DEEP 3D HUMAN POSE ESTIMATION UNDER PARTIAL BODY PRESENCE

Saeid Vosoughi

A thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science (Electrical and Computer Engineering)

Concordia University

Montréal, Québec, Canada

November 2018

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Saeid Vosoughi**

Entitled: **Deep 3D Human Pose Estimation under Partial Body Presence**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Thomas G. Fevens

_____ External Examiner
Dr. Thomas G. Fevens

_____ Examiner
Dr. Hassan Rivaz

_____ Supervisor
Dr. Maria A. Amer

Approved by _____
Dr. William E. Lynch, Chair
Department of Electrical and Computer Engineering

_____ 2018

_____
Dr. Amir Asif, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Deep 3D Human Pose Estimation under Partial Body Presence

Saeid Vosoughi

3D human pose estimation is estimating the position of the main body joints in the 3D space from 2D images. It remains a challenging problem despite being well studied in computer vision domain. This stems from the ambiguity caused by capturing 2D imagery from 3D objects and thus the loss of depth information. 3D human pose estimation is especially challenging when not all the human body is present (visible) in the input 2D image. This work proposes solutions to reconstruct the 3D human pose from a 2D image under partial body presence. Partial body presence includes all the cases in which some of the body's main joints do not fall inside the image. We propose two different deep learning based approaches to address partial body presence: 1) 3D pose estimation from 2D poses estimated from the 2D input image and 2) 3D pose estimation directly from the 2D input image. In both approaches, we use Convolutional Neural Networks (CNN) for regression. These networks are designed and trained to work under partial body presence but output the full 3D human pose (i.e., including not visible joints). In addition, we propose a detection CNN network to detect those joints present in the input image. We then propose to integrate both regression and detection networks so to estimate the partial 3D human pose, in addition to the full 3D human pose estimated by the regression network. Experimental results comparing the performance of the state-of-the-art demonstrate the effectiveness of our approaches under partial body presence. Experimental results also show that the direct regression of the 3D human pose from 2D images yields more accurate estimation compared to having 2D pose estimation as an intermediate stage.

Acknowledgments

I wish to express my deep and sincere gratitude and appreciation to my supervisor, Dr. Maria A. Amer, who has made this work possible with her invaluable supports and constructive attitude. Studying under her supervision and working within her team has been a golden opportunity for me.

I also send my deepest regards and emotions to my family who have been and are the most precious asset of my life and have been the biggest support for me throughout my life.

I would also like to thank all of my colleagues in the VidPro research group for their kind and friendly help and cooperation and for making the research environment pleasant for me. I also send my thanks to my dear friends who are a great source of energy and hope to me and have been my second family during my studies and research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The purpose of 3D human pose estimation is to enable computers to estimate (reconstruct) 3D human poses while being fed with 2D imagery [3]. Reconstruction of the 3D human poses from 2D imagery is a prominent field of research in computer vision [1, 2, 4, 3]. Applications include human-computer interaction, virtual and augmented reality, and robotics. In spite of the vast study of the subject in the literature, it is still assumed a challenging problem. This stems from the inherent loss of the depth information in the 2D images and from image related factors such as blur, noise, occlusion, etc.

3D pose estimation from 2D images is less challenging while using depth images captured by the depth sensors. Depth sensors render another channel pertaining to the distance of the object from the camera aperture; that is, in the rendered images, there exists some information representing the depth of the objects. On the other hand, 2D images do not contain any explicit depth information. Such images provide information on the intensity of the channels (grayscale, RGB, etc.) illustrating the 2D scenery.

(a) 2D images

(b) 2D poses

Figure 1: Two different input types of 3D human pose estimation.

The Human Visual System (HVS) can perceive depth from 2D images combining either stereo or monocular clues [5]. Despite the absence of explicit depth information in 2D images, the HVS is not fully impaired of estimating the 3D geometry while having a single view of an object. This perception originates from the ability of HVS to take advantage of the visual memory of object shapes [6]. Considering Figure 1 (a) as an example, the HVS can have an understanding of the 3D human pose by just looking at a single view 2D image.

1.2 Problem Statement

The problem of 3D human pose estimation consists of two different formulations in terms of input, which can either be a 2D image or a 2D pose assumed to be available, e.g., extracted using 2D pose estimation techniques. Figure 1 illustrates these two types.

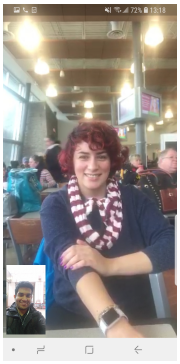
3D human pose estimation has been previously studied while assuming the full presence of the body parts in the input image. Full body presence means the full shaped human body is present in the image. Here, self-occlusion is excluded, since in these cases, the body is not partially present and the full shape of the body is present to the camera. Partial body presence means the body is partially present in the input image. This can be due to zoom-in photography, imperfect object (bounding box) segmentation, occlusion, or when the whole skeleton of the subject is not aimed to be reconstructed (see Figure 2). Figure 3 (a) shows a case of full body presence, while Figure 3 (b) and (c) illustrate two cases of partial body presence. An example of partial body presence where the whole subject body is not aimed to be reconstructed is when somebody is communicating through video call (see Figure 2 (c)) where the person is often filming oneself in the upper body parts (but not necessarily all of them).

Assuming the need for 3D human pose estimation (e.g., to be used in augmented reality), 3D pose estimators should be able to reconstruct the partial body pose from this partial presence of the body parts. State-of-the-art assume the full presence of the human body and thus do not have effective performance under these cases. Figure 3 (d-f) show the performance of the well-known VNect method [1] applied to the images in Figure 3 (a-c); while going from Figure 3 (a) to (c), the performance is considerably affected. Partial 3D pose estimation under partial body presence can also be viewed as a human pose estimation adaptive to the parts of the body present in the input image and does not make any special assumption about it.

An ability of the HVS is perception of the full body pose when encountering partial body images. An example is when half of the body's rigid limbs are absent from the input. In this case, the humans' visual perception's prior about the rigidity helps reconstruct the full body pose even though some of the body parts are absent.

(a) Zoom-in photography [7]

(b) Imperfect Segmentation



(c) Video Calling

(d) Occlusion [7]

(e) Zoom-in photography [7]

Figure 2: Examples of partial body presence.

For example, by looking at Figure 3 (b), although the ankles are not present in the image, the HVS can perceive the full body pose due to its priors about the rigidity of the legs. State-of-the-art in 3D pose estimation do not estimate 3D full pose under partial body presence.

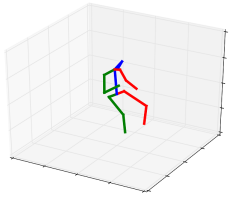
1.3 Research Objectives

This thesis is concerned with performing 3D pose estimation on a single person from a single view and using a single 2D monocular image, without assuming the existence of 2D poses nor the existence of the full human body in the input image. We assume that the subject is not *intensely absent* from the image such that significant pose information to recover the body pose is lost, for example, when only the legs or only the shoulder area is present. Figure 4 illustrates examples of not sufficient

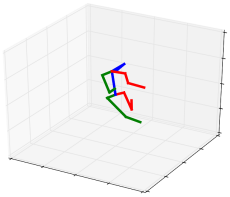
(a) Full body presence

(b) Partial body presence

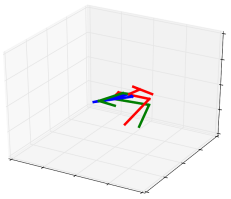
(c) Partial body presence



(d) VNect output for (a)



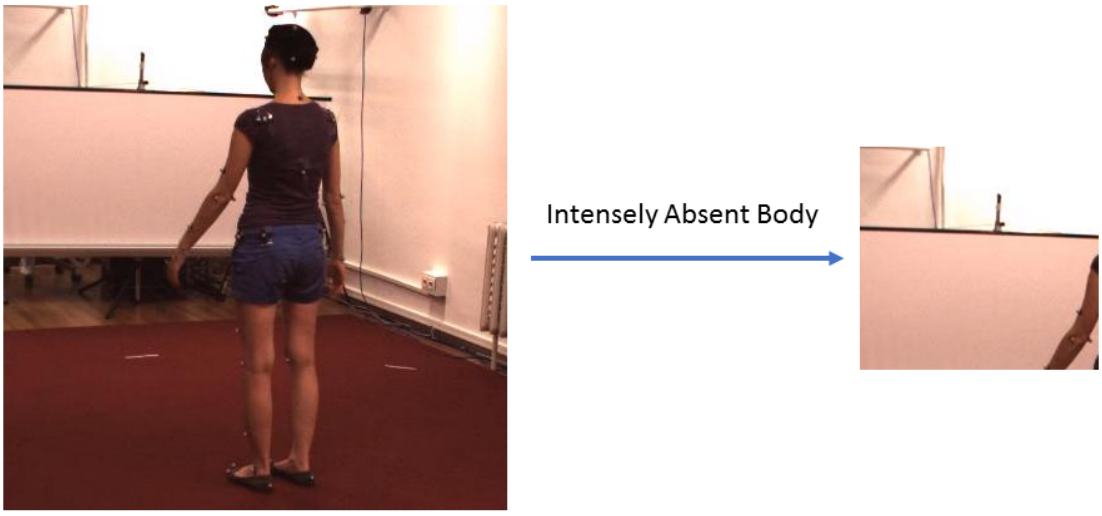
(e) VNect output for (b)



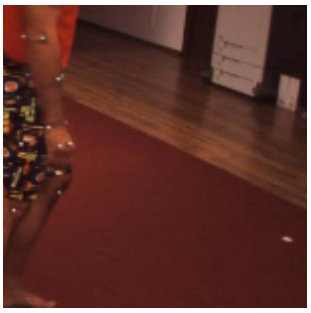
(f) VNect output for (d)

Figure 3: Input images with different body parts presence and the output of VNect [1].

pose information, while Figure 3 illustrates examples of sufficient partial presence of the pose information in the input. We design and train a deep neural network (regression network) to regress the full body pose regardless of what joints are present in the input image. We also design a detection stage consisting of detection networks corresponding to each of the main body joints (17 in this work) to classify the presence or absence of the body joints in the fed image. The output of this stage is meant to determine what joints are present in the input image. Thus, our approach to estimate 3D pose under partial body presence consists of two stages: regression and detection, and hence the output of our approach is twofold: the regression network outputs the full pose reconstructed from the input image, and the regression and detection networks output the body pose for the joints present in the input image.



(a)



(b)

Figure 4: Examples of intensely absent pose information.

1.4 Summary of Contributions

The main contribution of this thesis is handling the 3D human pose estimation under partial presence of the body parts. To this end, this work first introduces a method to employ deep neural networks to regress the full human body pose based on input images partially containing the human body. Two different approaches are proposed for this task, namely, using an intermediate 2D pose estimation stage and direct regression of the 3D human pose. Secondly, this work proposes a deep detection stage to classify the presence of each of the joints and provide the body presence

vector. Finally, the detection networks are integrated with the regression network to enable partial estimation of the human pose based on the joints present. Both joint and separate training of the regression and detection networks is proposed and experimented in this work.

1.5 Thesis Outlines

The rest of the thesis is organized as follows. In Chapter 2, we present a review of the background material related to our work. In the first part, we review the basic concepts and mathematical formulation of 3D human pose estimation. In the second part, a review of neural networks is presented as well as deep learning and convolutional neural networks. We also discuss different functionalities of neural networks, i.e., the regression and classification tasks.

In Chapter 3, we present a review of 3D human pose estimation literature. In this chapter, the evolution of 3D human pose estimation is discussed as well as the most related methods to our work.

In Chapter 4, we present the regression network while using a 2D pose estimator as an intermediate task for regressing the 3D pose. This design consists of a 2D pose estimation network followed by a network to extract 3D human pose from the 2D poses. The input to the regression network is a 2D image.

In Chapter 5, we propose a direct 3D human pose estimation architecture using a deep convolutional neural network which is fed with 2D intensity images. This Chapter also presents the architectures used to perform the joints' detection task. The detection stage is integrated with poses extracted from the regression networks to form the partial output poses. [1]

[1] A paper based on this chapter has been published in IEEE International Conference on Image Processing (ICIP) 2018, Athens, Greece [8].

In Chapter 6, we present and discuss the experimental setup and the performance evaluation of our method and compare it to the state-of-the-art.

Finally, Chapter 7 concludes this thesis by summarizing the main points and contributions, and giving directions for further work.

Chapter 2

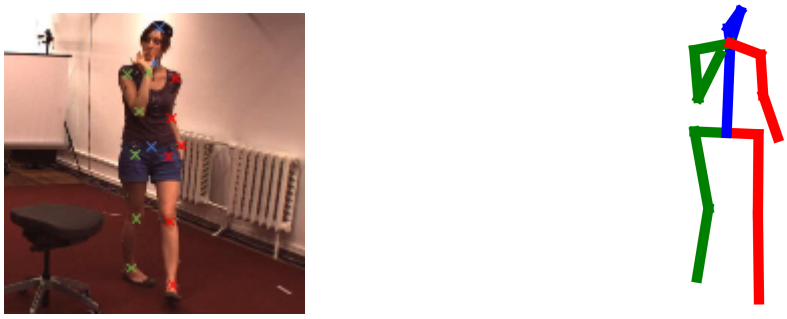
Background Material

2.1 Introduction

In this chapter, we discuss the background material required for our work. In section 2.2, we discuss the problem of 3D human pose estimation and its general formulations. In section 2.3, the basic concepts of neural networks will be presented with focus on deep learning concepts.

2.2 Formulating 3D human Pose Estimation

3D human pose estimation is estimating the position of the body's main joints in the 3D space [3] from 2D images. The body joints are the connections between the bones in a human body. By body's main joints, we mean the ones which can give us an overall view of the body skeleton (or pose) by connecting the corresponding joints. Figure 5 illustrates presenting the pose using the main body joints. Figure 5 (a) is an example of a human image with highlighted joints locations; cross marks on the figure show the locations of the main body joints. Figure 5 (b) illustrates the corresponding 3D body pose based on the marked joints. Formulations of 3D human



(a) Highlighted joints' locations (b) 3D body pose based on the marked joints

Figure 5: Human body's main joints example.

pose estimation differ depending on the input, namely, 2D poses or 2D images. We discuss these two formulations in sections 2.2.1 and 2.2.2, respectively.

2.2.1 3D Human Pose Estimation Using 2D Pose

The 2D human body pose can be defined as the projection of the 3D pose to the image plane [9]. The 2D pose practically illustrates the spatial position of the body joints on the 2D image. Assuming J the number of main body joints, the 2D human pose matrix $\mathbf{W} \in \mathbb{R}^{2\times J}$ is a 2-dimensional matrix in which each column represents the 2D Cartesian coordinates (x and y) for one of the J main body joints [6, 10, 11].

By defining $\mathbf{S} \in \mathbb{R}^{3\times J}$ as the 3D human pose matrix in which each column represents the 3D Cartesian coordinates (x, y, and z) for one of the j main body joints, the 3D human pose estimation problem can be formulated as a transformation $\mathcal{T}$ which maps the 2D joints' coordinates to their corresponding 3D coordinates in the 3D space. Therefore, the estimation of 3D human body pose based on 2D poses can be basically formulated as

$$\mathbf{S} = \mathcal{T}\{\mathbf{W}\}. \tag{1}$$

Different approaches can be employed to obtain the transformation $\mathcal{T}$. One of the widely-used approaches is *minimization of the reprojection error*,e.g., in [6]. In this approach, different projection models may be adopted. Assuming a **linear projection** between the 2D and 3D poses and $\mathbf{\Pi} \in \mathbb{R}^{2\times 3}$ as the *camera calibration matrix* projecting the 3D pose to the 2D space, the relationship between the 2D and 3D poses can be stated as

$$\mathbf{W} = \mathbf{\Pi}\mathbf{S}. \tag{2}$$

The *reprojection error* is defined as the difference between the actual 2D poses and the 2D poses extracted by projecting the 3D poses to the 2D space

$$e = d(\mathbf{W}, \mathbf{\Pi}\mathbf{S}), \tag{3}$$

where d is a *distance measure* (e.g., mean squared error) and e is the reprojection error. Therefore, the 3D human pose estimation can be formulated as the following optimization problem

$$\min_{\mathbf{\Pi},\mathbf{S}} d(\mathbf{W}, \mathbf{\Pi}\mathbf{S}), \tag{4}$$

which jointly searches for the best camera calibration matrix $\mathbf{\Pi}$ and 3D pose based upon minimization of the reprojection error in (3). The cost function (reprojection error) e in (3) is usually simplified to limit the searching domain by making assumptions on the projection type (such as using the weak-perspective camera model in [6]).

Another effective approach is modeling the 3D human pose as a linear combination

of some basis shapes

$$\mathbf{S} = \sum_{i=1}^{K} c_i \mathbf{B}_i, \tag{5}$$

where $\mathbf{B}_i$ is one of the K basis shapes and c_i is the corresponding weight. This approach stems from the *active shape model* [12]. To model the complex variations more effectively, some methods use *sparse representation* [2, 6, 13], where an over-complete dictionary is adopted and $\mathbf{S}$ is expressed as a sparse combination of the basis shapes from the basis shapes' dictionary. The cardinality term (or similar terms) is added to the cost function which represents the sum of the weights. This term regularizes the growth of the weights to address the sparseness of the pose model.

Another widely-used approach for extracting 3D pose estimation from 2D poses is using discriminative learning approaches. In these methods, a learning-based mapping between the 2D and 3D poses are learned (e.g., using neural networks) and then it can be used to make estimations about new queries. In this case, the discriminative mapping accounts for the transformation $\mathcal{T}$ in (1).

2.2.2 3D Human Pose Estimation Using 2D Imagery

A digital image $\mathbf{I}(x, y)$ is represented by its gray levels of intensity ($\mathbf{I}$) or its color components. The image signal is a function of its spatial coordinates in the 2D space (x and y) where the domain depends on the image size. A color image is represented by three components, the most important of which are RGB (red, green, and blue) and HSV (hue, saturation, and value) representations. The gray levels and the color components are usually presented by a number in the $(0,2^b)$ range, where b is the number of bits used to store each of the image pixels (typically $b = 8$).

Estimation of the 3D human body pose using 2D images can be formulated [14]

as

$$\mathbf{S} = \mathcal{T}\{\mathbf{I}\}, \tag{6}$$

where $\mathcal{T}$ is the transformation from the 2D imagery to 3D human poses and $\mathbf{S} \in \mathbb{R}^{3\times J}$ is the estimated human body pose in the 3D space. There exist different approaches for modeling the transformation $\mathcal{T}$. One approach is to first extract the 2D human body pose matrix $\mathbf{W} \in \mathbb{R}^{2\times J}$ using some 2D human pose estimation technique (e.g., discriminative methods) and then, the problem follows the discussions in section 2.2.1 (see [11]). Another approach is to view this problem as a discriminative mapping between the 2D images and the 3D poses. In this case, poses are directly regressed from 2D input images [15, 16].

2.3 Artificial Neural Networks

2.3.1 Neurons

Artificial neurons are the basic building blocks of artificial neural networks [17]. They are inspired by *biological neurons* in humans' and animals' brains which learn structures by being exposed to different sorts of examples and experiences. Figure 6 illustrates a multiple-input artificial neuron. The input-output relationship for this neuron can be written [17] as

$$y = f(n) = f(\sum_{i=1}^{p} w_{1,i}x_i + b) = f(\mathbf{W}\mathbf{x} + b), \tag{7}$$

where y is the output of the neuron, f is the activation function, b is the bias parameter, $\mathbf{W}$ is the weights' vector, and $\mathbf{x}$ is the input vector while assuming the neuron to have p inputs. As can be seen in Figure 6, a weighted sum of the inputs (where $W_{1,i}$

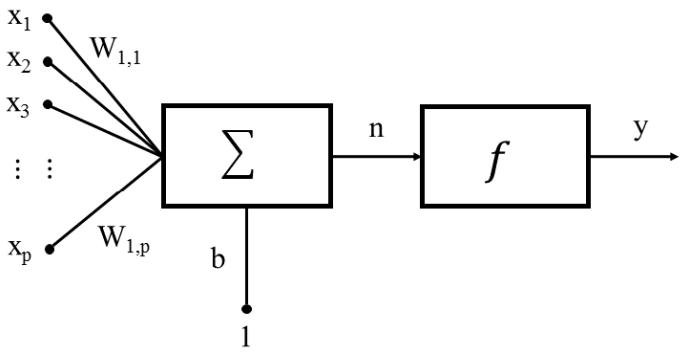
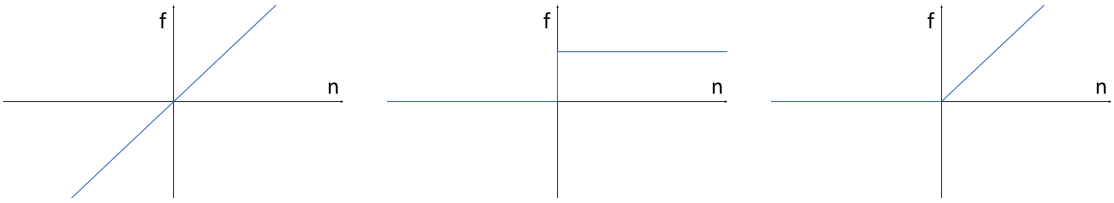


Figure 6: Multiple-input artificial neuron

is the weight for i^{th} input x_i) is added to a bias term b. In the figure, the value "1" is the identity element so that the bias term is simply added to the weighted sum of the inputs. Then, an activation function f takes effect on this summation which results in the output y. $\mathbf{W}$ and b are adjustable parameters which give learning capacity to the network and the activation function (also called transfer function) is a specific mathematical linear or non-linear function chosen based on the specifications of the problem. Some of the most common activation functions are the identity (linear), binary step, and rectified linear unit functions; these functions are plotted in Figure 7 (a)-(c), respectively.



(a) Identity (linear) (b) Binary step (c) Rectified linear unit (ReLU)

Figure 7: Plots of three common activation functions $f(n)$.

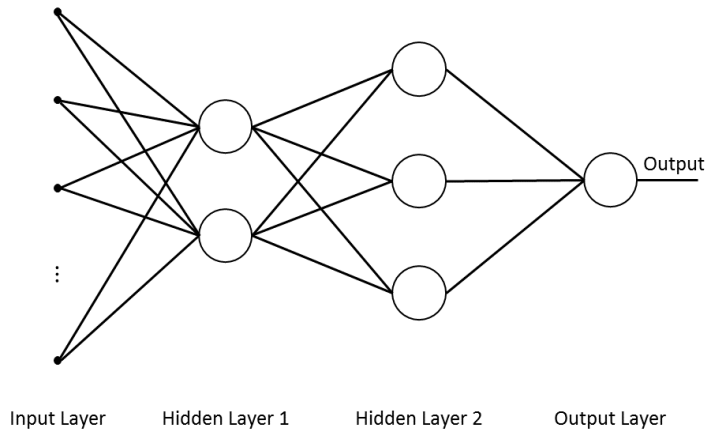


Figure 8: An example of a multilayer fully-connected feedforward neural network

2.3.2 Multilayer Neural Networks

Neurons can be grouped together to form networks representing more complex models [18]. Figure 8 illustrates a *fully-connected feedforward neural network*. Each circle in this figure represents a single multiple-input artificial neuron (node) as in Figure 6. This neural network is called a *feedforward neural network* since each of its neurons is only connected to neurons from the previous layers (connections between neurons do not make a cycle). It is also *fully-connected* as each of the neurons is connected to all of the neurons from the previous layer. The architecture of the network (number of layers and nodes and the way they are connected) is determined by the application and the learning capacity we need. Adding more neurons and layers to the neural network increases the trainable parameters of the network which may help the learning capacity provided that we have enough data to train the network.

2.3.3 Neural Networks' Learning

Learning in the neural networks can be defined as utilizing a set of pre-verified data to update the model so that the network's output gets improved, i.e., approaches

an optimal result. More intuitively, a neural network is learning as the adjustable parameters of the network (weights and biases) are updated resulting in less error in the output of the network [17].

A neural network is usually discriminatively trained using the *backpropagation algorithm* [19]. The first step in the backpropagation algorithm is determining the cost (loss) function of the network. The cost function depends on the network architecture, activation function, and the type of error chosen for the training stage (based on the task). *Mean squared error* and *cross-entropy* are two examples of the most common error functions. After determining the cost function, the neurons' parameters are initialized which can be done using different techniques, e.g., generating random values for the parameters. The parameters are then iteratively updated using gradient methods in which the weights are updated so that the cost function approaches its local minimum [17]. An *iteration* is defined as performing backpropagation on a single data point and backpropagating the error once on the whole training data is called an *epoch*. The idea in gradient methods is to modify the parameters in the opposite direction of the gradient of the output. This approach helps the network update itself so that the value of the cost function is decreased after each iteration. A convex cost function C approaches the global minimum using gradient methods. Using the gradient (steepest) descent method, each of the model weights can be updated using

$$w_{i,j}^{new} = w_{i,j}^{old} + \Delta w_{i,j}^{old} = w_{i,j}^{old} - \alpha \frac{\partial C}{\partial w_{i,j}^{old}}, \tag{8}$$

where $w_{i,j}$ is the weight parameter for the i^{th} layer and the j^{th} node, α is the learning rate, and C is the cost function. Decreasing the gradient of the cost function from the weights helps update the weights so that the loss is decreased in each iteration.

The learning rate α is a hyper-parameter which determines the steps by which the

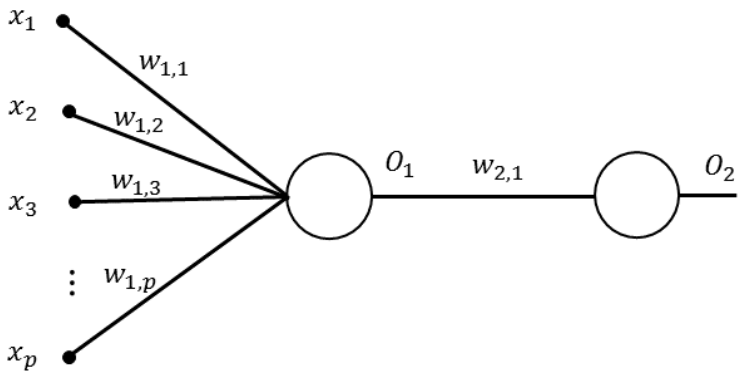


Figure 9: An example of a two-layer feedforward neural network.

network parameters are updated. A large learning rate can make the learning process unstable since the large steps in the cost function can result in overshooting the minimum, and thus even not converging to the optimal solution. Small learning rates mean slower learning speed, and thus the convergence time may be increased. Another solution for determination of the learning rate is to start with large learning rates and then decreasing it as we approach the minimum. Therefore, different functions (which can depend on the number of iterations) can be used for the learning rate, e.g., dividing the learning rate by a value after each iteration (or a group of iterations).

The weights for the hidden layers are updated using the *chain rule* from calculus. To explain the chain rule in training the neural networks, we assume a network with 2 layers (1 hidden layer and 1 output layer) as shown in Figure 9. Assuming each of the layers having a single neuron, C the cost function, O_2 the output of the output layer, O_1 the output of the hidden layer, and $w_{1,i}$ the weight for the i^{th} input, the partial derivative of the cost function with respect to this weight is

$$\frac{\partial C}{\partial w_{1,i}} = \frac{\partial C}{\partial O_2} \times \frac{\partial O_2}{\partial O_1} \times \frac{\partial O_1}{\partial w_{1,i}}. \tag{9}$$

Thus, this weight can be updated in each iteration using

$$w_{1,i}^{new} = w_{1,i}^{old} - \alpha \frac{\partial C}{\partial O_2} \times \frac{\partial O_2}{\partial O_1} \times \frac{\partial O_1}{\partial w_{1,i}}. \tag{10}$$

ADAM [20] is one of the widely used optimization methods for backpropagating the gradient errors in neural networks, due to its time and memory efficiency. ADAM method selects separate learning rates for different network parameters based on an estimation of the first and second moments of the gradients.

2.3.4 Deep Learning

Deep learning methods use models with multiple processing layers each of which correspond to different levels of abstraction [21]. More intuitively, the idea in deep neural networks is to use features extracted by each of the layers (which can be viewed as a level of abstracting the information) as the input to another neural layer so that we can have higher levels of abstraction. These features differ based on the application.

In practice, deep neural networks have more than a single layer [22]. These networks are trained to find a discriminative representation of the data. The types of layers in deep neural networks can differ based on the application. Some of the most famous layer types are feedforward, convolutional, and recurrent neural layers. Using these layers, different network types are designed based on the application, where convolutional neural networks (CNNs) are widely used in computer vision.

2.3.5 Convolutional Neural Networks

Convolutional neural networks (CNNs) are designed for processing the data coming in the form of multiple arrays, and thus are appropriate choices for dealing with image

data [21]. CNNs take advantage of a number of layer types making them a great fit for many computer vision applications. We introduce the main CNN layer types, namely, Convolutional and Pooling layers, briefly in the following.

2.3.5.1 Layer Types

Convolutional Layers: a convolutional layer consists of a set of trainable parameters (weights) which are called *filter banks*. The filter banks do not change spatially and are kept the same for all inputs of each layer (pixels in 2D images, features in higher layers). Convolutional layers apply a convolution operation between the input of the layer and the filter bank. The convolution operation helps extract features from the input benefiting from local conjunctions of the features in any neighborhood of the pixels. The feature extraction process directly applied on an image (first layer) can be considered as a sort of edge detection. In the rest of the layers, the extracted features correspond to higher levels of abstraction. Sharing the weights (using the same filter bank for all the inputs of each layer) helps making the extracted features invariant to position.

The output of the convolution operation is then passed to an activation function. A widely-used example of the activation functions is the *Rectified Linear Units* (ReLUs) expressed as

$$f(x) = max(0, x), \tag{11}$$

where x is the input to the function and $f(x)$ is the output of the ReLU function (see Figure 7 (c)).

When the input to the first layer is a 2D image, the output features are edge-related as they are weighted summation of input pixels. Features in next layers are extracted by the convolutional layer itself and are not necessarily known to human

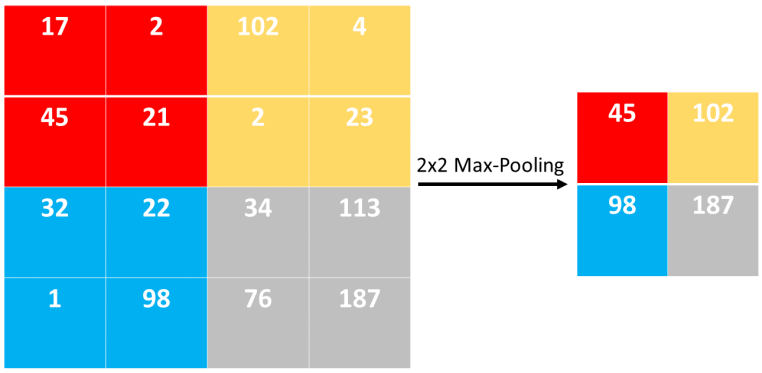


Figure 10: An example of 2×2 max-pooling process.

perception.

Pooling Layers: a pooling layer merges semantically relevant features into one. Max-pooling layers are one of the most popular pooling layers which calculate the maximum of the pixels' intensities in a local patch of a feature map (e.g., input image or output of any of the convolutional layers). For example, considering a 2×2 max-pooling layer, for any 2×2 patch of the input, the maximum value is kept to represent the whole patch. Therefore, the input will be downsized by a factor of 2 in each direction. Figure 10 illustrates the process of 2×2 max-pooling layer.

2.3.5.2 CNN Design

CNNs are usually formed by cascading a few number of convolutional (usually followed by a nonlinear activation function such as ReLU) and pooling layers. They are then followed by fully-connected neural networks (also called *dense* layers). This network can be then trained by backpropagating the cost function gradients such that all the weights are trained. Figure 11 illustrates an example of a CNN architecture with one convolutional layer, one pooling layer, and two fully-connected layers. Different CNN architectures can be designed using the explained layers depending on the application.

There are a number of useful methods which are used to improve the performance

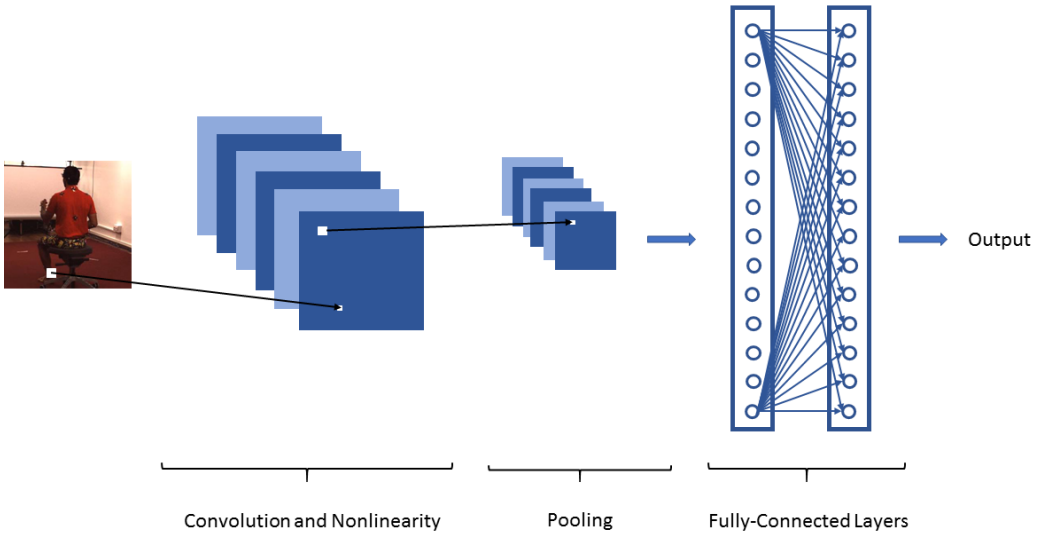


Figure 11: An example of a CNN

of CNNs. One of these methods is batch normalization [23]. In training CNNs, the data is usually fed to the network in groups of data points which are called *batches*. The batch size represents the number of data points (images in this work) which are fed jointly to the network and it is a hyper-parameter chosen based on the available computational capacity. In batch normalization, any batch is normalized before being fed to the fully-connected neural layer to prevent change of the input distribution in the middle of the training process.

Another useful method to enhance the performance of the CNNs is random *dropout* of the feature detectors (network parameters) [24]. For example, in a 0.5 dropout, half of the parameters are randomly removed in each training case. This method - especially when dealing with small-size datasets - can reduce the possibility of the *overfitting* problem by a great amount. Overfitting is a major problem in the learning process while dealing with small datasets. In overfitting, the network - especially when the network capacity is high - is so vulnerable to learning the data too closely that

the learning process can be considered as a sort of memorization. Thus, overfitting can prevent the network from performing the generalization task effectively.

2.3.5.3 CNN Tasks

The task, for which a CNN is designed, is a dominant factor in the architecture design of the network. The main two classes of tasks in CNNs are *classification* and *regression* tasks.

Classification: In this set of problems, the network's output is a label which is chosen among a limited set of possible outcomes (classes). Image classification is an example of this task, where a limited number of image classes exists; therefore, any image is assumed to belong to one of these classes. Detection problems are another example of the classification task in which the output can take only two values, namely, "detected" and "not detected". Classification problems with 2 possible labels are called *binary classification* problems.

Regression: Regression is the analysis of a quantity of interest using the information from one or more other quantities [25]. In regression problems, the network is designed to predict a real quantity (a continuous value). An example of the regression task is *depth estimation* [26, 27] where the input is an image and the objective is to estimate the depth values for each of the pixels. This depth is a continuous value, i.e., it is not chosen among a limited set of possible outcomes and can take any value in the valid range (e.g., positive values in depth estimation). *Precipitation prediction* [28] is another example of the regression task; these problems are categorized as a regression task since the output (precipitation amount) is a continuous quantity.

Chapter 3

Literature Review

3.1 Introduction

Techniques to human pose estimation [3] have various forms ranging from pose estimation for a specific part of the human body to the full body pose estimation (e.g.[1, 4, 2]). Pose estimation of specific human body parts includes hand pose estimation [29, 30, 31, 32], head pose estimation [33, 34, 35, 36], and upper-body pose estimation [37].

Human pose estimation techniques can be categorized into 2D and 3D pose estimation in terms of the dimensionality of the output. 2D pose estimation aims to find the location of the main body joints on the image plane, i.e., determining the pixel with the most probability of being the location of some specific body joint. 2D human pose estimation has been well studied (e.g., [4, 10, 38, 39, 40, 41, 42]). 3D pose estimation is an active research field [1, 2, 6, 11, 13] but remains challenging due to the ambiguity caused by the loss of depth information in the usual intensity images but also due to image related issues such as blur, noise, occlusion, etc. Some methods use depth images containing explicit depth information for 3D pose estimation [43, 44, 45, 46].

The remaining of this chapter is organized as follows. In section 3.2, we discuss the problem of structure from motion; section 3.3 summarizes literature of 3D human pose estimation and section 3.4 highlights one of the widely-used approaches, namely, the discriminative deep learning approach.

3.2 Structure from Motion Methods

Historically, the idea of extracting 3D structures from 2D imagery has been vastly studied under structure-from-motion (SfM) [47, 48]. SfM is often focused on mapping 2D key points or features to the 3D space. SfM usually follows a 2D key point detection or feature extraction stage. Some of the well-known feature extraction methods are *scale-invariant feature transform* (SIFT) [49, 50] and *speeded-up robust features* (SURF) [51].

Tomasi and Kanade in [47] develop a factorization method based on the decomposition of the input data (2D keypoints/feature) into two different matrices; one of these matrices correspond to the motion parameters and the other one to the 3D structure. In [47], the object is assumed to be rigid. Bregler et al. [48] generalize this method to deformable shapes assuming the deformable shapes as a linear combination of some basis rigid shapes.

Another class of relevant topics to human pose estimation is objects' pose estimation and tracking [52] which is usually formulated as matching some key points or features in a target image with those in new queries (input images). Zhou et al. [6] present a 3D car model estimation using 2D annotations as well as 3D human pose estimation.

3.3 3D Human Pose Estimation Methods

3D human pose estimation outputs an estimate of joints' 3D positions (or the angles) using 2D information, which is usually given in terms of either 2D images or joints' 2D locations (2D pose). 3D pose estimation methods based on 2D joints' location assume that the 2D joints have been given or already extracted using some 2D pose estimation technique. They use various techniques such as sparse representation[6, 11], factorization [13], and neural networks[53] to estimate the 3D poses from 2D poses. Zhou et al. in [6, 11] propose to use a convex approach while using sparse representation for the 3D human pose estimation from 2D landmarks. The method in [11] presents a 2D joints' uncertainty map predictor to handle the cases when 2D joints' information is not available. Wandt et al. [13] try to factorize 2D poses in camera parameters, base 3D human poses, and mixing coefficients. They also show that making periodic assumptions on the mixing coefficients can improve the performance in 3D human pose estimation. Tekin et al. [54] have introduced some method to fuse two different streams, one acting on 2D joints information and the other on the images to extract the 3D pose information. Martinez et al. [53] discuss different natures of the error between the ground truth and estimated poses while having a 3D pose estimator with a 2D pose estimator as an intermediate stage; that is, they compare the results while extracting the 3D pose directly from the ground truth 2D poses and while regressing the 3D pose from 2D poses extracted by some off-the-shelf 2D pose estimation techniques.

3D pose estimation methods directly from 2D images can be categorized to those using a single view for the estimations [55] and those leveraging multi-view imagery[37, 56, 57]. Single-view methods are used for cases in which there is only a single camera capturing image or video from the scene, and thus its view is the only present information to estimate the 3D pose. In these methods, even when using the datasets

providing the data from different viewpoints, frames from each viewpoint are considered as separate data points and each frame is treated as an individual single-view image. On the other hand, in multi-view cases, there are a few cameras filming from different viewpoints. The cases consisting of multi-view are easier to extract the depth information since, in the single-view, there is more loss of the depth information in the captured imagery.

3D pose estimation techniques can also be categorized into those estimating the pose assuming a single person present in the window (e.g., [58]) and those doing the estimation for multiple people [57].

Another important classification of direct 3D pose estimation methods is those performing the estimation on a single image(e.g.[15, 55]) as opposed to the ones performing on a sequence of images (video) [11, 14]. Zhou et al. [11] use an expectation-Maximization algorithm on the whole image sequence. They add a temporal smoothness prior to the penalty (cost) function to take advantage of the existing information over time. Tekin et al. [14] compensate for the motion to keep the subject centered and then regress the 3D pose in the central frame directly from a spatiotemporal volume of bounding boxes.

Another classification of human pose estimation methods is generative and discriminative approaches. Generative approaches use some a priori information to perform the estimation and thus include a modeling stage in advance to extracting the 3D poses[59]. An example of these priors is the size of each of the body parts and their topology as used in [59]. Discriminative approaches are the data-driven model-free methods which learn a mapping between the input images (also features or 2D poses) and the desired output(e.g.[15, 55, 60, 61]). The method in [60] leverages a large database of 3D human motion capture combined with a human model from 3D computer graphics to generate training pairs of 3D human pose with their realistic

2D silhouettes. Ning et al. in [61] use the bag of (visual) words approach to discriminatively estimate 3D pose from a 2D image. Deep learning approaches are another category of discriminative approaches which have been widely used in 3D human pose estimation.

3.4 Discriminative Deep Learning Approaches

Deep learning is a general concept for learning data representation using deep multilayer networks [21] which has drawn significant attention since its early introduction. Numerous papers have utilized deep learning concepts to address the problem of 3D human pose estimation[1, 2, 11, 15, 16, 55, 58]. Among these methods, some perform an initial 2D pose estimation stage and then use that information to estimate the 3D pose[11]. Mehta et al. [1] use 2D pose estimation to locate the subject and thus do not need tightly cropped windows. Some deep learning methods [15, 16] perform direct estimation of the 3D poses without any need to regress the 2D pose in advance. Brau and Jiang [58] perform direct regression of the 3D poses; however, they add a 2D projection layer to enforce pose constraints on the estimated output. Pavlakos et al. [55] handle the estimation by discretization of the 3D space and regressing per-voxel likelihood for the joints in the discrete 3D space.

3.5 Most Related Work

To the best knowledge of the author, no publication exists that handles 3D human pose estimation from 2D images under partial body presence. In this section, we discuss methods most related to ours. The method [62] estimates the pose partially; however, it is based on depth sensors and not monocular 2D imagery. The method in [16] has a detection network as a pre-training stage; however, it still does not take

into account partial body presence. It also performs joint training of the detection and regression networks. The detection stage in this work is not the same as ours in the sense that [16] detects the presence of joints concerning whether or not they encounter self-occlusion (and not partial body presence as defined in this work).

The methods VNect by Mehta et al. [1] and InWild by Zhou et al. [2] do not take into account partial body presence; however, they are close to our work as they perform 3D human pose estimation directly from 2D images using deep CNNs. VNect method [1] uses a CNN architecture combined with kinematic skeleton fitting to regress 2D and 3D joints' locations, jointly; see Figure 12. InWild method [2] cascades a 2D regression network with a 3D depth regression network to extract the 3D human pose from 2D images; as shown in Figure 13.

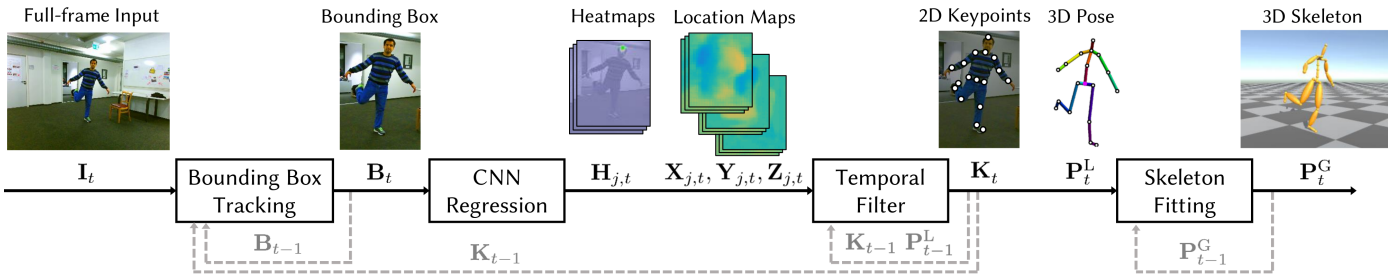


Figure 12: The block diagram of VNect method [1].

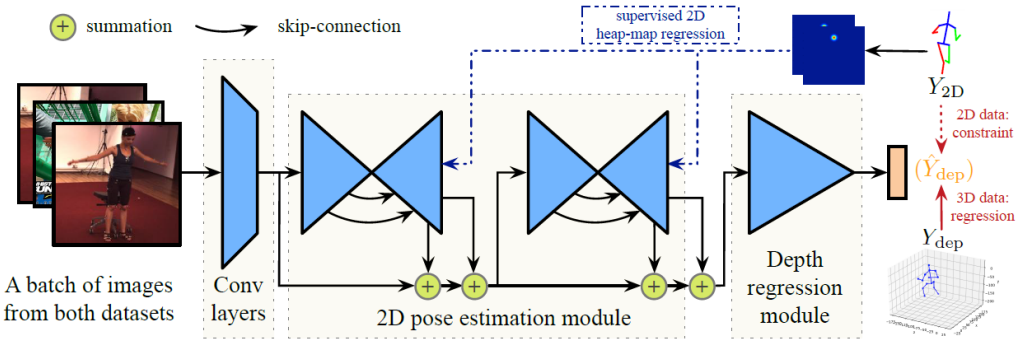


Figure 13: The block diagram of InWild method [2].

Chapter 4

3D Pose Regression Based on 2D Pose Regression from Partial Body Images

4.1 Introduction

In this chapter, we present our first approach to regress the 3D human pose from a 2D image using 2D pose regression as an intermediate stage under partial body (Part2D3D). In this approach, the image is primarily fed to a CNN to regress the 2D joints' locations directly from the input images. The following step is to map the 2D poses to the corresponding 3D poses.

The rest of this chapter is organized as follows: in section 4.2, we present the architecture used in our network to perform the regression; then, section 4.3 presents the training details of our method.

4.2 Proposed Network Architecture

The network architecture in our approach contains two separate networks used for the two different stages of the 3D pose regression, namely, extracting the 2D poses and mapping the 2D poses with the corresponding 3D poses. These networks will be presented in sections 4.2.1 and 4.2.2, respectively

4.2.1 2D Pose Estimation

We perform 2D pose estimation by direct discriminative regression of 2D poses from 2D input images using CNNs. The used CNN is illustrated in Figure 14, where the input is a single 2D intensity image and the output is 2D human pose, which is a matrix showing the position of the body's main joints using 2D Cartesian coordinates. The network consists of five convolutional layers with kernel sizes of 9×9, 9×9, 5×5, 3×3, and 3×3. The depths of these layers are 128, 256, 256, 128, and 64, respectively. It also contains three max-pooling layers each of which downsize the input features by the factor of 2×2. Each of these layers is followed by a batch normalization and a rectified linear unit. Then, we have two layers of fully-connected feedforward (dense) layers with 4096 neurons, each followed by a 0.5 dropout and a linear activation function.

The output of this network is normalized by dividing the 2D coordinates by the width and height of the input image size (both equal to 150 in this work since the input to our network is a 150×150 image). This normalization is done on the ground truth 2D poses since we want this network to address 2D pose estimation for partial cases. In other words, in this step, we assume to have a (hypothetical) full body image, and we try to extract its (full) 2D pose from the partial body input. Therefore, although this output has the same nature as 2D human pose, it is not exactly meant to determine the joints' locations on the input image. This output gives a normalized

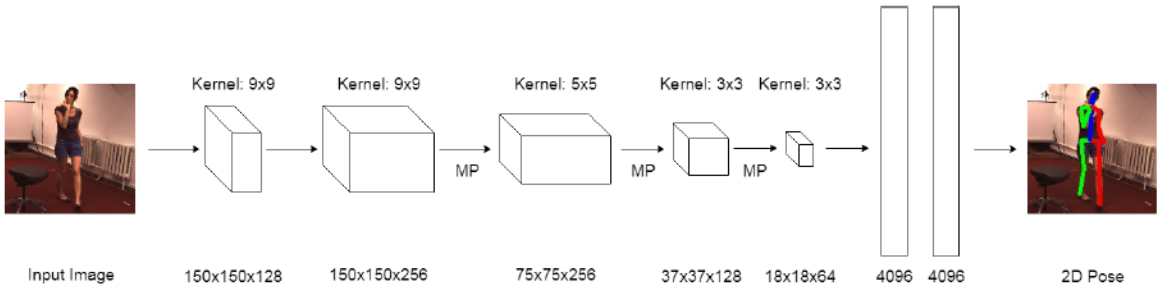


Figure 14: The proposed 2D pose regression network. All of the convolutional layers are followed by a Rectified Linear Unit (ReLU). MP indicates a 2x2 max-pooling layer. The dense (fully-connected feedforward) layers at the end of the network are followed by a linear activation function.

full 2D pose (in which each of the coordinates is a number between 0 and 1), generated to make a basis for extraction of 3D human pose information.

4.2.2 2D Pose to 3D Pose Mapping

The mapping from 2D poses to 3D poses is performed using a fully-connected feed-forward neural network. The input to this network is the normalized 2D pose and the output is the 3D human pose. This network is meant to learn different body poses by encountering different scenarios. The architecture consists of two fully-connected (dense) layers of size 2048. Figure 15 illustrates the proposed 2D pose to 3D pose mapping stage. Each of the layers are followed by a ReLU activation function. We used a 0.5 dropout on each layer to prevent overfitting.

4.3 Training Details

The loss function used for training the networks is the *mean squared error*

$$MSE(y^{gt}, y^{est}) = \frac{1}{DJ} \sum_{k_1=1}^{J} \sum_{k_2=1}^{D} (y^{gt}_{k_1,k_2} - y^{est}_{k_1,k_2})^2, \tag{12}$$

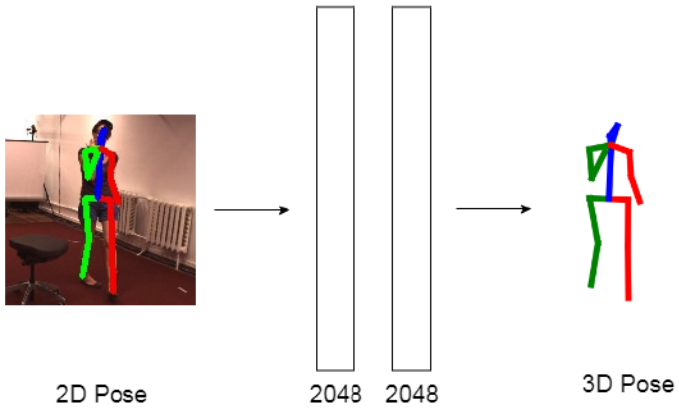


Figure 15: Proposed 2D pose to 3D pose mapping network; each of the dense layers are followed by a ReLU and a 0.5 dropout.

where y^{gt} and y^{est} are $J \times D$ matrices containing, respectively, the ground truths and estimated values of the body joints' locations. In these matrices, each row represents the Cartesian coordinates of one of the body's main joints. J is the total number of body's main joints and D is the dimensionality of the coordinates which equals 2 for the 2D extraction network and equals 3 for the 2D to 3D mapping network. The networks are trained using ADAM optimization [20] with a learning rate of 0.0001

We feed the network with full and partial body images. To make sure that the network is effectively trained, so to maintain the human body structure, we provide the network with the full poses (2D poses as the output of our 2D pose estimator and 3D poses from mapping 2D to 3D poses). Therefore, all the data that is presented to the network respects the human body structure, i.e., they are the full pose of the human body whether or not it is fully present in the input image. In other words, the network is trained using both cases of partial and full body presence as the input while having the full body pose fed as the ground truth for both of the cases.

Experimental setup and simulation results of our approach Part2D3D, i.e., 3D pose estimation using 2D pose estimation are presented and discussed in Chapter 6.

Chapter 5

Direct 3D Pose Estimation from Partial Body Images

5.1 Introduction

In this chapter, we present our method for direct estimation of 3D human poses under partial body presence (Part3D). It consists of a regression CNN network to regress the 3D human pose directly from 2D input images under partial body presence and a detection CNN network to detect the joints present in the input image. Figure 16 illustrates an overview of our proposed method having two different outputs. The first output is the 3D full body pose extracted by the regression network, where the input image can contain the body parts either fully or partially. The second output is the 3D partial body pose generated by integrating the regression and detection networks. As the detection networks are meant to determine the joints present in the input image, this output is the partial pose resulting from the 3D location of the joints present in the input image.

The rest of this chapter is organized as follows: in section 5.2, we present the architectures used in our regression and detection networks; then, section 5.3 presents

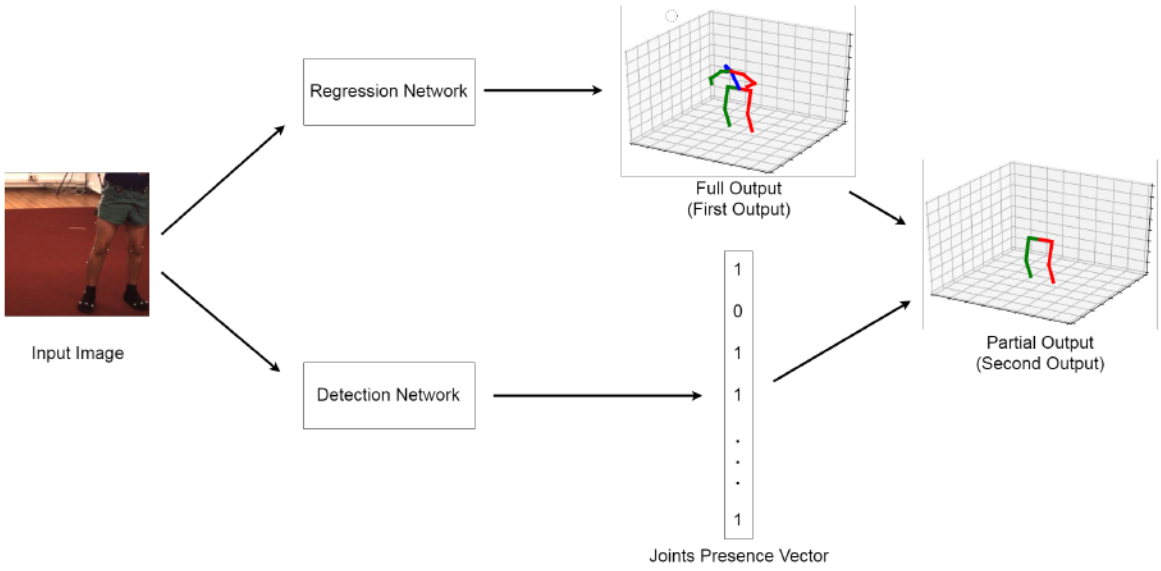


Figure 16: An overview of the proposed method.

the training details of our method.

5.2 Network Architecture

5.2.1 Regression Network's Architecture

The proposed regression network regresses the body's joints' locations in the 3D space for all the joints - regardless of whether or not it is fed with an image containing the full subject body. Figure 17 illustrates the CNNs used for the regression stage. As seen, the network's input is a 3-channel image. The input image is first resized to 150×150. The input is then fed to the convolutional layers. Our regression network consists of five convolutional layers and two fully-connected feedforward layers, the details of which are as follows.

Layer 1: this layer consists of a convolutional layer of size 128 with a 9×9 kernel followed by a Rectified Linear Unit as the activation function.

Layer 2: this layer consists of a convolutional layer of size 256 with a 9×9 kernel. The

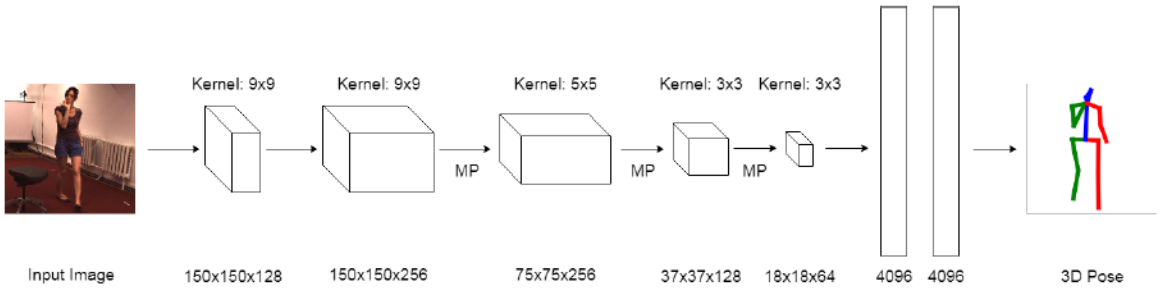


Figure 17: The 3D pose regression network; All of the convolutional layers are followed by a Rectified Linear Unit (ReLU); the dense layers at the end of the network are followed by a linear activation function. MP indicates a 2x2 max-pooling layer.

activation function is the Rectified Linear Unit and is followed by a 2×2 max-pooling layer which resizes the output features from 150×150 to 75×75.

Layer 3: this layer consists of a convolutional layer of size 256 with a 5×5 kernel and a Rectified Linear Unit activation function, followed by a 2×2 max-pooling layer which resizes the output features from 75×75 to 37×37.

Layer 4: this layer is a convolutional layer of size 128 with a 3×3 kernel, and a Rectified Linear Unit activation function, followed by a 2×2 max-pooling layer which resizes the output features from 37×37 to 18×18.

Layer 5: this layer is a convolutional layer of size 64 with a 3×3 kernel, followed by Rectified Linear Unit as the activation function.

Layer 6: this is a fully-connected feedforward layer of size 4096 that is fed with features extracted by the 5^{th} layer after being flattened, i.e., the output of the previous layer is concatenated and converted to a vector. The activation function of this layer is a linear activation function.

Layer 7: this is a fully-connected feedforward layer of size 4096; its output is a vector of size 51 which presents the predicted 3D locations of human body's main 17 joints. It is followed by a linear activation function.

All of the convolutional layers are followed by a batch normalization layer to increase the stability of the learning process. Each of the feedforward layers is followed

by a 0.5 dropout regularization to prevent overfitting.

5.2.2 Detection Networks' Architecture

The proposed detection stage classifies the presence or absence of each of the human body's main joints. Its output is a binary vector of size 17 indicating presence or absence of the joints in the input image. The detection stage includes 17 detection networks, each of which classifies the presence or absence of one of 17 main body joints. Figure 18 illustrates the network architecture for each of the 17 detection networks. As seen, a detection network consists of 2 convolutional layers and one fully-connected feedforward layer as follows, where the input is a 2D input image.

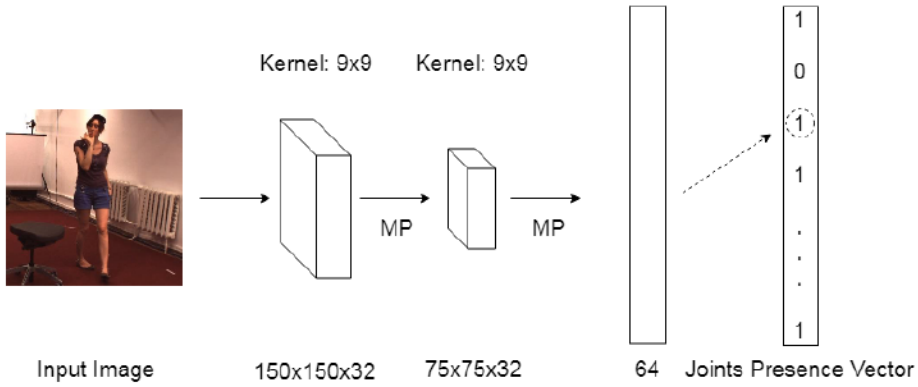


Figure 18: The proposed detection network. The convolutional layers are followed by a Rectified Linear Unit (ReLU). MP indicates a 2x2 max-pooling layer. The feedforward layer is followed by a softmax activation function.

Layer 1: this is a convolutional layer of size 32 with a 9×9 kernel, followed by a Rectified Linear Unit as the activation function. A 2×2 max-pooling layer follows which resizes the output features from 150×150 to 75×75.

Layer 2: this is a convolutional layer of size 32 with a 9×9 kernel, followed by a Rectified Linear Unit activation function. It is followed by a 2×2 max-pooling layer which resizes the output features from 75×75 to 37×37.

Layer 3: this is a fully-connected feedforward layer of size 64 that is fed with features

extracted by the 2^{nd} layer after being flattened. The activation function is a softmax activation function. The softmax function is defined as

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}} i = 0, 1, ..., k, \tag{13}$$

where x_i is the input of the softmax function, and f is the output. k is the number of classes which is equal to 2 in this case, where $k = 0$ represents the "absent" class and $k = 1$ represents the "present" class. Therefore, this layer has a binary output classifying the presence or absence of the corresponding body joint in the input image.

Both of the convolutional layers are followed by a batch normalization layer and the feedforward layer is followed by a 0.5 dropout regularization to prevent overfitting; this network is not much prone to overfitting considering its size, however.

5.3 Training Details

We used the *mean squared error* as in (12) as the loss function of our regression network. The cost function we used for the detection networks is the *cross-entropy*

$$CE_j = -[\theta_j log(p_j) + (1 - \theta_j) log(1 - p_j)], \tag{14}$$

where CE_j is the cross-entropy for the j^{th} data point (joint), θ_j is 1 if the joint is present in the input and 0 if not, and p_j is the probability by which the network has estimated the joint to be present. The optimization technique for the purpose of training the networks is ADAM optimization [20].

We have tested two different approaches to train the regression and detection networks. The first approach is joint training, in which the networks are combined and trained simultaneously. The second approach is to train the networks separately. These two approaches are explained in the following.

5.3.1 Joint Training

In the joint training approach, the regression and detection networks are assumed to be combined together, and thus only a single cost function is defined for the whole network. The cost function is a weighted sum of the individual cost functions. Therefore, the overall cost function of the joint training network C^t is defined as

$$C^t = \alpha C^{reg} + \beta \sum_{j=1}^{J} C_j^{det} = \alpha MSE + \beta \sum_{j=1}^{J} CE_j \tag{15}$$

where C^{reg} is the cost function of the regression network (mean squared error) as in (12), C_j^{det} is the cost function of the detection network (cross-entropy) corresponding to the j^{th} body joint as in (14), and J is the number of main body joints. In this equation, α and β are the cost function weights of the regression and the detection networks in the overall cost function, respectively. α and β are hyper-parameters determining the impact of their corresponding cost functions on the overall cost function. We set $\alpha = 1$ and $\beta = 0.1$, where all the 17 detection networks share the same weight.

We have used the same feature extraction process for the regression and detection networks in this approach; that is, we have kept the convolutional layers for all of the networks and the networks differ only in their fully-connected layers. We used the convolutional layers from the regression network since they are deeper and have more computational capacity. Figure 19 shows the joint regression and detection network.

We used a learning rate of 0.0001 (which is set experimentally by trying different values and monitoring the convergence of the network) and a batch size of 32. The weights are initialized randomly, and the data is shuffled before every epoch. By shuffling the data, we mean reorganizing the data points randomly so that they are fed to the network in a different order for every epoch.

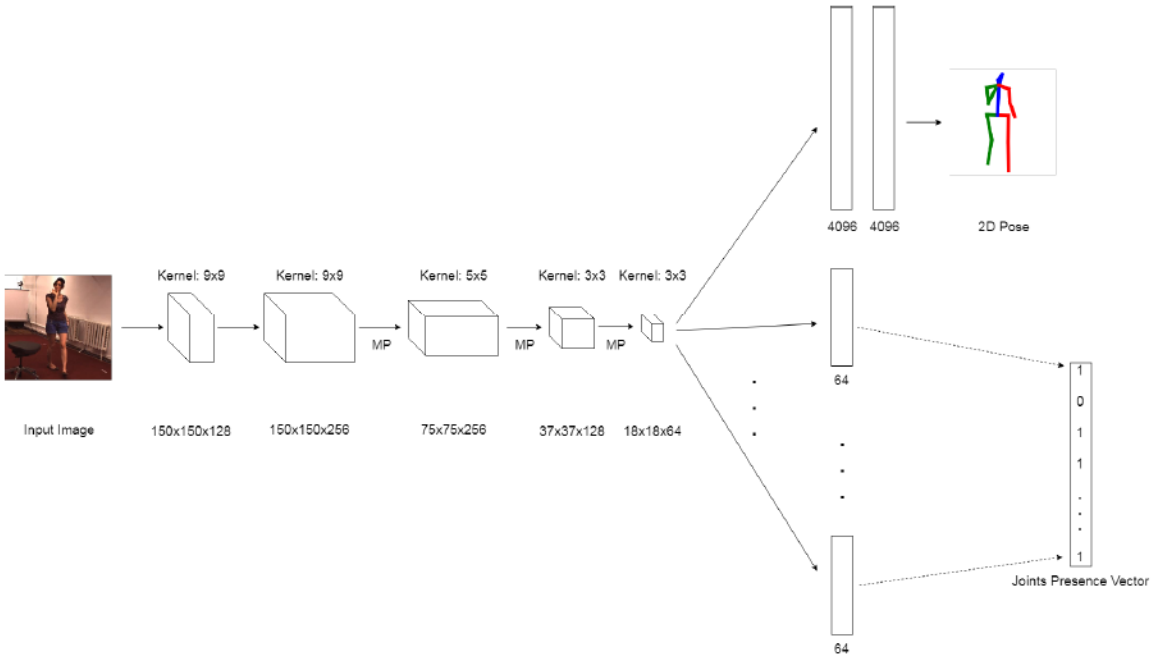


Figure 19: Joint regression and detection network.

5.3.2 Separate Training

In this approach, the detection and regression networks are treated as two completely separate networks. Therefore, for each of them, the network is initialized, trained, and fine-tuned separately as an individual network. We used a learning rate of 0.0001 for both the regression and the detection network. The batch size for the regression network's training is 32 while the batch size for the detection networks' training is 256. Having greater batch sizes usually helps the network converge more rapidly. However, because of the limitation of the computational infrastructure we could not use greater batch size for the regression network (noting that the regression network has much more trainable parameters than the detection network). The weights are initialized randomly, and the data is shuffled before each epoch.

The regression network is trained by feeding the full pose as the ground truth output of the network while giving both full and partial body presence cases as the input. This approach is adopted to enable the network to learn the whole human

body structure and enable the network to make full body estimations (although not accurate for all the body joints) under both the full and partial body presence cases.

The experimental setup and the simulation results of our method Part3D for direct estimation of the 3D human pose from a partial body input image are presented and discussed in Chapter 6.

Chapter 6

Experimental Setup and Results

6.1 Introduction

In this Chapter, we provide and discuss the experimental setup and the results of our methods presented in Chapter 4 and 5. In section 6.2, we present the experimental setup for our work; in section 6.3 we discuss the evaluation protocol; then, section 6.4 presents and discusses the simulation results of our methods.

6.2 Experimental Setup

The network architectures have been selected experimentally by evaluating the performance of different architectures. We have started with a network having a single convolutional layer as well as a single dense layer. Then, we added more layers (with different depths and kernel sizes) and evaluated the performance of the new network. Each layer was added only if it improves the performance. This process is stopped when a layer either does not improve the network performance or makes the network excessively large such that the training or predicting process gets very slow.

Since we have employed three different approaches for the regression network,

the network architecture design has been performed only for the first network (we started with direct pose estimation with separate training). We started with one convolutional layer and went up until 7 convolutional layers. For other two, we used the same convolutional layers and only tested the network with one more and one less layer to evaluate the effectiveness of our architecture. For the detection network, we tried 1 to 5 convolutional layers. The experimentations showed that having more than 2 layers does not improve the performance considerably; it reduces the efficiency, however, as we have 17 detection networks.

We use the Human3.6M dataset [63] to assess the performance of our methods. The Human3.6M dataset is a widely-used 3D human pose estimation dataset and includes 3.6 million frames in video sequences of frame size 1000×1000 on seven subjects, and each subject performs 15 different actions (e.g., Directions, Discussion, and Eating) while being filmed from four different viewpoints. We have used five subjects (i.e., S1, S5, S6, S7, and S8) for training and two subjects (i.e., S9 and S11) for testing. We have assessed our methods on all of the scenarios of the selected subjects.

To provide the networks with partial body images, firstly, we segment the subject in each of the original images using the bounding boxes provided in the dataset; that is, we draw a square region fully covering the subject body. Then we store two different images each of size 150×150: 1) the segmented image which accounts for the full body case and 2) the corresponding partial body version which is generated using a *random window selection* approach. Therefore, we have equal size of full and partial body presence cases as the input. In other words, for any full body presence case, there exists a case of partial body presence, both of which share the same subject and 3D full pose. We have used 150×150 input images since using this dimensionality, we did not need to upsample many of the images, and thus we could use input images

(a) (b) (c)

Figure 20: Example of our random window selection method: (a) an original image from the dataset; (b) image with perfectly segmented subject; (c) a randomly selected window.

with the original quality.

To generate partial body images (windows), we use a *random window selection*, where the top-left and bottom-right corners of the window are selected randomly with a uniform distribution. This is to make sure we have different cases of partial body presence cases generated among our new set of data. To ensure that sufficient joints exist in the input image, i.e., the image is not intensely cropped that it is not possible to extract pose information from the image, the randomly cropped window size must be larger than a quarter of the original image. Figure 20 illustrates an example of this random window selection: Figure 20 (a) is an original image from the dataset, Figure 20 (b) is the square bounding box drawn around the subject, and Figure 20 (c) is an example of a randomly selected window.

For training, using our above-mentioned random selection of windows, we extracted two windows from each of the video frames of the training set in [63] and one window from each of the video frames of the validation (test) set. In total, we have used 623,900 extracted windows for training and 548,819 for testing. The ground truth 3D poses and 2D poses are provided in the dataset. We have downsampled the video frames by a factor of five, before the random window selection, to avoid

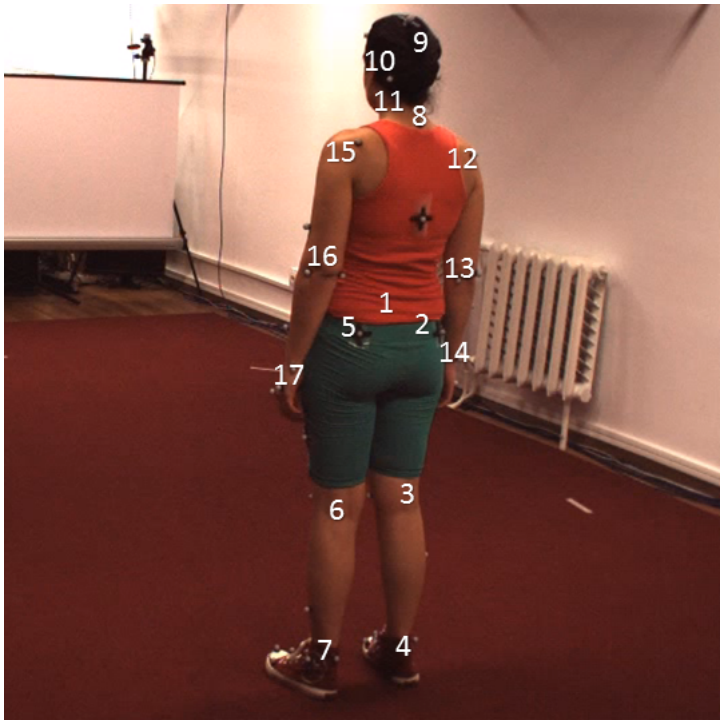


Figure 21: The main body joints used in this work.

redundancy in the training and each video frame is treated as an independent data point. The processing time of every epoch on the whole training data using a Titan Xp GPU was approximately 6 hours for the regression network and 30 minutes for the detection networks and the processing time for every new test image in validation stage was approximately 1∼2 seconds for the regression network and less than 1 second for the detection networks.

In this thesis, we assume 17 joints as the main body joints to form the human body pose. Figure 21 shows and indexes the main body joints used in our method and Table 1 lists these joints. One can use the same architecture with different number of the joints, however, since the features extracted for different number of joints do not seem to vary, and thus changing the number of the body joints should not much affect the performance.

To prepare the output for the detection network, we need to generate the joints'

existence vector which is a vector of size 17. Each of the elements of this vector is a binary value which is 1 if the corresponding joint is present in the image and 0 if not. The dataset does not provide this data. To extract these binary ground truth numbers, we use the data from 2D pose ground truth which are provided in the dataset. The 2D pose ground truth determines the exact location of each of the joints in the image plane. After each random window selection, all the joints' coordinates are compared to the selected window to see if the joint falls into the chosen window. Using this comparison, the 17 binary values corresponding to the main 17 body joints are determined and used as the ground truth to the detection stage. Figure 22 illustrates the process of joints' existence vector generation.

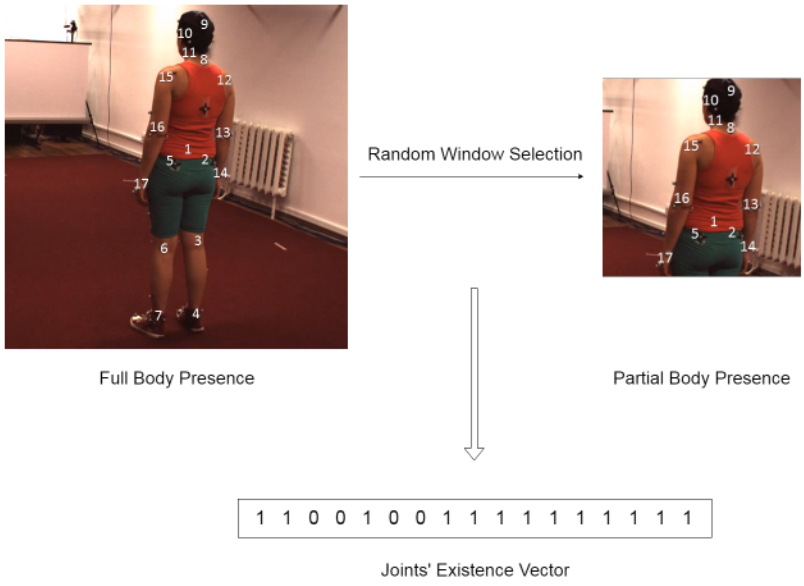


Figure 22: The process of joints' existence vector generation.

6.3 Evaluation Protocol

The performance assessment takes place on the last two subjects of the Human3.6M dataset (S9 and S11) which we use for validation. To evaluate the performance of the

regression stage, we use the *mean-per-joint-error* (expressed in millimeter)

$$L(y_{gt}, y_{est}) = \frac{1}{J} \sum_{j=1}^{J} ||C_j^{y_{gt}} - C_j^{y_{est}}||_2, \tag{16}$$

where y_{gt} is the ground truth joints' position matrix, y_{est} is the estimated joints' position matrix, $|| \cdot ||_2$ is the Euclidean norm, $C_j^{y_{gt}}$ is the vector of the j^{th} column of the matrix y_{gt}, $C_j^{y_{est}}$ is the vector of the j^{th} column of the matrix y_{est}, and J is the number of the joints (either $J = 17$ for full body pose or $J = J'$, where J' is the number of joints present). The mean-per-joint-error $L(\cdot)$ in (16) is calculated after aligning the root joint (a joint assumed as the reference; pelvis in this work); that is, the locations of the root joints of the ground truth and the estimated poses are aligned. We have calculated $L(y_{gt}, y_{est})$ both on the full body skeleton ($J = 17$) and the joints which are present inside the window ($J = J'$). These two measurements are used to assess the performance of our methods for the two different outputs; that is, the full body pose estimation and the partial body pose estimation (the joints present in the input image). We present these errors both on the individual scenarios and on the whole dataset.

The detection stage is evaluated separately for each of the body joints. As the evaluation measure, we use the *binary accuracy* which is the percentage of the detections in which the presence or absence of the corresponding joint has been successfully detected; this metric can be formulated as:

$$ACC = \frac{Number\ of\ true\ detections}{Number\ of\ detections}. \tag{17}$$

6.4 Simulation Results

We compare our methods to two of the state-of-the-art, namely, "VNect" method [1] and the "InWild" method [2]. The authors provide their trained networks and open-source code for experimentation. Since we need to assess the methods with the data prepared for the partial body presence cases, we need open-source methods for comparison.

6.4.1 Objective Evaluation

To asses the results from the regression task, we present the outputs for our methods as well as their comparison with VNect [1] and InWild [2] methods. The results of our methods are presented for the 3 different approaches used in this work; namely, 3D pose regression using 2D pose estimation (Part2D3D), direct 3D pose estimation with joint training of the regression and detection networks (Part3D), and direct 3D human estimation with separate training (Part3Ds). The outputs to be discussed are the "full pose estimation from full body presence", "full pose estimation from partial body presence", and "partial pose estimation from partial body presence". The results with these three outputs are summarized in Table 2 and detailed based on the 15 scenarios of the Human3.6M dataset in Tables 3, 4, and 5, respectively.

Table 2 summarizes the performance of our and related methods. It shows promising performance of our method compared to related work; our method well recovers part of the joints not present in the input. It is also noteworthy that our three approaches have very close overall performance. This can be interpreted by considering the fact that all three networks have the same architecture in terms of the convolutional layers. This causes the networks to have similar feature extraction stages and also their learning capacity to be very close to each other. The difference between them stems from the difference in the nature of their specific training process, where

they take advantage of 2D pose information, 3D pose information, or 3D pose information as well as joints' presence data. Considering their difference in Table 2, we can see that, among the three approaches adopted in this work, 3D pose estimation with separate training of regression and detection networks has the best performance. This can be interpreted by considering the fact that the feature extraction stage of this approach is only dedicated to the regression task (and not shared with the detection task as in joint training). Therefore, among these experimentations, the direct regression being separately trained is proposed as the main approach of this work. The approach employing the 2D intermediate stage has the worst performance among our approaches. This can be explained by noting the fact that while having a 2D intermediate stage, there are two different sources of error (i.e., 2D pose estimation error and mapping error) being added, and causing a more overall error in the 3D pose estimation.

Table 4 and 5 show the state-of-the-art are ineffective under partial body presence in the input images (for all 15 scenarios). This is since they are not designed and trained for these sets of queries. In other words, when these networks are fed with an input image partially containing the human body, they search for the full body parts, and thus they fail to use the information present in the image.

As can be seen in Table 3, the state-of-the-art methods have a better performance on the 3D human pose estimation while having full body presence cases in the input images. This is due to different nature of these methods with ours which aims to estimate the 3D human pose based on various cases of body presence and is robust to absence of body joints (which is unknown in general) in the input images.

Figure 23 illustrates the comparison of the learning rates of the separate and joint training approaches for the direct estimation of the 3D human pose. This figure shows the test (validation) error after different epochs. As can be seen, the separate training

Figure 23: Estimation of learning rates for separate and joint training.

approach has better performance as well as more rapid convergence.

Table 6 presents the data for the performance of the detection stage for the 17 main human body joints as well as the average. This performance is measured using binary accuracy as discussed in (17) and expressed in percentage. The results are presented for both joint training and separate training of the detection stage. As can be seen in the table, the network being trained jointly with the regression network has a performance similar to separately trained networks. The joint training provides the network with more training information, i.e., it shares the detection information with the regression network and vice versa. However, we see that it does not necessarily improve the performance of the networks. This can be due less learning capacity which is caused by sharing the convolutional layers between regression and detection networks. In other words, in this approach, there is a single feature extraction stage for all the networks, and thus there exist trainable parameters compared to separate training of the networks.

6.4.2 Subjective Evaluation

We present the subjective evaluation of our method (Part3D)[1] and related work in Figure 24. These results show the complete failure of the state-of-the-art under partial body presence.

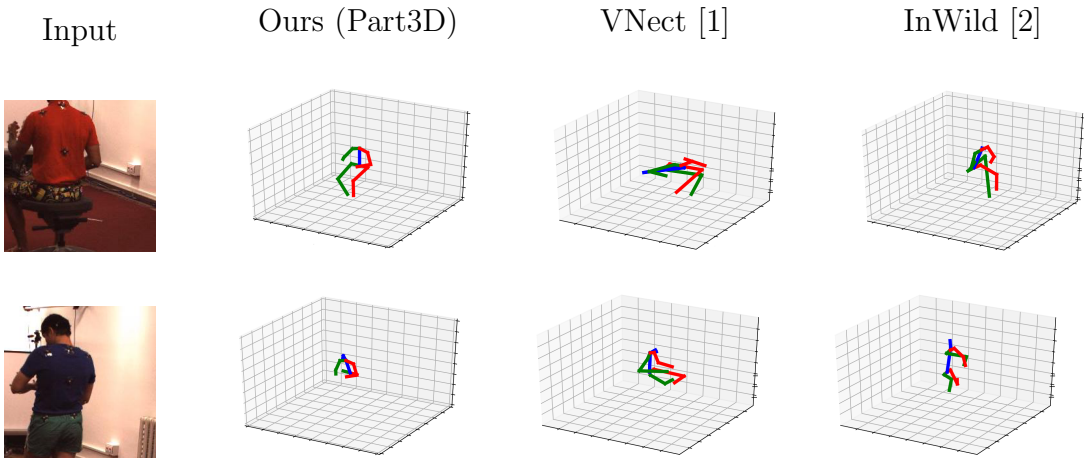


Figure 24: Subjective results under partial body presence.

Figure 25 presents some examples of our method's performance under full body presence for subjects 9 and 11 of the Human3.6M dataset. Figure 26 and Figure 27 present the subjective evaluation of our method under partial body presence for subject 9 and 11 of the Human3.6M dataset, respectively. These figures show the subjective effectiveness of our method for partial body presence cases.

6.4.3 Discussion

As can be seen in Table 3, 4 and 5, the performance of our methods is different for the different actions (scenarios). Figure 28 illustrates the average of full pose estimation from full body presence and partial pose estimation from partial body presence for different scenarios using direct pose estimation with separate training (Part3Ds). As

[1]A demo of our work (direct estimation with separate training) is under https://users.encs.concordia.ca/∼amer/Part3DPose/Partial3DHumanPose_mp4.avi

can be seen, the "smoking" scenario has the highest error, which shows generalizing from the training set to the validation set is more challenging for this scenario.

A shortcoming of our method compared to related work is its performance under full body presence. However, even though Table 3 shows that on average each joint is being estimated 16 centimeters further from its true value, our subjective inspection yield that our method is effective in following the human body structure; this means it does not make meaningless estimations in its failure cases under full body presence.

A challenging case for our method is when there are not sufficient body parts present in the input image, and thus not enough information exists to help reconstruct the 3D human pose; an example is in Figure 29; although we have followed our constraint in the random window selection (having windows larger than a quarter of the original window), there is not enough body parts in the input image to regress the human pose and estimate its orientation effectively.

Joints	**Name**
1	Pelvis
2	Right Hip
3	Right Knee
4	Right Ankle
5	Left Hip
6	Left Knee
7	Left Ankle
8	Neck Back
9	Head Back
10	Head Front
11	Neck Front
12	Right Shoulder
13	Right Elbow
14	Right Wrist
15	Left Shoulder
16	Left Elbow
1v	Left Wrist

Table 1: List of human body's main joints; the joints' numbering follows the indexes in Figure 21.

Method	Full Estimation from Full Body Input	Full Estimation from Partial Body Input	Partial Estimation from Partial Body Input
Ours-with 2D (Part2D3D)	167.09	210.17	188.83
Ours-direct/joint (Part3D)	166.41	<u>200.33</u>	<u>180.32</u>
Ours-direct/separate (Part3Ds)	160.69	**191.81**	**177.82**
VNect	<u>80.5</u>	396.44	338.01
InWild	**64.90**	400.50	332.48

Table 2: Overview of joints' regression stage on Human3.6M dataset measured on the whole dataset for all tested methods (mean-per-joint-error in mm). Lowest error is bold and second lowest underlined.

Scenario	InWild [2]	VNect [1]	Ours with 2D (Part2D3D)	Ours Direct/Joint (Part3D)	Ours Direct/Separate (Part3Ds)
Directions	**54.82**	62.6	170.11	163.54	154.14
Discussion	**60.70**	78.1	174.32	154.37	155.45
Eating	**58.22**	63.4	145.43	151.22	142.27
Greeting	**71.41**	72.5	163.87	144.74	142.55
Phone Call	**62.03**	88.3	149.93	166.18	153.55
Posing	65.53	**63.1**	196.89	179.07	174.76
Purchases	**53.83**	74.8	152.19	137.76	132.49
Sitting	**55.58**	106.6	201.40	170.74	178.95
Sitting Down	**75.20**	138.7	176.01	203.82	192.76
Smoking	111.59	**78.8**	225.80	264.46	254.62
Taking Photo	**64.15**	93.8	147.40	159.60	150.31
Waiting	**66.05**	73.9	159.64	144.25	146.12
Walking	**51.43**	55.8	187.88	180.06	176.92
Walking Dog	**63.22**	82.0	135.00	131.13	127.35
Walking Together	**55.33**	59.6	153.02	149.42	139.82
Total	**64.90**	80.5	167.09	166.41	160.69

Table 3: Average mean-per-joint-error of **full body pose** estimation based on **full body images** from Human3.6M dataset in mm. Lowest error is bold and second lowest underlined.

Scenario	InWild [2]	VNect [1]	Ours with 2D (Part2D3D)	Ours Direct/Joint (Part3D)	Ours Direct/Separate (Part3Ds)
Directions	491.17	370.07	215.01	<u>187.13</u>	**178.92**
Discussion	473.96	389.39	228.94	<u>195.49</u>	**190.57**
Eating	480.61	402.73	<u>175.38</u>	178.84	**168.69**
Greeting	497.63	387.13	213.98	<u>187.26</u>	**178.11**
Phone Call	463.24	396.00	<u>179.17</u>	185.60	**173.53**
Posing	506.48	425.14	251.03	<u>223.71</u>	**216.72**
Purchases	488.37	376.01	202.73	<u>173.50</u>	**163.21**
Sitting	494.55	418.73	239.15	<u>215.24</u>	**210.59**
Sitting Down	470.45	436.89	**210.31**	234.94	<u>222.52</u>
Smoking	456.14	460.92	**274.28**	297.52	<u>296.61</u>
Taking Photo	457.41	385.29	<u>184.35</u>	188.93	**176.24**
Waiting	471.57	372.12	204.63	<u>177.27</u>	**172.82**
Walking	483.46	409.19	244.83	<u>229.11</u>	**225.63**
Walking Dog	456.84	344.36	178.20	<u>168.40</u>	**157.21**
Walking Together	486.74	370.92	192.36	<u>183.44</u>	**170.54**
Total	400.50	396.44	210.17	<u>200.33</u>	**191.81**

Table 4: Average mean-per-joint-error of **full body pose** estimation based on **partial input images** from Human3.6M dataset in mm. Lowest error is bold and second lowest underlined.

Scenario	InWild [2]	VNect [1]	Ours with 2D (Part2D3D)	Ours Direct/Joint (Part3D)	Ours Direct/Separate (Part3Ds)
Directions	428.02	285.16	187.59	<u>163.53</u>	**161.17**
Discussion	428.94	330.79	210.28	**178.58**	<u>180.69</u>
Eating	433.95	348.60	**151.95**	158.90	<u>153.80</u>
Greeting	433.13	303.43	180.85	<u>156.42</u>	**154.04**
Phone Call	418.55	342.34	**163.84**	170.35	<u>164.97</u>
Posing	451.79	339.57	212.32	<u>184.57</u>	**183.27**
Purchases	423.92	292.78	171.83	<u>144.67</u>	**141.08**
Sitting	457.83	371.41	227.27	**203.38**	<u>205.00</u>
Sitting Down	428.93	392.29	**183.01**	207.33	<u>201.72</u>
Smoking	435.88	428.91	**250.58**	<u>272.21</u>	276.12
Taking Photo	411.69	338.94	<u>172.66</u>	177.21	**171.16**
Waiting	418.09	305.38	182.46	**156.80**	<u>157.35</u>
Walking	443.44	361.72	228.43	**215.32**	<u>216.72</u>
Walking Dog	399.03	290.22	166.14	<u>160.05</u>	**154.15**
Walking Together	430.66	304.13	175.54	<u>168.76</u>	**160.89**
Total	332.48	338.01	188.83	<u>180.32</u>	**177.82**

Table 5: Average mean-per-joint-error of **partial body pose** estimation based on **partial input images** from Human3.6M dataset in mm. Lowest error is bold and second lowest underlined.

Joint	Ours Joint Training (Part3D)	Ours Separate Training (Part3Ds)
Pelvis	92.50	92.55
Right Hip	90.46	90.25
Right Knee	86.81	84.67
Right Ankle	86.47	85.65
Left Hip	90.87	91.04
Left Knee	86.61	83.72
Left Ankle	85.68	85.64
Neck Back	93.17	93.13
Head Back	90.82	89.64
Head Front	94.83	94.88
Neck Front	93.11	91.33
Right Shoulder	90.49	90.13
Right Elbow	83.23	90.69
Right Wrist	77.75	80.51
Left Shoulder	91.64	89.25
Left Elbow	85.23	84.10
Left Wrist	80.30	70.10
Average	**88.23**	**88.00**

Table 6: Results of joints' detection stage on Human3.6M dataset measured using binary accuracy (percentage) as in (17).

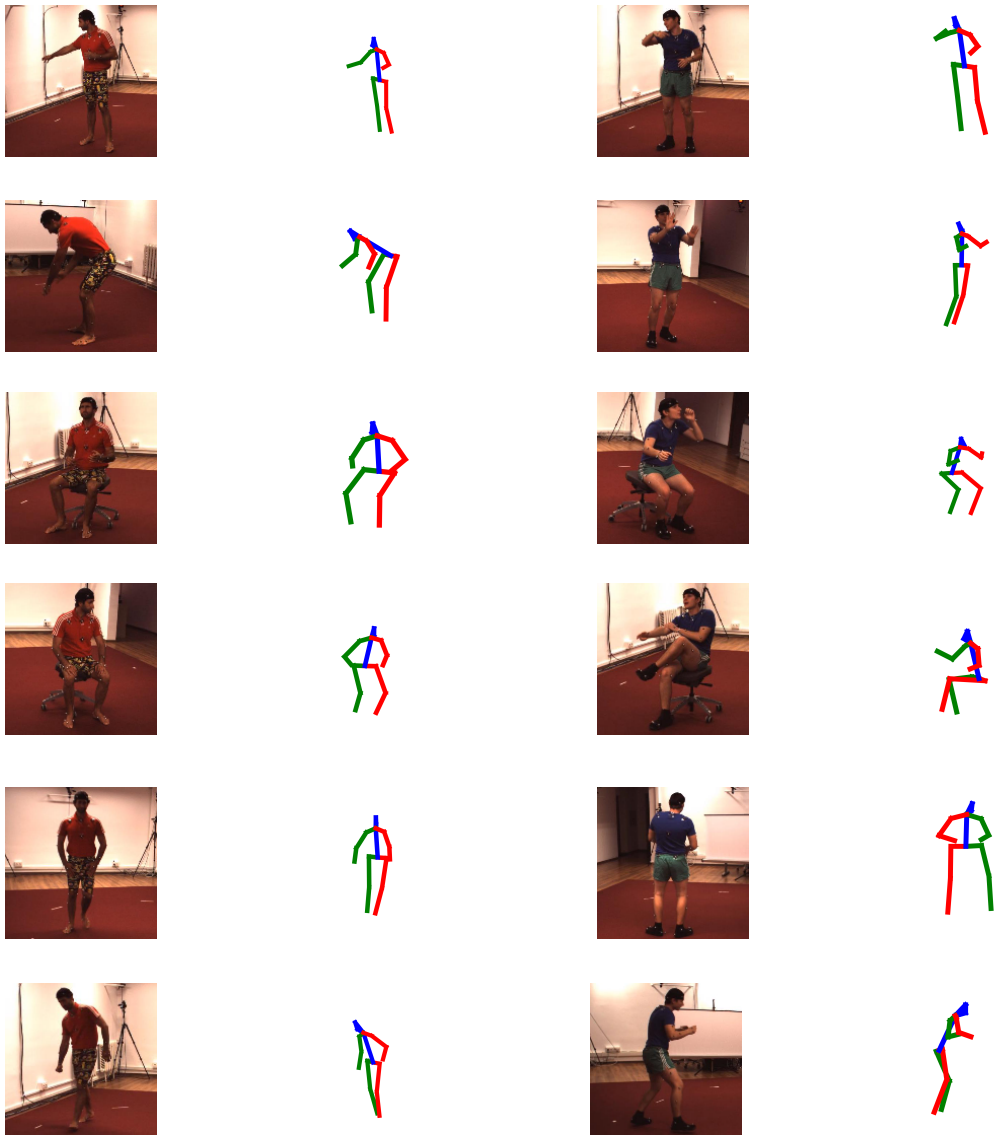
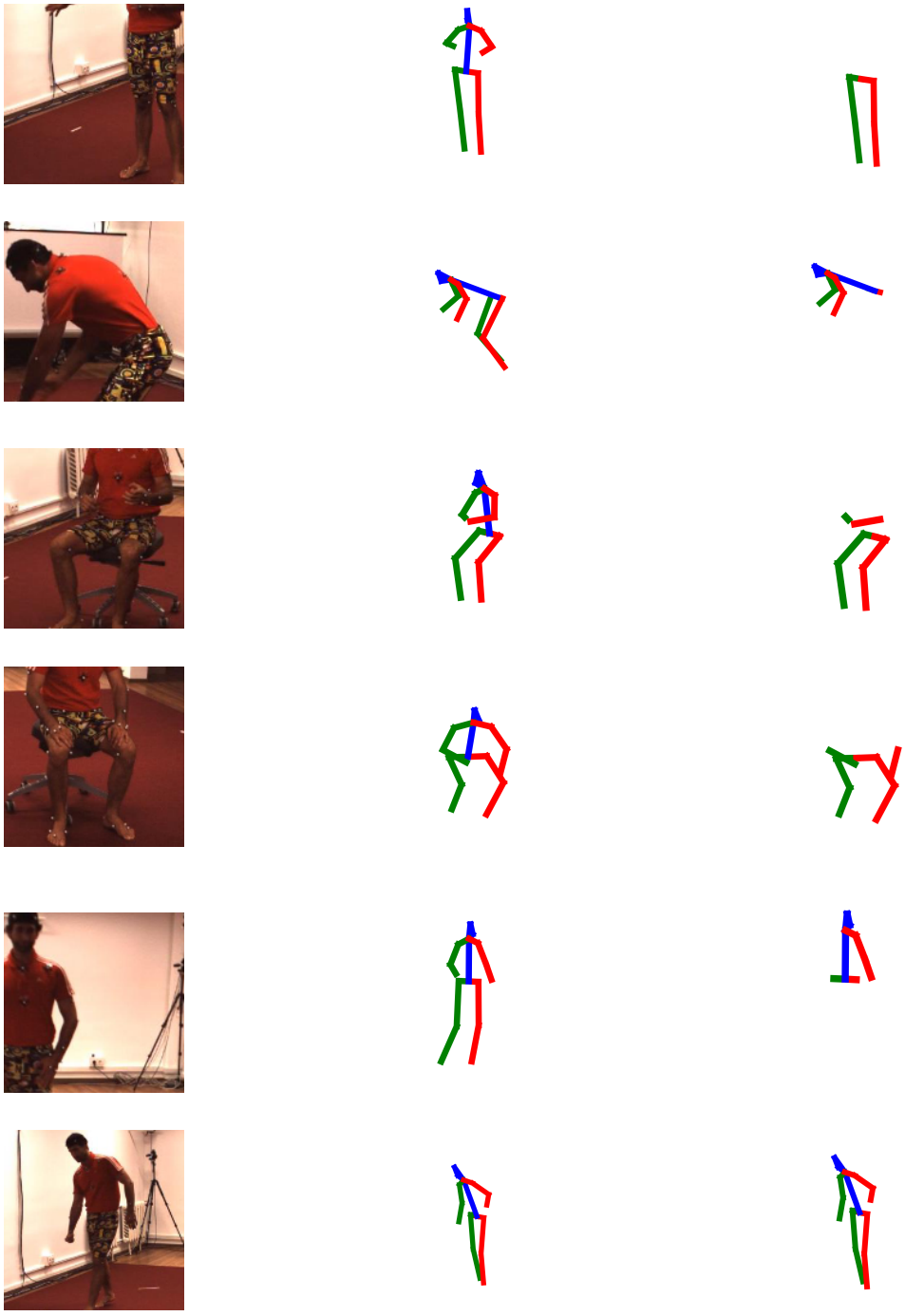


Figure 25: Subjective results of our method (Part3D) under full body presence.



(a) Input (b) Estimated Full Body (c) Estimated Present Joints

Figure 26: Subjective results of our method (Part3D) under partial body presence for subject 9 of the Human3.6M dataset.

(a) Input (b) Estimated Full Body (c) Estimated Present Joints

Figure 27: Subjective results of our method (Part3D) under partial body presence for subject 11 of the Human3.6M dataset.

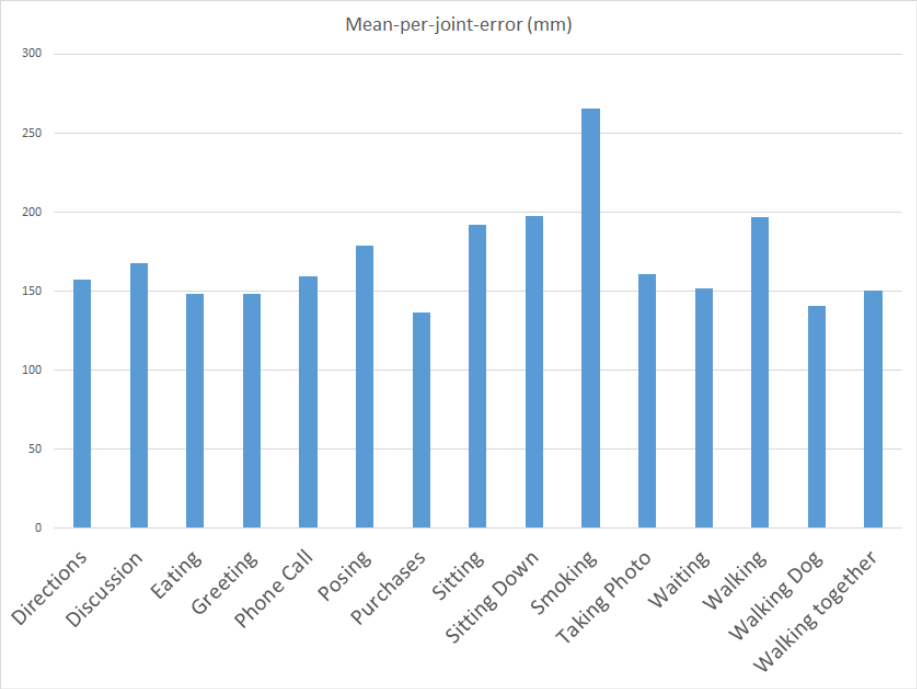
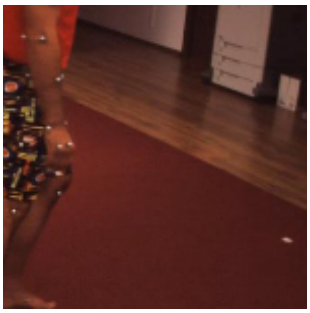
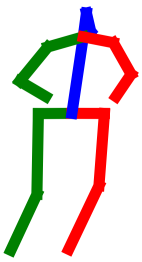


Figure 28: Average of full pose estimation from full body presence and partial pose estimation from partial body presence for different scenarios using direct pose estimation with separate training(Part3Ds).



(a)



(b)

Figure 29: An example of not-sufficient partial body presence that causes our method to not detect orientation.

Chapter 7

Conclusion

7.1 Summary and Conclusion

In this thesis, we presented methods to handle the 3D human pose estimation from 2D images under partial body presence. Although the human pose estimation for the full body [1, 2] (assuming the body is captured entirely in the input image) and for specific body parts (e.g., the head [33]) have been well addressed in the literature, we aimed to reconstruct a) the 3D full body pose from partial presence of the human body and b) the 3D partial pose of only the joints present in the input image.

In Chapter 4, we presented a method for regressing the full body pose from the partial body presence cases. Using an intermediate 2D pose regression network, here we divide the 3D pose regression into two steps: a) 2D pose regression from the 2D input image and b) 3D human pose regression from the extracted 2D human pose. In this approach, the task of the first network was to regress the full 2D pose regardless of the joints present and then the joints' locations were normalized to a distance between 0 and 1. The 3D pose regressor gets the normalized 2D poses as the input and regresses the 3D poses based on 2D results. The first step of this method is implemented using CNNs and the second step is implemented using fully-connected

feedforward neural networks.

In Chapter 5, we proposed a method for direct regression of the 3D human pose from 2D images with partial body presence, but without 2D pose estimation. We trained a CNN to estimate the full 3D human pose directly from the partial 2D image. We also proposed a detection stage which detects the joints present in the image. Integration of the results from the pose regression stage and the detection stage forms the partial output pose. For this method, we have used both joint training of the regression and detection networks and training them separately.

In Chapter 6, the results of our simulations were presented and compared to two of the state-of-the-art methods [1, 2]. Experimental observations demonstrate the effectiveness of our method. Although these methods have better performance while being fed with the cases of full body presence, our methods yield significantly better performance under partial body presence.

Our framework enables the 3D human pose estimation methods to work effectively under partial body presence. We also showed that the location of the joints absent from the input image can be successfully reconstructed (see Table 4). The network's performance for the joints absent is not as good as for the joints present, however. This is evident from the fact that the error for all the joints is more than when just measuring the error for the joints present in partial cases (see Table 2). We also examined different approaches for partial 3D human pose estimation, namely, using 2D intermediate stage, direct pose estimation with joint training of the regression and detection networks, and direct pose estimation with separate training of the networks. We showed that using similar convolutional layers, these methods have close performance; however, the direct estimation with separate training turned out to have the best performance among our three approaches due to the dedicated feature extraction in the regression network.

7.2 Future Work

In the following, we sum up some of the possible extensions for future work:

- An important extension to this work can be improvement of the performance while being fed by full body cases using a pre-training stage. Pre-training the network with the full body presence images and then training using both full and partial body cases may improve the performance under full body presence; however, the performance under partial body presence may be affected as well.

- Adding background subtraction can be an effective extension to this work. Removing background decreases the results' dependence on the environmental conditions since the background of the images can vary by a significant amount.

- Extending this work to video data is another possible extension. Although this work can be applied to video data by assuming different video frames as individual image data, there exists useful temporal information in the videos which can be helpful for 3D human pose estimation. An approach to consider is using the temporal information to track the pose in different video frames after estimating the pose in the first frame. Another approach is to feed multiple images (consecutive frames) to the CNN and try to estimate the poses for different moments jointly.

- Another possible extension to this work is adding priors (such as the priors used in [59] about the size of each of the body parts and their topology) about the human body. This prior can make the network generate more reasonable estimations according to the human body structure. The network is already familiar with human body structure since it has only been fed with poses following the human body structure. Adding explicit priors can guarantee the structure of the predictions, however.

- This work can be also extended to multi-person cases in which both the detection and regression stages work while having more than a single person present in the input image.

Bibliography

[1] 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 Trans. Graph.*, 2017, vol. 36, pp. 44:1–44:14.

[2] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, "Towards 3D human pose estimation in the wild: a weakly-supervised approach," in *Proc. IEEE Int. Conf. Computer Vision*, 2017.

[3] N. Sarafianos, B. Boteanu, B. Ionescu, and I.A. Kakadiaris, "3D human pose estimation: A review of the literature and analysis of covariates," in *Computer Vision Image Understanding*. 2016, vol. 152, pp. 1–20, Elsevier.

[4] Y. Yang and D. Ramanan, "Articulated human detection with flexible mixtures of parts," in *IEEE Trans. Pattern Anal. Machine Intell.*, 2013, vol. 35, pp. 2878–2890.

[5] A. Saxena, S. H. Chung, and A. Y. Ng, "3-d depth reconstruction from a single still image," Jan 2008.

[6] X. Zhou, M. Zhu, S. Leonardos, and K. Daniilidis, "Sparse representation for 3D shape estimation: A convex relaxation approach," in *IEEE Trans. Pattern Anal. Machine Intell.*, 2017, vol. 39, pp. 1648–1661.

[7] T.Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick, "Microsoft coco: Common objects in context," in *Proc. European Conf. Computer Vision*. Springer, 2014, pp. 740–755.

[8] S. Vosoughi and M. A. Amer, "Deep 3d human pose estimation under partial body presence," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, 2018, pp. 569–573.

[9] V. Ramakrishna, T. Kanade, and Y. Sheikh, "Reconstructing 3d human pose from 2d image landmarks," in *Proc. European Conf. Computer Vision*. Springer, 2012, pp. 573–586.

[10] L. Sigal, "Human pose estimation," in *Computer Vision: A Reference Guide*, pp. 362–370. Springer, 2014.

[11] X. Zhou, M. Zhu, S. Leonardos, K.G. Derpanis, and K. Daniilidis, "Sparseness meets deepness: 3D human pose estimation from monocular video," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2016, pp. 4966–4975.

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

[13] B. Wandt, H. Ackermann, and B. Rosenhahn, "3D reconstruction of human motion from monocular image sequences," in *IEEE Trans. Pattern Anal. Machine Intell.*, 2016, vol. 38, pp. 1505–1516.

[14] B. Tekin, A. Rozantsev, V. Lepetit, and P. Fua, "Direct prediction of 3D body poses from motion compensated sequences," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2016, pp. 991–1000.

[15] B. Tekin, I. Katircioglu, M. Salzmann, V. Lepetit, and P. Fua, "Structured prediction of 3D human pose with deep neural networks," in *British Machine Vision Conf.*, 2016.

[16] S. Li and A.B. Chan, "3D human pose estimation from monocular images with deep convolutional neural network," in *Asian Conf. Computer Vision*. Springer, 2014, pp. 332–347.

[17] M.T. Hagan, H.B. Demuth, M.H. Beale, and O. De Jess, *Neural network design*, Martin Hagan, 2014.

[18] V. Kurkova, "Kolmogorov's theorem and multilayer neural networks," in *Neural networks*. 1992, vol. 5, pp. 501–506, Elsevier.

[19] A. TC Goh, "Back-propagation neural networks for modeling complex systems," in *Artificial Intelligence in Engineering*. 1995, vol. 9, pp. 143–151, Elsevier.

[20] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv preprint arXiv:1412.6980*, 2014.

[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," in *Nature*. 2015, vol. 521, pp. 436–444, Nature Research.

[22] Y. Bengio et al., "Learning deep architectures for ai," in *Foundations and trends® in Machine Learning*. 2009, vol. 2, pp. 1–127, Now Publishers, Inc.

[23] S. Ioffe and Ch. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[24] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," in *arXiv preprint arXiv:1207.0580*, 2012.

[25] Nicholas H Bingham and John M Fry, *Regression: Linear models in statistics*, Springer Science & Business Media, 2010.

[26] A. Roy and S. Todorovic, "Monocular depth estimation using neural regression forest," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2016, pp. 5506–5514.

[27] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *Proc. European Conf. Computer Vision*. Springer, 2016, pp. 740–756.

[28] D. Silverman and J. A. Dracup, "Artificial neural networks and long-range precipitation prediction in california," in *Journal of applied meteorology*, 2000, vol. 39, pp. 57–66.

[29] A. Erol, G. Bebis, M. Nicolescu, R.D. Boyle, and X. Twombly, "Vision-based hand pose estimation: A review," in *Computer Vision Image Understanding*. 2007, vol. 108, pp. 52–73, Elsevier.

[30] C. Keskin, F. Kıraç, Y.E. Kara, and L. Akarun, "Real time hand pose estimation using depth sensors," in *Consumer depth cameras for computer vision*, pp. 119–137. Springer, 2013.

[31] J. Segen and S. Kumar, "Shadow gestures: 3d hand pose estimation using a single camera," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 1999, vol. 1, pp. 479–485.

[32] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2015, pp. 824–832.

[33] E. Murphy-Chutorian and M.M. Trivedi, "Head pose estimation in computer vision: A survey," in *IEEE Trans. Pattern Anal. Machine Intell.*, 2009, vol. 31, pp. 607–626.

[34] G. Fanelli, J. Gall, and L. Van Gool, "Real time head pose estimation with random regression forests," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2011, pp. 617–624.

[35] Y. Yan, E. Ricci, R. Subramanian, G. Liu, O. Lanz, and N. Sebe, "A multi-task learning framework for head pose estimation under target motion," in *IEEE Trans. Pattern Anal. Machine Intell.*, 2016, vol. 38, pp. 1070–1083.

[36] X. Liu, W. Liang, Y. Wang, S. Li, and M. Pei, "3D head pose estimation with convolutional neural network trained on synthetic images," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, 2016, pp. 1289–1293.

[37] M. Hofmann and D.M. Gavrila, "Multi-view 3d human pose estimation combining single-frame recovery, temporal integration and model adaptation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2009, pp. 2214–2221.

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

[39] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human pose estimation with iterative error feedback," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2016, pp. 4733–4742.

[40] J.J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Advances in neural information processing systems*, 2014, pp. 1799–1807.

[41] V. Ferrari, M. Marin-Jimenez, and A. Zisserman, "Progressive search space reduction for human pose estimation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2008, pp. 1–8.

[42] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2017, pp. 7291–7299.

[43] Y. Zhu and K. Fujimura, "Constrained optimization for human pose estimation from depth sequences," in *Asian Conf. Computer Vision*. Springer, 2007, pp. 408–418.

[44] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, et al., "Efficient human pose estimation from single depth images," in *IEEE Trans. Pattern Anal. Machine Intell.*, 2013, vol. 35, pp. 2821–2840.

[45] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images," in *Proc. IEEE Int. Conf. Computer Vision*, 2011, pp. 415–422.

[46] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, et al., "Efficient human pose estimation from single depth images," in *IEEE Trans. Pattern Anal. Machine Intell.*, 2013, vol. 35, pp. 2821–2840.

[47] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," in *Int. J. Computer Vision*. 1992, vol. 9, pp. 137–154, Springer.

[48] Ch. Bregler, A. Hertzmann, and H. Biermann, "Recovering non-rigid 3D shape from image streams," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2000, vol. 2, pp. 690–696.

[49] D.G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Computer Vision*, 1999, vol. 2, pp. 1150–1157.

[50] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," in *Int. J. Computer Vision*. 2004, vol. 60, pp. 91–110, Springer.

[51] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Proc. European Conf. Computer Vision*. Springer, 2006, pp. 404–417.

[52] V. Lepetit, J. Pilet, and P. Fua, "Point matching as a classification problem for fast and robust object pose estimation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2004, vol. 2, pp. II–II.

[53] J. Martinez, R. Hossain, J. Romero, and J.J. Little, "A simple yet effective baseline for 3D human pose estimation," in *Proc. IEEE Int. Conf. Computer Vision*, 2017, vol. 206, p. 3.

[54] B. Tekin, P. Márquez-Neila, M. Salzmann, and P. Fua, "Fusing 2D uncertainty and 3D cues for monocular body pose estimation," in *arXiv preprint arXiv:1611.05708*, 2016.

[55] G. Pavlakos, X. Zhou, K.G. Derpanis, and K. Daniilidis, "Coarse-to-fine volumetric prediction for single-image 3d human pose," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2017, pp. 1263–1272.

[56] M. Hofmann and D.M. Gavrila, "Multi-view 3d human pose estimation in complex environment," in *Int. J. Computer Vision*. 2012, vol. 96, pp. 103–124, Springer.

[57] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic, "3D pictorial structures revisited: Multiple human pose estimation," in *IEEE Trans. Pattern Anal. Machine Intell.*, 2016, vol. 38, pp. 1929–1942.

[58] E. Brau and H. Jiang, "3D human pose estimation via deep learning from 2D annotations," in *IEEE Int. Conf. 3D Vision*, 2016, pp. 582–591.

[59] E. Brau and H. Jiang, "A Bayesian part-based approach to 3D human pose and camera estimation," in *Proc. IEEE Int. Conf. Pattern Recognition*, 2016, pp. 1762–1767.

[60] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas, "Discriminative density propagation for 3d human motion estimation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2005, vol. 1, pp. 390–397.

[61] H. Ning, W. Xu, Y. Gong, and T. Huang, "Discriminative learning of visual words for 3d human pose estimation," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*. Citeseer, 2008, pp. 1–8.

[62] A. Haque, B. Peng, Z. Luo, A. Alahi, S. Yeung, and L. Fei-Fei, "Towards viewpoint invariant 3d human pose estimation," in *Proc. European Conf. Computer Vision*. Springer, 2016, pp. 160–177.

[63] 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 Trans. Pattern Anal. Machine Intell.*, 2014, vol. 36, pp. 1325–1339.