3d Human Pose Estimation from 2d Keypoints

By Richard Ludlow, June 2018

I. Summary Overview

While humans can generally estimate with ease the 3d pose of a human in a 2d image, 3d pose estimation remains a challenging problem for machines. In the context of a broader effort to apply 3d pose estimation to motion capture animation, I seek to improve an algorithm that estimates 3d keypoints of human poses with 2d keypoints as the only input. I pursue three key interventions to improve performance: a) modify the data normalization technique in preprocessing, b) modify the neural network architecture from a simple densely connected network to a multi-stage network modeled after state-of-the-art 2d pose estimation models, and c) generate synthetic data to augment training sets.

II. Domain Background

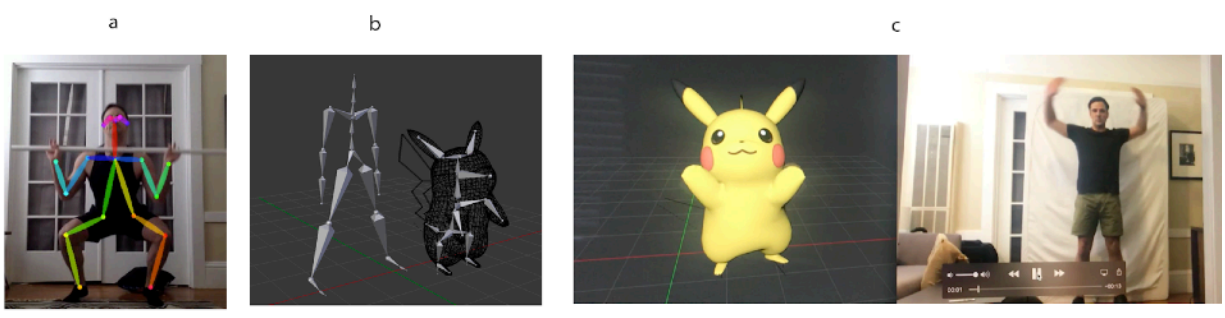
Within the field of computer vision, human pose estimation is the practice of analyzing an image of a human to determine estimates of the absolute and/or relative location of physical parts or joints (e.g. shoulder, elbow) and their combination into physical poses. Over a sequence of images such as a video, pose estimation enables the classification and analysis of actions. Pose estimation is utilized for various purposes in surveillance including gait analysis to track a surveillance target.

Marker-less Motion Capture from Standard RGB Images

My personal interest is in the potential to develop algorithms that will enable marker-less motion capture animation from videos generated from commonly available technology such as smartphone cameras. Professional motion capture setups include equipment such as multiple high-quality cameras, marker suits, and green screens. Researchers have applied pose estimation techniques to achieve quality motion capture without markers on the actors or additional specialized equipment, but requiring views from multiple RGB cameras [1].

Figure 1: Pose estimation for motion capture

a) visualization of 2d joint locations accurately estimated by 'OpenPose' [2][3], b) skeletal rigs in Blender animation software for CMU bvh file (left) and animated character (right), c) prototype implementation of 2d motion capture animation with character movements following those of person filmed.



Using state of the art pose estimation algorithms — specifically, Michael Faber's keras implentation [2] of OpenPose [3] — I have prototyped functioning 2d motion capture animation, in which 2d keypoint estimations drive the movement of an animated character primarily on a 2d plane. The work below is done in the context of an effort to improve 3d pose estimation from frames of a video of a single person filmed on a standard smartphone camera.

Current State-of-the-Art 2d Pose Estimation

In recent years, researchers have utilized convolutional neural networks to achieve substantial advances in the accuracy of 2d human pose estimation from standard RGB images [4]. These networks demonstrate high accuracy in estimating the x,y pixel coordinates of each joint in a 2d image, often even including occluded joints (those not directly visible in the image, such as a hand behind a back). Large training sets are available from several sources with annotated joint locations of humans in a wide variety of settings [5][6].

Challenges in Three Dimensions

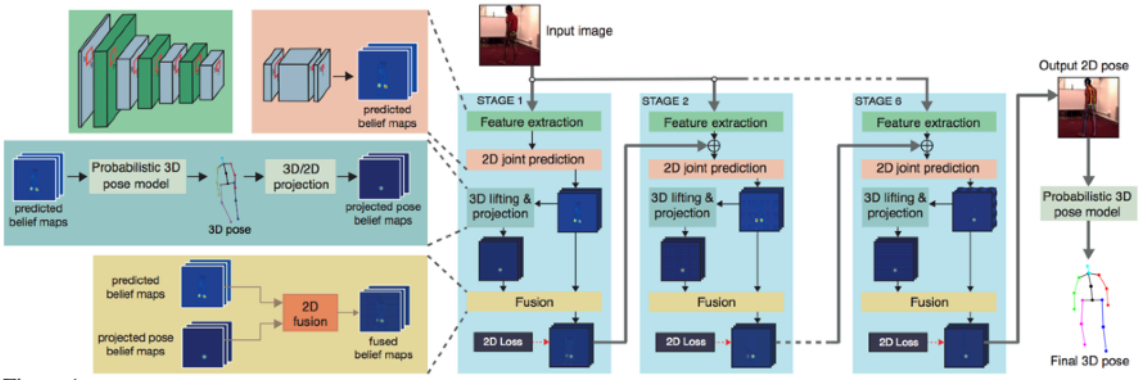
Attempting to estimate all three dimensions of a pose from a 2d image introduces significant additional challenges. First, 2d pose estimation is (primarily) a task of locating multiple objects in an image, whereas estimating depth requires additional inference. Second, the single added dimension doubles the degrees of freedom for object position and orientation to four in 3d from two in 2d. Third, it is very resource-intensive to produce training sets with accurately annotated 3d points, so there is less 3d data for researchers to work with; and even as additional training data becomes commonly available, it is difficult to produce complementary training data to meet a specific need.

Adding additional contextual information such as camera focal length, position, and rotation relative to filmed subjects [7] simplifies the challenge by limiting the range of possible 3d poses for any 2d pose, but such calibration information is often not available. Fitting estimations to a kinematic skeleton with fixed bone lengths [8] constrains the problem, but can limit a model's applicability to other contexts.

Pose "Lifting"

In several recent approaches to 3d pose estimation including Tome 2017 [9], a key included element has been a model trained to estimate 3d poses with only 2d coordinates (or 2d heat maps in some cases) as inputs, providing a first-order approximation that then can be refined further in a model. Figure 2 is Tome's visualization of his 3d pose estimation model. Therein, the "3D lifting and projection" step estimates a 3d pose from 2d coordinates repeatedly in each of six stages and again at the final step. The diagram also visualizes the multi-stage approach used by most state-of-the-art pose estimation models, and which I incorporate in my interventions below.

Figure 2: Diagram from Tome outlining multi-stage model for 3d pose estimation, including "3D lifting & projection" steps



The greater the reliability of pose "lifters" in such a framework, the lesser the overall difficulty 3d estimation presents. My project focuses on improving a 2d keypoint to 3d pose estimation algorithm that could serve this purpose.

III. Problem Statement

When testing images and/or raw 2d points on Tome and Zhao 2017, I found that despite generating quality 3d pose reconstruction on many poses, the models produced significant reconstruction errors often enough that they could not reliably be used for a task like motion capture animation. Notably, I observed that the models would show significantly higher error for certain categories of poses in a sequence, e.g. bending over to touch the ground, or leaning far in towards the camera when dancing.

My objective is to improve the accuracy of 3d pose reconstruction from 2d points relative to a benchmark of Zhao. As described below, the focus will be on reducing overall reconstruction error, as well as error on specific subsets of poses that are poorly estimated in the benchmark model.

Metrics

Consistent with Zhao (and two prior papers they benchmarks against), my evaluation metric is the accuracy of reconstruction as measured by the average Procrustes distance between ground truth and predicted 3d points in a test set. In Procrustes analysis, sets of points are normalized such that they are centered around the origin and the root mean squared distance from the points to the translated origin is one. (Then if relevant, additional transformations are applied to adjust for rotational or reflection difference between two sets of points being compared, though this component doesn't have much relevance in the work below.) The Procrustes distance for a given set of points is then the Euclidean distance between the ground truth and predicted points. Overall, the key benefit of this method is that it makes our analysis invariant to scale, even if different normalization procedures have been applied. In this paper, the terms Procrustes distance, Procrustes error will be used interchangeably, and referenced as simply 'reconstruction error' or 'error' in context.

My key success metrics are:

1. The average Procrustes error across the testing sets in aggregate, consisting of data from CMU Subjects 13, 14, and 15 (described below).
2. The average Procrustes error of a specific subset of poses I will select based on having extremely high reconstruction error in the benchmark model.

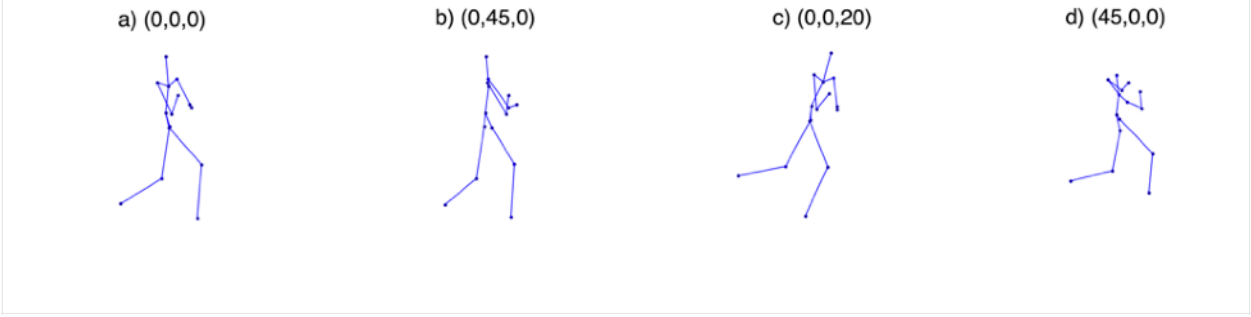
My objective is to apply interventions that result in Pareto improvement in metrics 1 and 2 over the benchmark model. Improvement solely in metric 1 would be unsatisfying as it wouldn't address the key challenge of very-high error poses; improvement solely in metric 2 would be unsatisfying as it would mean that the decreased error in the focus poses was offset by increased error in other poses.

IV. Benchmark Algorithm

In "A Simple, Fast, and Highly Effective Algorithm to Recover 3D Shape from 2D Landmarks on a Single Image," Zhao [10] reports state-of-the-art reconstruction accuracy using a simple network trained on synthetically augmented datasets. Zhao's strategy of synthetically augmenting datasets is particularly appealing given the aforementioned challenges associated with obtaining 3d data training sets.

With known 3d keypoints for an object, the 2d image points for that object viewed from any angle can be calculated through matrix transformations. Zhao augments a given training set of poses (or more generally, 3d shape keypoints) and their associated 2d key points with hundreds to thousands of synthetically generated 2d keypoints from different rotations of that pose.

Figure 3: Multiple rotations of a single pose
Tuple represents rotation of a "camera" about the x, y, and z axis.



Note that Zhao assumes weak-perspective projection, which in our case effectively assumes that the distance (in the z coordinates) between the camera and a subject is sufficiently greater than the potential differences and changes in the depth of the actor's joints in the scene, such that the impact of camera perspective projection can be ignored. Given this, the ground truth x-y coordinates of an object remain in proportion to the 2d x-y coordinates of the object in an image, and training and testing can be done without converting measured world coordinates into camera coordinates and then image pixel coordinates.

Zhao utilizes this augmented data to train a very simple six-layer fully densely-connected neural network, which accepts flattened array of x,y coordinates as inputs and outputs an estimate of z coordinates. In the neural network, between the input array and output array are five “hidden layers” with 30 nodes each. These nodes are linked by connections that are assigned a numerical weight. The nodes each have a value determined by a linear combination of the original inputs or nodes in the hidden layers – based on the weights of the connections – that is fed into a function (hyperbolic tangent in our case) to generate a value between 0 and 1.

During the training process, the model accepts the input arrays and tries values for the weights to generate the output array of predicted z coordinates. The extent that predicted values differ from the actual is the error of that prediction (our error function, Euclidean distance, is discussed below). This error is then “backpropagated” through the network until each node has an error value that reflects its contribution to the error of the output. Then, via batch gradient descent, the weights of each connection are modified incrementally in such a way that minimizes the error of the output. Using a very large sample allows this process to be repeated over and over many times to continually adjust the model to generate improved expected predictions. We save the weights of the connections in the model that generate the lowest error in a portion of our training data set aside from training to be used as validation data. Those weights are then used to generate predicted values in the testing sets below.

Code Segment 1: Network Design in Zhao & Benchmark Model (Keras)

```
input1 = Input(shape=(30,))
x = Dense(30, activation="tanh")(input1)
x = Dense(30, activation="tanh")(x)
x = Dense(30, activation="tanh")(x)
x = Dense(30, activation="tanh")(x)
x = Dense(30, activation="tanh")(x)
x = Dense(15, activation=“tanh")(x) #final
```

One benefit of the simple network architecture shown above that Zhao highlights is speed. Many applications of pose estimation depend on real-time performance, requiring processing of at least 10 frames per second, and thus require a repeated subprocess such as a pose lifting model to run significantly faster than this.

Zhao reports state-of-the-art 3d reconstruction accuracy from 2d key points of cars, flags flapping in the wind, human faces, and human poses.

Limitations

- Weak-perspective projection will not be valid assumption for all use cases. If the camera is relatively close to subjects, then the impact of perspective projection should be evaluated before using this model.

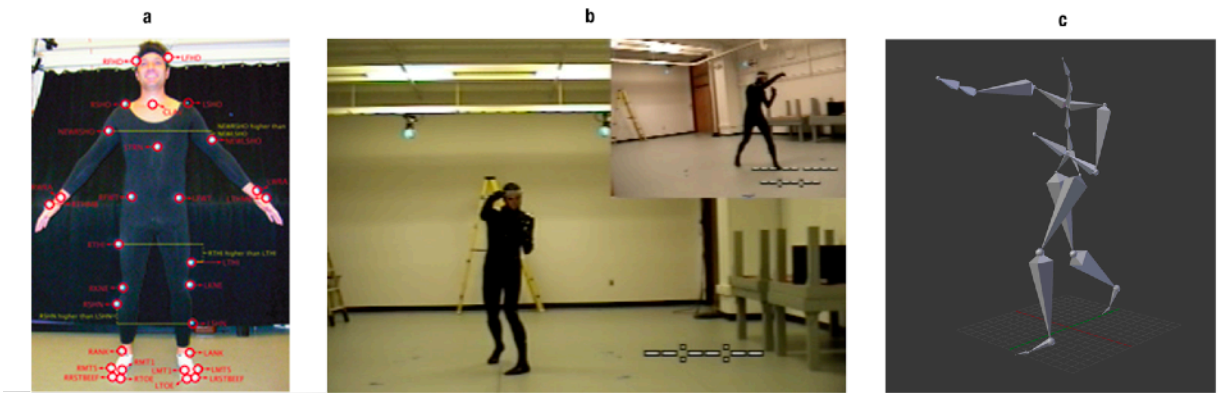
- This model is trained and tested using nearly perfect data; there are no missing joints, mis-measured coordinates, or even variability in the shape of the actors. It will need to be adjusted or integrated into a larger model that is trained appropriately with real-world data.
- In order to compare the model that of prior researchers, only one subject from the data source below is used for training. Utilizing more of the dozens available would greatly increase the range of poses covered in the training set.

V. Data Source

The dataset used in this project is the Carnegie Mellon University Graphics Lab Motion Capture Database [11]. This database includes 3d positions of human joints from hundreds of action sequences recorded from dozens of human subjects filmed while wearing suits with markers on joints. It is the largest publicly available set of annotated 3d joint positions and is frequently utilized by researchers working on tasks related to pose estimation and human movement. Like Zhao, Zhou 2015 [13], and Ramakrishna 2012 [14] I use subject 86's action sequences for training, and subjects 13, 14, and 15 for testing.

Figure 4: CMU Motion Capture

a) CMU subject with marker motion capture suit, b) CMU recording actions from multiple camera angles, c) visualizing action from (b) in Blender.



I access the data via .bvh files of CMU's data released via cgspeed.com [12]. A bvh file includes specifications for the dimensions of an animation skeletal rig, and defines frame-by-frame actions of that rig via translations of a root bone and rotations of all other bones. I import the bvh files into the open-source animation software Blender, which allows for clear visualization of the data, 3d and 2d coordinate measurement, and the ability to easily augment the dataset by creating additional poses and animations.

Each CMU subject was recorded for between 15 and 42 distinct sequences of various actions, each labelled with a very simple description – samples of which can be seen in Table X below – of a wide variety of activities. Deeper exploration of the data is done in the context of the reproduction error of poses.

Table 1: CMU Motion Capture Data Sets

Subject	# Sequences	Frames (Poses)	Sample sequence descriptions
86 (training)	15	115,870	walk, squats, run, stretch, jumps, punches, and drinking; walking, dragging, sweeping, dustpan, wipe window, and wipe mirror; bouncing basketball
13 (testing)	42	108,090	sit on high stool, stand up; forward jump; wash windows; climb ladder; boxing; laugh; sit on stepstool, hands against chin, fidget, stand up
14 (testing)	37	109,160	boxing; unscrew bottlecap, drink soda; mop floor; direct traffic, wave, point; sit on stepstool, stand up, swing legs
15 (testing)	14	145,730	walk/wander; wash windows, paint; hand signals; dance - Egyptian walk, the Dive, the Twist, the Cabbage Patch; boxing; lean forward, tiptoe

VI. Data Exploration and Visualization

While the code for Zhao's model is not publicly available and not every experimental detail is outlined in the paper, the model is explained in enough detail for one to produce a close reproduction (methodology outlined below in section VII). As seen in Table 2, my reproduction generates reconstruction errors that are very close to those published in Zhao.

Table 2: Comparison of reconstruction error in benchmark reproduction to Zhao and earlier papers

Methods	Subject 13	Subject 14	Subject 15	Weighted Average *% Diff from Zhao*
Zhao	0.0229	0.0201	0.0099	0.0168
Benchmark Reproduction	0.0220	0.0203	0.0123	0.0176 *+4.8%*
Zhou	0.0653	0.0643	0.0405	0.0550 *+227.4%*
Ramakrishna	0.0983	0.0979	0.0675	0.0858 *+410.7%*

The objective of this project is not explicitly to beat Zhao's model, but rather to improve the best available model that I have access to. Thus, I use my reconstruction of Zhao as the benchmark model against which I evaluate the interventions outlined below.

Distribution of Reconstruction Error

The median reconstruction error is at 0.010, but with a mean of .0176 the data is heavily right-skewed, and close to a log-normal distribution. Approximately 90% of poses have reconstruction errors less than one standard deviation (.023) above the mean, though the error level climbs quickly above the 95th percentile, telling us that a small proportion of poses have outsized errors.

Figure 5: Density plot of Procrustes error in predicted poses

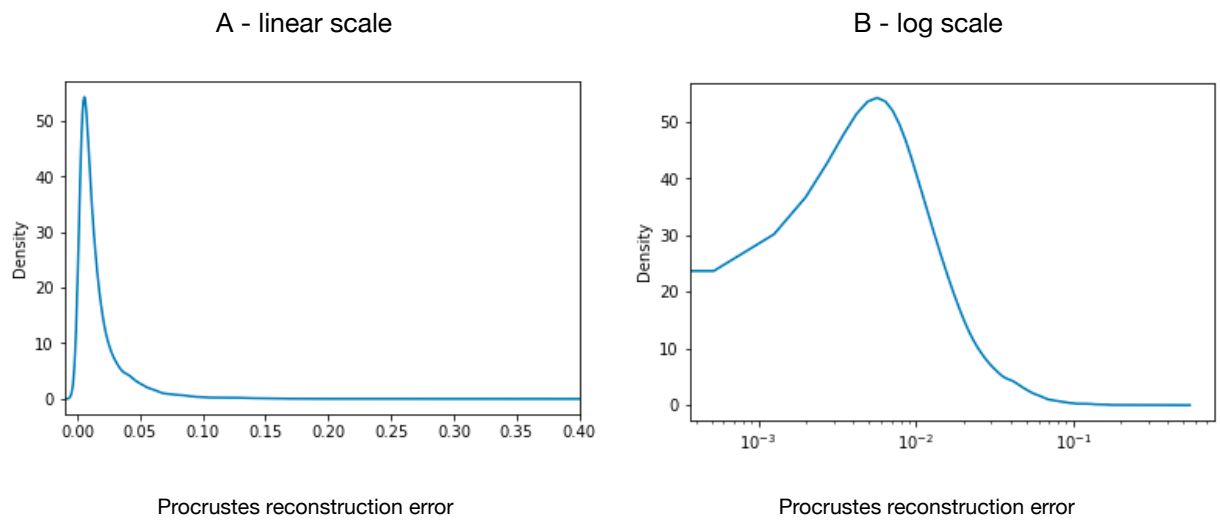


Table 3: Distribution of Procrustes error in predicted poses

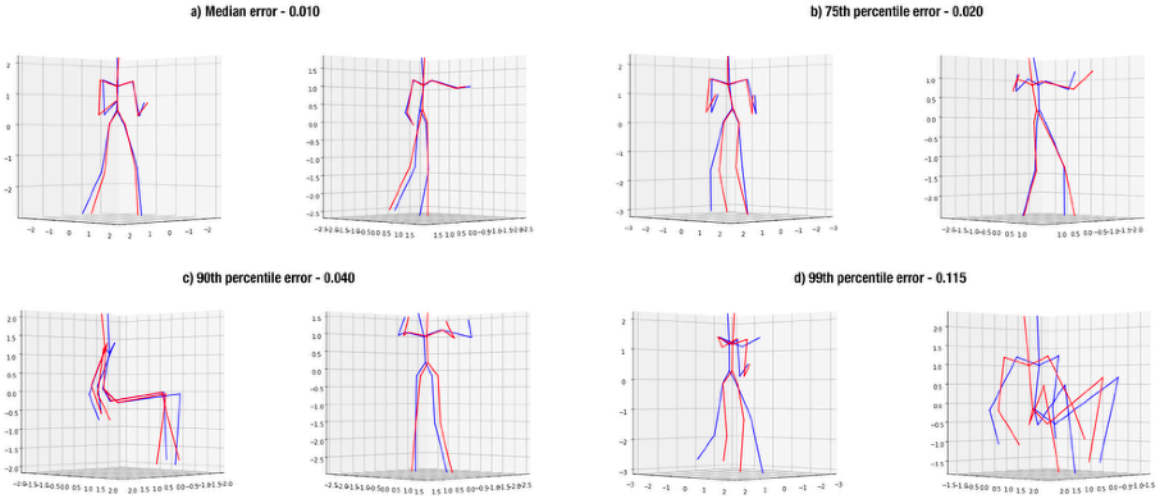
mean	std	min	25%	50%	75%	90%	95%	99%	99.5%	max
0.018	0.023	0.000	0.005	0.010	0.020	0.041	0.058	0.116	0.144	0.367

Visualization of Pose Reconstruction at Various Error Levels

Below we visualize poses at different levels of reconstruction error to help give meaning to the numbers, and to provide a sense of the error thresholds that might be acceptable for a given use case, such as motion capture animation. Pose reconstructions near the median Procrustes error of 0.010 are near-perfect, and those at the 75th percentile (0.020) are also very close to ground truth. At the 90th percentile (0.041), the errors become more visible, but the general pose and body proportions remain intact. By the 99th percentile (0.115), the errors are substantial enough to significantly distort the pose.

Figure 7: Pose reconstruction at various error levels

Ground-truth pose in blue; estimated reconstruction in red



Highest Error Poses and Focus Type

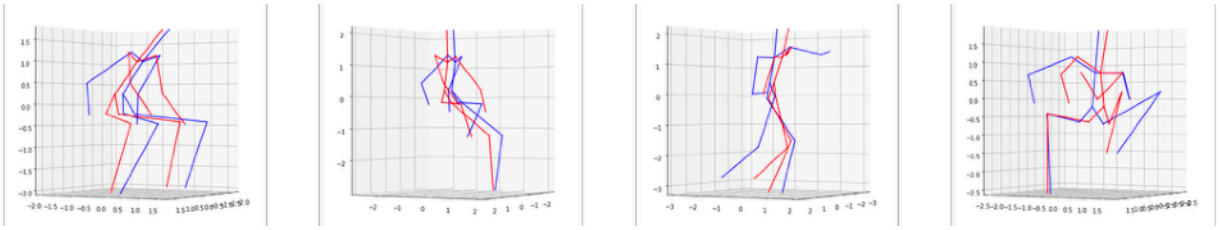
As mentioned earlier, a challenge I've encountered with 3d pose reconstruction models is that they function well for many poses but then have significant errors for certain types of poses. In order to determine if there were a class of poses from the testing set that generated outsized errors, and to ensure those poses were taken from multiple subject across various action sequences, I filtered for the highest error pose in each subject-action animation, then selected the four highest error poses for each of the three subjects.

Figure 8: High-error poses

Seated poses are starred

Subject 13

0.253* 0.202* 0.163 0.182*

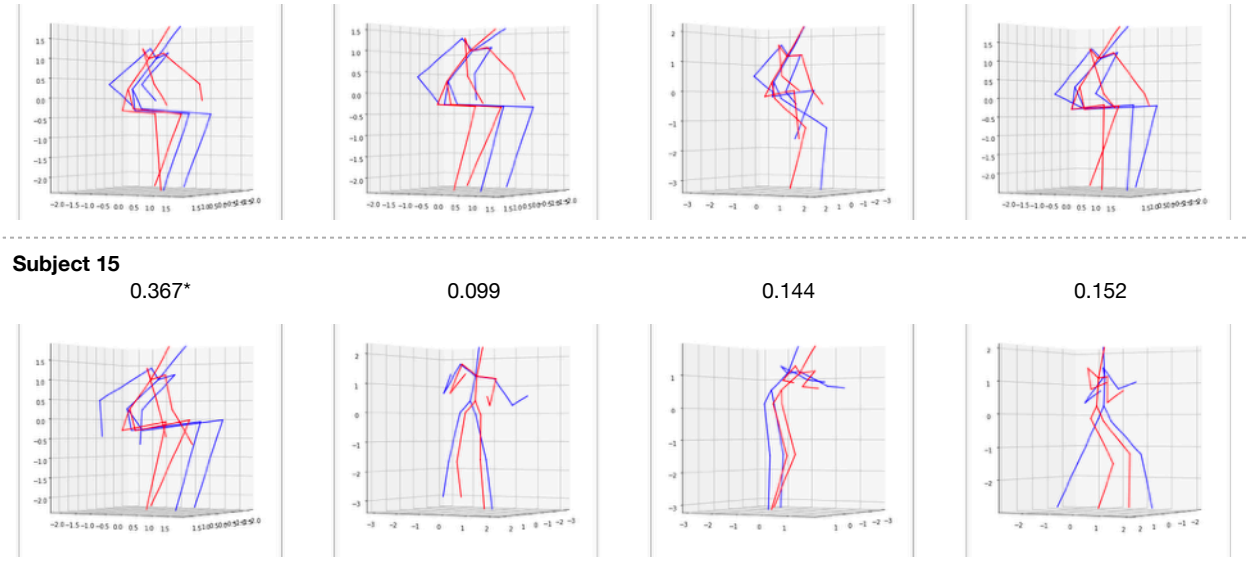


Subject 14

0.310* 0.250* 0.198* 0.310*

Figure 8: High-error poses

Seated poses are starred



Of the twelve high-error poses, eight were of a subject in a sitting motion or pose. Thus, our efforts will focus on reducing reconstruction error of seated poses. The set of seated poses we will measure will consist of the eight seated poses selected from the above, plus, for each of those, the two poses before and after it in the data set. These added poses are very similar to the originally selected poses, but different enough to decrease the impact of any idiosyncrasies from the eight original poses. This results in a pool of 40 seated poses and (after extending the pool of non-seated poses in the same way) 20 non-seated high-error poses.

VII. Training and Testing Methodology

PREPROCESSING

After downloading the bvh files, I edited training and testing files to include only every 10th frame; since keypoints are provided at a rate of 120 frames/second, adjacent frames contain nearly identical poses so increase file sizes without providing additional useful data. I edited each bvh file so that joint movements are applied to a consistent, symmetric skeleton (matching the dimensions of Subject 86, with any bilateral asymmetry removed). This decision assures that idiosyncrasies in how the skeleton of each subject was measured don't skew the results[1].

Bvh files were imported into Blender, in which we saved frame-by-frame 3d positions for 15 joints: head, neck, shoulders (2), elbows (2), wrists (2), hips / base of spine, left and right hip joint (2), knees (2), ankles (2). CMU's data includes coordinates for additional parts such as feet and different spine segments, but the 15 selected are consistent with Zhao and generally close to what I see used in pose estimation models.

[1] The skeletal rigs from different subjects specified in the bvh files have significant asymmetries. I'm not sure whether or not the file format used by Zhao contained similar asymmetries, but it is after making this adjustment that the reconstruction errors of Zhao and my benchmark reproduction of Zhao converge.

For training data only:

Create 720 augmented poses for each in the training set, based on rotations uniformly distributed of within the range [-20◦,20◦] about the x-axis, [-20◦,20◦] about the z-axis, and [0◦,720◦] around the y-axis.

Before training or testing, the data is normalized according to the procedures that are discussed in more section VIII.

NETWORK SETTINGS

Training and testing were conducted using Keras on Tensorflow. Training data was shuffled before and during training, and 20% of the data was apportioned for validation. To help prevent overfitting, early stopping is implemented if the validation error does not improve for 10 consecutive epochs (with this, none every reached the set maximum of 200 epochs).

Like Zhao, the loss function used was the Euclidean distance between the z (depth) coordinate of ground truth and predicted points:

$$\sqrt{\sum_{i=1}^{n}(ztrue_i - zpred_i)^2}$$

The network uses the Hyperbolic Tangent activation function, tanh, and is optimized with RMSProp. Zhao uses a batch size equal to the number of samples in the original (pre-augmented) dataset, although it wasn't clear precisely how many samples they used. Thus, I tried various batch sizes and settled on 1024 (note that each sample is a 16x3 array). As noted in the challenges discussed in Section IX, when batch sizes were too large, the models were converging too early and the interventions had little to no impact. At 1024, premature convergence was no longer a problem, while reducing batch size further increased training time without improving validation and testing performance.

In an effort to reduce reconstruction error on the seated poses and the training set as a whole, I pursued a multi-step strategy. First, I modified the methodology used to normalize the dataset in preprocessing; the revised normalization method significantly reduced both aggregate and seated pose errors. Second, I tried modifications to the network architecture. Simply adding dropout layers to the benchmark model increased reconstruction error; however, a multi-stage network significantly reduced both errors. Third, I synthetically augmented the training set using an animation rig in blender. Adding a sequence with a variety of sitting and squatting poses significantly reduced error on the seated poses. Adding a bilateral mirror image reduced error on seated poses by a lesser amount, but also significantly reduced aggregate error. Combining the original training set with both augmented data sets, applying the improved normalization technique, and utilizing a multi-stage network resulted in significant reduction in aggregate error and dramatic reduction in error on seated poses.

(Note that in discussing these interventions, models will be referenced by a combination of the network design used, normalization procedure used, and augmentation set, if applicable, e.g. *Multi-Sequence-Seated*. Reproduction error will generally be compared to the most successful prior model).

Table 4: Changes in error from modifications to preprocessing methodology and network design
Note: Model A is the benchmark model

			Procrustes Error	
Model	**Network**	**Normalization**	**Testing Data Aggregate**	**Seated Poses**
A	Base	Frame	0.018	0.228
B	Base	Sequence	0.015	0.114
C	Dropout	Frame	0.021	0.127
D	Dropout	Sequence	0.021	0.126
E	Multi	Frame	0.016	0.100
F	Multi	Sequence	0.013	0.087

INTERVENTION 1: Data Re-Normalization

In order to eliminate the effect of scaling and translation, Zhao normalizes the data according to the formulas below, with j representing the number of data points in a pose, and i the number of poses in a dataset. The denominator only includes the standard deviations of x and y since data for z coordinate is not available for the 2d points used as inputs:

$$\hat{x}_{ij} = \frac{x_{ij} - \overline{\mathbf{x}}_i}{(\sigma(\mathbf{x}_i) + \sigma(\mathbf{y}_i))/2}, \qquad \hat{y}_{ij} = \frac{y_{ij} - \overline{\mathbf{y}}_i}{(\sigma(\mathbf{x}_i) + \sigma(\mathbf{y}_i))/2}, \qquad \hat{z}_{ij} = \frac{z_{ij} - \overline{\mathbf{z}}_i}{(\sigma(\mathbf{x}_i) + \sigma(\mathbf{y}_i))/2},$$

Applying these formulas brings the mean of each vector to 0, and the standard deviation of all x and y coordinates to near 1; the standard deviation of z coordinates will vary.

Zhao's model is applied across various objects and use cases besides human poses, and I hypothesize that a simple modification will improve it for our use case. Given the constraints that a) we assume weak perspective projection in the scene, b) cameras remain in a fixed position, and c) there is only one actor per sequence, we then don't encounter changes in scale within a given scene. Thus, the only relevant scaling factor is a single number per scene that standardizes scale based on the size of the actor and general scale of the inputs.

When training the model, we don't input any information about skeletal proportions of relationships; rather, the model implicitly learns relationship. Thus, the formulas above are problematic when applied on a frame-by-frame basis as a change in pose will impact both the numerator *and* the denominator of the normalization formula, so normalized data will be out of proportion to the ground truth data. For example, if an actor looking straight at the camera changes from a) standing straight to b) kneeling down, then std(x) will remain

nearly unchanged, while std(y) will decrease. Thus, the denominator (std(x) + std(y)) / 2 will be smaller for b than for a, so for b, the normalized x coordinates will increase despite no change in the ground truth coordinates, and the normalized y coordinates will decrease less than their corresponding ground truth coordinates.

I hypothesized that selecting a single constant to scale poses over an entire scene will eliminate this issue and improve results. I tested the impact of modifying normalization procedures to divide all poses within a sequence by a constant value: the standard deviation of y values of the actor standing straight up with arms out in a T-pose. This was the first pose in each of the CMU animations[2] and is a proxy for the overall size of the actor, and the corresponding y values are labelled as y_0 in the revised normalization equations:

$$\widehat{x}_{ij} = \frac{x_{ij} - \overline{\mathbf{x}}_i}{\sigma(\mathbf{y}_0)} \qquad \widehat{y}_{ij} = \frac{y_{ij} - \overline{\mathbf{y}}_i}{\sigma(\mathbf{y}_0)} \qquad \widehat{z}_{ij} = \frac{z_{ij} - \overline{\mathbf{z}}_i}{\sigma(\mathbf{y}_0)}$$

This intervention, labelled “sequence normalization” below, significantly improved *Base-Sequence’s* accuracy relative to the *Base-Frame*, reducing aggregate reconstruction error to .015 from .018, and error on seated poses to .114 from .228.

INTERVENTION 2: Network Modification

Dropout Layers

As shown above in Code Segment 1, the benchmark model utilizes a simple six-layer fully densely connected network. With no dropout layers or other related interventions, overfitting the training data was a risk. Thus, I tested a model that included dropout layers[3].

Code Segment 2: Simple Dense Network Plus Dropout Layers (Keras)

```
input1 = Input(shape=(30,))
x = Dense(30, activation=“tanh")(input1)
x = Dropout(0.3)(x)
x = Dense(30, activation="tanh")(x)
x = Dense(30, activation=“tanh")(x)
```

---

[2] I chose this constant since it was readily available in the data and represented only a small change from the original formula, if starting again, I would chose a constant that is easier to measure or estimate across various datasets, such as the actor’s standing height.

[3] I tested additional variants for this model, varying the magnitude, number, and placement of dropout layers. I also tested additional variants of the multi-stage network below, such as the number and length of stages, dropout layer parameters, and number of nodes per layer, as well as the optimizer. Variants weren’t pursued further if they didn’t improve outcomes, only those conducted with identical experimental settings are included here.

```
x = Dropout(0.2)(x)
x = Dense(30, activation="tanh")(x)
x = Dropout(0.2)(x)
x = Dense(30, activation="tanh")(x)
x = Dense(15, activation="tanh")(x) #final
```

The addition of dropout layers was not a successful intervention because although seated pose error improved, the aggregate reconstruction error for both *Dropout-Frame* (.021) and *Dropout-Sequence* (.021) exceeded that of the benchmark (.018).

Multi-Stage Model

The next major variant in the model was inspired by the networks utilized in state-of-the-art 2d pose estimation models such as Convolutional Pose Machines and OpenPose, and in the diagram from Tome above. In these, the network includes multiple stages in which the first stage develops a first approximation of the output variables, and then subsequent stages refine this approximation utilizing both the original inputs and the prior stage's output.

Code Segment 3: Multi-Stage Network (Keras)

```
input1 = Input(shape=(30,))

# Stage 1
x = Dense(30, activation="tanh")(input1)
x = Dropout(0.20)(x)
x = Dense(30, activation="tanh")(x)
x = Dense(30, activation="tanh")(x)
x = Dense(30, activation="tanh")(x)
x = Dropout(0.20)(x)
x = Dense(30, activation="tanh")(x)
x = Dense(15, activation="tanh")(x)

# Repeat segment below for Stage 2-5
# Concatenate original input with output vector of prior stage
x = Concatenate()([input1, x])
x = Dense(45, activation="tanh")(x)
x = Dense(45, activation="tanh")(x)
x = Dropout(0.10)(x)
x = Dense(30, activation="tanh")(x)
x = Dense(15, activation="tanh")(x) # in stage 5, is final output
```

This intervention had significant positive effects. Comparing *Multi-Sequence* to *Base-Sequence*, seated pose error improved to 0.087 from 0.114, and overall error decreased to .013 from .015.

INTERVENTION 3: Synthetic Training Set Augmentation

As noted above, limited 3d training data continues to present challenges for researchers in this field, so opportunities to improve models with synthetically generated/augmented data are valuable. The benchmark model utilizes synthetic augmentation to generate rotations of poses. The interventions below involve synthetic generation of *additional* poses (that can then also be augmented via rotation).

These synthetic datasets were concatenated with the original training set, shuffled, then used to train the Model from scratch.This approach – rather than fine-tuning – was used because the new datasets were large (up to equal in size) relative to the original.

Table 5: Changes in error after training data augmentation

				Procrustes Error	
Model	Network	Normalization	Augmentation	Testing Data Aggregate	Seated Poses
F	Multi	Sequence	[None]	0.013	0.087
G	Multi	Sequence	Seated_Short	0.013	0.087
H	Multi	Sequence	Seated_Long	0.013	0.046
I	Multi	Sequence	Bilateral	0.010	0.055
J	Multi	Sequence	Seated_Long & Bilateral	0.011	0.040

Custom Animation - Seated Poses

Among the potential reasons that the model would perform poorly on a class of poses in the testing set is that those poses are underrepresented in the training set, or they are represented but biased in a manner that is inconsistent with the testing set. When watching the animations of the training set data, I see that seated poses are rare, and most often involve sitting at the height of a tall stool; in contrast, several of the problematic poses from the training set involve sitting at lower depths.

In Blender, I created a new rig that shares the same skeletal proportions as those used in the training and testing sets, but has controls (using inverse kinematics) that allow me to pose it for custom animations. I created animations of the rig sitting and squatting at various depths, with variation in the position and angle of other joints throughout the animation.

The objective was to increase the variety of seated positions in the training set and see if this would improve the model's reconstruction accuracy on seated poses. My initial attempts were unsuccessful. For example, a 150 frame animation of various seated poses (*Seated_Short*) had no impact on aggregate or seated pose

error. In some follow-up attempts, the impact on the seated poses would be mixed, sometimes suggesting a bias in the training data, e.g. lack of seated poses with the arms reaching far back. Ultimately, a 950-frame animation, in which I carefully included a wide range of depths, foot positions, arm positions, back and neck rotations, etc. was successful: *Multi-Sequence-Seated_Long* reduced error on seated poses to .046 from .087 relative to *Multi-Sequence*. It had a negligible impact on aggregate error.

Bilateral Augmentation

Whether due to the tendencies of individual actors or the idiosyncrasies of poses captured in a limited training set, the training set could be biased toward more right- or left- sided movements. This can be addressed with bilateral augmentation — generating a mirror image of each pose in the training set [9].

In Blender, I created a rig that moved according to a bilateral mirror image of Subject 86's original action sequences: the rotations of bilaterally paired joints (e.g. legs, arms) were swapped, and the y-axis rotations of unpaired joints (e.g. spine, neck) were inverted.

I first examined the impact of this data combined with the original training set, but without the seated pose augmentation. Compared to *Multi-Sequence,* aggregate reconstruction decreased error significantly to .010 from .013, and error on seated poses reduced to .055 from .087, a smaller change than *Multi-Sequence-Seated_Long*.

Combined Augmentation

Combining the two interventions into *Multi-Sequence-Seated_Long+Bilateral* by concatenating the original data with the bilateral *and* seated augmentations maintained nearly all the reduction in aggregate error from the bilateral augmentation, increasing just .0004 to .01052, but resulted in even further improvement on seated poses, dropping seated reconstruction error to .040.

IX. Discussion

Summary Outcomes

In summary, the following sequence of interventions successfully reduced our key metrics of overall testing set error and seated pose error:

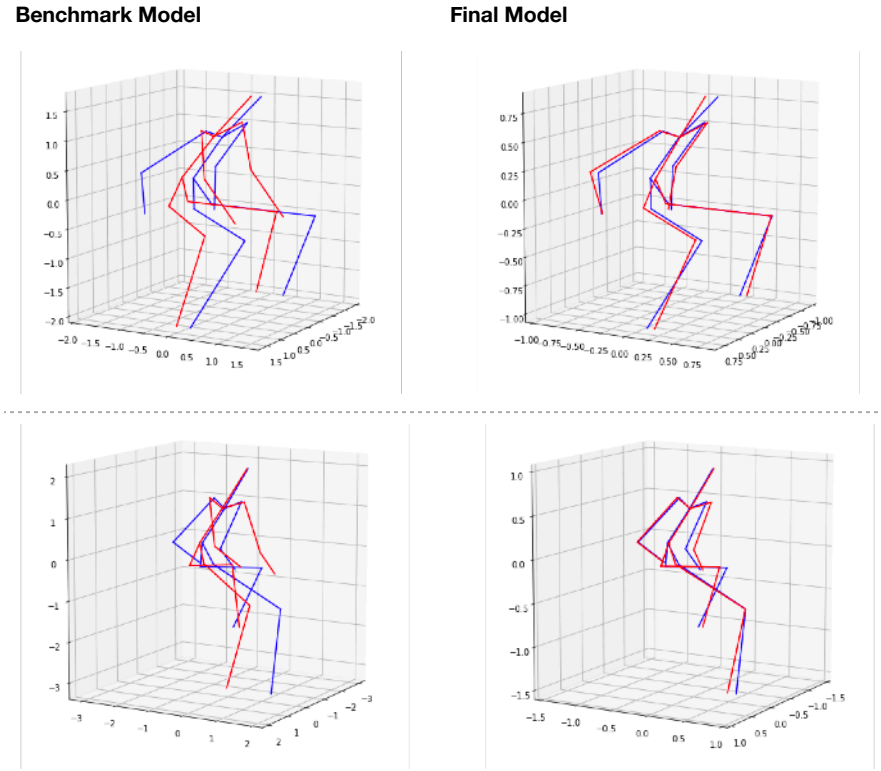
- Normalize data using consistent parameters across an entire recorded sequence rather than varying parameters frame by frame.
- Extend the neural network from a six-layer densely connected network to a five stage network.
- Augment the training set with synthetically generated poses, specifically bilateral mirrored poses, and custom-animated seated poses.

Across the entire testing set, Procrustes error decreased 40%, to .011 in the final augmented model (*Multi-Sequence-Seated+Bilateral)* from .0176 in the benchmark model (*Base-Frame)*.

Among the 40 seated poses, error decreased approximately 8 standard deviations to .040 in the final model from .228 in the benchmark model with 24 of the poses falling below the 0.01 mark for excellent

reconstruction. The 20 non-seated poses improved but to a lesser extent, to .064 in the final model from .092.

Figure 9: Sample high-error seated poses from benchmark and final reconstruction



Robustness Testing

To explore the extent to which the model generalizes belong the standard conditions above, I conducted multiple tests:

First, I selected a pseudo-random set of 16 additional test subjects from CMU's database. Together, these subjects cover a much broader range of activities than the the original three test subjects. As seen in Appendix Table i, the interventions were successful, with an impact close to what was seen on the original three subjects. Our final model above (*Multi-Sequence-Seated_Long+Bilateral)* reduced reconstruction error for 14 of the 16 subjects relative to the benchmark model, for an average decrease of 24%.

Second, I distorted all x,y coordinates in the testing sets by multiplying each coordinate by 1 + a Gaussian with variance of .05[4]. This change emulates the 'messiness' of data that would be inputed in less

[4] Note: a prior draft stated that this distortion was also done on all training data. When I sought to apply the distortion to testing data, I found that due to a coding error the distortion was not being applied to training data. Fortunately, outside of noting that difference, it does not affect the paper's analysis.

standardized environments than that analyzed above. Under these conditions, reconstruction error in each of the models was predictably higher in the benchmark and subsequent models. The interventions related to revising normalization methodology and the revisions to the network design were successful at a magnitude similar to that of the standard conditions, as reconstruction error in Sequence-Multi was approximately 25% lower than that of the benchmark model (Appendix Table ii). Aggregate reconstruction error did not improve further with data augmentation. However, the error on seated poses did improve significantly with the models trained on augmented poses, resulting in seated pose error (0.035) slightly below that of the undistorted test sets (0.040) in the final model.

Third, I tested the model on a set of CMU subjects – the original three testing sets plus the 16 subjects in the first robustness test – but instead of standardizing the skeletal rigs used for each action sequence as done in the standard tests above, I left them unchanged. Thus, the dimensions and positioning of the joints vary based on the size and proportions of the actor used, variation in the placement of markers, and other measurement and conversion error. Many of the bvh files specify skeletal rigs with meaningful asymmetries. The training set was unchanged, consisting of solely the skeletal rig of Subject 86. Under these conditions, the interventions were not successful, reducing aggregate measurement error on just 11 of 19 subjects.

Based on the tests above, the model generalizes well to an expand set of activities as well as moderate distortion to the inputs. However, the model breaks down when tested on skeletal rigs with a range proportions that vary from the training set. Since the model was trained on only one skeletal rig, all of the relationships between joints that the model implicitly learned fit the exact proportions of Subject 86. To address this, the model should be trained on a range of skeletal structures, which could be done by including sequences from different human subjects or synthetically varying the skeletal proportions.

Opportunities

Straightforward opportunities to improve the model further include training the model on a significantly expanded training set starting with multiple CMU subjects, and introducing variation in skeletal structure to make the model more robust.

There is significant additional potential with synthetic animation. Given the positive impact of the bilateral augmentation that was relatively straightforward to implement, I would be interested to pursue additional pose augmentation that can be done programmatically. For example, by setting appropriate physiological limits on the range of motion of limbs and any relationships between them, it may be possible to create something of a random pose generator that could cover a full range of realistic poses.

Pursuing a fully-convolutional approach would significantly alter the nature of the model but could improve performance. The introduction of probability heat maps as outputs – in place of coordinates – is one of the changes associated with the dramatic improvement in performance of 2d pose estimators in recent years. Since such models usually begin with images as inputs, convolutional layers are a natural starting point. But researchers are finding benefits in applying spatial mapping to aspects besides images – for example, the 3d pose estimator VNect has 2d probability heat map arrays for x, y, z estimates – and using convolutional

layers throughout the entire network. In the model above, input and output coordinates could be replaced by Gaussians and estimated using these techniques.

Sources

1. Carsten Stoll, Nils Hasler, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. Fast articulated motion tracking using a sums of Gaussians body model. In IEEE International Conference on Computer Vision (ICCV), 2011.
2. Michal Faber. Keras version of Realtime Multi-Person Pose Estimation ('OpenPose') project. https://github.com/michalfaber/keras_Realtime_Multi-Person_Pose_Estimation
3. Zhe Cao, Tomas Simon, Shih-En Wei and Yaser Sheikh. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields (OpenPose). In CVPR, 2017.
4. Shih-En Wei, Varun Ramakrishna, Takeo Kanade, vand Yaser Sheikh. Convolutional Pose Machines. In CVPR, 2016.
5. COCO Dataset. http://cocodataset.org/
6. MPII Human Pose Dataset. http://human-pose.mpi-inf.mpg.de/
7. Julieta Martinez, Rayat Hossain, Javier Romero, and James J. Little. A simple yet effective baseline for 3d human pose estimation. In ICCV, 2017.
8. Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera. In ACM Transactions on Graphics, July 2017.
9. Denis Tome, Chris Russell, and Lourdes Agapito. Lifting From the Deep: Convolutional 3D Pose Estimation From a Single Image. In CVPR, July 2017.
10. Ruiqi Zhao, Yan Wang, and Aleix M Martines. A Simple, Fast, and Highly-Accurate Algorithm to Recover 3D Shape from 2D Landmarks on a Single Image. In IEEE Transactions on Pattern Analysis and Machine Intelligence, November 2017.
11. CMU Graphics Lab Motion Capture Database. http://mocap.cs.cmu.edu/
12. The Motionbuilder-friendly BVH Conversion Release of CMU's Motion Capture Database. https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture/cmu-bvh-conversion
13. X. Zhou, S. Leonardos, X. Hu, and K. Daniilidis, "3d shape estimation from 2d landmarks: A convex relaxation approach," in CVPR, 2015.
14. V. Ramakrishna, T. Kanade, and Y. Sheikh, "Reconstructing 3d human pose from 2d image landmarks," in ECCV. Springer, 2012.

Appendix

Table i: Procrustes Error of Additional Testing Sets from Selected Models

Subject Number	Base-Frame (Benchmark)	Base-Sequence	Sequence-Multi	Sequence-Multi-Seated_Long	Sequence-Multi-Bilateral	Sequence-Multi-Seated_Long+ Bilateral
2	0.026	0.026	0.017	0.021	0.015	0.017
6	0.018	0.017	0.014	0.015	0.011	0.012
9	0.029	0.021	0.020	0.018	0.019	0.019
30	0.020	0.016	0.015	0.014	0.012	0.015
33	0.013	0.011	0.010	0.009	0.009	0.009
34	0.012	0.013	0.010	0.011	0.007	0.008
49	0.040	0.039	0.031	0.029	0.028	0.029
54	0.019	0.015	0.013	0.015	0.012	0.013
56	0.008	0.007	0.006	0.006	0.004	0.005
60	0.017	0.018	0.016	0.018	0.014	0.015
88	0.067	0.062	0.053	0.056	0.052	0.053
113	0.095	0.078	0.072	0.082	0.076	0.070
115	0.012	0.012	0.010	0.009	0.007	0.007
140	0.057	0.065	0.070	0.053	0.058	0.058
141	0.025	0.021	0.024	0.027	0.022	0.029
143	0.015	0.015	0.013	0.013	0.010	0.013
# Improved vs Benchmark	(n = 16)	10	15	14	15	14
Mean Change		-7%	-18%	-17%	-30%	-24%
Mean Value	0.031	0.028	0.026	0.026	0.023	0.024
Median Value	0.019	0.018	0.016	0.016	0.013	0.015

Table ii: Procrustes Error of Distorted Test Sets from Selected Models

		Base-Frame (Benchmark)	Base-Sequence	Sequence-Multi	Sequence-Multi-Seated_Long	Sequence-Multi-Bilateral	Sequence-Multi-Seated_Long+ Bilateral

Standard	Aggregate	0.018	0.015	0.013	0.013	0.010	0.011
	Seated	0.228	0.114	0.087	0.046	0.055	0.040
Distorted	Aggregate	0.026	0.024	0.019	0.019	0.019	0.020
	Seated	0.209	0.108	0.095	0.047	0.060	0.035

Table iii: Procrustes Error of Test Sets with Varied Skeletal Dimensions from Selected Models

	Base-Frame (Benchmark)	Base-Sequence	Sequence-Multi	Sequence-Multi-Seated_Long	Sequence-Multi-Bilateral	Sequence-Multi-Seated_Long+ Bilateral
13	0.029	0.026	0.021	0.019	0.019	0.020
14	0.024	0.021	0.019	0.019	0.021	0.021
15	0.016	0.026	0.018	0.017	0.029	0.032
2	0.031	0.030	0.025	0.029	0.031	0.033
6	0.033	0.028	0.024	0.027	0.029	0.027
9	0.042	0.027	0.025	0.022	0.026	0.029
30	0.039	0.035	0.031	0.029	0.033	0.041
33	0.015	0.012	0.012	0.011	0.018	0.017
34	0.036	0.048	0.035	0.035	0.030	0.025
49	0.058	0.049	0.044	0.045	0.048	0.050
54	0.044	0.054	0.043	0.049	0.058	0.058
56	0.021	0.022	0.018	0.015	0.021	0.020
60	0.026	0.028	0.024	0.022	0.036	0.040
88	0.068	0.087	0.074	0.070	0.094	0.095
113	0.101	0.088	0.082	0.091	0.105	0.091
115	0.018	0.021	0.015	0.009	0.025	0.019
140	0.056	0.057	0.075	0.049	0.063	0.078
141	0.072	0.078	0.065	0.077	0.096	0.073
143	0.025	0.050	0.045	0.052	0.066	0.067
# Improve vs Benchmark	(n = 19)	10	4	5	11	11

Mean Change		8.53%	-7.21%	-9.06%	17.22%	17.06%
Mean Value	0.040	0.041	0.037	0.036	0.045	0.044
Median Value	0.033	0.030	0.025	0.029	0.031	0.033