MÁSTER UNIVERSITARIO
EN VISIÓN ARTIFICIAL

TRABAJO FIN DE MÁSTER

Real-time 3D Human Pose Estimation for Robotics Applications

Autor: David Pascual Hernández

Tutor: José María Cañas Plaza

Co-tutora: Inmaculada Mora Jiménez

Curso académico 2019/2020

RoboCity2030 - Madrid Robotics Digital Innovation Hub
Programa de Actividades de I+D entre Grupos de investigación de la Comunidad de Madrid en Tecnologías 2018, Comunidad de Madrid.
Ref S2018/NMT-4331
Acrónimo: RoboCity2030-DIH-CM

This work has been partly supported by the Spanish Ministry of Economy under the grants with reference TEC2016-75361-R and PID2019-106623RB-C41.

Resumen

En los últimos años se ha producido un proceso de digitalización y aumento de la capacidad de cómputo sin precedentes. Todos los ámbitos de la sociedad se han visto afectados por la llamada *4ª Revolución Industrial*, especialmente aquellas áreas de conocimiento más transversales como la robótica, cuyas aplicaciones son embebidas habitualmente en dispositivos con recursos limitados y deben funcionar en tiempo real.

Uno de los campos de estudio más relevantes dentro de la róbotica es la *Interacción Humano-Robot*, presente siempre que el robot requiera tener en cuenta a las personas de su entorno para planificar sus acciones. En consonancia con nuestras formas de comunicación, esta tarea suele depender del procesamiento de señales de audio o vídeo, aunque esta última tiene la ventaja de no requerir interacción explícita por parte de dichas personas.

Nuestro trabajo se centra en la estimación de pose en humanos, un área de la visión artificial estrechamente relacionada con la *Interacción Humano-Robot*. Históricamente resuelta mediante complejos sistemas de captura de movimiento, en las últimas décadas se ha dedicado un gran esfuerzo al desarrollo de algoritmos sobre dispositivos más escalables. La inclusión de técnicas de aprendizaje profundo ha catapultado la efectividad de estos últimos. Sin embargo, la cantidad de ejemplos etiquetados y recursos computacionales requeridos puede ser prohibitiva, en especial cuando se trabaja con datos tridimensionales, que proporcionan al robot una información más completa sobre su entorno.

Teniendo estos retos en cuenta, presentamos un sistema completo para la estimación 3D de pose en humanos. El método propuesto funciona en tiempo real haciendo uso de un ordenador y un sensor RGBD comerciales. En este sentido, nuestras principales contribuciones son la evaluación cuantitativa de diferentes estimadores de pose 2D, el diseño y desarrollo del sistema propuesto, la evaluación cuantitativa de dicho sistema, incluyendo comparativas con el estado del arte, y el desarrollo de un demostrador en tiempo real que valida expirementalmente su rendimiento a nivel cualitativo.

Los resultados muestran que nuestra solución es competitiva en términos de precisión y suficientemente ligera para funcionar en tiempo real, como queda demostrado a través de la evaluación cuantitativa y cualitativa. Concluimos, por tanto, que su uso es apropiado en aplicaciones robóticas que requieren interacción con humanos.

Summary

In recent years, all areas of society have been affected by an unprecedented process of data digitization and a huge increment in computational capacity. Cross-sectional areas of knowledge are the ones that have benefited the most from this so-called *Fourth Industrial Revolution*. This is specially true for robotics, as most of its applications must be embedded in devices with limited resources and must work in real-time.

One of the most relevant fields of study within this area is *Human-Robot Interaction*, as it is a core component for systems in which the robot needs to be aware of the humans around for planning its actions. Owing to the nature of our own forms of communication, audio and video signals are usually employed to perform such task, but the latter has the advantage of giving the robot relevant information without explicit interaction.

Our work focuses on human pose estimation, a classic computer vision problem closely related with *Human-Robot Interaction*. This task has been historically addressed by means of cumbersome motion capture systems, but huge efforts have been made in the last decades in order to develop algorithms that can solve it making use of *off-the-shelf* devices. The inclusion of deep learning techniques has rocketed the effectiveness of such methods in just a few years. However, the amount of labeled data and computational resources required might be prohibitive, specially when dealing with three-dimensional data, which provides the robot with more extensive information about its surroundings.

Taking all of these challenges into account, we propose an *end-to-end* pipeline for estimating 3D human poses that works in real-time in an *off-the-shelf* computer with a commercial RGBD sensor. In that regard, our main contributions are the performance of a quantitative evaluation of 2D pose estimators in the literature, the design and development of the proposed system, a quantitative evaluation and comparison of this system against a *state-of-the art* solution for 3D pose estimation and a real-time demo that experimentally validates its performance in qualitative terms.

The obtained results reveal that our solution is very competitive in terms of accuracy, while being lightweight enough to run in real-time, as we demonstrate in both our quantitative and qualitative evaluations. Therefore, our system is suitable for being integrated in robotics applications in which the robot is required to interact with humans.

Contents

List of Figures

List of Tables

Acronyms

BMHAD Berkeley's Multimodal Human Action Database. 5, 31–33, 35, 38, 39, 41, 42, 47, 56, 58

CNN Convolutional Neural Network. 4, 11–16, 20–23

CP Chained Prediction. 23, 24, 35–37, 43, 45–49

CPM Convolutional Pose Machine. 13, 20–22, 24, 35–40, 42, 43, 45–50, 61

DL Deep Learning. 3, 4, 9, 11–13, 15, 17, 19, 20, 62

fps frames per second. 29, 49

GPU Graphics Processing Unit. 4

HRI Human-Robot Interaction. 2

ML Machine Learning. 3, 4

MoCap Motion Capture. 3, 31–33, 38

MPJPE Mean Per Joint Position Error. 39, 41–45, 58

NN Neural Network. 4

PCKh Percentage of Correct Keypoints. 35, 36, 38–45, 58

SDK Software Development Kit. 5, 24, 27, 33

SH Stacked Hourglass. 22, 24, 35–37, 43, 45–49

SOTA state-of-the-art. 5, 6, 17, 31, 47, 49, 55, 57–59, 62

Chapter 1

Introduction

This project does not stand alone. It has been built upon decades of research and has been strongly influenced by several external factors. In this chapter, we will present the social framework in which it has been developed and then particularize why it might be relevant for robotics and computer vision communities. After contextualizing our work, we will set the goals pursued during its development and describe the methodology followed to achieve them. For ease of reading, we will conclude this chapter with a brief overview of this report structure.

1.1 Context and motivation

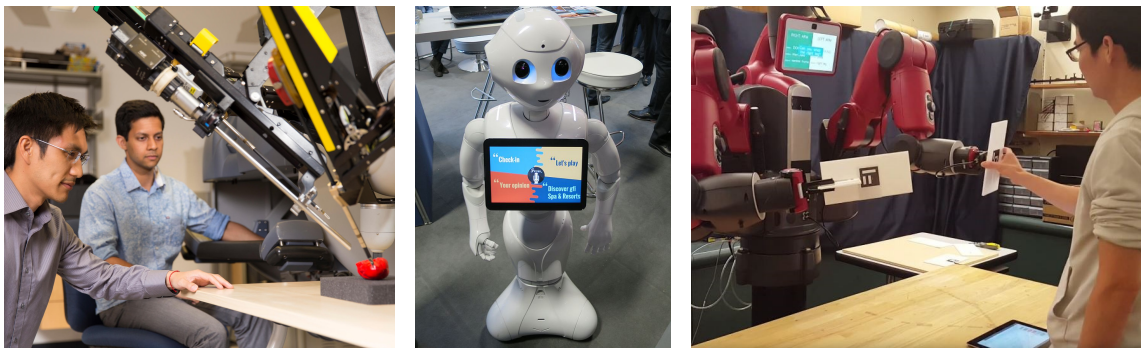
In recent years, we have assisted to an unprecedented process of digitization of data from the most diverse sources, which has been coupled with a very significant increment in computational capacity. We are immersed in what has come to be called the *Fourth Industrial Revolution* [1]. This revolution has reached, to some extent, all areas of society, *e.g.* health services, sports, education, finance, marketing, security or transport. In that sense, cross-sectional fields such as robotics have been immensely affected by these changes.

If we think about the nature of most robotics applications, they usually have strong requirements in terms of how fast the robot must take action given streams of data that must be integrated and interpreted. On top of that, a lot of these applications demand the robot to be portable and lightweight (*e.g.* drones or vacuuming robots). Taking these considerations into account, it is clear how heavily robotics rely on the development of very agile algorithms that can be embedded in computers with limited resources, even

though these resources are not as limited as they once were.

One of the subfields in robotics that has received most attention lately is Human-Robot Interaction (HRI). As a core component of robotics applications in areas such as assistive robotics, search and rescue or home automation, HRI is attracting interest from both academy and industry [2]. It is in itself a very interdisciplinary field of study, where researchers with expertise in multiple areas of engineering, natural language or psychology have come together to better understand the processes involved in the interaction between humans and robots in order to design increasingly intelligent systems.

Communication between humans and robots may appear in many forms. In Figure 1.1, examples of different forms of interaction are shown. In one side of the spectrum, it can mimic communication between humans, as it can be seen in the case of humanoid robots built for companionship. In the opposite side, it can be a very direct one-way communication. Such is the case of robot teleoperation, where the human acts as supervisor or operator. In between these, we can find applications in which the human teaches a robot to perform a task or vice versa; or those in which humans and robots work together as peers in industrial environments (*cobots*). In any of these cases, robots must present some kind of human-aware behaviors. In order to gain this kind of awareness, robotic systems rely on the streams of data provided by their sensors, being the most common audio and video signals, which are used for enabling verbal and non-verbal communication, respectively.



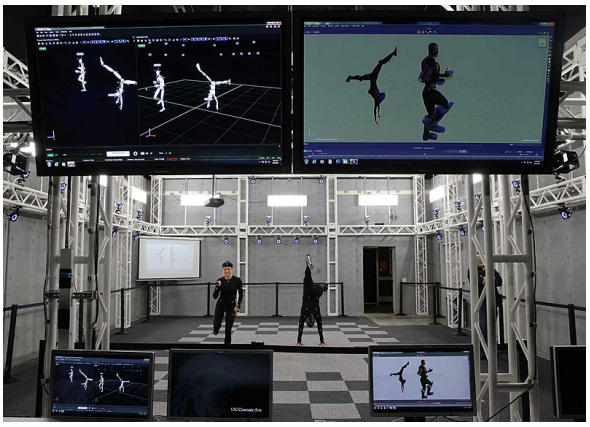
(a) ARCLab surgical robot [3]. (b) Pepper [4]. (c) Baxter [5].

Figure 1.1: Human-robot interaction can take many forms. From left to right, we present examples of (a) teleoperation, (b) companionship and (c) collaboration.

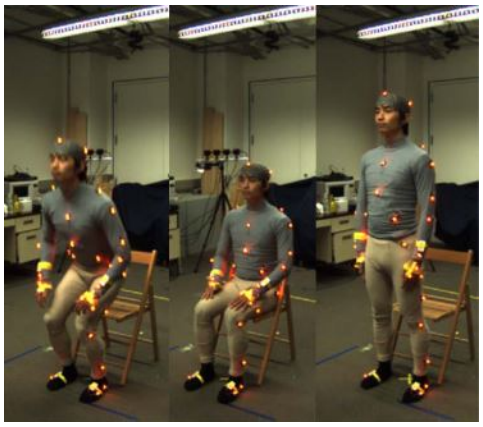
From video signals, useful information can be extracted by analyzing the subject and its environment. These kinds of analysis are dependent on subfields of computer vision, such

as human pose estimation, human detection and gesture recognition [6]. In particular, human pose estimation is a very helpful capability for human-aware robots, since it can serve as an input for solving higher level tasks. For instance, a robot could watch a human peer to avoid collisions or could detect if the human has fainted and alert the emergency services. Broadly speaking, a precise identification of human pose provides a better understanding of the scene, which is a major requirement for any human-robot interface.

Motion Capture (MoCap) systems are a well known solution for human pose estimation [7]. They provide very precise 3D coordinates and are used in both commercial and academic environments, as shown in Figure 1.2. As a major drawback, they usually rely on markers placed on the humans to be tracked and multiple cameras or sensors around the scenario. Typically, these cameras are close to the ceiling in order to have a high point of view and cover the area to monitor [8]. The larger the area to monitor, the greater the number of needed cameras. MoCap systems are therefore very cumbersome and involve high installation costs, which prevents their usage as part of robotics applications, as the already mentioned assistive or industrial robots. In general, any robot that needs to be portable or distributed massively in commercial applications, requires more lightweight solutions in terms of both hardware and software.



(a) *OptiTrack* commercial MoCap system [9].



(b) MoCap system in *UC Berkeley* [10].

Figure 1.2: MoCap systems are reliable solutions for both (a) commercial and (b) academic applications.

Coming back to the so-called *Fourth Industrial Revolution*, one of the fields of study that has benefited the most from the increase in data availability and computational capacity is Machine Learning (ML), and more specifically Deep Learning (DL). As

stated by Andrew Ng [11], scale drives ML progress, in the sense that as computational capabilities scale, so does the amount of data that can be used to improve the performance of the algorithms developed. In other words, as the amount of data available increases, so does the computational power needed to maximize the performance of ML algorithms. That is why, even if the core principles of DL solutions, such as Neural Networks (NNs), were developed a long time ago, the field did not start to get its well deserved attention until the requirements of data and computational capacity where met in the last decade. These advances, coupled with major breakthroughs in training algorithms, have positioned DL as the best approach for solving tasks which are highly dependent on robustness when dealing with real-world data. Such is the case of fraud detection, speech and face recognition or autonomous driving, just to mention a few ones [12].

Computer vision is one of the areas for which DL-based solutions have entailed a greater improvement in terms of performance. Particularly, Convolutional Neural Networks (CNNs) have unseated classic computer vision algorithms as the best solution for solving some of the hardest problems in the field, including human pose estimation. Unlike classic NNs, CNNs take advantage of the spatial relationship between variables that are close to each other, allowing for translation invariance. In that way, they are specially suitable for processing *grid-like* data such as pixels in an image or time-steps in an audio signal. Besides their effectiveness, they can be really fast during inference, even though training can take a long time. Given that a Graphics Processing Unit (GPU) is available for acceleration, DL algorithms can be embedded into a robot and work in real-time. A very illustrative case of success in the usage of CNNs for performing human pose estimation is *OpenPose* [13], an open-source real-time system for multi-person 2D pose estimation, which is often used as a base component for building higher level applications. An example of *OpenPose* impressive performance *in the wild* is shown in Figure 1.3.

While the irruption of DL has had an undeniably beneficial impact in the development of the field of human pose estimation, the needed amount of real-world data for designing models achieving a good performance is still an issue, as capture and annotation processes are costly and very time consuming. This is specially true for 3D images, where motion capture systems are usually needed for collecting reliable annotations [14, 15]. As a consequence, there is a strong gap in the number of publicly available large-scale datasets in favor of the two-dimensional ones. While 2D estimations might be enough for solving some particular tasks like action recognition [16], 3D estimations are needed in order to

Figure 1.3: Multi-person 2D poses estimated by *OpenPose* system in real-time [13].

achieve a complete visual understanding of the scene [17]. Fortunately, the emergence of robust RGBD sensors at affordable prices, such as *Microsoft Kinect* [18] or *Asus Xtion* [19], allows the integration of real 3D data into human pose estimation pipelines. Along with these sensors, companies usually provide easy to use Software Development Kits (SDKs) and drivers, what favors their inclusion as part of the equipment of robots in the industry.

1.2 Goals

In this work, we will design and develop an *end-to-end* pipeline for augmenting 2D human pose estimations into 3D estimations. This perceptive algorithm is intended to be run on-board in robotics systems, so it must fulfill the following requirements:

- It must work in real-time in an *off-the-shelf* computer.
- It must work with a single regular RGBD camera, which is a commonly used sensor in robotics.
- It must work independently of the camera point of view, as the robot might move around the scene.

We will validate its usability and perform a wide and detailed comparison with some of the most relevant human pose estimation algorithms in the state-of-the-art (SOTA). These comparisons will be carried out using Berkeley's Multimodal Human Action Database (BMHAD) as benchmark, which is a publicly available and well-known international dataset [10]. Therefore, the main goals of this project can be summarized as follows:

1. Quantitative evaluation and comparison of multiple SOTA methods for 2D human pose estimation, which will be included as the base of our 3D estimation pipeline.

2. Design and development of a straight-forward method for augmenting 2D estimations into 3D estimations in real-time using RGBD sensors.

3. Quantitative evaluation and comparison of our 3D human pose estimation method and a SOTA algorithm.

4. Qualitative evaluation of the performance of our proposed pipeline with a real-time demo running in an *off-the-shelf* laptop with a single regular RGBD sensor.

1.3 Methodology

The development of this project has been weekly followed by its supervisors. During each meeting, results from the previous week were discussed and goals for the next one were established. In that way, we have been able to rapidly adjust to unexpected issues along the way. Furthermore, the frequent open discussions during these meetings have enabled a deeper understanding of the studied problem.

Besides the weekly meetings, the following external tools have been used to keep track of the progress of this project:

- **Blog.** A blog hosted in the *Robotics Lab URJC* platform [1] has been maintained as a logbook of each week advances.

- ***GitHub.*** We have used *Git* [2] for our software version control. Enforcing *open-source* practices, our main repository is hosted in *GitHub* and is publicly available [3]. It contains not only the code developed to meet the goals defined in the previous section, but also this report and the aforementioned blog.

Regarding software development, we have loosely followed an iterative model [20]. Instead of establishing a full list of specifications to reach a final version of the software, a small list of basic requirements has been set for each new iteration. After each of these, a functional version of the project has been presented and new requirements have been

[1] `https://roboticslaburjc.github.io/2017-tfm-david-pascual/`

[2] `https://git-scm.com/`

[3] `https://github.com/RoboticsLabURJC/2017-tfm-david-pascual`

established. This iterative process forces the developer to divide the pending work in smaller, more manageable chunks. It also increases flexibility, as it allows to reduce or extend the scope of each iteration as needed. This kind of methodology may become messy if the software is developed by larger teams, but is very well suited for small ones.

1.4 Project structure

For ease of reading, the structure of the report and a brief description of the contents of each chapter is presented here.

Chapter 1. Introduction. In this chapter, we describe the context in which this project has been developed. Furthermore, our main goals are defined, along with the methodology followed to achieve them.

Chapter 2. Related work. Relevant human pose estimation algorithms, from classic approaches to the most recent ones, are described and categorized depending on the data used as input and the nature of their estimations.

Chapter 3. Proposed method. Our proposed algorithm takes as input 2D pose estimations and augments them to 3D using RGBD sensors. In this chapter, we give an overview of the tested 2D pose estimation methods and describe in detail the steps of our augmentation pipeline.

Chapter 4. Experiments. A wide quantitative comparison of 2D and 3D estimation methods is performed, both in terms of accuracy and computational burden. Besides that, we present a demo showcasing the proposed pipeline.

Chapter 5. Conclusions and future lines. Finally, we look back into the work we have performed and discuss to which extent the proposed goals have been accomplished, highlighting the pros and cons of our system and giving an outline of potential future work.

Chapter 2

Related work

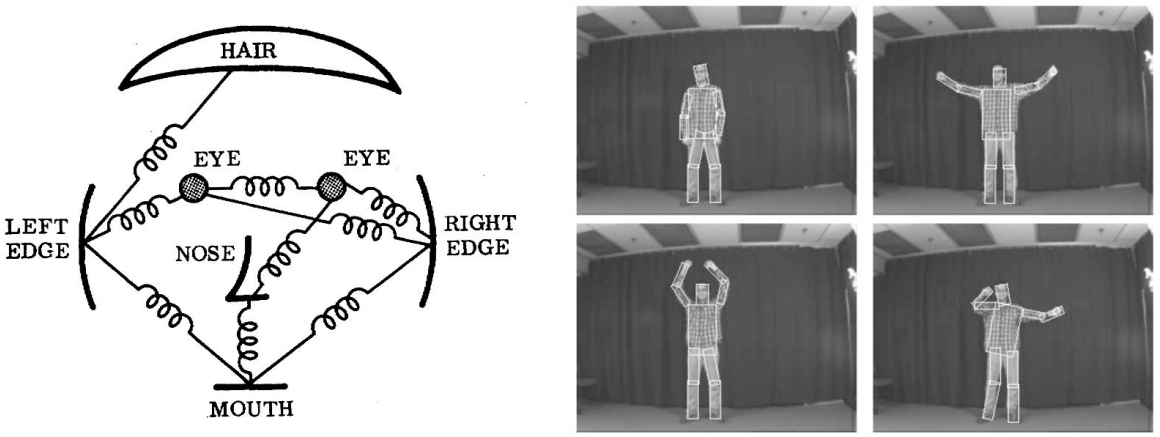
From a computer vision perspective, the human body can be considered as a collection of rigid parts or limbs connected by a number of joints [21]. Following this approach, the goal of human pose estimation methods is to determine the two-dimensional or three-dimensional location of these joints and parts within an image or a sequence of images. In this chapter, we explore previous work in the field of human pose estimation. We start presenting solutions for 2D pose estimation, categorized as classic approaches and DL-based solutions. The former rely on optimizations of explicit human body models to infer poses, while the latter are based on DL models with different architectures and training techniques. Then, 3D pose estimation methods are presented and categorized depending on the nature of the input data, *i.e.*, RGB or RGBD images.

2.1 2D Estimation

2.1.1 Classic methods

Conventional human pose estimation methods portray the human body as an articulated object. In that sense, most of the early works in the field present part-based models as a solution, not only for estimating human poses, but as tools for modeling any kind of object that can be interpreted as a collection of connected keypoints, *e.g.* hands or faces. Such is the case of the novel pictorial structures model proposed by Fischler and Elschlager [22]. In their work, they model objects as a collection of parts arranged in a deformable configuration, in which pairs of parts are linked by spring-like connections. The model is optimized by minimizing an energy function that measures a match cost

for each part and a deformation cost for each of these spring-like links between pairs. In Figure 2.1(a), we can see the intuitiveness of pictorial structures, exemplified in a diagram that models the human face.



(a) Human face as a pictorial structure [22]. (b) Pictorial structures application [23].

Figure 2.1: In (a), a human face is modeled as a pictorial structure, as originally introduced by Fischler and Elschlager [22]. In (b), results of applying pictorial structures to the human pose estimation problem, as revisited by Felzenszwalb and Huttenlocher [23].

To the best of our knowledge, Felzenszwalb and Huttenlocher [23] introduced the application of pictorial structures to the recognition of human body poses. Figure 2.1(b) presents some of their results. The main improvements with respect to the original work are the consideration of models with acyclic connections, which better matches the appearance of human bodies, and the inclusion of a supervised learning algorithm for the optimization stage. Their model also allows the detection of multiple instances of the objects recognized in the images. Other examples of successful extensions of the pictorial structures model for human pose estimation can be found in the works of Andriluka *et al.* [24] and Yang and Ramanan [25]. The latter propose a mixture of non-oriented pictorial structures, which further improves the representation of the human body. They use *Histograms of Oriented Gradients* as features and fit the model parameters using *Support Vector Machines*. Their method outperformed previous approaches dramatically in terms of both accuracy and efficiency.

The list of works that propose new appearance models or extensions of the ones mentioned above goes on and on [26, 27, 28, 29]. It is worth mentioning the adaptation presented by Eichner and Ferrari, which is able of dealing with multiple subjects in the

images. Their model takes into account the spatial relation between multiple human poses and how they occlude each other. Another active field of study is the inclusion of spatio-temporal cues for estimation in video sequences [30, 31, 32, 33].

2.1.2 Deep learning based methods

More holistic approaches started to gain popularity with the inclusion of DL techniques. In [34], Toshev and Szegedy explore the application of CNNs to estimate the body joints locations considering a cascade of *Deep Neural Networks* which refines a coarse initial estimation. In [35], Tompson *et al.* propose a joint training of a hybrid architecture composed of a CNN (as the part detector) with a part-based spatial model inspired on *Markov Random Fields*, significantly outperforming previous methods. Figure 2.2 shows how this method is capable of dealing with very complex poses and self-occlusions in images taken *in the wild*.

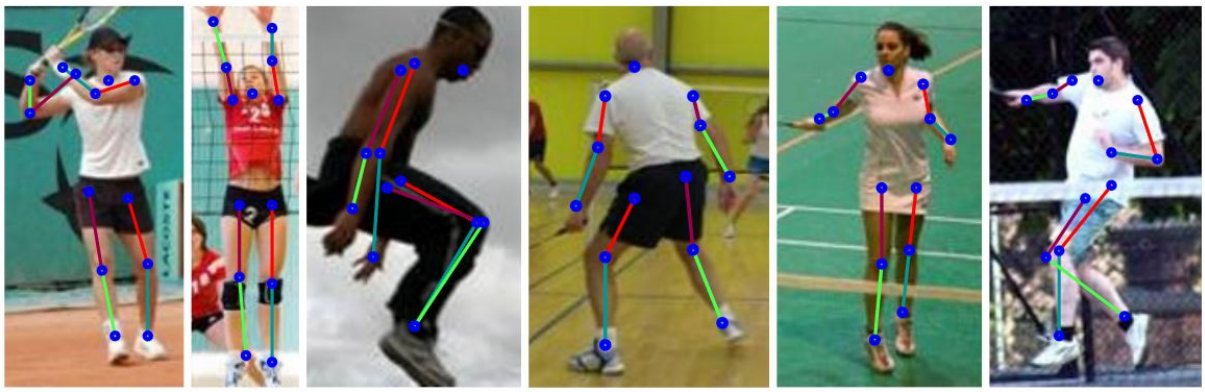


Figure 2.2: Examples of human poses estimated using the hybrid architecture proposed by Tompson *et al.* [35].

From these works, most research in human pose estimation has shifted towards DL-based solutions. Newell *et al.* [36] propose a novel CNN architecture for this particular task forcing *bottom-up* and *top-down* processing with intermediate supervision. In [37], Wei *et al.* inherit the *Pose Machines* architecture proposed by Ramakrishna *et al.* [38], but introducing CNNs as feature detectors. Their sequential model, based on multiple stages which take as input belief maps from the previous stages, enables learning long-range dependencies among parts. A more general approach for spatial localization tasks is proposed by Gkioxari *et al.* [39]. Because of their intuitiveness and performance, we have chosen these methods ([36], [37] and [39]) as 2D human pose estimators for our work.

We will further describe their architectures and learning strategies in Chapter 3.

One of the most successful works published in the last years is the one proposed by Chen *et al.* [40]. In order to deal with joint occlusions and overlapping of human body parts, they propose the usage of structure-aware CNNs and *Generative Adversarial Networks* to train a pose generator only yielding plausible poses, implicitly learning priors about the human body structure. This approach is out of the scope of this paper because, to the best of our knowledge, there is no open source code available.

The aforementioned works are focused on 2D pose estimation from individuals on single images. However, several authors have also tried to apply DL-based techniques by taking advantage of the temporal information provided by sequences of images [41, 42, 43, 44]. Such is the case of Pfister *et al.* [42], who introduce in their pipeline dense optical flow estimations to warp estimated per-joint heatmaps in consecutive frames. In that way, the trained CNN is forced to learn the temporal relationships between sequences of poses. Another approach to human pose estimation in 2D video is proposed by Girdhar *et al.* [44]. In their work, they propose a 3D extension of the *Mask R-CNN* architecture [45] to couple spatio-temporal information. DL-based methods have also been employed in multi-person scenarios [13, 46, 47, 48] and dense pose estimations, which map image pixels of the human body to its corresponding 3D surface [49]. In Figure 2.3, examples of multi-person and dense pose estimations are shown.

A major landmark in the field of human pose estimation from RGB images is the development of the *OpenPose* suite [13]. *OpenPose* is an *open-source* software developed mostly by researchers of the *Carnegie Mellon University*. It builds upon the works in single human pose estimation by Wei *et al.* [37], multi-person pose estimation by Cao *et al.* [13] and hand pose estimation by Simon *et al.* [50]. They claim the *CMU Panoptic Studio* dataset [51] as a core element for their software. Their SDK provides real-time keypoint detection for full human body, hands, feet and face poses.

2.2 3D Estimation

2.2.1 From RGB images

Estimating 3D locations from their 2D projections is a highly ill-posed problem, as very different 3D poses can generate very similar 2D projections. However, the *a priori* knowledge about the human body and its plausible poses have allowed researchers to

(a) Multi-person pose estimation [13]

(b) Dense pose estimation [49].

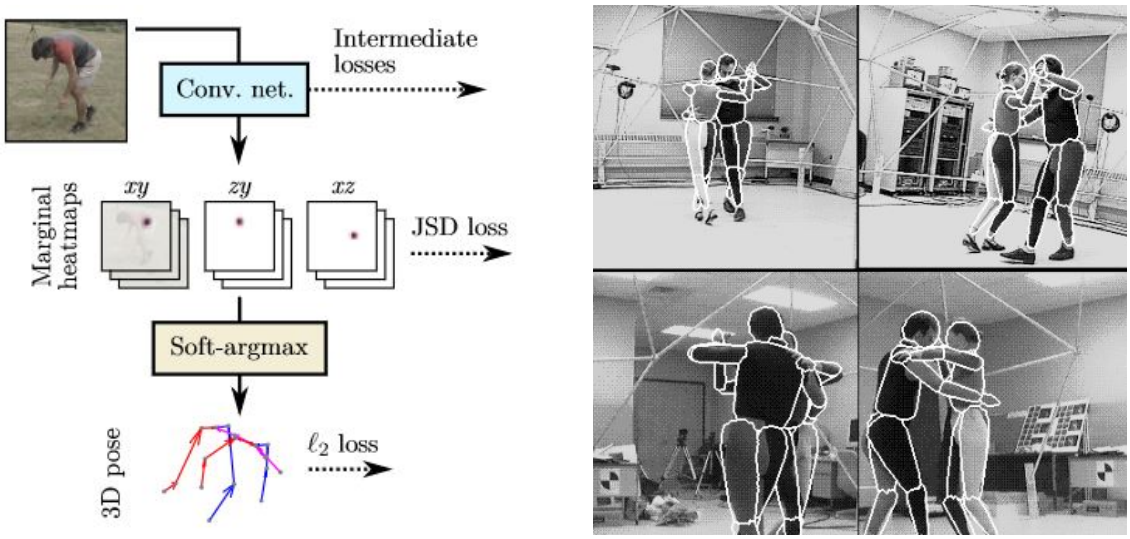
Figure 2.3: Besides the more classic scenario of detecting human joints for a single subject, a lot of efforts have been made to tackle down more complex problems. Such is the case of (a) multi-person and (b) dense pose estimations.

tackle this problem by means of hybrid approaches composed of two-stages: 2D estimation and 3D reconstruction from 2D projections. For instance, Andriluka *et al.* [52] propose a complete framework for multi-person pose detection in videos using *tracking-by-detection* and 3D exemplars. Regarding the 3D reconstruction from 2D projections, Ramakrishna *et al.* [53] propose an optimization proxy which jointly estimates 3D coordinates from 2D locations of anatomical landmarks and camera parameters while enforcing anthropometric regularity.

Most recent methods introduce CNNs for 3D estimation. Chen and Ramanan [54] make use of the Convolutional Pose Machines (CPMs) proposed by Wei *et al.* [37] for 2D estimation and then match the resulting pose with a library of 3D exemplars. In [55], Bogo *et al.* estimate 2D poses with the *DeepCut* model proposed by Pischulin *et al.* [27] and then fit a statistical body shape model to the resulting 2D joints. However, DL-based methods have not only been used for the 2D pose estimations, but they have also proved to be a very straight-forward and effective solution for inferring 3D from 2D poses. Thus, Zhou *et al.* [33] propose the inclusion of a depth regression module taking the 2D heatmaps generated by a CNN as input, providing a 3D pose estimation as the output. A simple but effective approach is proposed by Martinez *et al.* [56] by training a *Deep Neural Network* to estimate 3D body joint locations from the corresponding 2D positions.

For an adequate performance of the trained model, they consider the inverse transform of the camera to preprocess the 3D ground-truth before training, thus making the 2D to 3D problem similar across different cameras. Tome *et al.* [57] propose a more sophisticated solution which not only predicts 3D poses but also uses them to improve their previous 2D estimation, blurring the lines between the two previously mentioned stages.

Several authors [58, 59, 60, 61, 62] have chosen direct inference over hybrid approaches for estimating 3D coordinates from 2D images. Such is the case of the solution proposed by Mehta *et al.* [62]. In their work, not only they infer 3D poses but also add temporal filtering and fit a kinematic skeleton model in order to take advantage of temporal correlation between frames. More recently, Nibali *et al.* [58] introduce a CNN that, taking an RGB image as input, yields marginal heatmaps for each joint projected in xy, xz and yz planes. The final three-dimensional estimation is given by a *SoftMax* layer, which makes their model fully differentiable. In Figure 2.4(a) an overview of the pipeline proposed by Nibali *et al.* is shown. Another novel solution for inferring 3D poses directly from RGB images is the one proposed by Ramírez *et al.* [63]. In their work, they propose a brand new *Bayesian* formulation of the very promising, and yet to be fully explored, *Capsule Networks*.



(a) 3D estimation from a single view [58].

(b) 3D estimation from multiple views [64].

Figure 2.4: Examples of different approaches for 3D human pose estimation using RGB images from: (a) single and (b) multiple views.

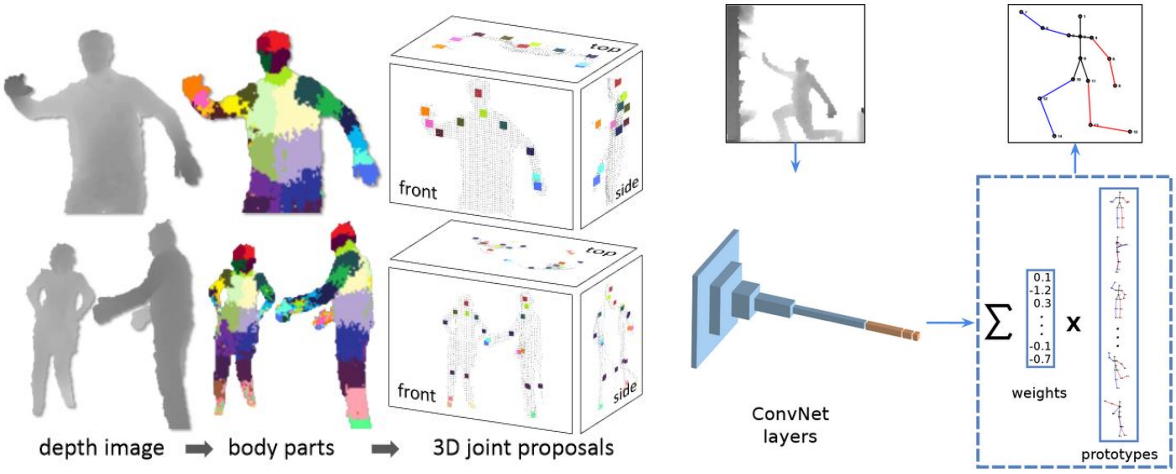
Another alternative for estimating three-dimensional locations from RGB images is combining multiple 2D views. Such is the case of the classic solution presented by Gavrila

and Davis [64]. They interpret the human body as a 3D graphical model with 22 degrees of freedom, in which each part is represented by superquadric geometric shapes. This model is fitted using frontal and lateral views of the subjects studied, as it is shown in Figure 2.4(b). More recently, Núñez *et al.* [65] proposed a pipeline that combines the usage of a CNN and a *Long-Short Term Memory Neural Network* for solving the same task. In their work, they estimate 2D coordinates in each view using the CNN architecture proposed by Cao *et al.* in [13]. From these estimations, they minimize the reprojection error to 3D by using least-squares optimization. The final 3D estimations are refined taking advantage of temporal cues with the aforementioned *Long-Short Term Memory Neural Network*.

2.2.2 From RGBD images

The inclusion of depth measurements allows for higher accuracy in the estimated poses and better understanding of the scene being analyzed. It greatly reduces the ambiguity caused by the lost of information implicit in the 3D to 2D projection, even though it is still a challenging problem due to self-occlusions and limitations in image resolution. One of the most relevant approaches for human pose estimation from RGBD data is the one proposed by Shotton *et al.* [66]. In their work, they subdivide the estimation of the 3D coordinates in two tasks: per-pixel body parts recognition and joint coordinates inference. The former is solved as a multi-class pixelwise classification problem, modeled with a *Randomized Decision Forest*. For the latter, they find modes in each labeled region using *Mean Shift* clustering. An overview of this pipeline is shown in Figure 2.5(a). This algorithm was introduced as a core component of the *Kinect* gaming platform [67]. Another classic solution for this task is presented by Youding *et al.* [68]. Using as input a video stream from a *time-of-flight* sensor, they detect and track the position of anatomical landmarks using a probabilistic inference algorithm that optimizes a simple kinematic model. Schwarz *et al.* [69] try to solve the same task using geodesic distances and optical flow. A different approach is presented in [70], where Ye *et al.* estimate body pose by matching and refining pre-captured exemplars.

Recently, DL-based solutions have been proposed to address pose estimation from depth maps. Marín-Jiménez *et al.* [71] train a CNN that estimates 3D body poses as a linear combination of prototype poses, as presented in Figure 2.5(b). Chang *et al.* [72] introduce the novelty of designing their model as a 3D CNN, which estimates the



(a) Shotton *et al.* proposal [66].

(b) Marín-Jiménez *et al.* approach [71].

Figure 2.5: Examples of different approaches for 3D human pose estimation in RGBD images using (a) classic computer vision methods and (b) CNNs.

likelihood per voxel for each joint in the pose. They test their model performance not only for addressing human pose, but also hand pose estimation. In [73], Zimmermann *et al.* use a 2D keypoint detector to estimate the 2D pose, which is next used together with the depth map as input to train a CNN yielding predictions of real-world 3D coordinates. These 3D predictions are used for robotic tasks learning. This approach will be described in detail in Section 4.3, where we will compare the performance of our proposed pipeline against the results yielded by the Zimmermann *et al.* algorithm.

Chapter 3

Proposed method

As we pointed out in the previous chapter, human pose estimation is a problem that can be approached from a multitude of different perspectives. After extensively reviewing previous work in the field, and taking into account the particular requirements of robotics applications, we propose a very lightweight solution for 3D estimation from RGBD data that builds upon some of the methods presented in Section 2.1.2. In this chapter, we present an overview of our system and its components. We have split our process into two major blocks: 2D and 3D estimation. For the former, three 2D human pose estimators have been tested. First, their architectures and training strategies are described and their pros and cons are discussed. Then, our method for getting 3D estimations from 2D coordinates is presented with greater detail, showing how each individual step works and how they contribute to the overall system performance.

3.1 System design

As previously stated in Section 1.2, our main objective is to build a system for 3D human pose estimation that is lightweight enough to be embedded in a robot and that can operate independently of the point of view, while perceiving in real-time the poses of the humans around. Keeping these requirements in mind, we have decided to design a hybrid system that relies on SOTA DL-based solutions for 2D human pose estimation and a very straight-forward method for augmenting the estimated 2D keypoints to 3D, leveraging the depth information provided by an RGBD sensor. An overview of this system is shown in Figure 3.1.

Our method does not make any assumption on the kind of algorithm used for

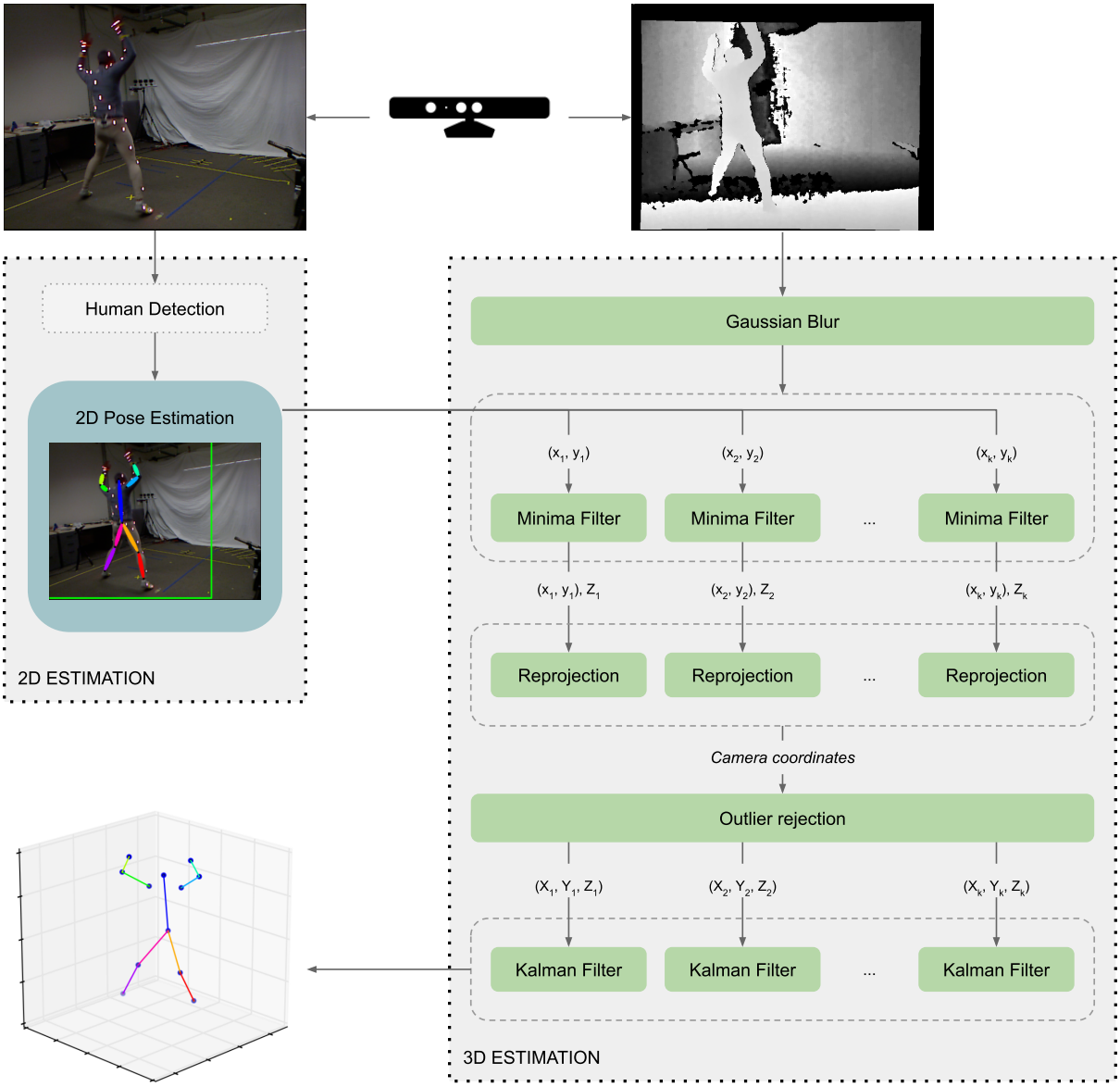


Figure 3.1: Overview of the pipeline proposed for 3D human pose estimation.

estimating the 2D coordinates, and so one can choose the 2D human pose estimator that better fits the particular application being developed. This flexibility allows the user to prioritize accuracy over computational cost or vice versa. It also ensures that our pipeline can be updated with novel solutions for 2D pose estimation that might be proposed in the years to come. It is important to mention that human detection is a required step before most of the available 2D pose estimators in the literature, but this step is out of the scope of this work, as there are some methods that might perform human and pose detection jointly [13]. For completeness, we will present a briefly description of how we achieve human detection in the next section, without going into great detail.

Regarding the 2D to 3D estimation step, we will assume that a commercial RGBD sensor is available. This is a reasonable assumption taking into account how widespread and affordable some of these sensors have become in the last years. Such is the case of the *Asus Xtion* sensor [19], which we will use to experimentally validate our system in Section 4.5. Our proposed method for estimating the 3D coordinates from the detected 2D keypoints relies on classic computer vision techniques that are well established and that fulfill our requirements in terms of computational costs. Furthermore, the only specific step for human pose estimation in this method is the outlier rejection, and so it can be easily adapted for any kind of 3D estimation given its 2D counterparts.

3.2 2D Estimation

In order to estimate the initial 2D locations of the human joints considered, we have evaluated the usage of three different DL-based solutions, which have provided good results in the literature. None of them make use of explicit graphical models. The tested methods have been chosen both for their performance, in terms of the leap forward they represented for the human pose estimation field, and their intuitiveness, which will be useful to ensure a clear analysis of the results obtained. Before presenting these methods, we will start by addressing how human detection can be performed for further improving the pose estimation results.

Human detection

Human detection is a well studied area of computer vision, which can be considered a subfield of the more generic object detection problem. Detecting humans in 2D images is specially challenging due to the variable appearance a human body can take when projected in a 2D surface, which is highly dependent on properties like pose and clothing. We are not going to dive deep into human detection literature, but it is worth mentioning the classic approach proposed by Dalal and Triggs [74], as it is probably one of the most relevant works in the field. In their work, they build a grid of *Histograms of Oriented Gradients* and classify the resulting patches using *Support Vector Machines*. This is a very efficient method and, for years, it has been established as the go-to solution for human detection.

However, as it has happened for most of the classic problems in computer vision,

DL solutions have demonstrated to be far more accurate. This is specially true in the case of human detection, as there is huge availability of potential training data. Taking that into account, we have used an *SSDLite MobileNetV2* model as our baseline human detector [75]. It is not only precise enough for our application, but it also has been specifically built to be as lightweight as possible to ensure it is suitable for being embedded in systems with scarce resources, hence the *mobile* prefix in its name. This model is based on the successful *Single Shot Multibox Detector*, initially introduced by Liu *et al.* [76] (see Figure 3.2), but tweaked with a novel layer that uses depth-wise separable convolutions to improve the efficiency of the model by reducing the number of parameters and operations needed during inference. The trained model is publicly available as part of the *TesorFlow Model Garden* [1].

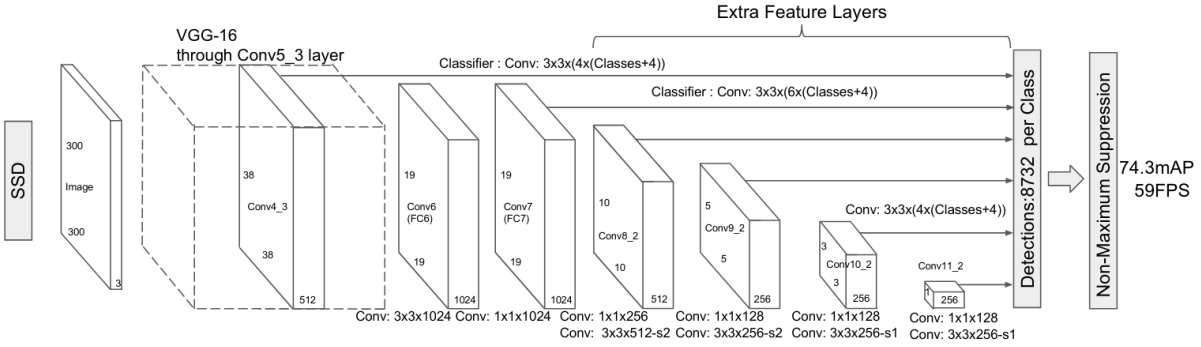


Figure 3.2: *Single Shot Multibox Detector* architecture as presented by Liu *et al.* [76]. In the *SSDLite MobileNetV2* adaptation, the convolutional layers are replaced by the depth-wise separable convolutions proposed in [75].

3.2.1 Convolutional Pose Machines

CPMs, introduced by Wei *et al.* in [37], is one of the most prominent works in the 2D human pose estimation field, as it laid the foundation for the *OpenPose* suite (see Section 2.1.2). Broadly speaking, the architecture proposed by Wei *et al.* is a combination of multi-class predictors organized into several stages. Each of these multi-class predictors is built as a CNN estimating the location of a single human joint. The image features and belief maps produced by each of these predictors in one stage are used as input for the next one. With each subsequent stage, the receptive field over the input image increases,

[1] `https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet`

implicitly encoding contextual information and allowing the network to learn correlations between parts. This effect is well exemplified in Figure 3.3, which summarizes the CPMs architecture and shows the receptive field at different stages for an example image. In the original article, results are reported for a CPM composed of 6 stages.

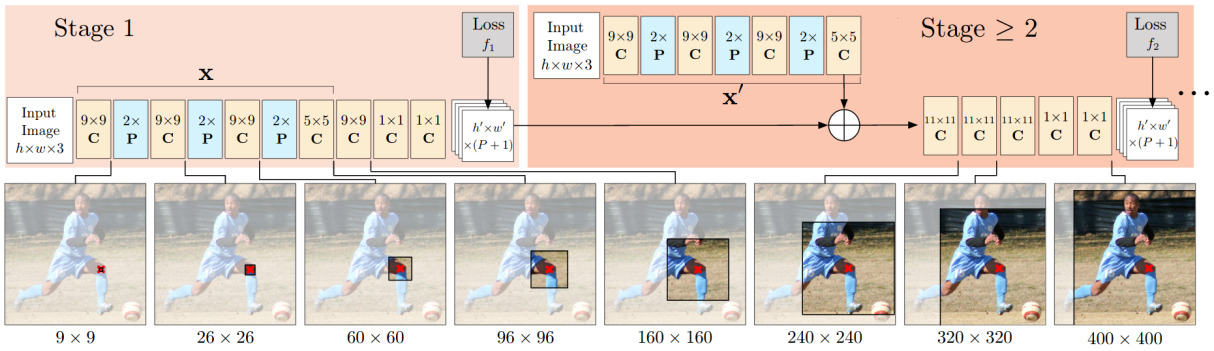


Figure 3.3: Overview of the CPMs model proposed by Wei *et al.* [37]. The top of the diagram shows the architecture of the first stage and the subsequent ones. In the bottom, the increment in the receptive field across each new stage is shown.

Regarding the model training, each stage and body part has its own specific loss function, computed as the squared error between the predicted and the target belief maps. It is worth clarifying that the target belief map of each body part is created by locating a *Gaussian* function at the ground-truth location. The individual losses that we have just described are then aggregated to provide the overall loss of the CPM being trained. In that way, all the weights involved in each of the stages and predictors can be jointly trained with intermediate supervision. During evaluation, input images are cropped in a square around the subject of interest and normalized to a size of 368x368 pixels.

3.2.2 Stacked Hourglass

Newell *et al.* [36] propose a novel CNN architecture, which leverage features in the image at different scales by repeated *bottom-up* and *top-down* processing steps, *i.e.*, from high to low resolution and vice versa, respectively. Each of these *bottom-up* and *top-down* blocks are given the name of *Hourglass Modules*. For the *bottom-up* stage, the images are downsampled by stacking several convolutional and *Max Pooling* layers. For the *top-down* stage, the lowest resolution image passes through several *Nearest Neighbors* upsampling layers until the original resolution is reached. Each *Hourglass Module* is symmetrical

and the combination of features at different scales is enforced by elementwise additions between maps of equal resolution from the *top-down* and *bottom-up* halfs of the module. In Figure 3.4, the architecture of a single *Hourglass Module* is shown in a very intuitive way. In order to further consolidate the multi-scale features extracted by the CNN and refine the estimated locations for each joint, several of these modules are stacked together. Newell *et al.* report their final results for a Stacked Hourglass (SH) model composed of 8 *Hourglass Modules*.

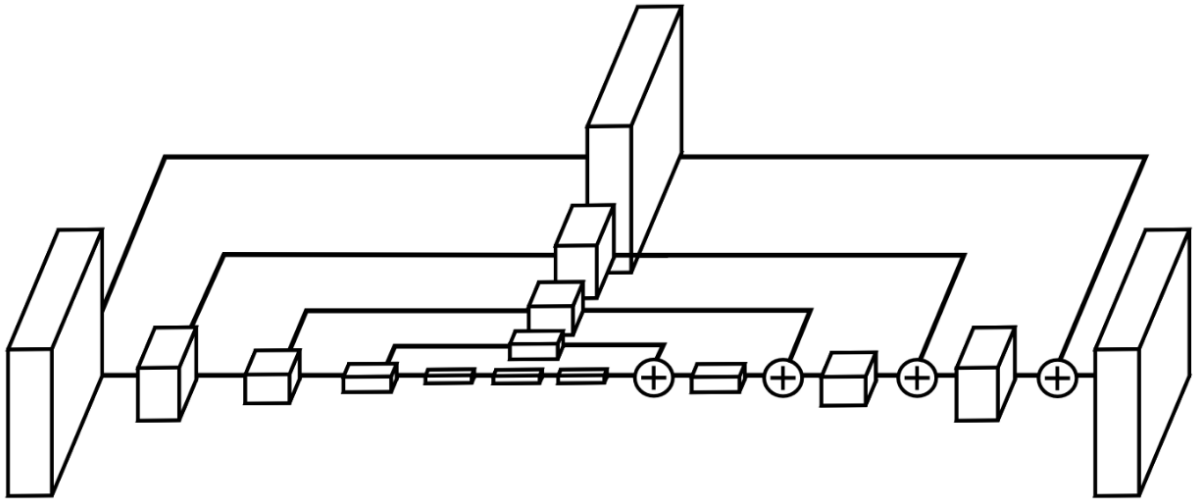


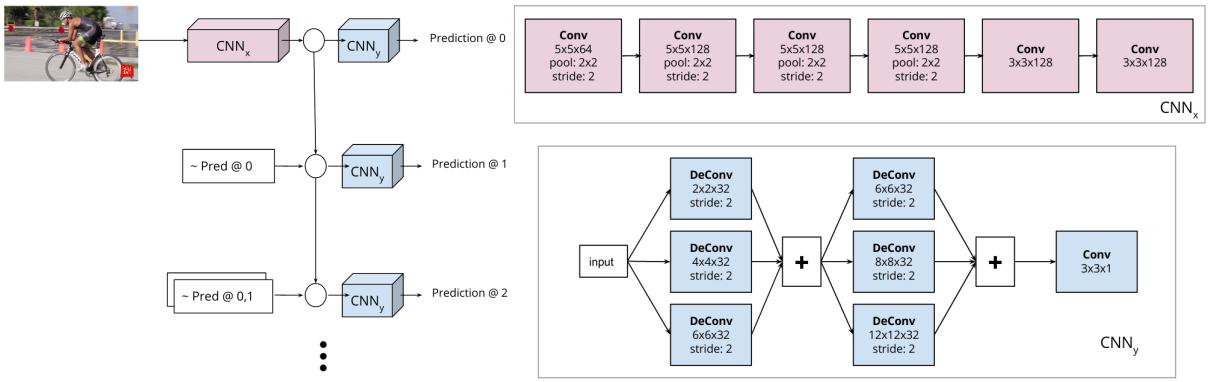
Figure 3.4: Diagram of a single *Hourglass Module*, the main building block of the *Stacked Hourglass* architecture proposed by Newell *et al.* [36].

It is worth mentioning that simply stacking *Hourglass Modules* without intermediate supervision would never take the SH model to its full potential. This intermediate supervision process is achieved by optimizing the *Mean Squared Error* between the predicted heatmaps and their ground-truth after each module. As in CPMs, the target heatmap is created by locating a bi-dimensional *Gaussian* function on each joint location. During evaluation, input samples are cropped and then resized to 256x256 pixels.

3.2.3 Chained Predictions

The method proposed by Gkioxari *et al.* [39] is based on *sequence-to-sequence* models [77], in which multiple output variables are predicted sequentially, *i.e.*, the prediction for a given output does not only depends on the input, but also on the previously predicted output variables. They propose an extension of this kind of models for more general structured outputs taking images as input and demonstrate its performance in human

pose estimation, which is a clear example of structured prediction. In their model, the prediction for each body joint is dependent on all previously predicted joints, following a fixed ordering, which is determined according to detection rates yielded by an unchained *feed-forward* net. In that way, the easier cases are processed first, while the harder ones take advantage of all the contextual information given by the previous detected joints. In Figure 4.2(a), a diagram showing the Chained Predictions (CPs) model can be found. Gkioxari *et al.* architecture is composed of two main blocks, an encoder that processes the input image with several convolutional operations (CNN_x), and a decoder composed of deconvolutional layers (CNN_y). The image encoded by the CNN_x is shared for every joint, while there is a single CNN_y block for each joint, which takes as input the encoded image and predictions from previous joints. The architecture of each of these blocks is presented in Figure 4.2(b).



(a) CPs architecture. (b) Main CPs model blocks.

Figure 3.5: In (a), a complete diagram of the CPs architecture proposed by Gkioxari *et al.* [39] is shown. The model proposed is composed of a single CNN_x that encodes the input image, and several CNN_y (decoders) which, taking as input the encoded image and the previous predictions, return a heatmap for each joint. Both blocks are shown in (b).

During the first iterations of the training stage, the results from the previous joints predictions are replaced by their ground-truths, as these estimations can be really inaccurate and can jeopardize model convergence. Once the previous predictions start to get better, they are gradually introduced instead of their ground-truths. This training strategy is called *scheduled sampling*, and was initially introduced by Bengio *et al.* [78]. During evaluation, input samples are rescaled to 299x299 pixels.

3.2.4 Conceptual comparison

As it has been already mentioned, none of these methods are based on an explicit spatial model that takes into account the relation between each joint. Instead, these relations are implicitly learned during training. In that way, the only model that makes an slight assumption regarding how each joint relates to each other is CPs, which have a fixed order for its sequential prediction. However, even in that case, the order is previously decided by the accuracy obtained for each joint when estimating their locations using a *feed-forward* net, so it is not imposed by any human decision.

Another shared key factor is that their training is performed end-to-end. The main difference then is how they manage to blend contextual information regarding other joints location and the input images. As stated in [39], both SH and CPMs are based on an iterative process that gradually refines predictions, building up on very naive results in the first stages, while the CPs model approach produces a single set of predictions, which are already quite accurate due to the imposed chained dependency.

The differences already mentioned set apart the CPs model from the other methods described. Meanwhile, the key distinction between SH and CPMs can be found in the main building blocks of their architectures. While SH is composed of multiple *bottom-up* and *top-down* subsequent stages, CPMs further increase the receptive field of the model with each new stage. Besides that, CPMs share weights across stages, while in [36], weights are not tied between each *Hourglass Module* in any way.

3.3 3D Estimation

Once 2D detection has been performed, the next step in our pipeline is the estimation of the 3D locations of the detected joints. For this task, we rely on image sequences captured with an RGBD sensor. We assume that color and depth images are registered, as most commercial sensors are coupled with SDKs that provide their own calibration data and algorithms for that purpose. In this section, we present a detailed description of each step that composes our proposed method for 3D estimation. This method has been developed taking in mind the hard requirements imposed by robotics applications in terms of computational costs.

In Figure 3.1, an overview of the steps involved in the algorithm we propose for 3D estimation can be seen. Here is a brief overview of these sequential steps:

1. Noise from depth image is reduced by filtering with a *Gaussian* kernel.

2. A minima filter centered at each 2D estimation over the depth image is used to ensure we are working with distances to the human subject in the foreground.

3. Corrected 2D estimations and their corresponding depth are reprojected to get three-dimensional camera coordinates.

4. Outliers in the resulting 3D keypoints are rejected.

5. The final 3D estimation for each joint is given by a *Kalman* filter that smooths the results leveraging temporal cues. It also provides estimations for the joints rejected in the previous step.

3.3.1 *Gaussian* blur

Gaussian blur is a basic tool widely used in computer vision for reducing noise in images, at the cost of losing high-frequency details. It is named after the mathematician Carl Friedrich Gauss. When working in the spatial domain, this filtering is equivalent to a convolution between the image and a two-dimensional *Gaussian* function, defined as:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.1}$$

where x and y represent the distance to the origin of the filter in each dimension and σ determine its standard deviation.

Depth images are prone to contain a significant amount of noise, as they can be highly influenced by environmental factors. Such is the case of cameras that rely on projected light in the infrared domain, which can be influenced by sunlight, or even stereo cameras, which might fail to match pixels between images if the surface does not have enough salient points. Besides that, losing high-frequency details in the resulting depth estimations is not a major issue, as distance to camera is supposed to follow a smooth progression within the surface of the objects of interest in the scene. Applying a *Gaussian* blur to the depth image alleviates the undesired effects mentioned above.

In our case, we apply a kernel of size 5x5. The standard deviation is automatically computed from the given kernel size by *OpenCV* [79]. In Figure 3.6, an example of a filtered depth map can be seen. It is important to note that the size of the kernel used

for filtering is highly dependent on the image resolution and the size of the objects in the scene, so it might need some tuning depending on the particular application.

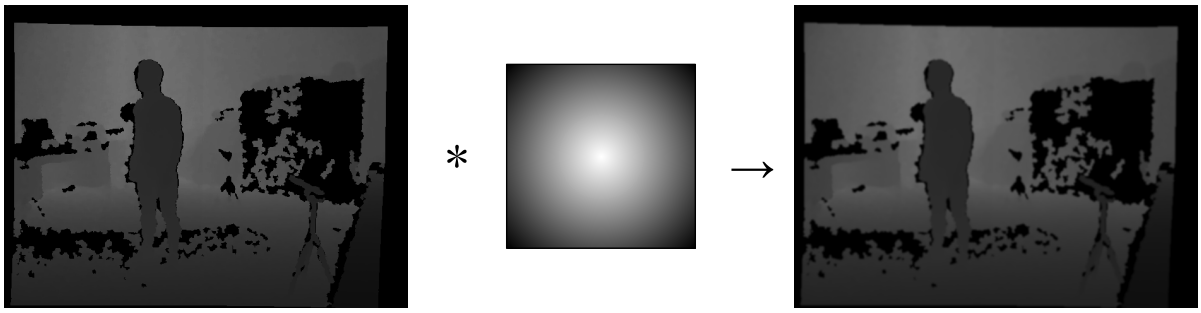


Figure 3.6: Naive example of a *Gaussian* blur applied to a depth map.

3.3.2 Minima filter

Besides the inherent noise in depth images, the 2D estimations provided by the methods described in Section 3.2 can be slightly displaced from the real joint location. Our next strategy for filtering out outliers in the depth estimation is a minima filter around the 2D detected region in the depth image. We choose a minima filter because we are retrieving spatial information from subjects who are consistently closer to the camera than the rest of the objects in the scene. Our minima filter is also useful if the pixel in the depth image for the detected 2D keypoint contains a null value, which is quite frequent in RGBD sensors. An example of how this technique can help filtering out noisy depth estimations is shown in Figure 3.7.

We fix the initial kernel size for this minima filter to 11x11 pixels around each joint. Once again, this kernel size might depend on the image resolution and the size of the objects in the scene. Let D be our depth image and W_{x_j,y_j} a window of the depth image D with size 11x11 centered around the estimated 2D coordinates x_j, y_j for the j-th joint, we estimate its distance to the camera as

$$z_j = min(W_{x_j y_j}) \tag{3.2}$$

Exceptionally, if no valid pixel is found in W_{x_j,y_j} due to corrupt data in D, we increase the size of the filter until at least one valid pixel is found.

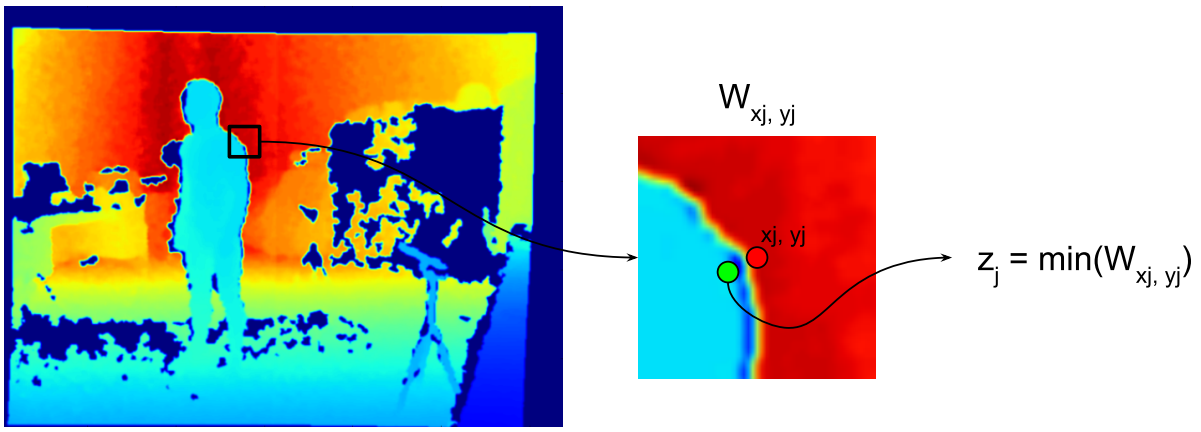


Figure 3.7: Minima filter over the detected 2D joint region can help filtering out noisy depth measurements.

3.3.3 Reprojection

For reprojecting the estimated 2D coordinates into a three-dimensional space, we will use a pinhole camera model, which simplifies how a real camera works by assuming an infinitely small aperture and no lens. In our case, this is a reasonable assumption as most SDKs provided by depth sensors manufacturers allow the user to correct the geometric distortions that the camera lens might have introduced in the image.

The image formation process for pinhole cameras can be described by its intrinsic and extrinsic matrices. The former describes the parameters of the system that are inherent to the camera itself, *i.e.*, its focal length in x and y dimensions (f_x, f_y) and the coordinates of its optical center (c_x, c_y). The latter includes a description of how the camera coordinates relate to the world coordinates, expressed as a three-dimensional rotation and translation matrix. In Figure 3.8, we can see how all of these parameters are involved in the image formation process, from world coordinates to pixels in the final image.

In our case, given the estimated distance to camera z_j, image coordinates x_j, y_j, depth image D dimensions (w, h) and camera intrinsic parameters f_x, f_y, c_x, c_y, we get the 3D camera coordinates of each joint j as:

$$X_{j,cam} = \frac{(w - x_j - c_x) * z_j}{f_x} \tag{3.3}$$

$$Y_{j,cam} = \frac{(h - y_j - c_y) * z_j}{f_y} \tag{3.4}$$

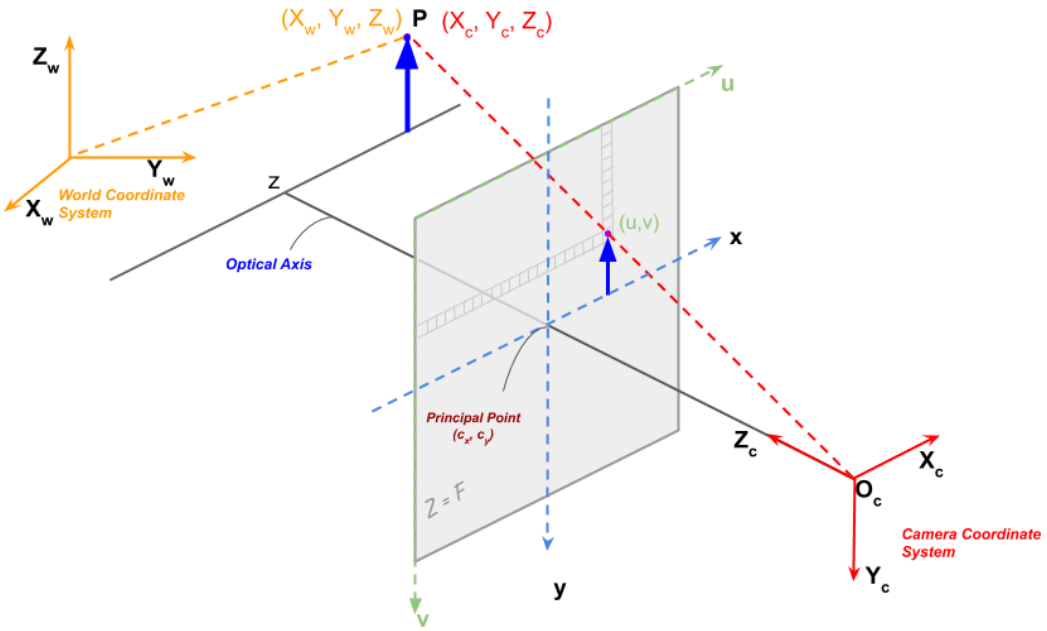


Figure 3.8: Projection of point P from world coordinates to camera, image and pixel coordinates, as modeled by a pinhole camera (source [80]).

$$Z_{j,cam} = z_j \tag{3.5}$$

If extrinsic camera parameters are also provided, these camera coordinates can be projected to real-world coordinates. However, camera coordinates are enough for the performance evaluations and demos presented in this work.

3.3.4 Outlier rejection

As we already mentioned in Section 3.3.2, we might find outliers in our predicted 3D coordinates due to inconsistencies in the depth maps and noisy 2D estimations. Some of these outliers are so clear that they can be rejected with very intuitive sanity checks. It is important to note that simply rejecting these outliers would be unacceptable if no further processing was performed on the estimations. However, thanks to the *Kalman* filtering presented in next section, we can replace rejected estimations for a joint with predictions yielded by its *Kalman* filter based on results from previous frames.

For our application, we will consider two criteria for identifying outliers: based on the distance of the joints to the body center and based on their motion speed between

consecutive frames.

Based on distance to body center

Taking as body center the estimated 3D coordinates for the pelvis center, we reject any joint with an estimated distance to that particular point higher than 150cm. This threshold has been imposed taking into account the common range of heights in adult humans (around 95% of population height lies between 150 and 190cm [81]). We choose as reference the pelvis estimation as it is rarely occluded and, even if the estimation is a bit off, is very unusual to have strong deviations from its true location. In that way, we try to ensure a more robust outlier rejection.

Based on per-joint velocity

Another easy way to check whether a joint has been erroneously estimated or not is validating the amount of displacement between frames. We reject any joint with a displacement between frames higher than 50cm, *e.g.* if we assume a frame rate of 20 frames per second (fps), that would be a velocity higher than 10m/s.

3.3.5 *Kalman* Filtering

Until this point of the process, human joints locations are estimated in a *frame-by-frame* manner, without imposing any temporal dependency. In order to reduce noise and uncertainty in the 3D coordinates estimated by our method when applied to RGBD video sequences, we use an independent *Kalman* filter [82] for each joint in the skeleton.

Kalman filters are a very powerful tool for filtering linear continuous data at a very low computational cost, both in terms of memory and time, which makes them suitable for embedded applications that must work in real-time. In a nutshell, they provide estimates of unknown or hidden variables given their uncertain measurements observed along time. Besides their lightweight implementation, another key advantage of *Kalman* filters is that they do not need labeled training data of any kind to make their estimates. A great limitation of this algorithm is that for obtaining an optimal result, *Gaussian* noise is assumed.

As it is shown in Figure 3.9, at each time step the algorithm performs two stages: prediction and update. During prediction, an estimate of the state of the variables

defined is given depending on the physical model imposed by means of a transition matrix, the previous measurement and predictions and its uncertainty matrix. With each new measurement, both the state of the system and its uncertainty are updated and a new prediction is yielded. In our case, the state vector is formed by the three-dimensional location of each human joint estimated.

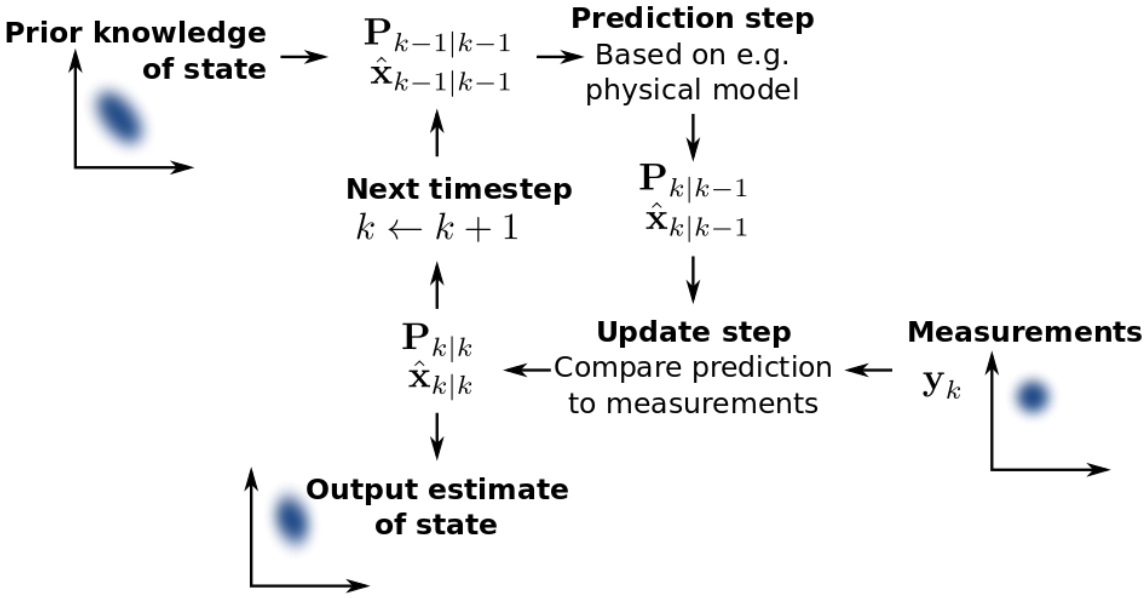


Figure 3.9: Definition of the *Kalman* filter algorithm in terms of its prediction and update stages [83].

In this work, we have used the *pykalman* library for *Python* for implementing *Kalman* filters [84]. It provides a very intuitive interface and allows returning predictions even if no new measurement is provided. We leverage this property for yielding new estimations for joints that have been marked as missing by our outlier rejection mechanisms. In order to avoid too much spatial drifting caused by these missing estimations, if no measurement is provided during n consecutive frames for a given joint, its *Kalman* filter is reinitialized as soon as a new estimation is available. In our tests, we have set n to 4 frames.

Chapter 4

Experiments

Each component of the system described in the previous chapter has its own influence in the performance of the whole method proposed in this work. We have yet to discover if their inclusion in the 3D human pose estimation pipeline is justified. Furthermore, we have to show how our lightweight algorithm compares with more complex solutions in the literature, and discuss if we have reached a fair trade-off between computational costs and accuracy.

For that purpose, we have performed both quantitative and qualitative evaluations. In this chapter, we show and discuss the results obtained for both of them. We will start presenting the publicly available dataset we have used as benchmark for both 2D and 3D pose estimations. Then, we will define the figures of merit used for evaluating 2D estimations and analyze the different results achieved by each of the methods described in Section 3.2. Regarding 3D estimation, relevant figures of merit will be defined too and used to evaluate the effects of the mechanisms described in Section 3.3. Quantitative evaluation will be concluded with a thorough comparison of our method against a SOTA solution, both in terms of accuracy and computational costs. Last but not least, a demonstration of our system working in real-time will be presented, which will help us to validate whether our system is suitable as a component for robotics applications.

4.1 Benchmark dataset

For quantiative evaluation, we will use as benchmark the publicly available BMHAD [10]. This database is comprised of synchronized data from multiple color cameras, *Kinect* sensors, accelerometers, microphones and a MoCap system. From all of these, we will use

Kinect RGBD videos as input data for evaluation and MoCap data as ground-truth. The database is divided into scenes recorded from 12 subjects performing 11 different actions, with 5 repetitions per action. In other words, a total number of 660 scenes are provided. Each of these scenes has a varying length, adding up to 82 minutes of recording. As the database provides data from two *Kinect* cameras capturing the scene from different viewpoints, we will include a total number of 1320 scenes in our analysis. In Figure 4.1, a diagram of the set-up used for capturing these scenes is presented.

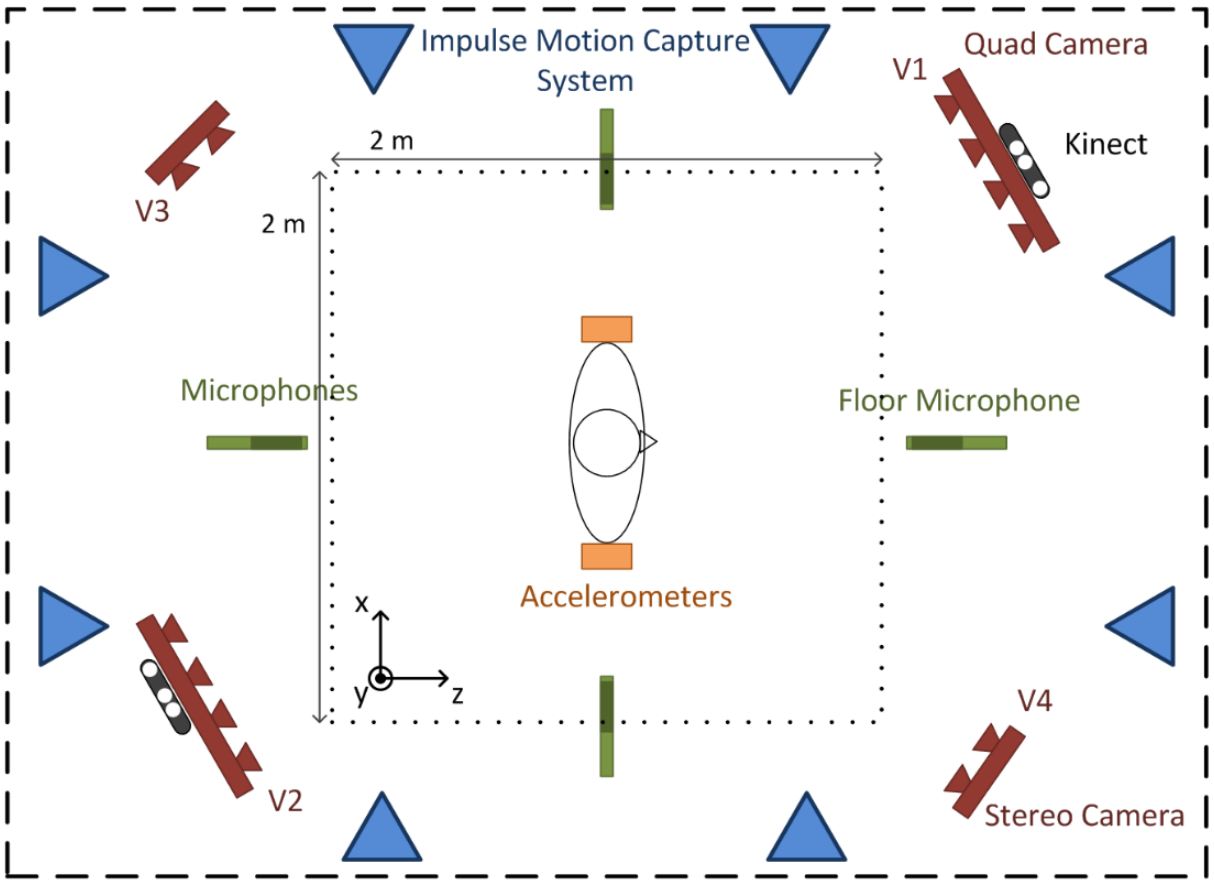


Figure 4.1: Diagram of the data acquisition set-up used for building BMHAD [10].

After studying other public datasets published in the last years, we found BMHAD to be the only one that satisfies our needs. First of all, most of the *large-scale* datasets in the literature are focused on 2D pose estimation (*e.g.* MPII [85] and *Leeds Sports Pose* [86]). Among those containing 3D annotations, it is very hard to find registered RGB and depth images, which is a major requirement for our algorithm. That is the case of the prominent *Human3.6M* [14] and *HumanEva* [15] datasets. Besides that, a lot of datasets with 3D information are more oriented to serve as input for action recognition tasks, and the joint

locations are estimated with SDKs provided by the manufacturers of the RGBD cameras, instead of high accuracy locations as provided by a professional MoCap system. Such is the case of the *Cornell Activity Datasets* [87]. Taking all of these factors into account, the only *large-scale* dataset that provides high-quality annotated joint locations registered and synchronized with both RGB and depth images is the BMHAD.

Originally, the BMHAD dataset provides labels for 21 joints extracted from the raw MoCap data. However, our final estimations depend on the 2D keypoints given by the models presented in Section 3.2, which have been trained with samples from the MPII dataset. In order to make our results comparable with BMHAD labels, we have reduced the number of joints provided to 12: head, shoulders, elbows, hands, pelvis, knees and feet. For head and pelvis, some modifications to the original labels have been performed in order to match BMHAD and MPII definitions. In the case of the head label, the *HeadEnd* and *Head* labels in BMHAD, and the *head top* and *upper neck* labels in MPII have been averaged, respectively, in order to get a unique comparable head keypoint. In the case of the pelvis label, the *LeftUpLeg* and *RightUpLeg* labels in BMHAD and the *l hip* and *r hip* labels in MPII have been averaged as well. For the latter, merging left and right hips labels has been necessary because of slightly different definitions in terms of what is considered *hip* or *UpLeg* in each of the datasets, looking much more similar between them after the averaging process. In Figure 4.2, a visual comparison between the original BMHAD labels and the result of our transformation is shown.

It is worth mentioning that a major limitation of using BMHAD for assessing our performance is its lack of scenes *in the wild*. This is an issue with no easy solution for the field of 3D human pose estimation, as precise labels can only be captured by means of cumbersome MoCap systems, which can hardly be used in more natural scenarios. Besides that, the dataset only provides two different points of view, so further research in terms of how our system performs under more intricate perspectives can only be done qualitatively. Another issue we have discovered is that, for frames with fast motions, MoCap labels might be displaced, as it can be seen in the left hand of the subject in Figure 4.3. Even though this effect might slightly drop our accuracy, it will equally affect every method tested and so comparisons will still be well-founded.

In order to provide a more intuitive interpretation of the results, the different actions considered in the dataset have been classified in terms of their dynamics and the occlusion degree of the human body joints, as follows:

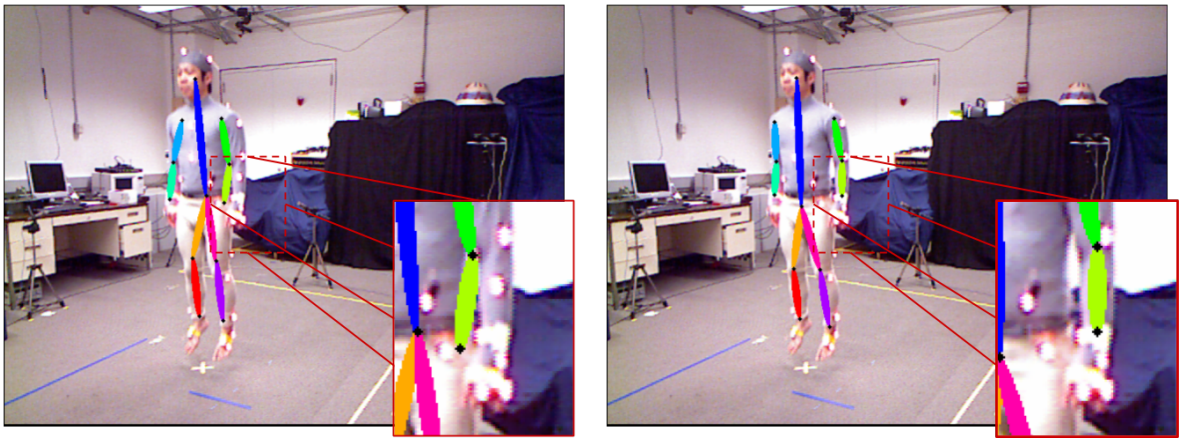
(a) Original BMHAD labels (red crosses). (b) Modified labels with joints linked.

Figure 4.2: In order to compare our results with the labels provided by BMHAD, we have transformed the original keypoints shown in (a) to the simpler configuration shown in (b). Please note that in (a) markers used for MoCap and final keypoints are shown in blue and red, respectively. Also, (a) and (b) show the same moment in time, but captured with different cameras.

- **High dynamics and heavily occluded**: Bending (*a03*) and throwing ball (*a08*). From now on, results for these actions will be presented in *red*.
- **High dynamics and slightly occluded**: Jumping in place (*a01*), jumping jacks (*a02*) and boxing (*a04*). From now on, results for these actions will be presented in *orange*.
- **Low dynamics and heavily occluded**: Sit down then stand up (*a09*), sit down (*a10*) and stand up (*a11*). From now on, results for these actions will be presented in *green*.
- **Low dynamics and slightly occluded**: Waving with two hands (*a05*), waving with one hand (*a06*) and clapping hands (*a07*). From now on, results for these actions will be presented in *blue*.

4.2 2D Estimation quantitative evaluation

Following the approach of many works in the literature, the person detection and subsequent pose estimation tasks are decoupled. For evaluation purposes, we will assume



(a) Original BMHAD label projected in 2D. (b) Joints estimated using CPMs.

Figure 4.3: In scenes with rapid movements, such as *a01*, BMHAD labels might suffer some displacements. In (a), this effect is demonstrated to be significant for the left hand label. In (b), the corresponding estimation is shown.

that a bounding box fitted around the subject can be inferred from the ground-truth. More precisely, we compute the center of the human pose as the average location between all the joints and crop a squared region around it, which is 1.25 times bigger than the biggest difference between joints in x or y dimensions. It is important to note that these bounding boxes have been resized for each method evaluated according to the sizes defined in their respective articles: 368 pixels for CPMs and 256 pixels for both SH and CPs.

As mentioned in Section 4.1, we will use BMHAD for evaluation. However, as BMHAD only provides labels in 3D coordinates, we have projected the 3D joint locations into the RGB video sequences using the camera calibration parameters included in the dataset. For this evaluation, both our estimations and the ground-truth labels have been converted to the 12 joints model described in the previous section.

4.2.1 Figures of merit

The performance of the 2D pose estimation approaches presented in Section 3.2 has been evaluated according to the Percentage of Correct Keypoints (PCKh) indicator [85]. According to the PCKh score, the estimation of the joint position is considered to be correct if the distance between the estimated and the ground-truth joint locations is below a threshold dependant on the head segment length. The threshold is usually indicated after the *at* symbol, thus for a PCKh@0.1, we would consider as correct keypoints those

with an error lower than the 10% of the head link size.

For the results presented in this report, the head segment length is obtained for each frame from the corresponding ground-truth. The PCKh score is computed for each joint considering all the frames in the same video sequence for thresholds ranging between 0 and 1 head segment lengths. Regarding missing joints, they are considered to be wrong regardless of the threshold.

4.2.2 Experimental results

For the 2D estimation evaluation, results per joint and action are presented for the three methods tested: CPMs [37], SH [36] and CP [39]. As it can be seen in Figure 4.4, these methods show similar performance with respect to each other regardless of the human joint. Having said that, CPMs perform slightly better than the other two for every joint if we set the threshold to the length of one head segment (PCKh@1), with a mean score of 95.85% of parts correctly detected (see Table 4.1). Regarding the differences in performance between joints, hand locations are harder to disambiguate than any other joint, with a mean PCKh@1 below 90% in every method tested. This can be justified by the higher level of occlusion they might have in comparison with other joints. Besides that, hands are the joints with higher dynamics in the actions included in the dataset, and labels might be displaced as we previously mentioned in Section 4.1. Figure 4.3 presents an example of a misplaced hand label, along with our estimation.

	Head	Shoulders	Elbows	Hands	Pelvis	Knees	Feet	TOTAL
CPMs	**98.73**	**97.34**	**93.28**	**88.20**	**99.31**	**98.03**	**99.26**	**95.85**
SH	96.06	95.11	91.89	86.71	96.26	96.04	96.74	93.77
CPs	96.11	95.12	88.93	80.97	95.88	95.35	94.80	91.86

Table 4.1: Quantitative 2D pose estimation results on BMHAD per joint, using as figure of merit PCKh 2D @ 1 (%). Values in bold correspond to the best results achieved for each category.

The relation between the actions performed in the evaluated scenes and the results in the 2D pose estimation for each method are presented in both Table 4.2 and Table 4.3. In the latter, the actions have been grouped in terms of dynamics and occlusion, as described in Section 4.1. Taking a look at Table 4.2, it can be concluded that, in general, CPMs

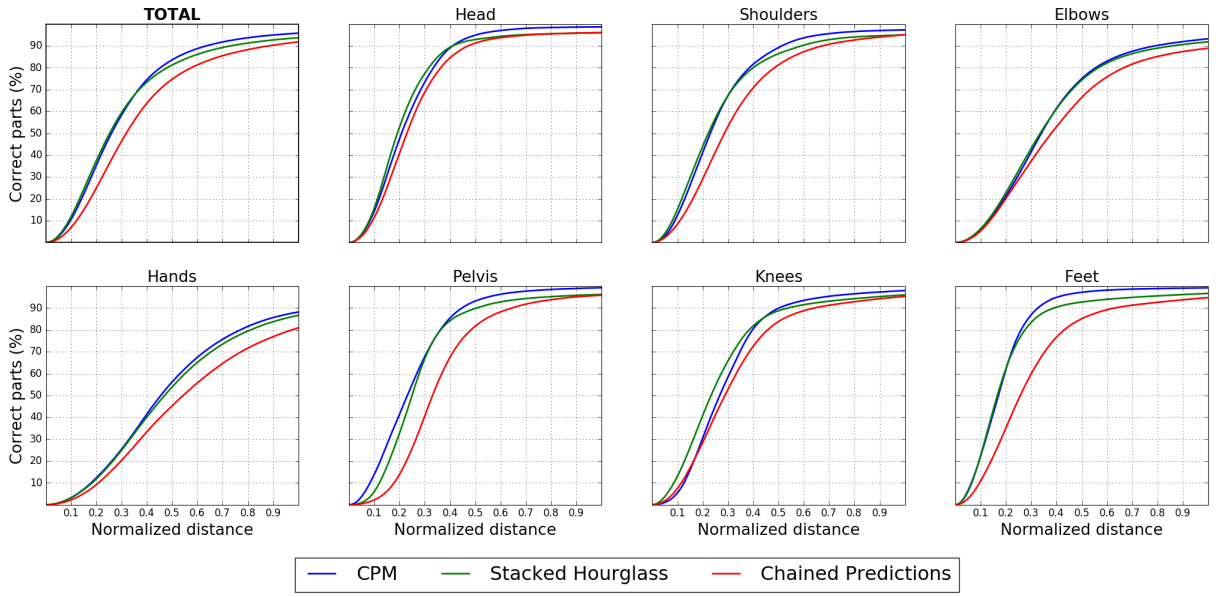


Figure 4.4: Quantitative 2D pose estimation results on BMHAD per joint, using as figure of merit PCKh 2D @ 0-1 (%).

	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10	a11	TOTAL
CPMs	97.98	96.90	**87.49**	**97.04**	**98.23**	**97.26**	**99.30**	**96.25**	**98.04**	**97.14**	**98.16**	**95.85**
SH	**98.35**	**98.40**	80.42	96.30	97.42	95.80	97.91	93.96	96.77	94.73	96.13	93.77
CPs	96.55	94.07	77.69	95.06	94.05	93.71	97.40	93.34	96.19	92.60	95.64	91.86

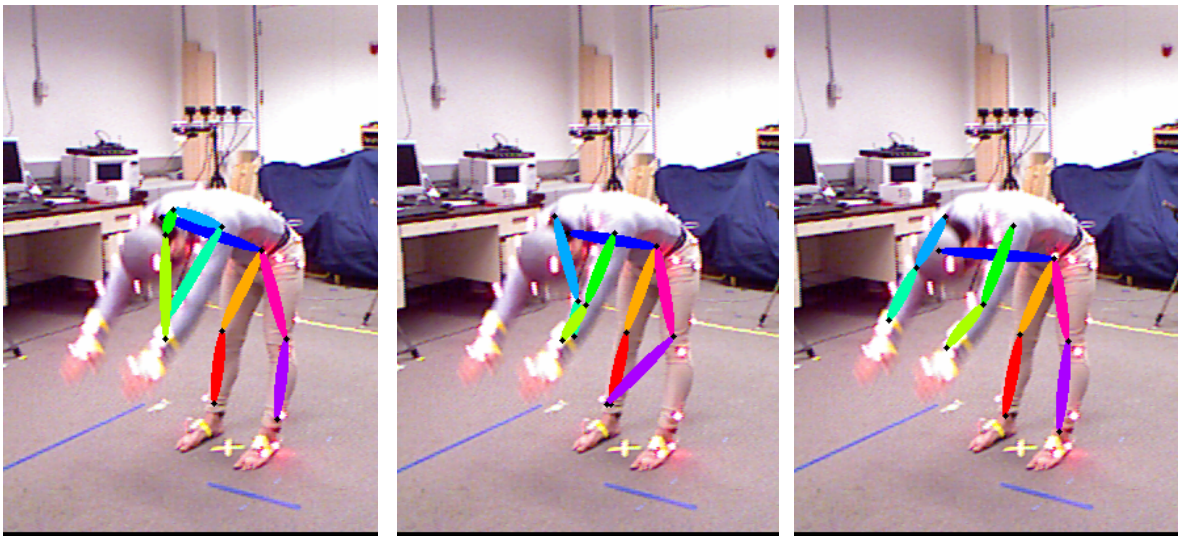
Table 4.2: Quantitative 2D pose estimation results on BMHAD per action, using as figure of merit PCKh 2D @ 1 (%). Values in bold correspond to the best results achieved for each category. Columns are colored following the convention established in Section 4.1.

	High Dynamics		Low Dynamics	
	Heavily Occluded	Slightly Occluded	Heavily Occluded	Slightly Occluded
CPMs	**91.87**	97.31	**97.78**	**98.26**
SH	87.19	**97.68**	95.88	97.04
CPs	85.52	95.22	94.81	95.05

Table 4.3: Quantitative 2D pose estimation results on BMHAD per action type, using as figure of merit PCKh 2D @ 1 (%). Values in bold correspond to the best results achieved for each category. Columns are colored following the convention established in Section 4.1.

performs better than SH and CPs, as it gets a significantly better score for most of the actions. Besides, as presented in Table 4.3, all methods perform similarly for actions with

low dynamics and/or slightly occluded joints, with a difference of only ±3% in the final score. However, there is a much greater gap in performance when the scenes with the hardest actions are evaluated, *i.e.*, heavily occluded actions with high dynamics. For this kind of actions, CPMs get a PCKh@1 of 91.87%, while SH and CPs achieve a 87.19% and 85.52% of correct parts, respectively. This is a very relevant result . As BMHAD has been captured in a controlled environment, determining which method performs better under adverse circumstances can give us an idea of how these methods would perform in the real world. Figure 4.5 displays an example where CPMs outperforms its competitors when estimating a very challenging pose.



(a) CPs failed estimation. (b) SH failed estimation. (c) CPMs accurate estimation.

Figure 4.5: In challenging scenes like *a03*, with high dynamics and occlusions, CPMs (c) outperform CPs (a) and SH (b) methods.

4.3 3D Estimation quantitative evaluation

For 3D estimation, we will rely again on the labels provided by the MoCap data included in the BMHAD dataset. For simplicity, all of the estimations generated in this work are given in camera coordinates. Taking that into account, the corresponding BMHAD labels have been transformed from world to camera coordinates for the evaluation. Also, following the same strategy described in the previous section, both estimations and ground-truth labels have been further transformed to match our 12 joints model defined in Section 4.1. It is important to note that these transformations have been applied on the final 3D

estimations, but the 2D estimations used as input have preserved their original MPII joints definition. In that way, we can fairly compare our results with those provided by other methods that might have been trained for 3D estimation with different label definitions.

4.3.1 Figures of merit

The performance of the 3D estimation methods presented in this work has been evaluated using the Mean Per Joint Position Error (MPJPE) and the 3D version of the PCKh score, already described in Section 4.2.1. MPJPE is simply defined as the average *Euclidean* distance between the prediction for the joints location in the skeleton and their respective ground-truths. Regarding the PCKh score, we use the same definition presented in the previous section, being the only difference that the head link segment length will be extracted from the three-dimensional data in terms of real-world distance, instead of pixels in a 2D image.

4.3.2 Experimental results

In order to assess the effectiveness of our method for 3D estimation described in Section 3.3, we compare the MPJPE obtained using as baseline the direct 3D estimation from the 2D keypoints, without any filtering. For this test, we have used CPMs as 2D estimator, since it is the most accurate among the evaluated ones, as we demonstrated in Section 4.2.2. We have selected a subset of 132 scenes from the BMHAD dataset, containing every action performed by every subject, but only its first repetition and captured with the first of the *Kinect* sensors.

As it can be shown in Table 4.4, the overall performance of the 3D pose estimation improves significantly with the inclusion of the processing steps described in Section 3.3. Specifically, the total MPJPE gets reduced in half. It is important to note that this difference is mostly driven by the results obtained for the joints that belong to the upper body. For most of the actions in the dataset, these joints are the ones with higher dynamics and self-occlusions, which brings to light how our algorithm is able to perform under these circumstances when compared with a direct inference. When *pelvis*, *knees* and *feet* get compared, the baseline results are slightly better. As these joints are less challenging to estimate throughout the scenes in the dataset, our mechanisms can actually distort their

locations to some extent. For instance, if both the depth map and the 2D estimation are completely accurate, our minima filter could pick a location which is closer to the camera that it actually is. Nonetheless, this mechanism is very effective for correcting errors when the conditions are not ideal.

	Head	Shoulders	Elbows	Hands	Pelvis	Knees	Feet	TOTAL
Baseline	1413.2	194.5	164.0	355.5	**174.9**	**90.5**	**65.8**	277.4
Ours	**104.2**	**146.3**	**125.6**	**137.6**	198.2	103.7	79.0	**123.9**

Table 4.4: Quantitative 3D pose estimation results on BMHAD per joint, compared against our baseline, using as figure of merit MPJPE (mm). Values in bold correspond to the best results achieved for each category.

In that sense, it is worth discussing in detail the great difference for the *head* joint. In that particular case, as we explained in Section 4.1, the labels get transformed after 3D estimation, but the original 2D estimations are yielded in MPII format, which gives two different labels for the head of the subject: *upper neck* and *head top*. The latter is placed right above the head, which makes the 3D estimation consistently fail if no further processing is applied. Thanks to our minima filter, this error gets dramatically reduced. An example of such case can be seen in Figure 4.6.

In order to better understand the nature of the error we get in the estimation, we have analyzed how it affects each dimension (X, Y, Z) independently. For this purpose, we have plotted the module of the error per joint and dimension for our 3D estimation, using once again CPMs as 2D estimator. As we can see in Figure 4.7, most of the error in the evaluation comes from the Z dimension, *i.e.*, from the estimated depth or distance to the camera. As demonstrated in Section 4.2.2, the 2D estimators tested perform very accurately, with PCKh scores up to 95%. However, very slight deviations in the 2D estimation and self-occlusions can make huge differences in the estimated depth.

4.3.3 Comparison with other methods

Our approach follows the strategy proposed by Zimmermann *et al.* [73], in the sense that our estimations also rely on registered RGB and depth images. For detecting the 2D keypoints in the RGB images, they use a model provided by the *OpenPose* library [13] with fixed weights. Then, an occupancy voxel grid with a resolution of about 3 cm is

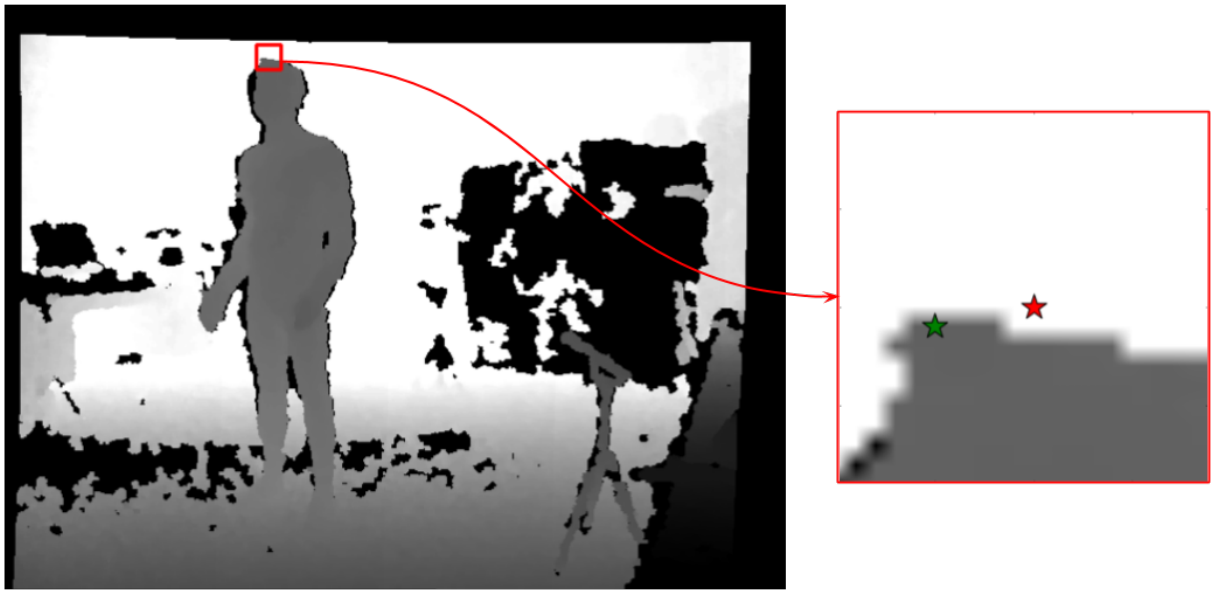


Figure 4.6: Small deviations in the 2D estimation can lead to huge differences in depth. At the left side of the figure, the depth image with the region of interest around the *head top* joint. At the right side, we can see highlighted in red the original 2D estimation and in green the point we used for depth estimation after minima filter correction.

computed by transforming the depth map into a point-cloud and, using the calibration matrix, the initial 3D coordinates of the keypoints are estimated. The voxel grid and the 2D likelihood maps are processed by a 3D CNN which optimizes the *Sum of Squared Errors* between the ground-truth and the estimated location in 3D. An overview of the architecture proposed by Zimmermann *et al.* can be seen in Figure 4.8.

Zimmermann *et al.* provided results on their own datasets, defining the skeleton as a set of 18 joints, which are based on the definition of the COCO dataset keypoints [88]. In order to compare our results with those yielded by Zimmermann's approach, we have averaged their *LEar* and *REar* keypoints and *RHip* and *LHip* keypoints to get unique head and pelvis labels, respectively. By doing so, we match the modifications presented in Section 4.1 for BMHAD and MPII datasets.

For some frames in the evaluated scenes, specially in scenarios with high occlusions, Zimmermann's method did not return any estimated location for certain joints. When computing the PCKh score, as it is simply a percentage of parts correctly detected, these joints can be included in the evaluation and are taken into account as *not correct*. However, as MPJPE is defined as an average of distances, such empty estimations are invalid and must be discarded. In order to avoid a biased comparison, such joint estimations have

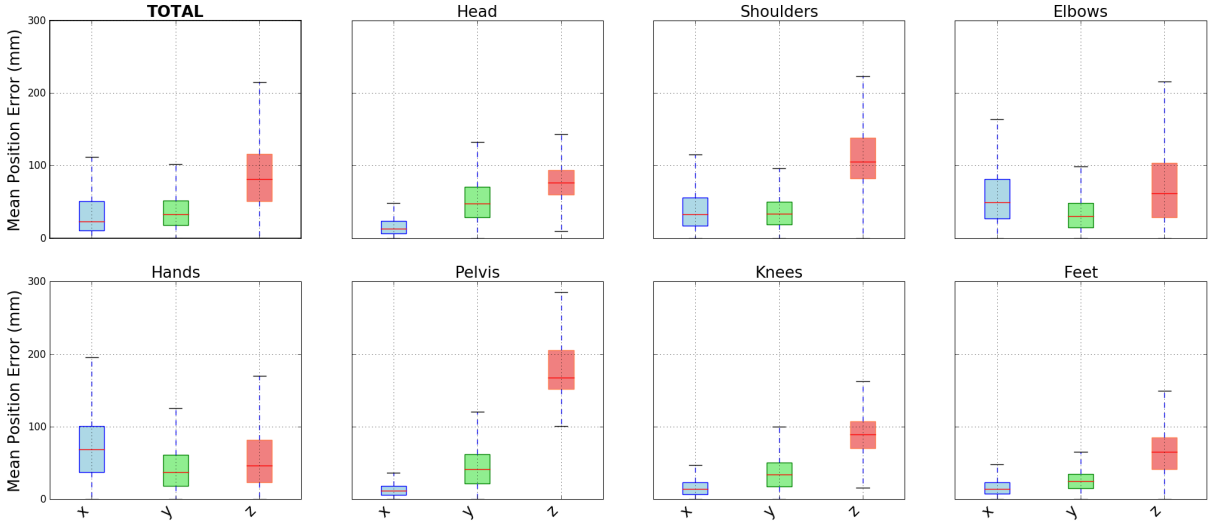


Figure 4.7: Quantitative 3D pose estimation results on BMHAD per joint, using as figure of merit the module of the difference between estimations and ground-truth for each dimension (mm).

been removed for every method when measuring the MPJPE. For this comparison, we have used the BMHAD dataset in its entirety.

Taking a look at the PCKh curves shown in Figure 4.9, we conclude that, except for the pelvis joint, the overall performance of Zimmermann's method is similar to ours, specially when using CPMs as our 2D pose estimator. In fact, Zimmermann's method only performs better for shoulders and pelvis joints, with a total PCKh@1 of 81.55%, versus an 80.23% of our method with CPMs (see Table 4.5). It is also important to note that, when evaluating 3D estimations with PCKh, final scores are lower than when evaluating 2D locations, as the added dimension increases the magnitude of the error, even though the head segment length also increases.

If we take a look at the results obtained for the same estimations when evaluated in terms of MPJPE, we see that the gap between our method and Zimmerman's increases. First of all, boxplots in Figure 4.10 show that Zimmermann's method error present a lower standard deviation for most of the joints. Moreover, according to Table 4.6, Zimmermann's mean error is lower for every joint, with the exceptions of head and feet, with a total MPJPE of 128.1 mm, versus a MPJPE of 144.1 mm for our method with CPMs.

It is still a very competitive result for our approach, particularly taking into account its simplicity and low computational burden, which will be explored in Section 4.4. Also,

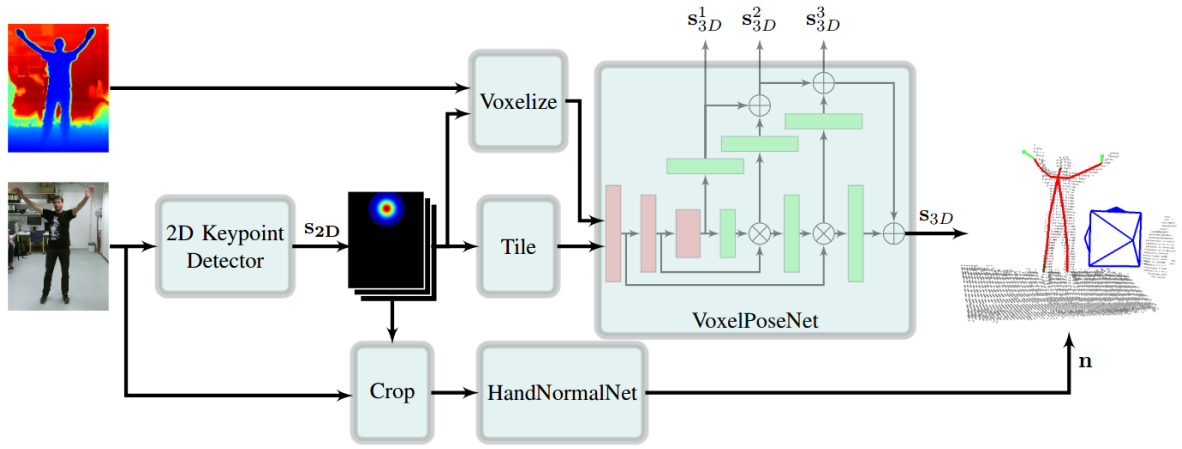


Figure 4.8: Overview of the solution introduced by Zimmmermann *et al.* for 3D human pose estimation [73].

	Head	Shoulders	Elbows	Hands	Pelvis	Knees	Feet	TOTAL
CPMs	**91.93**	86.05	**75.59**	**61.95**	68.17	**89.93**	**87.84**	80.23
SH	90.71	85.36	74.54	60.76	67.85	87.20	85.06	78.70
CPs	87.41	83.32	70.33	54.08	66.87	84.49	79.94	74.88
Zimm.	89.13	**88.05**	74.61	61.50	**91.71**	87.50	87.21	**81.55**

Table 4.5: Quantitative 3D pose estimation results on BMHAD per joint, using as figure of merit PCKh 3D @ 1 (%). Values in bold correspond to the best results achieved for each category.

	Head	Shoulders	Elbows	Hands	Pelvis	Knees	Feet	TOTAL
CPMs	**99.5**	129.4	149.9	200.5	179.9	132.8	**112.5**	144.1
SH	102.3	124.8	152.0	208.7	179.0	138.6	127.2	148.7
CPs	153.6	127.2	167.9	232.2	178.9	148.5	147.1	164.8
Zimm.	101.2	**102.1**	**143.5**	**198.7**	**95.1**	**108.2**	117.7	**128.1**

Table 4.6: Quantitative 3D pose estimation results on BMHAD per joint, using as figure of merit MPJPE (mm). Values in bold correspond to the best results achieved for each category.

our explicit outlier rejection and Kalman filtering plays a vital role in the final results, as evidenced by the differences shown in the PCKh and MPJPE scores. Even when our

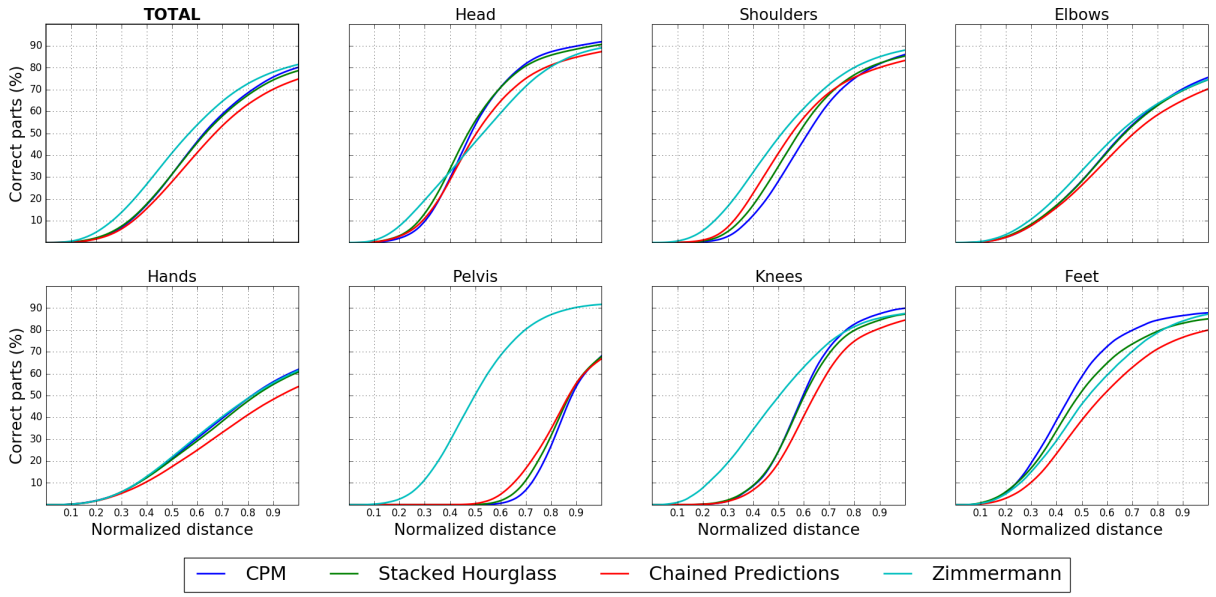


Figure 4.9: Quantitative 3D pose estimation results on BMHAD per joint, using as figure of merit PCKh 3D @ 0-1 (%).

average error in terms of distance was higher than the one yielded by Zimmermann's method, we manage to get very similar or even better percentages of correct parts, which means that joints estimated with broad error were properly identified and corrected. In Figure 4.11, some examples of such cases are presented.

Regarding the wide difference in terms of pelvis estimation, it is justified by the nature of our method. While Zimmermann's method is trained with labeled 3D joint locations, ours rely on no more than a stream of depth images, from a single point of view, to get the 3D locations from the 2D estimations. With that approach, we cannot *see through* the human body and the only information that we have is the closest point to the camera in the scene when we trace a ray between the sensor and the 2D estimation. Figure 4.12 exemplifies how this effect might have an impact on our estimations. In that regard, the joint for which this assumption is less precise is the pelvis, as it is placed in a region of the human body thicker than the other keypoints considered, and depending on how the subject is placed on the scene, this thickness changes. In general, our algorithm is more prone to failure if the subject is self-occluded, as it can be see in Figure 4.13. A possible solution that might be explored in the future in order to correct this kind of deviations is simply including in our estimations the average thickness of each human body part detected.

The same imbalance in PCKh and MPJPE results mentioned in the per joint analysis

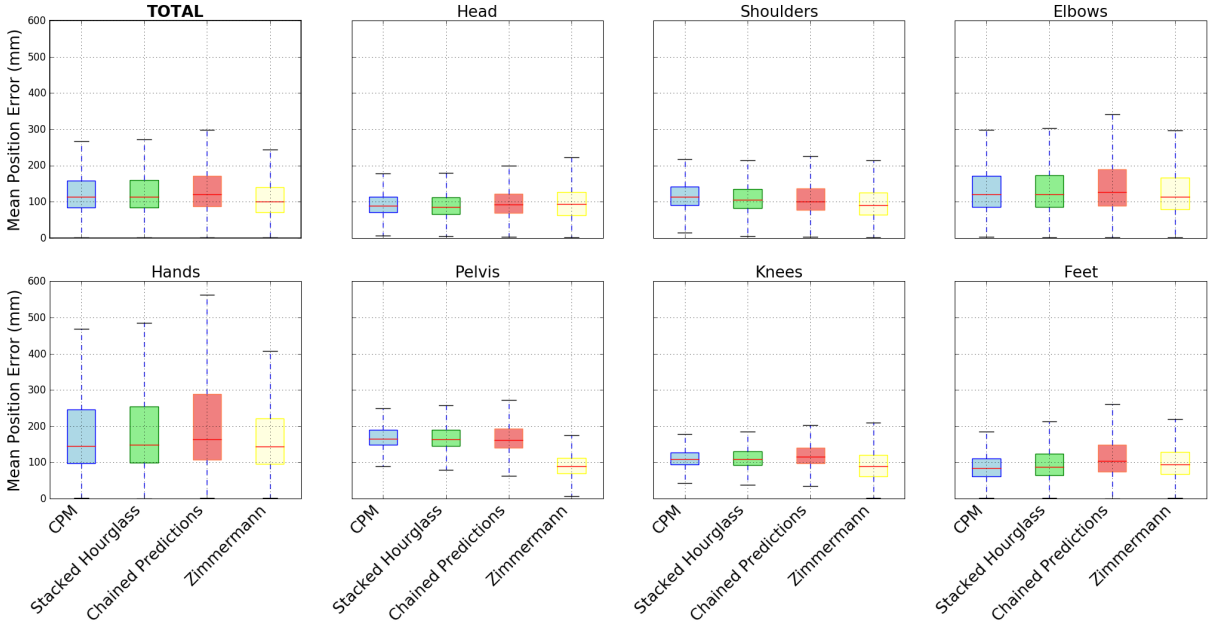


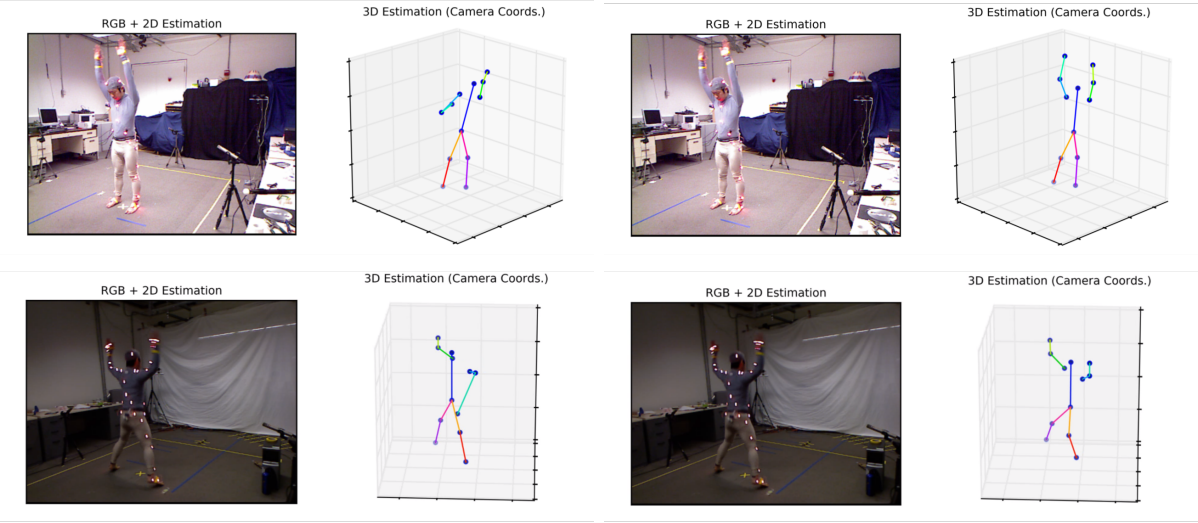
Figure 4.10: Quantitative 3D pose estimation results on BMHAD per joint, using as figure of merit the *Euclidean* distance between estimations and groun-truth (mm).

can be seen when evaluating per action. While our method performs better in three out of eleven actions in terms of PCKh, it only outperforms Zimmermann's in one of the scenes when evaluating MPJPE (see Table 4.7 and Table 4.8).

	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10	a11	TOTAL
CPMs	124.3	123.8	**159.7**	156.1	117.5	116.9	139.2	135.8	173.6	178.0	166.1	144.1
SH	123.7	119.8	163.1	158.1	122.6	128.9	141.8	146.4	179.7	186.5	171.6	148.7
CPs	138.6	150.5	181.0	172.5	146.6	146.4	155.2	171.0	185.9	206.0	175.7	164.8
Zimm.	**105.8**	**114.5**	179.4	**120.6**	**115.9**	**102.7**	**116.0**	**118.0**	**137.4**	**157.6**	**133.7**	**128.1**

Table 4.7: Quantitative 3D pose estimation results on BMHAD per action, using as figure of merit MPJPE (mm). Values in bold correspond to the best results achieved for each category. Columns are colored following the convention established in Section 4.1.

If we compare results after grouping the actions in terms of dynamics and occlusion, it is shown that our method performs better in the most challenging scenarios, *i.e.*, very dynamic actions with occluded joints. This observation holds true for both PCKh in Table 4.9 and MPJPE in Table 4.10. These differences can be caused again by our outlier rejection and Kalman filtering stages, which reduce the number of greatly deviated estimations in an explicit manner. Meanwhile, Zimmermann's method does not apply



(a) Outliers found in Zimmermann's estimations.

(b) Our results for the same frames, using CPMs as 2D estimator.

Figure 4.11: Our outlier rejection and filtering mechanisms help alleviate strong deviations like the ones shown in (a). By doing so, even if the overall error in favorable conditions is higher, we get plausible poses consistently (b).

	a01	a02	a03	a04	a05	a06	a07	a08	a09	a10	a11	TOTAL
CPMs	89.48	91.16	**71.25**	78.15	**92.94**	91.57	84.67	84.73	70.54	67.69	73.03	80.23
SH	89.39	**91.99**	65.69	77.54	92.12	90.08	83.33	82.10	70.36	67.32	72.42	78.70
CPs	86.29	85.18	60.46	75.21	85.60	84.45	81.59	79.21	69.00	64.52	71.12	74.88
Zimm.	**94.15**	90.52	55.91	**87.83**	89.25	**93.07**	**89.06**	**88.57**	**81.97**	**73.02**	**80.78**	**81.55**

Table 4.8: Quantitative 3D pose estimation results on BMHAD per action, using as figure of merit PCKh 3D @ 1 (%). Values in bold correspond to the best results achieved for each category. Columns are colored following the convention established in Section 4.1.

any post-processing on the yielded estimations and so great errors might appear when, for instance, a joint gets out of view or the subject is moving too fast.

An interesting lesson from the results we have just discussed is that our method for filtering 3D estimations could easily be plugged in after another 3D human pose estimation method such as Zimmermann's, further improving their results. By doing so, we would increase the computational cost, which is a major requirement in this work, but whether this could be a reasonable solution or not depends on the application.

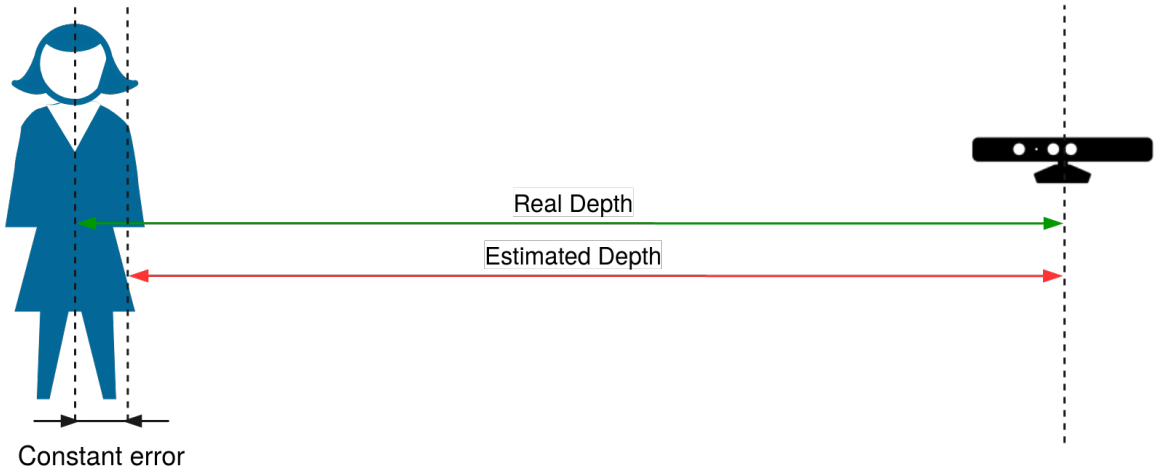


Figure 4.12: As our 3D estimations rely on depth images captured from a single point of view and no prior about the human body is imposed, we might commit a constant error when detecting joints that are not labeled close to the body surface.

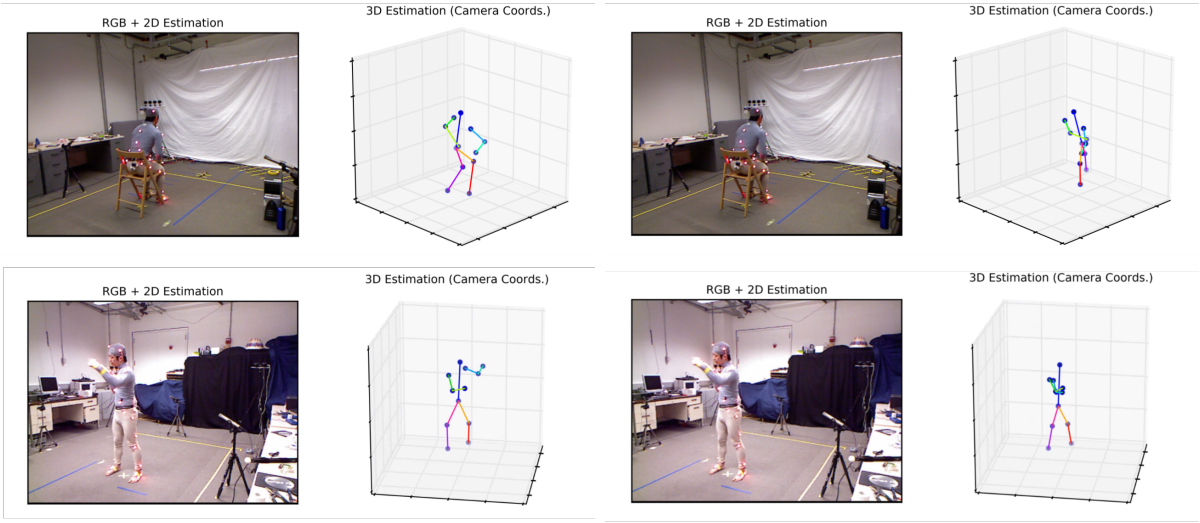
	High Dynamics		Low Dynamics	
	Heavily Occluded	Slightly Occluded	Heavily Occluded	Slightly Occluded
CPMs	**77.99**	86.26	70.42	89.73
SH	73.90	86.31	70.03	88.51
CPs	69.83	82.22	68.21	83.88
Zimm.	72.24	**90.83**	**78.59**	**90.46**

Table 4.9: Quantitative 3D pose estimation results on BMHAD per action type, using as figure of merit PCKh 3D @ 1 (%). Values in bold correspond to the best results achieved for each category. Columns are colored following the convention established in Section 4.1.

4.4 Computational burden

Besides achieving very competitive quantitative results when compared with the SOTA method introduced by Zimmermann *et al.* [73], we keep the computational costs as low as possible by using very lightweight algorithms for the 2D to 3D augmentation of the estimated coordinates. In Table 4.11, a comparison of the computational burden associated with each evaluated method in previous sections is presented. All the tests have been performed after evaluating 2108 frames of 10 different scenes extracted from BMHAD, using a PC with the following specifications:

- **RAM memory**: 12Gb.



(a) Zimmermann's estimation.

(b) Our estimation (using CPMs).

Figure 4.13: While Zimmerman's method (a) is able to deal with self-occlusions in the scene, our algorithm (b) relies only on the depth information without any learned prior on the human pose configuration, which causes estimation errors under these circumstances.

	High Dynamics		Low Dynamics	
	Heavily Occluded	Slightly Occluded	Heavily Occluded	Slightly Occluded
CPMs	**147.8**	134.7	172.6	124.5
SH	154.8	133.9	179.3	131.1
CPs	176.0	153.9	189.2	149.4
Zimm.	148.7	**113.6**	**142.9**	**111.6**

Table 4.10: Quantitative 3D pose estimation results on BMHAD per action type, using as figure of merit MPJPE (mm). Values in bold correspond to the best results achieved for each category. Columns are colored following the convention established in Section 4.1.

- **GPU**: Nvidia GeForce GTX 1050.
- **CPU**: Intel Core i7-7700HQ @ 2.80GHz

As it can be seen, our methods perform with much higher framerates, which facilitates their inclusion in robotic systems as embedded software. For instance, our method is around four times faster than Zimmermann's when using CPMs as 2D estimator, and more than seven times faster when using SH. When measuring the time that it takes to

	CPMs	SH	CPs	Zimm.	Our 2D to 3D registration module alone
fps	8.11	**16.07**	10.64	2.22	**99.76**

Table 4.11: Computational cost of each method, measured as the average number of frames per second.

carry out the 2D to 3D step, without taking into account the 2D estimation process, our algorithm only adds 0.01s to the total computation time for a single frame.

It is also worth mentioning that the computational burden can be further reduced depending on the boxsize of the input image. Depending on the accuracy required by the application, an optimal trade-off between the quality of the estimation and the computational cost may be explored.

4.5 Qualitative evaluation: real-time demo

We have already discussed how our method compares with the SOTA, both in terms of accuracy and computational burden. Our approach cannot outperform the most recent works in the literature in 3D human pose estimation, but it is competitive and much faster. However, the tests carried out have not assessed its performance with more common scenes *in the wild*. For that purpose, we have developed a real-time demo that works in an *off-the-shelf* computer with a commercial RGBD sensor. The set-up used and the system designed for that demo are presented in this section, along with a discussion about the obtained results.

In Figure 4.14, the simple set-up built for this demo is presented. Our demo has been run on a PC with the specifications described in Section 4.11. This PC is equipped with an Nvidia GPU [89], which allows the 2D estimator to make use of the CUDA parallel computing platform [90] for accelerating the inference of the human poses. The RGBD sensor used for the demo is an Asus Xtion PRO LIVE [19]. This camera captures RGBD video sequences at 30fps with a resolution of 640x480 pixels and a working distance that ranges between 0.5 and 3.5 meters. It relies on an infrared sensor and structured light projected on the scene for estimating depth.

Regarding the software used for the demo, our system employs ROS [91] for PC-camera communication. In particular, we have used the *openni2* node [92], which allows depth

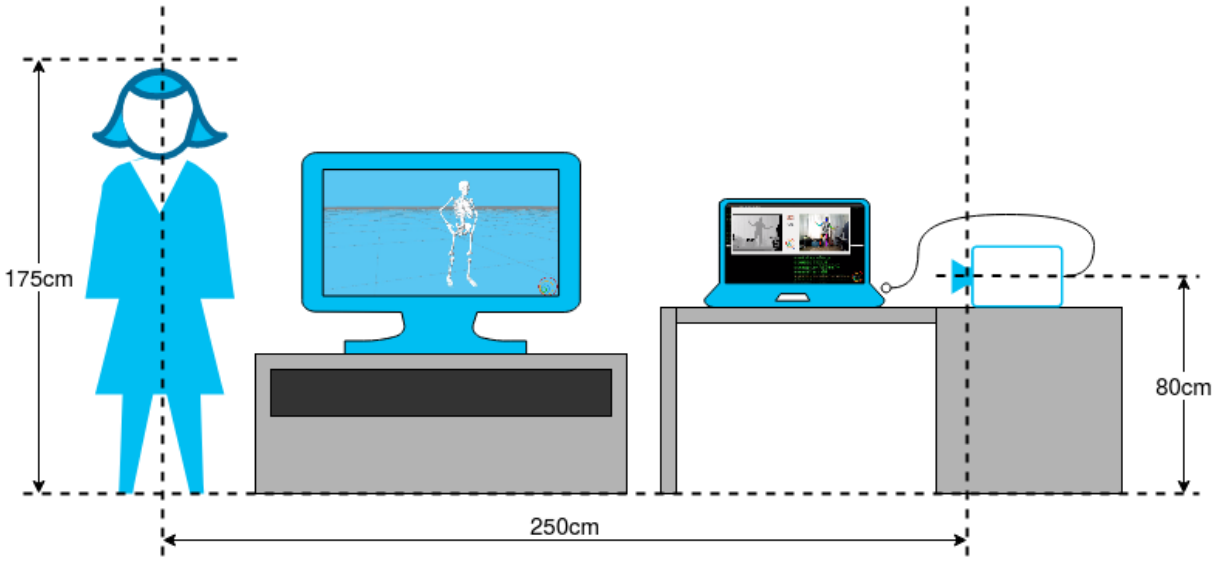
and RGBD image registration and distortion correction, based on our camera calibration data. Our demo application has been entirely developed in *Python* in a threaded manner. A thread is in charge of receiving and serving the color and depth images provided by the ROS node. Another thread updates a GUI that shows in real time these images and the result of the 2D pose estimation. Last but not least, a third thread is in charge of estimating the 2D poses and, from these, generating the final 3D estimations as described in Section 3.3. These 3D estimations can then be inspected in real-time thanks to the 3D visualizer gently shared by Roberto Pérez [93]. This visualizer has been built as a desktop app developed with Electron [94]. It is important to note that, for this demo, we have used CPMs as 2D estimator. In order to further improve the framerate we have used a boxsize of 256x256, which yields good enough results. A complete diagram of the developed software can be seen in Figure 4.15. The developed application is publicly available and hosted in *GitHub* [1].

Results for this demo are in Figure 4.16. The application runs fluently in real-time and, in general, achieves good results for the tested poses. It is worth mentioning that we identify failures in certain situations, *e.g.* when the subject is too close to the camera or stands sidewards, which increases the number of occluded joints. Such cases can be seen in Figure 4.17. A complete video showcasing the described demo is publicly available on *YouTube* [2].

[1] `https://github.com/RoboticsLabURJC/2017-tfm-david-pascual/`

[2] `https://www.youtube.com/watch?v=W3XirsadmNg`

(a) Image from the real set-up.



(b) Simplified diagram with relevant distances.

Figure 4.14: Diagram of the set-up used for the real-time demo.

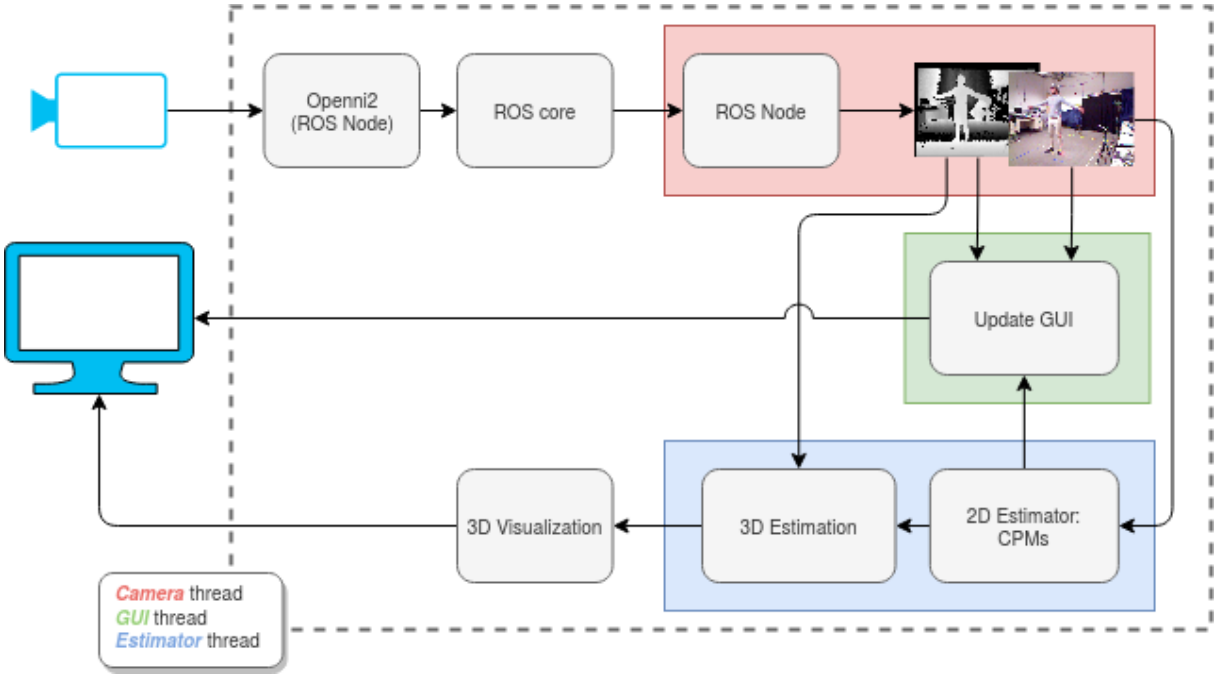
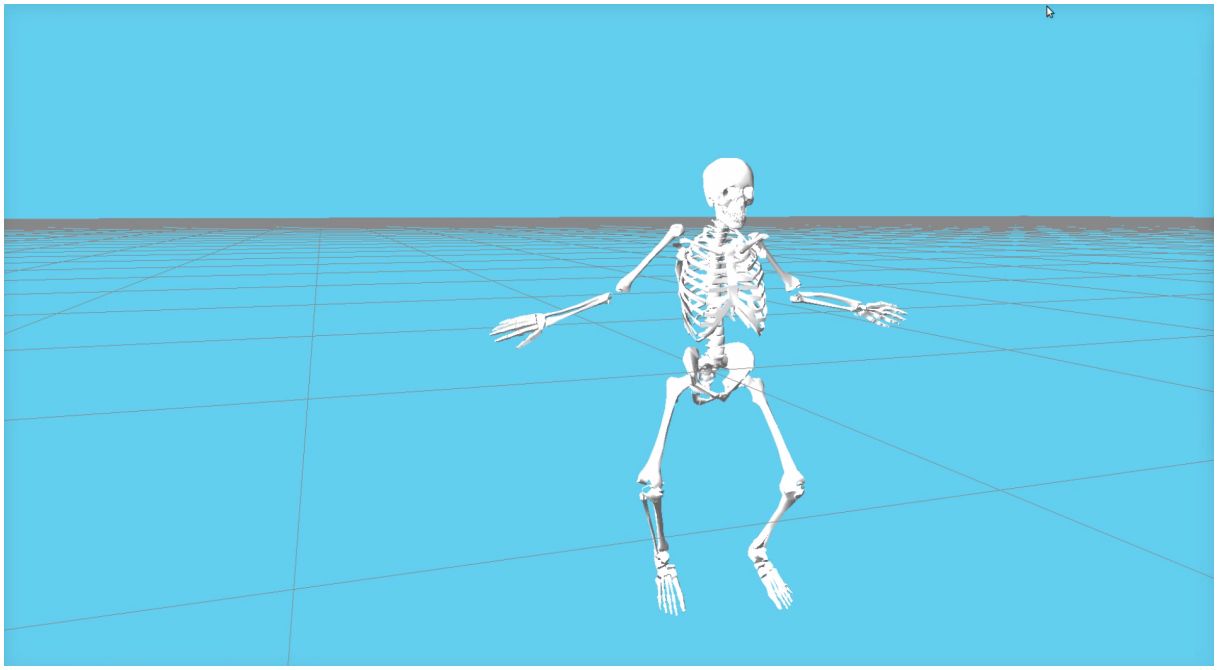
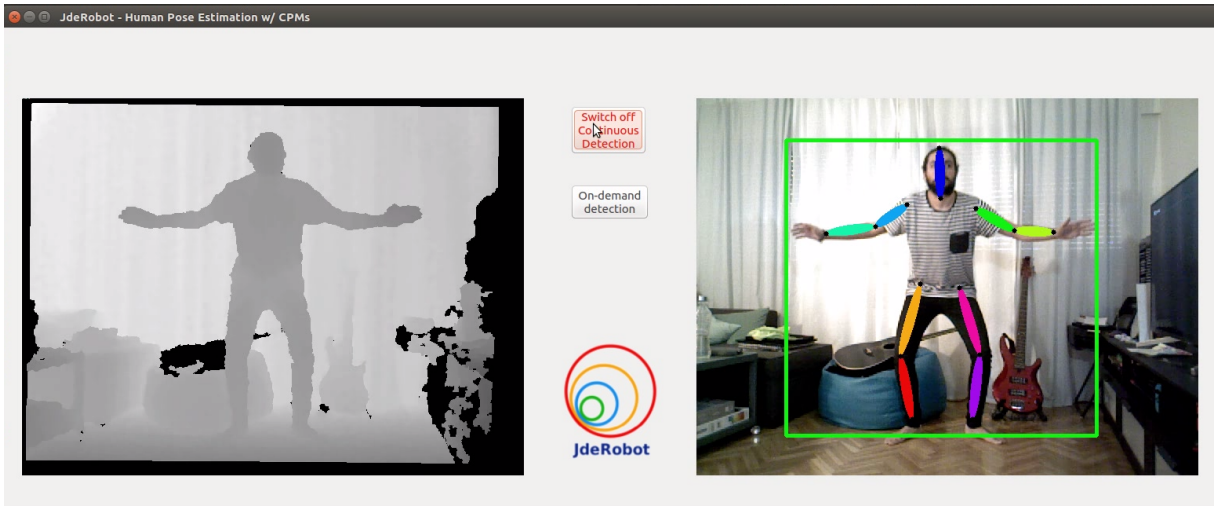


Figure 4.15: Overview of the application developed.

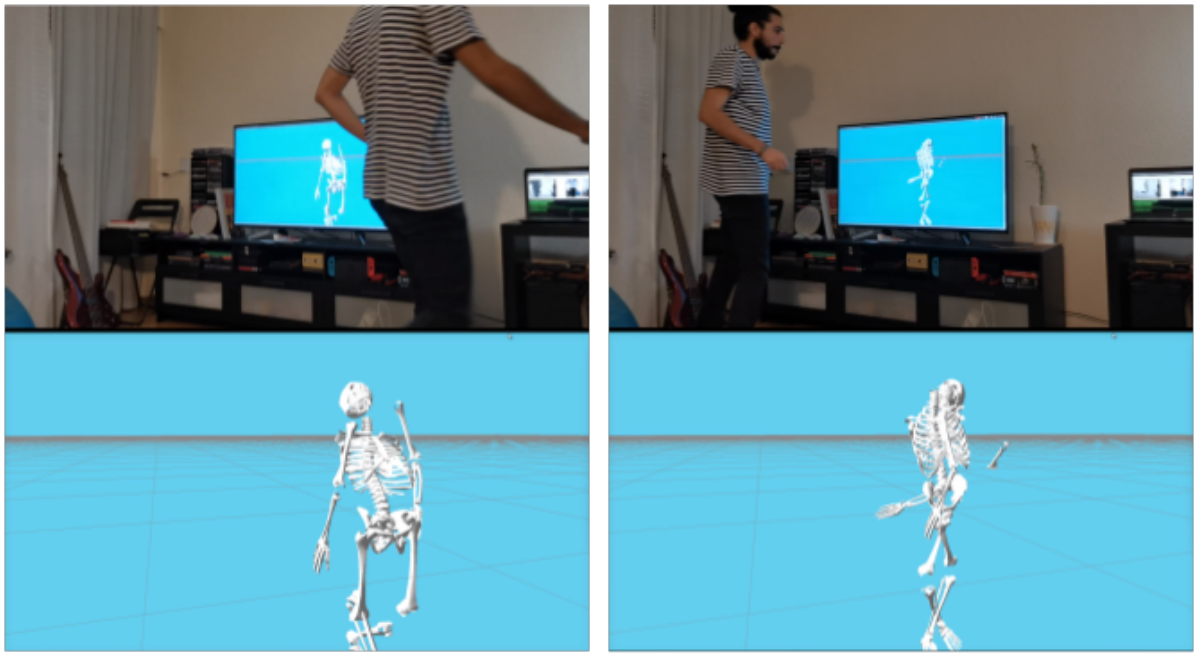


(a) 3D estimation results.



(b) 2D estimation results.

Figure 4.16: Real-time results as presented by our application for (a) 3D pose estimation and (b) 2D estimation.



(a) Estimation fails due to subject proximity.

(b) Estimation in auto-occlusion.

Figure 4.17: Our algorithm might fail in challenging scenarios, such as (a) the subject being too close to the camera or (b) self-occlusions.

Chapter 5

Conclusions and future lines

Taking into consideration the results presented in Chapter 4, we are now able to draw conclusions and discuss whether the goals established in Section 1.2 can be considered fulfilled or not. For ease of reading, we will articulate our conclusions following the structure of the aforementioned goals, pointing out our main contributions and the challenges that we have found along the way. Finally, we will conclude our report looking ahead for potential improvements of the system we have designed, as well as proposing new applications and related research topics that are yet to be explored.

5.1 Main conclusions

Quantitative evaluation of 2D human pose estimation methods

The proposed method relies on an accurate 2D human pose estimation from which to infer the corresponding three-dimensional locations of the joints. Taking this requirement into consideration, we needed a robust 2D pose estimator that could serve as a backbone for the system we were designing. Here are our conclusions in that regard:

- A thorough review of the literature has been done in order to found those solutions that better fit our requirements. Human pose estimation is a very active area within the field of computer vision, and so there is large number of available publications that we have explored and analyzed.

- After this first stage of exploration, we have selected three SOTA methods: *Convolutional Pose Machines* [37], *Stacked Hourglass* [36] and *Chained Predictions* [39].

All of them share some characteristics that made them appropriate for our purposes: they are relevant works in the field, their code is publicly available and they claim to reach a good trade-off between performance and computational cost.

- In order to validate the results provided in their respective articles, we have performed an exhaustive quantitative evaluation and comparison of the three methods mentioned above. This evaluation has been carried out using an international publicly available dataset (BMHAD) that contains more than 1300 scenes, which shows different subjects performing a variety of actions. Overall, *Convolutional Pose Machines* have achieved better results than *Stacked Hourglass* and *Chained Predictions*, and so it is our go-to estimator for the designed system. Besides that, the difference between *Convolutional Pose Machines* and the other methods is specially significant for the most challenging scenes in the dataset.

- After validating their performance, the three tested estimators have been successfully integrated in our pipeline. Looking back at the designed system for the real-time demo in Section 4.5, the user can choose between any of these methods for 2D estimation before launching the application by simply updating a configuration file.

We can conclude that suitable solutions for 2D pose estimation have been researched, tested and, finally, integrated in our system, accomplishing the goal established in Section 1.2. It is important to note that this field of research is evolving at an impressive pace, and so our method may fall short in comparison with most recent works if this core component of our application does not get regularly updated.

Development of a pipeline for augmenting 2D to 3D pose estimations

The main focus of this project has been the development of a system that, taking as input an RGBD video sequence, estimates 3D human poses keeping a fair trade-off between accuracy and computational cost. We have already talked about the first step in that process, which is estimating the 2D joint locations in the RGB image. In order to take these 2D estimations to 3D using the information provided by the depth image, a lightweight algorithm was required. Here are our conclusions in that regard:

- In order to understand what kind of processing our algorithm needed to perform on the input data, we have explored the main issues that arise when trying to infer 3D

estimations from the 2D locations and the depth image without any intermediate step.

- After exploring these results, we have looked for solutions that were both effective and lightweight. Taking these considerations into account, we have developed a very straight-forward algorithm that allows us to estimate the 3D poses with respectable accuracy in real-time. Our approach pre-processes the input depth map and 2D coordinates by means of a *Gaussian* blur and a minima filter, reprojects the result to 3D, rejects outliers and smooths the resulting estimations using a *Kalman* filter for each joint.

- We have implemented our system in *Python* using common libraries for image and data processing. Our source code has been made publicly available and it is hosted in a *GitHub* repository [1].

In summary, we have developed an agile algorithm that is able to yield 3D human pose estimations given its corresponding 2D joint locations and a depth map. Our pipeline takes a video sequence from an RGBD camera and, making use of the 2D estimation methods previously discussed, returns three-dimensional poses for the subject in the scene in real-time. In that sense, we have fulfilled the goal established in Section 1.2. Nonetheless, further improvements can be explored, *e.g.* our mechanisms for filtering the estimations could be more adaptive and some processes performed independently for each joint could be parallelized.

Quantitative evaluation and comparison of our proposed method

In order to experimentally validate the conclusions reached in the paragraph above, an exhaustive quantitative evaluation was needed. Furthermore, it is important to put our work in context and check how the achieved results compare with the SOTA. Again, one major requirement of this project is keeping the computational cost as low as possible, and so this is a topic that also required analysis. Here are our conclusions regarding the quantitative evaluation of our work:

- A comparison between the proposed method and a simple baseline has been performed. For that purpose, we define as baseline the direct inference of the 3D

[1] `https://github.com/RoboticsLabURJC/2017-tfm-david-pascual`

coordinates given the 2D locations and the depth map, without further processing. Using a subset of the BMHAD dataset (132 scenes) and using MPJPE as figure of merit, we have validated that the inclusion of the proposed method improves significantly the final estimations.

- Further analysis of this comparison has demonstrated how our processing is specially beneficial when 2D estimations are slightly misplaced, as this can result in huge differences in 3D. It is worth mentioning that, if the 2D location is accurate and there is no self-occlusions or other kind of interference, we find no significant difference in the results yielded by our method and the baseline.

- After assessing the improvements introduced with respect to the baseline, we have analyzed how the measured error correlates with the dimensions of the three-dimensional space. We have discovered that most of the committed error comes from the Z dimension, *i.e.*, the distance between the camera and the subject. As mentioned before, slight deviations in the 2D estimation might also cause great differences in 3D.

- A comparison with a SOTA method has been performed. A large portion of the amount of effort put into this work has been dedicated to this comparison. We have thoroughly evaluated our algorithm against the solution proposed by Zimmermann *et al.* [73] in order to assess whether the designed system is good enough to justify its application in real scenarios. For that purpose, we have used the BMHAD dataset in its entirety. PCKh and MPJPE have been employed to measure performance. All of the results have been presented grouped by joints and by actions and action types.

- This experimental comparison has allowed to reach several relevant conclusions. First of all, our method is very competitive when compared with the SOTA, but we cannot outperform it. Surprisingly, our method performed better for some of the most heavily occluded scenes and yielded a very respectable PCKh. We attribute these results mainly to our explicit outlier rejection mechanism, which avoids committing huge errors even if the scene is very challenging. However, under normal circumstances, Zimmermann's algorithm deals better with self-occlusions as their model implicitly learns priors about how a human body is supposed to appear.

- Last but not least, we have analyzed the computational burden associated with our system. We demonstrate that our method is significantly faster than Zimmermann's. Our module for 3D estimation alone takes about 10ms per frame, fulfilling the major requirement of keeping the computation costs low. These small times per frame means that the final estimation time are highly dependant on the 2D estimator used.

Taking into consideration all of the conclusions drawn above, we can consider the goal defined in Section 1.2 accomplished. We have carried out a very exhaustive evaluation of our method against a baseline and a SOTA method, both in terms of accuracy and computational cost. With these results, our system seems to fulfill the major requirements we impose for it to work as component in robotics applications, reaching a reasonable trade-off between its agility and performance.

Qualitative evaluation of our proposed method: real-time demo

The resulting product of all of the aforementioned research in the previous section is a real-time demo which has allowed us to validate how our system might work in a real environment. This is a major milestone as it puts the theoretical knowledge acquired during this project into practice. Here are our conclusions in that regard:

- A working real-time application for 3D human pose estimation has been developed. This application uses an *off-the-shelf* computer and a commercial RGBD sensor, and integrates the method described in Chapter 3 within an environment built in *Python* that handles communication with the sensor, estimation and visualization in a threaded manner. Code and instructions for running the application are publicly available in *GitHub* [2].

- A demo has been carried out to assess the feasibility of our system. The results have been recorded and the resulting video is publicly available [3]. For this demonstration, we built a very simple setup comprised of a common PC and an *Asus XTION PRO LIVE* RGBD camera. A subject placed in front of the camera interacts with the application in real-time successfully.

[2] `https://github.com/RoboticsLabURJC/2017-tfm-david-pascual`

[3] `https://www.youtube.com/watch?v=W3XirsadmNg`

- During the recording of this demo, we also documented failure cases, being the most significant self-occlusions and excessive proximity to the camera. These results match the conclusions drawn during the quantitative evaluation of our method.

Recalling the requirements imposed for an algorithm to be embedded in a robotic system, we pointed out that it must work in real-time in an *off-the-shelf* computer, with a regular RGBD camera and independently of the point of view. Taking these into account, we can consider our major goals fulfilled as demonstrated by our real-time demo. The weakest point of the proposed method might be the point of view of the camera, not because of the position of the camera itself, but due to the possibility of increasing the number of occlusions, which our algorithm struggles to handle. Nonetheless, we have reached our major milestone of building an application that runs in real-time and estimates 3D human poses with an average error of 144mm, which is a very reasonable trade-off that would justify its inclusion as a core component in robotics applications.

5.2 Future work

Thanks to the knowledge acquired during the development of this project and given that we have built a stable infrastructure for the task of 3D pose estimation, a great amount of potential applications and improvements are on the table. In this section, we will discuss some of them, leaving an open door for further research on the topic.

- Probably the most evident improvement we can make to our pipeline is regularly updating the model used for estimating the 2D locations of the joints. This is possibly a never-ending task, as new works claiming to achieve better results arise every year. However, it seems like the magnitude of these incremental improvements is starting to plateau, as it can be seen in Figure 5.1. Nonetheless, it is worth reviewing the literature regularly and updating our system consequently.

- Another possible improvement regarding 2D estimation is further reducing the inference time. As accuracy starts to peak, efficiency should become one of the main priorities for new works in the field. For instance, in the object detection field very efficient architectures have been explicitly developed for being embedded in systems with scarce resources [75]. Such is the case of the very recently introduced *EfficientPose* [96], which claims to be 20 times more computationally efficient than

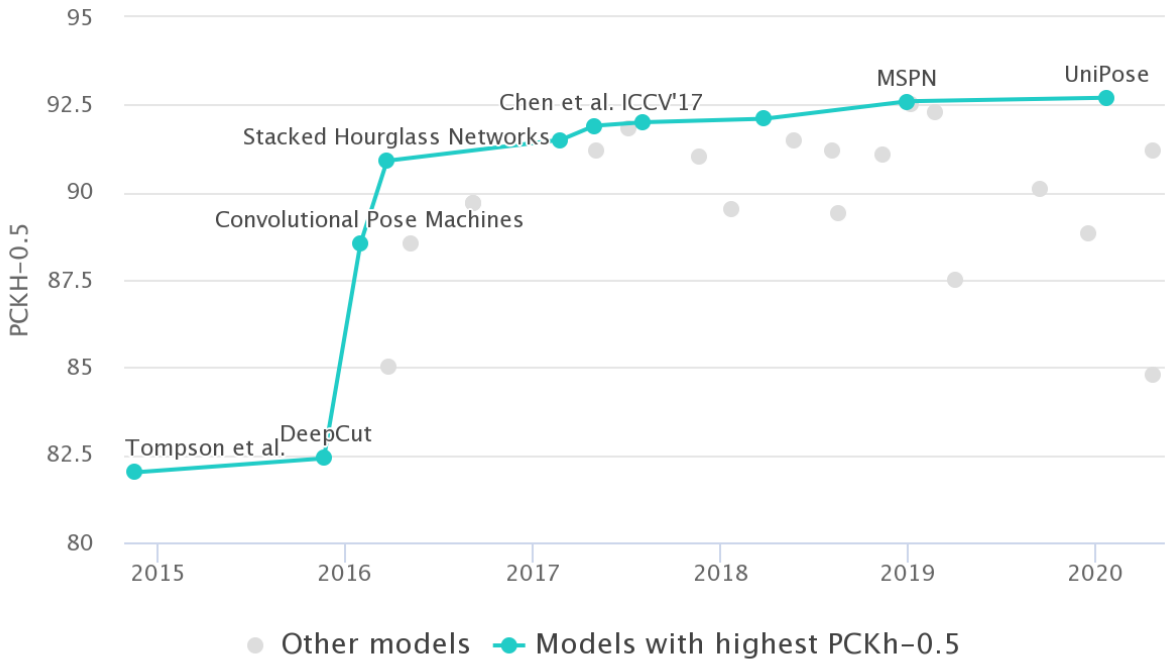


Figure 5.1: In this graph, publicly available methods for 2D pose estimation presented in the last years evaluated against the MPII dataset are shown (source [95]). Note that the methods we tested (CPMs and SH) were a major breakthrough in the field. Since then, new methods have slightly outperformed them, but the improvements are starting to plateau.

OpenPose [13] while outperforming it. Including this kind of new approaches could be a major improvement for our application which could then be integrated in cheaper devices.

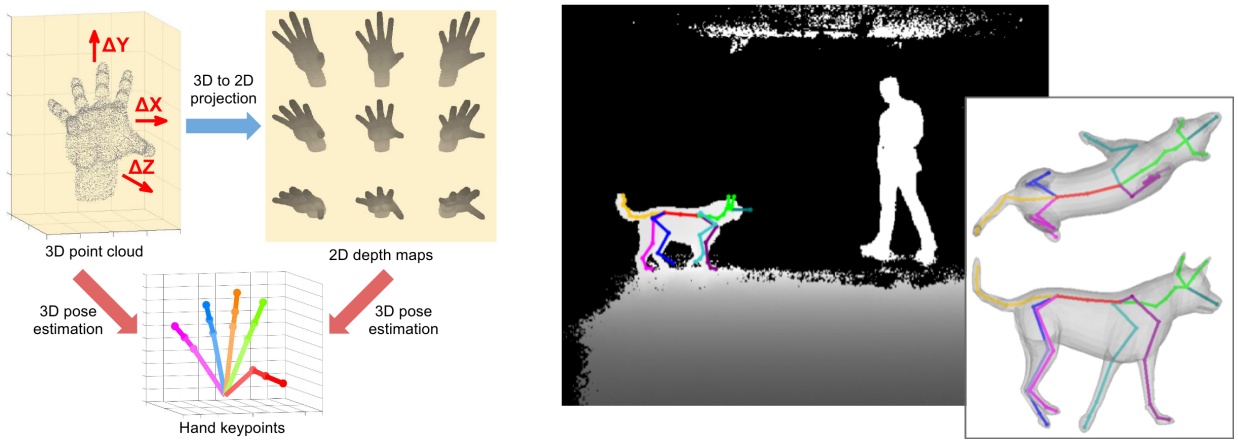
- Multi-person pose estimation has been left unexplored in this project, but it could be an important requirement depending on the particular application. To that extent, we presented some works focusing on multi-person scenarios in Chapter 2, being the most prominent the one proposed by Cao *et al.* [13], which builds upon CPMs and is the core of the *OpenPose* suite. Evaluating how this method performs and analyzing whether its computational cost is low enough for robotics applications might be a natural extension of our work.

- Yet another improvement that could be considered for our current pipeline is including stronger hints of how a human body should look like to further improve our results. In other words, introducing priors for the human poses could allow us to

manage complex scenearios with self-occlusions and so on. DL models such as the one proposed by Zimmermann *et al.* [73] are too computationally intensive for our approach, but we could consider fitting simpler human body models or researching more lightweight DL architectures for that purpose.

- Regarding the evaluation stage of our project, it would be desirable to include more natural *in-the-wild* scenes with more points of view to further validate how our method and SOTA solutions perform in the real world. In that sense, finding or building a dataset containing RGBD data and accurate labels for the 3D poses would be needed.

- One possibility that we have not yet mentioned is using our approach for getting 3D coordinates from 2D pose estimation data from other articulated objects or subjects that might not necessarily be humans, after updating our outlier rejection criteria. For instance, it could be used for estimating hand poses [72] or poses in animals [97], which could open the doors for very interesting applications. In Figure 5.2, examples of such approaches can be seen.



(a) 3D hand pose estimation [72]

(b) 3D dog pose estimation [97]

Figure 5.2: Our algorithm could be adapted for estimating 3D poses for other articulated objects or subjects, such as (a) hands or (b) animals.

- Finally, the most obvious follow-up for our work is integrating our approach, as demonstrated in Section 4.5, in a real robotic application. 3D human pose estimation can be included as a core component for solving tasks such as fall detection or action recognition of any kind, which is very useful information for any human-

robot interaction. In that sense, our algorithm could be applied in fields such as assistive robotics, for which human-awareness is a major requirement.

In summary, our work can be considered in itself the basis from which bigger embedded applications can be built upon. Further improvements and evaluations may be considered in order to make our method more effective and stable, and more robust in challenging scenarios. Besides these improvements, it can be extended for estimating poses in multi-person scenarios or supporting other articulated objects. Integrating our system in a real-world robotic application, with or without these extensions and improvements, is the next major milestone for this project.

Bibliography

[1] Klaus Schwab. *The fourth industrial revolution*. New York: Crown Business, 2016. ISBN: 1524758868.

[2] Michael A. Goodrich and Alan C. Schultz. "Human-robot interaction: a survey". In: *Foundations and Trends in Human-Computer Interaction* 1.3 (2007), pp. 203–275. DOI: 10.1561/1100000005. URL: https://doi.org/10.1561/1100000005.

[3] ARCLab. *Surgical robotics seminar with UCSD's Michael Yip*. 2019. URL: https://robotics.utoronto.ca/news/surgical-robotics-seminar-with-ucsds-michael-yip/ (visited on 10/09/2020).

[4] Travelarz. *Humanoid robot*. 2017. URL: https://commons.wikimedia.org/wiki/File:Humanoid_Robot_(2)_ITB_2017.JPG (visited on 10/09/2020).

[5] Alessandro Roncone, Olivier Mangin, and Brian Scassellati. "Transparent role assignment and task allocation in human robot collaboration". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2017. DOI: 10.1109/icra.2017.7989122. URL: https://doi.org/10.1109/icra.2017.7989122.

[6] Marco Leo et al. "Deep learning for assistive computer vision". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2019, pp. 3–14. DOI: 10.1007/978-3-030-11024-6_1. URL: https://doi.org/10.1007/978-3-030-11024-6_1.

[7] Vicon. *The Vicon difference: motion capture systems*. 2020. URL: https://www.vicon.com/about-us/ (visited on 10/09/2020).

[8] Alessandro Leone, Giovanni Diraco, and Pietro Siciliano. "Detecting falls with 3D range camera in ambient assisted living applications: A preliminary study". In: *Medical Engineering & Physics* 33.6 (2011), pp. 770–781. DOI: 10.1016/j.medengphy.2011.02.001. URL: https://doi.org/10.1016/j.medengphy.2011.02.001.

[9] OptiTrack. *Optitrack studio*. 2019. URL: https://optitrack.com/public/images/prime41ActivisionVolume04.jpg (visited on 10/09/2020).

[10] Ferda Ofli et al. “Berkeley MHAD: a comprehensive multimodal human action database”. In: *2013 IEEE Workshop on Applications of Computer Vision (WACV)*. IEEE, 2013. DOI: 10.1109/wacv.2013.6474999. URL: https://doi.org/10.1109/wacv.2013.6474999.

[11] Andrew Ng. “Nuts and bolts of building AI applications using deep learning”. In: *NIPS Keynote Talk*. Curran Associates Inc., 2016.

[12] Weibo Liu et al. “A survey of deep neural network architectures and their applications”. In: *Neurocomputing* 234 (2017), pp. 11–26. DOI: 10.1016/j.neucom.2016.12.038. URL: https://doi.org/10.1016/j.neucom.2016.12.038.

[13] Zhe Cao et al. “OpenPose: realtime multi-person 2D pose estimation using part affinity fields”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019), pp. 1–1. DOI: 10.1109/tpami.2019.2929257. URL: https://doi.org/10.1109/tpami.2019.2929257.

[14] Catalin Ionescu et al. “Human3.6M: large scale datasets and predictive methods for 3D human sensing in natural environments”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (2014), pp. 1325–1339. DOI: 10.1109/tpami.2013.248. URL: https://doi.org/10.1109/tpami.2013.248.

[15] 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”. In: *International Journal of Computer Vision* 87.1-2 (2009), pp. 4–27. DOI: 10.1007/s11263-009-0273-6. URL: https://doi.org/10.1007/s11263-009-0273-6.

[16] Mengyuan Liu and Junsong Yuan. “Recognizing human actions as the evolution of pose estimation maps”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018. DOI: 10.1109/cvpr.2018.00127. URL: https://doi.org/10.1109/cvpr.2018.00127.

[17] Nikolaos Sarafianos et al. “3D Human pose estimation: a review of the literature and analysis of covariates”. In: *Computer Vision and Image Understanding* 152 (2016), pp. 1–20. DOI: 10.1016/j.cviu.2016.09.002. URL: https://doi.org/10.1016/j.cviu.2016.09.002.

[18] Zhengyou Zhang. "Microsoft Kinect sensor and its effect". In: *IEEE Multimedia* 19.2 (2012), pp. 4–10. DOI: 10.1109/mmul.2012.24. URL: https://doi.org/10.1109/mmul.2012.24.

[19] ASUSTeK Computer Inc. *Xtion PRO LIVE: 3D sensor*. 2020. URL: https://www.asus.com/3D-Sensor/Xtion_PRO_LIVE/ (visited on 10/09/2020).

[20] C. Larman and V.R. Basili. "Iterative and incremental developments: a brief history". In: *Computer* 36.6 (2003), pp. 47–56. DOI: 10.1109/mc.2003.1204375. URL: https://doi.org/10.1109/mc.2003.1204375.

[21] Wenjuan Gong et al. "Human pose estimation from monocular images: a comprehensive survey". In: *Sensors* 16.12 (2016), p. 1966. DOI: 10.3390/s16121966. URL: https://doi.org/10.3390/s16121966.

[22] M.A. Fischler and R.A. Elschlager. "The representation and matching of pictorial structures". In: *IEEE Transactions on Computers* C-22.1 (1973), pp. 67–92. DOI: 10.1109/t-c.1973.223602. URL: https://doi.org/10.1109/t-c.1973.223602.

[23] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. "Pictorial structures for object recognition". In: *International Journal of Computer Vision* 61.1 (2005), pp. 55–79. DOI: 10.1023/b:visi.0000042934.15159.49. URL: https://doi.org/10.1023/b:visi.0000042934.15159.49.

[24] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. "Pictorial structures revisited: People detection and articulated pose estimation". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009. DOI: 10.1109/cvpr.2009.5206754. URL: https://doi.org/10.1109/cvpr.2009.5206754.

[25] Yi Yang and Deva Ramanan. "Articulated pose estimation with flexible mixtures-of-parts". In: *CVPR 2011*. IEEE, 2011. DOI: 10.1109/cvpr.2011.5995741. URL: https://doi.org/10.1109/cvpr.2011.5995741.

[26] Benjamin Sapp, Chris Jordan, and Ben Taskar. "Adaptive pose priors for pictorial structures". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010. DOI: 10.1109/cvpr.2010.5540182. URL: https://doi.org/10.1109/cvpr.2010.5540182.

[27] Leonid Pishchulin et al. “Poselet conditioned pictorial structures”. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2013. DOI: 10.1109/cvpr.2013.82. URL: https://doi.org/10.1109/cvpr.2013.82.

[28] Martin Kiefel and Peter Vincent Gehler. “Human pose estimation with fields of parts”. In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 331–346. DOI: 10.1007/978-3-319-10602-1_22. URL: https://doi.org/10.1007/978-3-319-10602-1_22.

[29] Fang Wang and Yi Li. “Beyond physical connections: tree models in human pose estimation”. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2013. DOI: 10.1109/cvpr.2013.83. URL: https://doi.org/10.1109/cvpr.2013.83.

[30] Anoop Cherian et al. “Mixing body-part sequences for human pose estimation”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014. DOI: 10.1109/cvpr.2014.302. URL: https://doi.org/10.1109/cvpr.2014.302.

[31] Alireza Fathi and Greg Mori. “Human pose estimation using motion exemplars”. In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007. DOI: 10.1109/iccv.2007.4409073. URL: https://doi.org/10.1109/iccv.2007.4409073.

[32] Vittorio Ferrari, Manuel Marin-Jimenez, and Andrew Zisserman. “Progressive search space reduction for human pose estimation”. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008. DOI: 10.1109/cvpr.2008.4587468. URL: https://doi.org/10.1109/cvpr.2008.4587468.

[33] Dong Zhang and Mubarak Shah. “Human pose estimation in videos”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015. DOI: 10.1109/iccv.2015.233. URL: https://doi.org/10.1109/iccv.2015.233.

[34] Alexander Toshev and Christian Szegedy. “DeepPose: human pose estimation via deep neural networks”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014. DOI: 10.1109/cvpr.2014.214. URL: https://doi.org/10.1109/cvpr.2014.214.

[35] Jonathan J Tompson et al. "Joint training of a convolutional network and a graphical model for human pose estimation". In: *Advances in Neural Information Processing Systems 27*. Curran Associate, Inc., 2014, pp. 1799–1807. URL: https://papers.nips.cc/paper/5573-joint-training-of-a-convolutional-network-and-a-graphical-model-for-human-pose-estimation.

[36] Alejandro Newell, Kaiyu Yang, and Jia Deng. "Stacked hourglass networks for human pose estimation". In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 483–499. DOI: 10.1007/978-3-319-46484-8_29. URL: https://doi.org/10.1007/978-3-319-46484-8_29.

[37] Shih-En Wei et al. "Convolutional pose machines". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. DOI: 10.1109/cvpr.2016.511. URL: https://doi.org/10.1109/cvpr.2016.511.

[38] Varun Ramakrishna et al. "Pose machines: articulated pose estimation via inference machines". In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 33–47. DOI: 10.1007/978-3-319-10605-2_3. URL: https://doi.org/10.1007/978-3-319-10605-2_3.

[39] Georgia Gkioxari, Alexander Toshev, and Navdeep Jaitly. "Chained predictions using convolutional neural networks". In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 728–743. DOI: 10.1007/978-3-319-46493-0_44. URL: https://doi.org/10.1007/978-3-319-46493-0_44.

[40] Yu Chen et al. "Adversarial PoseNet: a structure-aware convolutional network for human pose estimation". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 1221–1230. DOI: 10.1109/ICCV.2017.137. URL: https://doi.org/10.1109/ICCV.2017.137.

[41] Arjun Jain et al. "MoDeep: a deep learning framework using motion features for human pose estimation". In: *Computer Vision – ACCV 2014*. Springer International Publishing, 2015, pp. 302–315. DOI: 10.1007/978-3-319-16808-1_21. URL: https://doi.org/10.1007/978-3-319-16808-1_21.

[42] Tomas Pfister, James Charles, and Zisserman Andrew. "Flowing convnets for human pose estimation in videos". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 1913–1921. DOI: 10.1109/ICCV.2015.222. URL: https://doi.org/10.1109/ICCV.2015.222.

[43] Jie Song et al. “Thin-slicing network: a deep structured model for pose estimation in videos”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. DOI: 10.1109/cvpr.2017.590. URL: https://doi.org/10.1109/cvpr.2017.590.

[44] Rohit Girdhar et al. “Detect-and-track: efficient pose estimation in videos”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018. DOI: 10.1109/cvpr.2018.00044. URL: https://doi.org/10.1109/cvpr.2018.00044.

[45] Kaiming He et al. “Mask R-CNN”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017. DOI: 10.1109/iccv.2017.322. URL: https://doi.org/10.1109/iccv.2017.322.

[46] Umar Iqbal, Anton Milan, and Juergen Gall. “PoseTrack: joint multi-person pose estimation and tracking”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. DOI: 10.1109/cvpr.2017.495. URL: https://doi.org/10.1109/cvpr.2017.495.

[47] George Papandreou et al. “Towards accurate multi-person pose estimation in the wild”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. DOI: 10.1109/cvpr.2017.395. URL: https://doi.org/10.1109/cvpr.2017.395.

[48] Eldar Insafutdinov et al. “ArtTrack: articulated multi-person tracking in the wild”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. DOI: 10.1109/cvpr.2017.142. URL: https://doi.org/10.1109/cvpr.2017.142.

[49] Riza Alp Guler, Natalia Neverova, and Iasonas Kokkinos. “DensePose: dense human pose estimation in the wild”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018. DOI: 10.1109/cvpr.2018.00762. URL: https://doi.org/10.1109/cvpr.2018.00762.

[50] Tomas Simon et al. “Hand keypoint detection in single images using multiview bootstrapping”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. DOI: 10.1109/cvpr.2017.494. URL: https://doi.org/10.1109/cvpr.2017.494.

[51] Hanbyul Joo et al. "Panoptic Studio: a massively multiview system for social interaction capture". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.1 (2019), pp. 190–204. DOI: 10.1109/tpami.2017.2782743. URL: https://doi.org/10.1109/tpami.2017.2782743.

[52] 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*. IEEE, 2010. DOI: 10.1109/cvpr.2010.5540156. URL: https://doi.org/10.1109/cvpr.2010.5540156.

[53] Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. "Reconstructing 3D human pose from 2D image landmarks". In: *Computer Vision – ECCV 2012*. Springer Berlin Heidelberg, 2012, pp. 573–586. DOI: 10.1007/978-3-642-33765-9_41. URL: https://doi.org/10.1007/978-3-642-33765-9_41.

[54] Ching-Hang Chen and Deva Ramanan. "3D human pose estimation = 2D pose estimation matching". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. DOI: 10.1109/cvpr.2017.610. URL: https://doi.org/10.1109/cvpr.2017.610.

[55] Federica Bogo et al. "Keep it SMPL: automatic estimation of 3D human pose and shape from a single image". In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 561–578. DOI: 10.1007/978-3-319-46454-1_34. URL: https://doi.org/10.1007/978-3-319-46454-1_34.

[56] Julieta Martinez et al. "A simple yet effective baseline for 3d human pose estimation". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017. DOI: 10.1109/iccv.2017.288. URL: https://doi.org/10.1109/iccv.2017.288.

[57] Denis Tome, Chris Russell, and Lourdes Agapito. "Lifting from the deep: convolutional 3D pose estimation from a single image". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. DOI: 10.1109/cvpr.2017.603. URL: https://doi.org/10.1109/cvpr.2017.603.

[58] Aiden Nibali et al. "3D human pose estimation with 2D marginal heatmaps". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019. DOI: 10.1109/wacv.2019.00162. URL: https://doi.org/10.1109/wacv.2019.00162.

[59] Diogo C. Luvizon, David Picard, and Hedi Tabia. "2D/3D pose estimation and action recognition using multitask deep learning". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018. DOI: 10.1109/cvpr.2018.00539. URL: https://doi.org/10.1109/cvpr.2018.00539.

[60] Georgios Pavlakos et al. "Coarse-to-fine volumetric prediction for single-image 3D human pose". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. DOI: 10.1109/cvpr.2017.139. URL: https://doi.org/10.1109/cvpr.2017.139.

[61] 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)*. IEEE, 2017. DOI: 10.1109/iccv.2017.373. URL: https://doi.org/10.1109/iccv.2017.373.

[62] Dushyant Mehta et al. "VNect". In: *ACM Transactions on Graphics* 36.4 (2017), pp. 1–14. DOI: 10.1145/3072959.3073596. URL: https://doi.org/10.1145/3072959.3073596.

[63] Iván Ramírez et al. "Bayesian capsule networks for 3D human pose estimation from single 2D images". In: *Neurocomputing* 379 (2020), pp. 64–73. DOI: 10.1016/j.neucom.2019.09.101. URL: https://doi.org/10.1016/j.neucom.2019.09.101.

[64] D.M. Gavrila and L.S. Davis. "3-D model-based tracking of humans in action: a multi-view approach". In: *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1996. DOI: 10.1109/cvpr.1996.517056. URL: https://doi.org/10.1109/cvpr.1996.517056.

[65] Juan Carlos Núñez et al. "Multiview 3D human pose estimation using improved least-squares and LSTM networks". In: *Neurocomputing* 323 (2019), pp. 335–343. DOI: 10.1016/j.neucom.2018.10.009. URL: https://doi.org/10.1016/j.neucom.2018.10.009.

[66] Jamie Shotton et al. "Real-time human pose recognition in parts from single depth images". In: *CVPR 2011*. IEEE, 2011. DOI: 10.1109/cvpr.2011.5995316. URL: https://doi.org/10.1109/cvpr.2011.5995316.

[67] Jungong Han et al. "Enhanced computer vision with Microsoft Kinect sensor: a review". In: *IEEE Transactions on Cybernetics* 43.5 (2013), pp. 1318–1334. DOI: 10.1109/tcyb.2013.2265378. URL: https://doi.org/10.1109/tcyb.2013.2265378.

[68] Youding Zhu, Behzad Dariush, and Kikuo Fujimura. "Controlled human pose estimation from depth image streams". In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2008. DOI: 10.1109/cvprw.2008.4563163. URL: https://doi.org/10.1109/cvprw.2008.4563163.

[69] Loren Arthur Schwarz et al. "Estimating human 3D pose from Time-of-Flight images based on geodesic distances and optical flow". In: *Face and Gesture 2011*. IEEE, 2011. DOI: 10.1109/fg.2011.5771333. URL: https://doi.org/10.1109/fg.2011.5771333.

[70] Mao Ye et al. "Accurate 3D pose estimation from a single depth image". In: *2011 International Conference on Computer Vision*. IEEE, 2011. DOI: 10.1109/iccv.2011.6126310. URL: https://doi.org/10.1109/iccv.2011.6126310.

[71] Manuel J. Marín-Jiménez et al. "3D human pose estimation from depth maps using a deep combination of poses". In: *Journal of Visual Communication and Image Representation* 55 (2018), pp. 627–639. DOI: 10.1016/j.jvcir.2018.07.010. URL: https://doi.org/10.1016/j.jvcir.2018.07.010.

[72] Ju Yong Chang, Gyeongsik Moon, and Kyoung Mu Lee. "V2V-PoseNet: voxel-to-voxel prediction network for accurate 3D hand and human pose estimation from a single depth map". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018. DOI: 10.1109/cvpr.2018.00533. URL: https://doi.org/10.1109/cvpr.2018.00533.

[73] Christian Zimmermann et al. "3D human pose estimation in RGBD images for robotic task learning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018. DOI: 10.1109/icra.2018.8462833. URL: https://doi.org/10.1109/icra.2018.8462833.

[74] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern*

Recognition (CVPR05). IEEE, 2005. DOI: 10.1109/cvpr.2005.177. URL: https://doi.org/10.1109/cvpr.2005.177.

[75] Mark Sandler et al. "MobileNetV2: inverted residuals and linear bottlenecks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018. DOI: 10.1109/cvpr.2018.00474. URL: https://doi.org/10.1109/cvpr.2018.00474.

[76] Wei Liu et al. "SSD: single shot multibox detector". In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2. URL: https://doi.org/10.1007/978-3-319-46448-0_2.

[77] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in Neural Information Processing Systems 27*. Curran Associate, Inc., 2014, pp. 3104–3112. URL: https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.

[78] Samy Bengio et al. "Scheduled sampling for sequence prediction with recurrent neural networks". In: *Advances in Neural Information Processing Systems 28*. Curran Associate, Inc., 2015, pp. 1171–1179. URL: http://papers.neurips.cc/paper/5956-scheduled-sampling-for-sequence-prediction-with-recurrent-neural-networks.

[79] OpenCV team. *OpenCV*. 2020. URL: https://opencv.org/ (visited on 10/09/2020).

[80] Satya Mallick. *Geometry of image formation*. 2020. URL: https://www.learnopencv.com/geometry-of-image-formation/ (visited on 10/09/2020).

[81] Cameron Appel Max Roser and Hannah Ritchie. "Human height". In: *Our World in Data* (2013). URL: https://ourworldindata.org/human-height (visited on 10/08/2020).

[82] Rudolf Kalman. "A new approach to linear filtering and prediction problems". In: *Journal of Fluids Engineering* 82D (1959), pp. 35–45.

[83] Petteri Aimonen. *Basic concept of Kalman filtering*. 2011. URL: https://commons.wikimedia.org/wiki/File:Basic_concept_of_Kalman_filtering.svg (visited on 10/09/2020).

[84] Daniel Duckworth. *pykalman*. 2012. URL: https://pykalman.github.io (visited on 10/09/2020).

[85] Mykhaylo Andriluka et al. “2D human pose estimation: new benchmark and state of the art Analysis”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014. DOI: 10.1109/cvpr.2014.471. URL: https://doi.org/10.1109/cvpr.2014.471.

[86] Sam Johnson and Mark Everingham. “Clustered pose and nonlinear appearance models for human pose estimation”. In: *Procedings of the British Machine Vision Conference 2010*. British Machine Vision Association, 2010. DOI: 10.5244/c.24.12. URL: https://doi.org/10.5244/c.24.12.

[87] Jaeyong Sung et al. “Unstructured human activity detection from RGBD images”. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012. DOI: 10.1109/icra.2012.6224591. URL: https://doi.org/10.1109/icra.2012.6224591.

[88] Tsung-Yi Lin et al. “Microsoft COCO: common objects in context”. In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48. URL: https://doi.org/10.1007/978-3-319-10602-1_48.

[89] NVIDIA Corporation. *NVIDIA: world leader in artificial intelligence computing*. 2020. URL: https://www.nvidia.com/en-us/ (visited on 10/07/2020).

[90] John Nickolls et al. “Scalable parallel programming with CUDA”. In: *ACM SIGGRAPH 2008 classes on - SIGGRAPH 08*. ACM Press, 2008. DOI: 10.1145/1401132.1401152. URL: https://doi.org/10.1145/1401132.1401152.

[91] Open Robotics. *ROS.org — Powering the world’s robots*. 2020. URL: https://www.ros.org/ (visited on 10/09/2020).

[92] Open Source Robotics Foundation. *Wiki: openni2_launch*. 2020. URL: http://wiki.ros.org/openni2_launch (visited on 10/09/2020).

[93] Roberto Pérez González. *Rperez-tfg*. 2019. URL: https://wiki.jderobot.org/Rperez-tfg (visited on 10/09/2020).

[94] OpenJS Foundation. *Build cross-platform desktop apps with JavaScript, HTML, and CSS*. 2020. URL: https://www.electronjs.org/ (visited on 10/09/2020).

[95] Papers With Code. *Papers with Code - MPII human pose benchmark (pose estimation)*. 2020. URL: https://paperswithcode.com/sota/pose-estimation-on-mpii-human-pose (visited on 10/09/2020).

[96] Daniel Groos, Heri Ramampiaro, and Espen Ihlen. *EfficientPose: scalable single-person pose estimation*. 2020. arXiv: 2004.12186 [cs.CV].

[97] Sinead Kearney et al. "RGBD-Dog: predicting canine pose from RGBD sensors". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020. DOI: 10.1109/cvpr42600.2020.00836. URL: https://doi.org/10.1109/cvpr42600.2020.00836.