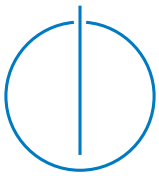
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Biomedical Computing

Real-time 3D Human Pose Estimation with a single RGB-camera for Screen-based Augmented-Reality

Tobias Czempiel



DEPARTMENT OF INFORMATICS

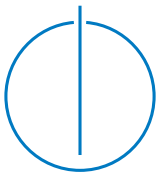
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Biomedical Computing

Real-time 3D Human Pose Estimation with a single RGB-camera for Screen-based Augmented-Reality

Echtzeit 3D Human Pose Estimation mit einer einzigen RGB-Kamera für Monitor basierte Augmented-Reality Anwendungen

Author:	Tobias Czempiel
Supervisor:	Prof. Dr. Nassir Navab
Advisor:	Felix Bork
Submission Date:	February 15, 2019



I confirm that this master's thesis in biomedical computing is my own work and I have documented all sources and material used.

München, February 15, 2019 Tobias Czempiel

Abstract

Giving a system the ability to track and estimate a 3-dimensional (3D) human pose optically is the foundation of many human-machine interaction designs and Augmented-Reality (AR)-applications. They are required in many different application areas such as sports, biomechanics, and medicine. One of the best known human pose estimation systems on the market is the Microsoft Kinect. This system combines a depth camera with a RGB-camera in order to add the third dimension to a 2-dimensional (2D) red, green and blue (RGB) image. To make AR-systems available for a broader audience; however, factors like space, cost, and simplicity make it favorable to rely on a single RGB-camera only, which is available in most computers and smartphones. Several methods to estimate 3D poses in such a setting have been published. This thesis reviews a selection of such methods. It identifies and describes VideoPose3D as one that performs well enough to replace the Kinect in a real-world AR application developed at the CAMP chair of TU Munich, the medical Magic Mirror.

Contents

1 Introduction

1.1 Motivation

Computer vision aims to give computers high-level understanding from images or videos. For 3D human pose estimation, the objective is to provide computer systems the ability to estimate the spatial arrangements of all individual body parts. More formally expressed, "given an image - a 2-dimensional representation - of a human being, 3D pose estimation is the task of producing a 3-dimensional figure that matches the spatial position of the depicted person" [17]. In the real world humans can rely on the stereopsis created by our eye pair but even on photos humans can understand the arrangement of the three-dimension human pose by building upon the experience gained during childhood. However, there are examples where even humans cannot explain a 3D pose correctly, especially in situations where too many depth ambiguities and occlusions by objects are present.

The ability to understand 3D human poses from a single RGB-camera is required in a large number of applications. Tasks like computer interaction, autonomous driving or even apparel size estimation would become more effective if there exists a reliable method to recover the pose on already available consumer hardware. In medicine, this could be used to analyze the range of motion of individual joints or to conduct a gait study without the need for specialized hardware. Virtual reality and especially augmented reality would also benefit. Both concepts are becoming more significant, and the ability to obtain a sound 3D pose for all the detected humans in a camera image would extend the range of tasks for virtual and augmented-reality devices drastically.

1.2 Problem Statement

Solving 3D human pose estimation is a very challenging problem due to large appearance variances, non-rigidity of the human body, different viewpoints of the camera, cluttered

background, and self-occlusions [40] as well as the large number of degrees of freedom [26]. The problem contains three characteristics that are the main reasons why 3D human pose estimation from a single RGB image is hard to solve.

1) It is a *ill-conditioned* problem. When detecting the 2D joints minor errors in the 2D detection can lead to large errors in the 3D detection. Many 3D human pose estimation systems perform a prior 2D human pose estimation and use the results as input for a subsequent 2D to 3D lifting of the joints to a 3D pose [29].

2) It is a severely *ill-posed* problem. This means that small errors when interpreting the 2D data can lead to large errors in the 3D solution. Two very different 3D poses might get projected back to a similarly 2D pose. A good example of this can be made with the position of the arms. The 2D projections of two 3D poses with one having the arm before the body root with an arbitrary angle γ, between the arm and the torso, and the other one having the arm behind the body root with the same angle γ will be identical if the projections are made from the front [29].

3) It is a *high dimensionality* problem [1]. The task of 3D human pose estimation cannot be solved by relying on a small amount of image features (e.g., average color, amount of green, image edges). A large number of features has to be used in combination to create adequate results. This also means that for learning based algorithms a huge number of training examples is needed for effective learning. If N training samples suffice to cover a 1D feature space, then N^2 samples are needed to cover a 2D feature space with the same density, and N^{1000} samples are needed in a 1000D feature space. The number of training data needed grows exponentially with the number of dimensions used [33].

1.3 Contributions

In this thesis, we provide a detailed introduction to 3D human pose estimation and proceeded with a review of relevant and recent literature. In a theoretical analysis of current state-of-the-art publications, we examined the techniques used in the different papers and tried to reason about the in-the-wild performance of each individual framework. In our three staged filtering process, we narrowed down our collection of possible methods from eight to two and conducted more experiments on the remaining two methods, to identify the overall best performing RGB method for real-time 3D human pose estimation in-the-wild. In a follow up detailed comparison, we opposed the Kinect's 3D human pose estimation system with the best performing RGB method, identified in the previous

step. The detailed comparison contains six actions that were executed by four different participants. The 3D human pose estimation results of the best RGB method and the Kinect were compared on several keyframes to evaluate the performance differences. Overall, this thesis provides a comprehensive introduction into the field of 3D human pose estimation and explains the key concepts used in this research field.

2 Background Theory

2.1 Human Pose Estimation

Human pose estimation or articulated pose estimation is a computer vision task to estimate the configuration of the human body in a single image or a series of images. A human has many limbs with different length and joints which have a variety of angles they can attain. The body configuration is commonly represented using a stick figure model of a human seen in Figure 2.1. In this example, the human pose is described using 15 individual points on the body joints, but different models use 16 or even up to 34 points that describe the pose. The additional points are commonly used to extend the pose information between the neck and pelvis or to give more details to hand and head position. In addition to the number of joints, a predefined underlying structure is

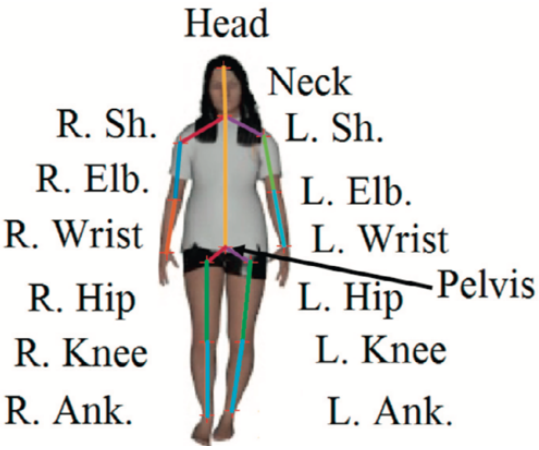


Figure 2.1: Skeleton Body Model with 15 joints [29, p.5]

used to express the relationships between the joints. A connection between two joints is a limb. In Figure 2.2 you can see the underlying structure of Figure 2.1. While some joints like the ankles or knees have only one to one connections the neck forms three

connections to both elbows and the head. The pelvis gets commonly used as the root joint for the human body.

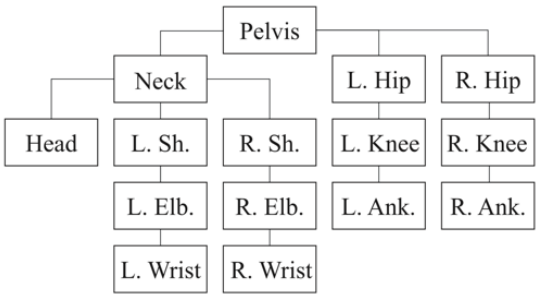


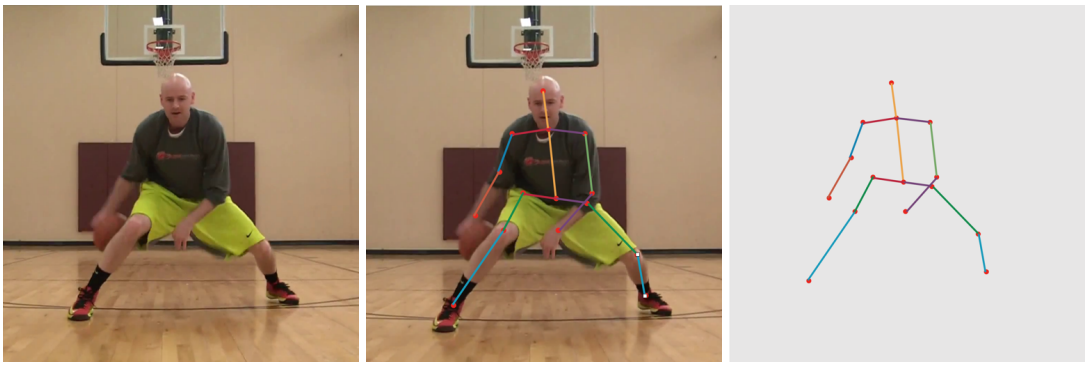
Figure 2.2: Tree-structured representation of the 15 joints [29, p.5]

2.2 2D Human Pose Estimation - Definition

2D human pose estimation is the task of determining the precise pixel location of all joints of the body. This can be done on a single RGB image. The accuracy of the joint position is also related to the image resolution. For each body joint we want to know the exact x and y pixel coordinate in the given image [21]. Once we have all the pixel locations of each joint, we can assemble the skeleton model of the human using the predefined structure (Figure 2.2), and we can overlay the created model on top of the original image as seen in Figure 2.3.

2.3 3D Human Pose Estimation - Definition

3D human pose estimation extends the concept of the 2D human pose estimation and adds the missing third dimension, the depth of each joint, on top of the x and y coordinates. In 3D human pose estimation every joint is defined by three variables x, y and z. An exemplary 3D human pose can be seen in Figure 2.4 using the same image of the basketball player that was already used for 2D human pose estimation. In each subfigure of Figure 2.4 the same pose is visualized, but the viewing angle of the camera is adjusted. To attain those coordinates, a wide variety of methods to automatically attain the pose information in 3D are available. It is very common to use the more reliable 2D results as a baseline for the 3D human pose estimation.

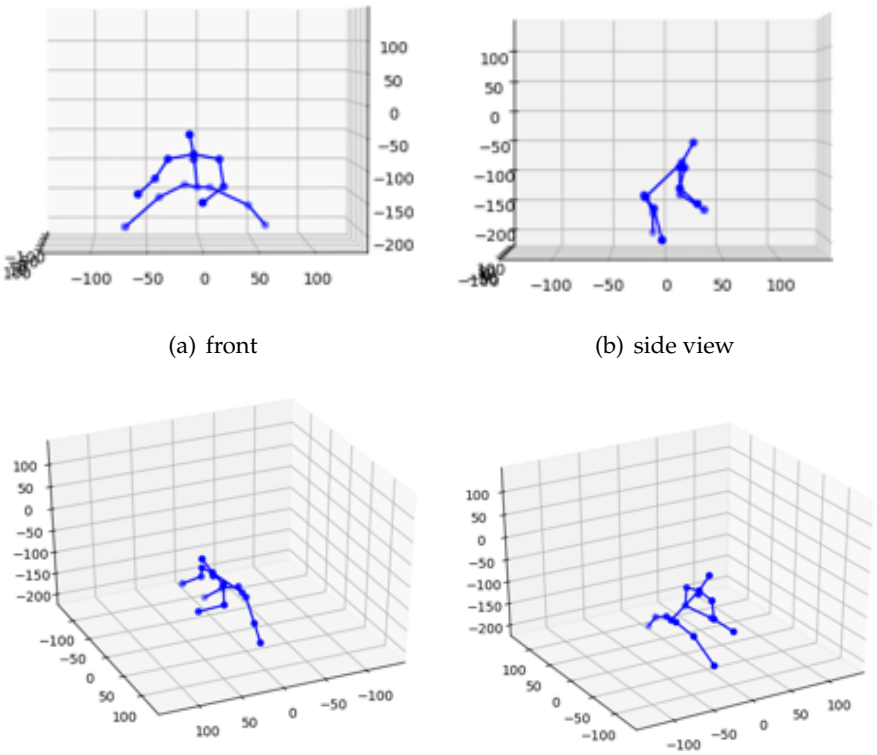


(a) Basketball original (b) original and 2D skeleton (c) 2D skeleton

Figure 2.3: Pose estimation in 2D - Image from MPII-dataset [2]

2.4 MagicMirror AR-System

Different pose estimation systems have different strengths and weaknesses. Consequently, this thesis focuses on finding the right pose estimation system for one particular example, the Magic Mirror designed by the CAMP chair at TUM. The Magic Mirror is a research project and is frequently improved with new concepts and ideas. The Magic Mirror is a screen-based AR system that enables a virtual view inside of the users own body as seen in Figure 2.5(a). This helps its users to understand the dynamic behavior of the human anatomy interactively [3]. Especially for medical education, this AR experience can significantly improve the understanding of human physiology. Studies confirmed that the system could be a valuable addition to the traditional curriculum [4]. The system facilitates mental mapping by augmenting the users' virtual organs and physiology on top of the students' bodies. This helps to improve the spacial understanding of the internal anatomical structures and enhances the motivation and engagement in the learning process. Apart from education another appealing field to use the AR system for is patient rehabilitation. Metrics like the joint range of motions could be calculated without the physical therapist and by standardizing the process of acquiring the desired numbers would be more reliable. These advances lay the foundations for non-supervised patient-specific rehabilitation. For future applications, the AR Magic Mirror systems could also be utilized for patient education and information and intraoperatively serving as a smart AR information system. A 3D human pose estimation system is core to deploy the Magic Mirror system. Currently the Kinect 3.10 and the intel realsense [13] are used. Those systems have to be additionally purchased and are not compatible with



(a) front (b) side view

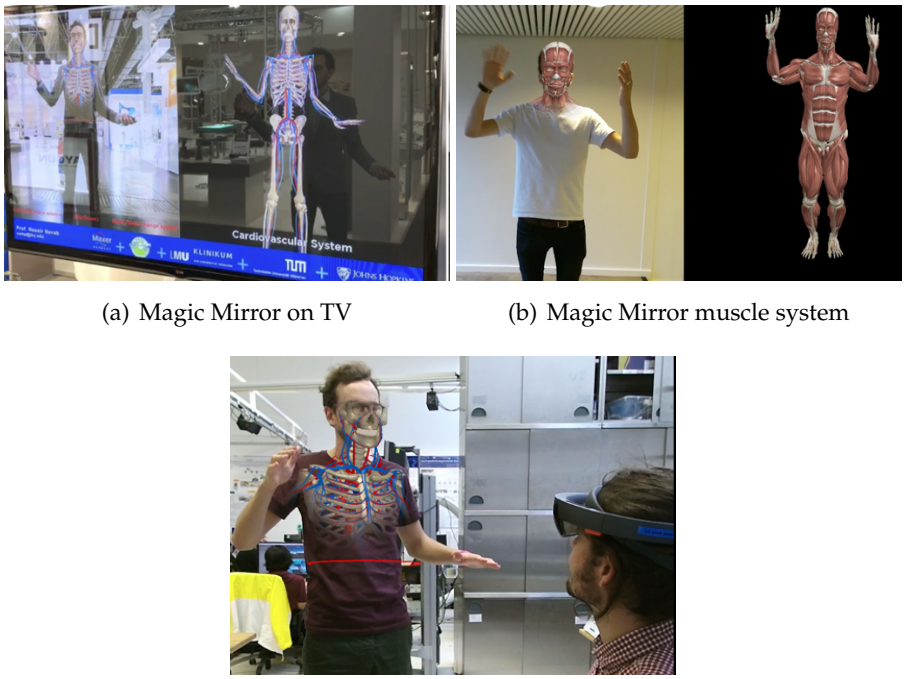
(c) top left (d) back right

Figure 2.4: 3D Pose Estimation Example of the basketball player seen in Figure 2.3

modern smartphones, tablets or in the case of Kinect different operating systems. With a reliable 3D human pose estimation based on a single RGB camera one could also use the Microsoft HoloLens as a device to experience Magic Mirror without the need of an additional sensor attached to the device (cf. Figure 2.5(c)). By replacing the Kinect and the realsense systems with an RGB based 3D human pose estimation the availability of the Magic Mirror AR-System would be significantly increased.

2.4.1 Requirements for MagicMirror

The Magic Mirror AR-System has specific requirements for the 3D human pose estimator component. In Table 2.1 a variety of features that 3D human pose estimators possess is listed. Here, the selection of the features is made for the MagicMirror AR-System.



(a) Magic Mirror on TV

(b) Magic Mirror muscle system

(c) Magic Mirror on HoloLens

Figure 2.5: Magic Mirror AR-System

Depending on the application the composition of the requirements might differ. The Magic Mirror depends on a 3D human pose and a 2D human pose to track the trajectory of the person that was detected in each frame. The system should be a real-time system meaning that the framework is capable of detecting 15-30 poses per second. The Magic Mirror system is usually used in different environments like exhibitions or classrooms. Therefore the system must still perform well if the background has changed and should be robust to background clutter. Most occlusions that appear while using the Magic Mirror system when other people are walking between the current user of the system and the camera. Therefore, it is beneficial if the system is robust to occlusions, but it is not a mandatory feature. The accuracy of the detection should be precise, but it is not necessary to have a sub-cm accuracy. If the system can detect multiple people, the Magic Mirror could be explored collaborative, but in the current stage this is also not a mandatory feature. We want to replace the currently used Kinect hardware and only rely on an RGB camera to bring AR to a broader audience. The system should be platform independent with the option to port the system even to mobile platforms.

	MagicMirror AR-System
2D human pose estimation	yes
3D human pose estimation	yes
real-time	yes
robust to background clutter	yes
robust to occlusion of body parts	favoured
accuracy of joint detection	cm range
multi-person detection	optional
single shot	yes
camera & sensor	RGB camera
platform	Windows, Unix

Table 2.1: Features of 3D human pose estimator systems and requirements for MagicMirror AR-System

3 Related Work

3.1 2D Human Pose Estimation Systems

Earlier works on 2D pose estimation done in 2006 used a *model based approach*. The edge-based deformable model they were using is seen in Figure 3.1. Dominant edges of an image are extracted, and a prior designed edge based deformable model is matched onto the edges of the image. The results of this process are the image body part regions. The image body part regions represent where the locations with the highest probabilities for the individual body parts are located in the original image. In Figure 3.1 The location of the right upper arm (ru-arm) was estimated pretty accurately whereas the left lower leg (ll-leg) has a lot of high probability locations and could therefore not accurately be localized by the method. In further works, the results are used as an initial parse and are refined in further steps (cf. Figure 3.2).

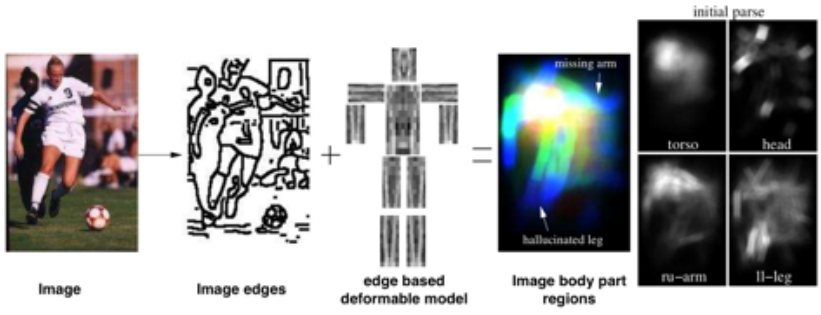


Figure 3.1: edge-based deformable model 2D pose estimation [26, p.3]

The major problem to those model-based approaches is the predefined edge based deformable model itself. If objects or body parts occlude a person in the image, the model cannot be accurately matched. Also, the results may be very inaccurate if the person does not appear in a frontal perspective to the camera. In this case, it would

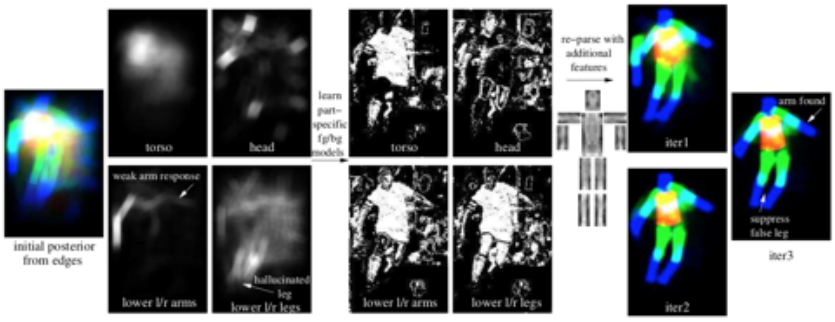


Figure 3.2: edge-based deformable model 2D pose estimation advanced [26, p.3]

be necessary to apply a different model of a human, in side view, that can describe the human pose more precisely [26].

In 2014 a more recent method introduced a deep neural network (DNN) to regress the 2D joint locations. A **DNN-based 2D pose regressor** consists of a variety of layers strung together after each other. A linear and a non-linear transformation constructs each layer. In Figure 3.3 an example of such a network containing convolutional and fully connected layers can be seen. Both the convolutional and the linear layers contain

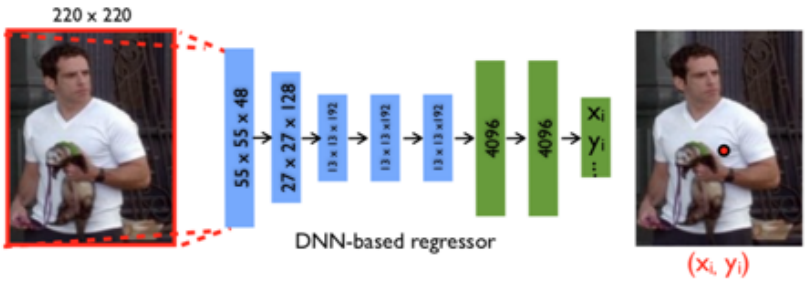


Figure 3.3: DNN-based pose regression - blue: convolutional layer, green: fully connected layer [36]

learnable features. To train this network with labeled training data a loss needs to be defined that can express the quality of the prediction using the difference between the predicted and correct joint locations. A straight forward loss function for the loss layer is the L2 loss consisting of the predicted joint location coordinates minus the correct joint

coordinates squared.

$$argmin \sum_{(x,y)\in D} \sum_{i=1}^{k} \|y_i - \psi_i(x;\theta)\|^2 \tag{3.1}$$

Where y are true joint location and x the corresponding images in the dataset D. ψ_i is the DNN network itself that outputs the predicted joint locations using the weights θ. k is the number of joints to be calculated. The advantage of using machine learning to solve the task of 2D pose estimation is that there is no need to define models or geometric constraints manually. Those constraints will be learned by the network itself using many examples of the task. However, it is challenging to find the right architecture for the network itself. Even though one architecture might give good results a slightly modified version might yield even better results, and in practice, many different modifications have to be tried to find a fitting one. It also takes a lot of computational resources and time to train such a network on thousands or even millions of examples and the more layers are added to the architecture the more time and resource consuming the training gets. DNN-based 2D pose regressors enhanced 2D pose estimation, and almost every new publication is based on it. Therefore following examples, will only consider DNN-based approaches.

3.1.1 Simple baselines for human pose estimation and tracking

In *Simple baselines for human pose estimation and tracking*, Xiao et al. used a rather simple approach by first extracting image features and further adding a few deconvolutional layers on top of it. To extract the image features the authors are using a pre-trained residual neural network (ResNet) [12] framework. By using adherent deconvolutional layers, they generate k heatmaps, where k is the number of joints to be detected. The whole network is visualized in Figure 3.4. To train the network they are using the mean squared error as loss between the predicted heatmap and the target heatmaps. Since the ground truth, joint locations are absolute coordinates and not comparable to a heatmap they have to be transformed into heatmaps. This is done by generating a 2D Gaussian centered at the ground truth location of each joint [38]. The results are not as good as the one created by Stacked Hourglass 3.1.2 or from Open Pose3.1.3, but despite being a straightforward approach it generates good results and shows that it is not always necessary to create very deep and complex networks to generate comparable results.

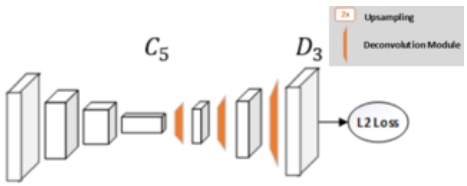


Figure 3.4: simple baselines for human pose estimation and tracking - framework [38]
C5: last layer of ResNet, D3: last layer of deconvolution

3.1.2 Stacked Hourglass

The main requirements for this approach published by Newell et al. where that good pose estimation systems "must be robust to occlusion and severe deformation, successful on rare and novel poses, and invariant to changes in appearance due to factors like clothing and lighting" [21]. Key to the design of the framework is the use of convolutional neural network (CNN) as the main building blocks that replace hand-crafted features or models and drastically improve the performance. As seen in Figure 3.5 stacked hourglass refers to pooling and subsequent up-sampling to obtain the desired results of the network. The name of the architecture can be explained with its graphical visualization forming a laying hourglass. The Stacked Hourglass network samples down to a low resolution and is combining features across multiple scales using a symmetric topology. The network can contain many hourglasses stacked together in an end-to-end manner. Especially with difficult joints like knees and ankles, this framework improved the benchmark by 4-5 %. [21].

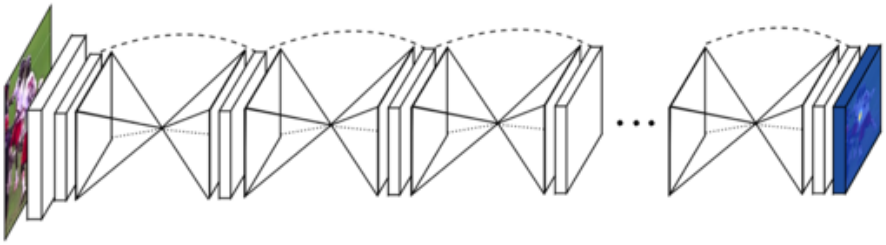


Figure 3.5: Stacked hourglass network [21]

3.1.3 Open Pose

Convolutional Pose Machines consist of a sequence of convolutional networks that produce 2D belief maps. The inputs for a later network are intermediate image features and the belief maps produced by the previous stages. That way the location can be refined stepwise. Belief maps express the spatial uncertainty for each joint. The CPM

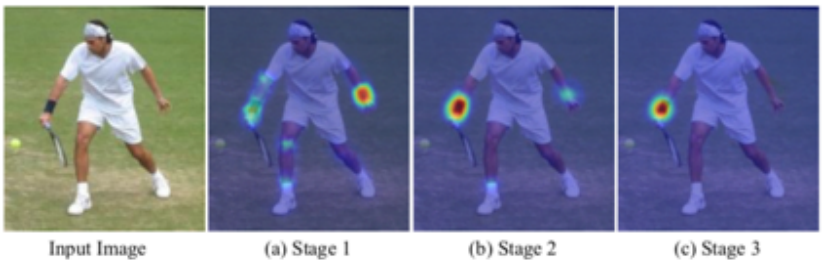


Figure 3.6: CPM [37]

architecture remains more complicated when being compared to Stacked Hourglass (cf. section 3.1.2). The complexity of CPM is also reflected in the prolonged time needed for one forward pass of the network. In Figure 3.7 the architecture of the framework is visualized. Each stage receives the belief maps of the previous stage as input(b) together with the original image (x'). The Convolutional Pose Machines have become famous due to their excellent performance and robustness to occlusions under the name OpenPose. OpenPose describes itself as the first real-time, multi-person, joint, hand, body, face and food detector on single images [37].

3.2 3D Human Pose Estimation Systems

3D human pose estimation became popular with the advances in marker-based 3D human pose estimation systems like MoCap. Today's methods try to avoid markers and attempt to reduce the constraints of their systems further. Modern RGB based 3D human pose estimation systems use deep learning to estimate a 3D human pose and are capable enough to detect correct and reasonable pose locations.

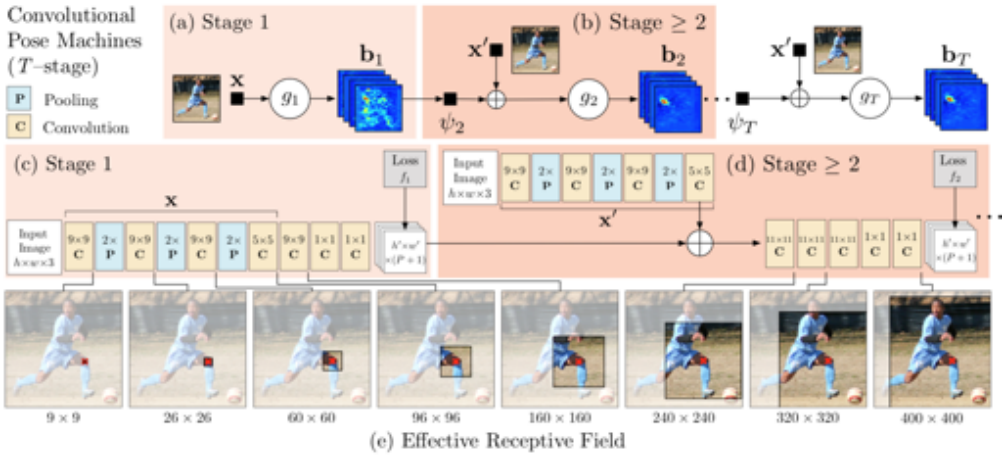
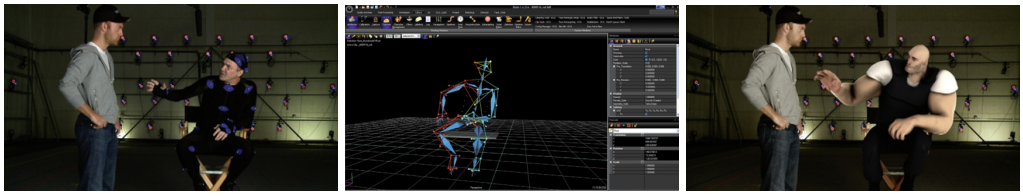


Figure 3.7: CPM [37]

3.2.1 Marker-based 3D Human Pose Estimation

The most popular marker-based system for 3D human pose estimation are the Motion Capturing (MoCap) systems. Optical MoCap systems consist of a camera rig which captures visibly detectable markers. The number of cameras used in such a studio setup can vary between eight up to hundreds of cameras at a time. They have to be calibrated with each other by tracking a target with a known dimension. The views of all the cameras are combined with the known dimension of the calibration target to calculate the exact position of each camera in space. At least two cameras have to be in the line of sight of a marker to acquire the markers exact 3D position. Additional cameras help to maintain the view on each marker even for complicated geometries or occlusions through different objects and interactions in the scene. To track a persons's 3D position accurately, markers have to be placed all over the person. It is helpful to use more markers than needed to add some redundancy. The marker's shape and dimensions have to be known to the system that receives the camera inputs. As a first step, each marker 2D location for each view has to be calculated. The 3D locations of the markers are then calculated using the cameras positions in the room and the prior calculated 2D coordinates of each camera. An example of a MoCap system can be seen in Figure 3.8. In 3.8(a) an actor is sitting on a chair with many markers attached to him. In the background are cameras with their red light sources surrounding their lenses. In 3.8(b) the extracted 3D skeleton is visible and in 3.8(c) the actor was replaced by a virtual character that is deformed using the 3D skeleton of the actor [20, p.19-21]. The advantages of marker-based systems



(a) MoCap with marker (b) MoCap extracted 3D (c) MoCap final animation

Figure 3.8: Motion Capturing [20]

are that the acquired data is highly accurate. By using many markers and groups of markers that are registered to the internal skeleton of the body, the pose can be easily calculated [20, p.20]. Tracked 3D poses from MoCap systems are commonly used as data-source for 3D human pose datasets such as Human3.6m (cf. section 4.1.2) due to their preciseness.

3.2.2 3D Human Pose Estimation additional depth sensors

It is possible to get the depth information (z - coordinate) and the color information (x and y) completely independent from each other. In *Structured light* methods we are illuminating the scene with a particular light pattern. Frequently used patterns are horizontal lines or checkerboard patterns. If an object is brought into the scene, the pattern will be deformed. The closer the object, the larger the pattern will grow on the surface of the object. This effect can be used to detect the depth of each point of the image.

Time-of-flight depth imaging techniques are a more modern and more precise way to measure the depth. Those techniques measure the depth of a scene by using light pulses and measuring how long a pulse of light needs to reach back to the camera after being reflected. The closer a reflecting object is to the light source, the smaller the amount of time such a round trip takes. The exact distance can be calculated using the speed of light. Usually, those sensors are operating in the infrared spectrum and are therefore not visible to our eyes. Once the depth information is calculated, it can be combined with a regular RGB camera image. To obtain a 3D human pose estimation, it is also common to perform 2D pose estimation and use additional depth sensors to attain the depth information for the detected joints.

Kinect

(a) Kinect V1 (b) Kinect V2

Figure 3.9: In 2014 Microsoft released the second version of the Kinect 3.9(b). The Kinect V1 uses structured light to calculate the depth image. The Kinect V2 however uses time of flight technique, which gives a better resolution, to calculate the depth image.

The Microsoft Kinect System is probably the most famous proprietary system for 3D human pose estimation. It was first released for the gaming console Xbox in 2009 and later in 2011 was made compatible with Microsoft Windows computers. It is the first consumer hardware that was able to capture depth images with interactive frame rates. It uses real-time depth images as inputs for their deep randomized decision forest classifier to calculate body part distribution. In the next step, the body part distribution is pooled across pixels to generate accurate coordinates for the 3D skeletal joints (cf. 3.10). To avoid overfitting, they used hundreds of thousands of training examples. This classifier can recover the human shape in minimal 5ms per frame resulting in a theoretical maximum frames per second (fps) rate of 200 fps. This was up to ten times faster than the existing methods at that time and enabled real-time 3D human pose estimation for the first time [31].

3.2.3 RGB learning based 3D Human Pose Estimation Systems

The amount of publications for this topic is immense. This is why we cannot give a complete overview of all the available methods and publications without exceeding the limits of this thesis. *FBI-pose* published by Shi et al. [30] use a similar approach to Pavlakos et al. (cf. section 3.2.10). Fang et al. tried to use pose grammar to solve the task of 3D human pose estimation [7]. At ECCV 2018 Coskun et al. [6] published their

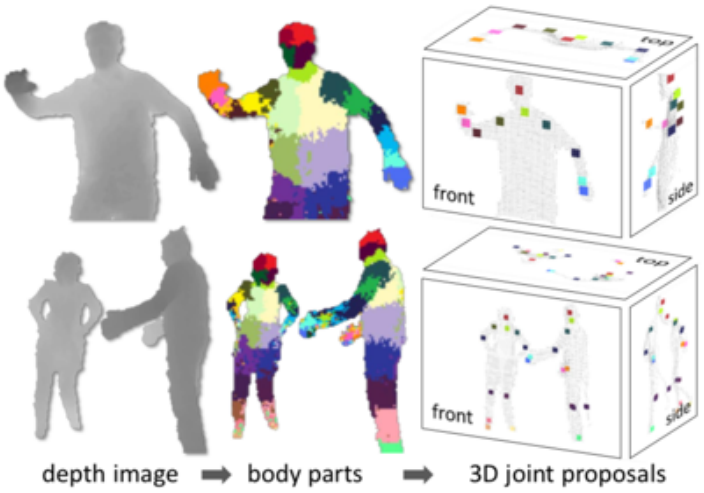


Figure 3.10: Kinect from depth image to body parts to 3D joint proposals [31]

Human Motion Analysis with Deep Metric Learning paper to address this topic. Also at ECCV 2018 Rhodin et al. [27] published their paper on *Unsupervised Geometry-Aware Representation for 3D Human Pose Estimation*. *Densepose*, another new publication comes from Güler et al. and Facebook [10] tries to map all individual pixels of an RGB image to a 3D surface of the body. There are many more publications worth mentioning, but we choose to narrow down our literature review to those papers that we think work best for our purpose.

3.2.4 VNect

In 2017 Mehta et al. published *VNect: Real-time 3D human pose Estimation with a Single RGB Camera* which they said is the "first real-time method to capture the full global 3D skeletal pose of a human in a stable, temporally consistent manner using a single RGB camera" [19]. They combined a CNN pose regressor network with an adherent kinematic skeleton fitting. An overview of the framework is visualized in Figure 3.11. The CNN pose regression module outputs 2D and 3D joint positions in real-time. Similar to *simple baselines* (cf. section 3.4)the CNN used in VNect also uses a heatmap formulation for 2D pose description. The authors extended this concept further to a novel 3D heatmap formulation using three additional location-maps X_j, Y_j and Z_j per joint j. The locations x_j, y_j and z_j are given relative to the distance of the center of the hip. After returning j

2D heatmaps, for each joints probability, the coordinate of the 2D points with the highest probability is used, and the corresponding 3D coordinates from the 3D location-maps are taken to give the final 3D pose. This is visualized in Figure 3.12 here the maximum 2D response coordinate for the first joint H_1 is used, and the values for the 3D pose are extracted at that location. Doing this for each joint gives a premature 3D human pose that in the next step is refined using a custom filter [19].

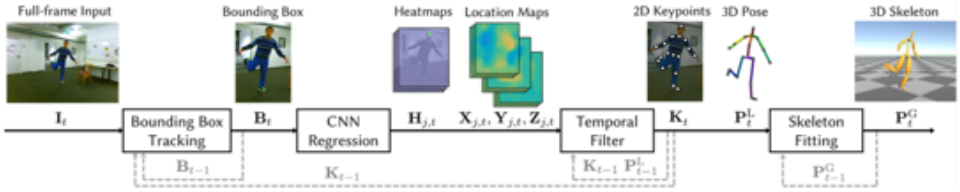


Figure 3.11: VNect - framework [19]

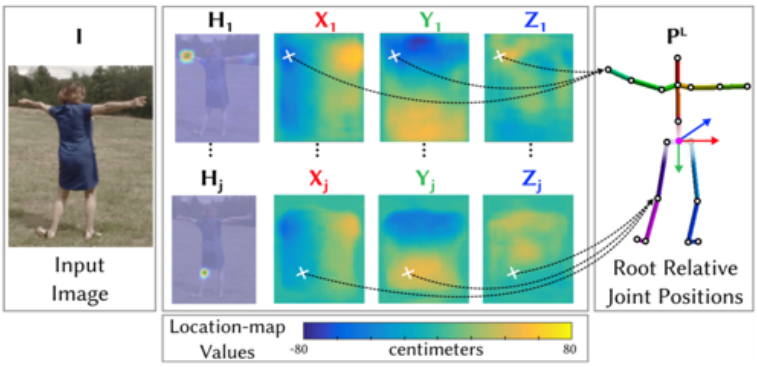


Figure 3.12: VNect - 2D and 3D heatmaps formulation [19]

3.2.5 LCR-Net

When designing the *Localization-Classification-Regression Network* (LCR) Rogez et al. [28] tried focusing on occlusions of body parts that can make it very hard for current networks to reason about the 3D pose. The cause why current networks cannot handle occlusions very well is that to train a 3D pose estimator training data that usually is captured using Motion Capturing (section 3.8) devices is used. If the training data does not contain images with occluded body parts in it, the trained network cannot learn to predict the

pose of a body when occlusions are present in the test image. Therefore those networks do not generalize well which also means that they cannot handle occlusions very well. As a first step Rogez et al. took a subset of the training sets 3D poses and used K-means to cluster them which gave them the most common 3D poses for the given set. Through experiments, they found out that 100 initial points for K-means gave them the best results on Human 3.6m. By using too few initial points, the number of joints might not be sufficient to cover the pose space. When using too many the chosen anchor points by K-means might become too similar resulting in ambiguities of the classification. In

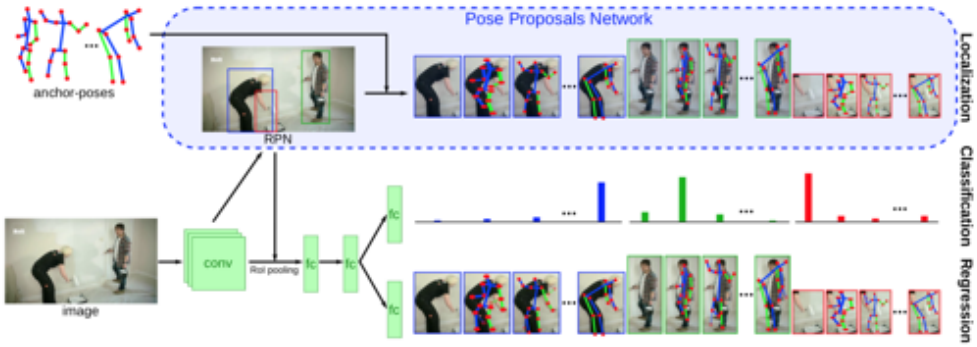


Figure 3.13: LCR-Net framework

Figure 3.13 an overview of the LCR-Net can be seen. In a first step, convolutional image features are computed. Those features are used in a Pose Proposals Network that returns a set of anchor-poses that fit best to the computed convolutional features. After two fully connected layers, the network splits into a classification path that estimates the probability to be correct at the given location the regression path refines the results [28]. Since all the anchor-poses are full-body 3D poses even though the image is highly occluded or only parts of a human are visible the network returns a full-body 3D pose (cf. Figure 3.14).

3.2.6 A simple yet effective baseline for 3d human pose estimation

Martinez et al. [17] wanted to find out if a limited 2D pose understanding introduces the 3D pose error created by deep end-to-end systems or if systems fail to map the 3D pose correctly, given a correct 2D pose. An end-to-end 3D pose estimation system has to recover a 3D pose from a single RGB image and therefore has to be invariant to image factors such as background, lighting, the texture of clothing, skin color and image

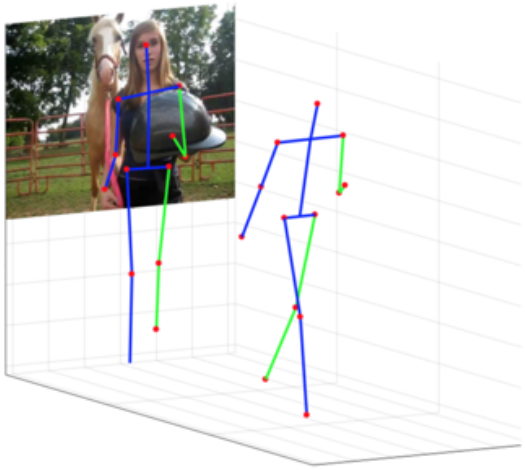


Figure 3.14: output of full-body 3D pose with high occlusion

imperfections. To give a system the ability to be invariant to those factors a huge variety of ground truth training data is necessary. However, there are only a few 3D human pose training sets available. For 2D human more and more versatile training data with a large variety of image factors are available. With this in mind, the idea was to build a system that "lifts" 2D joint locations into 3D. Contrary to other approaches Martinez et al. decoupled 3D pose estimation into two individual problems. Problem one, 2D pose estimation, is considered almost solved with the advances of Stacked Hourglass (section 3.1.2) and Open Pose (section 3.1.3). Problem two is the lifting of the 2D pose into 3D. Figure 3.15 shows the architecture of the network Martinez et al. used for lifting

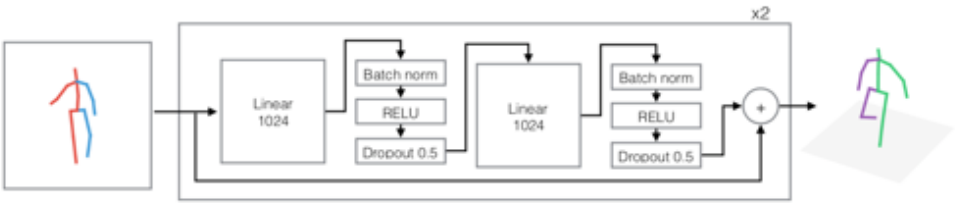


Figure 3.15: Network overview - Linear layer followed by Batch norm, ReLU and a Dropout layer. This is repeated twice and a residual connection is wrapped around it. This whole block is then repeated twice.

the 2D pose into 3D. To estimate the 2D pose they used the stacked hourglass network (section 3.5). They trained the network on the Human3.6M dataset and could archive competitive results with it [17].

3.2.7 Towards 3D human pose estimation in the Wild: a Weakly-supervised Approach

Zhou et al. tried to improve 3D human pose estimation in-the-wild by using a weakly-supervised transfer learning method. They pointed out that currently existing datasets contain either in-the-wild images with 2D ground truth information or have 3D ground truth information but the images are captured in a constrained lab environment and algorithms trained on those images tend to overfit and do not generalize very well. In Figure 3.16 the network architecture is illustrated. It consists of two modules a 2D pose estimation module and a depth regression module. Even though this approach seems similar to 3.2.6 it is different because this network can be trained in an end-to-end fashion. Intermediate image features created by the 2D pose estimation module are used as input for the depth regression module and are not discarded. Those image features can help to solve the 3D pose recovery and are a valuable addition to the prior estimated 2D pose. The 2D pose estimator is an adapted version of Stacked Hourglass (section 3.1.2).

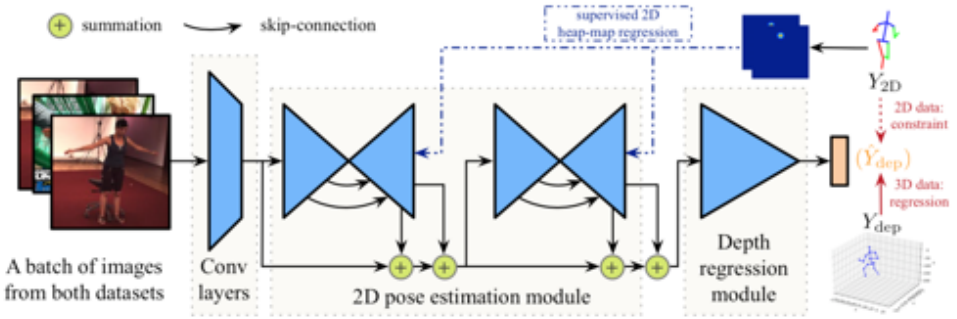


Figure 3.16: Network overview - End-to-End network to estimate the 3D pose from an image. The network contains two parts a 2D pose estimation module and a depth regression module

This network has to be trained in multiple stages. Stage 1 initializes the 2D pose module and is trained on the 2D in-the-wild MPII dataset. In this step only the 2D pose module is active. In Stage 2 the 3D pose estimation module is initialized, and the

2D pose module is also fine-tuned on the ground truth 3D pose information from the Human3.6M dataset. In Stage 3 a geometric constraint for joint length and proportions is activated to fine tune the whole network. With this method, it should be possible to estimate a reliable 3D pose from an in-the-wild image since the 2D pose module was trained on in-the-wild [42]. In Figure 3.17 you can see a sample output of the trained network.

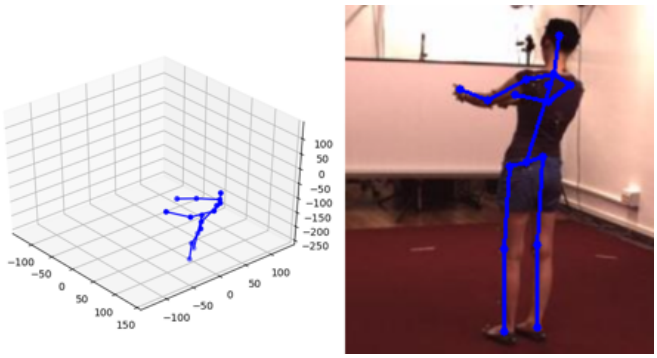


Figure 3.17: Example output 3D pose- Image from Human3.6M dataset

3.2.8 3D Human Pose Estimation in-the-wild by Adversarial Learning

Yang et al. [39] used adversarial learning with a generative adversarial network (GAN) to improve the in-the-wild performance of an already existing 3D human pose estimation system. A GAN is a minimax game with two players. Player one, the discriminator estimates the probability that a sample came from the training data rather than from the second player, the generator. The generator's objective is to maximize the probability of the discriminator to make a mistake [9]. The generator is trained to create "fake" 3D human poses in a way so that the discriminator cannot understand if it is a "fake" or "real" pose. Yang et al. used an adapted version of *Towards 3D Human Pose Estimation in the Wild* (cf. section 3.2.7) as their generator. The discriminator has three information sources that are used for decision making - 1) the original image, 2) pairwise relative locations and distances and 3) the 2D heatmaps of the joint locations and their respective depth maps. The original image I is used for rich contextual information that helps to reduce ambiguities (cf. Figure 3.19(a)). The pairwise relative locations (cf. Figure 3.19(b)) form a geometric descriptor that helps to learn the body's articulation constraints. This idea is based on traditional methods similar to edge-based deformable models

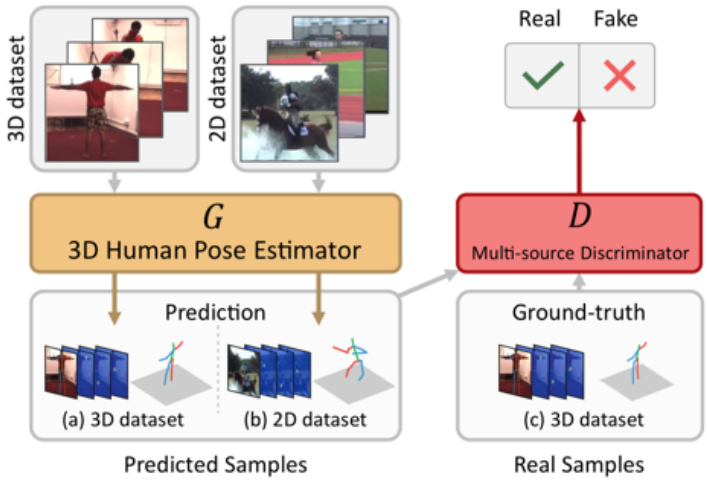


Figure 3.18: Network overview [39]

(cf. section 3.2). It enforces similar joint lengths and limits of joint angles. The third information source for the descriptor is the 3D human pose estimation consisting of the heat map formulation of the 2D estimation and the corresponding depth map [39]. By pretraining the generator, convergence is faster, and the overall results are better. After the pretraining step, both the generator and the discriminator are alternately optimized. The generator produces a 3D human pose consisting of a 2D heatmap, and its' respective depth map. The discriminator receives this output mixed with ground-truth data. Within each bach half of the data is "fake" and created by the generator and the other half "real," ground-truth information from a 3D human pose dataset [39]. Yang et al. showed that they could achieve excellent results on both the Human 3.6m (section 4.1.2) and the MPI-INF-3DHP dataset (section 4.1.3). In some scenes, they could even improve the performance of their backbone architecture (cf. section 3.2.7) by 17.5%.

3.2.9 Unsupervised Adversarial Learning of 3D Human Pose from 2D Joint Locations

Kudo et al. [16] propose a 3D human pose estimating system from 2D ground-truth information. Their unsupervised method aims to learn a 3D human pose without any 3D dataset. The system is based on a GAN. The primary concept is that if a network predicts a 3D human pose precisely, the projection onto a 2D plane should still look

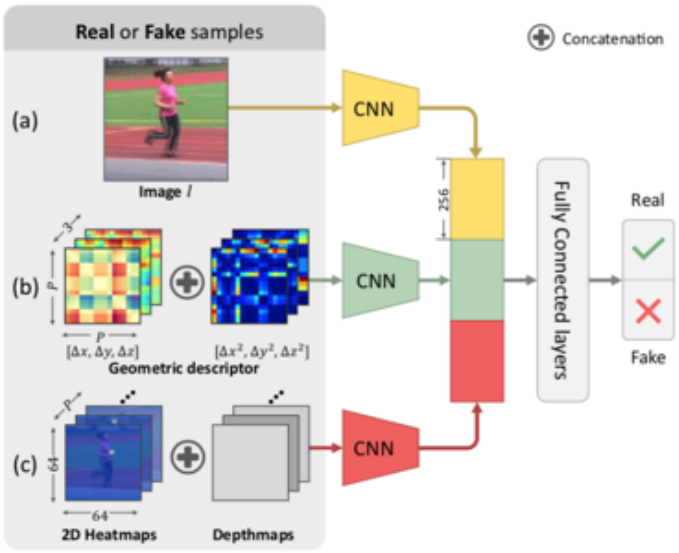


Figure 3.19: Multi dimensional Discriptor [39]

like a valid 2D pose even if rotated perpendicularly. The framework 3.20 is based on a correct 2D Human pose. Using a 2D human pose as input the network generates the corresponding z location for each joint. This resulting 3D pose is rotated around the y-axis and projected back onto the xy-plane. The discriminator is trained to distinguish the "real" 2D pose used as input from the "fake" back-projected 2D poses. The design for the generator is similar to the network presented by Martinez et al. in section 3.2.6. The system enforces that its 3D pose is correct even when rotated around the y-axis assuming the z estimation was correct [16]. A problem with their design was that the 2D pose discriminator could not detect a wrongly generated, fake 3D poses where all the z-components where inverted since the projection of a correct 3D pose to 2D looks identical to a wrong 3D pose projected to 2D seen in Figure 3.21. They could overcome this problem by introducing a new loss function based on the assumption that the right shoulder joint should be on the right side of the head and the left shoulder on the left side of the head. This assumption can be made since a human cannot rotate his head by 180°. A considerable advantage of this method is that the training for 3D human pose estimation can be done entirely on 2D human poses. To use this framework as a 3D human pose detector from a webcam or any image source a prior 2D human pose detector, like 3.5 is necessary. Given that the 2D human pose detector used does produce reliable results there exists an infinite amount of training data due to the massive amount

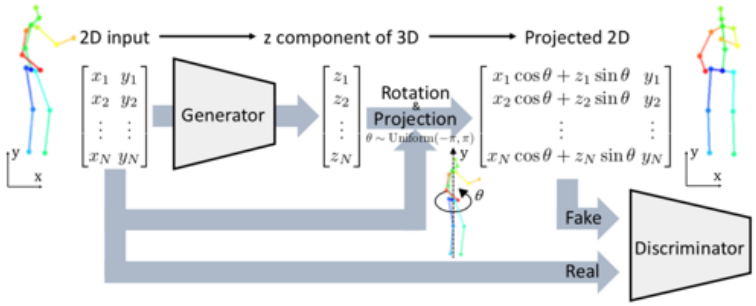


Figure 3.20: Unsupervised Adversarial Learning of 3D Human Pose - framework [16]

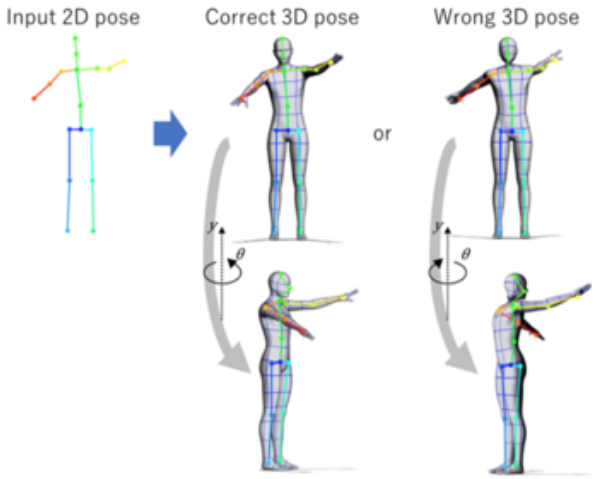


Figure 3.21: Invertion error [16]

of photo and video footage with humans in it available online.

3.2.10 Ordinal Depth Supervision for 3D Human Pose Estimation

Pavlakos et al. [23] start their publication with the observation that the "ability to train end-to-end systems for 3D human pose estimation from single images is currently constrained by the limited availability of 3D annotations for natural images". They suggest using ordinal depths of human joints, a weak supervision signal that can be acquired by manual labeling from human annotators. A 2D dataset can be extended with ordinal depths information. For each pair of joints (i, j), $r_{(i,j)}$ denotes the pairwise

relations between them. The value of $r_{(i,j)}$ is $+1$ if joint i is closer to the camera than joint j. $r_{(i,j)}$ is -1 if joint j is closer than i. If the depth of both joints (i, j) are roughly the same the value is set to 0. For humans, ordinal relations of joints are way easier to annotate then absolute depth values of joints. The authors name the combination of 2D ground-truth joint locations with the additional annotated ordinal depth information a *weak 3D information*. Their network predicts for each joint in the image a depth value. The network uses the following loss term for optimization:

$$L_{(i,j)} = \begin{cases} \log\left(1 + \exp\left(z_i - z_j\right)\right), & r_{(i,j)} = +1 \\ \log\left(1 + \exp\left(-z_i + z_j\right)\right), & r_{(i,j)} = -1 \\ \left(z_i - z_j\right)^2, & r_{(i,j)} = 0. \end{cases} \tag{3.2}$$

If the ground truth label for the pair of (i, j) is $r_{(i,j)} = +1$ the joint i is closer to the camera than j. In this case, the loss function tries to increase the margin between the values of z_i and z_j. If the margin is huge the term $\exp\left(z_i - z_j\right)$ gets very small. In that case, only $\log(1)$, which corresponds to a loss of 0.0, remains since the other term is almost 0. If $r_{(i,j)} = 0$ the loss term uses the Euclidean squared distance to enforce similarity between z_i and z_j. Another loss term is used to train the networks 2D human pose estimation capabilities. Over each joint N the Euclidean squared distance is used between the estimated keypoints coordinates $\hat{w}_n$ and the ground truth locations w_n:

$$L_{keyp} = \sum_{n=1}^{N} \|w_n - \hat{w}_n\|_2^2 \tag{3.3}$$

Pavlakos et al. emphasize that the predicted depth values do not match the exact metric depths of the joints since no full 3D pose example has been used to train the network. That is why they use a reconstruction component that was trained on MoCap (section 3.8) data. This reconstruction component can scale the depth values to a more realistic depth. A visualization of the whole network can be seen in Figure 3.22.

3.2.11 3D human pose estimation in video with temporal convolutions and semi-supervised training

Pavllo et al. divide the problem of 3D human pose estimation into 2D keypoints detection followed by 3D pose estimation. By splitting up the problem into two parts, the difficulty of the problem gets reduced. On the other side, this leads to ambiguities because multiple 3D poses can be mapped to the same 2D poses. Therefore the authors suggest using

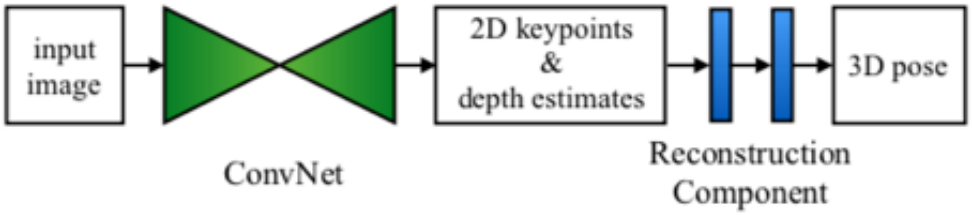


Figure 3.22: Ordinal Depth Supervision for 3D Human Pose Estimation - framework [23]

temporal convolutions over 2D keypoints for precise 3D pose prediction. They created a fully convolutional model that uses dilated temporal convolutions over the 2D keypoints given by any state of the art 2D human pose estimator. In Figure 3.23 the architecture of the proposed network can be seen. The inputs for this network are 2D keypoints detected by a 2D human pose estimator system that detects 17 joints x and y coordinates (34). The input does not only contain one frame (1, 34) but 243 frames chosen accordingly to the predefined temporal convolutional model seen in Figure 3.24. The network uses four blocks consisting of convolutional layers (green), batchnorm, dropout, and ReLU. The output is one 3D human pose with 17 individual joints each one specified with x, y and z coordinate (1, 51). As seen in Figure 3.24 the network relies on 2D poses in the future to

Figure 3.23: 3D human pose estimation in video with temporal convolutions and semi-supervised training - Network [24]

calculate a 3D pose in the present. This temporal model was specifically designed for calculating 3D poses in videos and not for real-time applications. The authors overcome this problem by changing the model slightly as seen in Figure 3.25. Figure 3.25 (a) shows the already introduced symmetric model. Figure 3.25 (b) illustrates how to change the information flow to enable real-time 3D human pose estimation. Pavllo et al. used semi-supervised training to improve the performance of their network and make it generalize better in settings where no ground truth 3D data is available. Figure 3.26 visualizes their approach. The upper part of the Figure visualizes how training on a labeled dataset is done. The lower part of the Figure explains the semi-supervised

Figure 3.24: 3D human pose estimation in video with temporal convolutions and semi-supervised training - temporal convolutional model [24]

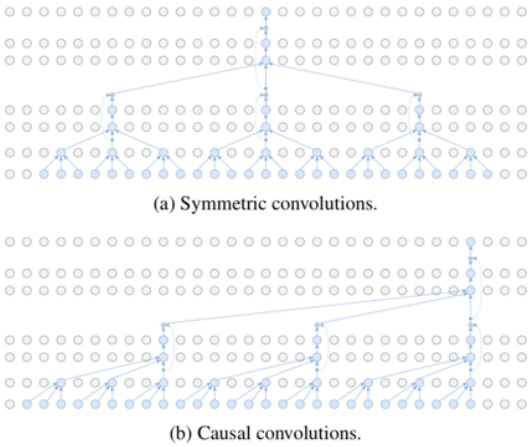


Figure 3.25: 3D human pose estimation in video with temporal convolutions and semi-supervised training - real-time model [24]

training process where a 2D human pose estimation system estimates 2D poses in an unlabeled video. The 3D pose is estimated using the 2D pose and by using the trajectory model the 3D pose can be projected back to 2D the difference between the input 2D pose and the projected 2D pose is used as in the loss function for training. This way the network can be trained both on in-the-wild unlabeled data and 3D human pose datasets like Human3.6m (cf. section 4.1.2).

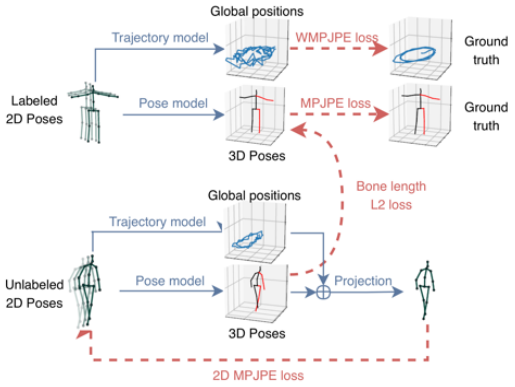


Figure 3.26: 3D human pose estimation in video with temporal convolutions and semi-supervised training - training [24]

4 Datasets & Metrics

In this section the most common datasets for 2D and 3D body pose are presented. Learning based methods depend on a large and diverse dataset that can be used in the training process.

4.1 Datasets

4.1.1 MPII

MPII dataset [2] is a large in-the-wild human pose dataset. The dataset is a collection of images from countless online videos that are manually annotated by humans. For each image, the 2D joint locations of 16 joints are collected. The images cover over 800 different human activities including recreational, occupational and house-holding activities. Also, a broad variety of different camera viewpoints is covered. MPII contains 25000 training images and 2957 validation images.

4.1.2 Human3.6M

The Human 3.6M dataset is a widely used dataset for 3D human pose estimation. This dataset contains 3.6 million RGB images together with the corresponding 3D joint positions caught by a MoCap System in an indoor lab environment. Each scenario is captured from four different camera viewpoints and also includes a wide variety of human activities such as taking photos, talking on the phone, posing, greeting, eating and more. They also provide a statistical evaluation baseline to evaluate a methods accuracy [14].

4.1.3 MPI-INF-3DHP

MPI-INF-3DHP is a relatively new dataset captured in a multi-camera studio with ground truth from marker-less motion capture system [11]. For the ground truth capture, no special suits or markers were required which means that this dataset contains people wearing loose everyday clothing. All the data is recorded in front of a green screen to allow automatic segmentation and augmentation. They used 8 activity sets ranging from walking and sitting to more occluded and covering poses. The dataset consists of 1.3M frames captured by 14 different cameras [18].

4.2 Metric

A common evaluation metric makes comparisons between different results easier. For 3D human pose estimation, there are many different metrics available each adapted to the to the problem the author wanted to solve [29]. In 2010 Sigal et al. developed the 3D Error ϵ that is also named *Mean Per Joint Position Error* (MPJPE) [32] [14]. It is the mean difference between the ground truth 3D position x and the estimated pose $\hat{x}$. The MPJPE is used in the Human3.6m *Protocol 1* evaluation protocol.

$$\epsilon\left(x, \hat{x}\right) = \frac{1}{M} \sum_{i=1}^{M} \| m_i\left(x\right) - m_i\left(\hat{x}\right) \| \tag{4.1}$$

where x is the ground truth position and $\hat{x}$ the estimation. M is the number markers and $m_i(x)$ is the 3D position of the i^{th} marker. A lower MPJPE means that the accuracy of the tested framework is higher. A high MPJPE means the accuracy was lower.

In *Protocol 2* of Human3.6m evaluation the *P-MPJPE* is used. Before calculating the MPJPE, a rigid alignment with the ground truth has been done [24].

In *Protocol 3* of the Human3.6m evaluation, the estimated pose is first aligned to the ground truth. This is called *N-MPJPE* and is especially use full for semi-supervised experiments [27].

The *Percentage of Correctly estimated Parts* (PCP) error is frequently used for 2D human pose estimation but can also be applied in 3D human pose evaluation [8]. It measures the percentage of the body parts that have been located correctly. A body part is located correctly if:

$$\frac{\|s_n - \hat{s}_n\| + \|e_n - \hat{e}_n\|}{2} \leq \alpha \|s_n - e_n\| \tag{4.2}$$

where s is the start and n the end point of a body part n. $\hat{s}$ and $\hat{n}$ are the estimated start and end points for that body part, and α is the threshold.

The *Mean Joint Angle Error* (MJAE) is a metric that captures the error in degree [22]. It gives the error between the estimated and the ground truth joint angles. It is defined as:

$$MJAE = \frac{\sum_{i=1}^{M} |(y_i - y_i')mod \pm 180°|}{M} \tag{4.3}$$

y_i is the i^{th} ground truth joint vector and y' the respective estimated vector over all joints M. The $mod \pm 180°$ reduces the number of all possible values to a minimum of $-180°$ and a maximum of $180°$.

5 Theoretical Performance Analysis

The most common metric to evaluate the accuracy of 3D human pose estimation systems is *Protocol 1* of the Human3.6m dataset. Human3.6m is the most extensive 3D human pose dataset which is valuable for deep neural networks because they need to be trained on large amounts of data to perform well. Authors of datasets often provide their own evaluation metric that can be used to verify the quality of the outputs created by a network that was trained on the dataset. However, the metric might not represent an in-the-wild scenario or a different dataset very well. That is why the comparison in table 5.1 is an important indication for the accuracy of a network but cannot be applied solely to judge the in-the-wild performance of a framework.

	Avg. Protocol 1	Avg. Protocol 2
Mehta et al. [19] 2017	80.5	
Rogez et al. [28] 2018	53.5	61.2
Martinez et al. [17] 2017	62.9	47.7
Zhou et al. [41] 2018	64.9	
Yang et al. [39] 2018	58.6	37.7
Kudo et al. [16] 2018	173.2	
Pavlakos et al. [23] 2018	58.6	41.8
Pavllo et al. [24] 2018	51.8/49.0	40/28.6

Table 5.1: accuracy comparison of frameworks - MPJPE in mm - lower is better

We can see that the estimation error of all frameworks is within the range of 5-10cm. The only outlier here is just the publication from Kudo et al. which has the disadvantage that it was not trained on 3D ground truth locations but unsupervised on 2D locations. The most accurate framework is the one from Pavllo et al. with an accuracy of 51.8mm MPJPE for protocol 1 or 49.0mm if used in causal mode. All frameworks are independent of the platform as their concepts are transferable to nearly any programming language. Also in this comparison, all frameworks are only using an RGB-camera and rely neither on additional depth information nor on a multi-camera setups. Additionally, all frameworks

operate on a single image and do not rely on a continuous video stream. In the latter case, errors can propagate and accumulate in later frames. The only exception here forms Pavllo et al. by using 2D estimation from previous frames to improve their generated 3D human pose. However, even in the single-shot mode the framework of Pavllo et al. produces a highly accurate 3D human pose (table 5.2). Table 5.2 shows which requirements from table 2.1 each framework can fulfill. Another commonality of all frameworks is their potential to be adapted for real-time analyses(Table 5.2).

	real-time	robust to bg-clutter	robust to occlusion	multi-person
Mehta et al. [19] 2017	yes	yes	no	no
Rogez et al. [28] 2018	yes	yes	yes	yes
Martinez et al. [17] 2017	yes	dep. on 2D	dep. on 2D	no
Zhou et al. [41] 2018	yes	yes	yes	no
Yang et al. [39] 2018	yes	yes	yes	no
Kudo et al. [16] 2018	yes	dep. on 2D	dep. on 2D	no
Pavlakos et al. [23] 2018	yes	yes	yes	no
Pavllo et al. [24] 2018	yes	dep. on 2D	dep. on 2D	no

Table 5.2: requirements comparison of frameworks

Mehta et al. [19] pretrained their 2D pose estimation on the MPII dataset (section 4.1.1) and additionally augmented the Human3.6m dataset by adding background occluders that are randomly chosen to change the texture of the background. This enhances in-the-wild performance and also helps to increase robustness against background clutter. The network however always requires an entire human pose with all its joints which can be difficult if objects or other persons occlude the recorded human. Also, strong self-occlusions in frames appears to be an immense problem for the method, and precise detection is impossible in those scenes. This framework does not support multi-person detection.

Rogez et al. [28] utilize pseudo-ground-truth 3D poses and anchor-poses to train their network on the MPII dataset (cf. section 4.1.1). The goal of this design is to give the trained network a solid in-the-wild performance and robustness to background clutter. Occlusions are the primary cause of incomplete detected 2D poses. Even though the 2D pose might be incomplete, the objective is to find the best fitting 3D anchor pose. All anchor poses are complete, and therefore even if the 2D pose is incomplete, the framework delivers a full 3D pose. The framework is mainly designed as a multi-person

human pose estimator which makes it stand out the large amount of single-person detectors.

Martinez et al. [17], Kudo et al. [16] and Pavllo et al. [24] do not infer a 3D human pose in an end-to-end fashion directly from an image. They rely solely on a 2D human pose as input to their algorithms. The advantage is that 2D human pose estimators are usually very robust to background clutter and occlusion due to the highly diverse MPII (cf. section 4.1.1) training set. The robustness of a network can only be judged accurately when the underlying 2D-pose estimator and its training are known as well. 2D pose detectors based on the stacked hourglass architecture (section 3.1.2) and trained on the MPII dataset have proven to work well as foundation for robust 3D pose estimators.

Zhou et al. [41] propose an end-to-end 3D human pose estimation system that uses a weakly-supervised transfer learning method to combine 2D and 3D ground-truth data for learning. The 2D pose estimation sub-network is trained on the MPII dataset (section 4.1.1), and the following 3D pose estimation network is trained on Human3.6m (section 4.1.2). This two-stage approach helps to transfer the learned knowledge of in-the-wild data from the 2D estimator to the 3D estimator. This framework consequently is designed to be both robust to background clutter and occlusions.

Yang et al. [39] use adversarial learning to improve upon the results of the method developed by Zhou et al. [41]. Therefore, robustness to background clutter and occlusion is very similar to Zhou et al.

Pavlakos et al [23] use additional ordinal depth information which was manually annotated to the training data. This information is represented as pairwise relationships between joints and is used as additional input to train the network. Since 2D datasets are very versatile, this aims to give the network the ability to generalize better and be more robust to occlusion and background clutter.

Pavllo et al. [24] introduced a semi-supervised training method for their framework. That way training for 3D human pose can be performed on unlabeled footage as input. A 2D estimator generates a 2D pose for a frame which is used by the network to estimates the corresponding 3D pose. By combining the authors' trajectory model and the 3D pose, the 3D pose can be back-projected to 2D. Using the back-projected 3D pose and the 2D input pose a loss can be calculated. That way a wide variety of training examples can be used for training. This specialized training of the network targets to improve generalization as well as robustness to background clutter and occlusion.

In theory, all of the introduced frameworks should perform well and satisfy established requirements. For end-to-end DNN systems pretraining on 2D datasets or weakly-supervised transfer learning should improve generalization. Also, additional features like the ordinal depth information can help the network to avoid overfitting. A prior 2D detection as input for the 3D estimation is beneficial due to the robustness of the 2D estimation systems. To determine the quality of the frameworks generalization capacities, we conducted a three-staged in-the-wild analysis in the following chapter.

6 Methods

Based on the results presented in section 5 we tested the in-the-wild performance of the introduced frameworks. As mentioned in section 5 it is not sufficient to just compare the MPJPE error. In the real world, the performance of a network can be much less accurate due to reasons which we will examine in this section. We divided the comparison into three stages. Stage one is an initial screening of the networks and the produced outputs. Assuming the results in state one are satisfactory we compare the remaining frameworks with the same in-the-wild video. Since we do not have ground truth data for our in-the-wild test the performance of each framework can only be estimated by visually comparing the input image with the produced output pose. We complete our comparison with a study in stage three by rivaling the best performing framework to the 3D human pose output produced with the Kinect in the same scenario. The Kinect is still the first choice for many AR applications which is why it is interesting to compare the results with the it.

6.1 Stage 1 - Initial Screening

In stage 1 we compare the general performance of the networks and try to reproduce the results reported by their creators. All frameworks that successfully pass stage 1 are described in stage 2 or 3. The methods presented in this stage did not pass the first stage because of restrictions or non-reproducible performance.

6.1.1 VNect

Mehta et al. [19] compared their results to the Kinect (section 3.10) and concluded that their frameworks performance is similar to the Kinect and can even outperform it when limbs are close to a scene object such as a chair when sitting down. Together with their publication, they released a demo showing the real-time application of their framework

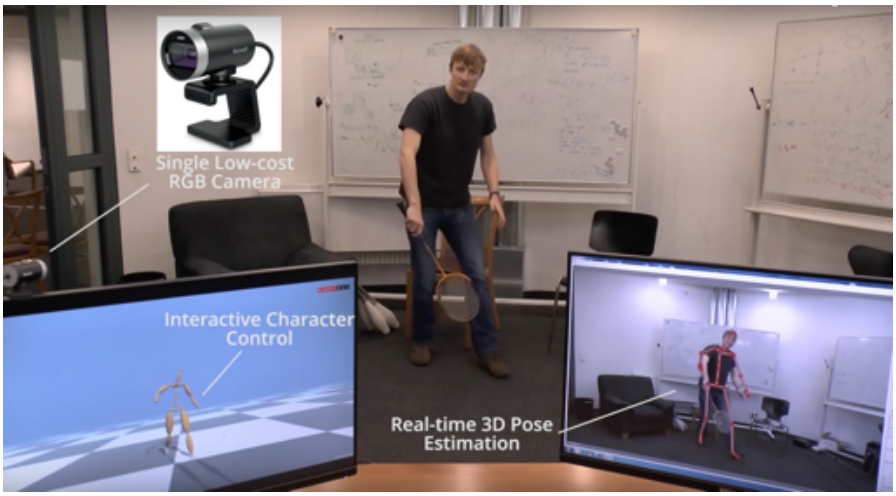


Figure 6.1: VNect video [34]

(cf. Figure 6.1). Even though the results in this video look promising, we could not reproduce them. The system often failed to detect the pose when body parts where occluded by objects, and we could not achieve the promised accuracy (cf. Figure 6.2). The system also uses a bounding box tracker as seen in Figure 3.11. The tracker can loose the tracked body since it relies on the input of the previous frame. This becomes especially problematic when the body is visible only partially, or many different persons are in the camera's field of view. Overall we could not achieve the results promised by

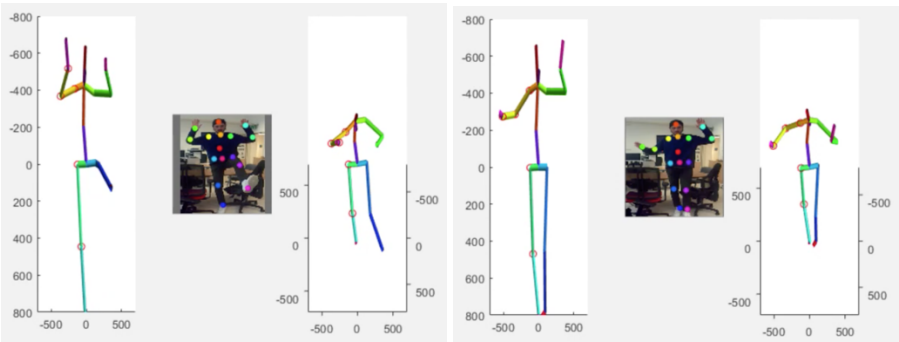


Figure 6.2: VNect - Test, we tested the estimation accuracy of the legs and arms

the authors mainly due to the limitation that the framework performs poorly when only half of the body is visible and the problems we encountered due to the bounding box tracker.

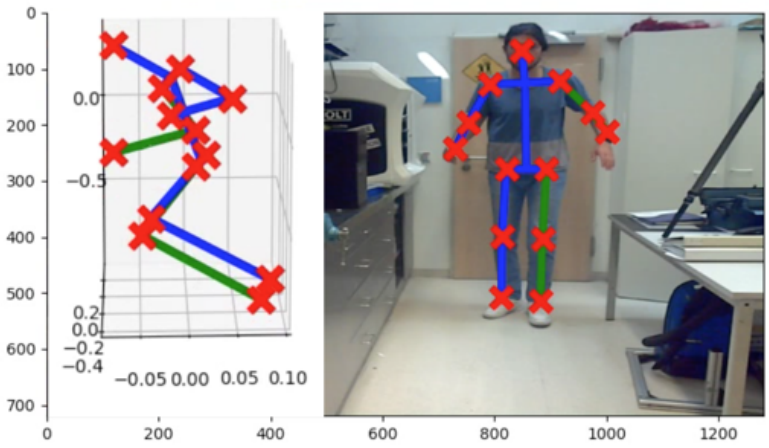


Figure 6.3: LCR - squatted position error

6.1.2 LCR++

Rogez et al. [28] came up with a interesting idea which they called anchor-poses. These anchor poses are found by clustering the data set into the 100-200 most representative poses. One of the advantages of this framework is that even partial 2D poses can be matched with a complete 3D pose. Another advantage is the capability to detect multiple persons. When we tried this network, we realized that some anchor-poses are dominant and chosen by the network too frequently. As one example, the network frequently predicts a person to be in a squatted position despite standing upright (cf. Figure 6.3). The cause for this behavior is hard to isolate. It is probable that the training data had too many squatted positions which also means that there are many anchor-poses in a squatted position. Even though we liked that the framework always estimates a full 3D pose for any detected 2D pose and is running in real-time with support for multiple persons, the accuracy we have been able to achieve for in-the-wild data was insufficient. If a pose is represented only once in the training set, it is improbable that it gets chosen as an anchor-pose and therefore it is unlikely that the framework will predict it correctly. One way to solve this problem might be to choose the anchor-poses manually. However, by choosing the anchor-poses manually, we defy the purpose of learning based algorithms and tend to go back to manually designed models.

6.1.3 Towards 3D Human Pose Estimation in the Wild: a Weakly-supervised Approach

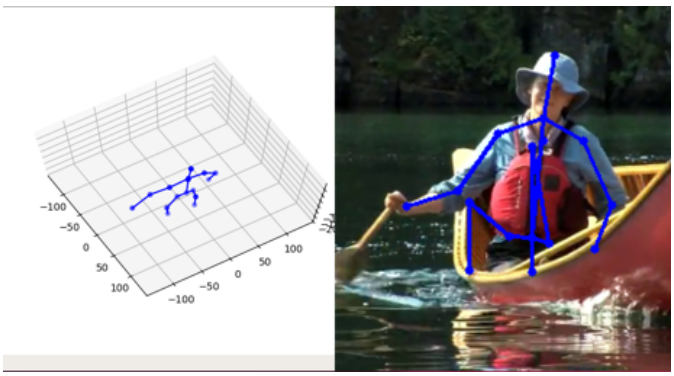
Zhou et al. [42] used a multi-stage learning process for their network to give their network the ability to perform well on in-the-wild data. In Figure 6.4 two outputs of the network can be seen. In Figure 6.4(a) the 2D estimation failed at the legs due to the large occlusions of the boat. This image is from the MPII dataset, on which the network has been trained. The 3D estimation does not predict significant variations in the z-coordinates of the joints. Most joints share a similar depth, and even though the right arm of the man should be closer to the camera, the predicted depth is almost the same as the estimated depth of the hip. In Figure 6.4(b) the network operates on a new random example. The 2D pose module seems to work very well, but again the depth module does not produce reliable outputs. The hands are clearly in front of the head and the hip joints. In the estimated 3D pose however they seem to be almost at equal depth. By comparing the results created during the in-the-wild tests seen in Figure 6.4(b) to the almost perfect results seen in Figure 3.17 a clear difference is notable. It is likely that the network has scaling problems in the z-direction or overfits to the Human3.6M dataset, which was used to train the networks 3D module since the results created using images from the Human3.6M dataset are the only examples of accurately estimated 3D poses. A promising attempt to overcome the overfitting issue would be to train the network on a different 3D pose dataset or to use data augmentation.

6.1.4 3D Human Pose Estimation in the Wild by Adversarial Learning

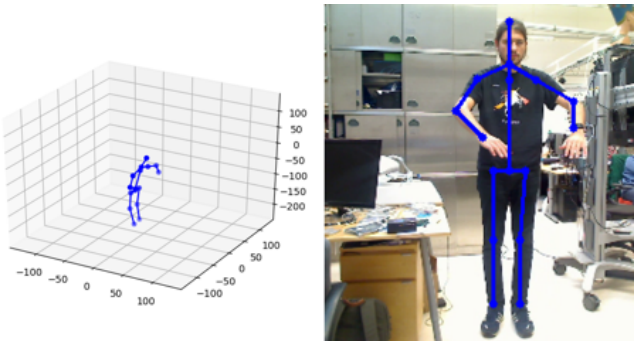
Yang et al. [39] used the network from Zhou et al. [42] as backbone architecture and added an adversarial learning method on-top of it. The backbone architecture has the role of the generator. The results sound promising, could however not be reproduced since no code has been published. We decided against reimplementing it since the in-the-wild results of the backbone paper (section 6.1.3) where not very promising. We do not expect a giant improvement and think that the in-the-wild performance of Yang et al. is therefore are analogous to Zhou et al.

6.1.5 Unsupervised Adversarial Learning of 3D Human Pose from 2D Joint Locations

The framework presented by Kudo et al. [16] is interesting since it can be trained without 3D pose data. This unsupervised training is a huge advantage, and it is effortless to add



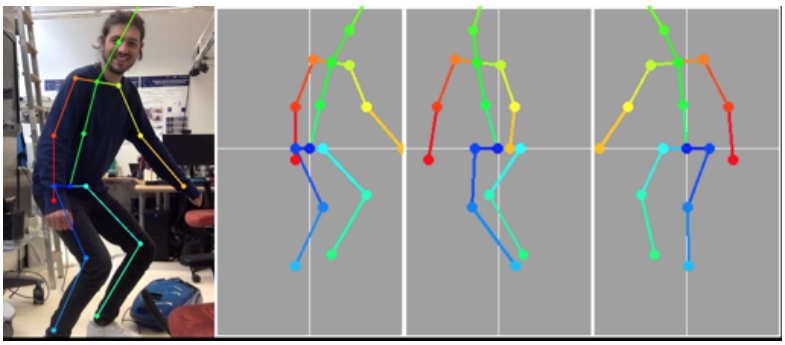
(a) Image from MPII dataset



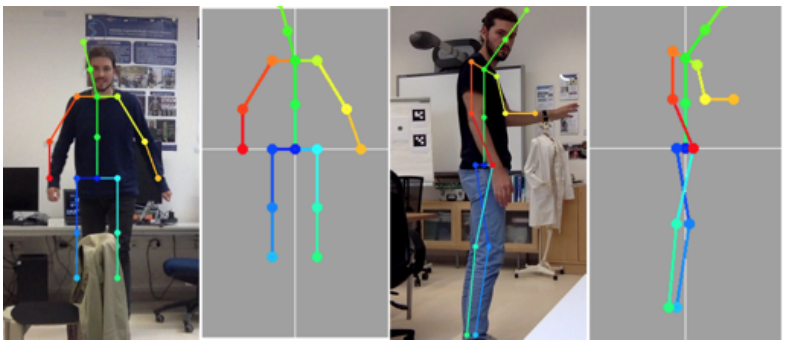
(b) Own example

Figure 6.4: Examples - Towards 3D Human Pose Estimation in the Wild: a Weakly-supervised Approach

training data to the training process since 2D pose data is more available than 3D data. We conducted our experiments with this framework since the idea of having unlimited in-the-wild training data is very appealing. We have been able to produce results with the framework as seen in Figure 6.5(a). Nevertheless, when the 2D estimator failed, or the person did not appear frontal in the image (cf. Figure 6.5(b)) the network failed to recover a valid 3D human pose. Another problem might appear while training the framework. GANs are harder to train than convolutional networks, for instance, the discriminator network can reach a state where it outperforms the generator network so consistently that no further learning takes place. There are some tips on how to train GANs which we followed [5], but our results are still not accurate enough using this method.



(a) Kudo et al. - good Test



(b) Kudo et al. - failed Test

Figure 6.5: Unsupervised Adversarial Learning of 3D Human Pose from 2D Joint Locations - in-the-wild tests

6.1.6 Ordinal Depth Supervision for 3D Human Pose Estimation

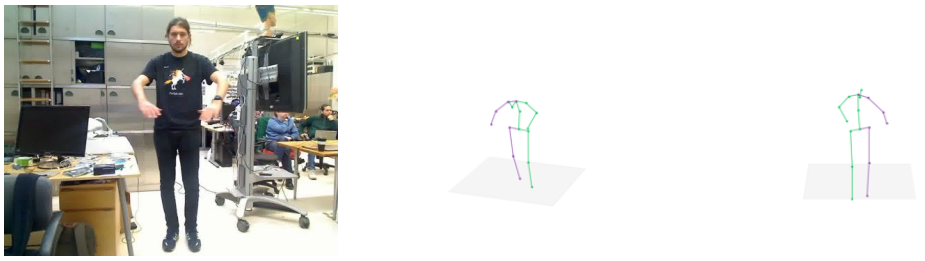
Pavlakos et al. used an additional relationship between pairs of joints, the ordinal depth information, to use 2D training data for 3D training. Ordinal depth information has to be annotated in a manual labeling process. Even though the results of Pavalkos et al. are encouraging we do not have time to label our own training data. Deep networks need a lot of training data to be trained adequately, and for the task of 3D human pose estimation networks tend to get deeper and deeper. Hence it needs much manual effort to label as many images as possible, and this is the main reason we did not consider this framework further.

6.2 Stage 2 - Comparison

In stage 1 we filtered out six of eight methods because of insufficient performance. In stage 2, we compare the remaining two models based on the same in-the-wild recording. Since there is no 3D pose data for this video, we conduct the comparison visually on a qualitative basis.

6.2.1 A simple yet effective baseline for 3d human pose estimation

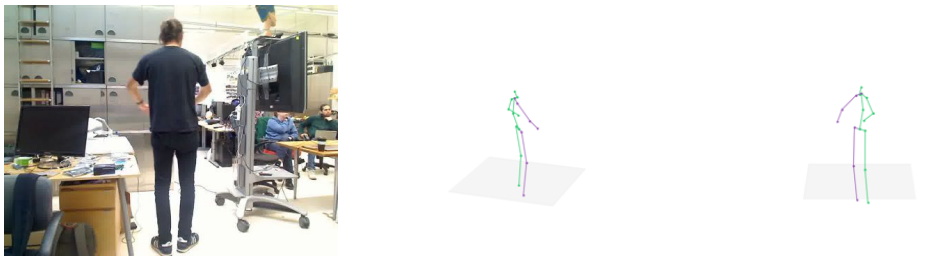
Martinez et al. [17] decoupled the task of determining the 3D human pose estimation into two parts. Part one is finding the 2D pose and part two is lifting the 2D pose into a 3D pose. The used network is rather simple when compared to the other frameworks. An overview of the framework is presented in section 3.2.6. The creators themselves talk about it as baseline framework because they did not focus on optimising the performance but rather on achieving solid accuracy with minimal complexity. We were surprised by the performance of this network in our own experiments (cf. Figure 6.6). Even though the person is turned with the back to the camera, the framework manages to estimate the arm positions fairly accurate as seen in Figure 6.7. The leg joints are also estimated precisely as seen in Figure 6.8. In Figure 6.9 the person is leaning backward, but the framework detected a forward-leaning pose. This error probably occurs due to depth ambiguities. Due to additional dimension the same 2D pose can correspond to several 3D poses.



(a) Frame (b) Pose Front (c) Pose Back

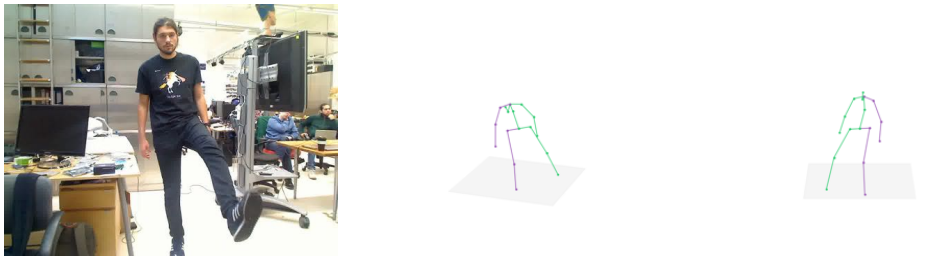
Figure 6.6: Example 1 output baseline 3D Martinez et al.

In the fifth example in Figure 6.10, the person is leaning forward, and in this case, the framework correctly detected the forward leaning person. The last example (Figure 6.11)



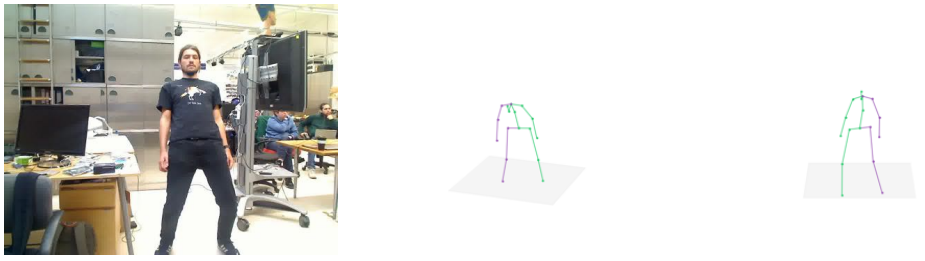
(a) Frame (b) Pose Front (c) Pose Back

Figure 6.7: Example 2 output baseline 3D Martinez et al.



(a) Frame (b) Pose Front (c) Pose Back

Figure 6.8: Example 3 output baseline 3D Martinez et al.

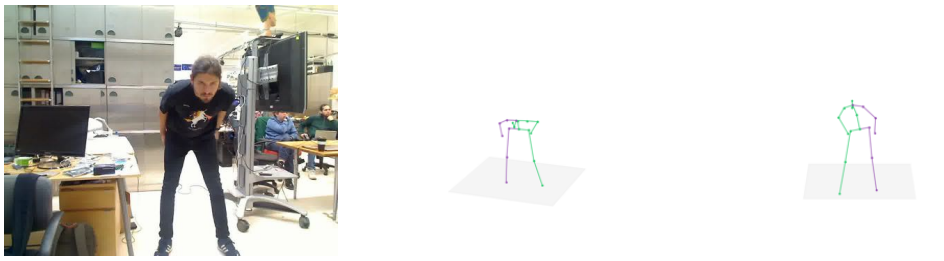


(a) Frame (b) Pose Front (c) Pose Back

Figure 6.9: Example 4 output baseline 3D Martinez et al.

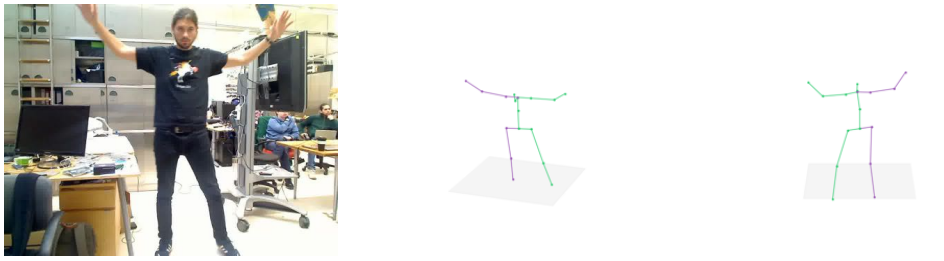
shows the person raising the arms up and again the estimated 3D pose seems similar to the actual posture of the person.

Overall we think that lifting 2D poses to 3D as seen by Martinez et al. is a promising concept to detect in-the-wild 3D poses robustly. Lifting 2D poses to 3D depends on



(a) Frame (b) Pose Front (c) 3D Pose Back

Figure 6.10: Example 5 output baseline 3D Martinez et al.



(a) Frame (b) Pose Front (c) Pose Back

Figure 6.11: Example 6 output baseline 3D Martinez et al.

precise and fast 2D pose estimation. Another limitation of this method is that 3D poses can only be estimated when a full 2D pose is available.

6.2.2 3D human pose estimation in video with temporal convolutions and semi-supervised training

The approach of Pavllo et al.[24] also starts with a 2D pose estimation and lifts it to a 3D pose subsequently. It differs in the fact that not only the current frame is used, but also temporally adjacent frames are regarded. This aims at leveraging trajectories to improve the pose estimation. The method seems to do well even when the person is facing away from the camera seen in Figure 6.13

The positions of the legs is also predicted precisely as seen in Figure 6.14. Differentiating between forward- and backward-leaning poses is still problematic due to depth ambiguities as seen in Figure 6.15. For many use cases this problem will be acceptable

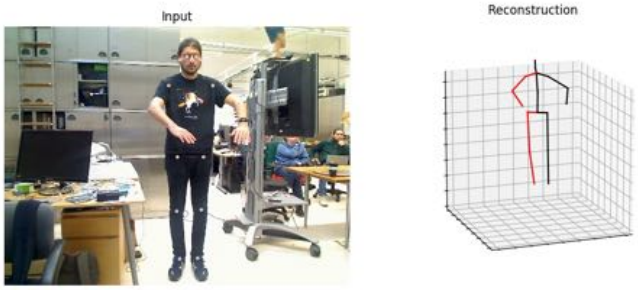


Figure 6.12: Example 1 Pavllo et al. [24]

Under *Input*, the RGB image and the detected 2D joint positions for this frame overlaid as points are shown. Under *Reconstruction* the output of the network, a reconstructed 3D human pose created by utilizing the prior detected 2D poses can be seen.

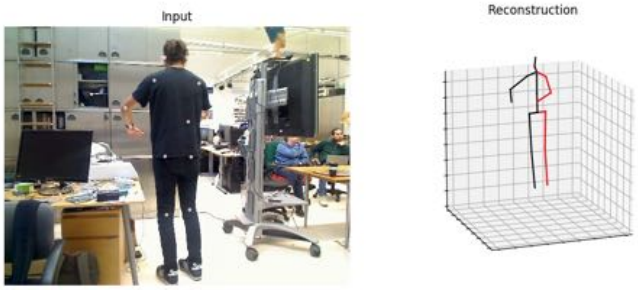


Figure 6.13: Example 2 Pavllo et al. [24]

since such poses are not common. Poses with people leaning forward did not cause any difficulties for the framework (Figure 6.16). Even when parts of the image are blurred through fast motions, the 3D position is predicted accurately due to the good 2D baseline and inclusion of the temporal context. Another accurate example for 3D human pose detection can be seen in Figure 6.17.

6.2.3 Outcome of Stage 2

Martinez et al. [17] have shown the general potential of models that lift 2D to 3D poses. Their presented method only analysis individual frames and does not take context into

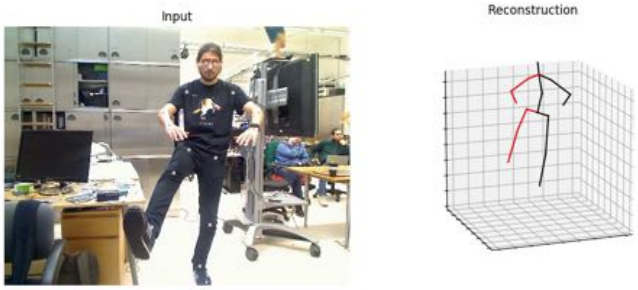


Figure 6.14: Example 3 Pavllo et al. [24]

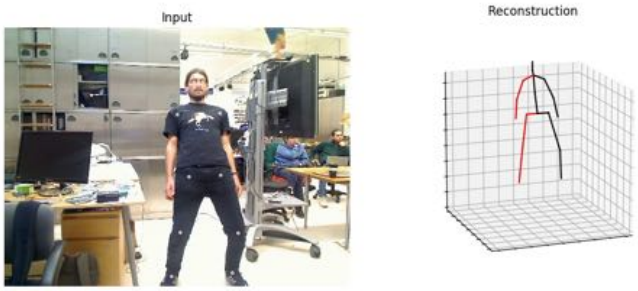


Figure 6.15: Example 4 Pavllo et al. [24]

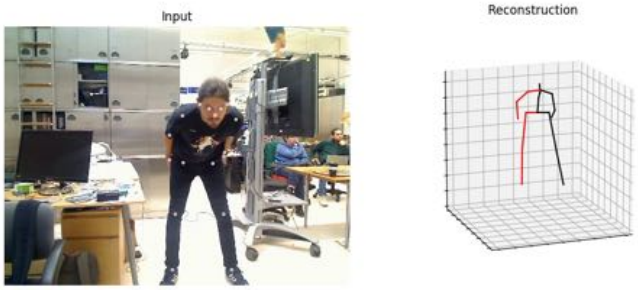


Figure 6.16: Example 5 Pavllo et al. [24]

account. As seen in Figure 6.9 lifting a 2D pose into 3D can lead to errors due to depth ambiguities. For the framework, it is impossible to detect those errors using a single 2D

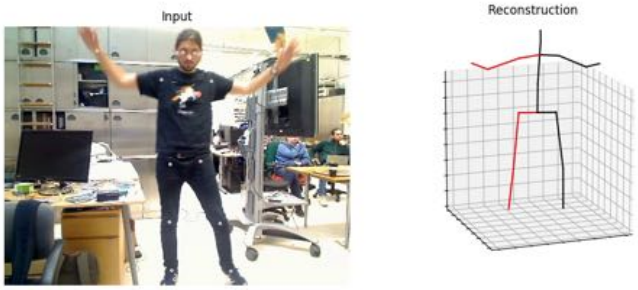


Figure 6.17: Example 6 Pavllo et al. [24]

pose as input. However, it is possible to use temporal context to improve the results. Pavllo et al. [24] utilize this temporal information to enhance their results, and our experiments show that this works well.

Overall the *3D human pose estimation in video with temporal convolutions and semi-supervised training* proposed by Pavllo et al. which the authors also call *VideoPose3D* performed best in our test scenario. The depth detection is accurate and there where almost no obvious outliers in between the frames. To generate the 2D poses from the images, we used Detectron [25], a publicly available object detection framework. We think Detectron is one of the few downsides of this framework. Detectron is a general purpose object detection tool and not specialized on detecting human bodies in images. This is why Detectron needs comparably long to extract a 2D pose out of an image. We could not replace Detectron with a faster 2D detection because a change in the 2D estimator would also mean that we had to retrain our 3D network, since the 2D outputs created by different methods can slightly vary.

In the current setup, it is therefore not possible to run this framework in real-time. We are confident that this method can be used for realtime pose estimation by using a more suitable 2D estimator. We think that of the compared frameworks, the one proposed by Pavllo et al. is most promising and analyse it further in the next section.

6.3 Stage 3 - Detailed Comparison

To compare the VideoPose3D framework with the Kinect, we choose poses that are either especially interesting in practice or that existing frameworks have problems with. We designed and described six movement sequences that cover the defined poses. We conducted our experiment with four different people. The first participant completed all the actions while being filmed frontal by the Kinect camera. To temporally synchronise the sequences between subjects, we used the first recording as instructional video for the other subjects. We, therefore, placed the Kinect camera right on top of the monitor that showed the participants the movements of participant one. This also leads to all participants looking directly towards the Kinect camera which corresponds to the MagicMirror setup (cf. section 2.5). We played the instructional videos of the first participant in mirrored x direction. That way the other participants could follow the actions in the video more easily. We marked the standing position for the recorded subject and did not move the Kinect camera in order to ensure comparable recordings.

Figure 6.18: Experimental setup - The participant on the left has to follow the movements of the person seen on the monitor on the right

Following we explain each action sequence, visualize interesting frames and summarise the performance of Kinect and VideoPose3D in tabular form. The figures are structured as follows: The left part shows the original frame including the 2D pose used as baseline for VideoPose3D (point overlay). In the middle we present the 3D pose as estimated by VideoPose3D and on the right the 3D pose as estimated by the Kinect system. In both 3D pose visualisations red limbs indicate the right side and black limbs indicate the left

side of the person. The writing at hand highlights only interesting frames, the full video sequences can be found on our website [35]. Not every step of an action sequence was used for the study because some steps in the sequence appeared repeatedly for example the *neutral pose*.

6.3.1 Action 1

Action 1 tests pose estimation for recordings from the side. For the Magic Mirror it is important that the 3D estimation is accurate even when the person stands in a 90-degree angle to the camera. Our experience showed that the Kinect system does not perform well in such settings.

Action Sequence

1. Stand straight facing the camera with the arms parallel to your spine - from now on called *neutral position*.
2. Turn 90 degree to your left but keep looking into the camera.
3. Raise both your arms to chest height.
4. Lower your right arm so it is parallel to spine.
5. Also lower your left arm so it is parallel to spine.
6. Rotate 45 degree to your right so you are standing in a 45-degree angle to the camera.
7. Raise both your arms to chest height.
8. Lower your right arm so it is parallel to spine.
9. Lower your left arm so it is parallel to spine.

Vizualization

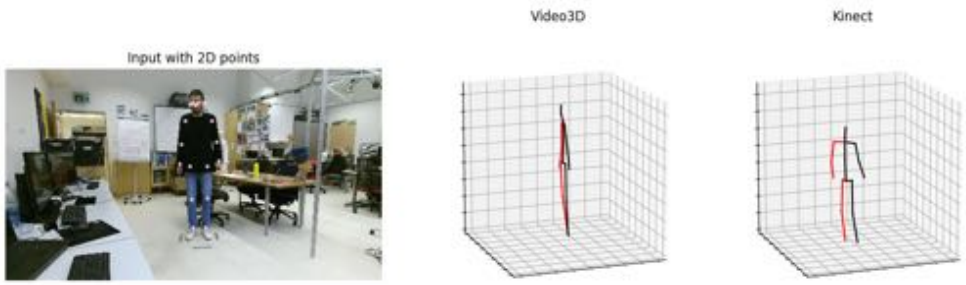


Figure 6.19: Action 1.1 P1

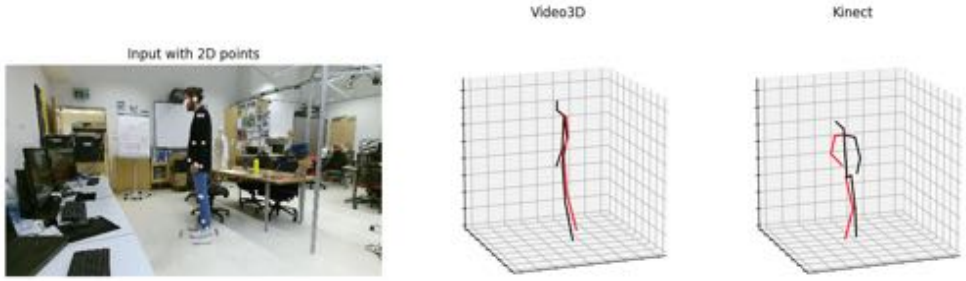


Figure 6.20: Action 1.2 P1

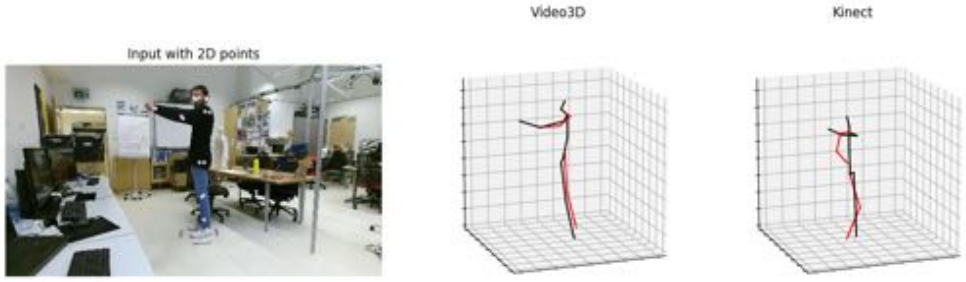


Figure 6.21: Action 1.3 P1

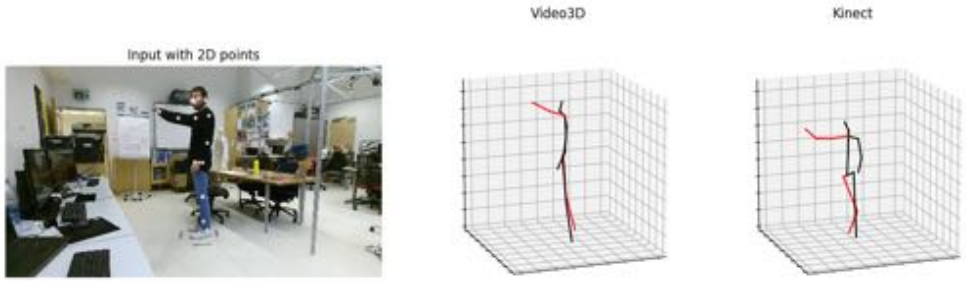


Figure 6.22: Action 1.4 P1

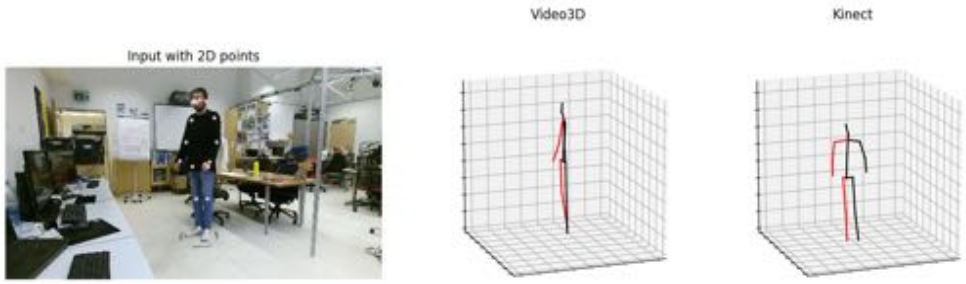


Figure 6.23: Action 1.6 P1

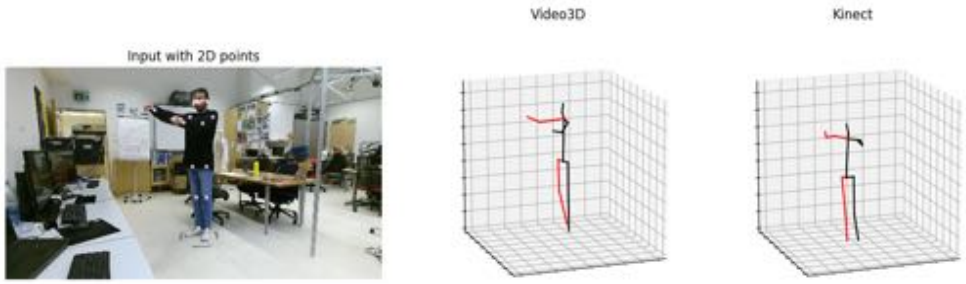


Figure 6.24: Action 1.7 P1

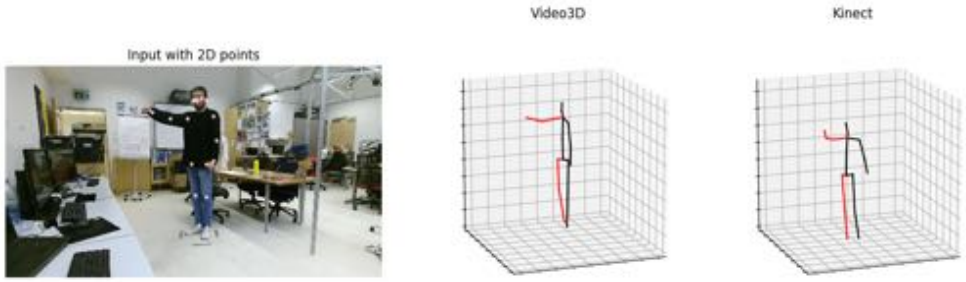


Figure 6.25: Action 1.8 P1

Qualitative results action 1

For the neutral pose in action 1.1, VideoPose3D had issues estimating the width of the shoulders correctly. Because of that VideoPose3D failed three out of four times in action 1.1. The Kinects estimations for the neutral positions was very accurate and it succeeded to estimate all poses correctly. However, the Kinect failed in the following actions 1.2 to 1.4 where VideoPose3D operated accurate enough to pass.

	Participant	1.1	1.2	1.3	1.4	1.6	1.7	1.8
VideoPose3D	1	×	✓	✓	✓	✓	✓	✓
	2	×	✓	✓	✓	✓	✓	✓
	3	✓	✓	✓	✓	✓	✓	✓
	4	×	✓	✓	✓	✓	✓	✓
Kinect	1	✓	×	×	×	✓	✓	✓
	2	✓	×	×	×	✓	✓	✓
	3	✓	×	×	×	✓	✓	✓
	4	✓	×	×	×	✓	✓	✓

Table 6.1: Action 1 - performance

6.3.2 Action 2

In *Action 2* we tested pose estimation with crossed arms. Despite this pose not being essential for practical application, crossed limbs are a challenging situation for a pose estimator and therefore provide an interesting scenario for our comparison.

Action Sequence:

1. Stand in neutral position.
2. Cross your arms in front of your chest.
3. Go back to neutral position.
4. Cross arms again so that the arm that was in front earlier is now in the back.
5. Go back to neutral position.
6. Cross arms again but try to add another variation.

Vizualization

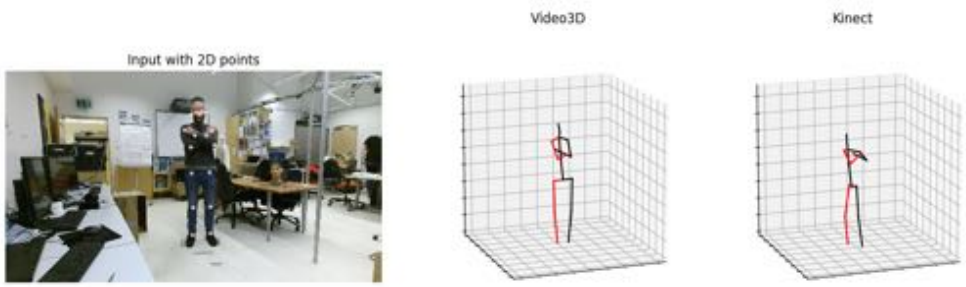


Figure 6.26: Action 2.2 P4

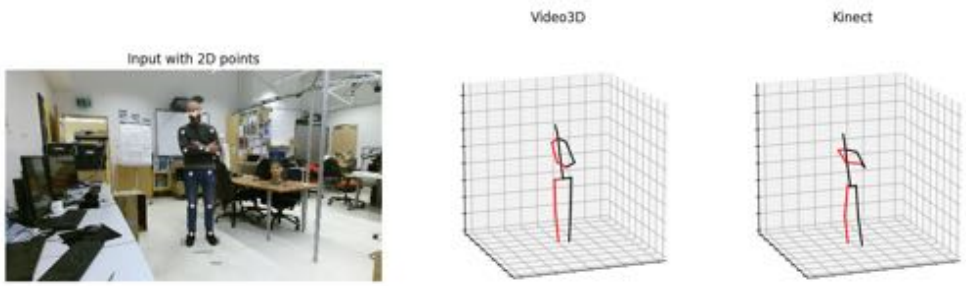


Figure 6.27: Action 2.4 P4

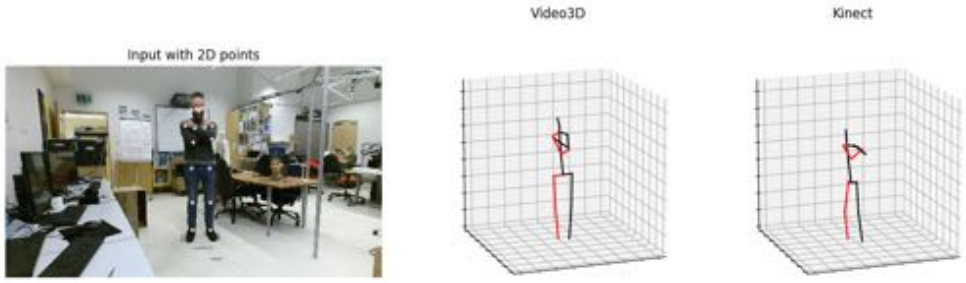


Figure 6.28: Action 2.6 P4

Qualitative results action 2

The crossed arms pose is problematic to detect due to large self occlusions of body parts. Nevertheless, both frameworks produced good results but also had some wrongly detected arm position.

	Participant	2.2	2.4	2.6
VideoPose3D	1	✓	✓	✓
	2	✓	✓	✓
	3	×	✓	✓
	4	✓	×	×
Kinect	1	✓	✓	✓
	2	✓	×	✓
	3	×	✓	×
	4	✓	✓	✓

Table 6.2: Action 2 - performance

6.3.3 Action 3

In *Action 3* we reproduce movements that are used by the user interaction model of the Magic Mirror system. Vertical placement of the hands is commonly used to change viewing modalities.

Action Sequence:

1. Go to the neutral pose.
2. Slowly raise right arm over your head while the hand stays close to body root.
3. Slowly raise right arm over your head while the hand stays close to body root.
4. Go back to neutral position.
5. Turn right foot so the camera sees the inner side of your thigh.
6. Go back to neutral pose.
7. Turn left foot so the camera sees the inner side of your thigh.
8. Go back to neutral pose.
9. Wave wildly with your arms and move your whole body.

Vizualization

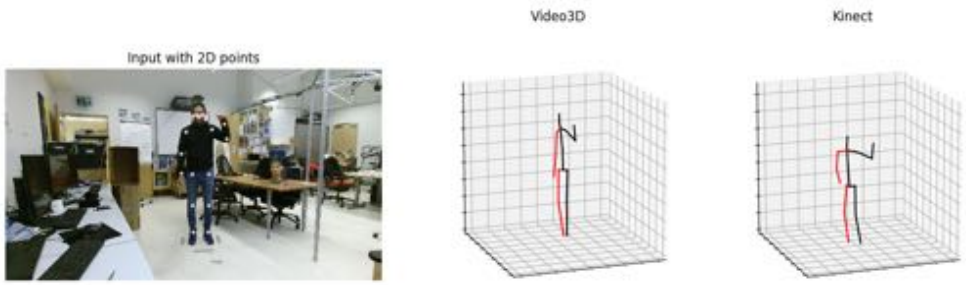


Figure 6.29: Action 3.2 P2

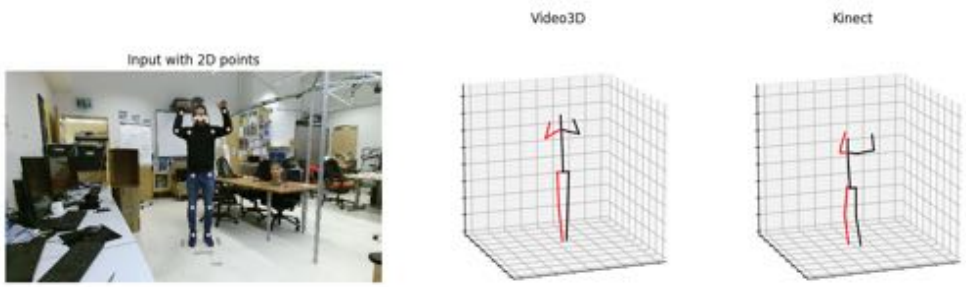


Figure 6.30: Action 3.3 P2

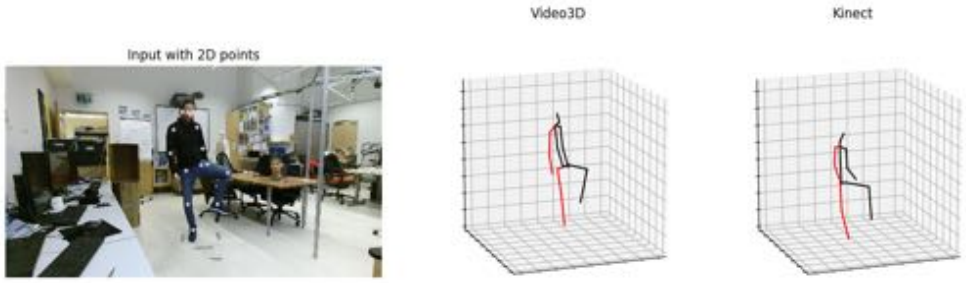


Figure 6.31: Action 3.5 P2

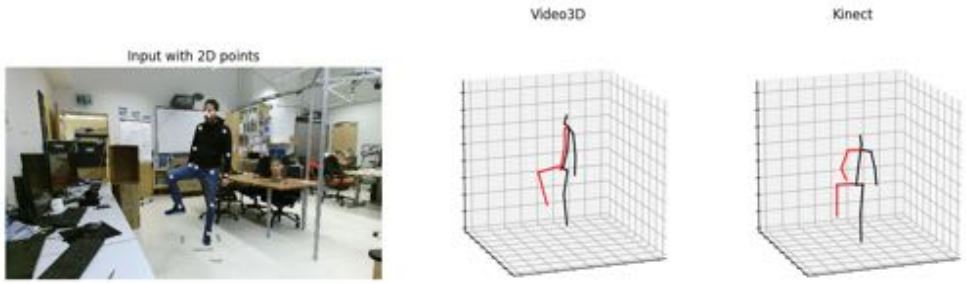


Figure 6.32: Action 3.7 P2

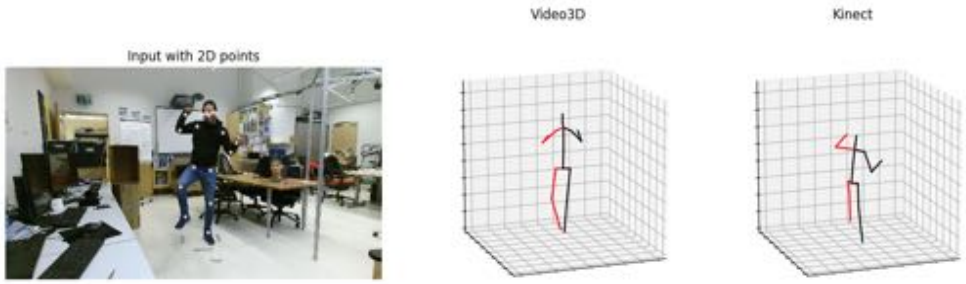


Figure 6.33: Action 3.9 P2

Qualitative results action 3

Both frameworks performed very well, but we noticed that that VideoPose3D had problems in action 3.9 when participants wildly waved their arms it looked like VideoPose3D

was smoothing the movements between frames too much and was therefore not able to react fast enough to detect quick movements.

	Participant	3.2	3.3	3.5	3.7	3.9
VideoPose3D	1	✓	✓	✓	✓	✓
	2	✓	✓	✓	✓	×
	3	✓	✓	✓	✓	×
	4	✓	✓	✓	✓	×
Kinect	1	✓	✓	✓	✓	✓
	2	✓	✓	✓	✓	✓
	3	✓	✓	✓	✓	✓
	4	✓	✓	✓	✓	✓

Table 6.3: Action 3 - performance

6.3.4 Action 4

In *Action 4* we tested the performance of the frameworks in situations where limbs point towards the camera which leads to joints occluding each other.

Action Sequence:

1. Point your right arm to the camera.
2. Go back to neutral pose.
3. Point your left arm to the camera.
4. Go back to neutral pose.
5. Point both arms to the camera.

Vizualization

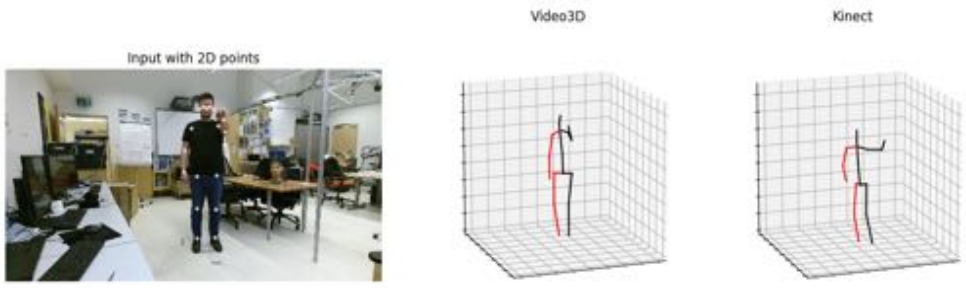


Figure 6.34: Action 4.1 P3

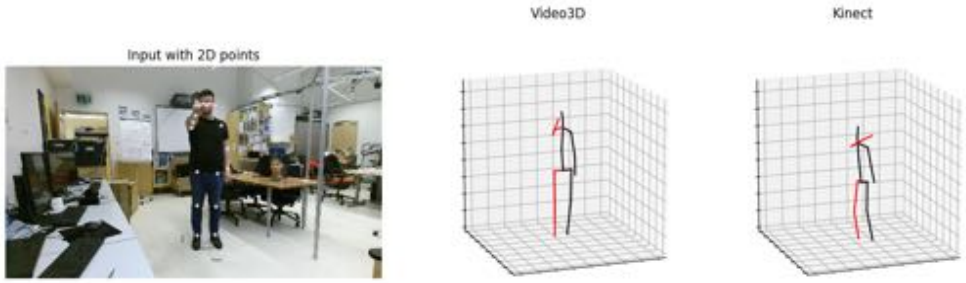


Figure 6.35: Action 4.3 P3

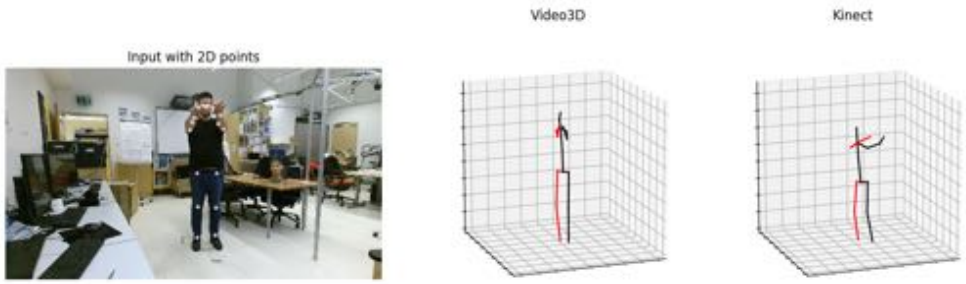


Figure 6.36: Action 4.5 P3

Qualitative results action 4

In this action VideoPose3D detected six poses correctly but failed to detect the remaining six. The Kinect produced only two failures. The integrated depth sensor in the Kinect is

an advantage. Knowing the absolute depth, the system can infer that the first joint is relatively close to the camera and other joints are probably occluded.

	Participant	4.1	4.3	4.5
VideoPose3D	1	✓	×	✓
	2	×	×	×
	3	✓	✓	✓
	4	×	✓	×
Kinect	1	✓	×	✓
	2	✓	✓	✓
	3	✓	✓	✓
	4	×	✓	✓

Table 6.4: Action 4 - performance

6.3.5 Action 5

In *Action 5* we tested pose estimation for sitting positions. Detecting sitting positions is difficult due to self-occlusions and limited training data. Accurate pose estimation in sitting positions might be of practical importance in rehabilitation contexts e.g.

Action Sequence:

1. Sit on the chair and look into the camera in a neutral position.
2. Turn yourself, while remaining at the same position in the room, on the chair 360 degrees until you again face the camera.

Vizualization

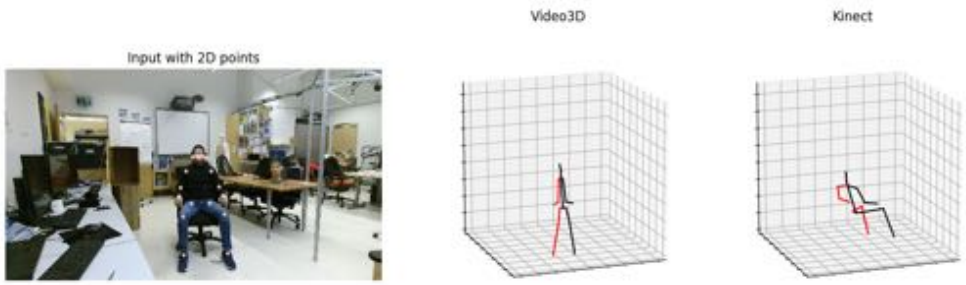


Figure 6.37: Action 5.1 P2

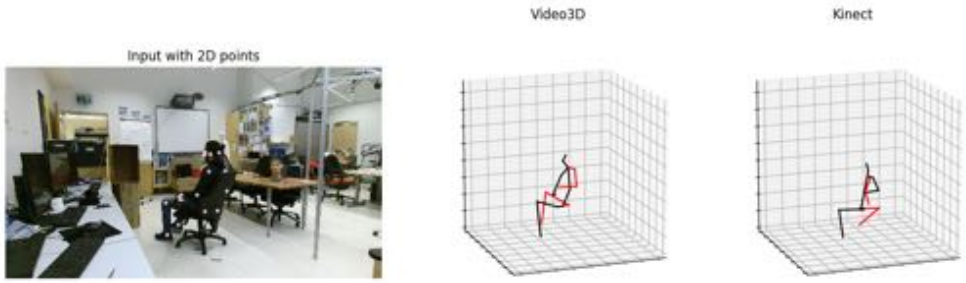


Figure 6.38: Action 5.2 P2

Qualitative results action 5

For action 5.1 the Kinect estimated all poses correctly. The corresponding VideoPose3D poses had similar to the neutral standing pose only a small distance between the left and right shoulder joint and therefore just passed two out of for tests. In action 5.2 the Kinect could not produce a single correct result where VideoPose3D detected all poses correctly.

	Participant	5.1	5.2
VideoPose3D	1	×	✓
	2	✓	✓
	3	×	✓
	4	✓	✓
Kinect	1	✓	×
	2	✓	×
	3	✓	×
	4	✓	×

Table 6.5: Action 5 - performance

6.3.6 Action 6

In *Action 6* we tested pose estimation with larger occlusions.

Action Sequence:

1. Go to neutral pose.
2. Take first paper box and place it in front of you.
3. Stack the second paper box on top of the first box.

Vizualization

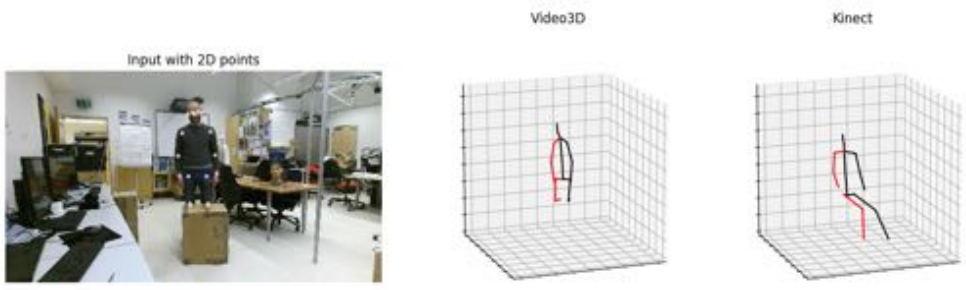


Figure 6.39: Action 6.2 P4

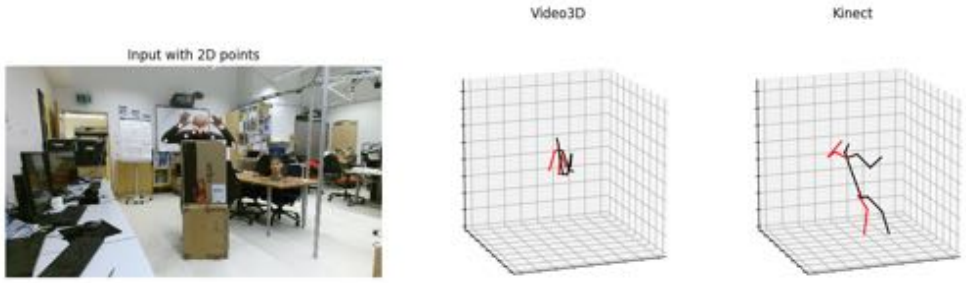


Figure 6.40: Action 6.3 P4

Qualitative results action 6

None of the frameworks could detect a correct pose in action 6. The Kinects pose estimation looks like the participant was sitting and for VideoPose3D the pose collapsed entirely.

	Participant	6.2	6.3
VideoPose3D	1	×	×
	2	×	×
	3	×	×
	4	×	×
Kinect	1	×	×
	2	×	×
	3	×	×
	4	×	×

Table 6.6: Action 6 - performance

6.3.7 Special actions

In this section, we test the limits of the systems with several poses that are inherently difficult due to occlusions and ambiguities.

Cross-legged sitting pose

Cross-legged sitting is a difficult task because it combines a sitting pose with occlusions through body parts. The Kinect performed this task perfectly. VideoPose3D suffered from the reappearing problem of the collapsing should joints from action 1.1.

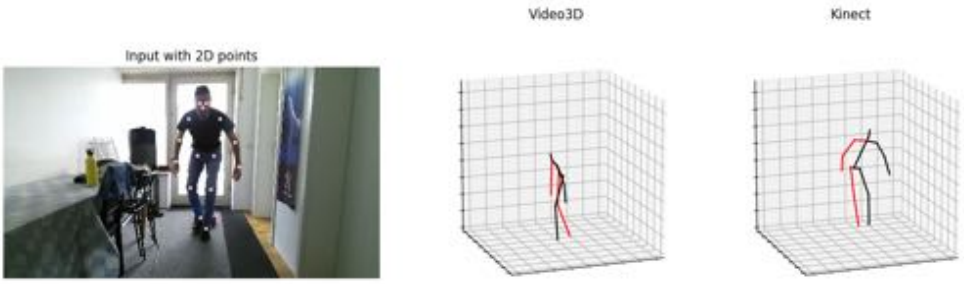


Figure 6.41: cross-legged sitting pose 1

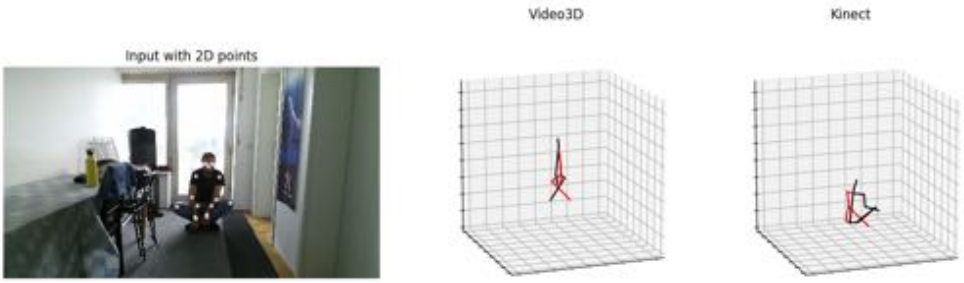


Figure 6.42: cross-legged sitting pose 2

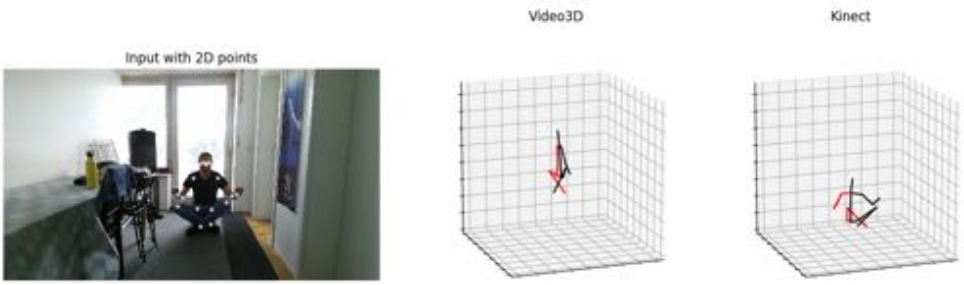


Figure 6.43: cross-legged sitting pose 3

Outdoors estimation

Since the Kinect system uses infrared lights, outdoor operation is problematic due to the amount of noise. This is an area where purely-RGB based pose estimators have the advantage. As expected VideoPose3D did a significantly better job than Kinect in this action.

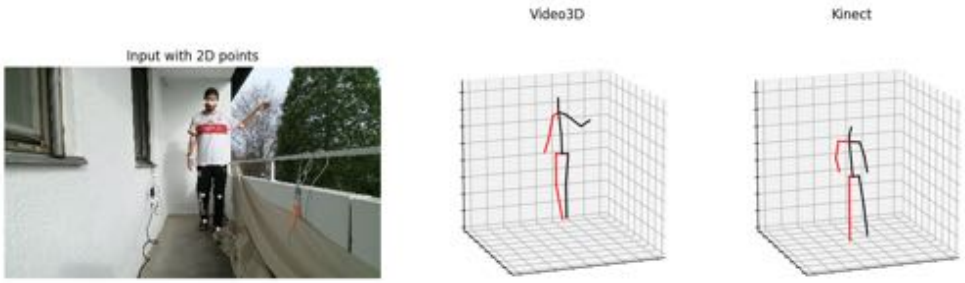


Figure 6.44: outdoors estimation 1

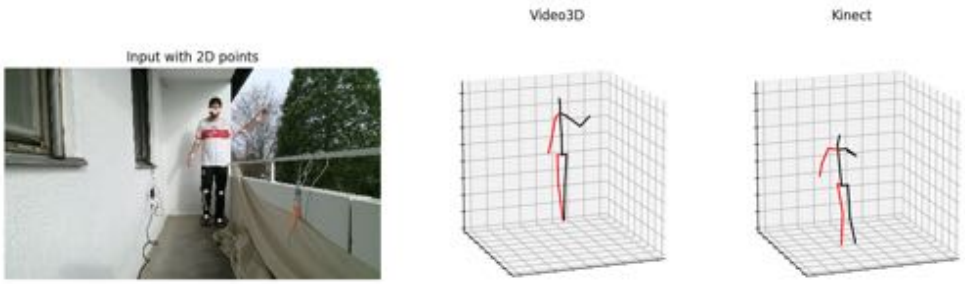


Figure 6.45: outdoors estimation 2

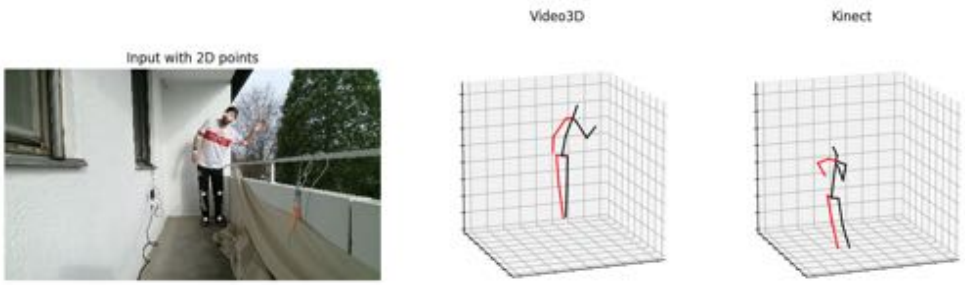


Figure 6.46: outdoors estimation 3

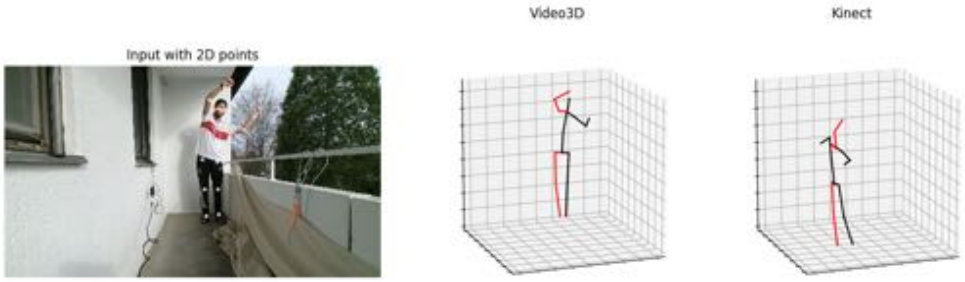


Figure 6.47: outdoors estimation 4

Headstand pose

A headstand is challenging because most algorithms today are learning based algorithms. The datasets used for training can be very versatile but they usually do not contain examples of people doing headstand. The Kinect totally failed and estimated the head of the participant between the legs. VideoPose3D had difficulties but at least managed to estimate the body orientations correctly. The reason for the good performance of VideoPose3D is probably the surprisingly good 2D detections produced by Detectron.

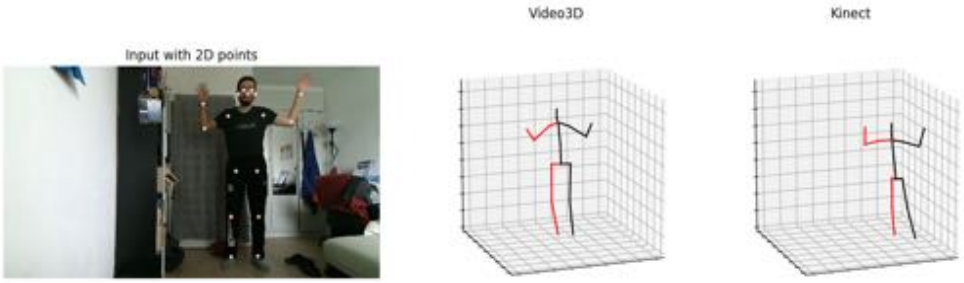


Figure 6.48: headstand pose 1

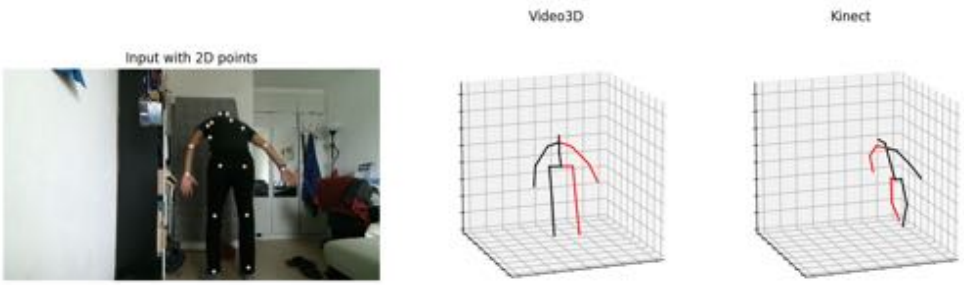


Figure 6.49: headstand pose 2

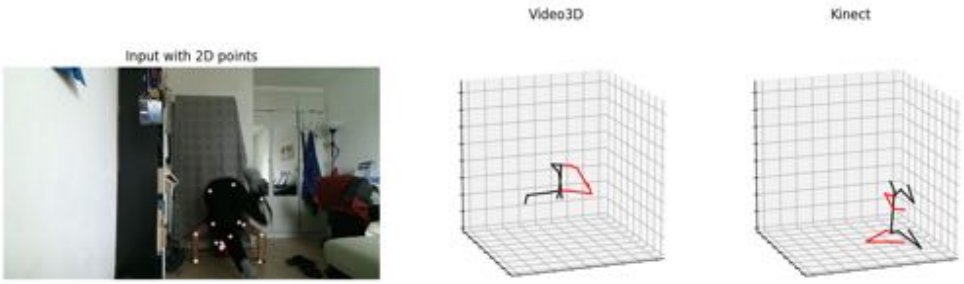


Figure 6.50: headstand pose 3

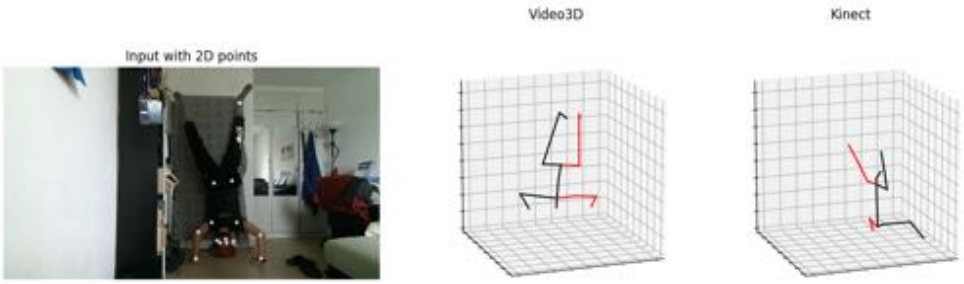


Figure 6.51: headstand pose 4

7 Results

In this section, we analyze the results obtained in section 6. After initial preselection, we ended up comparing VideoPose3D with the Kinect using a defined set of reference poses. We classified the results either with a ✓ when the they matched the RGB image or with a × if the estimation was wrong. In section 7.0.1 we summarized the results.

The Kinect is a solid 3D human pose estimation system and produces convincible results especially for poses that appear very frequently, e.g. the neutral pose. Poses that are 90° rotated to the view of the camera could not be detected correctly. The Kinects problem with lateral perspectives limits the application area. Further limits are outdoor settings where the sunlight interferes with the infrared depth detection module.

VideoPose3D had no issues detecting poses from a lateral perspective but interestingly had problems with correct body proportions. In basic scenes the system produced poses where the distance between the left and right shoulder joints collapsed. Also very quick movements could not be precisely detected and it looked like the estimated poses are a smoothed version of the real ones. Outdoor environments however are no problem for VideoPose3D since it operates on RGB data only.

After our detailed comparison, we counted 59 passes and 29 fails for the Kinect. For VideoPose3D we counted 63 passes and 25 fails. This means that the accuracy of the VideoPose3D system is sufficient for most AR-Systems that are currently using the Kinect. VideoPose3D, the method introduced by Pavllo et al, compares well or even outperforms Kinect, despite its additional depth sensing camera. This result is especially promising for devices that do not have the right hardware because of available space or cost.

Overall, VideoPose3D achieves a solid performance that is sufficient for many AR applications.

7.0.1 Result data

	Participant	1.1	1.2	1.3	1.4	1.6	1.7	1.8	2.2	2.4	2.6	3.2	3.3	3.5	3.7	3.9
VideoPose3D	1	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	2	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×
	3	✓	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	×
	4	×	✓	✓	✓	✓	✓	✓	✓	×	×	✓	✓	✓	✓	×
Kinect	1	✓	×	×	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	2	✓	×	×	×	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓
	3	✓	×	×	×	✓	✓	✓	×	✓	×	✓	✓	✓	✓	✓
	4	✓	×	×	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

	Participant	4.1	4.3	4.5	5.1	5.2	6.2	6.3
VideoPose3D	1	✓	×	✓	×	✓	×	×
	2	×	×	×	✓	✓	×	×
	3	✓	✓	✓	×	✓	×	×
	4	×	✓	×	✓	✓	×	×
Kinect	1	✓	×	✓	✓	×	×	×
	2	✓	✓	✓	✓	×	×	×
	3	✓	✓	✓	✓	×	×	×
	4	×	✓	✓	✓	×	×	×

Table 7.1: Combined action performance

8 Discussion

The first interesting result to discuss is that the only two models that passed our preselection phase are Simple Baseline and VideoPose3D. Both methods approach 3D pose estimation by lifting a preliminary 2D pose estimate produce by an upstream model.

Only relying on a 2D pose as input means narrowing down the features for estimation to 17 2D-coordinates. Comparing 17 2D-coordinates with a full HD image of dimensionality 1920 x 1080 x 3, it is evident that those estimation systems omit a lot of information. Especially for poses where the 2D position can lead to two plausible 3D poses, this information might be beneficial to determine the actual pose. Deep end-to-end models perform well on the data sets they have been trained on but do not generalise well to in-the-wild data. Performance of deep learning-based 3D pose estimators is currently severely limited by a lack of training data. For a 3D human pose dataset, some of the requirements are diversity in the images, different camera positions and different focal lengths. It would be helpful if we could use images from the internet and acquire corresponding ground truth depth information. However, in contrast to 2D pose data sets, the annotation of 3D pose data requires expensive motion capture (MoCap if introduced earlier) or multi-camera systems. Besides inhibiting the creation of labeled 3D data sets, this puts natural limits on the variety that can be obtained in such data sets. There are few notable exceptions like the MPI-INF-3DHP dataset (cf. 4.1.3). As long as there is not enough in-the-wild 3D human pose estimation data to train end-to-end algorithms on, the best system might remain the systems that lift the pose from 2D to 3D.

Once a new 3D human pose estimation dataset is released that is large and versatile enough, end-to-end DNN models might possibly outperform these lifting models.

The goal of this thesis was to find a real-time 3D human pose estimation system. We have not been able to achieve this goal yet, but we have identified and validated a method which has the potential to do so. The VideoPose3D source-code has a lot of potential to be streamlined and improved. Further work is necessary to make the VideoPose3D system operate in a real-time setting. Most importantly, the current underlying Detectron [25] 2D pose estimator is slow and should be replaced by a more suitable algorithm. The

most important part is to replace the slow 2D human pose detector, Detectron with a faster more suitable algorithm. Replacing the underlying 2D pose estimator requires the rest of the model to be trained anew.

9 Outlook

After establishing that VideoPose3D can be used as a replacement for the Kinect in the previous section, we integrated the VideoPose3D poses with the Magic Mirror software.

9.1 Interpolation of missing joints

The Magic Mirror software was developed for the Kinect and therefore also requires the complete Kinect skeleton which consists of 25 joints. VideoPose3D only estimates 17 joints and the additional joints locations must be derived from the existing joint information. VideoPose3D does not estimate the right and the left hand, so we interpolated those points by calculating the vector from the elbow to the wrist and extended this vector by a fixed factor to get an approximate hand position. The feet positions cannot simply be extrapolated because the Kinect places them close to the toes, which means a displacement from the axis of the lib. We approximated the location of the toes using the ankle positions and an auxiliary vector which is perpendicular to the spine.

9.2 Pose transformation

Subsequently, we adjusted the pose estimate of VideoPose3D to match the scale of the Kinect skeleton using the Kabsch algorithm [15]. The Kabsch algorithm is a rigid body transformation that minimizes the root mean square deviation between two sets of points. The algorithm produces a rotation matrix, translation vector and scaling factor which we applied to the VideoPose3D coordinates. The MagicMirror output using both the VideoPose3D and the Kinect pose estimates can be seen side-by-side in Figure 9.1.

Overall the results are promising. Even with more difficult poses as shown in Figure 9.2, the Magic Mirror visualization based on VideoPose3D is very close to the one based on Kinect. These results confirm that the VideoPose3D method can indeed replace the Kinect. We still have to improve the system further, but we are confident to bring VideoPose3D and the Magic Mirror together in the near future.

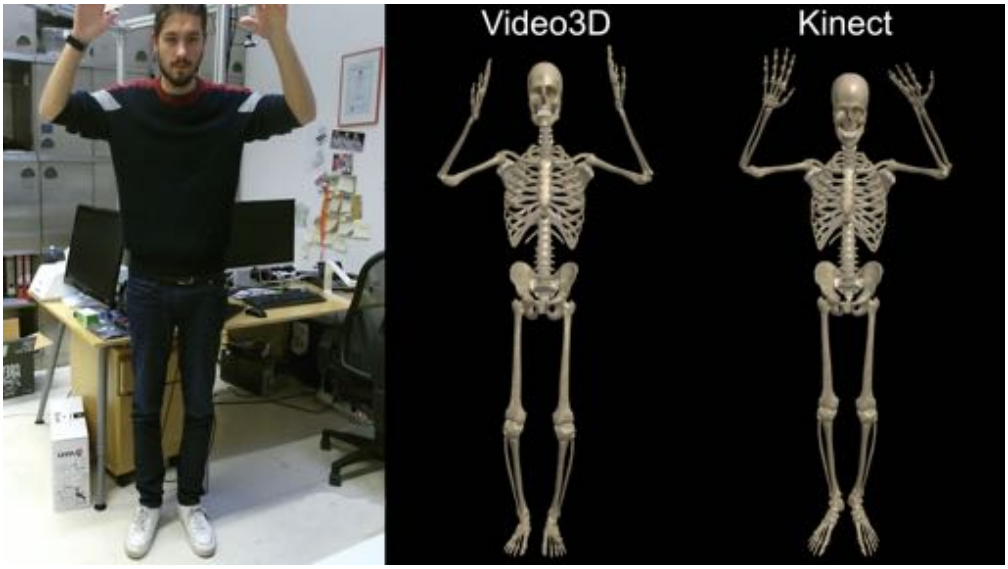


Figure 9.1: Magic Mirror visualization 1

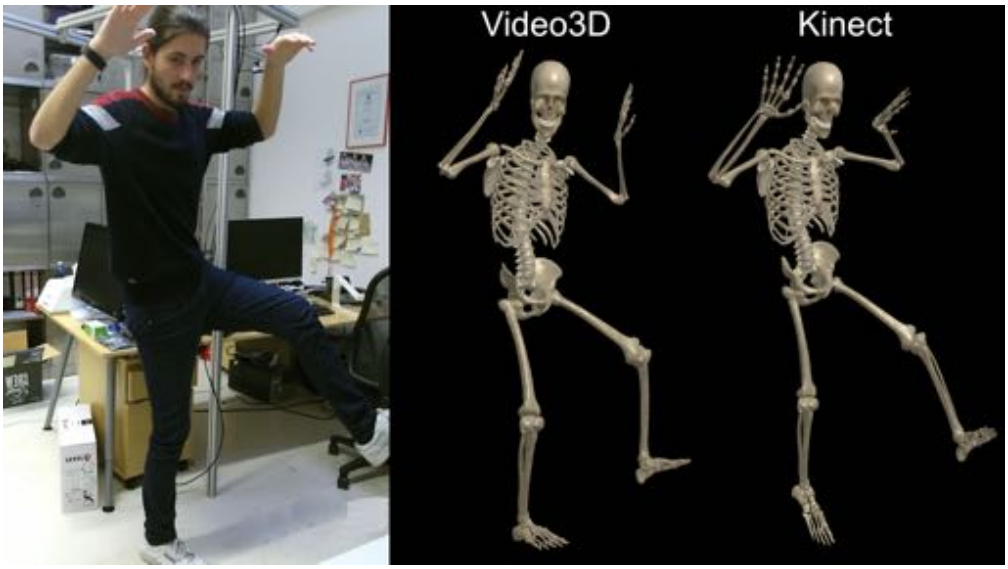


Figure 9.2: VMagic Mirror visualization 1

10 Conclusion

In this thesis, we introduced 2D and 3D human pose estimation. The Magic Mirror system was described as exemplary AR-application. After defining the requirements of the Magic Mirror system we summarized the current state of the art and techniques for 3D pose estimation that rely on additional cameras or markers. We identified eight promising publications that met the requirements of the Magic Mirror AR system. We compared the reported performances of the selected methods and highlighted the authors' arguments of why their methods work well on in-the-wild data. We then tried to reproduce the reported results on in-the-wild data using a 3-stage comparison (cf. chapter 6). In the first stage we discarded six frameworks due to bad performance or other unacceptable limitations. We then compared the remaining two frameworks and selected VideoPose3D by Pavllo et al. [24] as the most promising one. Finally, in the third stage we compared VideoPose3D with the Kinect system, which is currently the standard solution for most screen-based AR applications. For this comparison, we designed six action sequences that represent a wide range of relevant poses. Our action sequences contain a total of 23 actions based on which we compared the two systems. We recorded four subjects performing these action sequences with the Kinect camera, thereby obtaining both raw image data and 3D pose information. In the next step, we estimated the 3D pose on the same videos using the VideoPose3D algorithm. We qualitatively compared the pose estimates of VideoPos3D with those of the Kinect. We found that VideoPose3D performed well and could even outperform the Kinect in some situations. We conclude this thesis by affirming that DNN-based 3D pose estimators perform well enough to replace the Kinect, a specialized and discontinued device, in screen-based augmented reality applications. This is an essential result which has the potential to widely increase the availability of such systems, from mobile devices to simple download-and-go software packages.

Acknowledgments

I thank everyone who helped me to write this thesis. Thanks to all my friends and family that kept me motivated. Thanks to the whole team at NARVIS for your support and interest. Thank you, Coco Hannemann, for your love and for keeping me focused. Thank you, Felix Bork, for supervising my thesis while staying calm and helpful. And a special thanks to, Yawar Siddiqui and Jeffrey Jedele for your technical and scientific support while composing this Masterthesis.

Glossary

Computer vision scientific field that deals with how computers can get high-level understanding of a camera image or stream

convolutional layer core building block of a convolutional neural network. The layers consist of learnable filters. During the forward pass each filter is convolved with the input, creating an activation map

convolutional neural network is a class of deep neural networks that is commonly used to analyse images

deconvolutional layer also known as transposed convolutional layers are layers that perform an inverse convolution which reverses the effect of a convolutional layer

deep neural network is a computing system that is able to learn tasks by considering examples. Deep Neural Network have multiple hidden layers between the input and the output.

end-to-end training is training done as full pipelines without any intermediate hand crafted steps

fully connected layer connects every neuron in one layer to every neuron in the next layer

HoloLens a mixed-reality glass developed by Microsoft

in-the-wild collected under unconstrained conditions

loss layer specifies how the deviation between the prediction and the correct labels penalizes. Different loss functions with various properties are used for various learning tasks

NARVIS navigated augmented reality visualization system Lab

pooling is used to reduce the spatial dimension but not the depth in a convolutional neural network

ReLu is a non-saturating activation function that removes negative values from activation map by setting them to zero

residual neural network use skip connections to skip some of the networks layers. This helps to avoid the vanishing gradient problem

Acronyms

2D 2-dimensional

3D 3-dimensional

AR Augmented-Reality

CNN convolutional neural network

DNN deep neural network

fps frames per second

GAN generative adversarial network

MoCap Motion Capturing

ReLU rectified linear unit

ResNet residual neural network

RGB red, green and blue

TUM Technische Universität München

List of Figures

List of Tables

Bibliography

[1] A. Agarwal and B. Triggs. "3D human pose from silhouettes by relevance vector regression." en. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Volume 2. Washington, DC, USA: IEEE, 2004, pages 882–888.

[2] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. "2D Human Pose Estimation: New Benchmark and State of the Art Analysis." In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA: IEEE, June 2014, pages 3686–3693.

[3] T. Blum, V. Kleeberger, C. Bichlmeier, and N. Navab. "mirracle: An augmented reality magic mirror system for anatomy education." en. In: *2012 IEEE Virtual Reality (VR)*. Costa Mesa, CA, USA: IEEE, Mar. 2012, pages 115–116.

[4] F. Bork, R. Barmaki, U. Eck, P. Fallavollita, B. Fuerst, and N. Navab. "Exploring Non-Reversing Magic Mirrors for Screen-Based Augmented Reality Systems." In: *arXiv:1611.03354 [cs]* (Nov. 2016). arXiv: 1611.03354.

[5] S. Chintala. "How to Train a GAN? Tips and tricks to make GANs work." In: *https://github.com/soumith/ganhacks* (2016).

[6] H. Coskun, D. J. Tan, S. Conjeti, N. Navab, and F. Tombari. "Human Motion Analysis with Deep Metric Learning." In: *arXiv:1807.11176 [cs]* (July 2018). arXiv: 1807.11176.

[7] H. Fang, Y. Xu, W. Wang, X. Liu, and S.-C. Zhu. "Learning Pose Grammar to Encode Human Body Configuration for 3D Pose Estimation." In: *arXiv:1710.06513 [cs]* (Oct. 2017). arXiv: 1710.06513.

[8] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. "Progressive search space reduction for human pose estimation." In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. Anchorage, AK, USA: IEEE, June 2008, pages 1–8.

[9] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Networks." In: *arXiv:1406.2661 [cs, stat]* (June 2014). arXiv: 1406.2661.

[10] R. A. Güler, N. Neverova, and I. Kokkinos. "DensePose: Dense Human Pose Estimation In The Wild." In: *arXiv:1802.00434 [cs]* (Feb. 2018). arXiv: 1802.00434.

[11] N. Hasler, M. Richter, S. Dimitrov, and C. Theobalt. "thecaptury - markerless motion capture." In: *http://thecaptury.com* (2016).

[12] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition." In: *arXiv:1512.03385 [cs]* (Dec. 2015). arXiv: 1512.03385.

[13] "Intel RealSense." In: *https://realsense.intel.com* (2018).

[14] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. "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 (July 2014), pages 1325–1339.

[15] W. Kabsch. "A solution for the best rotation to relate two sets of vectors." In: *Acta Crystallographica Section A* 32.5 (Sept. 1976), pages 922–923.

[16] Y. Kudo, K. Ogaki, Y. Matsui, and Y. Odagiri. "Unsupervised Adversarial Learning of 3D Human Pose from 2D Joint Locations." In: *arXiv:1803.08244 [cs]* (Mar. 2018). arXiv: 1803.08244.

[17] J. Martinez, R. Hossain, J. Romero, and J. J. Little. "A simple yet effective baseline for 3d human pose estimation." In: *arXiv:1705.03098 [cs]* (May 2017). arXiv: 1705.03098.

[18] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt. "Monocular 3D Human Pose Estimation In The Wild Using Improved CNN Supervision." en. In: *arXiv:1611.09813 [cs]* (Nov. 2016). arXiv: 1611.09813.

[19] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt. "VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera." In: *ACM Transactions on Graphics* 36.4 (July 2017). arXiv: 1705.01583, pages 1–14.

[20] A. Menache. *Understanding motion capture for computer animation*. 2nd ed. OCLC: ocn641537758. Burlington, MA: Morgan Kaufmann, 2011.

[21] A. Newell, K. Yang, and J. Deng. "Stacked Hourglass Networks for Human Pose Estimation." en. In: *arXiv:1603.06937 [cs]* (Mar. 2016). arXiv: 1603.06937.

[22] W. ning, C. Hong-ming, and J. Li-hong. "A dynamical adjustment partitioning algorithm for distributed virtual environment systems." en. In: *Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry - VRCAI '08*. Singapore: ACM Press, 2008, page 1.

[23] G. Pavlakos, X. Zhou, and K. Daniilidis. "Ordinal Depth Supervision for 3D Human Pose Estimation." In: *arXiv:1805.04095 [cs]* (May 2018). arXiv: 1805.04095.

[24] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli. "3D human pose estimation in video with temporal convolutions and semi-supervised training." In: *arXiv:1811.11742 [cs]* (Nov. 2018). arXiv: 1811.11742.

[25] I. Radosavovic and R. Girshik. "Detectron object detection facebook AI Research." In: *https://github.com/facebookresearch/Detectron* (2019).

[26] D. Ramanan. "Learning to parse images of articulated bodies." en. In: *Advances in Neural Information Processing Systems 19* (), pages 1129–1136.

[27] H. Rhodin, M. Salzmann, and P. Fua. "Unsupervised Geometry-Aware Representation for 3D Human Pose Estimation." In: *arXiv:1804.01110 [cs]* (Apr. 2018). arXiv: 1804.01110.

[28] G. Rogez, P. Weinzaepfel, and C. Schmid. "LCR-Net++: Multi-person 2D and 3D Pose Detection in Natural Images." en. In: *arXiv:1803.00455 [cs]* (Mar. 2018). arXiv: 1803.00455.

[29] Sarafianos, Boteanu, Ionescu, and Kakadiaris. "3D human pose estimation: A review of the literature and analysis of covariates." In: (Aug. 2016).

[30] Y. Shi, X. Han, N. Jiang, K. Zhou, K. Jia, and J. Lu. "FBI-Pose: Towards Bridging the Gap between 2D Images and 3D Human Poses using Forward-or-Backward Information." In: *arXiv:1806.09241 [cs]* (June 2018). arXiv: 1806.09241.

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

[32] L. Sigal, A. O. Balan, and M. J. Black. "HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion." en. In: *International Journal of Computer Vision* 87.1-2 (Mar. 2010), pages 4–27.

[33] V. Spruyt. "'Curse of Dimensionality' - The Curse of Dimensionality in classification." In: *http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/* (Apr. 2014).

[34] C. Theobalt. *VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera - SIGGRAPH2017*. Feb. 2017.

[35] C. Tobias. "Results study - 3D Human Pose Estimation with a single RGB-camera." In: *https://github.com/tobiascz/Results3DHumanPoseEstimation* (2019).

[36] A. Toshev and C. Szegedy. "DeepPose: Human Pose Estimation via Deep Neural Networks." In: *2014 IEEE Conference on Computer Vision and Pattern Recognition* (June 2014). arXiv: 1312.4659, pages 1653–1660.

[37] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. "Convolutional Pose Machines." In: *arXiv:1602.00134 [cs]* (Jan. 2016). arXiv: 1602.00134.

[38] B. Xiao, H. Wu, and Y. Wei. "Simple Baselines for Human Pose Estimation and Tracking." In: *arXiv:1804.06208 [cs]* (Apr. 2018). arXiv: 1804.06208.

[39] W. Yang, W. Ouyang, X. Wang, J. Ren, H. Li, and X. Wang. "3D Human Pose Estimation in the Wild by Adversarial Learning." In: *arXiv:1803.09722 [cs]* (Mar. 2018). arXiv: 1803.09722.

[40] D. Zhang and M. Shah. "A Framework for Human Pose Estimation in Videos." In: *arXiv:1604.07788 [cs]* (Apr. 2016). arXiv: 1604.07788.

[41] K. Zhou, J. Cai, Y. Li, Y. Shi, X. Han, N. Jiang, K. Jia, and J. Lu. "Adversarial 3D Human Pose Estimation via Multimodal Depth Supervision." In: *arXiv:1809.07921 [cs]* (Sept. 2018). arXiv: 1809.07921.

[42] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei. "Towards 3D Human Pose Estimation in the Wild: a Weakly-supervised Approach." In: *arXiv:1704.02447 [cs]* (Apr. 2017). arXiv: 1704.02447.