Human Pose Recognition via Occlusion-Preserving Abstract Images

Saad Manzur[1] and Wayne Hayes[1]

University of California Irvine, Irvine CA 92617, USA
smanzur@uci.edu, whayes@uci.edu

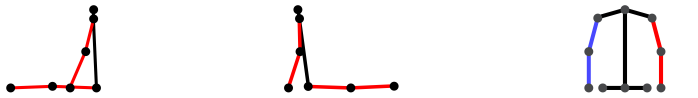
Abstract. Existing 2D-to-3D pose lifting networks suffer from poor performance in cross-dataset benchmarks. Although 2D keypoints joined by "stick-figure" limbs is the dominant trend, stick-figures do not preserve occlusion information that is inherent in an image, resulting in significant ambiguities that are ruled out when occlusion information is present. In addition, datasets with ground truth 3D poses are much harder to obtain in contrast to similar human annotated 2D datasets. To address these issues, we propose to replace stick figures with *abstract images*—figures with opaque limbs that preserve occlusion information while implicitly encoding joint locations. We then break down the pose estimation task into two stages: (1) Generating an abstract image from a real image, and (2) garnering the pose from the abstract image. Crucially, given the GT 3D keypoints for a particular pose, we can *synthesize* an arbitrary number of abstract images of the same pose as seen from arbitrary cameras, even without a part map. Given a *set* of 3D GT keypoints, this allows training of Stage (2) on an unlimited dataset without over-training, which in turn allows us to correctly interpret poses from arbitrary viewpoints not included in the original dataset. Additionally, our unlimited training of Stage 2 allows good generalizations across datasets, demonstrated through a significant improvement in cross-dataset benchmarks, while still showing competitive performance in same-dataset benchmark.

Keywords: 3D Human Pose Estimation · Cross-Dataset Generalization · Viewpoint Encoding

1 Introduction

Recent work on 3D human pose estimation from still images can be classified into two main groups: direct-from-image methods [17, 18, 26, 27, 32, 33, 47], and methods that use intermediate 2D/2.5D keypoints [4, 5, 7, 10, 11, 16, 20, 42, 45, 50]. Keypointing has resulted in a number of methods that "lift" the 3D pose from the intermediate 2D keypoints and their associated "stick-figure" limbs (cf. Fig. 1).

Unfortunately, stick figures are "see-through"—they do not preserve occlusion information that is clearly visible in real images [26, 46]. This introduces myriad problems, for example failing to correctly distinguish left-from-right viewpoints when limbs from opposite sides overlap (Figs. 1a and 1b). Current methods also



(a) Left view (sitting) **(b)** Left view (yoga) **(c)** Front view

Fig. 1: Poses that are indistinguishable without occlusion. Crucially, this implies that the "stick-figure"-to-pose mapping is, mathematically, not a 1-to-1 function.

perform poorly across datasets [10, 39] because they depend on both camera viewing angle, and z-score normalization [16,23,30,32,36,37,45,50], which likely changes the prior between datasets. Solving these problems requires us to (1) estimate the viewpoint accurately, (2) avoid discarding occlusion information, and (3) avoid camera-dependent metrics derived from the training set.

To simultaneously solve all of these problems, we propose training on synthetic "abstract" images of real human poses viewed from a virtually unlimited number of viewpoints. We address the viewpoint bias (Fig. 1) with a novel encoding that creates a mathematical 1-to-1 mapping between the camera viewpoint and the input image; a similar 1-to-1 encoding defines the pose. Both encodings support fully-convolutional training. As depicted in Fig. 2, our method uses the abstract image as input to two networks: one for viewpoint, and another for pose. At inference time, we (1) generate the abstract image from a real image, and (2) take the predicted viewpoint and pose from the abstract image to reconstruct the 3D pose. Since reconstruction does not ensure the correct forward facing direction of the subject, the ground-truth target pose will be related to the reconstructed pose by a simple rotation to compare with other methods.

A key observation is that the camera viewpoint as seen from the subject, and the subject's observed pose as seen from the camera, are *independent*: although they are intimately tied together in the sense that both are needed to fully reconstruct an abstract image —and thus a pose— they answer completely separate questions. More explicitly: (1) the location of the camera as viewed from the subject is completely independent of the subject's pose; and (2) the pose of the subject is completely independent of where the camera is located.

The "abstract-to-pose" part of our method (Stage 2) decomposes 3D human pose recognition into the above two orthogonal components: (1) the camera's location in subject-centered coordinates, and (2) the observed pose of the subject in camera coordinates. Note that identical three-dimensional poses from different viewpoints will change *both* answers, but combining the answers should always allow us to reconstruct the same subject-centered pose.

This, then, is our "secret sauce": by incorporating occlusion information, we can independently train two fully convolutional systems: one that learns a 1-to-1 mapping between images and the subject-centered camera viewpoint, and another that learns a 1-to-1 mapping between images and camera-centered pose. The final ingredient is to train these two CNN's using a virtually unlimited set of abstract images, with occlusion, generated from randomly chosen camera

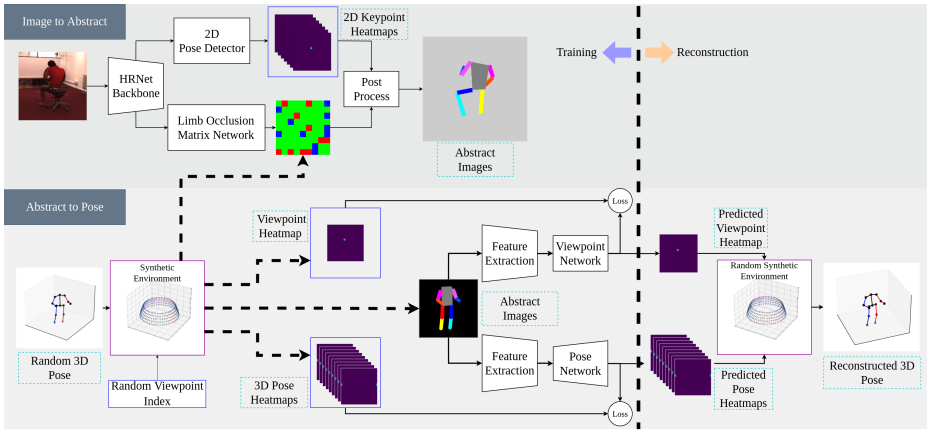


Fig. 2: Overview Dashed arrows indicate our custom supervision targets: limb occlusion matrix, viewpoint heatmap, 3D pose heatmap, and abstract images. **Stage 1** has two phases: first, the "image-to-abstract" network optimizes a binary cross-entropy loss for the limb occlusion matrix and MSE loss for the 2D keypoint heatmaps; the second draws the abstract image with limb occlusion matrix as a z-buffer and 2D keypoints as the core structure. **Stage 2**: generate abstract images (both flat and cube variants) from a random viewpoint and 3D pose pairs using the synthetic environment. The viewpoint and pose heatmaps—generated by the synthetic environment—are used as supervision targets for the abstract to viewpoint and pose networks. The "abstract to pose and viewpoint" network optimize the L2 loss on the output of viewpoint and pose network. The **Reconstruction** stage takes viewpoint and pose heatmaps from Stage 2, using a random synthetic environment to reconstruct the 3D pose.

viewpoints observing the ground-truth 3D joint locations of real humans in real poses. Given a sufficiently large (synthetic) dataset of abstract images, we are able to independently train two CNNs that reliably encode the two 1-to-1 mappings. After the above CNNs are trained, the last ingredient we need for a true image-to-pose is a method to create an abstract image from a real input image. We outline our "image-to-abstract" method in Secs. 3.2 and 4 which shows competitive performance indicating adaptibility in real-world scenario.

The key contributions of our paper are:

1. We replace "stick figures" with *abstract images* as the intermediate representation between images and poses. We represent limbs as opaque, solid, rectangular blocks that preserve occlusion and part-mapping. Using 2D/3D GT keypoints, we can generate *synthetic* abstract images from an unlimited number of camera viewpoints.
2. Novel viewpoint and pose encoding schemes, which facilitate learning a 1-to-1 mapping with input while preserving a spherical prior; and
3. We significantly improve state-of-the-art performance in cross-dataset benchmark without relying on dataset dependent normalization, and only marginal effects on same-dataset performance.

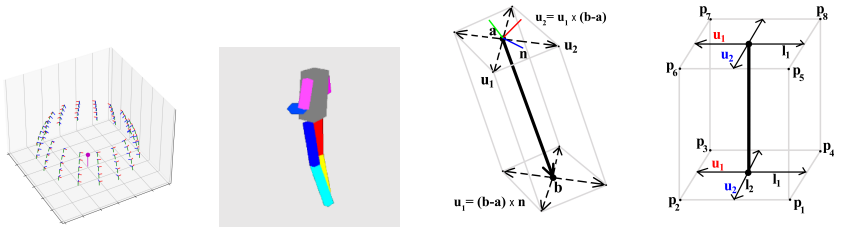
2 Related Work

3D Pose Estimation generally involves regression [4, 7, 10, 16, 20, 23, 42, 45, 50] ending with a fully-connected layer, or a voxel-based approach [24,26,27,46] with fully-convolutional supervision, the latter with a target space of size $w \times h \times d \times N$, where w is the width, h height, d depth, and N is the number of joints. However, position regression requires a training-set-dependent normalization (*e.g.* z-score) [16, 23, 30, 32, 36, 37, 45, 50]. Both the graph convolution based approach [7, 20, 42, 45, 50] and hypothesis generation approach [16, 30, 37] rely on z-score normalization to improve same-, and most crucially, cross- dataset performance. To address missing depth information, Pavlakos *et al.* [26] include ordinal relations in training; Zhou *et al.* [46] use a heatmap triplet-based intermediate representation per part.

Part Based Approach Kundu *et al.* [15] applies an unsupervised part-guided approach to 3D pose estimation. From an image, they generate part-segmentation with the help of intermediate 3D pose and a 2D part dictionary.

Viewpoint Viewpoint estimation generally boils down to regressing some form of (θ, ϕ) [8], rotation matrix [35, 49], or quaternions [39]. Regardless of the approach, everyone agrees on viewpoint estimation relative to the subject. However, relative subject rotation makes it harder to estimate viewpoint accurately.

Relation to previous work We train on synthetically-generated abstract images of ground-truth 3D human poses. The abstract image contains opaque limbs that are uniquely color-coded (implicitly defining a part-map). We believe the abstract image contains the minimum information required to completely describe a human pose. Opaque 3D humanoid limbs have been used previously; for example their 3D bodies have been ray-traced as a means of determining occlusion [6]; but so far as we know such opaque limbs have never been used to train a CNN directly to estimate 3D pose, nor to augment image datasets by transforming 3D GT keypoints into an unlimited number of synthetic viewpoints (both used to aid training our Stage 2). Our own early tests showed that regression on 3D joint positions performs extremely badly across datasets when the same z-score parameters are used for both training and test sets, and improves only marginally if the normalization parameters are independently computed for both training and test sets (which is infeasible in the field, but is reported in Tab. 3 below). Conversely, voxel regression [27] presents a trade-off in performance vs. memory footprint as voxel resolution is increased. Our pose encoding (1) does not require training set dependent normalization, (2) takes much less memory than a voxel-based representation, and (3) being heatmap-based, it integrates well in a fully-convolutional setup. Finally, most methods encode the viewpoint using a rotation matrix, sine and cosines, or quaternions; all of these methods suffer from a discontinuous mapping at 2π. In contrast, our method avoids discontinuities by mapping both pose and viewpoint heatmaps to a cylinder that wraps around at 2π.



(a) Synthetic Env. **(b)** Cube Abstract **(c)** Limb Generation **(d)** Torso Generation

Fig. 3: (a) Synthetic environment setup: Cameras are arranged spherically, all pointing to $\boldsymbol{f}$ (magenta dot near the center). (b) Blue-coloring the left forearm and femur allows easy identification of the subject's front—in contrast to "stick figures" that omit occlusion information, introducing ambiguity in determining the subject's "front". (c) Limb generation from a vector; (d) Torso generation from right and forward vectors.

3 Method

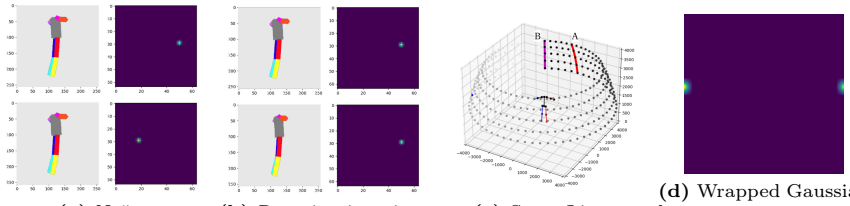
3.1 Synthetic Environment

The environment we use to generate an unlimited supply of synthetic abstract images from any given pose consists of a room full of cameras all pointing to the same fixed point at the center of the room. We define $\mathbf{T} \in \mathbb{R}^{\mathrm{X}\times\mathrm{Y}\times 3}$, the translation/position of the cameras in X columns and Y rows. As shown in Fig. 3a, the fixed point is defined as, $\boldsymbol{f} = \frac{c}{\mathrm{XY}} \sum \mathbf{T}$, where the constant $c < 0.5$ defines the desired height of the "center"; each camera is related to the room via a rotation matrix, $\mathrm{R} \in \mathbb{R}^{\mathrm{X}\times\mathrm{Y}\times 3\times 3}$. We compute the look vector as $\boldsymbol{l}_{ij} = \boldsymbol{f} - \mathbf{T}_{ij}$ for camera (i, j) and take a cross-product with $-\hat{z}$ as the up vector to compute the right vector $\boldsymbol{r}$, all of which are fine-tuned to satisfy orthonormality by a series of cross-products. Refer to Sec. 4, for predefined values.

3.2 Abstract Shape Representation

Our abstract shapes come in two variants: Cube and Flat. Using a mixture of both helps the network learn the underlying pose structure without overfitting. We show later that the flat variant is easy to obtain from images. To ensure that occlusion information is clear in both variants, our robot's 8 limbs and torso use 9 easily-distinguishable, high-contrast colors (Fig. 3b).

The Cube Variant has 3D limbs and torso formed by cuboids with orthogonal edges formed via appropriate cross-products; limbs (Fig. 3c) have a long axis (a to b) along the bone with a square cross-section, while the torso (Fig. 3d) is longest along the spine and has a rectangular cross-section. While the limb cuboid is generated from a single vector (a to b), the torso is generated by a body-centered coordinate system [39] (see Supplementary for details). All endpoints are compiled into a matrix $\mathrm{X}_{3D} \in \mathbb{R}^{3\times N}$, where N is the number of vertices. We project these points to $\mathrm{X}_{2D} \in \mathbb{R}^{2\times N}$ using the focal length f_{cam} and camera



(a) Naïve **(b)** Rotation invariant **(c)** Seam Lines **(d)** Wrapped Gaussian heatmap

Fig. 4: (a) Naïve approach of encoding viewpoint. As can be seen, for a rotated subject, we have same image but different viewpoint encoding. (b) Rotation invariant approach makes sure we have the same encoding if the image is same even if the subject is rotated. (c) Seam lines after computing cosine distances. Camera indices (black=0, white=63), rotate with subject. Seam line A (red) is the original starting point of the indices. Seam line B (purple) is the new starting point consistent with subject's rotation. (d) A Gaussian heatmap warped horizontally.

center c_{cam} (predefined for a synthetic room). Using the QHull algorithm [2], we compute the convex hull of the projected 2D points for each limb. We compute the Euclidean distance between each part's midpoint and the camera. Next, we iterate over the parts in order of longest distance, extract the polygon from hull points, and assign limb colors. While obtaining the 3D variant this way, we obtain a binary limb occlusion matrix for l limbs, $\mathbf{L} \in \mathbb{Z}^{l \times l}$, where each entry (u, v) determines whether limb u is occluding limb v if there is polygonal overlap above certain threshold.

The Flat Variant utilizes the limb occlusion matrix $\mathbf{L}$ and 2D keypoints $\mathbf{X}_{2D}$ to render the abstract image. $\mathbf{L}$ is used to topologically sort the order to render the limbs farthest to nearest. The limbs in this variant can be easily obtained by rendering a rectangle with the 2D endpoints forming a principal axis. If the rectangle area is small —for example if the torso is sideways or a limb points directly at the camera— we inflate to make the limbs more visible. We follow a similar approach while rendering the torso with four endpoints (two hips and two shoulders). The detailed algorithm is presented in the supplementary.

3.3 Viewpoint Encoding

Fig. 4a shows a naive encoding of azimuth (θ) and elevation (ϕ), which can result in two viewpoints generating the same image—a serious problem because not even a neural network can circumvent the fact that the viewpoint and pose problems must be 1-to-1 mappings with the input image—*i.e.*, mathematically *injective*. Our solution preserves the 1-to-1 nature of poses to images (cf. Fig. 5b).

The matrix representation of our cylindrical coordinates wrap around at the *seam line*; crucially, we arrange for the viewpoint seam to always lie *behind* the subject (Fig. 4c), which ensures the coordinates on the matrix always stay in a fixed point related to the subject's orientation.

Formally, we compute the cosine distance between the subject's forward vector $\mathbf{F}_s$ projected onto xy-plane of the room $\mathbf{F}_{sp}$, and camera's forward vector

$\mathbf{F}_c$ and place the seam line (index 0 and 63 of the matrix) directly behind the subject. Fig. 4b reflects the improvement from Fig. 4a. Note for the same input, we have same viewpoint encoding now.

To learn a spherical mapping, we have to make the network understand the spherical positioning of the cameras. In general, a normal heatmap-based regression will clip the Gaussian at the border of the matrix. On the contrary, we allow the Gaussian heatmaps in the matrix to wrap around at the boundaries — corresponding to the seam line. Our Gaussian is

$$\mathcal{G}(x, y, \mu_x, \mu_y) = \exp^{-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma^2}}. \tag{1}$$

The associated heatmap is

$$\mathcal{H}^v[i,j] = \begin{cases} \mathcal{G}(j, i, \mu_x, \mu_y), & \text{if } |\mu_x - j| < W_k, \\ \mathcal{G}(j - I_w, i, \mu_x, \mu_y), & \text{if } |j - I_w - \mu_x| < W_k, \\ \mathcal{G}(j + I_w, i, \mu_x, \mu_y), & \text{if } |\mu_x - I_w - j| < W_k, \end{cases} \tag{2}$$

where (μ_x, μ_y) is the index of the viewpoint in our rotated synthetic room, I_w is the image size, and $W_k = 2\sigma + 1$ is the kernel width, with $\sigma = 1$. Algorithm 1 is used to rotate the camera indices in the synthetic room to ensure the camera position is consistent with the subject. This encodes the camera position in subject space and addition of Gaussian heatmap relaxes the area for network to optimize on (i.e. picking an almost approximate neighboring camera).

3.4 Pose Encoding

We decompose the pose into bone vectors $\mathcal{B}_r$, and bone lengths B_r, both relative to parent joint. Let the synthetic environment's selected camera rotation matrix be R_{ij}, and $\mathcal{B}_{r_{ij}} = \mathrm{R}_{ij}{}'\mathcal{B}_r$ be the bone vectors in R_{ij}'s coordinate space. Then, we normalize the spherical angles (θ, ϕ) of $\mathcal{B}_{r_{ij}}$ from range $[-2\pi, 2\pi]$ to range

Algorithm 1: Rotate Camera Array

Data: S_e (Synthetic Environment), $\mathbf{F}_s$ (Subject Forward Vector)
Result: T', R'

$\mathbf{F}_c \leftarrow \mathrm{S}_e$.camera_forwards;
$\mathbf{F}_{sp} \leftarrow \mathbf{F}_s - (\mathbf{F}_s \cdot \hat{z})\hat{z}$;
$\mathrm{D} \leftarrow \mathbf{F}_c \cdot \mathbf{F}_{sp}$;
$\mathcal{S} \leftarrow$ argmax D;
$\mathcal{I} \leftarrow \mathrm{S}_e$.original_index_array;
$\mathcal{I}_r \leftarrow$ rotate_index_array($\mathcal{I}$, $\mathcal{S}$);
$\mathrm{T} \leftarrow \mathrm{S}_e$.camera_position;
$\mathrm{R} \leftarrow \mathrm{S}_e$.camera_rotation;
$\mathrm{T}' \leftarrow \mathrm{T}[\mathcal{I}_r]$;
$\mathrm{R}' \leftarrow \mathrm{R}[\mathcal{I}_r]$;

(a) Input image **(b)** Predicted viewpoint **(c)** Predicted pose **(d)** Ground truth pose

Fig. 5: A sample output from our network. (a) is the abstract image (cube variant) fed into the network. (b) the predicted viewpoint heatmap. (c) is the reconstructed pose from the pose and viewpoint heatmaps. The arrow shooting out the pose's left indicates the camera was left of the subject. (d) is the ground-truth 3D pose which is the reconstructed pose related with a rotation and the visible difference is due to camera viewpoint, not actual pose, which has error 37 mm.

$[0, 127]$. Note that this encoding is not dependent on any normalization of the training—and by implication, is also independent of any normalization of the test set. We now have (θ, ϕ) normalized on a 128×128 grid. We take a similar approach to viewpoint encoding and allow the Gaussian heatmap generated around the matrix locations to wrap around at the boundaries. Note that in the viewpoint encoding, we only need to account for horizontal wrapping, whereas in pose we wrap both coordinates. For joint i and $k_1, k_2 \in [-\frac{W_k}{2}, \frac{W_k}{2})$,

$$\mathcal{H}_i^p[h, g] = \mathcal{G}(k_1, k_2, 0, 0) \tag{3}$$

where $h = \mu_y + k_2 (\text{mod } I_w)$ and $g = \mu_y + k_1 (\text{mod } I_w)$. Thus, we have another heatmap-based encoding for the pose. This encoding $\mathcal{H}^p \in \mathbb{R}^{128 \times 128 \times N}$, where N is the number of joints. Fig. 4d shows a wrapped version of the heatmap.

3.5 Pose Reconstruction in Stage 2

Since the camera viewpoint is encoded in a subject-based coordinate system, the first step of pose reconstruction is to transform the camera's position from subject-centered coordinates to world coordinates.The mathematical details are in the Supplementary, but the upshot is shown in Fig. 5, which shows the unseen test output from our actual network. Specifically, in Fig. 5c, note how the reconstructed pose is rotated from the ground-truth pose in Fig. 5d. The arrow shooting out from the subject's left in Fig. 5c, indicates the relative position of the camera when the picture was taken.

4 Implementation

We calculated average bone lengths for the H36M training set [13]. The viewpoint was discretized into 24×64 indices, encoded to occupy rows $[21, 45]$ of a 64×64 heatmap matrix, giving angular resolution 5.625°. Our synthetic environment uses fixed point scalar (cf. Sec. 3.1) of 0.4, with radius 5569 mm. (In principle, we

could easily include cameras covering the entire sphere, for example to account for images of astronauts floating in the ISS viewed from *any* angle.) The pose was normalized to fall into the range $[0, 128]$ to occupy a $13 \times 128 \times 128$ matrix, with 13 being the number of limbs since we have 14 joints. For simplicity, we followed similar network architecture for all of the networks. We always use HRNet [31] pretrained on MPII [1], and COCO [19] for feature extraction.

The "image-to-abstract" network outputs 2D keypoint heatmaps and a binary limb occlusion matrix of dimension 9×9. We keep the original architecture for the 2D pose prediction and add a separate branch with a series of three interleaved convolution and batch normalization block with kernel size 3×3 to reduce the resolution. The reduced output is flattened and passed through two fully connected blocks to output the limb occlusion matrix as an 81D vector. We optimized the mean squared error loss on the 2D heatmaps and binary cross-entropy loss for the limb occlusion matrix.

The pose network consists two Convolution and Batch Normalization block pairs, followed by a transposed Convolution to match the output size of 128×128. All the convolution blocks use a 3×3 kernel with padding and stride set to 1. The final transposed convolution uses stride 2 and outputs a $13 \times 128 \times 128$ size tensor. For viewpoint estimation, we apply only one Convolution and Batch Normalization pair on the output of HRNet. The final stage is a regular convolution block that shrinks the output channel to 1 and outputs a $1 \times 64 \times 64$ size tensor. Since our target is a heatmap, we applied the standard L2 loss.

All training used batch size 64, with Adam [14] as our optimizer using Cosine Annealing with warm restart [21] and a learning rate warming from 1×10^{-9} to 1×10^{-3}. The viewpoint network ran for 200 epochs (2 days), and the pose network ran for 300 epochs (4 days), both on an RTX 3090, though it was stopped early due to convergence. When training the "abstract to pose and viewpoint" network, on every epoch we pick a random set of camera indices and render an abstract image from one of the two variants with equal probability. Thus, no two epochs are the same. This randomness minimizes overfitting—in turn helping with generalization—though it results in longer time-to-convergence.

5 Experiments

After we explain the datasets, metrics and models used in Sec. 5.1, we first show the results from our method under same dataset benchmark in Sec. 5.2. Then, we shift our focus to generalization capability across datasets in Sec. 5.3. We also report qualitative results in Sec. 5.4 and reflect on other experiments in Sec. 5.5.

5.1 Datasets, Metrics and Models

Datasets We train our HRNet backbone framework on both MPII [1] and COCO [19] first one 2D keypoint prediction task. For the rest of the task, we only train on Human3.6M dataset [13]. We report cross-dataset results on the test sets of Geometric Pose Affordance Dataset (GPA) [38], 3D Poses in the Wild

Dataset [22], and SURREAL Dataset [34] dataset. For the details of the dataset please refer to [39].

Evaluation Metrics We report Mean Per Joint Position Error (MPJPE) in millimeters, which we call Protocol #1 and MPJPE after Procrustes Alignment (PA-MPJPE) as Protocol #2, following convention. Since the reconstructed pose is related with the ground truth by rotation only, we report it under Protocol #1. Further, PA-MPJPE reduces the error since the reconstruction uses preset bone-lengths. This becomes important in cross-dataset benchmarks.

Models First, we train our "abstract-to-pose" network with a uniform mixture of Cube and Flat variant of the abstract images, which we will refer to as **"Model (Hybrid)"**. We also trained the same network for fewer epochs on the Cube variant separately which we will refer to as **"Model (Cube)"**. The former is tuned to handle both the flat and cube variant of the abstract image. Since we get the "Flat Abstract Image" from the real world image through the Stage 1 of our approach, the hybrid model can reconstruct the pose from this input. However, the Flat variant is a simplified version of the Cube variant, which is more robust and preserve occlusion. Pitting these two models in contrast to each other helps us further assess the capabilities of both variants and models.

5.2 Conventional Comparisons - Same Dataset

During training Stage 1, "image to abstract", we fine-tuned the trained HRNet network with H36M's training set along with limb ordering annotations. In Stage 2, "abstract to pose and viewpoint", we take each GT pose and pair it with the corresponding GT synthetically generated abstract image from a randomly chosen synthetic viewpoint taken from our 64×12 synthetic cameras; for testing, we use only poses and cameras from the real dataset.

Results are in Tab. 1 and Tab. 2. Our method is competitive with both image (Stage 1 & 2) and abstract image (Stage 2 only) as input. Since the latter is comparable to methods using GT 2D keypoints as input, we compare against such methods when available. We believe this is a fair comparison with our work, since the GT 2D keypoints are projected from 3D joints. During reconstruction, we always use a preset bone-length. PA-MPJPE score on Tab. 2, which includes rigid transformation, accounts for bone-length variation and reduces the error even more. Note that, in both these table we do not claim the lead even though we lead in quite a few cases. Our focus is to prove viability of using abstract image as a replacement for 2D keypoints in lifting networks. Our method is comparable, and sometimes superior, across the board.

5.3 Cross-Dataset Generalization

Our primary focus is improving cross-dataset performance through extensive training on synthetically generated images. For this experiment, we only consider the case where we train on H36M dataset without any *domain adaptation training* (*e.g.* [44]). To test generalization capabilities, we render the images from

Table 1: Quantitative comparisions of MPJPE (Protocol #1) between the ground truth 3D pose and reconstructed 3D pose. Our method shows competitive performance when image is used as an input. Hybrid abstract-to-pose model is trained on mixed Flat and Cube abstract images. Note: the results of others, obtained from GT 2D keypoints, are marked with an asterisk. Multi-frame methods [12, 29, 41, 48] not included.

Method	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	Avg.
Moreno [25]	69.5	80.1	78.2	87	100.7	102.7	76	69.6	104.7	113.9	89.7	98.5	79.2	82.4	77.2	87.3
Chen [4]	71.6	66.6	74.7	79.1	70.1	93.3	67.6	89.3	90.7	195.6	83.5	71.3	85.6	55.7	62.5	82.7
Martinez [23]	51.8	56.2	58.1	59	69.5	78.4	55.2	58.1	74	94.6	62.3	59.1	65.1	49.5	52.4	62.9
Yang [40]	51.5	58.9	50.4	57	62.1	65.4	49.8	52.7	69.2	85.2	57.4	58.4	43.6	60.1	47.7	58.6
Sharma [30]	48.6	54.5	54.2	55.7	62.6	72	50.5	54.3	70	78.3	58.1	55.4	61.4	45.2	49.7	58
Zhao [45]	47.3	60.7	51.4	60.5	61.1	49.9	47.3	68.1	86.2	55	67.8	61	42.1	60.6	45.3	57.6
Pavlakos [26]	48.5	54.4	54.4	52	59.4	65.3	49.9	52.9	65.8	71.1	56.6	52.9	60.9	44.7	47.8	56.2
Ci [7]	46.8	52.3	44.7	50.4	52.9	68.9	49.6	46.4	60.2	78.9	51.2	50	54.8	40.4	43.3	52.7
Li [16]	43.8	48.6	49.1	49.8	57.6	61.5	45.9	48.3	62	73.4	54.8	50.6	56	43.4	45.5	52.7
Martinez [23]*	37.7	44.4	40.3	42.1	48.2	54.9	44.4	42.1	54.6	58	45.1	46.4	47.6	36.4	40.4	45.5
Zhao [45]*	37.8	49.4	37.6	40.9	45.1	41.4	40.1	48.3	50.1	42.2	53.5	44.3	40.5	47.3	39	43.8
Zhou [46]	34.4	42.4	36.6	42.1	38.2	39.8	34.7	40.2	45.6	60.8	39	42.6	42	29.8	31.7	39.9
Gong [10]*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	38.2
Zhai [43]*	31.3	34.0	28.0	32.0	33.1	42.1	34.1	28.1	33.6	39.8	31.7	32.9	33.8	26.7	28.9	32.7
Ours (Hybrid)	40.7	52.1	43.8	48.2	46.9	54.4	49.1	51.4	55.6	65.2	47	49.5	44.1	42.1	42.3	48.8
Ours (Cube)	29.0	32.5	29.1	32.0	29.9	38.8	32.1	32.5	38.7	49.5	33.2	33.7	32.1	29.9	29.9	33.5

Table 2: Quantitative comparison of PA-MPJPE (Protocol #2) between the ground truth 3D pose and reconstructed 3D pose. We follow the same notation as Tab. 1. Lower is better.

Method	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	Avg.
Moreno [25]	66.1	61.7	84.5	73.7	65.2	67.2	60.9	67.3	103.5	74.6	92.6	69.6	71.5	78	73.2	74
Martinez [23]	39.5	43.2	46.4	47	51	56	41.4	40.6	56.5	69.4	49.2	45	49.5	38	43.1	47.7
Li [16]	35.5	39.8	41.3	42.3	46	48.9	36.9	37.3	51	60.6	44.9	40.2	44.1	33.1	36.9	42.6
Ci [7]	36.9	41.6	38	41	41.9	51.1	38.2	37.6	49.1	62.1	43.1	39.9	43.5	32.2	37	42.2
Pavlakos [26]	34.7	39.8	41.8	38.6	42.5	47.5	38	36.6	50.7	56.8	42.6	39.6	43.9	32.1	36.5	41.8
Sharma [30]	35.3	35.9	45.8	42	40.9	52.6	36.9	35.8	43.5	51.9	44.3	38.8	45.5	29.4	34.3	40.9
Zhou [46]	21.6	27	29.7	28.3	27.3	32.1	23.5	30.3	30	37.7	30.1	25.3	34.2	19.2	23.2	27.9
Ours (Hybrid)	32.8	38.0	35.3	38.2	38.1	44.8	37.9	37.9	47.1	59.4	40.4	39.7	38.1	33.3	34.4	39.7
Ours (Cube)	23.8	27.6	25.5	27.1	26.1	34.5	27.0	28.7	34.4	46.8	28.9	29.0	28.4	24.4	23.9	29.1

GPA, 3DPW, and SURREAL dataset. We report results obtained from all three variants of our models.

To our knowledge, Wang *et al.* [39], is the only work with an extensive cross-dataset analysis on 4 datasets. To add more contenders, we re-implemented methods from Martinez *et al.* [23] and Zhao *et al.* [45], since their code was available and easily adapted. Both rely on z-score normalization of the testing set separately from the training set—though we note that such a normalization is impossible "in the wild". Even with this advantage, our method still leads in most cases by 30%–40% in cross-dataset performance (cf. Tab. 3). Goel *et al.* [9] outperforms us in 3DPW dataset. We believe this can be attributed to training a transformer based network on two 3D pose datasets. Gong *et al.* [10] reported cross-dataset performance for few networks on 3DPW dataset in PA-MPJPE. As shown in Tab. 3, we outperform the other methods by about a factor of 2.

Fairness of Comparison. We choose to report all the results both obtained through GT 2D keypoints and images, because cross-dataset results are hard to obtain. Both Martinez *et al.* [23] and Zhao *et al.* [45] pass only 2D keypoints to

Table 3: Cross-Dataset results on GPA, 3DPW, SURREAL in MPJPE and PA-MPJPE. We show significant improvement across all dataset except 3DPW. Asterisk marks our own experiment. Note: Goel *et al.* [9] trained their network on two 3D pose datasets. All the other networks were trained on H36M.

Method	MPJPE				PA-MPJPE		
	H36M	GPA	3DPW	SURR.	GPA	3DPW	SURR.
Martinez [23]*	55.52	117.37	135.53	108.63			
Zhao [45]*	53.59	115.01	154.3	103.75			
Wang [39]	52	98.3	124.2	114			
Goel [9] (H36M+3DHP Training)	44.8	-	**70.0**	-			
Zhao [45]					-	152.3	-
Martinez [23]					-	145.2	-
ST-GCN [3] (1-Frame)					-	154.3	-
VPose [28] (1-Frame)					-	146.3	-
Zhao + Gong [10]					-	140	-
Martinez + Gong [10]					-	130.3	-
ST-GCN (1-Frame) + Gong [10]					-	129.7	-
VPose (1-Frame) + Gong [10]					-	129.7	-
Goel [9]					-	**44.5**	-
Ours (Hybrid)	40.01	99.43	106.27	80.13	70.1	71.39	59.06
Ours (Cube)	**33.52**	**92.31**	95.83	**65.62**	**69.48**	64.28	**51.53**

the later 3D pose estimation phase. Note the PA-MPJPE reported by Gong *et al.* [10] is higher than the MPJPE score we obtained for Martinez *et al.* [23] and Zhao *et al.* [45], which shows that we have given them all possible advantages.

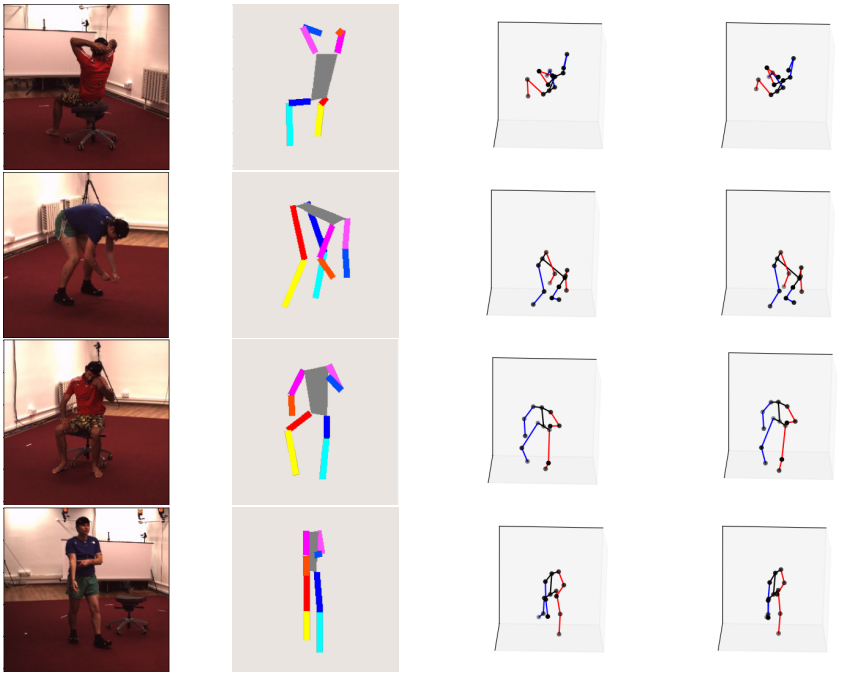
5.4 Qualitative Results

Fig. 6 shows the qualitative performance of our network on H36M. We see viewpoint estimation indicated by the blue arrow on the second column of each test sample. This, indeed shows the accuracy and efficacy of our method on separating viewpoint from pose.

5.5 Ablative and Other Experiments

Number of Vertical Bins Our synthetic environment has hundreds of cameras placed systematically in "levels" on the sphere, pointing inwards (cf. Sec. 3.1). For this study, we reduced the number of levels. Intuitively, decreasing the number of levels should reduce performance, which is what we see in Fig. 7a. Increase in number of vertical bins shows a global improvement in cross-dataset performance. In Figs. 7b and 7c, we have plotted the azimuth and elevation distribution across all datasets, demonstrating the generalization and augmentation ability of our approach. Notice how our approach that relies on random viewpoints, have an almost uniform distribution. In contrast, notice how majority of the datasets have more data points distributed at the level of the subject.

Angular Error vs Bin Size With an $N_g \times N_g$ heatmap, we expect performance to decrease with decreasing N_g. From our experiments, the angular error goes up as we decrease the grid resolution. We see the biggest jump from 16×16



(a) Input Image **(b)** Abstract Image **(c)** Ground Truth **(d)** Prediction

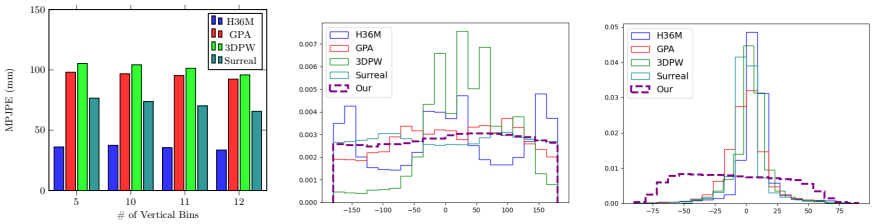
Fig. 6: Qualitative results on H36M dataset.

(13.35°) to 32×32 (9.24°), with a smaller change when going to 64×64 (8.44°), implying diminishing returns at higher resolutions.

Limb Ablation: we randomly skip rendering a subset of the limbs, causing an expected increase in error and uncertainty (cf. Fig. 8a).

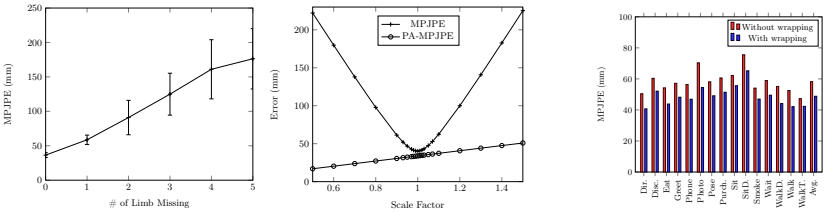
Variation in Reconstruction Scale For this experiment, we change the bone lengths that are used to reconstruct the pose, deviating 50% scale up and down from the average bone lengths of an adult person. When comparing we keep the ground-truth at default scale (cf. Fig. 8b. Obviously, the reconstructed pose will not match in scale and the error will increase when deviating from the average bone lengths. However, at the bottom of the V-shaped curve the error deviation is small – which means, if the subject scale varies by a small amount, the error is negligible. As Procrustes alignment is performed first before measuring the error in PA-MPJPE, the curve forms a straight line and stays below the V curve.

Effect of wrapping on Error We observe the impact of wrapping on the error by computing pose estimation error on all the actions of H36M dataset with and without wrapping enabled. As can be seen in Fig. 8c), we achieve an average of 9.4mm of improvement - from 58.2mm (red) to 48.8mm (blue). In addition, when we toggle wrapping in predicting viewpoint (*i.e.* naïve vs our viewpoint encoding), we see an angular errors are respectively 55.44° *vs.* 8.44°— an improvement factor of 6.6.



(a) Error vs Vertical Bins **(b)** Azimuth relative to subject **(c)** Elevation relative to subject

Fig. 7: (a) shows how the global cross-dataset error changes as the number of vertical bins is increased. (b) and (c) demonstrate how our synthetic training set using (unlimited in principle) synthetic viewpoints levels the the distribution of trained viewpoints in (b) azimuth and (c) elevation. Notice that majority of the datasets have a relative distribution that is at the level of the subject, whereas ours follow a uniform distribution.



(a) Effect of Missing Limbs **(b)** Error vs Scale **(c)** Effect of wrapping

Fig. 8: (a) error (MPJPE) vs number of missing parts. As more parts are missing from the input image, error and uncertainty increases. (b) error as a function of scaled bone length in the synthetic environment. As expected, error increases as the scale deviates 1.0. (Note that PA-MPJPE applies affine transformation which compensates for scale difference.) (c) error with (blue) and without (red) wrapping. The errors are plotted across different actions on H36M dataset. Wrapping is clearly better.

6 Conclusion

We have shown that using abstract images in the task of lifting the pose into 3D gives better cross-dataset results than using 2D keypoints, at negligible cost to same-dataset results. Moreover, abstract images also enable augmenting the "lifting" part of the framework to train from a virtually unlimited number of viewpoints. We have also presented an approach to obtain abstract image from real-world images, which shows the applicability of our approach.

References

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

2. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. **22**(4), 469–483 (dec 1996). https://doi.org/10.1145/235815.235821, https://doi.org/10.1145/235815.235821
3. Cai, Y., Ge, L., Liu, J., Cai, J., Cham, T.J., Yuan, J., Thalmann, N.M.: Exploiting spatial-temporal relationships for 3d pose estimation via graph convolutional networks. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 2272–2281 (2019). https://doi.org/10.1109/ICCV.2019.00236
4. Chen, C.H., Ramanan, D.: 3d human pose estimation= 2d pose estimation+ matching. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7035–7043 (2017)
5. Chen, C.H., Tyagi, A., Agrawal, A., Drover, D., Mv, R., Stojanov, S., Rehg, J.M.: Unsupervised 3d pose estimation with geometric self-supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5714–5724 (2019)
6. Cheng, Y., Yang, B., Wang, B., Yan, W., Tan, R.T.: Occlusion-aware networks for 3d human pose estimation in video. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019)
7. Ci, H., Wang, C., Ma, X., Wang, Y.: Optimizing network structure for 3d human pose estimation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 2262–2271 (2019)
8. Ghezelghieh, M.F., Kasturi, R., Sarkar, S.: Learning camera viewpoint using cnn to improve 3d body pose estimation. In: 2016 Fourth International Conference on 3D Vision (3DV). pp. 685–693 (2016). https://doi.org/10.1109/3DV.2016.75
9. Goel, S., Pavlakos, G., Rajasegaran, J., Kanazawa, A., Malik, J.: Humans in 4d: Reconstructing and tracking humans with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 14783–14794 (October 2023)
10. Gong, K., Zhang, J., Feng, J.: Poseaug: A differentiable pose augmentation framework for 3d human pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8575–8584 (2021)
11. Habibie, I., Xu, W., Mehta, D., Pons-Moll, G., Theobalt, C.: In the wild human pose estimation using explicit 2d features and intermediate 3d representations. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10905–10914 (2019)
12. Hu, W., Zhang, C., Zhan, F., Zhang, L., Wong, T.T.: Conditional directed graph convolution for 3d human pose estimation. In: Proceedings of the 29th ACM International Conference on Multimedia. pp. 602–611 (2021)
13. Ionescu, C., Papava, D., Olaru, V., Sminchisescu, C.: Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. IEEE Transactions on Pattern Analysis and Machine Intelligence **36**(7), 1325–1339 (2014). https://doi.org/10.1109/TPAMI.2013.248
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
15. Kundu, J.N., Seth, S., Jampani, V., Rakesh, M., Babu, R.V., Chakraborty, A.: Self-supervised 3d human pose estimation via part guided novel image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6152–6162 (2020)
16. Li, C., Lee, G.H.: Generating multiple hypotheses for 3d human pose estimation with mixture density network. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9887–9895 (2019)

17. Li, S., Chan, A.B.: 3d human pose estimation from monocular images with deep convolutional neural network. In: Asian Conference on Computer Vision. pp. 332–347. Springer (2014)
18. Li, S., Zhang, W., Chan, A.B.: Maximum-margin structured learning with deep networks for 3d human pose estimation. In: Proceedings of the IEEE international conference on computer vision. pp. 2848–2856 (2015)
19. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. pp. 740–755. Springer (2014)
20. Liu, K., Ding, R., Zou, Z., Wang, L., Tang, W.: A comprehensive study of weight sharing in graph networks for 3d human pose estimation. In: European Conference on Computer Vision. pp. 318–334. Springer (2020)
21. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
22. von Marcard, T., Henschel, R., Black, M., Rosenhahn, B., Pons-Moll, G.: Recovering accurate 3d human pose in the wild using imus and a moving camera. In: European Conference on Computer Vision (ECCV) (sep 2018)
23. Martinez, J., Hossain, R., Romero, J., Little, J.J.: A simple yet effective baseline for 3d human pose estimation. In: Proceedings of the IEEE international conference on computer vision. pp. 2640–2649 (2017)
24. Mehta, D., Sridhar, S., Sotnychenko, O., Rhodin, H., Shafiei, M., Seidel, H.P., Xu, W., Casas, D., Theobalt, C.: Vnect: Real-time 3d human pose estimation with a single rgb camera. Acm transactions on graphics (tog) **36**(4), 1–14 (2017)
25. Moreno-Noguer, F.: 3d human pose estimation from a single image via distance matrix regression. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2823–2832 (2017)
26. Pavlakos, G., Zhou, X., Daniilidis, K.: Ordinal depth supervision for 3d human pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7307–7316 (2018)
27. Pavlakos, G., Zhou, X., Derpanis, K.G., Daniilidis, K.: Coarse-to-fine volumetric prediction for single-image 3d human pose. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7025–7034 (2017)
28. Pavllo, D., Feichtenhofer, C., Grangier, D., Auli, M.: 3d human pose estimation in video with temporal convolutions and semi-supervised training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
29. Shan, W., Liu, Z., Zhang, X., Wang, Z., Han, K., Wang, S., Ma, S., Gao, W.: Diffusion-based 3d human pose estimation with multi-hypothesis aggregation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14761–14771 (2023)
30. Sharma, S., Varigonda, P.T., Bindal, P., Sharma, A., Jain, A.: Monocular 3d human pose estimation by generation and ordinal ranking. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 2325–2334 (2019)
31. Sun, K., Xiao, B., Liu, D., Wang, J.: Deep high-resolution representation learning for human pose estimation. In: CVPR (2019)
32. Sun, X., Shang, J., Liang, S., Wei, Y.: Compositional human pose regression. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2602–2611 (2017)

33. Tekin, B., Katircioglu, I., Salzmann, M., Lepetit, V., Fua, P.: Structured prediction of 3d human pose with deep neural networks. arXiv preprint arXiv:1605.05180 (2016)
34. Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M.J., Laptev, I., Schmid, C.: Learning from synthetic humans. In: CVPR (2017)
35. Wandt, B., Little, J.J., Rhodin, H.: Elepose: Unsupervised 3d human pose estimation by predicting camera elevation and learning normalizing flows on 2d poses. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6635–6645 (June 2022)
36. Wandt, B., Rosenhahn, B.: Repnet: Weakly supervised training of an adversarial reprojection network for 3d human pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7782–7791 (2019)
37. Wang, M., Chen, X., Liu, W., Qian, C., Lin, L., Ma, L.: Drpose3d: Depth ranking in 3d human pose estimation. arXiv preprint arXiv:1805.08973 (2018)
38. Wang, Z., Chen, L., Rathore, S., Shin, D., Fowlkes, C.: Geometric pose affordance: 3d human pose with scene constraints. In: Arxiv (2019)
39. Wang, Z., Shin, D., Fowlkes, C.C.: Predicting camera viewpoint improves cross-dataset generalization for 3d human pose estimation. In: European Conference on Computer Vision. pp. 523–540. Springer (2020)
40. Yang, W., Ouyang, W., Wang, X., Ren, J., Li, H., Wang, X.: 3d human pose estimation in the wild by adversarial learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5255–5264 (2018)
41. Yu, B.X., Zhang, Z., Liu, Y., Zhong, S.h., Liu, Y., Chen, C.W.: Gla-gcn: Global-local adaptive graph convolutional network for 3d human pose estimation from monocular video. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8818–8829 (2023)
42. Zeng, A., Sun, X., Huang, F., Liu, M., Xu, Q., Lin, S.: Srnet: Improving generalization in 3d human pose estimation with a split-and-recombine approach. In: European Conference on Computer Vision. pp. 507–523. Springer (2020)
43. Zhai, K., Nie, Q., Ouyang, B., Li, X., Yang, S.: Hopfir: Hop-wise graphformer with intragroup joint refinement for 3d human pose estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 14985–14995 (October 2023)
44. Zhang, X., Wong, Y., Kankanhalli, M.S., Geng, W.: Unsupervised domain adaptation for 3d human pose estimation. In: Proceedings of the 27th ACM International Conference on Multimedia. p. 926–934. MM '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3343031.3351052, https://doi.org/10.1145/3343031.3351052
45. Zhao, L., Peng, X., Tian, Y., Kapadia, M., Metaxas, D.N.: Semantic graph convolutional networks for 3d human pose regression. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 3425–3435 (2019)
46. Zhou, K., Han, X., Jiang, N., Jia, K., Lu, J.: Hemlets pose: Learning part-centric heatmap triplets for accurate 3d human pose estimation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 2344–2353 (2019)
47. Zhou, X., Sun, X., Zhang, W., Liang, S., Wei, Y.: Deep kinematic pose regression. In: European Conference on Computer Vision. pp. 186–201. Springer (2016)
48. Zhu, W., Ma, X., Liu, Z., Liu, L., Wu, W., Wang, Y.: Motionbert: A unified perspective on learning human motion representations. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15085–15099 (2023)

49. Zimmermann, C., Brox, T.: Learning to estimate 3d hand pose from single rgb images. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (Oct 2017)
50. Zou, Z., Tang, W.: Modulated graph convolutional network for 3d human pose estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 11477–11487 (2021)