2016 Fourth International Conference on 3D Vision

3D Human Pose Estimation via Deep Learning from 2D annotations

Ernesto Brau Hao Jiang
Boston College
Chestnut Hill, MA 02467
{brauavil,jianghd}@bc.edu

Abstract

We propose a deep convolutional neural network for 3D human pose and camera estimation from monocular images that learns from 2D joint annotations. The proposed network follows the typical architecture, but contains an additional output layer which projects predicted 3D joints onto 2D, and enforces constraints on body part lengths in 3D. We further enforce pose constraints using an independently trained network that learns a prior distribution over 3D poses. We evaluate our approach on several benchmark datasets and compare against state-of-the-art approaches for 3D human pose estimation, achieving comparable performance. Additionally, we show that our approach significantly outperforms other methods in cases where 3D ground truth data is unavailable, and that our network exhibits good generalization properties.

1. Introduction

The ability to automatically extract the pose of a person from an image has many applications, including robotics and surveillance, among others. It is also important to many other areas within the computer vision field, such as activity recognition, tracking, and scene understanding. For this and other reasons, pose estimation has attracted a lot of attention in the last decade. Until recently, 2D pose estimation was the main focus of investigation, which resulted in well-known approaches like pictorial structure [10] and deformable parts models [33].

3D human pose estimation has also started to attract a lot of interest from researchers, especially in the activity recognition and tracking communities. Many different kinds of approaches have been tried, with the most success being achieved largely with generative modeling approaches, such as 3D versions of pictorial structure [5, 8, 2], pose-conditioned joint angle models [1], and approaches based on Gaussian processes [6]. Although progress has been made, reliably estimating 3D human pose from a single image remains an unsolved problem.

In general, human pose estimation is an extremely challenging task. The vast variance in appearance and the high-dimensional pose space are two of the dozens of issues faced when tackling the problem. Moving to the 3D realm only makes things more difficult, in large part due to the ambiguity inherent in obtaining a 3D understanding of a 2D image. Indeed, estimating any 3D structure from 2D observations is a fundamentally ill-posed problem due to the irreversible (and often unknown) camera projection.

Due to the growing availability of large datasets, discriminative approaches to pose estimation are beginning to gain traction. Ionescu et al. [11] recently used a random forest model for joint pixel part labeling and 3D human pose estimation. Later, Li et al. [18] trained a multi-task deep neural network regressor which predicts 3D joint positions from monocular images. Although these approaches have achieved some success, they rely on the availability of large sets of images annotated with 3D joint positions and camera parameters for training. This type of data is very difficult to generate, often requiring sophisticated motion capture setups. This places a significant limitation on the data collection process, and usually restricts the scenes to laboratory conditions. In contrast, annotating images with 2D joints is comparatively easy, and can be performed very quickly and inexpensively using services such as Amazon Mechanical Turk. Consequently, there exist dozens of 2D pose estimation datasets, featuring a wide-range of poses and appearance, background, and lighting conditions. The main goal of this work is to leverage the availability of such data for the 3D human pose estimation task.

In this paper, we present a deep convolutional neural network (CNN) to jointly predict 3D human pose and camera parameters from a single image, which learns from images annotated with only 2D joint locations (see Figure 1 for an illustration of our network architecture). We propose to address the ambiguity resulting from using 2D pose information by imposing prior knowledge on the 3D pose variables. We do this by placing a neuron layer on top of the 3D pose output that computes the lengths of the body parts and using the loss function to encourage 3D skeletons that have

978-1-5090-5407-7/16 $31.00 © 2016 IEEE
DOI 10.1109/3DV.2016.84

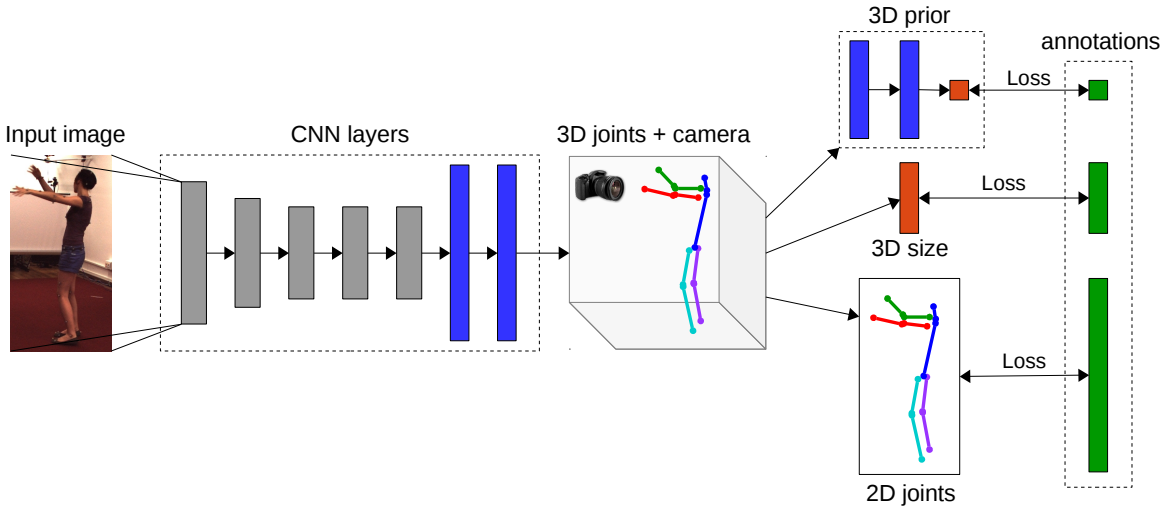


Figure 1. Our proposed deep learning architecture consists of three main components, a convolutional neural network (CNN), a camera projection and bone length computation layer, and a 3D pose prior network. During a forward pass of the network, an input image is passed through the CNN, which outputs a vector of 3D joint positions and camera parameters. This output is then fed into the camera projection layer, which computes the 2D joints positions. Similarly, the 3D joints are input to the bone length layer which computes the length of each body part. The 3D joints are also the input to the pose prior network, which outputs the log prior distribution value for the pose. During training, the final outputs are compared with the annotations under $\mathcal{L}_2$ norm loss functions. See Section 3 for details.

the correct proportions. The output layer also projects the 3D pose onto 2D, allowing us to use a simple L_2 norm cost objective for training. Additionally, we propose a simple neural network to learn the kinematic structure of 3D poses, which we train separately and is plugged into our main network to constrain the output. We demonstrate the viability of our approach by extensive evaluation on several benchmark datasets. We also perform experiments to show the generalization capability of our neural net.

The rest of the paper is organized as follows. In Section 2 we briefly summarize the related work. Section 3 discusses the CNN regression model in detail, Section 4 shows our experimental results, and Section 5 concludes.

2. Related work

Human pose estimation is one of the most studied problems in computer vision [5, 30, 4, 26, 12, 32, 31, 22, 13, 28, 29, 15]. 2D pose estimation approaches range from part-based models, such as pictorial structure [10] and deformable parts models [33] to deep neural networks [30]. In the last decade, 3D pose estimation has become a subject of much interest, attracting a wide variety of techniques and approaches.

There are several approaches where 3D pose is estimated from 2D landmarks [1, 24]. For example, Akhter and Black [1] train a pose-conditioned prior over 3D pose, and group body parts into sets to estimate 3D pose and camera from 2D joint locations. Ramakrishna et al. [24] propose an activity-independent prior over poses, and use a matching pursuit algorithm to estimate the 3D pose and camera from 2D landmarks. While effective, these methods suffer from the disadvantage that there is often no reliable way to obtain accurate 2D joint locations, or any other 2D landmarks.

The idea of pictorial structure, although originally used for 2D estimation, has been extended to 3D in several ways [5, 8, 2]. Belagiaonnis et al. [5] use a 3D pictorial structure model to estimate multiple poses in multiple views, by minimizing the energy function that arises from their model. Similarly, Amin et al. [2] use a mixture of pictorial structure models to estimate 3D pose from one or more views. Burenius et al. [8] use a similar model, and discretize the solution space and use the max-product algorithm to estimate the pose. The main drawback from the pictorial structure model is that, in general, it constrains the model to be discrete. While this works very well in 2D pixel space, it is more difficult to use in 3D continuous space, where one must discretize space, and the number of cells can very quickly become unmanageable.

Another scheme which is related to the work presented here uses approaches that require only a single image as input, and estimate both 3D pose and camera parameters [31, 28]. Simo-Serra et al. [28] generate pose candidates from part detections, then minimize the reprojection error to find the best pose. Wang et al. [31] model 3D pose as a linear combination of a set of pose bases, and infer the coefficients for a given image using the alternating direction method. One of the disadvantages of this approach is the reliance on hard 2D pose detection, which is often inaccurate.

In contrast, we train a regression function directly from images, without any pre-processing of the data. Also, we use a full camera model, whereas they assume a weak-perspective camera model. Finally, Radwan et al. [23] synthesize a set of poses from part detections and a self-occlusion model, then enforce a set of kinematic and orientation constraints. Like our work, this approach is able to deal well with self occlusion. One key difference is that they assume camera parameters are known, which simplifies the problem. In addition to the differences mentioned above, these methods all use generative models, whereas we train a discriminative regressor.

Bo et al. train a Twin Gaussian process model [6] on images annotated with ground truth 3D poses. Although they show excellent results, they only evaluate their approach on a single benchmark dataset. Ionescu et al. [11] propose a random forest model which they train in an iterative way for both pixel classification and pose estimation. Li et al. [18] propose a deep convolutional neural network similar to our own, but it is trained directly on 3D pose annotations. Further, all of these approaches assume camera parameters to be available at least during training, and in one case even at test-time [11]. Finally, the well-known DeepPose work [30] also uses deep learning for 2D pose estimation in a similar way to our approach.

3. Model and representation

Given an image of a person, our goal is to estimate their 3D pose, as well as the parameters of the camera used to generate the image. We denote a pose by $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_M)$, where $\mathbf{y}_m = (y_{m1}, y_{m2}, y_{m3})$ represents the 3D location of the mth joint, with $M = 14$ (three joints for each limb, the neck, and the head). We use the perspective camera model and we assume that the camera is at the world origin, and is aligned with the world coordinate frame. Consequently, we only need to represent four intrinsic parameters: the focal lengths in each axis α_u and α_v, and the principal point coordinates u_0 and v_0. We use $\mathbf{c}$ to denote both the parameter vector $(\alpha_u, \alpha_v, u_0, v_0)$ and the function

$$\mathbf{c}(\mathbf{y}_m) = \frac{1}{y_{m3}} \begin{pmatrix} \alpha_u y_{m1} + u_0 y_{m3} \\ \alpha_v y_{m2} + v_0 y_{m3} \end{pmatrix} \tag{1}$$

which projects 3D joint $\mathbf{y}_m$ onto its 2D image location. Further, we let $\mathbf{c}(\mathbf{y}) = (\mathbf{c}(\mathbf{y}_1), \ldots, \mathbf{c}(\mathbf{y}_M))$ represent the projection of a whole pose $\mathbf{y}$ onto the image, resulting in a $2M$-dimensional vector of 2D joint positions. We will also make use of the body parts (bones) of a pose, which we denote by $\mathbf{w}_j$, $j = 1, \ldots, 10$, representing the eight upper and lower arms and legs, the neck, and the torso. Additionally, we define l_j to be the length of bone j. Finally, we define a function $l : \mathbb{R}^{3M} \to \mathbb{R}^J$ which computes the squared lengths of all body parts associated with $\mathbf{y}$, i.e., $l(\mathbf{y}) = (l_1^2, \ldots, l_{10}^2)$.

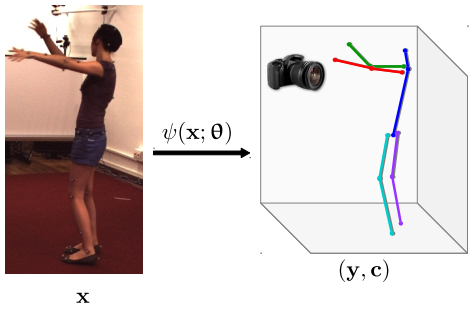


Figure 2. We formulate 3D human and camera pose estimation as a regression problem, where the goal is to estimate the regression function $\psi(\cdot; \theta)$ which maps the inputs, in this case an image $\mathbf{x}$, to the target values, in our case the 3D pose $\mathbf{y}$ and camera $\mathbf{c}$. The regression function has parameters θ, which in our model is given by a convolutional neural network, which we must learn from labeled data $(\mathbf{x}, \mathbf{u})$, where $\mathbf{u}$ is the 2D pose.

For example, if $\mathbf{y}_1$ and $\mathbf{y}_2$ are the left and shoulder and elbow of pose $\mathbf{y}$, then the component of $l(\mathbf{y})$ corresponding to the left upper arm is given by $(\mathbf{y}_1 - \mathbf{y}_2)^\mathsf{T}(\mathbf{y}_1 - \mathbf{y}_2)$.

Prior on 3D pose As mentioned above, we need to explicitly enforce the 3D pose constraints to address some of the ambiguity resulting from the use of 2D annotations. We use the prior distribution proposed by Brau and Jiang [7], which consists of two factors: a kinematic prior and a self-intersection prior. The kinematic prior places a 4-dimensional Von Mises-Fisher distribution on the orientation of the neck and the upper arms and legs, e.g., $\mathbf{w}_j \sim \mathrm{VMF}(\boldsymbol{\mu}_j, \kappa_j)$ for bone j, and a Gamma distribution on the bending angle of the lower arms and legs, e.g., $\mathbf{w}_{j'} \sim \mathrm{Gamma}(k_{j'}, \psi_{j'})$ for bone j'. The self-intersection prior is a distribution on the entire pose vector that penalizes poses which have bones that intersect in space. The result is a prior distribution over the space of 3D poses, which we denote by $p(\mathbf{y} \mid \boldsymbol{\phi})$, where $\boldsymbol{\phi}$ are the distributional parameters of the different factors, e.g., $\boldsymbol{\mu}_j$ and $k_{j'}$ above. We obtained values for $\boldsymbol{\phi}$ from the authors [7], who trained the distribution on the CMU mocap dataset.

3.1. Regression for 3D pose estimation

We wish to estimate the regression function $\psi(\mathbf{x}; \theta) \in \mathbb{R}^{3M+4}$ which maps an input image $\mathbf{x}$ to a 3D pose $\mathbf{y}$ and camera $\mathbf{c}$, where θ denotes the model parameters (see Figure 2). We train the model θ on labeled images $\mathcal{D} = \{(\mathbf{x}, \mathbf{u})\}$, where $\mathbf{x}$ is an image and $\mathbf{u} \in \mathbb{R}^{2M}$ is the ground-truth 2D pose, consisting of M joints, i.e., $\mathbf{u} = (\mathbf{u}_1, \ldots, \mathbf{u}_M)$. Then, given a test image $\mathbf{x}_*$, the predicted 3D pose and camera is given by $(\mathbf{y}_*, \mathbf{c}_*) = \psi(\mathbf{x}_*; \hat{\theta})$, where $\hat{\theta}$ are the learned model parameters.

We use the deep learning framework, whereby ψ is a deep convolutional neural network (CNN) consisting of

several layers, each representing different linear and non-linear functions which are composed to give ψ. The first layer takes an input RGB image of a fixed size, and the last layer outputs the target values for the regression, in our case the 3D pose $\mathbf{y}$ and the camera $\mathbf{c}$.

Note that we train our network on images annotated with 2D joints, but our output variables are in 3D space. This has two implications. First, we must project the 3D pose onto the image to compute the learning objective function, an operation which we encode in a network layer. More importantly, we must address the scale and orientation ambiguity inherent in estimating 3D information from images, i.e., there are an infinite number of 3D joints $\mathbf{y}$ which project onto the same 2D joint locations. To resolve the scale ambiguity, we use an additional network layer to enforce constraints on the lengths of the body parts $\mathbf{w}_1, \ldots, \mathbf{w}_{10}$ associated to $\mathbf{y}$. Specifically, our objective function penalizes elements of $l(\mathbf{y})$ which are far from the average lengths of human body parts (in meters, obtained from human population data). For the orientation ambiguity, we introduce a separate neural network to learn the 3D pose prior described in Section 3. This prior network takes as input a 3D pose $\mathbf{y}$ and outputs the value of the log-prior distribution $\log p(\mathbf{y})$. The prior network is independently pre-trained and plugged into the overall deep network with its weights fixed.

3.2. Network architecture

Our overall network architecture is shown in Figure 1. It has three main components: the convolutional neural network (CNN) portion, the projection and part-length layers, and the 3D pose prior network. The CNN feeds into the projection and length network, as well as into the prior network; these in turn contain the final output layers, which are connected to the training outputs. During a forward pass of the network, an input image $\mathbf{x}$ is fed into the CNN, which outputs a 3D pose vector $\mathbf{y}$ and a camera vector $\mathbf{c}$. Then, the projection and part-length layers use this as input and compute 2D joint locations $\mathbf{c}(\mathbf{y})$ and bone lengths $l(\mathbf{y})$, and the pose prior network computes the prior distribution value $\log p(\mathbf{y})$ of 3D pose $\mathbf{y}$.

The convolutional network The architecture of our CNN is based on the well-known AlexNet [17] introduced by Krizhevsky et al. in 2012 and used for image classification. The net contains eight layers with learnable weights. The first five are convolutional layers and the final three are fully-connected layers. The first convolutional layer uses 48 11×11 filters and the second one 256 5×5 filters. The remaining three convolutional layers apply 384, 256, and 256 filters, respectively, of size 3×3. All five layers are composed with rectified linear units (ReLU) for non-linearity, as well as max-pooling layers, all of which contain no learnable parameters. The three fully-connected layers contain 4096 neurons each. The final fully-connected layer feeds into $3M + 4 = 46$ (recall that $M = 14$) output neurons, consisting of the 3D joints $\mathbf{y}$ and camera parameters $\mathbf{c}$. See Figure 3 for an illustration of the architecture.

The projection and part-length layers As mentioned above, we add additional (parameter-free) layers which take as input a 3D pose $\mathbf{y}$ and a camera vector $\mathbf{c}$ from the CNN, and output $\mathbf{c}(\mathbf{y})$ and $l(\mathbf{y})$. We do this by constraining the connections between neurons and applying the appropriate operations. For example, the neurons corresponding to y_{11}, y_{13}, α_u, and u_0 are connected to a single neuron in the final output layer and computes $\frac{1}{y_{13}}(\alpha_u y_{11} + u_0 y_{13})$. Similarly, the neurons corresponding the shoulder and elbow joints $\mathbf{y}_1$ and $\mathbf{y}_2$ connect to a single output layer associated with upper arm and computes its squared length $(\mathbf{y}_1 - \mathbf{y}_2)^\mathsf{T}(\mathbf{y}_1 - \mathbf{y}_2)$. Figure 4 illustrates this layer design. We note that the same result could be obtained by performing these operations in the loss function; however, using network operations seamlessly takes advantage of the features present in CNN tools (such as Caffe[14]), such as auto-differentiation and GPU implementation. During training, these layers feed into the annotated 2D joint locations and a fixed 3D length vector.

The 3D pose prior network Figure 5 shows an illustration of the proposed pose prior neural network. The network takes a pose vector $\mathbf{y}$ as input and outputs the value of the log-prior distribution evaluated at it, $\log p(\mathbf{y})$. It contains two hidden, fully-connected layers, each with 1024 neurons and a $\tanh$ activation function. The output layer has a single node, and feeds into the training data which consists of the true values of $\log p(\mathbf{y})$. As mentioned before, we train this network separately on a set of 3D pose vectors from the CMU mocap dataset, along with their corresponding prior values, computed analytically [7]. The prior network is trained once, and plugged into the overall network with its weights fixed.

3.2.1 Training

The final output layer of our network contains $2M+J+1 = 39$ ($M = 14$, $J = 10$) neurons. We train a linear regression on top of it by minimizing the $\mathcal{L}_2$ distance between the predicted and the training outputs. As mentioned above, given an input image $\mathbf{x}$, our network produces three different types of outputs: 2D joints $\mathbf{c}(\mathbf{y})$, bone lengths $l(\mathbf{y})$, and prior value $\log p(\mathbf{y})$, where $\mathbf{y}$ and $\mathbf{c}$ are the values of the 3D joint and camera parameter layers, respectively, i.e., $(\mathbf{y}, \mathbf{c}) = \psi(\mathbf{x}; \boldsymbol{\theta})$ (see Section 3.1).

We define the following loss functions for each of these outputs. For the 2D joints, we use the $\mathcal{L}_2$ distance between $\mathbf{c}(\mathbf{y})$ and $\mathbf{u}$, the annotated 2D joint locations corresponding

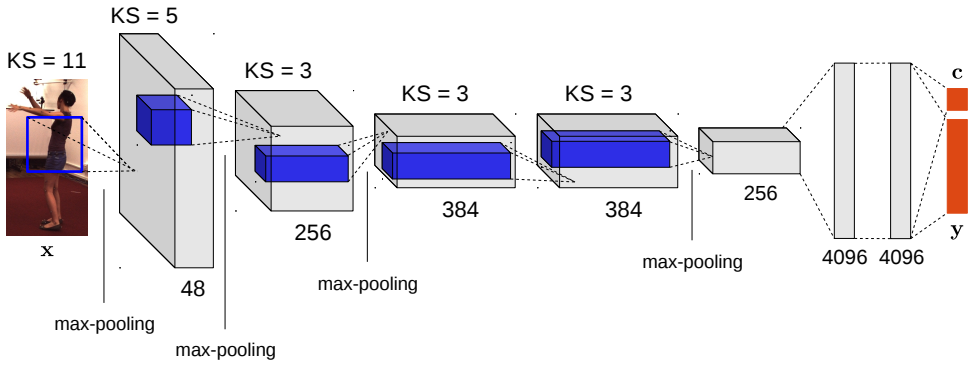
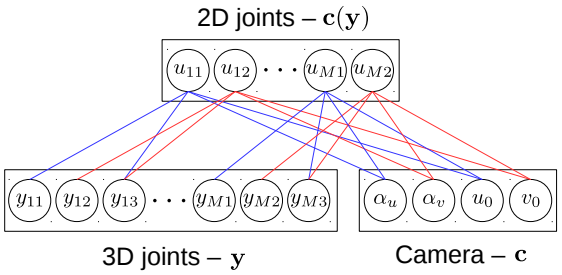
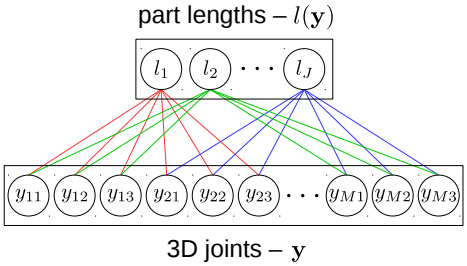


Figure 3. Our CNN architecture consists of several layers of convolution with different kernel sizes (KS), illustrated by the blue volumes, followed by rectified linear units (ReLU) and max-pooling. After three fully-connected layers, we have output neurons (in orange) corresponding to the set of 3D joints $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_M)$ and the camera parameters $\mathbf{c}$. See Section 3 for details.



(a) Projection layers



(b) Part-length layers

Figure 4. Illustration of the final output layer of our neural net. (a) The bottom row of neurons represents the first output layer, consisting of the 3D joint and camera neurons. The top row represents the final output layer, which contains the projected joints on the image plane (in 2D) and the lengths of the body parts. Notice the connections between the neurons. Specifically, u_{11} is connected to y_{11}, y_{13}, α_u, and u_0, and computes the projection operation $\frac{1}{y_{13}}(\alpha_u y_{11} + u_0 y_{13})$. (b) Similarly, the nodes corresponding to $\mathbf{y}_1 = (y_{11}, y_{12}, y_{13})$ and $\mathbf{y}_2 = (y_{21}, y_{22}, y_{23})$ feed into the first part length neuron l_1, which computes the length of bone $\mathbf{w}_1$.

to input image $\mathbf{x}$, i.e., $L_1(\theta; \mathbf{x}, \mathbf{u}) = \|\mathbf{c}(\mathbf{y}) - \mathbf{u}\|_2^2$. For bone lengths we use the $\mathcal{L}_1$ norm loss function with respect to a vector of the squared average human body part lengths, denoted by $\mathbf{l}$, which we obtained from publicly

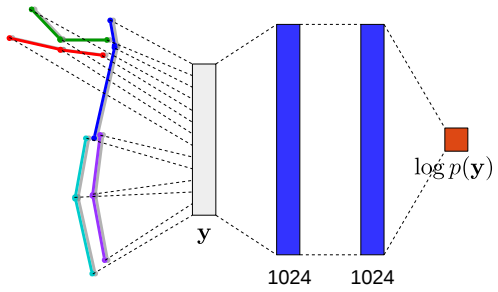


Figure 5. An illustration of the pose prior network architecture. The input layer has $3M$ neurons, and represents a 3D pose vector $\mathbf{y}$. This is followed by two fully-connected layers of size 1024 each (shown in blue) with $\tanh$ activation functions. Finally, the output layer (shown in orange) contains a single neuron, which represents the value of prior distribution of the input pose, $\log p(\mathbf{y})$. See Section 3.2 for more details.

available population data, and is the same for all training images. This has the effect of encouraging 3D poses of size which is close to average. The bone length loss is, thus, $L_2(\theta; \mathbf{x}, \mathbf{l}) = \|l(\mathbf{y}) - \mathbf{l}\|_1$. Similarly, for the pose prior output we use the $\mathcal{L}_2$ norm loss given by $L_3(\theta; \mathbf{x}, p^*) = \|\log p(\mathbf{y}) - p^*\|_2^2$, where $p^* = \max_{\mathbf{y}} \log p(\mathbf{y})$ is the log-prior distribution evaluated at its mode. This loss function penalizes poses that have a low prior.

The overall loss function $L(\theta; \mathcal{D}, \mathbf{l}, p^*)$ for dataset $\mathcal{D}$ is given by

$$\sum_{(\mathbf{x},\mathbf{u})\in\mathcal{D}} (w_1 L_1(\theta; \mathbf{x}, \mathbf{u}) + w_2 L_2(\theta; \mathbf{x}, \mathbf{l}) + w_3 L_3(\theta; \mathbf{x}, p^*)), \tag{2}$$

where w_1, w_2, and w_3 are weights that control the influence of each loss component, and which we set empirically. We minimize Eq. (2) with respect to θ using the back-propagation method, using a GPU implementation available

in the Caffe suite [14]. We process the data in batches of size 64, and use an adaptive gradient algorithm for optimization. The learning rate parameter is set to 0.01 and the weight decay to 0.0005. Additionally, we normalize the data by subtracting out the mean. Finally, we augment the data by generating random image rotations, as well as flipping the images both horizontally and vertically.

4. Experiments and results

We evaluate our approach in three different ways. First, we test the performance of our algorithm on standard benchmark datasets and compare it with other 3D human pose estimation approaches. Next, we test the generalization power of our system by assessing its performance when trained exclusively on datasets which only provide 2D ground truth annotations. Finally, we also report performance on the 2D human pose estimation task.

Datasets We use two 3D pose datasets in our experiments: HumanEva-I [27] and Human3.6M [13]. The HumanEva-I dataset consists of thousands of frames featuring three subjects performing five different actions (walking, jogging, throwing/catching, gesturing, and boxing) and is divided into training, validation, and testing sets. The Human3.6M dataset contains 3.6 million poses consisting of 11 subjects acting in 17 scenarios (discussion, smoking, talking on the phone, etc.). It is divided into training and testing sets, each featuring different subjects. For both datasets, 3D joint annotations are provided for training and validation sets, as well as ground truth camera parameters.

We also make use of two 2D pose datasets, Leeds Sports Pose [16] and MPII Human Pose [3]. The Leeds Sports Pose dataset consists of 2000 images featuring sports poses. Similarly, the MPII Human Pose dataset contains roughly 25000 images of a wide-range of poses. Both datasets are separated into training and test sets, and both are annotated with 2D body joint locations.

Evaluation metrics For the 3D human pose estimation task we use the mean joint error metric, also known as the mean per joint position error (MPJPE), which measures the distance between the ground truth and estimated 3D joints averaged over all the body joints. For 2D pose estimation we use the probability of correct pose (PCP) metric, which is the fraction of the body parts which are labeled correctly, where a body part is labeled as correct if its segment endpoints lie within 50% of the length of the ground-truth annotated endpoints.

4.1. 3D benchmarks

We evaluate our method on two 3D benchmark datasets, HumanEva-I and Human3.6M, and compare our results against several state-of-the-art approaches which use an experimental setup comparable to ours.

		Walking	Jogging	Throwing	Gesturing	Boxing
Ours	S1	70.3	73.2	129.4	40.9	69.2
	S2	68.1	75.8	110.3	94.8	80.3
	S3	80.8	68.1	99.1	63.1	90.8
Radwan [23]	S1	75.1	79.2	-	-	-
	S2	99.8	89.8	-	-	-
	S3	93.8	99.4	-	-	-
Simo-Serra [28]	S1	99.6	109.2	-	-	-
	S2	108.3	93.1	-	-	-
	S3	127.4	115.8	-	-	-
Wang [31]	S1	79.1	62.6	-	-	-
	S2	75.7	77.7	-	-	-
	S3	85.3	54.4	-	-	-
Bo [6]	S1	71.7	72.6	149.7	12.3	55.8
	S2	46.5	72.4	91.8	95.2	96.8
	S3	80.2	73.2	-	36.2	81.8

Table 1. Quantitative comparison of our algorithm with state-of-the-art methods on the HumanEva-I dataset. The reported values are the average joint errors (in mm) for each activity (Walking, Jogging, etc.) and sequence (S1, S2, and S3), where "-" indicates no results were reported for that configuration. All approaches use HumanEva-I for training except [23], and our approach and [28] estimate the camera jointly with the pose.

4.1.1 HumanEva-I

We train our network on the training image set, using all subjects (S1, S2, and S3) and all views (C1, C2, C3). Since HumanEva-I does not directly provide 2D joint locations, which are needed to train our model, we project the 3D joint annotations onto the image plane using the appropriate camera parameters. Additionally, since all images in the dataset have the same resolution, we use them as-is, without scaling or cropping. Table 1 shows the performance of our approach on the validation images of the dataset, measured in average error in mm, as well as the results reported by four state-of-the-art methods [23, 28, 31, 6]. In general, our performance is comparable to that of the reported approaches, and we outperform all of them on some activities. It is worth noting that all of these methods are trained on 3D data, whereas our model is trained strictly on 2D annotations, and that only one of them [28] (besides ours) estimates the camera parameters. Further, the approach by Bo et al. [6] assumes background subtraction is available. Figure 6 shows some examples of predicted poses.

4.1.2 Human3.6M

We compare our performance on the Human3.6M dataset with another deep learning approach [18], as well as with the baseline method provided with the dataset [13]. Following [18], we use the images from five subjects (S1, S5, S6, S7, and S8) for training, and from two different subjects (S9 and S11) for testing. We also crop the images

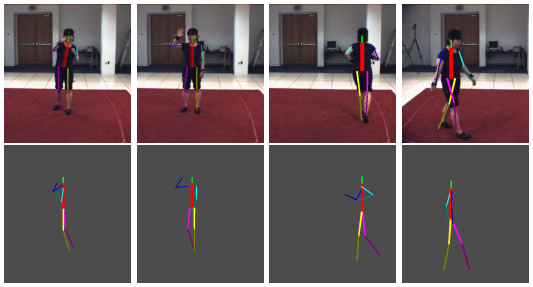


Figure 6. Sample results on the HumanEva-I dataset. The top row shows four sample images, one from each activity (from left to right: boxing, gesturing, jogging, walking), with the 3D pose superimposed. The bottom row shows the pose from an alternate angle. Note that the torso is not part of the model, and is shown for clarity.

	Discuss	Eat	Greet	Photo	Walk	WalkDog	H80K
Ours	141.1	90.3	117.9	189.3	59.8	137.1	82.1
Li [18]	148.8	104.0	127.2	189.1	77.6	146.6	-
Baseline [13]	183.1	132.5	162.3	206.5	97.1	177.8	-
Ionescu [11]	-	-	-	-	-	-	92.0

Table 2. Quantitative comparison of our algorithm with state-of-the-art methods, as well as a baseline [13] algorithm, on the Human3.6M dataset. The reported values are the average joint errors (in mm) for six activities and for the Human80K set (final column). In all cases, we follow the same experimental setup as the other approaches (see Section 4.1).

around the subject, using the foreground masks provided with the dataset, and scale them to 224×224. The results are reported in Table 2, where we can see that we outperform the baseline method, while obtaining comparable performance to [18]. We also evaluate our approach on the Human80K set, which consists of roughly 80000 images (cropped around the subjects) from the Human3.6M dataset. The right-most column on Table 2 shows our results, as well as the performance of the kernel dependency estimation (KDE) method introduced by Ionescu et al. [11]. As before, we adopt their training/testing splits for a valid comparison, and note that our method is the only one which estimates the camera in addition to the pose. See Figure 7 for examples of predicted poses on this dataset.

4.2. Training on 2D datasets

One of the strengths of our approach is the ability to learn 3D human pose structure entirely from 2D annotations, which allows us to train on a more diverse set of images for which 3D annotations are not available. To demonstrate this, we designed an experiment where we train our neural network model on images from 2D pose estimation benchmark datasets, and we evaluate its 3D pose predictions on 3D benchmark datasets. Specifically, we train our model on the training images of the Leeds Sports Pose (LSP) and MPII Human Pose (MPIIHP) datasets, consisting of a total of roughly 15000 images. As before, we use provided

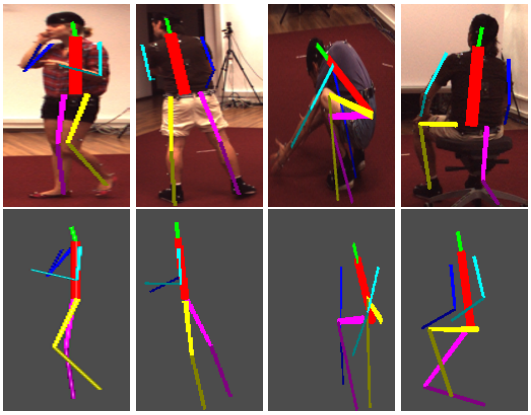


Figure 7. Sample results on the Human3.6M dataset. The top row shows four sample images (including two particularly difficult ones) from different subjects with the 3D pose superimposed. The bottom row shows the pose from an alternate angle. Note that the torso is not part of the model, and is shown for clarity.

	HumanEva-I		Human80K
	Walking	Jogging	
Ours	70.2	79.7	97.0
Ionescu [11]	97.7	94.7	109.9
Baseline CNN	101.3	113.5	122.6

Table 3. Performance of three approaches on the HumanEva-I and Human80K datasets, when trained on 2D pose estimation datasets LSP and MPII. The values are the average joint errors, measured in mm. For [11] and the baseline convolutional neural network (CNN), training data was obtained by reconstructing the 3D joints from the annotations in LSP and MPII. See Section 4.2 for details.

subject-centered crops and we scale the images to the same size. We then evaluate the trained network on the Walking and Jogging sequences (Subject S1) of the HumanEva-I dataset, as well as on the training images of Human80K.

We complete the experiment by comparing our performance with methods which learn exclusively from 3D joint data. To do this, we first estimate the 3D joint locations from the 2D annotations of the LSP and MPIIHP training sets using a 3D human pose reconstruction algorithm [1]. We use the resulting 3D joints to train the model from the KDE method [13] referenced above, using the code made available from the authors[1], and we test it on the HumanEva-I S1 Walking/Jogging images and on the Human80K training set (the same testing images used for our network, see above). We also implement a convolutional neural network that learns from 3D joint data as described in [18], which we train and test using the same setup.

Figure 8 shows some example predictions from our model, and Table 3 compares the performance of all three methods with the mean joint error metric, measured in mm. Our approach achieves a significantly smaller error in all

[1]This approach also requires pixel-level labels of human body parts; we estimated them using a fully convolutional neural network (similar to [19]) trained on the Human80K dataset.

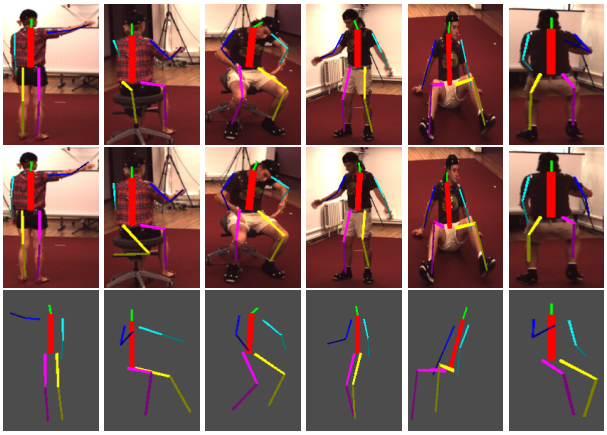


Figure 8. Sample results on the Human80K dataset, with models trained on 2D datasets LSP and MPII Pose. The top row shows images with the pose estimated by our approach superimposed. The middle row shows the poses obtained by [11]. The last row shows a different view of the pose obtained by our CNN. Note that the torso is not part of the model, and is shown for purposes of clarity. In general, these images show our model reconstructing more accurate poses, avoiding major mistakes such as in the second and sixth rows. This is likely due to the fact that our model is able to learn from 2D annotations, whereas [11] is designed to train on ground truth 3D annotations and pixel labels.

	Torso	Up. leg	Lo. leg	Up. arm	Lo. arm	Head
Ours	88.0	70.8	68.5	59.3	41.4	79.6
Chen [9]	92.7	82.9	77.0	69.2	55.4	87.8
Pishchulin [21]	88.7	78.9	73.2	61.8	45.0	85.1
Ouyang [20]	88.6	77.8	71.9	61.9	45.4	84.3
Ramakrishna [25]	88.1	79.0	73.6	62.8	39.5	80.4
Toshev [30]	78.0	71.0	56.0	38.0	-	-

Table 4. Quantitative evaluation of the 2D human pose estimation task on the Leeds Sports Pose dataset. The values in each column correspond to the PCP measure obtained for the corresponding body part, computed using a 50% threshold, using observer-centric (OC) annotations. An entry of "-" indicates the values were not reported by the authors.

three datasets. This is expected, as the two competing models were trained on *estimated* 3D joints, which are significantly inferior to ground truth annotations. In contrast, our model is trained on manually annotated 2D information, from which we learn 3D pose and camera structure.

4.3. 2D benchmarks

We also evaluate our method on the Leeds Sports Pose dataset. We use the training and testing images for learning and prediction, respectively, scaled to fit the CNN size. For this experiment, during testing we output the 2D joint positions $\mathbf{c}(\mathbf{y})$ instead of the 3D joints $\mathbf{y}$ and camera $\mathbf{c}$. We measure performance using the probability of correct pose (PCP) metric on our predicted 2D poses. Table 4 summarizes our results, as well as the results obtained by several other 2D human pose estimation approaches, as reported by the respective authors. Our model is able to achieve performance comparable to the state-of-the-art. This provides further evidence that of robustness of our approach, as well as of the quality of 3D pose and camera estimates. Figure 9 shows example pose predictions output by our model.

Figure 9. Sample results on the LSP dataset. For this experiment, we trained our CNN on the training images of the LSP and the MPII Human Pose datasets. The top row shows the original frames with the predicted pose superimposed on the image, and the second row shows the predicted pose from a different view. As these images show, our model is able to predict high-quality poses even for the most difficult images.

5. Conclusion

We present a deep convolutional neural network for 3D human pose and camera estimation that is capable of learning from 2D joint annotations. To our knowledge, this is a novel approach. We avoid the projective ambiguity by incorporating prior knowledge into the model in the form of average human limb sizes in 3D, specified in world units, and by training a simple deep neural network that learns a 3D prior over poses that has factors corresponding to kinematic and self-intersection constraints. We enforce these constraints, as well as the camera projection operation, using a network layer whose output is designed to be fed directly into a simple loss function, such as L_2 norm.

We demonstrated the viability of our model with extensive quantitative evaluation, achieving performance comparable to the state-of-the-art. Additionally, we show our model has the capacity to generalize across different datasets, making it all the more useful. Importantly, we showed that we can train our model on images from datasets for which only 2D joint annotations exist, and achieve high performance on 3D joint prediction.

Acknowledgments

This work was supported by NSF Grant IIS-1018641, as well as by a gift from NVIDIA.

References

[1] I. Akhter and M. J. Black. Pose-conditioned joint angle limits for 3d human pose reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1446–1455, 2015.

[2] S. Amin, M. Andriluka, M. Rohrbach, and B. Schiele. Multi-view pictorial structures for 3d human pose estimation. In *British Machine Vision Conference (BMVC)*, September 2013.

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

[4] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1014–1021. IEEE, 2009.

[5] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic. 3d pictorial structures for multiple human pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1669–1676. IEEE, 2014.

[6] L. Bo and C. Sminchisescu. Twin gaussian processes for structured prediction. *International Journal of Computer Vision*, 87(1-2):28–52, 2010.

[7] E. Brau and H. Jiang. A Bayesian part-based approach to 3d human pose estimation. In *To appear in International Conference on Pattern Recognition (ICPR)*, 2016.

[8] M. Burenius, J. Sullivan, and S. Carlsson. 3d pictorial structures for multiple view articulated pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3618–3625. IEEE, 2013.

[9] X. Chen and A. Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[10] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[11] C. Ionescu, J. Carreira, and C. Sminchisescu. Iterated second-order label sensitive pooling for 3d human pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1661–1668. IEEE, 2014.

[12] C. Ionescu, F. Li, and C. Sminchisescu. Latent structured models for human pose estimation. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2220–2227. IEEE, 2011.

[13] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(7):1325–1339, 2014.

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

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

[16] S. Johnson and M. Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *Proceedings of the British Machine Vision Conference*, 2010. doi:10.5244/C.24.12.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] S. Li and A. B. Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *Computer Vision–ACCV 2014*, pages 332–347. Springer, 2014.

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[20] W. Ouyang, X. Chu, and X. Wang. Multi-source deep learning for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2329–2336, 2014.

[21] L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele. Poselet conditioned pictorial structures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 588–595, 2013.

[22] G. Pons-Moll, D. J. Fleet, and B. Rosenhahn. Posebits for monocular human pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2345–2352. IEEE, 2014.

[23] I. Radwan, A. Dhall, and R. Goecke. Monocular image 3d human pose estimation under self-occlusion. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1888–1895. IEEE, 2013.

[24] V. Ramakrishna, T. Kanade, and Y. Sheikh. Reconstructing 3d human pose from 2d image landmarks. In *Computer Vision–ECCV 2012*, pages 573–586. Springer, 2012.

[25] V. Ramakrishna, D. Munoz, M. Hebert, J. A. Bagnell, and Y. Sheikh. Pose machines: Articulated pose estimation via inference machines. In *Computer Vision–ECCV 2014*, pages 33–47. Springer, 2014.

[26] B. Sapp and B. Taskar. Modec: Multimodal decomposable models for human pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3674–3681. IEEE, 2013.

[27] L. Sigal, A. O. Balan, and M. J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International journal of computer vision*, 87(1-2):4–27, 2010.

[28] E. Simo-Serra, A. Ramisa, G. Alenyà, C. Torras, and F. Moreno-Noguer. Single image 3d human pose estimation from noisy observations. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2673–2680. IEEE, 2012.

[29] J. Sung, C. Ponce, B. Selman, and A. Saxena. Unstructured human activity detection from rgbd images. In *Robotics*

and Automation (ICRA), 2012 IEEE International Conference on, pages 842–849. IEEE, 2012.

[30] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1653–1660. IEEE, 2014.

[31] C. Wang, Y. Wang, Z. Lin, A. L. Yuille, and W. Gao. Robust estimation of 3d human poses from a single image. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2369–2376. IEEE, 2014.

[32] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1385–1392. IEEE, 2011.

[33] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(12):2878–2890, 2013.