

This approach allowed us to improve the root-relative keypoints and better handle the problem of multiple people in the scene. As a result, the relative keypoint precision was enhanced from 63.8% with the basic GAST-Net to 82.5% on PCK with the upgrad GAST-Net, contributing to an increase of 1.6 percentage points in absolute points on 3D-PCK*abs* compared to the first methodology of two-stage approach. This highlights the benefit of using a database containing videos of multiple people, as well as the significance of normalizing the 2D pose inputs by the camera's intrinsic parameters before lifting the 2D poses to 3D. This will be explained in further detail in chapter 6. Normalizing the 2D poses allows for correcting for image distortion caused by the camera's lens, variations in image scale, and other factors that can affect the accuracy of the 3D pose estimates. Additionally, by normalizing the 2D poses, it reduces the number of parameters that need to be estimated, as it allows the 3D pose estimation process to make use of prior knowledge of the camera's field of view, image distortion, and other factors. This eliminates the need to estimate these parameters separately, and instead the 3D pose estimation process can use the normalized values to accurately calculate

the 3D pose.
We observed that the AP25^{root}, which measures the root depth estimation of GAST-NET*ABS*, is better than the state-of-the-art in Table 5.1 and 5.2, but still does not perform as well as the first methodology of the two-stage approach. .

5.5.3 Conclusion

The second proposition is a one-stage approach for 3D absolute pose estimation, which is based on a single model to estimate the absolute keypoints instead of combining the GAST-Net to estimate root-relative keypoints and RootNet to estimate the absolute root. For this, we upgraded the model GAST-Net to GAST-Net_*ABS*. On the MuPoTS-3D dataset, the system using the one-stage method outperformed previous methods by 8.1 percentage points on 3D-PCK*abs*, compared to the most accurate state-of-the-art method TDBU_Net [178]. The AP25^{root} was increased by 10.6 percentage points, while PCK was less accurate by 7.1 percentage points compared to TDBU_Net [178]. However, this performance was better than the two-stage methodology, which was less accurate by 25.8 percentage points.

5.6 Hybrid approach for 3D absolute pose estimation

Since the one-stage approach with GAST-Net$_{ABS}$ does not provide root depth estimation that is more accurate than the two-stage approach but yields more accurate root-relative keypoints, we decided to take advantage of both approaches. We used the output of GAST-Net$_{ABS}$ to compute the root-relative keypoints and combined this with the RootNet network for root depth estimation to obtain the final absolute keypoints.

Method	PCK	AP^{root}_{25}	3D-PCK$_{abs}$
Root-GAST (Two-Stage approach) [278]	63.8	58.4	54.7
Root-GAST (One-stage approach) [278]	**82.5**	56.8	56.1
Root-GAST (Hybrid approach) [278]	**82.5**	**58.9**	**56.8**

Table 5.4: Evaluating the Performance of Root-Relative keypoints, absolute root and Absolute Keypoints on the MuPoTS-3D Database. The best is in bold.

5.6.1 Conclusion

In this way, we have achieved an improvement in accuracy, compared to the existing literature approaches, of more than 8.8 percentage points on 3D-PCK$_{abs}$ on MuPots dataset as shown in table 5.4.

5.7 Geometric method for absolute root keypoint

Indeed, it has become common practice to delegate tasks entirely to machines. We rely on black boxes to perform mathematical calculations as they see fit, often yielding more complex computations when simpler ones could achieve the same outcome. Therefore, we propose a geometric solution that can be practically applied to estimate absolute posture, instead of using a neural network such as RootNet.

The method involves determining the intersection point between the ground plane (P) and the line that connects the camera to the keypoint, which represents the depth distance. In this context, the keypoint typically refers to a human joint

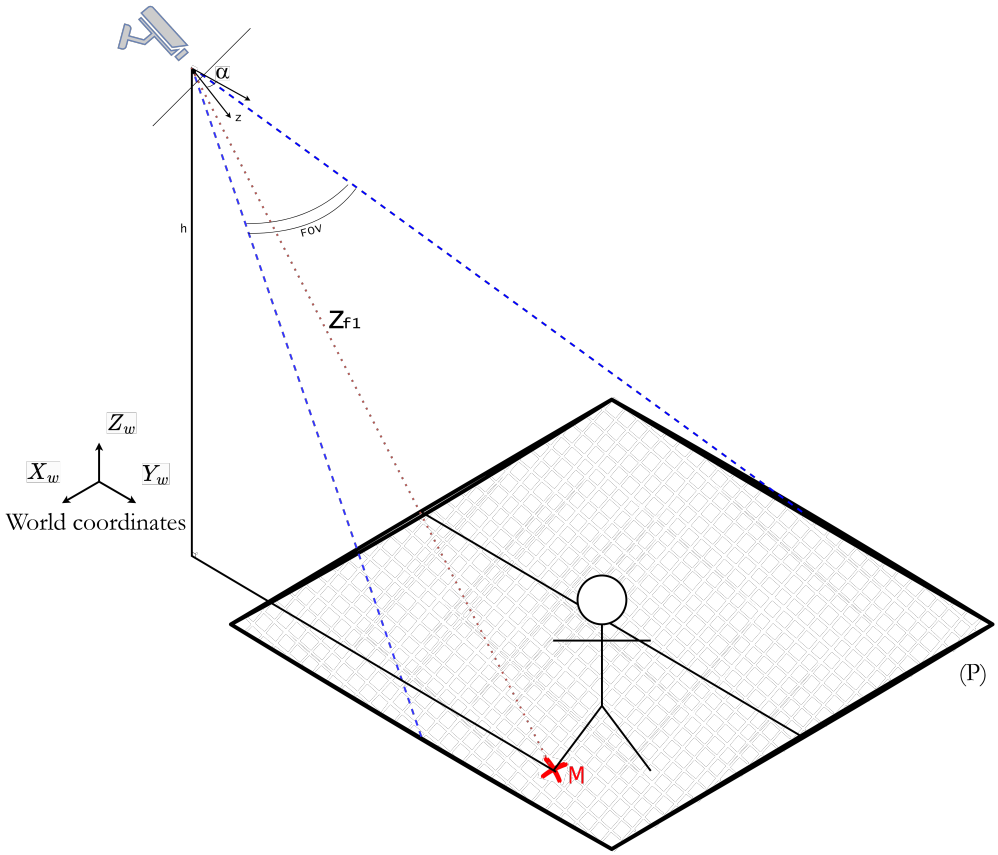

Figure 5.3: Geometric Method for Determining the Absolute Position of a Person's Keypoint

that is in contact with the ground, often the foot. To accomplish this, we make the assumption that the camera is positioned at a height h relative to the ground and is tilted at an angle α with respect to the horizontal plane. This angle corresponds to the angle between the optical axis of the camera and the horizontal plane, as illustrated in figure 5.3

Understanding the intrinsic projection matrix P is indeed crucial in mapping a 3D point onto a 2D image plane in computer vision and 3D reconstruction. This matrix contains essential information about internal camera parameters, such as

focal length and pixel coordinates. The focal length, denoted by f_x and f_y for x and y directions respectively, represents the distance between the optical center of the camera and the image plane. It determines how much the camera lens converges or diverges from the incoming light rays. By knowing these focal lengths, we can calculate the intrinsic projection matrix P, which allows the transformation of the coordinates of the 3D world into the coordinates of the 2D image.

The projection matrix P is defined as: $P = \begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix}$

Where cx and cy denote the principal point coordinates on the image plane.

Conversely, we can obtain the inverse of P, denoted as P^{-1}, which allows us to project points from the image plane back into three-dimensional space. The inverse matrix is given by: $P^{-1} = \begin{bmatrix} \frac{1}{fx} & 0 & -\frac{cx}{fx} \\ 0 & \frac{1}{fy} & -\frac{cy}{fy} \\ 0 & 0 & 1 \end{bmatrix}$

Estimating the depth of a point in three-dimensional space requires its projection onto a two-dimensional image plane, represented as (u,v,w) pixel, where w represents a homogeneous coordinate. This projection is facilitated by a matrix that maps a point from the 3D world to the 2D image plane. Conversely, given the projection of a point on the image plane, we can infer its position in three-dimensional space using the inverse of this matrix.:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = P \begin{bmatrix} x \\ y \\ z \end{bmatrix} \implies \begin{bmatrix} x \\ y \\ z \end{bmatrix} = P^{-1} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

(x,y,z) represents a point in three-dimensional space, belonging to $\mathbb{R}^3$, enabling accurate depth estimation and reconstruction.

Let's consider that $w = 1$. We can then express the projection of a 3D point (x,y,z) onto a 2D image plane using the inverse of the projection matrix, denoted

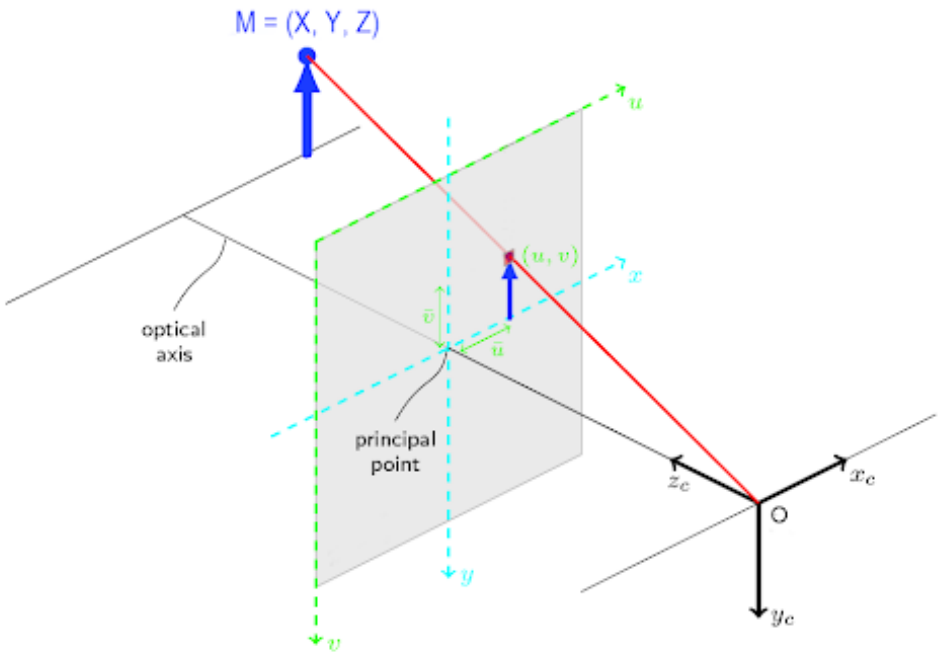

Figure 5.4: From Camera coordinates to Image plane

as P^{-1}, as follows:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = P^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{fx} & 0 & -\frac{cx}{fx} \\ 0 & \frac{1}{fy} & -\frac{cy}{fy} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{u-cx}{fx} \\ \frac{v-cy}{fy} \\ 1 \end{bmatrix}$$

$(u, v, 1)$ represents the pixel coordinates of the point in the image, where w is set to 1. By substituting these values into the equation, we obtain the transformed coordinates in homogeneous form. The resulting expression represents a point on the line in 3D space passing through the corresponding pixel (u, v) on the image plane and converging at a common point M as in image 5.4.

Note that w is set to 1 to maintain the homogeneous representation of the coordinates. Furthermore, let's consider the vector $\overrightarrow{(OM)}$, which represents the direction from the camera center O to the 3D point M. By substituting the trans-

formed homogeneous coordinates into the equation, we can express this vector as: $\overrightarrow{(OM)} = \begin{bmatrix} \frac{u-cx}{fx} \\ \frac{v-cy}{fy} \\ z \end{bmatrix}$

Therefore, the line can be defined by: (D): $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + t \cdot \overrightarrow{(OM)} = t \cdot \begin{bmatrix} \frac{u-cx}{fx} \\ \frac{v-cy}{fy} \\ z \end{bmatrix}$, where $t \in \mathbb{R}$.

This equation represents the parametric representation of a 3D line, where the line passes through the camera center O and extends in the direction of the vector $\overrightarrow{(OM)}$. We seek the 'correct' value of parameter t to find the point **M** on the ground. **M** is located somewhere along this line defined by (u, v).

To determine the exact location of **M** on the ground, we need to know the equation of the ground plane P. Once we have the equation of the ground plane, we can find the intersection point between the line defined by (D) and the ground plane. The ground is modeled by a 3D plane (P) with the equation $(P) : ax + by + cz + d = 0$ whose normal vector is $\overrightarrow{n}$. Thus, to determine this equation, it is necessary to express the normal $\overrightarrow{n}$ in the frame of the camera $oxyz$.

A coordinate system, $oxyz$, is associated with the camera, where the $\overrightarrow{oz}$ axis represents the optical axis perpendicular to the image plane (Figure 5.5). The objective is to determine the distance between a point M on the ground and the camera. This point M corresponds to the pixel coordinates (u, v) in the image, which represent a 2D keypoint in contact with the ground (typically the foot point) (Figure 5.4). No other coordinate system is utilized in this process.

The vector $\overrightarrow{n}$, which is orthogonal to the ground plane, can be obtained by rotating the vector $\overrightarrow{oz}$ using a rotation Rx (refer to matrix Equation 5.3) by an angle

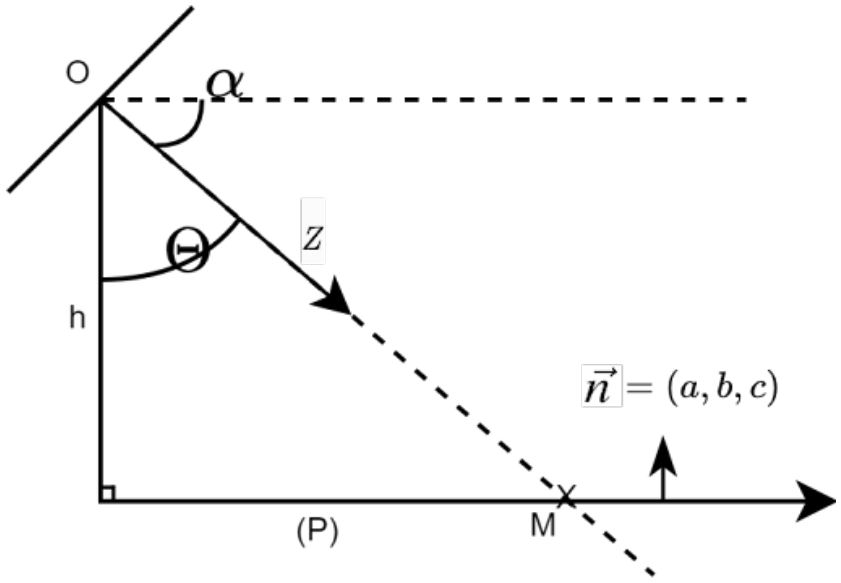

Figure 5.5: Using Trigonometry to Determine Camera Angle and Measure Absolute Distance

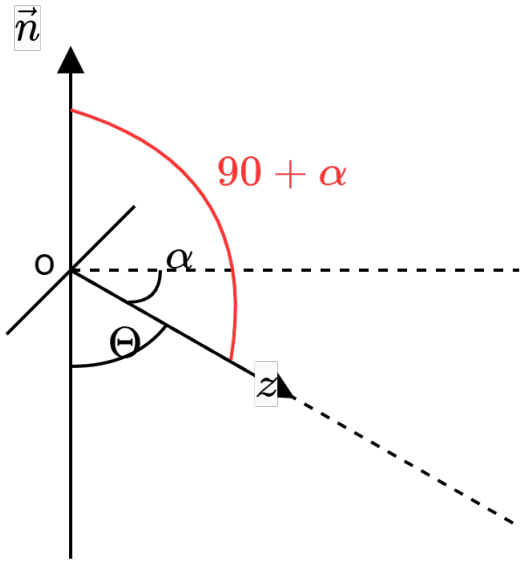

Figure 5.6: Representation of Rotation Angle to obtain the orthogonal vector

$\beta = 90 + \alpha$ (see Figure 5.6).

$$R_x(\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \tag{5.3}$$

We can obtain the rotated vector $\overrightarrow{(n)}$.

$$\overrightarrow{(n)} = R_x \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -\sin(\beta) \\ \cos(\beta) \end{bmatrix}$$

$(P) : -\sin(\beta) \cdot y + \cos(\beta) \cdot z + d = 0$ To find the value of d in the equation of the ground plane (P), we can choose a specific point $M = (0, 0, z_M)$ on the ground plane. Since M lies on the (oz) axis, the coordinates x_M and y_M are both zero. $\cos(\theta) = \frac{h}{z_M} \implies z_M = \frac{h}{\cos(\theta)}$ (figure 5.5).
As $M \in (P)$: $d = -\frac{\cos(\beta)}{\cos(\theta)} \cdot h \implies (P) : -\sin(\beta) \cdot y + \cos(\beta) \cdot z + -\frac{\cos(\beta)}{\cos(\theta)} \cdot h = 0$
To find the intersection between (D) and (P), we need to substitute the coordinates of the line into the equation of the ground plane and solve for the parameter t. We found then:

$$t = \frac{\cos(\beta)}{\cos(\theta)} \cdot \frac{h \cdot fy}{fy\cos(\beta) - \sin(\beta)(v - cy)}$$

And:

$$\begin{cases} x = \frac{t(u-cx)}{fx} \\ y = \frac{t(v-cy)}{fy} \\ z = t \end{cases}$$

To perform numerical calculations, we need the pixel coordinates (u, v), the height h, and the angle α. With these values, we can calculate the parameter t and the intersection point (X, Y, Z) between the line (D) and the ground plane (P) in camera coordinates using the equations provided.

In fact, utilizing our method to estimate absolute posture requires information such as the camera's height h from the ground and its angle alpha. However, the lack of these data in existing databases poses a challenge when it comes to evaluating our method. To overcome this, we conducted tests using our own camera and measured the depth distances using a laser rangefinder. Through these tests, we observed that the geometric estimation provided by our method appeared to be more accurate compared to the estimations obtained by RootNet.

By comparing the results of our geometric approach with the depth measurements obtained from the laser rangefinder, we were able to assess the accuracy of our method. This comparison allowed us to demonstrate that our geometric estimation outperformed RootNet in terms of accuracy, providing a valuable alternative for absolute posture estimation tasks, especially in the industrial use case where we have control over and can fix the required parameters.

Furthermore, RootNet relies on estimating distances based on the sizes of bounding boxes, which can vary significantly across images. Consequently, the estimated poses may appear unstable, even if they are actually stable. This instability arises due to the reliance on bounding box sizes as an indirect measure of distance. In contrast, our geometric method directly calculates the depth distances using camera parameters, resulting in more consistent and reliable results.

By bypassing the reliance on bounding box sizes and incorporating explicit geometric calculations, our method overcomes the instability issue encountered with RootNet. The direct estimation of distances using camera parameters leads to more accurate and consistent pose estimations, providing a robust alternative for absolute posture estimation tasks.

5.8 Conclusion

The one-stage approach in deep learning refers to a method where the model takes in raw data, such as images or video, and produces a prediction in a single step. To our knowledge, this term is commonly used for object detection, where it is in contrast to two-stage approaches, which first identify regions of interest in the data before making a prediction. However, in this thesis, we use the term "one-stage approach" to differentiate between the approach that involves two models: one model for root location and another model for root-relative keypoints, which then combine to estimate the absolute coordinates. On the other hand, the one-stage approach directly estimates the absolute keypoints without the need for the root location. Finally, we refer to the third approach proposed as a hybridization approach. This approach combines a one-stage model that estimates the absolute keypoints, then we recompute the root-relative keypoints to be used with a root-depth location model in two-stage approach manner.

The geometric method's success and accuracy may vary depending on the context, and further evaluation in different scenarios and datasets would be necessary to validate its generalizability. However, our results show the potential of using geometric solutions to boost absolute posture estimate accuracy.

In this chapter, we delved into the current state of research on single-view, multi-person absolute position and pose estimation tasks. We proposed a solution that is expected to meet the initial requirements and achieve better results in comparison to existing methods. The next chapter presents the implementation details of this framework and provides additional experimental results to support our proposed solution.

CHAPTER

6

Software system implementation pipeline and experimental results

6.1 Introduction

In this thesis, we are working on the software system design for 3D human pose estimation. The pipeline of this technology encompasses multiple stages, each of which presents its own challenges and possibilities. The three main components of a software system design pipeline are: data collection, model training and Inference.

The first step in designing a software system for 3D human pose estimation is data collection. This includes collecting images from various sources such as cameras or videos that capture both static poses and dynamic motions of people or objects being tracked by the algorithm. It also includes capturing information about body parts such as height measurements to help with more accurate tracking results later on in the process. Additionally, any other relevant environmental factors should be taken into account when collecting data so that they can be included during feature extraction later on down the line if needed.

As the research presented in this thesis is focused on an industrial application, the images used for inference are those captured by monocular cameras installed at the customer's site. To measure the effectiveness of our framework, we need a database that is representative of real-world images; image sequences that contain multiple people in the frame that move and can occlude each other, and that have different backgrounds and not taken in a lab setting like Human3.6m or HumanEva. For this reason, we decided to evaluate our pipeline on the MuPoTS-3D database.

In the following sections, we discuss the implementation of the framework's training and inference processes, and then present the experimental results.

6.2 Implementing the framework's training and Inference

The aim of this thesis is to develop a software that can accurately produce a sequence of 3D camera-centric coordinates for each individual in a given scene. To achieve this goal, the system will include components such as feature detection, feature matching, pose estimation, and 3D coordinate generation. The software developed will utilize deep learning networks.

6.2.1 GAST-Net$_{ABS}$ training

The GAST-Net$_{ABS}$ model was trained using the Adam optimizer with a learning rate of 1×10^{-3} and a batch size of 32. The training was performed for 80 epochs on the MPII-3DHP [149] and MuCo-Temp [202] datasets. The computations were carried out on the supercomputer facilities at Mésocentre Clermont Auvergne University, and the training process took one week to complete.

6.2.2 Pre-processing phase

Pre-processing is a step that is often performed before image processing. It refers to a set of operations that are applied to an image or a set of images in order to prepare them for further analysis. The main goal of pre-processing is to make the image more suitable for the intended analysis. The common step in deep learning pre-processing is subtracting the mean values for the red, green and blue channels of the image, in order to center the data and reduce the variability in the data and makes the model more robust. Then divided to the scale factor from the pixel values of each channel to normalize the image data.
The HRNet model and the RootNet model, as well as many other deep learning

models, that process images require this step. This step is used to normalize the image, so that the pixel values are in a specific range to standardize the data and make it more suitable for the model. Keep in mind that these values are specific to the dataset that the models were trained on, so it's important to use the same pre-processing values when processing new images to feed the model. Then scaling factor is multiplied by the mean values.
The HRNet and RootNet models normalize the image data by using the mean values for the red, green, and blue channels of the ImageNet data (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) before inputting it into the model.
Then the image data should be resized to the appropriate dimensions. The RootNet model should be resized to 256x256, and the HRNet model should be resized to 288x384. This resizing will ensure that the image is the correct size to be processed by the model.

6.2.3 2D human pose estimation

The pre-trained YOLO-v3 network is used to generate bounding boxes (bbox) for people in monocular RGB videos in real-time. These bounding boxes are then used as input to a pose estimation system, which generates 2D poses from the bounding boxes using HRNet-w32. As explained in section 4.2.2, YOLO-v3 is able to identify a wide range of objects and classify them into classes such as 'person', 'vehicle', 'animal', 'road sign', and other traffic elements. The set of classes that YOLO-v3 is able to detect depends on how the network was trained.The input resolution used is 608×608.
To track the objects presented in the current image, each object is assigned a unique ID i. For the purpose of posture estimation, we only focus on the "person" category. Since GAST-Net uses information from 27 sequential frames, it is important to track the objects in order to save their 2D poses without confusion with other persons found in the image. To accomplish this, a multiple object tracking algorithm is

used, which is based on two L2 distances. The first distance, calculated between the centers of the bounding boxes, is used as a filter to quickly eliminate unlikely matches between objects in consecutive frames. This improves the computational efficiency of the algorithm. Careful consideration is given to the threshold value for this distance, which in our case is 30% of the image width. The second filter, which is calculated for those matches that pass the first filter, is based on the L2 distance between the bottoms of the rectangles. This provides a more robust match between objects, even if the bounding box center has moved slightly.
To keep track of the objects' previous and current IDs, a weight matrix is used. The rows of the matrix represent the current IDs, and the columns represent the previous IDs. This allows the algorithm to match objects in the current frame with those in the previous frame. When the L2 distance between the centers of the bounding boxes of two objects is calculated, if it is greater than the chosen threshold, the corresponding cell in the weight matrix is set to infinity, indicating that the objects are unlikely to be the same. If the L2 distance is less than the threshold, it means that the match is not ruled out, so the second L2 distance value between the bottoms of the rectangles is calculated and stored in the corresponding cell of the matrix. This distance provides a more reliable match between objects, even if the bounding box center has moved slightly. The lower the value in the matrix, the higher the likelihood that the match is correct, and that value will be used for the matching process.
Objects whose ID is not found in the previous frame are given a new ID, which indicates that they are a new object that is being tracked. The figure 6.1 illustrates an example.

The cropped image of the bounding box is transformed to specific size (288×384) to be used as input for the 2D pose estimator. The affine transformation applied preserves collinearity, parallelism and the ratio of distances between the points as in [86], which makes the images more robust. Subsequently, HRNet-w32 is implemented on each frame to generate 17 heatmaps of a resolution 72×96,

Figure 6.1: Tracking Result: The ID assigned to the person across the frames is 2.

each of which predicts 2D human joint locations P_{2D} for each detected individual in the MS-COCO format, since it was pre-trained on the COCO dataset [196].

6.2.4 3D human pose estimation

The 2D-poses $P2D^i$ for each unique person ID of the 27 successive frames were then gathered in a Map and passed to GAST-Net, a 3D single-pose estimator. This is to perform 2D-to-3D mapping and the recovery of the 3D root-relative pose $P3Drel^i$, where all generated joints are represented by their distances from the pelvis keypoint. GAST-Net is subsequently applied, as many times as the number of individuals in the frame.

At the same time, the images of the humans that have been cropped are resized to 256x256 in order to be processed by RootNet for depth root prediction, Z_{abs}^{root}, in each frame. These predictions are also stored in a map for each unique ID. Once we have obtained the root-relative keypoints of a frame after a delay of 13 frames, we calculate the absolute coordinates of all the keypoints.

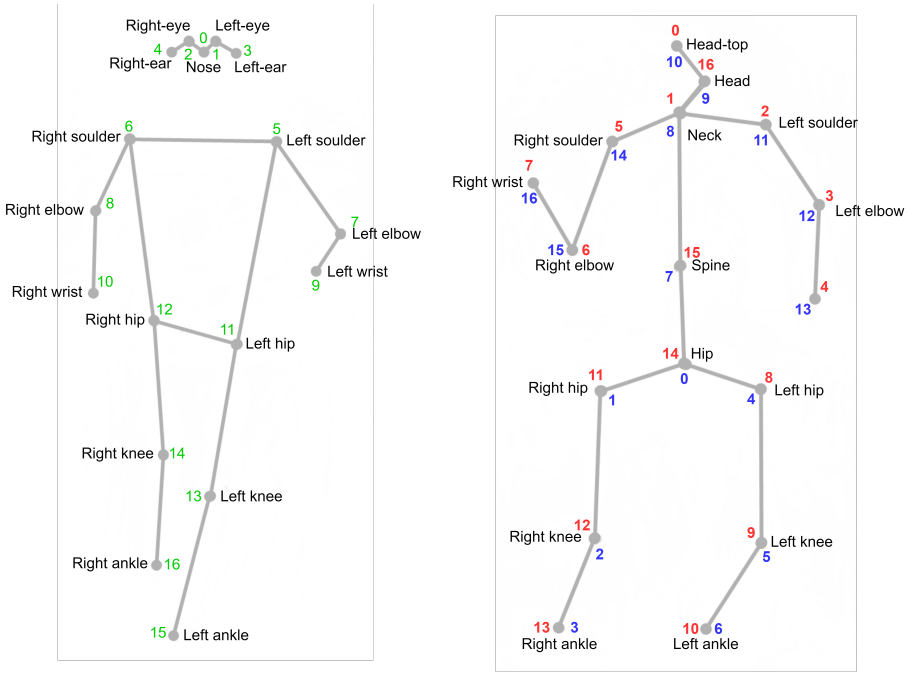

Figure 6.2: Comparison of keypoint formats. The first skeleton, marked with numbers in green, represents the MS-COCO format. The second skeleton illustrates the h36m format (in blue) and the MuPoTS-3D format (in red). The primary difference between h36m and MuPoTS-3D is the number of keypoints. Notably, the MS-COCO format includes additional keypoints for eyes and ears, distinguishing it from the other two formats

As we explained, we have developed two versions of the GAST-Net model: the original GAST-Net, which is pre-trained on the Human3.6m dataset [200] and is designed to process 2D keypoints in the h36m format; and our modified version, GAST-Net$_{ABS}$, which is trained on the multi-person video database MuCo-Temp [202] and the single-person video database MPII-3DHP [149]. The GAST-Net$_{ABS}$ model requires keypoints in the MuPoTS-3D format. For both models, a conversion of the keypoint format is necessary between stages.

In fact, in the software configuration, you can set the networks and keypoints

you wish to use, whether they be relative or absolute, the original GAST-Net or the modified one, and whether you want to use RootNet or the raw output of GAST-Net.

Before being used to estimate the 3D pose of a person, 2D keypoints must first be converted to the required format at the chosen network and then flipped.

The left keypoints must be flipped to become right keypoints and the right keypoints must be flipped to become left keypoints in the pre-processing stage, meaning the left and right sides are approximately mirror images of each other. In order to incorporate information about the 3D structure of the human body, GAST-Net uses symmetry to infer the 3D pose from 2D images by considering the correlations between different body parts and using a statistical model of body shape and articulation that includes the symmetry constraint. To maintain the same orientation for all keypoints, regardless of whether they are on the right or left side of the image, the symmetry should be multiplied by -1. Additionally, flipping the keypoints in this way preserves the relative position of each keypoint to the others, making it easier to accurately estimate the 3D pose of a person. This allows for more robust and accurate estimation of 3D poses.

Then, it is important to remember that before converting 2D poses to 3D poses, the input tenor of keypoints should be normalized. In the case of the original GAST-Net network, this involves transforming the x and y coordinates so that they range from -1 to 1. This is done by dividing the x-coordinate by the width of the image, multiplying the result by 2 and subtracting 1, and dividing the y-coordinate by the height of the image and multiplying the result by 2, then subtracting the height divided by the width. This ensures that the coordinate values are comparable across different images, regardless of their widths or heights. Normalizing the 2D keypoints is essential for accurate 3D pose estimation, as it ensures that the keypoint data is consistent across different images. This is also a common preprocessing step in image processing, which allows images to be scaled to the same size before they are used in further steps.

For the GAST-Net$_{ABS}$, the network is trained to directly estimate the absolute coordinates, and the normalization of the 2D keypoints is performed by multiplying them with the inverse of the calibration matrix. The calibration matrix is a matrix that contains the intrinsic parameters of the camera, such as the focal length and principal point. These parameters can vary between cameras and can also change over time for a single camera. The x and y coordinates of the keypoints are adjusted by subtracting the values of the principal points in x and y and then dividing by the respective focal lengths. However, for keypoints calculated by the symmetry pre-processing, the subtraction value for x is $(width\_sample - principle\_point\_x)$ instead of just the principal point, in order to preserve the relative position of the keypoints. This normalization step corrects for any distortion caused by the camera's intrinsic parameters, ensuring that the coordinates are in a consistent and accurate reference frame. This is important for accurate 3D pose estimation, as it ensures that the keypoint information is comparable across different images.

In addition to the normalization techniques mentioned above, another normalization step is applied to standardize the range of values for each keypoint in the dataset when using GAST-Net$_{ABS}$. This is achieved by subtracting the mean value of the keypoints from each keypoint in the dataset used and then dividing by the standard deviation of the dataset used. This process helps to ensure that all keypoints have similar ranges of values and are on the same scale.

The input to the model is a tensor that contains keypoints and their symmetry from 27 frames of 17 joints in 2D, having a shape of [2, 27, 17, 2]. The model is then run and its output is transformed into a tensor. The output tensor has a shape of [2, 1, 17, 3] and contains normalized values, which means that they are scaled to a specific range. In order to denormalize the tensor, we need to apply the inverse of the normalization that was applied during the pre-processing step. This typically involves for GAST-Net$_{ABS}$ multiplying the elements of the first index by the standard deviation (std) and adding the mean values of dataset's annotations. Then we are taking the exponential of the output keypoint that presents the root

to be transformed back from relative space to the absolute coordinates and then recalculate the coordinates of rest of keypoints. For GAST-Net$_{ABS}$, this typically involves multiplying the elements of the first index by the standard deviation (std) of the dataset's annotations and adding the mean values. Then, the output keypoints are transformed back from relative space to absolute coordinates by taking the exponential of the root and recalculating the coordinates of the remaining keypoints. Additionally, normalizing the 2D pose inputs helps to reduce the computational complexity of the 3D pose estimation process, as fewer parameters need to be estimated. This is because the inputs are normalized to a specific range, which simplifies the optimization process and reduces the number of variables that need to be estimated.

The symmetry is used to the output tensor in the post-processing stage to further refine the estimated 3D coordinates of the body joints. Finally, the mean of the 3D output tensor is calculated. The mean is calculated by summing the elements of the 0th dimension and dividing by the number of elements in that dimension. This means that the mean is calculated between the keypoints and symmetry.

In the final pipeline, we employed a hybrid approach using the RootNet model, which is applied after the HRNet model that estimates 2D keypoints. This means that we calculate the absolute depth of the root keypoint before estimating the 3D root-relative keypoints. This is because the GAST-Net model takes a sequence of images as input, while the RootNet model processes each bounding box in each image separately. As a result, there is a difference in timing between the results from these two models. To overcome this issue, we store the depth values for each bounding box (identified by an "id") in a list until we have the corresponding root-relative keypoints to calculate the absolute pose. We followed a top-down pipeline, where the RootNet model is applied to as many people as are present in the image, and we save the root depth coordinates for each "id".

Once the Z-coordinate of the root key point depth has been estimated, the X and Y coordinates of the root key point can be calculated with the camera-intrinsic

parameters, the image coordinates of the root, and the predicted absolute root depth (as in equation 5.3.2). The camera-intrinsic parameters are the parameters of the camera that describe its optical characteristics and perspective projection. These parameters include the focal length (fx, fy) of the camera, which determines the field of view and the degree of perspective distortion, and the center (cx, cy) of the camera, which determines the perspective projection of the 3D world onto the 2D image plane.

However, in our system we didn't use the image-coordinates generated by RootNet, we computed it from the 2D keypoints generated with HRNet. Since HRNet was pre-trained on MS-COCO dataset, it generates the 2D keypoints in COCO format that we transform to h36m format (the format of human3.6m skeleton). The keypoints of index 0 (the index of spine keypoint) is used to calculate its absolute coordinates.

Once the root key point coordinates $(X^{abs}_{root}, Y^{abs}_{root}, Z^{abs}_{root})$ have been estimated, these coordinates can be used as offsets to calculate the full absolute 3D pose of the person in the scene as in algorithm 1.

6.2.5 Determining the focal length of a camera

The focal length of a camera lens is a measure of the degree to which light is concentrated or dispersed by the lens. It is the distance from the lens's optical center to the point where all light rays meet to form a sharp image. This distance is typically expressed in millimeters (mm) and is an important factor in determining the field of view when the lens is paired with a camera. Generally speaking, shorter focal lengths produce wider fields of view [279].

The focal length of a camera lens can be determined in two ways: through the intrinsic matrix of the camera, or by physically measuring the distance between the lens and the image sensor in the camera. The intrinsic matrix is a mathematical representation of the camera's optical characteristics, and can be used to calculate

Algorithm 1 Compute Absolute 3D Landmarks

```
    procedure computeAbsolute3DLandmarks(landmarks3D, rootCoord, fx,
    fy, cx, cy)
    pt3d_r := landmrks3D.at(index_root) // Root-relative 3D landmark
    pt3d_r.z := pt3d_r.z + rootCoord.z
    pt3d_r.x := (rootCoord.x - cx) / fx * pt3d_r.z
    pt3d_r.y := (rootCoord.y - cy) / fy * pt3d_r.z

    cam_coord := [] // List to store absolute keypoints

    for p := 0 to nb_pts_3D - 1 do
        pt3d := landmarks3D.at(p) // Root-relative keypoints

        if p != index_root then
            pt3d.z := pt3d.z + pt3d_r.z
            pt3d.x := pt3d.x + pt3d_r.x
            pt3d.y := pt3d.y + pt3d_r.y
        else
            pt3d := pt3d_r

        cam_coord.push_back(pt3d) // Store absolute keypoints
    end for

    return cam_coord
end procedure
```

the focal length. Alternatively, the physical distance between the camera lens and the image sensor can be measured directly to calculate the focal length.
In our case, we determined the focal length of the camera through the intrinsic matrix approach using OpenCV. This method uses a checkerboard pattern as a calibration object which involves capturing several images of the checkerboard pattern from different angles. The 2D coordinates of the checkerboard can be easily located by using the function *cv2.findChessboardCorners()* in OpenCV. Additionally, the corresponding 3D points of the checkerboard pattern are also known. We then pass these 2D and 3D points to the *cv2.calibrateCamera()* function in OpenCV, which calculates the intrinsic and extrinsic parameters of the camera, including the focal length, represented as the matrix elements "fx" and "fy" in the intrinsic matrix.

6.2.6 Pose Visualization

Human pose visualization is the process of representing the estimated 3D pose of a person in a graphical format using a skeleton model, where a set of lines or bones are used to connect the different joints of the body. For visualization and posture analysis tasks, it is often necessary to use world coordinates for the skeleton, which is more effective than visualizing it in camera coordinates. This is because world coordinates provide a fixed and consistent reference frame that allows for better understanding of the 3D structure of the body, as well as the movement of the body over time. Additionally, it makes it easier to compare the posture or activity to other objects in the scene and analyze the behavior. While when visualizing the estimated 3D pose in camera coordinates, the pose can appear different depending on the camera's position and orientation, which can make it difficult to recognize the posture or activity. Furthermore, the camera coordinates are affected by the camera's intrinsic and extrinsic parameters, making the visualization harder to interpret. To conclude, using world coordinates makes it easier to accurately and

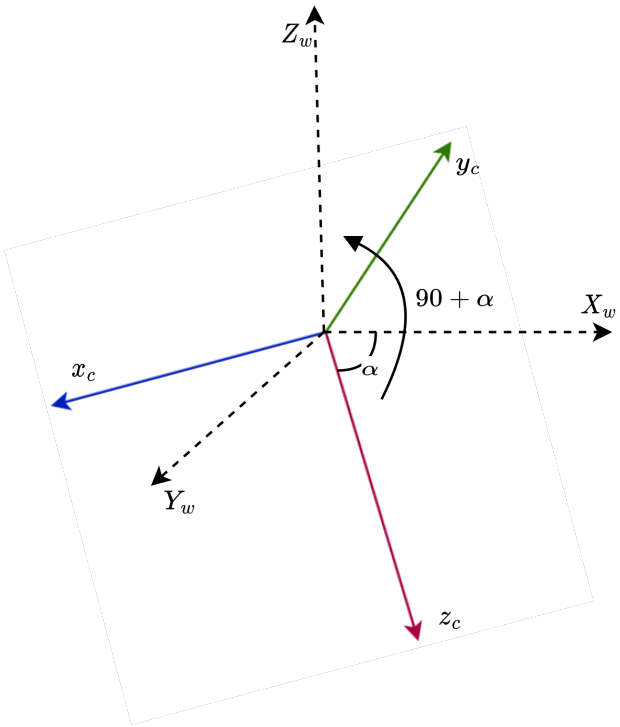

Figure 6.3: 3D Vector Transformation with Rotation about the X-Axis by Angle α.

consistently compare and evaluate the estimated 3D poses across different frames and cameras. It also allows for better understanding of the spatial relationship between the different body joints, facilitating the detection of errors in the estimated 3D poses.

To do that with our camera, we start by fixing the camera at a height h relative to the ground and inclining it by an angle α with respect to the horizontal plane. We then apply a rotation matrix along the x-axis to transform the points obtained from the camera marker to the environment or workshop marker. The rotation matrix is calculated using the measured camera rotation angle α in radian (see Figure 6.3). Then we search for the minimal value of the world Z_w coordinates of all the keypoints, and then subtract that value from each of the keypoints. This will effectively shift the human to the ground plane. The algorithm 2 describes the

Figure 6.4: Human detection + tracking + 2D pose estimation

function that converts camera coordinates to world coordinates.

6.2.7 Taxonomy of the Framework

Figure 6.5 presents the pipeline of the framework. It outlines the networks used in each step, the inputs given to each network, and the outputs they produce. This diagram gives an overview of the implementation section and summarizes the various components and their interactions. An object detector (YOLO) is applied to each image, and each bounding box is tracked through frames by assigning it an ID. The result is an image with green rectangles overlaid on each object, along with a small black rectangle containing information such as the ID of the tracking, the class of the object 'Person' in our scenario, the score of certainty, and the height and width of the rectangle. Then, the 2D pose is estimated for each bounding box in MS-COCO format, as shown in the figure 6.4. The 3D pose is then estimated and oriented to the world coordinate system to be represented using OpenGL. However, all people's root keypoint is located at (0,0,0) and the rest of the keypoints are represented relatively to that point. So that all poses are superimposed. This is the role of absolute 3D poses that allow an accurate representation of the scene.

Algorithm 2 Camera coordinates to world coordinates

```
procedure cam2world(landmarks3D_cam, landmarks3D_world, angle_rot_degree)
    alpha := Convert(angle_rot_degree)
    min_z := landmarks3D_world.at(0).z

    // Rotate points using the x-axis rotation matrix
    for p := 0 to landmarks3D_cam.size() - 1 do
        landmarks3D_world.at(p).x := landmarks3D_cam.at(p).x
        landmarks3D_world.at(p).y := (cos(alpha) * landmarks3D_cam.at(p).y)
                                   - (sin(alpha) * landmarks3D_cam.at(p).z)
        landmarks3D_world.at(p).z := (sin(alpha) * landmarks3D_cam.at(p).y)
                                   + (cos(alpha) * landmarks3D_cam.at(p).z)
    end for

    // Find the minimum z-coordinate
    for p := 1 to landmarks3D_world.size() - 1 do
        if min_z > landmarks3D_world.at(p).z then
            min_z := landmarks3D_world.at(p).z
        end if
    end for

    // Shift points to ground level
    for p := 0 to landmarks3D_world.size() - 1 do
        landmarks3D_world.at(p).z := landmarks3D_world.at(p).z - min_z
    end for
end procedure
```

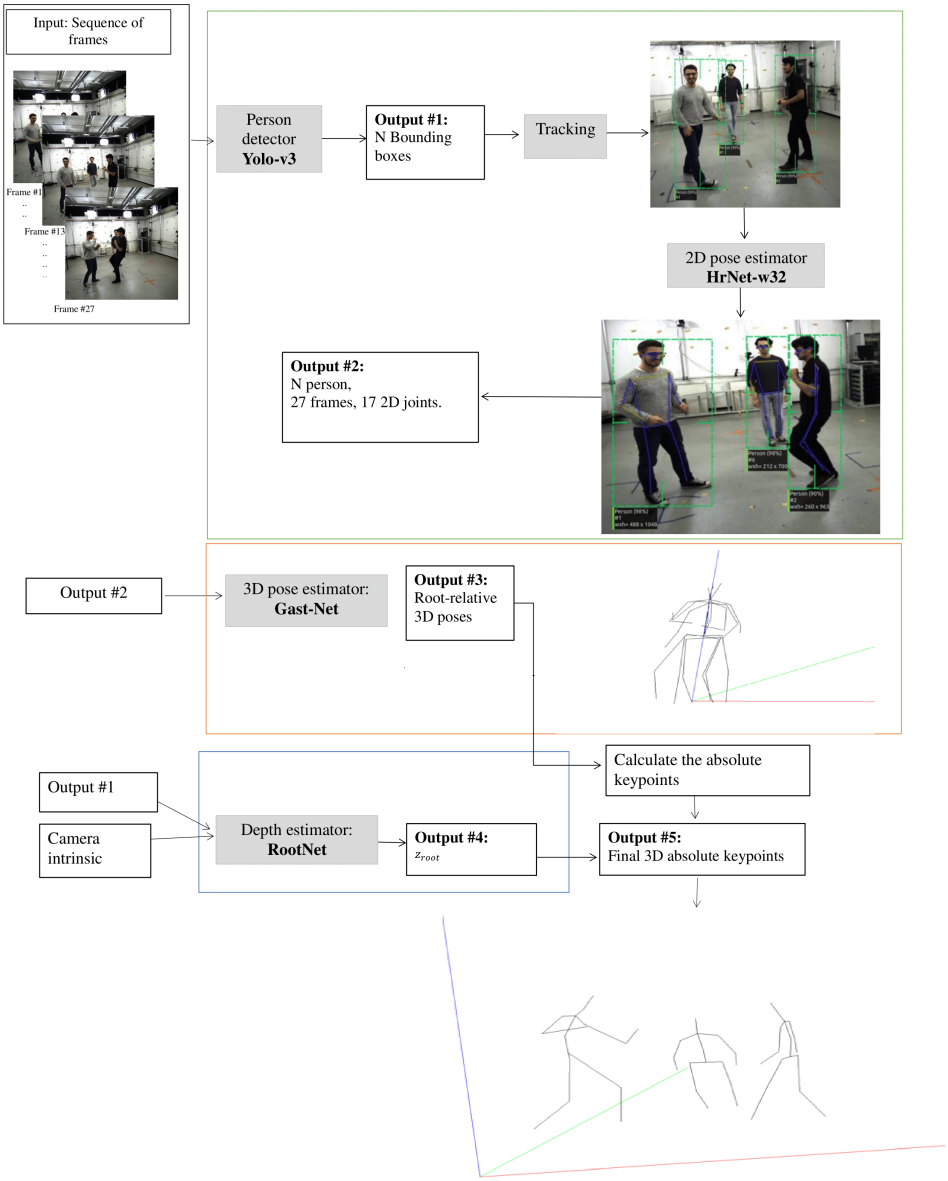

Figure 6.5: The pipeline of the proposed framework (Root-GAST-Net)

6.2.8 Posture analysis and fall detection

In order to detect falls, we analyze the verticality of 3D poses of detected objects by calculating the angle between the segment connecting the spine and hip and the z-axis. This angle should be close to 180 degrees for the standing case and close to 90 degrees for the lying state. Additionally, we also determine the height of a person in the z-axis by taking the absolute value of the ratio between the distance between the highest point (the head) and the lowest point (the feet) in the z-axis, and the total height of the person. The total height of the person is calculated as the square root of the sum of squares of the distance between the head and ankle in all three axes.

The code then compares the absolute value of the sine of this angle to a threshold sine of angle of verticality, and if the absolute result is greater than or equal to the threshold, it considers the pose to be lying. Additionally, it compares the ratio of the height of the person in the z-axis to a threshold, and checks if it is consistent with a standing or lying person. If the pose is determined to be lying for a certain number of consecutive frames, a fall is detected and the rectangle overlaid on the bounding box becomes red, as shown in Figure 6.6. It is important to track the person's ID over time and monitor the continuity of these angles and heights, and define a threshold to detect the occurrence of a fall. Additionally, monitoring the z position of the head point, if it decreases significantly through frames, may indicate a fall of the person.

Furthermore, we use other 3D keypoints such as the hip and knee joints to calculate angles between different segments of the body to determine the pose of the person. For example, the angle between the thigh segment and the hip-spine segment can be used to determine if the person is sitting or standing. We also check angles between the shin segment and the thigh segment to detect if the person is standing or sitting on the ground. These angles can help to refine the fall detection process.

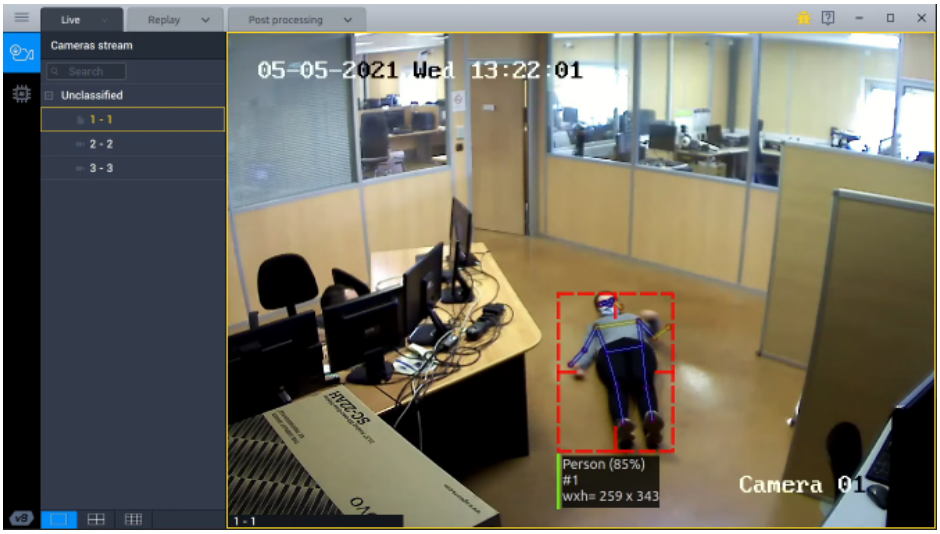

Figure 6.6: Fall detection

6.3 Experiments results

6.3.1 Performance of Sequence-wise on the MuPoTS-3D

In this subsection, we present an evaluation of the performance of sequence-wise 3D pose estimation using the 3D-PCK$_{abs}$ metric. We compare our results to those of state-of-the-art methods and observe an improvement in estimation accuracy in most of the sequences from the MuPoTS-3D dataset, as shown in Table 6.1.

The average precisions throughout the entire dataset were then examined using various threshold settings ranging from 25 to 10 cm. AP measured the accuracy of the root key point; we only evaluated the Root-GAST system's performance using the one-stage approach since two-stage approach and hybrid approach employed RootNet to predict the root joint. They produced the same result as the original paper. Table 6.2 displays the results. When compared to the state-of-the-art approaches, our method significantly achieves greater AP across all levels of thresholds. This indicates that our method is able to estimate more correct root

Method	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**
3D MPPE PoseNet (*) [259]	59.5	45.3	51.4	46.2	53.0	27.4	23.7
HDNet [261]	21.4	22.7	58.3	27.5	37.3	12.2	49.2
SMAP (*) [267]	42.1	41.4	46.5	16.3	53.0	26.4	47.5
GnTCN (*) [263]	64.7	59.3	59.4	63.1	52.6	42.7	31.9
TDBU_Net [178]	69.2	57.1	49.3	68.9	55.1	36.1	49.4
Root-GAST with Hybrid approach (*)	**89.8**	**77.0**	**73.4**	**77.0**	**81.0**	**54.3**	**68.4**
Method	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**
3D MPPE PoseNet (*) [259]	26.4	39.1	23.6	8.3	14.9	38.2	29.5
HDNet [261]	40.8	53.1	43.9	43.2	**43.6**	39.7	28.3
SMAP (*) [267]	18.7	36.7	**73.5**	46.0	22.7	24.3	38.9
GnTCN (*) [263]	35.2	53.0	28.3	37.6	26.7	46.3	44.5
TDBU_Net [178]	33.0	43.5	52.8	**48.8**	36.5	51.2	37.1
Root-GAST with Hybrid approach (*)	**60.5**	**71.3**	65.4	33.5	26.1	**67.3**	**46.9**
Method	**S15**	**S16**	**S17**	**S18**	**S19**	**S20**	**Avg**
3D MPPE PoseNet (*) [259]	36.8	23.6	14.4	20.0	18.8	25.4	31.8
HDNet [261]	49.5	23.8	18.0	26.9	25.0	38.8	35.2
SMAP (*) [267]	47.5	34.2	35.0	20.0	38.7	64.8	38.7
GnTCN (*) [263]	50.2	47.9	39.4	23.5	**61.0**	**56.1**	46.3
TDBU_Net [178]	47.3	**52.0**	20.3	**43.7**	57.5	50.4	48.0
Root-GAST with Hybrid approach (*)	**66.9**	35.7	**40.1**	38.5	26.0	35.3	**56.8**

Table 6.1: Sequence-wise 3D-PCK$_{abs}$ comparison with the state-of-the-art on the MuPoTS-3D dataset. (*) The accuracies of methods are measured on matched ground truths. The best is in bold, the second best is underlined.

keypoints even with a low distance threshold. These results demonstrate the effectiveness of our proposed method in accurately predicting the root key point in 3D pose estimation.

Method	$\mathbf{AP}^{root}_{25}$	$\mathbf{AP}^{root}_{20}$	$\mathbf{AP}^{root}_{15}$	$\mathbf{AP}^{root}_{10}$
3D MPPE PoseNet [259]	31.0	21.5	10.2	2.3
HDNet [261]	39.4	28.0	14.6	4.1
Root-GAST with one-stage approach	56.8	47.1	36.8	22.4

Table 6.2: Average precision of the root keypoint evaluation by different distances on the MuPoTS-3D dataset.

To compare with most of the existing methods that evaluate person-centric 3D pose estimations on MuPoTS-3D using MPJPE, we report our results using the same metric in Table 6.3. Our result was 101.9 mm, the result of [201] was 132 mm, the result of [280] was 120 mm, the result of [202] when adding the pose refinement model was 103 mm. Our method also outperforms the existing methods on this metric.

6.3.2 Performance on the Human3.6m

To validate the system, we chose the Human3.6M dataset, which contains only single-person videos. We compared the results using the mean root position error (MRPE) metric, which measures the accuracy of the root key point. For this comparison, we only evaluated the Root-GAST system's performance using the one-stage approach. The two-stage approach and the hybrid approach employed RootNet to predict the root joint, and produced the same result as the original paper. The root localization results of our GAST-Net$_{ABS}$ and the RootNet model are shown in Table 6.4. Even though the evaluation was performed on the Human3.6M

Method	**Year**	**MPJPE (mm)**
Temporal smoothing [202]	2020	107
Temporal smoothing + Pose refinement [202]	2020	<u>103</u>
Depth Prediction Network [280]	2019	120
LCR-Net [135]	2017	146
Mehta et al. [201]	2018	132
GAST-Net$_{ABS}$	2022	**101,9**

Table 6.3: MPJPE of relative poses on MuPoTS-3D dataset. Best in bold, second best underlined.

Method	**MRPE (mm)**	**MRPE$_x$ (mm)**	**MRPE$_y$ (mm)**	**MRPE$_z$ (mm)**
3D MPPE PoseNet [259]	289.28	35.95	58.65	268.49
Root-GAST with GA	178	33	41.9	158

Table 6.4: MRPE results comparison with RootNet [259] on the Human3.6M dataset. MRPE$_x$, MRPE$_y$, and MRPE$_z$ are the average MRPE errors in the x, y, and z axes, respectively.

dataset, we employed the GA model that was retrained on MPII and the MuCo-Temp dataset, while comparing it to the RootNet model that was trained on the MuCo dataset to make a fair comparison. Our measurement error amounted to 158 mm, while that of [259] was 289.28 mm. We could expect greater improvement if we train our model on the Human3.6m dataset, as it would take into account the specific characteristics of the dataset.

6.3.3 End-to-End Real-time system responsiveness

The response time of a framework depends on the material configurations. The Root-GAST pipeline is implemented in C++ and executed on a machine equipped with Intel Core i5-9500 processor, 32GB of dedicated memory, and an Nvidia GeForce GTX 1080 with 8GB of dedicated memory.

Table 6.5 presents a comparison of the response times of each network. The processing time was measured on batches of monocular images from the Human3.6m dataset, each containing one person. Note that the processing time of the tracking step is negligible. The frame rate of the entire pipeline with each

Model	**Min Response Time (ms)**	**Max Response Time (ms)**	**Average Response Time (ms)**
YOLO-v3	24	30	28
HRNet-w32	9	12	10
GAST-Net	27	33	29
GAST-Net$_{ABS}$	23	29	26
RootNet	4	8	5

Table 6.5: Response time per model.

approach is provided in Table 6.6. The proposed Root-GAST system achieved a

frame rate of 15 frames per second, which is suitable for real-time scenarios. This demonstrates that the proposed pipeline not only achieves good performance in terms of metrics but also runs at a speed that is suitable for real-time applications. Therefore, improving the metrics does not negatively impact the real-time aspect of the pipeline.

The proposed approach	**Average Frame Rate (fps)**
Two-Stage approach	13
One-Stage approach	16
Hybrid approach	15

Table 6.6: Frame rate per approach.

6.3.4 Qualitative results

As the system follows a top-down approach, the final result depends on the accuracy of all previous outputs. Incorrect detection can lead to inaccurate estimation of 2D keypoints and depths, which in turn negatively impacts the absolute pose estimation. Additionally, if there are multiple people within the same bounding box or body parts that are partially outside the box's bounds, the full-body joint calculation may be incorrect, as illustrated in Figure 6.7. This confusion arises from erroneous 2D point estimations, which have a cascading effect on the 3D-lifting process.

6.4 Conclusion

In conclusion, this chapter has presented the development and implementation of software and the improvements made to it. The effectiveness of the software has been evaluated through experimental results on both single person and multi-person databases, using various evaluation metrics to assess its performance. Furthermore,

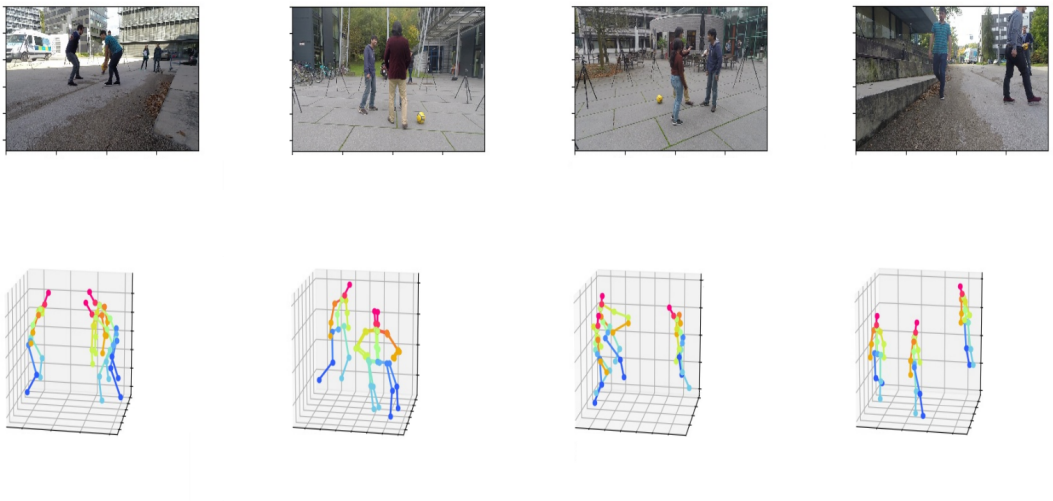

Figure 6.7: Erroneous 3D multi-person pose estimation. The first two images represent two similar poses of different people because one is completely occluded. In the right two images, one pose is incorrect because the body parts are partially outside the boxes.

the real-time aspect of the software has been considered throughout the evaluation process to ensure its practicality in real-world applications.

This chapter has provided a comprehensive understanding of the software's capabilities and potential for further advancement.

In this chapter, we have provided an in-depth analysis of the complete pipeline of the proposed framework, starting from the selection of appropriate datasets and determination of image focal lengths, to the final 3D pose result visualization and pose analysis. The different phases and components of the software have been thoroughly explained, providing a clear understanding of the entire process and how the various components work together to achieve the desired results.

In summary, the key takeaways from this chapter include: the use of the Hungarian optimization algorithm for people tracking, although it should be noted that it may not always be the optimal choice for handling complex or dynamic scenes, and the application of various treatments to the pre-processing and post-processing stages. Of particular importance is the symmetry treatment, which was applied to both phases and is detailed in this chapter. It is applied in both pre-processing and post-processing for 3D human pose estimation to improve the accuracy of the model. In pre-processing, symmetry can be used as a form of data augmentation to generate additional training examples by mirroring existing ones. In post-processing, symmetry can be used to refine the predictions of the model by enforcing symmetry constraints on the joints. This can help to reduce errors and improve the overall accuracy of the pose estimation.

In this chapter, we have also presented the experimental results on multiple datasets, which demonstrate that our framework has a significant improvement in comparison with recent approaches in 3D absolute multi-pose estimation on Mupots-3D. Furthermore, the system is efficient in terms of real-time performance, as each frame containing one person takes around 60 milliseconds to process using the Nvidia GeForce GTX 1080 graphics card. This performance can be further improved by utilizing high-performance hardware and using FP16 precision.

In addition, we have demonstrated how absolute 3D pose estimation can be applied in the field of 3D pose analysis. We have also presented our technique for detecting falls, which is based on analyzing the verticality of 3D poses by calculating angles and distances between keypoints and body parts. However, it's worth noting that it's also possible to use machine learning algorithms for fall detection by training them on a dataset of collected data or using these extracted features.

Part III

Software delivery

CHAPTER

Software system industrialization

7.1 Introduction

Industrialization has led to significant improvements in productivity, efficiency, and accuracy in many industries, including manufacturing, transportation, healthcare, and finance. The use of neural networks and artificial intelligence represents a huge qualitative leap in almost every industry. It is increasingly becoming the main driver of new technologies, which has allowed it to dominate much of industrial development as well as academic research. However, both academia and industry face challenging obstacles in adapting Deep Learning (DL) to pervasive applications [281, 282].
In the field of AI research, data scientists are more focused on finding the most accurate results possible by improving the performance of training and inference using data and neural network approaches with little or no concern for computational costs. Typically, when we focus more on improving accuracy, the effort does not necessarily translate into smaller network sizes and faster speeds. On the other hand, during the industrialization phase, data scientists need models that go faster even if they lose some accuracy. They are led to use specific hardware, and their main priority becomes to accelerate and optimize these deep learning models. This pushes them to use libraries adapted to the hardware used. In order to adjust the models to the design requirements of resource-constrained applications and the lack of dedicated hardware, they usually convert the DNNs, and sometimes they fine-tune or even reform these models.

In a CIFRE thesis research, we had to find a better compromise between performance and practicality in the development of a machine learning approach. This led to the development of an approach that outperforms the literature in terms of performance while adapting it to industrial constraints. It is important to develop approaches that are able to achieve good performance in terms of accuracy, speed, or some other metric, but that are also feasible to implement and deploy in real-world applications. Finding this balance between performance and practicality

can be challenging, and it often requires a combination of careful design and optimization of the model and the underlying hardware and software.
This optimization and adaptation of the models to the hardware available in the target environment, such as a customer's machine, is necessary to be able to run in real-time or near-real-time on the target hardware.

Indeed, to convince customers to purchase a product, it is crucial to take the cost and resource requirements into account. Offering software that can reuse existing hardware resources, such as cameras that are already installed at a customer's site, can help to lower the cost of the solution and make it more appealing to the customer. Because of this, we rely on RGB monocular cameras, the most prevalent type in the surveillance area, to estimate 3D postures. By using this type of camera, it may be possible to reuse existing hardware resources and avoid the need to purchase additional equipment.

In the following sections of this chapter, we discuss the various choices and motivations behind our approach, including the development challenges and economic constraints that we faced. We also provide details on the materials used for the hardware, software dependencies, and programming tools that were employed in our work.
Industrial software often employs the principles of DevOps to improve software quality and automate testing in order to detect errors early. We explore this philosophy in more detail in the section 7.4.2. In the following section 7.4.3, we also discuss the principles of MLOps, which aim to apply the best practices of DevOps to a machine learning project.

7.2 Main challenges of the system

Artificial intelligence (AI) has the potential to transform a wide range of industries, and many startups and companies have started to adopt AI technologies in recent years. However, it is not uncommon for these organizations to face challenges that

prevent them from successfully implementing AI projects and bringing them into production.

In the following, we discuss the specific economic viability challenges and application constraints that were experienced in order to understand the difficulties that the organization faced and how they were addressed.

7.2.1 Challenge 1: Simultaneously launch the AI models with other processes

The company Pryntec offers and uses the video recorder Digipryn. The system does not only process, but also does many other tasks in parallel as:

- Capturing and replaying video streams from multiple cameras;
- Storing the video streams in a database or file system;
- Analyzing the video streams in real-time to detect particular events or patterns;
- Displaying the video streams on a user interface for viewing;
- Sending notifications or alerts when particular events are detected.

Figure 7.1 displays a diagram of the video stream's "processing chain" from the video recorder. In order to reduce the size of the video stream sent by the cameras and speed up network transmission, it must be compressed into a video conversion codec. We use the H264 or H265 codec. H264 is the most popular codec and the most suitable for broadcast technology. While H265 is more optimized than the first codec. It provides a Full-HD image quality that is more detailed, with almost the same bitrate. The protocol used for the network is Real Time Streaming Protocol (RTSP). RTSP is a network control protocol designed for controlling streaming media servers. It allows a client to remotely control a streaming media

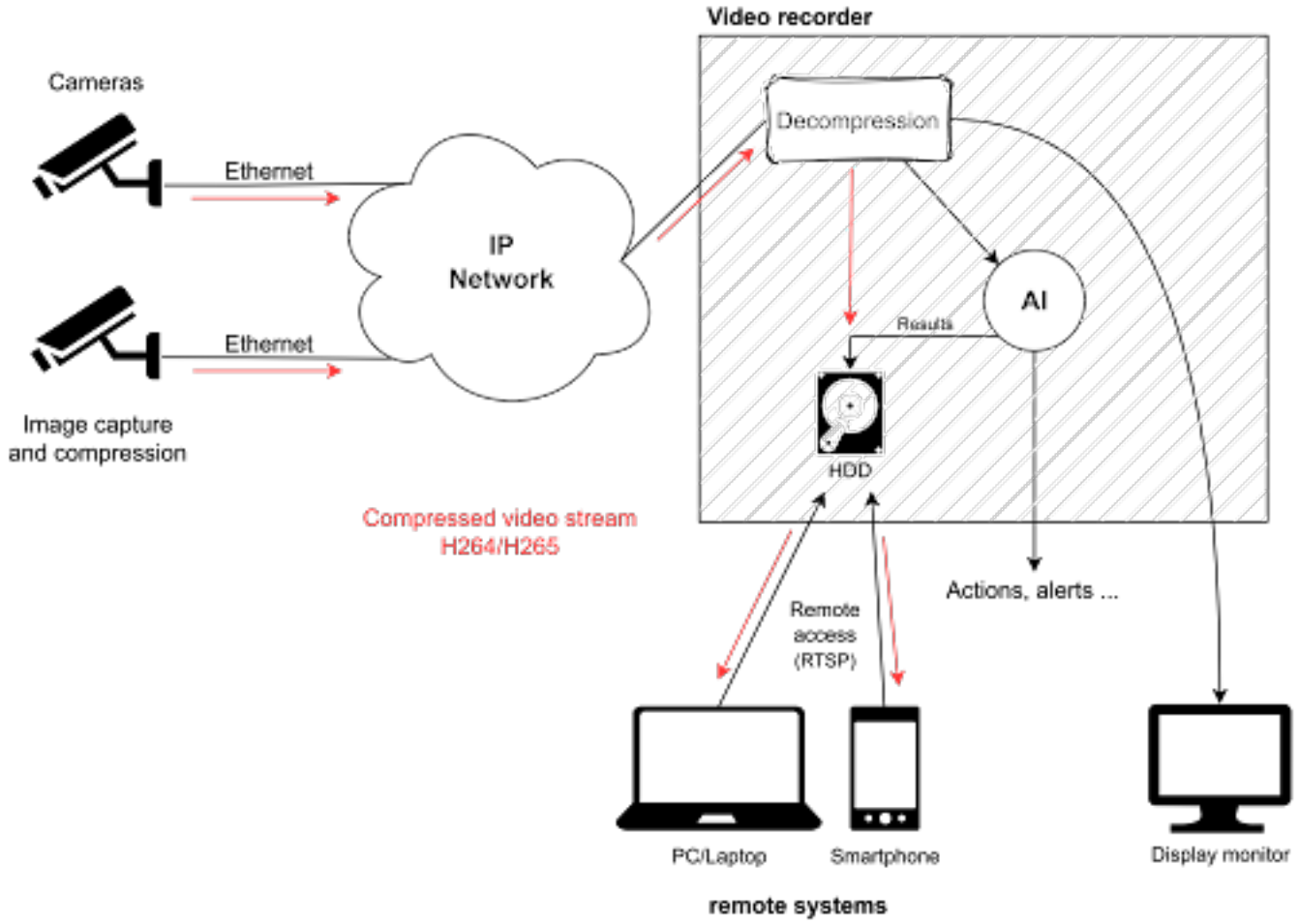

Figure 7.1: Processing chain of the compressed video stream

server, issuing VCR-style commands such as play and pause. RTSP can be used to control the video stream from a remote location, allowing the user to view the video feed from a camera on a device such as a phone or computer. RTSP is also often used in security camera systems, allowing users to remotely view and control the camera feeds.

The video recorder saves the received data in a compressed format on the hard disk to maximize storage capacity for the streams. It also decompresses the images

for display to the operator or for processing by artificial intelligence algorithms. The results generated by the image processing should be saved as well, and can be visualized or used to perform an action or make a decision.
Furthermore, this global processing chain is executed for N parameterized video streams, meaning that it is run as many times as the number of parallel cameras that are used as input.
Additionally, there may be external systems that allow remote access through another software or smartphone app, enabling users to view live video, replay recordings, export videos in MKV format, and so on. This adds additional tasks to the recorder, such as searching for an image on the disk, loading it, decompressing it, and sending it to the phone, among others.

To summarize, the video recorder performs a variety of tasks, and one of the main challenges is to run AI models concurrently with these other operations. This can be difficult because running AI models can be resource-intensive and may require a large amount of computational power and memory. To address this challenge, it may be necessary to reduce the number of cameras being used, which would in turn reduce the number of treatments carried out.

7.2.2 Challenge 2: Reuse existing material as much as possible

In section 5.3 of chapter 5.5, we have reviewed the various visual cues that can contribute to the perception of depth in images. We have classified the inputs of 3D pose estimation approaches into two main categories: multi-images and single-image.

Multi-images based techniques that requiring two or more images to use as input. In this case, several sources can be distinguished. Indeed, the depth estimation can be acquired either from:

- Multi-view images: obtained from multiple fixed cameras capturing the same scene from different angles, which are generally inspired by

the biological model and employing binocular cues. Approaches based on these images generally rely on triangulation techniques.

- Monocular image sequences: multiple images taken obtained from the same fixed camera at different times filming moving objects, and employ motion-related cues.

Single-images-based techniques In the other hand, some approaches use only indices of one image, usually from Depth cameras.

- camera RGB-D (Red Green Blue -Depth) is an optical technology, which relies on infrared light to extract the depths of a scene. The best known RGB-D cameras are the Time Of Flight (TOF) cameras or the Kinect sensors, explained previously in chapter 5.5.
- Depth images: captured from cameras RGB-D (Red Green Blue - Depth), an optical technology that relies on infrared light to extract the depths of a scene.
- Monocular images: In this case it is not possible to apply the principle of triangulation and there is then an infinity of 3D solutions for a given image.

Accurate 3D pose estimation can be achieved by processing multiple RGB images taken from different viewpoints, which contain sufficient 3D information to solve the occlusion problem, or by using depth images (RGB-D), which provide 3D information about the distance between the object in the image and the camera. However, obtaining multi-view observations can be expensive in real-life scenarios. This increases the installation costs, as each surface now needs to be seen by at least two cameras rather than just one. It also adds additional video streams to the recorder, which increases the computational load for recording, decompressing, processing, and other tasks. This may require more memory and may necessitate switching to a more powerful machine to run the software system, significantly

increasing the overall cost of the process. The same is true for adding another processing server.
Additionally, RGB-D cameras are not practical for use in real-world applications. They often use infrared illumination and are sensitive to natural light, making them suitable for use in indoor scenes only. Furthermore, if these cameras are not reused for other purposes, it may be difficult to convince the customer to purchase, install, and maintain additional specialized cameras for this specific use.
In conclusion, both options increase the cost of the AI solution for the customer and make the product less competitive. Therefore, using monocular cameras is the only viable option. One benefit of using monocular cameras is the ability to reuse existing materials in future applications and utilize data from security camera images to improve the study and analysis of a person's activity. In the PTI ("Protection du travailleur isolé") application, designed to protect lone workers, we can use the keyboard as an additional sensor to confirm periods of person's inactivity and ensure the safety of the worker. This can provide valuable information for monitoring and safety purposes.

7.2.3 Challenge 3: Error accumulation in a chain of algorithms

Another major challenge that we faced was creating a pipeline that incorporates multiple algorithms, as each algorithm can be imprecise or even fail, whereas in the evaluations, only one brick is tested rather than the entire set. In the system, we have observed that the human detector may sometimes miss or have difficulty recognizing a person due to luminosity constraints, background complexity, or confusing activities when processing real-world videos. Indeed, the databases typically provide the coordinates of the people's locations, even when they are obscured, in order to preserve the tracking and ensure that the person is processed in all frames. Researchers rely on these ground-truth coordinates of detection to ensure that the performance of their pose estimation models is not affected by

incorrect person detection or by the choice of a detector, particularly when these databases are used in the evaluation phase of pose estimation approaches.

In the same way, while evaluating a 3D posture estimator that uses 2D points as input, we utilize annotated 2D points, which are perfect. Therefore, the published precision measured for images in the databases will be better than the precision of real-world images, since they were achieved under circumstances far from the real-world.

Currently, all proposed models for object detection, particularly people detection, and for 2D pose estimation are still in development and none of them are ideal. In practice, the estimation of 3D poses is influenced by errors in the estimation of the 2D points it uses as input, which in turn can be impacted by mistakes in detection and tracking.

This makes the models that analyze videos and process different 2D person's postures through successive frames to produce the 3D pose even more challenging. If a person is detected again after being missing for one or more frames, the 3D model will process the last estimated 2D postures, even if they are based on older images and there is no continuity of poses, which can negatively impacts the result. Furthermore, the fact that the models are trained using ground truth coordinates makes matters worse, as they learn to predict the 3D posture based on features and parameters that may not always be accurate, particularly when the input consists of 2D poses from only a few images. While using more images could reduce the impact of a person being missing in several images, it would also increase the processing time and the number of images between the input image and its output. As a result, data scientists need models that are faster, even if they lose some accuracy, to maintain the notion of real-time, which is essential for applications that rely on security and monitoring.

7.3 System hardware and software

In this section, we will discuss the hardware and software used in our system. This includes the hardware platforms we considered for the training and inference stages, as well as the learning and inference libraries and programming tools we used. We will also discuss the trade-offs we faced when choosing these elements and how we balanced the need for performance with other factors such as cost, energy efficiency, and size.

7.3.1 Hardware

No matter the field, the performance of the algorithms is only one factor in any industrial project's success; the associated costs, including the hardware employed, must also be taken into account. The purpose and constraints specific to each system determine the capacity and resources needed, hence the choice of hardware. A PC that runs fixed systems does not have the same components as the PCs used for development and programming by the company, or in the servers in the data center where the equipment is more powerful that enables the training of complex models, or in the electronic cards used to operate directly the system embedded on the devices with limited capacity. Therefore, it will be crucial to differentiate between hardware used for inference and hardware used for training. While the inference phase can be carried out on either a Central Processing Unit (CPU) or a Graphics Processing Unit (GPU) depending on performance requirements, the training phase is almost always carried out on GPUs. GPU provides large resources to train and execute the deep learning and neural networks based models faster by performing all operations simultaneously in parallel on a large number of cores rather than sequentially for better computation. Additionally, as deep learning models are frequently trained on big datasets, memory-intensive computations are needed. The GPU is the optimum option for effectively processing this type

of data. But this does not mean that we can free ourselves from the use of CPU. CPU remains a reliable processor for general purpose computing, while GPU is more suitable for intensive computing. Especially since each CPU core is faster and more powerful than a GPU core, but the GPU is more powerful to perform specific tasks very quickly, especially those that can be processed in parallel. And this is the case of neural networks. Operations such as the calculation of weights and activation functions of each layer of the network or back propagation can be performed in parallel.
Using hardware acceleration, such as GPUs, can significantly improve the performance of deep learning inference. It is important to choose hardware that is appropriate for the task at hand and is capable of running the model efficiently.

There are many different hardware options available for the inference stage of a machine learning model, as mentioned. The choice between GPUs, desktop x86 CPU, embedded platforms GPU (as Nvidia Jetson series), Intel CPU, other embedded platforms depend on the specific requirements of the customer and the application. The computing power varies on these different hardwares. Automated registration of license plates (LAPI), as example of such inference which take place in-device, couldn't integrate the big graphics cards that heat or consume more energy. LAPI is a vehicle equipped with a system that can read license plates dedicated to homeland security. It uses a Box embedded the plate analysis algorithm which required a battery for its power supply. thus, embedded development is often driven by the need to deploy highly optimized systems. Overall, when choosing hardware for the inference stage of a machine learning model, it is important to consider:
- The computational requirements of the model: The hardware should have enough processing power to run the model at the desired speed and with the desired accuracy.
- The energy efficiency of the hardware: If the device will be portable or battery-

powered, it is important to consider how much energy the hardware will consume.
- The cost of the hardware: It is important to consider the cost of the hardware in relation to the intended use case.
- The size and weight of the hardware: If the device will be portable or embedded, the size and weight of the hardware may be important considerations.
- The availability of the hardware: The hardware should be readily available and easy to obtain

Table 7.1 compares some graphics processing unit in terms of power energy consumed (watt) and the computing power (TFLOPs) in single and half precision, this information is taken from techpowerup website. Teraflops (TFLOPs) is defined as the number of floating point operations (FLOPS) that a device can perform in a second. 500 TFLOPS represents 500 tera-FLOPS, or 500 thousand billion operations per second. Floating point operations are arithmetic operations that involve numbers with decimal points, such as addition, subtraction, multiplication, and division. They are typically used to perform scientific and mathematical calculations, and are important for many applications, including deep learning and data analytics.

Note that the TFLOPs performance of a GPU can depend on a number of factors, including the specific architecture of the GPU, the clock speed of the GPU, and the memory bandwidth available to the GPU. The values in the table above are approximate and may vary depending on the specific configuration of the GPU.
In general, devices with higher TFLOPS ratings are able to perform more complex calculations and can be more suitable for tasks that require high levels of computational power, such as deep learning and data analytics.
Another crucial feature of graphics cards that significantly impacts the size of the training batch is the GPU memory. It enables the use of temporary memory for storing local variables of kernel implementations as well as storing network parameters, including weights and biases. For training, it was also necessary to save intermediate variables, such as momentum, and intermediate calculations,

GPU Model	Single Precision (FP32)	Half Precision (FP16)	Energy power
NVIDIA GeForce RTX 3090	35.60 TFLOPS	35.60 TFLOPS	350 W
NVIDIA GeForce RTX 3090 Ti	40.00 TFLOPS	40.00 TFLOPS	450 W
NVIDIA GeForce RTX 3080 Ti	34.10 TFLOPS	34.10 TFLOPS	350 W
NVIDIA GeForce GTX 1650 SUPER	4.416 TFLOPS	8.832 TFLOPS	100 W
NVIDIA GeForce GTX 1660	5.027 TFLOPS	10.05 TFLOPS	120 W
NVIDIA GeForce GTX 1080 Ti	11.34 TFLOPS	177.20 GFLOPS	250 W
NVIDIA Tesla V100	7.80 TFLOPS	125.00 TFLOPS	300 W
NVIDIA Jetson Xavier	32.00 TFLOPS	64.00 TFLOPS	15 W
NVIDIA Jetson Nano	4.40 TFLOPS	8.80 TFLOPS	5 W

Table 7.1: Common GPU models and their approximate TFLOPs performance for different types of floating point operations and the energy power

such as activation outputs, which are computed from the forward pass of each layer and used to calculate gradients in the layer's backward pass. For bigger AI projects, it is frequently to use multiple GPUs to help with distributed training, which enhances productivity and efficiency throughout the training phase.
The memory capacity and speed of an ML/AI system depends on the tasks being performed. For example, running a deep learning project that relies heavily on input and processing of enormous volumes of data may ultimately require a greater memory load. Even though RAM (Random Access Memory) size does not affect deep learning performance, it can prevent full execution of GPU code. We need to have enough RAM to work comfortably with the GPU. In fact, having at least as much RAM as GPU memory and an additional 25% for expansion is essential for deep learning. Also, the quantity of analysis data that will need to be stored in memory for processing and statistical work is another factor to take into account. Due to the availability of PCs equipped with many GigaByte or even TeraByte disks, data storage in the training phase is currently not a serious concern. The main question often asked regarding this component is whether to utilize SSD or HDD for AI applications. Before responding, let's first explain both these types of storage devices and the different technology behind them:

- **Solid State Drives (SSD)** which is newer and faster that can store data on fast access memory chips, it uses flash memory which provides better performance and reliability.

- **Hard Drive Disks (HDD)** accesses data by using mechanical platters and a moving head; relatively slower and more susceptible to damage.

There will be plenty of temporary storage of datasets, and it is going to be incredibly convenient to have an SSD that will quickly migrate data as needed. They also consume less power and generate less heat than HDDs, which can be beneficial in some applications. However, it is worth noting that SSDs can be more expensive than HDDs. Additionally, SSDs have a limited number of write cycles, so they may

not be the best choice for storing large datasets that will be frequently modified. In those cases, it may be more cost-effective to use HDDs or some other type of storage; data that won't be moved frequently or will eventually land in a permanent storage situation. Thus, AI software may need both of these. It will need SSD for temporary storage of datasets that will allow rapid migration of data as needed and accelerates the learning. However, for data that will not be moved frequently or will eventually be stored permanently, HDD will work just fine and will be much cheaper.

In our thesis work, we have benefited from the cloud-based high-performance compute cluster performed at the supercomputer facilities at Mésocentre Clermont Auvergne University for training DL models, we used NVIDIA v100 as GPU resources and a RAM memory of 100G in partition "audace2018"). For development, debugging and inference step, we used a Linux machine equipped with Intel Core i5-9500, with a dedicated memory of 32GB, and the Nvidia GeForce GTX 1080, with a dedicated memory of 8GB.

Subsequently, the choice of hardware imposes the use of specific libraries, as the conversion of the models is necessary for inference. The next section covered the software dependencies for training and inference.

7.3.2 Software libraries

The hardware used will guide the choice of learning and inference libraries and programming languages that are available. Some libraries and programming languages are optimized for use with specific types of hardware, such as GPUs or CPUs from a particular manufacturer. For example, the Nvidia libraries cannot be used on CPUs from other manufacturers, such as Intel. Similarly, libraries that are optimized for use with Intel CPUs may not work with GPUs from other manufacturers. It is important to carefully consider the hardware that will be used for machine learning and choose libraries and programming languages that are

optimized for that hardware. This can help to ensure that the model is able to run efficiently and achieve the desired performance.
On the other hand, there are several learning platforms among Keras, TensorFlow, PyTorch and many others for training phase. Each framework has its own set of capabilities and features, and the choice of framework can have a significant impact on the development and training process. One factor to consider when choosing a framework is the types of layers and operations that are supported. Some frameworks may have a more limited set of available layers and operations, which can limit the types of models that can be developed with that framework.

PyTorch is a popular choice for training deep learning models, and it offers a wide range of features and capabilities. The modern networks are typically developed on PyTorch. We will go into more detail about it in the following section. For the inference phase, we frequently have to make conversions to the models between the training and inference libraries to optimize the network and make it much faster. Likewise, it is necessary that the libraries used are in compatible with those used for training. For example, we use TensorRt on Nvidia platforms while OpenVino on CPU platforms.

7.3.2.1 Deep learning libraries for training

As mentioned, GPUs must be used for DL model training because they require intensive computational resources. Hence, we must use one of the various GPU programming libraries. At this point, the decision is determined by the type of GPU used; in fact, software and hardware are closely related. We therefore implement with **CUDA**(Compute Unified Device Architecture) [283], the programming associated with the graphics cards used, Nvidia. The CUDA toolkit is a complete set of tools that includes a C/C++ compiler, debugger and runtime libraries to deploy the GPU based applications. CUDA designed specifically to give the GPU instructions to perform a task. This technology, Initially launched in 2007, allows for fully exploiting the image processing algorithms integrated into video

surveillance software constitute an important decision-making aid. power of the GPU for parallel computing in a way optimized for better performance. In our CIFRE thesis research, we had to find a better compromise between these two different types. We developed a 3D human pose estimation approach that surpasses state-of-the-art approaches by increasing the percentage of correct points in the MuPoTs-3D database, which contains image sequences captured outside the labs and therefore closer to the real-world image. While we optimized the hardware required by using pre-trained architectures for processing steps that precede the 3D pose estimation. With the CUDA Toolkit, you can develop, optimize, and deploy your applications on GPU-accelerated embedded systems, desktop workstations, enterprise data centers, cloud-based platforms and HPC supercomputers [283].
Keep in mind that NVIDIA graphics cards are the only ones that support the CUDA language. Conversely, OpenCL, the other primary GPU programming framework, is an open standard that is not exclusive to NVIDIA processors. It can be used for CPUs, FPGAs, digital signal processors (DSP) and other types of processors. FPGAs are tiny semiconductor devices with the capacity to be programmed to perform incredibly specialized actions fast and effectively. OpenCL, an acronym for Open Computing Language, was launched by Apple and the Khronos group to provide a reference for heterogeneous computing, which allows parallel computing on different devices. But CUDA has the advantage of being created by the same company that creates the hardware on which it runs, which means it will obviously better match the computational of the GPU and provide better access to features and performance.
To build deep neural networks in the CUDA programming language, we need the **CuDNN** package [284]. The NVIDIA CUDA Deep Neural Network (CuDNN) is a GPU-accelerated primitive library for deep neural networks that provides very accurate implementations of standard routines such as forward and backward propagation, convolution, pooling, normalization, activation functions etc. in order to speed up processing. This allows developers to focus on training neural networks

and developing software applications rather than spending time tuning low-level GPU performance. It is also optimized for the GPU and can take full advantage of Nvidia graphics cards.
The second choice to make before creating or training deep learning models is the learning framework. Sometimes, it is the type of desired networks used that force you to make a particular choice. However, generally speaking, PyTorch [285] has grown in popularity and effectiveness for building Deep Learning (DL) models since the Facebook AI Research (FAIR) team first released it in 2017. This open-source optimized machine learning tensor library was created to boost deep neural network implementation speed and flexibility. It is based on Python and Torch and is mainly used for applications using GPUs and CPUs.

In both academia and industry, PyTorch is currently the most widely used library for AI (Artificial Intelligence) researchers and practitioners over other deep learning frameworks like TensorFlow and Keras since it is completely Pythonic and uses dynamic computation graphs. It enables scientists, programmers, and neural network debuggers to real-time testing and execution of certain portions of code. Users can therefore check whether a part of the code works or not without having to wait for the complete software program to be implemented.

The framework of PyTorch uses dynamic computational graphs to determine the gradient values for the built-in neural networks. This type of graph is more flexible since it let users develop and evaluate the graph sequentially. Additionally, because the code is executed line-by-line, it is simpler to debug and locate errors in the code. These computational graphs can use arbitrary Python control flow instructions since they are generated from scratch after each iteration, which can change the overall size and structure of the graph. The advantage is that you don't need to code every path before launching the training. You could execute what you differentiate.
Like most frameworks, PyTorch is CUDA-enabled, which means that entry-level GPUs can be used.

PyTorch 1.6 GPU version is used in our work as software dependencies. Thus, learning of the models was programmed in Python language. An interpreted, object-oriented, high-level programming language with dynamic semantics. The Python interpreter and extensive standard library are available free of charge in source or binary form for all major platforms, and can be freely distributed. Python provides code that is clear and readable. While machine learning and artificial intelligence are built on complicated algorithms and flexible workflows, Python's simplicity makes it possible for programmers to create dependable systems. Instead of focussing on the technical details of the language, developers can devote all of their attention on solving an ML problem. The chosen Python version is 3.7.
Since the last century, scientists have employed computers to automate the processing of digital to automate the processing of digital images in many fields. How to process a very big amount of data in the shortest amount of time with the minimum number of processing resources has always been a major challenge for application developers. To optimize time and memory, developers are currently employing other programming techniques. One of these methods that enables the execution of the same operation on several variables simultaneously is vectorization. It is a programming technique that allows developers to perform operations on entire arrays of data instead of individual elements, which can significantly improve the performance and efficiency of their code. It is commonly used in image processing, where developers may need to perform the same operation on a large number of images or pixels. By vectorizing their code, developers can avoid the overhead of looping through individual elements and can instead perform operations on the entire array at once, which can significantly reduce the time and resources required to process the data. Vectorization is often used in conjunction with other optimization techniques, such as parallelization, to further improve the performance of image processing and other data-intensive applications [286]. In Python, we use NumPy for vectorization. It is a library used for scientific computing and provides tools for working with arrays and matrices of numerical data, which allows operations to be

performed on entire arrays or matrices, rather than individual elements. This can significantly speed up code, since vectorized operations are typically much faster than equivalent loops in Python.
Another approach to optimizing the performance of your deep learning models is to use distributed training. This involves training your model on multiple GPUs or machines, which can significantly reduce the training time. This allows multiple tasks to be performed at the same time. There are several tools and libraries available for distributed training, which can help you easily scale your training across multiple GPUs or machines.

7.3.2.2 Deep learning libraries for inference

CUDA and cuDNN are also used for inference to enable faster calculations. The GPU related libraries are CUDA 11.0.0 and CuDNN 8.0.0.
The design and deployment of an industrial embedded inference system using deep learning models is a large procedure. After training a neural network, it is possible to optimize these pre-trained computational networks for speed (both throughput and latency) at runtime.
For this purpose, TensorRT is widely adopted by the industry. TensorRT is an NVIDIA library essential for computer vision engineers to create AI applications and better deploy complex models in production. TensorRT is built on CUDA, NVIDIA's parallel programming model. Thus, on NVIDIA graphics processing units (GPUs), this library takes deep learning pre-trained model and 'compile' it into an 'engine' that runs fast for quicker inference.It can provide inference on numerous real-time services and embedded applications around 4–5 times faster. Let's review some of optimization's strategies that are carried out by TensorRT to boost the throughput of deep learning models and create an optimized engine.

- **Precision optimization:** TensorRT can optimize the precision of the model's parameters and activations to reduce memory usage and improve perfor-

mance.

- **Layers fusion:** TensorRT can merge multiple layers of a neural network into a single layer, reducing the overall number of operations that need to be executed and minimizing the overhead of moving data between layers.
- **kernet auto-tuning:** TensorRT automatically selects the best layers, algorithms, and batch sizes based on the intended GPU platform to set kernel parameters.
- **Dynamic Tensor Memory:** TensorRT improves memory reuse, especially on embedded systems with limited memory by only allocating memory to the tensor while it is in use. It minimizes the memory footprint and gets rid of allocation overhead for quick and efficient execution.
- **Multiple stream execution:** Similar to CUDA, TensorRT enables you to conduct inferences concurrently over various "streams."

By applying these techniques, TensorRT can significantly improve the performance and efficiency of deep learning models, making them more suitable for deployment in real-time applications such as autonomous vehicles, video analytics, and natural language processing. In the following, we go into more detail about these techniques.

Precision optimization: Precision optimization in TensorRT refers to the process of optimizing the precision of the model's parameters and activations for better performance.
In computer science, real numbers are numbers that can represent any value on the number line, including fractions and decimals. They are different from integers, which are whole numbers.
Real numbers are usually represented in computers using a binary floating-point representation. This means that the number is stored as a series of binary digits

(bits), with a fixed number of bits reserved for the integer part of the number, a fixed number of bits reserved for the fractional part of the number, and a single bit reserved for the sign of the number (positive or negative). The actual value of the number is determined by the combination of these bits, along with a pre-determined base (usually 2) and exponent (which determines the position of the binary point). FP32 means using 32 Bits to represent a floating point number. It is the more commonly used single precision floating point. We call half precision floating point, the numerical representation that uses 16 bits to represent a number (FP16) and double precision floating point (FP64), which uses 64 bits. There are three components in the representation: the sign bit, the exponent bit, and the mantissa In FP32 for example, the number 0.5 in binary floating-point representation might be stored as 01000000010000000000000000000000, with the first bit representing the sign (0 for positive), the next 8 bits representing the integer part (00000001, or 1 in decimal), the next 23 bits representing the fractional part (00000000000000000000, or 0 in decimal), and the exponent indicating that the binary point should be shifted 23 places to the right.

There are some limitations to the accuracy and range of real numbers that can be represented in this way, but for most practical purposes, binary floating-point representation is sufficient for working with real numbers in computers.

During the process of training of deep learning models, parameters and activations values are represented in FP32 precision. Higher precision that allows for more accurate representation of the values, but also requires more memory and can be slower to compute. However, the lower precision (e.g. 16 bits or 8 bits) can reduce memory usage and improve performance, but may sacrifice some accuracy.

TensorRT allows users to choose the precision of the model's parameters and activations based on the requirements of the application. For example, a model used for image classification may be able to tolerate lower precision without sacrificing too much accuracy, while a model used for speech recognition may require higher precision to maintain accuracy.

TensorRT includes algorithms that can automatically optimize the precision of the model based on the desired accuracy and performance. This can help users find the optimal balance between accuracy and performance for their specific application. Note that FP16 is supported by many modern hardware devices, including CPUs, GPUs, and specialized hardware accelerators. However, not all devices support FP16, and the level of support can vary.
For example, many modern GPUs support FP16, but they may not support all the features and operations that are available for FP32 or FP64. Some GPUs may also have different performance characteristics for FP16 operations compared to FP32 or FP64 operations.
In general, FP16 is useful for reducing the memory usage and computational requirements of deep learning models, which can be beneficial for deploying models on devices with limited resources. However, it is important to carefully consider the trade-offs between precision and performance when using FP16, as it may not be suitable for all applications.

Layers fusion: Another technique used by TensorRT for the optimization of deep learning models is layers Fusion. It is used to merge several layers into one more efficient kernel.

Merging layers can improve model performance by reducing the total number of layers to be executed and by minimizing the overhead of moving data between layers. There are two common types of merging layers: horizontal fusion and vertical fusion (figure 7.5).

Horizontal fusion consists of merging several successive layers into a single layer. For example, if a model includes a convolution layer followed by a pooling layer, they can be merged into a single layer that combines the convolution and pooling operations. This minimizes the overhead of moving data between layers. The second way is the vertical fusion which merges multiple layers that process the same input data into a single layer. For example, if a model has multiple

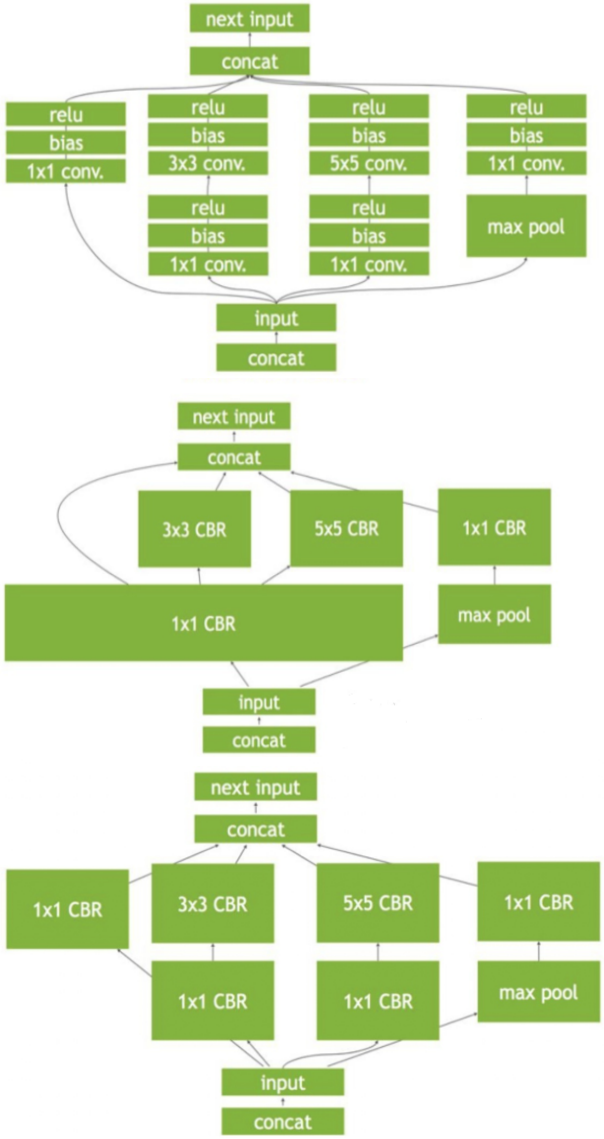

Figure 7.5: Fusion network layers by TensorRT

convolution layers that all process the same input data, they can be merged into a single layer that combines all convolution operations. This reduces memory usage.

Kernel auto-tuning Kernel autotuning is a feature of TensorRT that automatically selects the optimal kernel parameters for convolution and data shift (data expansion) operations. In a neural network, convolution and data shift layers are common operations that are often used to extract features from input data. The kernel parameters of these layers, such as the convolution filter and the shift step, have a large influence on the performance of the model. Automatic kernel tuning allows TensorRT to automatically select the optimal kernel parameters for each convolution and data shift layer based on the target computing architecture and model workload. This can improve model performance by selecting kernel parameters that are best suited to the characteristics of the target computational architecture. To use automatic kernel tuning in TensorRT, simply specify the automatic kernel tuning option during model configuration. TensorRT will automatically detect the convolution and data shift layers and perform the automatic kernel tuning during model optimization.

Dynamic Tensor Memory: Dynamic Tensor Memory is a feature of TensorRT that allows for efficient management of memory usage when executing a deep learning model.
In a neural network, each layer of the model requires a certain amount of memory to store input and output data. During model execution, memory must be allocated and released efficiently to avoid access conflicts and maximize memory utilization. Dynamic Tensor Memory enables TensorRT to transparently manage the allocation and release of memory during model execution. This can improve performance by avoiding memory access conflicts and efficiently using available memory resources. To use Dynamic Tensor Memory in TensorRT, simply specify the dynamic tensor memory option when configuring the model. TensorRT will automatically manage

the allocation and release of memory during model execution.

Multiple stream execution is a technique used to optimize the performance of deep learning models, particularly when the models are deployed on parallel computing architectures such as GPUs.
In a deep learning model, a stream is a sequence of operations that are executed in order on a set of data. When multiple streams are used, different sets of data can be processed in parallel, allowing the model to fully utilize the available computing resources and maximize performance.
TensorRT supports multiple stream execution by providing APIs that allow users to configure the number of data streams to be processed in parallel and specify how the data should be partitioned among the different streams. This enables users to take full advantage of available computing resources and maximize model performance.
It is important to note that multiple stream execution can increase code complexity and memory usage, and is not always beneficial in terms of performance. To make the most of this technique, it is important to understand the characteristics of the application and the target computing architecture, and choose an appropriate number of data streams to process in parallel.

7.3.2.2.1 PyTorch to TensorRT conversion

The specific steps and API calls required to convert a PyTorch model to TensorRT may vary depending on the model architecture and the requirements of the task. The common way to convert a PyTorch model to TensorRT by first converting the PyTorch model to ONNX format and then using TensorRT to import the ONNX model and optimize it.
Note that ONNX [287] (Open Neural Network Exchange) is an open source format for exchanging deep learning models between different frameworks and hardware

platforms. It defines a common set of operators and a standard data format for representing deep learning models, allowing models trained in one framework to be imported and used in another.
Using ONNX as an intermediate format can simplify the process of converting a PyTorch model to TensorRT, as it allows the model to be represented in a framework-agnostic format that can be easily imported by TensorRT. The function of conversion to ONNX takes the PyTorch model and a set of input tensors as input and produces an ONNX model file as output. Once the PyTorch model has been converted to ONNX format, it can be imported into TensorRT and converted to TensorRT engine object that represents the TensorRT version of the model.
We often have to perform model conversions between the training and inference libraries and there may be instances where certain layers or calculations are not supported. As stated earlier, all models used in this work were trained with PyTorch which is primarily designed to be used with the Python programming language. However, for inference, we implement with the C++ language. Therefore, to be able to use the models whose conversion to TRT engine is not possible, we use the LibTorch library.
LibTorch is built on top of the PyTorch C++ frontend API, and provides a set of C++ classes and functions that expose the core functionality of PyTorch.
The version adopted by TensorRT is 1.7 and by LibTorch, we have to keep the same version as the one used in training, which in our case is 1.6. The GPU related libraries are CUDA 11.0.0 and CuDNN 8.0.0.

7.3.2.2.2 Integration with existing systems

Integrating AI projects into existing systems can be a complex and time-consuming process. It is crucial to ensure that the AI system is compatible with the existing system and does not disrupt or interfere with its normal operations.

To achieve this, we must ensure that the AI system has compatible hardware and software requirements and use versions of libraries that have already been used by the rest of the system. Careful planning and coordination are also essential to ensure a smooth and successful integration process.

7.3.2.2.3 Pre-processing images phase

However, the images must first go through the pre-processing stage before the deep learning models can begin processing them. It is an important step in the deep learning pipeline to prepare the input data for inference in the same manner as it was prepared for training. The main goal of preprocessing is to transform the raw input data into a form that is suitable to be treated with the deep learning model. This can involve a range of tasks, including data cleaning, normalization, augmentation, and feature engineering, depending on the characteristics of the data and the requirements of the task. One of the most important preprocessing used before each model developed in our system is mean subtraction. This preprocessing step involves subtracting the mean value of the input data from each channel. The mean value subtracted from each channel are 0.485 for red, 0.456 for green, and 0.406 for blue channel. Then, dividing the input data by the standard deviation to normalize the input data and remove any scaling issues that may be present. The resulting value is divided by 0.229 for red, 0.224 for green, and 0.225 for blue channel as standard deviation. This can be useful for improving the convergence and generalization of the model. It is important to keep in mind that developing an image processing pipeline for a product requires a lot of work and attention to detail. It involves a range of tasks, including capturing frames from industrial cameras, correcting lens distortion, rectifying images, and communicating the results. While it may seem like these tasks can be accomplished with just a few calls to OpenCV functions [288], this is often not the case. In fact, using low-level SDKs provided by camera manufacturers, can require hundreds of lines of code just to capture a

frame and convert it to a cv::Mat format. It is important to carefully consider the requirements and constraints of the product, and to choose the appropriate tools and techniques to ensure the smooth and efficient operation of the image processing pipeline. OpenCV framework is an Open Source Computer Vision (OpenCV) is a popular library for computer vision and image processing tasks utilized in a variety of applications and industries. It is written initially in C and upgrade then to C++ to take advantage of templates.

7.3.2.3 Deep learning visualization

Deep learning visualization is the process of using visual tools and techniques to understand, evaluate, and interpret the outputs of deep learning models. Visualization can be used to analyze the behavior of deep learning models, identify patterns and trends in the data, and assess the model's accuracy and reliability. There are several tools that can be used to visualize deep learning models and results as DeepVision, Neu.ro, Biases& Weights etc. However, in what follows, we will only focus on the tools used in my thesis work; TensorBoard and Netron.

TensorBoard is a visualization tool provided by TensorFlow that widely used in the machine learning community. It allows users to view and analyze various aspects of their model, such as training and validation loss, accuracy, and gradient flow. It is particularly useful for us to analyzing and debugging deep learning models, as it provides a range of features for visualizing complex neural network architectures and tracking the training process.

Netron is a tool for visualizing deep learning models, including their architecture, weights, and activations. It supports a variety of model formats including PyTorch, TensorFlow, and ONNX, and can be used to analyze and debug models. It helps us especially to troubleshoot issues with model conversion, such as when converting a PyTorch model to TensorRT.

Last But Not Least, OpenGL is the useful tool used in our work to display and render the results of the 3D pose estimation in real time. Open Graphics Library

(OpenGL) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics.

7.4 Industrial Software system delivery

7.4.1 Software delivery

Software delivery is the entire process of getting a software product to customers, from conceptualization, through development, ending in the actual purchase, installation and deployment to the customer. The team prepared the software product for release using software delivery models, also referred to as the software delivery life-cycle, the software delivery pipeline, or the software delivery process. There are many models that have been developed. Yet, no single one of them works in an optimal way. Thus, many companies combine different software delivery models.

The goal of software delivery is to ensure that the software is delivered to users in a timely, reliable, and high-quality manner. To achieve this, a variety of activities may be required, such as:

Coding and development: Software delivery typically involves coding and developing the software using a range of tools and technologies, such as programming languages, libraries, and frameworks.

Testing and debugging: Software delivery also involves testing the software to ensure that it meets the required specifications and is free of defects. This can involve a range of activities, such as unit testing, integration testing, and system testing.

Packaging and deployment: Software delivery involves packaging the software in a format that is suitable for distribution and deployment, such as an installer or a container. It also involves deploying the software to the appropriate environment, such as a development server or a production server.

Maintenance and updates: Software delivery also involves maintaining and updating the software over time to fix defects, add new features, or address security vulnerabilities. This can involve a range of activities, such as patch management, version control, and release management

To improve the efficiency and effectiveness of the software delivery process, we benefit from DevOps practices. By implementing DevOps practices, industries can reduce the time and effort required to release and deploy software updates and new features, increase the reliability and quality of software releases, enable faster feedback and iteration on software development projects, improve communication and collaboration between development and operations teams, and automate and streamline the software delivery process.

7.4.2 DevOps delivery pipeline

The term DevOps was introduced to represent a movement in computer engineering and technical practice aimed at unifying software development (dev) and IT infrastructure administration/Operations (ops). Which fosters better collaboration between development and operations teams to deliver value to users faster. Indeed, DevOps-based techniques reduce the barriers between developers, who want to innovate and deliver faster, and operations, who want to ensure the stability of production systems and the quality of the changes they make to systems. This has reduced the time between system changes and the deployment of updates and changes to production, while maintaining software quality in terms of code and delivery. Continuous delivery allows for incremental and continuous delivery of frequent software releases by systematically automating the creation, testing, and validation of software without having to rebuild and redeploy the entire application. There are various tools and technologies that can be used in DevOps, which are chosen based on the specific needs and constraints of the project. These tools include **version control systems** such as Git, which help developers track and manage

changes to code over time; **Continuous integration and delivery CI/CD tools** like Jenkins, Travis CI, and CircleCI, which automate the process of building, testing, and deploying code; **deployment tools** like Ansible, Terraform, and Docker, which automate the deployment and management of software applications; **package management tools** like npm, pip, and Maven, which help developers to manage and distribute software packages and dependencies; **monitoring tools** like New Relic and Splunk, which allow monitoring the performance, availability, and security of their applications and infrastructure; and **collaboration and communication tools** like Slack and Trello, allow developers to communicate and collaborate with other team members and stakeholders. These tools can be used to track and manage code changes, automate the build, test, and deployment process, manage software packages and dependencies, deploy and manage software applications, monitor the performance and availability of applications and infrastructure, and facilitate communication and collaboration among team members and stakeholders.

In Pryntec, we use Gitlab which can be used to store, version, and share code, and to collaborate with other developers. GitLab is a popular open-source tool in the industry for managing software development projects and delivering software. It is a web-based platform that combines version control, project management, and continuous integration and delivery (CI/CD) tools into a single package.

There are several reasons why GitLab is the popular choice for software delivery in the industry is that it is all-in-one approach, open-source nature, large community, and ability to handle large and complex projects with features such as scalability, performance, and security. It offers a range of tools and features commonly used in software development, including version control, package management, project management, and CI/CD, and can be customized and extended to meet specific needs. In more details:

Version control which allows tracking changes to the codebase over time and making it easier to understand how the code has evolved, and rolling back to a previous version of the code if a problem is introduced. It also facilitates

collaboration between multiple developers by working on the same codebase simultaneously and creating new branches to develop new features or fixing bugs without affecting the main codebase.

Built-in continuous integration and delivery (CI/CD) functionality which allows developers to automate the build process for software applications as compiling and testing tasks. In our GitLab setup, we compile all programs every night in order to quickly identify errors in code that was added during the day. If a function doesn't work or a test fails, we are alerted so that we can take action to resolve the issue. This helps us ensure the quality and reliability of our software by catching problems as early as possible in the development process.

Project management features allows developers to create and assign tasks, set deadlines, and track progress on projects. This helps to ensure that work is organized, and that team members are aware of their responsibilities. GitLab includes features such as code review and merge request approval workflows that allow developers to review and discuss code changes before they are merged into the main codebase. This helps to ensure that code changes are of high quality and adhere to standards.

Package management functionality which allows developers to manage and distribute software packages and dependencies. It provides tools for installing, updating, and uninstalling software packages and their dependencies, and for managing the dependencies of a software project. It helps to ensure that they have the right versions of packages, the right tools and libraries available to support their work.

7.4.3 MLOps delivery pipelines

Traditional software is generally created to perform specific tasks or functions, and it operates using predetermined rules or algorithms. This type of software does not utilize machine learning techniques. When traditional software is compiled, it produces an executable file.

On the other hand, machine learning software is designed to learn from data and adapt to changing input. It uses algorithms and techniques that allow it to improve its performance over time, based on the data it processes. As a result, machine learning projects typically consist of the code source and input data, to produce trained models and performance metrics as output rather than executable files.

Thus, in addition to versioning the code, it is also important to version the data because it is a critical component of the machine learning model, and changes to the data can have a significant impact on the performance of the model. Using Git or other version control systems to version large data files can be challenging, as they are typically designed to handle relatively small files. In order to version large data files, it is often necessary to use specialized tools such as DVC (Data Version Control) or Pachyderm, which are specifically designed to handle large data files and provide version control and data lineage for machine learning projects. This can help the developers to ensure that the data used in the machine learning model is consistent and reliable, which is essential for achieving good model performance.

In addition, the process of developing and training machine learning models often involves experimentation with different configurations, such as altering datasets, changing batch size, using data augmentation, and adjusting the learning rate. It is then important to have a way to track and manage machine learning experiments in order to ensure that the final model is robust, accurate, and reproducible.

To manage these processes, Machine Learning Operations (MLOps) practices are often employed. It is a practice that aims to improve the collaboration and automation of machine learning workflows. It combines principles and practices from the

fields of machine learning and DevOps, with the goal of making it easier to develop, deploy, and maintain machine learning models in production environments.
The MLOps process can generally be divided into three broad phases: designing the ML-powered application, ML experimentation and development, and ML operations [289] as illustrated in the figure 7.6.
During the design phase, data scientists define the requirements and goals for the ML-powered application, and design the overall architecture and workflow. This includes identifying the data sources and algorithms that will be used, as well as the infrastructure and tools that will be needed to support the application.
During the ML experimentation and development phase, data scientists and other team members develop and test the machine learning models and algorithms that will be used in the application. This typically involves iterative experimentation and testing, using techniques such as data augmentation, hyperparameter tuning, and model selection.
Finally, during the ML operations phase, the machine learning models and algorithms are deployed and integrated into the production environment. This involves deploying the models to the appropriate infrastructure, and setting up processes for monitoring, maintaining, and updating the models over time.

Overall, the goal of the MLOps process is to facilitate the development, deployment, and maintenance of machine learning models in production environments in a way that is efficient and effective for data scientists and other team members. This may involve balancing the need for high performance with the need for high accuracy, in order to optimize the overall effectiveness of the ML-powered application.
There are many tools that are commonly used together as part of an overall MLOps process which could serve different purposes. In one hand, DVC and Pachyderm are examples of tools that can be used to manage the data and code associated with machine learning projects. They allow data scientists to track, version and share their experiments, data files, metrics and models, to reproduce results and

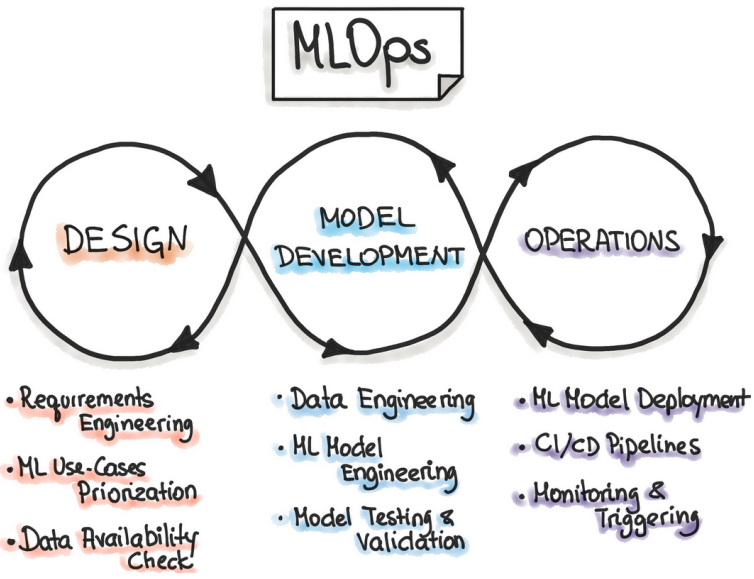

Figure 7.6: MLOps process. [290].

automate the process of data processing and model training. Both integrate with a range of machine learning libraries and frameworks. DVC is designed to be used in conjunction with Git. While Pachyderm offers additional features such as project management and more customization and collaboration options, as well as integration with other tools and visualization and analysis capabilities.

In the other hand, Weights & Biases (W&B) and Neptune are examples of tools that are used for tracking and managing machine learning experiments. They provide features for tracking and organizing machine learning experiments, such as experiment tagging and filtering, and the ability to see the results of previous experiments. They provide tools for tuning and optimizing hyperparameters, such as automated hyperparameter search and Bayesian optimization. They provide interactive visualizations of performance metrics, such as training and validation loss, accuracy, and precision. They both offer data versioning and integration with various ML libraries. However, they have different strengths and focus areas. Neptune offers additional features such as projects and commenting, and it provides

more options for customization, more features for collaboration and integration with other tools, while W&B offers commenting and sharing, and focuses more on visualization and interactive analysis as shown in table 7.2. Data scientists should choose the tool that best fits their needs and workflow.

Feature	**W&B**	**Neptune**	**DVC**	**Pachyderm**
Data versioning	Yes	Yes	Yes	Yes
ML library integration	Yes	Yes	Yes	Yes
Project management	No	Yes	No	Yes
Commenting	Yes	Yes	No	Yes
Collaboration features	Limited	Many options	Limited	Many options
Customization	Limited	Many options	Limited	Many options
Visualization	Strong focus	Moderate focus	No	No
Interactive analysis	Strong focus	Moderate focus	No	No
Integration with other tools	Limited	Yes	Limited	Yes

Table 7.2: table comparing W&B, Neptune, DVC and Pachyderm

Overall, DVC and Pachyderm are primarily focused on data versioning and integration with ML libraries, while W&B and Neptune offer more options for collaboration, customization, and integration with other tools. W&B also has a focus on visualization and interactive analysis. Data scientists should choose the tool that best fits their needs and workflow.

Ultimately, the goal of MLOps is to enable data scientists and other team members to focus on the core task of developing and improving machine learning models, rather than being slowed down by the logistics of the development and deployment process.

7.5 Conclusion

Industrialization refers to the process of using technology, machines, and specialized techniques to produce goods and services on a large scale, often in a factory setting. In this CIFRE thesis, we added a functionality to the video surveillance system PrynCore, developed by the company Pryntec. The system as well as my part of the work are based on image processing and deep learning algorithms that lead to a valuable help for decision-making or to automate tasks based on the processed images.

The design of high-performance image processing, as in many other areas, begins with identifying a problem and analyzing state-of-the-art solutions. Then, the researcher/developer conceives and elaborates an idea to improve the resolution of this problem, which is then developed in the form of application/product composed of many algorithms and implemented in a programming language taking into consideration the hardware which meets the application constraints and economics. Finally, the developed solution is tested and evaluated, and if necessary, further optimized and refined. This chapter focuses on the deployment and inference phase, also mentioning information useful for the training phase that was not covered in previous chapters. It discusses the challenges and considerations that need to be taken into account when deploying machine learning models in production environments and describes the various tools and techniques that can be used to optimize the performance and reliability of the models. In addition, the chapter covers the basics of model inference, including how to make predictions using trained models and how to evaluate the accuracy of the models. We have seen how

deep learning visualization can be a powerful tool for understanding and improving the performance of deep learning models.

We discussed several approaches we have taken to optimize the performance of deep learning models, such as vectorization, parallel computing on GPUs, and distributed training. As well as the support of the TensorRT library in inference. Using these techniques, we were able to improve the speed and efficiency of the models and allow them to handle larger and more complex data sets.

In addition, we discussed the importance of software delivery chain management (DevOps) and machine learning monitoring tool (MLOps) in ensuring the quality and reliability of machine learning models in production. In fact, both MLOps and DevOps involve collaboration and automation, but they focus on different areas of work. DevOps is focused on software development and operations more generally, while MLOps is specifically focused on machine learning workflows. Both practices aim to improve the efficiency and reliability of the processes they support, and to make it easier to develop, deploy and maintain software and machine learning models in production environments.

It should also be noted that software delivery and DevOps are related, but they are not the same thing. Software delivery refers to the process of publishing and distributing software updates or new software products to users. It involves a series of activities, including coding, testing, debugging, packaging and deploying the software, as well as managing and maintaining the software throughout its life-cycle. DevOps, meanwhile, is a set of practices and culture that aims to promote collaboration and communication between software development and IT operations teams. It aims to automate and optimize the software delivery process, with an emphasis on continuous integration, continuous delivery and continuous deployment.

In practice, software delivery is a subset of DevOps, as it is one aspect of the larger process of creating, testing and deploying software. However, DevOps also involves other activities, such as infrastructure management, monitoring and security.

Overall, software delivery focuses on the process of releasing and distributing software, while DevOps is a broader approach to software development and delivery that emphasizes collaboration, automation and continuous improvement. Implementing DevOps practices can help organizations improve the effectiveness and efficiency of their software delivery process.

In summary, effective and reliable implementation of machine learning models in production requires careful attention to the technical and organizational aspects of machine learning industrialization. Taking these aspects into account can help to improve the performance of models, ensure their quality and reliability, and facilitate their maintenance and evolution over time.

7.6 General Conclusion and Perspectives

This thesis navigates through the complex yet gratifying journey of human pose estimation, a field that has seen rapid growth and development in recent years. The focus has been on the absolute estimation of multi-person 3D poses from monocular RGB images.

The thesis is divided into three main parts. The first part offers an in-depth review of the literature, starting with an exploration of deep learning techniques and progressing to a detailed examination of human pose estimation. The second part delves into the system we developed, breaking down the architecture of each component and the software implementation into three chapters. The final part discusses the industrialization process, which involves the use of technology and specialized techniques to produce goods and services on a large scale.

Our methodology begins with the detection of individuals in the image, each assigned a unique identifier and tracked throughout the sequence. For each frame and individual, we first predict the 2D coordinates of all joints, followed by a grouping process to predict 3D skeletons using temporal information. We have trained a

network to estimate the depth of one of the joint points, enabling us to accurately determine the true position of each person in the room relative to the camera.
Our contribution is the development of a solution we named Root-GAST-Net, which is designed to meet the initial requirements and outperform existing methods. The system is the result of a series of improvements that have increased accuracy by more than 8.8 percentage points on 3D-PCK*abs* on the MuPoTS-3D dataset. Furthermore, the system can be used in real-time, as the execution time of each frame containing one person is approximately 60 milliseconds using the Nvidia GeForce GTX 1080. This time can be further reduced using high-performance materials and FP16 precision.
As part of a CIFRE thesis, we have incorporated a new feature into the PrynCore video surveillance system, developed by Pryntec. Both the system and our contribution are rooted in image processing and deep learning algorithms, providing valuable assistance for decision-making and task automation based on the processed images.
Moving forward, our contribution is focused on advancing both relative and absolute 3D pose estimation by reconstructing information from 2D joints. In future works, we have a particular interest in enhancing the initial stage of the process. To achieve this goal, we propose a novel method called YoloPose, which seamlessly integrates two crucial tasks: human detection and 2D keypoints estimation. The fundamental concept behind YoloPose is to utilize the anchor pose as a reference for identifying the 2D poses within the images, rather than relying on traditional bounding box rectangles used in YOLO. However, it is important to acknowledge that the estimation of each joint may introduce a certain level of error.

Bibliography

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[2] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5:64–67, 2001.

[3] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017.

[4] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[5] Saul Dobilas. Lstm recurrent neural networks — how to teach a network to remember the past, 2022.

[6] Abdelhadi Azzouni and Guy Pujolle. A long short-term memory recurrent neural network framework for network traffic matrix prediction. *arXiv preprint arXiv:1705.05690*, 2017.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[8] Rahul Dey and Fathi M Salem. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

[9] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165, 2017.

[10] Guijuan Zhang, Yang Liu, and Xiaoning Jin. A survey of autoencoder-based recommender systems. *Frontiers of Computer Science*, 14:430–450, 2020.

[11] Jingyuan Liu, Hongbo Fu, and Chiew-Lan Tai. Posetween: Pose-driven tween animation.

[12] Nora S Willett, Hijung Valentina Shin, Zeyu Jin, Wilmot Li, and Adam Finkelstein. Pose2pose: pose selection and transfer for 2d character animation. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 88–99, 2020.

[13] Yuen-Jen Lin, Hsuan-Kai Kao, Yih-Chih Tseng, Ming Tsai, and Li Su. A human-computer duet system for music performance. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 772–780, 2020.

[14] Taravat Anvari, Kyoungju Park, and Ganghyun Kim. Upper body pose estimation using deep learning for a virtual reality avatar. *Applied Sciences*, 13(4):2460, 2023.

[15] Gokcen Cimen, Christoph Maurhofer, Bob Sumner, and Martin Guay. Ar poser: Automatically augmenting mobile pictures with digital avatars imitating poses. In *12th international conference on computer graphics, visualization, computer vision and image processing*, 2018.

[16] Manolis Vasileiadis, Sotiris Malassiotis, Dimitrios Giakoumis, Christos-Savvas Bouganis, and Dimitrios Tzovaras. Robust human pose tracking for realistic service robot applications. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1363–1372, 2017.

[17] Kengo Ichihara, Masaru Takeuchi, and Jiro Katto. Accuracy evaluations of video anomaly detection using human pose estimation. In *2020 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–2. IEEE, 2020.

[18] Ismail Haritaoglu, David Harwood, and Larry S. Davis. W/sup 4: real-time surveillance of people and their activities. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):809–830, 2000.

[19] Feng-Cheng Lin, Huu-Huy Ngo, Chyi-Ren Dow, Ka-Hou Lam, and Hung Linh Le. Student behavior recognition system for the classroom environment based on skeleton pose estimation and person detection. *Sensors*, 21(16):5314, 2021.

[20] The Waymo Team. Utilizing key point and pose estimation for the task of autonomous driving. Retrieved on February 22, 2022 from https://blog.waymo.com/2022/02/utilizing-key-point-and-pose-estimation.html.

[21] Liubov Zatolokina. Human pose estimation technology capabilities and use cases in 2023. Retrieved on Oct 21, 2022 from https://mobidev.biz/blog/human-pose-estimation-technology-guide.

[22] Yucheng Chen, Yingli Tian, and Mingyi He. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, 192:102897, 2020.

[23] Thomas B Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer vision and image understanding*, 81(3):231–268, 2001.

[24] Nikolaos Sarafianos, Bogdan Boteanu, Bogdan Ionescu, and Ioannis A Kakadiaris. 3d human pose estimation: A review of the literature and analysis of covariates. *Computer Vision and Image Understanding*, 152:1–20, 2016.

[25] Zhao Liu, Jianke Zhu, Jiajun Bu, and Chun Chen. A survey of human pose estimation: the body parts parsing based methods. *Journal of Visual Communication and Image Representation*, 32:10–19, 2015.

[26] Wenjuan Gong, Xuena Zhang, Jordi Gonzàlez, Andrews Sobral, Thierry Bouwmans, Changhe Tu, and El-hadi Zahzah. Human pose estimation from monocular images: A comprehensive survey. *Sensors*, 16(12):1966, 2016.

[27] Qi Dang, Jianqin Yin, Bin Wang, and Wenqing Zheng. Deep learning based 2d human pose estimation: A survey. *Tsinghua Science and Technology*, 24(6):663–676, 2019.

[28] Yi Li and Zhengxing Sun. Vision-based human pose estimation for pervasive computing. In *Proceedings of the 2009 workshop on Ambient media computing*, pages 49–56, 2009.

[29] Hong-Bo Zhang, Yi-Xiang Zhang, Bineng Zhong, Qing Lei, Lijie Yang, Ji-Xiang Du, and Duan-Sheng Chen. A comprehensive survey of vision-based human action recognition methods. *Sensors*, 19(5):1005, 2019.

[30] Xavier Perez-Sala, Sergio Escalera, Cecilio Angulo, and Jordi Gonzalez. A survey on model based approaches for 2d and 3d visual human pose recovery. *Sensors*, 14(3):4189–4210, 2014.

[31] Ronald Poppe. Vision-based human motion analysis: An overview. *Computer vision and image understanding*, 108(1-2):4–18, 2007.

[32] Mark Dilsizian. *Hybrid discriminative-generative methods for human pose reconstruction from monocular imagery*. Rutgers The State University of New Jersey-New Brunswick, 2016.

[33] Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100(1):67–92, 1973.

[34] Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International journal of computer vision*, 61(1):55–79, 2005.

[35] Elisabeta Marinoiu, Dragos Papava, and Cristian Sminchisescu. Pictorial human spaces: How well do humans perceive a 3d articulated pose? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1289–1296, 2013.

[36] Vasileios Belagiannis, Sikandar Amin, Mykhaylo Andriluka, Bernt Schiele, Nassir Navab, and Slobodan Ilic. 3d pictorial structures for multiple human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1669–1676, 2014.

[37] Leonid Sigal, Michael Isard, Horst Haussecker, and Michael J Black. Loose-limbed people: Estimating 3d human pose and motion using non-parametric belief propagation. *International journal of computer vision*, 98(1):15–48, 2012.

[38] Ryuzo Okada and Stefano Soatto. Relevant feature selection for human pose estimation and localization in cluttered images. In *European Conference on Computer Vision*, pages 434–445. Springer, 2008.

[39] Weipeng Zhang, Jie Shen, Guangcan Liu, and Yong Yu. A latent clothing attribute approach for human pose estimation. In *Asian Conference on Computer Vision*, pages 146–161. Springer, 2014.

[40] S Sedai, Mohammed Bennamoun, and DQ Huynh. Evaluating shape and appearance descriptors for 3d human pose estimation. In *2011 6th IEEE Conference on Industrial Electronics and Applications*, pages 293–298. IEEE, 2011.

[41] Suman Sedai, Mohammed Bennamoun, and Du Huynh. Context-based appearance descriptor for 3d human pose estimation from monocular images. In *2009 Digital Image Computing: Techniques and Applications*, pages 484–491. IEEE, 2009.

[42] Huazhong Ning, Wei Xu, Yihong Gong, and Thomas Huang. Discriminative learning of visual words for 3d human pose estimation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[43] Grégory Rogez, Jonathan Rihan, Srikumar Ramalingam, Carlos Orrite, and Philip HS Torr. Randomized trees for human pose detection. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[44] Ankur Agarwal and Bill Triggs. Recovering 3d human pose from monocular images. *IEEE transactions on pattern analysis and machine intelligence*, 28(1):44–58, 2005.

[45] Gregory Shakhnarovich, Paul Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *Computer Vision, IEEE International Conference on*, volume 3, pages 750–750. IEEE Computer Society, 2003.

[46] Dariu M Gavrila. A bayesian, exemplar-based approach to hierarchical shape matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1408–1421, 2007.

[47] Miodrag Dimitrijevic, Vincent Lepetit, and Pascal Fua. Human body pose detection using bayesian spatio-temporal templates. *Computer vision and image understanding*, 104(2-3):127–139, 2006.

[48] Mathieu Salzmann and Raquel Urtasun. Combining discriminative and generative methods for 3d deformable surface and articulated pose reconstruction. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 647–654. IEEE, 2010.

[49] Leonid Sigal, Alexandru Balan, and Michael Black. Combined discriminative and generative articulated pose and non-rigid shape estimation. *Advances in neural information processing systems*, 20:1337–1344, 2007.

[50] Ce Zheng, Wenhan Wu, Chen Chen, Taojiannan Yang, Sijie Zhu, Ju Shen, Nasser Kehtarnavaz, and Mubarak Shah. Deep learning-based human pose estimation: A survey. *ACM Computing Surveys*, 2020.

[51] Wu Liu, Qian Bao, Yu Sun, and Tao Mei. Recent advances of monocular 2d and 3d human pose estimation: a deep learning perspective. *ACM Computing Surveys*, 55(4):1–41, 2022.

[52] Tomas Pfister, James Charles, and Andrew Zisserman. Flowing convnets for human pose estimation in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1913–1921, 2015.

[53] Martin Kiefel and Peter Vincent Gehler. Human pose estimation with fields of parts. In *European Conference on Computer Vision*, pages 331–346. Springer, 2014.

[54] Lipeng Ke, Ming-Ching Chang, Honggang Qi, and Siwei Lyu. Multi-scale structure-aware network for human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 713–728, 2018.

[55] Yu Chen, Chunhua Shen, Xiu-Shen Wei, Lingqiao Liu, and Jian Yang. Adversarial posenet: A structure-aware convolutional network for human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1212–1221, 2017.

[56] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. Towards accurate multi-person pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4903–4911, 2017.

[57] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. Rmpe: Regional multi-person pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2334–2343, 2017.

[58] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 466–481, 2018.

[59] Wenbo Li, Zhicheng Wang, Binyi Yin, Qixiang Peng, Yuming Du, Tianzi Xiao, Gang Yu, Hongtao Lu, Yichen Wei, and Jian Sun. Rethinking on multi-stage networks for human pose estimation. *arXiv preprint arXiv:1901.00148*, 2019.

[60] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.

[61] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807, 2014.

[62] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4733–4742, 2016.

[63] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N Metaxas. Semantic graph convolutional networks for 3d human pose regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3425–3435, 2019.

[64] Adrian Bulat and Georgios Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. In *European Conference on Computer Vision*, pages 717–732. Springer, 2016.

[65] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[66] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

[67] Norimichi Ukita and Yusuke Uematsu. Semi-and weakly-supervised human pose estimation. *Computer Vision and Image Understanding*, 170:67–78, 2018.

[68] Guanghan Ning, Zhi Zhang, and Zhiquan He. Knowledge-guided deep fractal neural networks for human pose estimation. *IEEE Transactions on Multimedia*, 20(5):1246–1259, 2017.

[69] Ke Li, Shijie Wang, Xiang Zhang, Yifan Xu, Weijian Xu, and Zhuowen Tu. Pose recognition with cascade transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1944–1953, 2021.

[70] Sijin Li, Zhi-Qiang Liu, and Antoni B Chan. Heterogeneous multi-task learning for human pose estimation with deep convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 482–489, 2014.

[71] Xiaochuan Fan, Kang Zheng, Yuewei Lin, and Song Wang. Combining local appearance and holistic view: Dual-source deep neural networks for human pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1347–1355, 2015.

[72] Diogo C Luvizon, David Picard, and Hedi Tabia. 2d/3d pose estimation and action recognition using multitask deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5137–5146, 2018.

[73] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656, 2015.

[74] Ita Lifshitz, Ethan Fetaya, and Shimon Ullman. Human pose estimation using deep consensus voting. In *European Conference on Computer Vision*, pages 246–260. Springer, 2016.

[75] Hao Jiang. 3d human pose reconstruction using millions of exemplars. In *2010 20th International Conference on Pattern Recognition*, pages 1674–1677. IEEE, 2010.

[76] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in neural information processing systems*, pages 2277–2287, 2017.

[77] Wei Tang, Pei Yu, and Ying Wu. Deeply learned compositional models for human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 190–206, 2018.

[78] Wei Yang, Shuang Li, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Learning feature pyramids for human pose estimation. In *proceedings of the IEEE international conference on computer vision*, pages 1281–1290, 2017.

[79] Chia-Jung Chou, Jui-Ting Chien, and Hwann-Tzong Chen. Self adversarial training for human pose estimation. In *2018 Asia-Pacific Signal and Information Processing*

Association Annual Summit and Conference (APSIPA ASC), pages 17–30. IEEE, 2018.

[80] Jia Li, Wen Su, and Zengfu Wang. Simple pose: Rethinking and improving a bottom-up approach for multi-person pose estimation. *arXiv preprint arXiv:1911.10529*, 2019.

[81] Z Cao, R Wang, X Wang, et al. Improving human pose estimation with self-attention generative adversarial networks [c]. In *2019 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 567–572, 2019.

[82] Jia Li, Wen Su, and Zengfu Wang. Simple pose: Rethinking and improving a bottom-up approach for multi-person pose estimation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11354–11361, 2020.

[83] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[84] Yu Chen, Chunhua Shen, Hao Chen, Xiu-Shen Wei, Lingqiao Liu, and Jian Yang. Adversarial learning of structure-aware fully convolutional networks for landmark localization. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[85] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3d human pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2640–2649, 2017.

[86] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019.

[87] Daniel Groos, Heri Ramampiaro, and Espen AF Ihlen. Efficientpose: Scalable single-person pose estimation. *Applied Intelligence*, 51(4):2518–2533, 2021.

[88] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[89] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[90] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[91] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[92] Shaoli Huang, Mingming Gong, and Dacheng Tao. A coarse-fine network for keypoint localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3028–3037, 2017.

[93] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7103–7112, 2018.

[94] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[95] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4929–4937, 2016.

[96] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision*, pages 34–50. Springer, 2016.

[97] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.

[98] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis amp; Machine Intelligence*, 43(01):172–186, 2021.

[99] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–286, 2018.

[100] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. Pifpaf: Composite fields for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11977–11986, 2019.

[101] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[102] Feng Zhang, Xiatian Zhu, Hanbin Dai, Mao Ye, and Ce Zhu. Distribution-aware coordinate representation for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7093–7102, 2020.

[103] Liefeng Bo, Cristian Sminchisescu, Atul Kanaujia, and Dimitris Metaxas. Fast algorithms for large scale conditional 3d prediction. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[104] Greg Mori and Jitendra Malik. Recovering 3d human body configurations using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1052–1062, 2006.

[105] Cristian Sminchisescu, Atul Kanaujia, Zhiguo Li, and Dimitris Metaxas. Discriminative density propagation for 3d human motion estimation. In *2005 IEEE Computer*

Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 1, pages 390–397. IEEE, 2005.

[106] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013.

[107] Catalin Ionescu, Fuxin Li, and Cristian Sminchisescu. Latent structured models for human pose estimation. In *2011 International Conference on Computer Vision*, pages 2220–2227. IEEE, 2011.

[108] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *CVPR 2011*, pages 1297–1304. Ieee, 2011.

[109] Ankur Agarwal and Bill Triggs. 3d human pose from silhouettes by relevance vector regression. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004.

[110] Liefeng Bo and Cristian Sminchisescu. Twin gaussian processes for structured prediction. *International Journal of Computer Vision*, 87(1-2):28, 2010.

[111] Ilya Kostrikov and Juergen Gall. Depth sweep regression forests for estimating 3d human pose from images. In *BMVC*, volume 1, page 5, 2014.

[112] Catalin Ionescu, Joao Carreira, and Cristian Sminchisescu. Iterated second-order label sensitive pooling for 3d human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1661–1668, 2014.

[113] Sijin Li and Antoni B Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *Asian Conference on Computer Vision*, pages 332–347. Springer, 2014.

[114] Xingyi Zhou, Xiao Sun, Wei Zhang, Shuang Liang, and Yichen Wei. Deep kinematic pose regression. In *European Conference on Computer Vision*, pages 186–201. Springer, 2016.

[115] Bugra Tekin, Isinsu Katircioglu, Mathieu Salzmann, Vincent Lepetit, and Pascal Fua. Structured prediction of 3d human pose with deep neural networks. *arXiv preprint arXiv:1605.05180*, 2016.

[116] Bugra Tekin, Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Direct prediction of 3d body poses from motion compensated sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 991–1000, 2016.

[117] Leonid Sigal, Alexandru O Balan, and Michael J Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International journal of computer vision*, 87(1-2):4, 2010.

[118] Sungheon Park, Jihye Hwang, and Nojun Kwak. 3d human pose estimation using convolutional neural networks with 2d pose information. In *European Conference on Computer Vision*, pages 156–169. Springer, 2016.

[119] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7025–7034, 2017.

[120] Denis Tome, Chris Russell, and Lourdes Agapito. Lifting from the deep: Convolutional 3d pose estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2500–2509, 2017.

[121] Mona Fathollahi Ghezelghieh, Rangachar Kasturi, and Sudeep Sarkar. Learning camera viewpoint using cnn to improve 3d body pose estimation. In *2016 fourth international conference on 3D vision (3DV)*, pages 685–693. IEEE, 2016.

[122] Sijin Li, Weichen Zhang, and Antoni B Chan. Maximum-margin structured learning with deep networks for 3d human pose estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 2848–2856, 2015.

[123] Bugra Tekin, Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. Learning to fuse 2d and 3d image cues for monocular body pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3941–3950, 2017.

[124] Francesc Moreno-Noguer. 3d human pose estimation from a single image via distance matrix regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2823–2832, 2017.

[125] Hsi-Jian Lee and Zen Chen. Determination of 3d human body postures from a single view. *Computer Vision, Graphics, and Image Processing*, 30(2):148–168, 1985.

[126] Xiaowei Zhou, Menglong Zhu, Georgios Pavlakos, Spyridon Leonardos, Konstantinos G Derpanis, and Kostas Daniilidis. Monocap: Monocular human motion capture using a cnn coupled with a geometric prior. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):901–914, 2018.

[127] Jiajun Wu, Tianfan Xue, Joseph J Lim, Yuandong Tian, Joshua B Tenenbaum, Antonio Torralba, and William T Freeman. Single image 3d interpreter network. In *European Conference on Computer Vision*, pages 365–382. Springer, 2016.

[128] Wei Yang, Wanli Ouyang, Xiaolong Wang, Jimmy Ren, Hongsheng Li, and Xiaogang Wang. 3d human pose estimation in the wild by adversarial learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5255–5264, 2018.

[129] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 529–545, 2018.

[130] Georgios Pavlakos, Xiaowei Zhou, and Kostas Daniilidis. Ordinal depth supervision for 3d human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7307–7316, 2018.

[131] Bastian Wandt and Bodo Rosenhahn. Repnet: Weakly supervised training of an adversarial reprojection network for 3d human pose estimation. In *Proceedings of*

the IEEE Conference on Computer Vision and Pattern Recognition, pages 7782–7791, 2019.

[132] Edgar Simo-Serra, Arnau Ramisa, Guillem Alenyà, Carme Torras, and Francesc Moreno-Noguer. Single image 3d human pose estimation from noisy observations. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2673–2680. IEEE, 2012.

[133] Camillo J Taylor. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Computer Vision and Image Understanding*, 80(3):349–363, 2000.

[134] Grégory Rogez and Cordelia Schmid. Mocap-guided data augmentation for 3d pose estimation in the wild. In *Advances in neural information processing systems*, pages 3108–3116, 2016.

[135] Gregory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. Lcr-net: Localization-classification-regression for human pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3433–3441, 2017.

[136] Gerard Pons-Moll, David J Fleet, and Bodo Rosenhahn. Posebits for monocular human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2344, 2014.

[137] Yu Du, Yongkang Wong, Yonghao Liu, Feilin Han, Yilin Gui, Zhen Wang, Mohan Kankanhalli, and Weidong Geng. Marker-less 3d human motion capture with monocular image sequence and height-maps. In *European Conference on Computer Vision*, pages 20–36. Springer, 2016.

[138] Xiaowei Zhou, Menglong Zhu, Spyridon Leonardos, and Kostas Daniilidis. Sparse representation for 3d shape estimation: A convex relaxation approach. *IEEE transactions on pattern analysis and machine intelligence*, 39(8):1648–1661, 2016.

[139] Xiaowei Zhou, Menglong Zhu, Spyridon Leonardos, Konstantinos G Derpanis, and Kostas Daniilidis. Sparseness meets deepness: 3d human pose estimation from

monocular video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4966–4975, 2016.

[140] Ching-Hang Chen and Deva Ramanan. 3d human pose estimation= 2d pose estimation+ matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7035–7043, 2017.

[141] Ankur Gupta, Julieta Martinez, James J Little, and Robert J Woodham. 3d pose from motion for cross-view action recognition via non-linear circulant temporal encoding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2601–2608, 2014.

[142] Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Reconstructing 3d human pose from 2d image landmarks. In *European conference on computer vision*, pages 573–586. Springer, 2012.

[143] Chunyu Wang, Yizhou Wang, Zhouchen Lin, Alan L Yuille, and Wen Gao. Robust estimation of 3d human poses from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2361–2368, 2014.

[144] Hashim Yasin, Umar Iqbal, Bjorn Kruger, Andreas Weber, and Juergen Gall. A dual-source approach for 3d pose estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4948–4956, 2016.

[145] Edgar Simo-Serra, Ariadna Quattoni, Carme Torras, and Francesc Moreno-Noguer. A joint model for 2d and 3d pose estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3634–3641, 2013.

[146] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015.

[147] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it smpl: Automatic estimation of 3d human pose and

shape from a single image. In *European Conference on Computer Vision*, pages 561–578. Springer, 2016.

[148] Shashank Tripathi, Siddhant Ranade, Ambrish Tyagi, and Amit Agrawal. Posenet3d: Unsupervised 3d human shape and pose estimation. *arXiv preprint arXiv:2003.03473*, 2020.

[149] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *2017 International Conference on 3D Vision (3DV)*, pages 506–516. IEEE, 2017.

[150] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017.

[151] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Monocular 3d pose estimation and tracking by detection. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 623–630. IEEE, 2010.

[152] Mir Rayat Imtiaz Hossain and James J Little. Exploiting temporal information for 3d pose estimation. *arXiv preprint arXiv:1711.08585*, 2017.

[153] Grégory Rogez, Philippe Weinzaepfel, and Cordelia Schmid. Lcr-net++: Multi-person 2d and 3d pose detection in natural images. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[154] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[155] Kyoungoh Lee, Inwoong Lee, and Sanghoon Lee. Propagating lstm: 3d pose estimation based on joint interdependency. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 119–135, 2018.

[156] Laxman Kumarapu and Prerana Mukherjee. Animepose: Multi-person 3d pose estimation and animation. *arXiv preprint arXiv:2002.02792*, 2020.

[157] Haokui Zhang, Chunhua Shen, Ying Li, Yuanzhouhan Cao, Yu Liu, and Youliang Yan. Exploiting temporal consistency for real-time video depth estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1725–1734, 2019.

[158] Mude Lin, Liang Lin, Xiaodan Liang, Keze Wang, and Hui Chen. Recurrent 3d pose sequence machines. In *CVPR*, 2017.

[159] Wenkang Shan, Haopeng Lu, Shanshe Wang, Xinfeng Zhang, and Wen Gao. Improving robustness and accuracy via relative information encoding in 3d human pose estimation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3446–3454, 2021.

[160] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7753–7762, 2019.

[161] Junfa Liu, Yisheng Guang, and Juan Rojas. Gast-net: Graph attention spatio-temporal convolutional networks for 3d human pose estimation in video. *arXiv preprint arXiv:2003.14179*, 2020.

[162] Yujun Cai, Liuhao Ge, Jun Liu, Jianfei Cai, Tat-Jen Cham, Junsong Yuan, and Nadia Magnenat Thalmann. Exploiting spatial-temporal relationships for 3d pose estimation via graph convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2272–2281, 2019.

[163] Hongsuk Choi, Gyeongsik Moon, and Kyoung Mu Lee. Pose2mesh: Graph convolutional network for 3d human pose and mesh recovery from a 2d human pose. In *European Conference on Computer Vision*, pages 769–787. Springer, 2020.

[164] Wenbo Hu, Changgong Zhang, Fangneng Zhan, Lei Zhang, and Tien-Tsin Wong. Conditional directed graph convolution for 3d human pose estimation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 602–611, 2021.

[165] Wenhao Li, Hong Liu, Runwei Ding, Mengyuan Liu, Pichao Wang, and Wenming Yang. Exploiting temporal contexts with strided transformer for 3d human pose estimation. *IEEE Transactions on Multimedia*, 2022.

[166] Hsiao-Yu Fish Tung, Adam W Harley, William Seto, and Katerina Fragkiadaki. Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4364–4372. IEEE, 2017.

[167] Bastian Wandt, Hanno Ackermann, and Bodo Rosenhahn. A kinematic chain space for monocular motion capture. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[168] Ehsan Jahangiri and Alan L Yuille. Generating multiple diverse hypotheses for human 3d pose consistent with 2d joint detections. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 805–814, 2017.

[169] Xingyi Zhou, Qixing Huang, Xiao Sun, Xiangyang Xue, and Yichen Wei. Weakly-supervised transfer for 3d human pose estimation in the wild. In *IEEE International Conference on Computer Vision, ICCV*, volume 3, page 7, 2017.

[170] Helge Rhodin, Jörg Spörri, Isinsu Katircioglu, Victor Constantin, Frédéric Meyer, Erich Müller, Mathieu Salzmann, and Pascal Fua. Learning monocular 3d human pose estimation from multi-view images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8437–8446, 2018.

[171] Xingyi Zhou, Arjun Karpur, Chuang Gan, Linjie Luo, and Qixing Huang. Unsupervised domain adaptation for 3d keypoint estimation via view consistency. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 137–153, 2018.

[172] Jogendra Nath Kundu, Siddharth Seth, Varun Jampani, Mugalodi Rakesh, R Venkatesh Babu, and Anirban Chakraborty. Self-supervised 3d human pose estimation via part guided novel image synthesis. *arXiv*, pages arXiv–2004, 2020.

[173] Muhammed Kocabas, Salih Karagoz, and Emre Akbas. Self-supervised learning of 3d human pose using multi-view geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1077–1086, 2019.

[174] Wen-Li Wei, Jen-Chun Lin, Tyng-Luh Liu, and Hong-Yuan Mark Liao. Capturing humans in motion: Temporal-attentive 3d human pose and shape estimation from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13211–13220, 2022.

[175] Jian Liu, Naveed Akhtar, and Ajmal Mian. Deep reconstruction of 3d human poses from video. *IEEE Transactions on Artificial Intelligence*, 2022.

[176] Jeongjun Choi, Dongseok Shim, and H Jin Kim. Diffupose: Monocular 3d human pose estimation via denoising diffusion probabilistic model. *arXiv preprint arXiv:2212.02796*, 2022.

[177] Rahul Mitra, Nitesh B Gundavarapu, Abhishek Sharma, and Arjun Jain. Multiview-consistent semi-supervised learning for 3d human pose estimation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 6907–6916, 2020.

[178] Yu Cheng, Bo Wang, Bo Yang, and Robby T Tan. Monocular 3d multi-person pose estimation by integrating top-down and bottom-up networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7649–7659, 2021.

[179] Yu Cheng, Bo Wang, and Robby Tan. Dual networks based 3d multi-person pose estimation from monocular video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[180] Guillaume Rochette, Chris Russell, and Richard Bowden. Weakly-supervised 3d pose estimation from a single image using multi-view consistency. *arXiv preprint arXiv:1909.06119*, 2019.

[181] Umar Iqbal, Pavlo Molchanov, and Jan Kautz. Weakly-supervised 3d human pose learning via multi-view images in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5243–5252, 2020.

[182] Bastian Wandt, Marco Rudolph, Petrissa Zell, Helge Rhodin, and Bodo Rosenhahn. Canonpose: Self-supervised monocular 3d human pose estimation in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13294–13304, 2021.

[183] Peishan Cong, Yiteng Xu, Yiming Ren, Juze Zhang, Lan Xu, Jingya Wang, Jingyi Yu, and Yuexin Ma. Weakly supervised 3d multi-person pose estimation for large-scale scenes based on monocular camera and single lidar. *arXiv preprint arXiv:2211.16951*, 2022.

[184] Cheng-Yen Yang, Jiajia Luo, Lu Xia, Yuyin Sun, Nan Qiao, Ke Zhang, Zhongyu Jiang, Jenq-Neng Hwang, and Cheng-Hao Kuo. Camerapose: Weakly-supervised monocular 3d human pose estimation by leveraging in-the-wild 2d annotations. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2924–2933, 2023.

[185] Dylan Drover, Rohith MV, Ching-Hang Chen, Amit Agrawal, Ambrish Tyagi, and Cong Phuoc Huynh. Can 3d pose be learned from 2d projections alone? In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[186] Ching-Hang Chen, Ambrish Tyagi, Amit Agrawal, Dylan Drover, Stefan Stojanov, and James M Rehg. Unsupervised 3d pose estimation with geometric self-supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5714–5724, 2019.

[187] Zhenbo Yu, Bingbing Ni, Jingwei Xu, Junjie Wang, Chenglong Zhao, and Wenjun Zhang. Towards alleviating the modeling ambiguity of unsupervised monocular 3d human pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8651–8660, 2021.

[188] Bastian Wandt, James J Little, and Helge Rhodin. Elepose: Unsupervised 3d human pose estimation by predicting camera elevation and learning normalizing flows on 2d poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6635–6645, 2022.

[189] Dejing Xu, Jun Xiao, Zhou Zhao, Jian Shao, Di Xie, and Yueting Zhuang. Self-supervised spatiotemporal learning via video clip order prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10334–10343, 2019.

[190] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Self-supervised learning of interpretable keypoints from unlabelled videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8787–8797, 2020.

[191] Kehong Gong, Bingbing Li, Jianfeng Zhang, Tao Wang, Jing Huang, Michael Bi Mi, Jiashi Feng, and Xinchao Wang. Posetriplet: co-evolving 3d human pose estimation, imitation, and hallucination under self-supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11017–11027, 2022.

[192] Wenkang Shan, Zhenhua Liu, Xinfeng Zhang, Shanshe Wang, Siwei Ma, and Wen Gao. P-stmo: Pre-trained spatial temporal many-to-one model for 3d human pose estimation. In *European Conference on Computer Vision*, pages 461–478. Springer, 2022.

[193] Sina Honari, Victor Constantin, Helge Rhodin, Mathieu Salzmann, and Pascal Fua. Temporal representation learning on monocular videos for 3d human pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[194] Jogendra Nath Kundu, Siddharth Seth, Pradyumna YM, Varun Jampani, Anirban Chakraborty, and R Venkatesh Babu. Uncertainty-aware adaptation for self-supervised 3d human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20448–20459, 2022.

[195] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt 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.

[196] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[197] Umar Iqbal, Anton Milan, and Juergen Gall. Posetrack: Joint multi-person pose estimation and tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2011–2020, 2017.

[198] Benjamin Sapp and Ben Taskar. Modec: Multimodal decomposable models for human pose estimation. In *In Proc. CVPR*, 2013.

[199] Umer Rafi, Bastian Leibe, Juergen Gall, and Ilya Kostrikov. An efficient convolutional network for human pose estimation. In *BMVC*, volume 1, page 2, 2016.

[200] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, jul 2014.

[201] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Srinath Sridhar, Gerard Pons-Moll, and Christian Theobalt. Single-shot multi-person 3d pose estimation from monocular rgb. In *2018 International Conference on 3D Vision (3DV)*, pages 120–130. IEEE, 2018.

[202] Márton Véges and A Lőrincz. Temporal smoothing for 3d human pose estimation and localization for occluded people. In *International Conference on Neural Information Processing*, pages 557–568. Springer, 2020.

[203] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[204] Shekofa Ghoury, Cemil Sungur, and Akif Durdu. Real-time diseases detection of grape and grape leaves using faster r-cnn and ssd mobilenet architectures. In *International conference on advanced technologies, computer engineering and science (ICATCES 2019)*, pages 39–44, 2019.

[205] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.

[206] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. pages 2117–2125, 2017.

[207] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[208] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[209] Pu Li and Wangda Zhao. Image fire detection algorithms based on convolutional neural networks. *Case Studies in Thermal Engineering*, 19:100625, 2020.

[210] Shiuh-Ku Weng, Chung-Ming Kuo, and Shu-Kang Tu. Video object tracking using adaptive kalman filter. *Journal of Visual Communication and Image Representation*, 17(6):1190–1208, 2006.

[211] Yvo Boers and Johannes N Driessen. Particle filter based detection for tracking. In *Proceedings of the 2001 American Control Conference.(Cat. No. 01CH37148)*, volume 6, pages 4393–4397. IEEE, 2001.

[212] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596, 2014.

[213] Martin Danelljan, Gustav Häger, Fahad Shahbaz Khan, and Michael Felsberg. Discriminative scale space tracking. *IEEE transactions on pattern analysis and machine intelligence*, 39(8):1561–1575, 2016.

[214] Kaihua Zhang, Lei Zhang, Qingshan Liu, David Zhang, and Ming-Hsuan Yang. Fast visual tracking via dense spatio-temporal context learning. In *European conference on computer vision*, pages 127–141. Springer, 2014.

[215] S. K. Singh and U Sharma. Simulink model for object tracking using optical flow. *International Journal of Science and Research (IJSR)*, 4(6):2323–2326, 2015.

[216] S Akshay. Single moving object detection and tracking using horn-schunck optical flow method. *International Journal of Applied Engineering Research*, 10(11):30135–30152, 2015.

[217] Zheng Zhu, Qiang Wang, Bo Li, Wei Wu, Junjie Yan, and Weiming Hu. Distractor-aware siamese networks for visual object tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 101–117, 2018.

[218] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6182–6191, 2019.

[219] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4293–4302, 2016.

[220] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[221] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4282–4291, 2019.

[222] František Galčík and Radoslav Gargalík. Real-time depth map based people counting. pages 330–341, 2013.

[223] Sam Johnson and Mark Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *Proceedings of the British Machine Vision Conference*, 2010. doi:10.5244/C.24.12.

[224] Saurabh Sharma, Pavan Teja Varigonda, Prashast Bindal, Abhishek Sharma, and Arjun Jain. Monocular 3d human pose estimation by generation and ordinal ranking. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2325–2334, 2019.

[225] Rishabh Dabral, Anurag Mundhada, Uday Kusupati, Safeer Afaque, Abhishek Sharma, and Arjun Jain. Learning 3d human pose from structure and motion. pages 668–683, 2018.

[226] Yu Cheng, Bo Yang, Bo Wang, Wending Yan, and Robby T Tan. Occlusion-aware networks for 3d human pose estimation in video. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 723–732, 2019.

[227] Huy Hieu Pham, Houssam Salmane, Louahdi Khoudour, Alain Crouzil, Sergio A Velastin, and Pablo Zegers. A unified deep framework for joint 3d pose estimation and action recognition from a single rgb camera. *Sensors*, 20(7):1825, 2020.

[228] Yu Cheng, Bo Yang, Bo Wang, and Robby T Tan. 3d human pose estimation using spatio-temporal networks with explicit occlusion training. *arXiv preprint arXiv:2004.11822*, 2020.

[229] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2252–2261, 2019.

[230] Haiping Wu and Bin Xiao. 3d human pose estimation via explicit compositional depth maps. In *AAAI*, pages 12378–12385, 2020.

[231] Tianlang Chen, Chen Fang, Xiaohui Shen, Yiheng Zhu, Zhili Chen, and Jiebo Luo. Anatomy-aware 3d human pose estimation in videos. *arXiv preprint arXiv:2002.10322*, 2020.

[232] Jiahao Lin and Gim Hee Lee. Trajectory space factorization for deep video-based 3d human pose estimation. *arXiv preprint arXiv:1908.08289*, 2019.

[233] Wenhao Li, Hong Liu, Runwei Ding, Mengyuan Liu, Pichao Wang, and Wenming Yang. Exploiting temporal contexts with strided transformer for 3d human pose estimation. *arXiv preprint arXiv:2103.14304*, 2021.

[234] PapersWithCode. 3D Human Pose Estimation on Human3.6M. https://paperswithcode.com/sota/3d-human-pose-estimation-on-human36m. [Online; accessed 19-January-2022].

[235] Bruce Xiaohan Nie, Ping Wei, and Song-Chun Zhu. Monocular 3d human pose estimation by predicting depth on joints. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3467–3475. IEEE, 2017.

[236] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7297–7306, 2018.

[237] Kun Zhou, Xiaoguang Han, Nianjuan Jiang, Kui Jia, and Jiangbo Lu. Hemlets pose: Learning part-centric heatmap triplets for accurate 3d human pose estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2344–2353, 2019.

[238] Jue Wang, Shaoli Huang, Xinchao Wang, and Dacheng Tao. Not all parts are created equal: 3d pose estimation by modeling bi-directional dependencies of body parts. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7771–7780, 2019.

[239] Hai Ci, Chunyu Wang, Xiaoxuan Ma, and Yizhou Wang. Optimizing network structure for 3d human pose estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2262–2271, 2019.

[240] Zhi Li, Xuan Wang, Fei Wang, and Peilin Jiang. On boosting single-frame 3d human pose estimation via monocular videos. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2192–2201, 2019.

[241] Xipeng Chen, Kwan-Yee Lin, Wentao Liu, Chen Qian, and Liang Lin. Weakly-supervised discovery of geometry-aware representation for 3d human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10895–10904, 2019.

[242] Xipeng Chen, Kwan-Yee Lin, Wentao Liu, Chen Qian, and Liang Lin. Weakly-supervised discovery of geometry-aware representation for 3d human pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10895–10904, 2019.

[243] Dominic Jack, Frederic Maire, Sareh Shirazi, and Anders Eriksson. Ige-net: Inverse graphics energy networks for human pose estimation and single-view reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7075–7084, 2019.

[244] Matthew Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. Total capture: 3d human pose estimation fusing video and inertial sensors. In *Proceedings of 28th British Machine Vision Conference*, pages 1–13, 2017.

[245] Ye Yuan, Shih-En Wei, Tomas Simon, Kris Kitani, and Jason Saragih. Simpoe: Simulated character control for 3d human pose estimation. In *Proceedings of the*

IEEE/CVF conference on computer vision and pattern recognition, pages 7159–7169, 2021.

[246] Mir Rayat Imtiaz Hossain and James J Little. Exploiting temporal information for 3d human pose estimation. pages 68–84, 2018.

[247] Ruixu Liu, Ju Shen, He Wang, Chen Chen, Sen-ching Cheung, and Vijayan Asari. Attention mechanism exploits temporal contexts: Real-time 3d human pose reconstruction. pages 5064–5073, 2020.

[248] Zhongwei Qiu, Qiansheng Yang, Jian Wang, Haocheng Feng, Junyu Han, Errui Ding, Chang Xu, Dongmei Fu, and Jingdong Wang. Psvt: End-to-end multi-person 3d pose and shape estimation with progressive video transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21254–21263, 2023.

[249] Thiemo Alldieck, Marc Kassubeck, Bastian Wandt, Bodo Rosenhahn, and Marcus Magnor. Optical flow-based 3d human motion estimation from monocular video. In *Pattern Recognition: 39th German Conference, GCPR 2017, Basel, Switzerland, September 12–15, 2017, Proceedings 39*, pages 347–360. Springer, 2017.

[250] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. Humor: 3d human motion model for robust pose estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11488–11499, 2021.

[251] Kevin Xie, Tingwu Wang, Umar Iqbal, Yunrong Guo, Sanja Fidler, and Florian Shkurti. Physics-based human motion estimation and synthesis from videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11532–11541, 2021.

[252] Colin Lea, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks: A unified approach to action segmentation. pages 47–54, 2016.

[253] Junfa Liu, Juan Rojas, Yihui Li, Zhijun Liang, Yisheng Guan, Ning Xi, and Haifei Zhu. A graph attention spatio-temporal convolutional network for 3d human pose estimation in video. pages 3374–3380, 2021.

[254] Tianlang Chen, Chen Fang, Xiaohui Shen, Yiheng Zhu, Zhili Chen, and Jiebo Luo. Anatomy-aware 3d human pose estimation with bone-based pose decomposition. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[255] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. pages 5253–5263, 2020.

[256] Wentao Zhu, Xiaoxuan Ma, Zhaoyang Liu, Libin Liu, Wayne Wu, and Yizhou Wang. Motionbert: Unified pretraining for human motion analysis. *arXiv preprint arXiv:2210.06551*, 2022.

[257] Jinlu Zhang, Zhigang Tu, Jianyu Yang, Yujin Chen, and Junsong Yuan. Mixste: Seq2seq mixed spatio-temporal encoder for 3d human pose estimation in video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13232–13242, 2022.

[258] Andrei Zanfir, Elisabeta Marinoiu, and Cristian Sminchisescu. Monocular 3d pose and shape estimation of multiple people in natural scenes-the importance of multiple scene constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2148–2157, 2018.

[259] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. Camera distance-aware top-down approach for 3d multi-person pose estimation from a single rgb image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10133–10142, 2019.

[260] Abdallah Benzine, Florian Chabot, Bertrand Luvison, Quoc Cuong Pham, and Catherine Achard. Pandanet: Anchor-based single-shot multi-person 3d pose estimation. pages 6856–6865, 2020.

[261] Jiahao Lin and Gim Hee Lee. Hdnet: Human depth estimation for multi-person camera-space localization. In *European Conference on Computer Vision*, pages 633–648. Springer, 2020.

[262] Jiefeng Li, Can Wang, Wentao Liu, Chen Qian, and Cewu Lu. Hmor: Hierarchical multi-person ordinal relations for monocular multi-person 3d pose estimation. *arXiv preprint arXiv:2008.00206*, 2020.

[263] Yu Cheng, Bo Wang, Bo Yang, and Robby T Tan. Graph and temporal convolutional networks for 3d multi-person pose estimation in monocular videos. 4(7):12, 2021.

[264] Matteo Fabbri, Fabio Lanzi, Simone Calderara, Stefano Alletto, and Rita Cucchiara. Compressed volumetric heatmaps for multi-person 3d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7204–7213, 2020.

[265] Changgong Zhang, Fangneng Zhan, and Yuan Chang. Deep monocular 3d human pose estimation via cascaded dimension-lifting. *arXiv preprint arXiv:2104.03520*, 2021.

[266] Andrei Zanfir, Elisabeta Marinoiu, Mihai Zanfir, Alin-Ionut Popa, and Cristian Sminchisescu. Deep network for the integrated 3d sensing of multiple people in natural images. In *Advances in Neural Information Processing Systems*, pages 8410–8419, 2018.

[267] Jianan Zhen, Qi Fang, Jiaming Sun, Wentao Liu, Wei Jiang, Hujun Bao, and Xiaowei Zhou. Smap: Single-shot multi-person absolute 3d pose estimation. In *European Conference on Computer Vision*, pages 550–566. Springer, 2020.

[268] Zitian Wang, Xuecheng Nie, Xiaochao Qu, Yunpeng Chen, and Si Liu. Distribution-aware single-stage models for multi-person 3d pose estimation. *arXiv preprint arXiv:2203.07697*, 2022.

[269] Yawen Lu, Sophia Kourian, Carl Salvaggio, Chenliang Xu, and Guoyu Lu. Single image 3d vehicle pose estimation for augmented reality. In *2019 IEEE Global*

Conference on Signal and Information Processing (GlobalSIP), pages 1–5. IEEE, 2019.

[270] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651, 2015.

[271] Megha Kalia, Nassir Navab, and Tim Salcudean. A real-time interactive augmented reality depth estimation technique for surgical robotics. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8291–8297. IEEE, 2019.

[272] Najib Metni, Tarek Hamel, and François Derkx. Visual tracking control of aerial robotic systems with adaptive depth estimation. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 6078–6084. IEEE, 2005.

[273] Jaehyun Im, Jaehoon Jung, and Joonki Paik. Single camera-based depth estimation and improved continuously adaptive mean shift algorithm for tracking occluded objects. In *Pacific Rim Conference on Multimedia*, pages 246–252. Springer, 2015.

[274] Tristan Laidlow, Jan Czarnowski, and Stefan Leutenegger. Deepfusion: Real-time dense 3d reconstruction for monocular slam using single-view depth and gradient predictions. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4068–4074. IEEE, 2019.

[275] Shahnewaz Ali and Ajay K Pandey. Arthronet: monocular depth estimation technique toward 3d segmented maps for knee arthroscopic. *Intelligent Medicine*, 2022.

[276] Ashutosh Saxena, Jamie Schulte, Andrew Y Ng, et al. Depth estimation using monocular and stereo cues. In *IJCAI*, volume 7, pages 2197–2203, 2007.

[277] Yue Ming, Xuyang Meng, Chunxiao Fan, and Hui Yu. Deep learning for monocular depth estimation: A review. *Neurocomputing*, 438:14–33, 2021.

[278] Amal El Kaid, Denis Brazey, Vincent Barra, and Karim Baïna. Top-down system for multi-person 3d absolute pose estimation from monocular videos. *Sensors*, 22(11):4109, 2022.

[279] Laurent Breillat. La distance / longueur focale d'un objectif, 2022.

[280] Márton Véges and András Lőrincz. Absolute human pose estimation with depth prediction network. pages 1–7, 2019.

[281] Chunlei Chen, Peng Zhang, Huixiang Zhang, Jiangyan Dai, Yugen Yi, Huihui Zhang, and Yonghui Zhang. Deep learning on computational-resource-limited platforms: a survey. *Mobile Information Systems*, 2020, 2020.

[282] Wilfried Haensch, Tayfun Gokmen, and Ruchir Puri. The next generation of deep learning hardware: Analog computing. *Proceedings of the IEEE*, 107(1):108–122, 2018.

[283] NVIDIA DEVELOPER. Cuda toolkit, 2022.

[284] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[285] PyTorch Developers Team. Pytorch.

[286] New Zealand eScience Infrastructure. Vectorising.

[287] take cheeze. Onnx, 2022.

[288] OpenCV Developers Team. Opencv.

[289] Isabel Bär Alexander Kniesz Larysa Visengeriyeva, Anja Kammer and Michael Plöd. Mlops principles, 2022.

[290] NVIDIA DEVELOPER. Deploying deep neural networks with nvidia tensorrt, 2022.

List of Figures

1.1 Thesis structure 25

2.1 Typical Deep Neural Network architecture 33
2.2 Comparison between semantic segmentation, classification, object detection and instance segmentation 36
2.3 Regression Analysis Using Artificial Neural Networks 36
2.4 Convolutional Neural Network Architecture for Classification . . 37
2.5 Recurrent Neural Network Architecture [5] 38

3.1 An example of Augmented reality use case [15] 50
3.2 An example of autonomous driving application using pose estimation [20] 52
3.3 Examples in the sport analysis and fitness industry using pose estimation [21]. 53
3.4 Commonly used human body models. (a) skeleton-based model; (b) contour-based models; (c) volume-based models. [22] 56
3.5 Generative Approach to Monocular 3D Human Pose Estimation [32]. 57
3.6 Pictorial structures model by Fischler and Elschlager. Objects (as, e.g., faces) are modeled as a set of parts that appear in some typical spatial relationship with some flexibility concerning the relative locations [33]. 58

3.7 Discriminative Approach to Monocular 3D Human Pose Estimation [32]. 59
3.8 A taxonomy of 2D Human Pose Estimation approaches. 61
3.9 A taxonomy of 3D Human Pose Estimation approaches. 71

4.1 Faster R-CNN architecture [204]. 101
4.2 SSD architecture [204]. 102
4.3 YOLO-v3 architecture [209]. 104
4.4 HRNet architecture [86]. 111
4.5 GAST-Net architecture [253]. 116

5.1 Network architecture of the RootNet [259]. 129
5.2 Summary Diagram of the Networks Used in the System. 131
5.3 Geometric Method for Determining the Absolute Position of a Person's Keypoint 139
5.4 From Camera coordinates to Image plane 141
5.5 Using Trigonometry to Determine Camera Angle and Measure Absolute Distance 143
5.6 Representation of Rotation Angle to obtain the orthogonal vector . 143

6.1 Tracking Result: The ID assigned to the person across the frames is 2. 152
6.2 Comparison of keypoint formats. The first skeleton, marked with numbers in green, represents the MS-COCO format. The second skeleton illustrates the h36m format (in blue) and the MuPoTS-3D format (in red). The primary difference between h36m and MuPoTS-3D is the number of keypoints. Notably, the MS-COCO format includes additional keypoints for eyes and ears, distinguishing it from the other two formats 153

6.3 3D Vector Transformation with Rotation about the X-Axis by Angle α. 160
6.4 Human detection + tracking + 2D pose estimation 161
6.5 The pipeline of the proposed framework (Root-GAST-Net) 163
6.6 Fall detection 165
6.7 Erroneous 3D multi-person pose estimation. The first two images represent two similar poses of different people because one is completely occluded. In the right two images, one pose is incorrect because the body parts are partially outside the boxes. 171

7.1 Processing chain of the compressed video stream 179
7.2 Inception structure in GoogleNet 198
7.3 Horizontal Fusion 198
7.4 Vertical Fusion 198
7.5 Fusion network layers by TensorRT 198
7.6 MLOps process. [290]. 210

List of Tables

2.1 Applications of Deep Learning in various industries 29

2.2 Learning Methods 41

3.1 Comparison of Bottom-up and Top-down Approaches on MS-COCO test-dev 69

3.2 Advantages, Disadvantages, Suitable Applications, and Examples for Each Learning Paradigm 86

3.3 Common Databases for 2D Human Pose Estimation 88

3.4 Characteristics of 2D Human Pose Estimation Evaluation Metrics 89

3.5 Common databases for 3D human pose estimation 91

3.6 Evaluation metrics. Note that T denotes the total number of test samples and N denotes the number of joints. Ground-truth joint and the predicted joint are indicated by J and $\hat{J}$, respectively. i represents each joint from all joints and *root* represents the root-joint. 93

4.1 Monocular Image-Based Methods 113

4.2 Monocular Video-Based Methods 115

4.3 Summary of methods and their accuracy in terms of MPJPE on Human3.6m and number of frames used. 118

5.1 Camera-centric evaluations on the MuPoTS-3D dataset. The best is in bold, the second best is underlined. 133

5.2 Person-centric and camera-centric evaluations on the MuPoTS-3D dataset. The best is in bold, the second best is underlined. 133
5.3 Evaluating the Performance of Root-Relative keypoints, absolute root and Absolute Keypoints on the MuPoTS-3D Database. The best is in bold. 136
5.4 Evaluating the Performance of Root-Relative keypoints, absolute root and Absolute Keypoints on the MuPoTS-3D Database. The best is in bold. 138

6.1 Sequence-wise 3D-PCK$_{abs}$ comparison with the state-of-the-art on the MuPoTS-3D dataset. (*) The accuracies of methods are measured on matched ground truths. The best is in bold, the second best is underlined. 166
6.2 Average precision of the root keypoint evaluation by different distances on the MuPoTS-3D dataset. 167
6.3 MPJPE of relative poses on MuPoTS-3D dataset. Best in bold, second best underlined. 168
6.4 MRPE results comparison with RootNet [259] on the Human3.6M dataset. MRPE$_x$, MRPE$_y$, and MRPE$_z$ are the average MRPE errors in the x, y, and z axes, respectively. 168
6.5 Response time per model. 169
6.6 Frame rate per approach. 170

7.1 Common GPU models and their approximate TFLOPs performance for different types of floating point operations and the energy power 187
7.2 table comparing W&B, Neptune, DVC and Pachyderm 211