MeshPose: Unifying DensePose and 3D Body Mesh reconstruction

Supplementary Material

Eric-Tuan Lê[1*†] Antonis Kakolyris[2*] Petros Koutras[2] Himmy Tam[2] Efstratios Skordos[2]
George Papandreou[2] Rıza Alp Güler[2] Iasonas Kokkinos[2]
[1]University College London [2]Snap Inc.
meshpose.github.io

In this supplementary material, we present additional results and ablations for our proposed MeshPose system. We also provide more technical details on our architecture and its training for reproducibility.

We begin in Section 1 by showing more qualitative results across multiple image datasets (COCO [21], 3DPW [30], H36M [7] and 3DOH [35]) to demonstrate the wide applicability of our approach to multiple scenarios. We also provide more results on videos to showcase the temporal stability of our method even without temporal smoothing post-processing. We refer the readers to our *mp4* video provided in the *zip* file as our results are best viewed as videos.

Then, in Section 2.1, we ablate - with more metrics - the design choice for our novel VertexPose module that leverages DensePose [5] annotations to learn 2d vertex localization. We show first how our vertex-based representation (VertexPose) compares to the UV-based representation introduced in DensePose [5] in terms of DensePose metrics. Then, we evaluate multiple strategies to aggregate vertex UVs into pixel UVs to show the superiority of the barycentric UV aggregation.

In Section 2.2, we further evaluate our system with respect to occlusion on the 3DOH and 3DPW-OCC datasets. We demonstrate that MeshPose is robust to occlusion and on par with other methods.

Then, we provide a more comprehensive 2d evaluation of our approach with other competing methods by reporting more metrics in Section 2.3.

In Section 2.4, we quantitatively demonstrate our real-time inference speed on mobile device making our approach a prime candidate for AR applications.

Finally, in Section 3, we provide further details on our architecture and its training. We detail the architecture of our backbone networks, our losses and our decoding strategy allowing us to predict high resolution meshes from the low-poly topology used throughout our pipeline.

1. Qualitative Evaluation

1.1. Visualizations on COCO

In Figure 1, we showcase more results on the COCO [21] dataset. Our approach demonstrates the ability to generate image-aligned meshes even in challenging scenarios such as occlusion or truncation of the body, which are common failure modes for other human mesh recovery systems. Unlike parametric methods bound to SMPL [23] models, our non-parametric mesh prediction approach, combined with DensePose supervision, offers greater flexibility and accurately captures very diverse body shapes that previous models struggle with.

1.2. Visualizations on 3DPW

We present additional visualizations of our 3D mesh reconstruction on the 3DPW [30] dataset in Figure 2. In contrast to COCO [21], 3DPW showcases fewer occlusions, resulting in more full body meshes, and we thus focus our visualisations on 3DPW occluded subset. We show that our 3D mesh reconstruction is also competitive in these scenarios with more accurate 2D reprojection (see elbows, shoulders, limbs) while offering strong depth prediction. We also present the rendered visibility weights predicted by our system with a color map ranging from green (fully visible - predicted by the VertexPose branch) to red (non visible - predicted by the regression branch).

1.3. Visualizations on H36M

In Figure 3 (top), we display our 3D mesh reconstruction performance on the H36M [7] motion-capture dataset. We observe similar performance on this dataset that - similarly to 3DPW - exhibits only few occlusions.

*Equal contribution
†Work done while interning at Snap Inc.

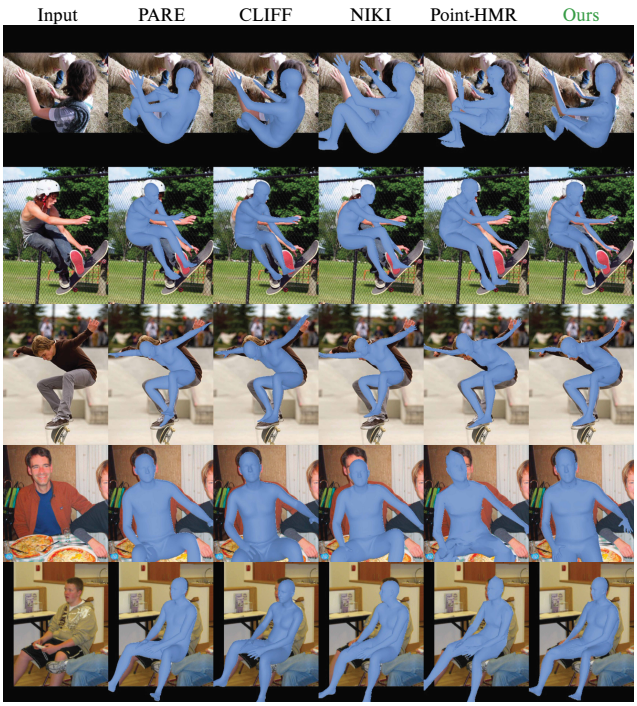


Figure 1. Qualitative comparison on COCO against 4 state-of-the-art mesh reconstruction systems. MeshPose is consistently more aligned to the silhouette of the person.

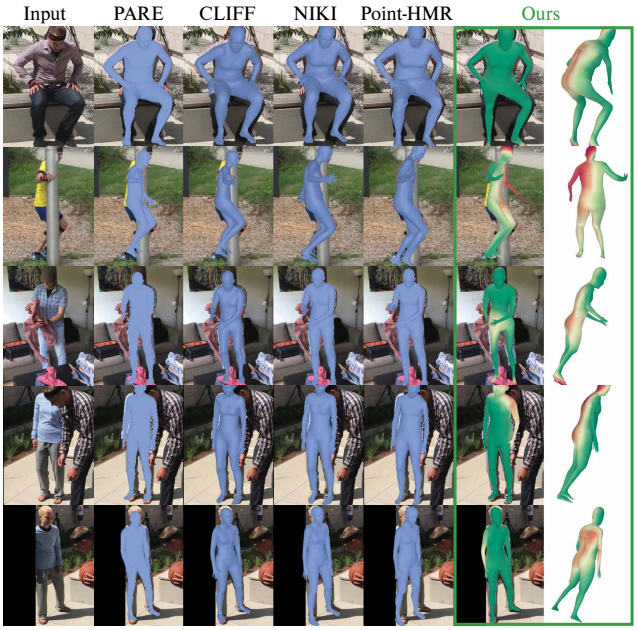


Figure 2. Qualitative comparison on 3DPW (occluded subset) against 4 state-of-the-art mesh reconstruction systems. We also display the rendered visibility weights predicted by our system with a color map ranging from green (fully visible) to red (non visible).

1.4. Visualizations on 3DOH

To demonstrate the robustness of our approach with respect to occlusion, we further showcase 3D mesh reconstruction visualisations on the 3DOH [35] dataset in Figure 3 (bottom). We compare our approach to other state-of-the-art methods in complex occluded scenes which are very common in real-world applications.

1.5. Visualizations on Internet Videos

To provide a complete qualitative analysis, we also evaluate our approach on videos of humans in action [1]. A concatenated video is available in the supplementary zip file. We also show a collection of frames in Figure 4 but we recommend watching the results on the videos. We demonstrate very strong temporal stability (low jitter) even when applied frame-by-frame without any post-processing. In the videos attached, only the detected bounding box is temporally-smoothed. Our method is lightweight and simple with real-time inference, making it a prime candidate for AR applications.

2. Quantitative Evaluation

In this section, we present an additional quantitative analysis. To isolate and eliminate potential inaccuracies coming from the bounding box predictor, each network is assessed using ground truth bounding boxes.

2.1. VertexPose Ablation Study

We provide more metrics for Table 2 introduced in the main paper. We report the standard Average Precision **AP** and Average Recall **AR**. We also provide $\mathbf{AP}_{50}$ and $\mathbf{AP}_{75}$ which measure precision at 50% and 75% IoU thresholds, $\mathbf{AP}_M$ and $\mathbf{AP}_L$ that evaluate precision for medium and large objects. The same metrics are also used for **AR**. First, in order to better understand the impact of training with vertex-based (VertexPose) representation compared to UV-based (DensePose [5]) representation, we compare both approaches in Tab. 1a evaluated in terms of DensePose metrics. For a fair comparison, we re-trained DensePose with our backbone and our training settings. We closely follow the DensePose multi-head architecture introduced in [5]. More specifically, for each pixel, we predict (i) the foreground segmentation mask I via the classification branch and (ii) the patch label c and the corresponding $[U, V]$ on that patch via the regression branch. The patch label c^* is predicted by P a 25-way (24 patches and background) classification unit $c^* = \arg\max_c P(c|i)$ while the UV is predicted by the UV regressor R_{c^*} of the predicted patch c^*, $[U, V] = R_{c^*}(i)$. We observe that the VertexPose-based results compare favorably to their DensePose-based counterparts for all metrics, thus confirming the merit of the proposed approach. The

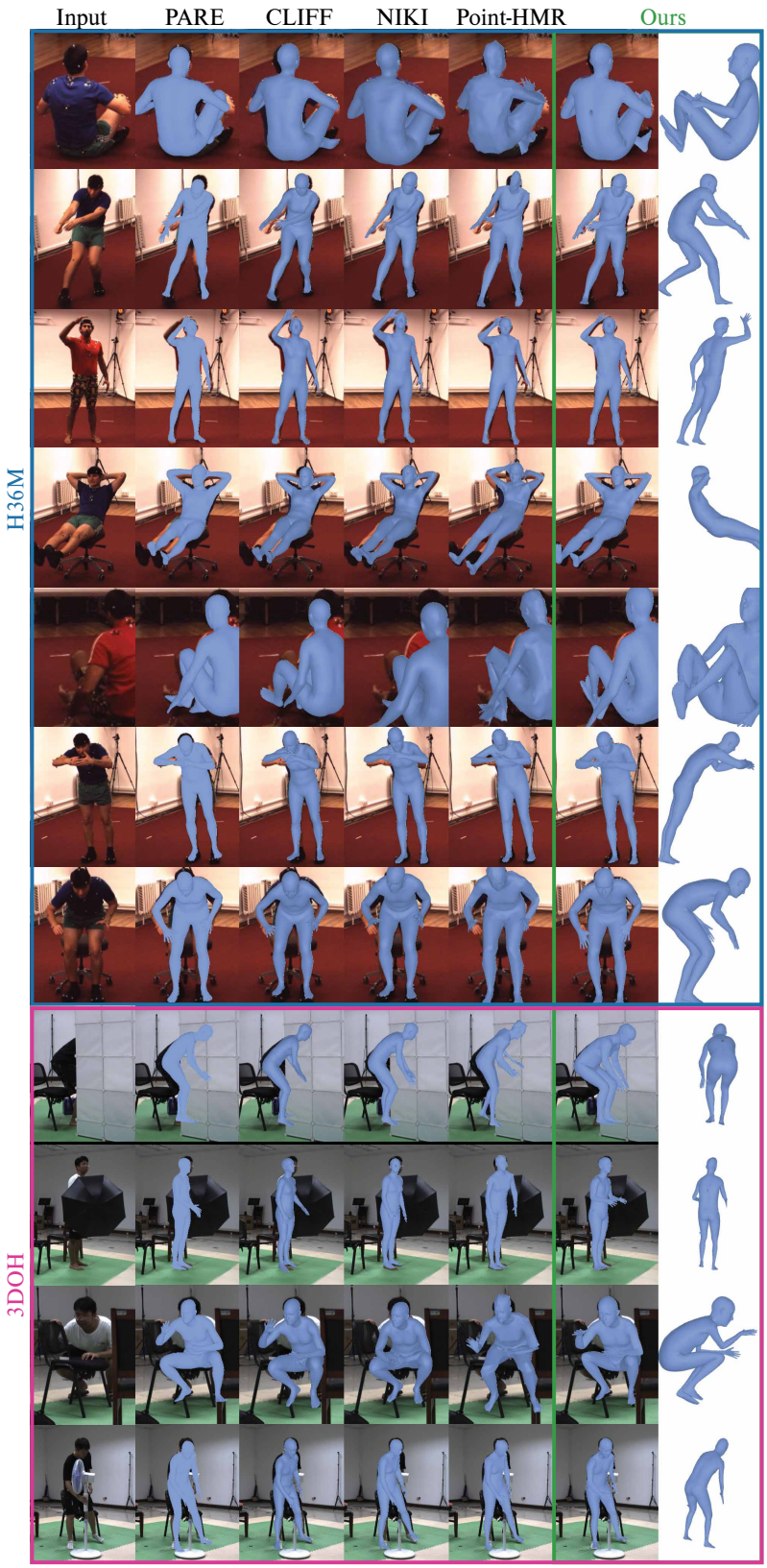


Figure 3. Qualitative comparison on H36M and 3DOH against 4 state-of-the-art mesh reconstruction systems. MeshPose exhibits stronger 2d reprojection accuracy especially on complex regions (elbows, shoulders, limbs).

Figure 4. Frames extracted from the videos provided in the supplemental as *mp4*. Results are best viewed on videos.

same results are observed across all tested architecture (MobileNet [27], ResNet [6], HRNet32 and HRNet48 [31]).

In Tab. 1b we analyze the impact of the barycentric interpolation strategy proposed in Section 3.1.1 of the main paper. As a reminder, to decode the pixel UV, we compute the barycentric combination of the UV values of the vertices belonging to the face with the largest score $f^* = \arg\max_{f \in \mathcal{F}} \sum_{n \in f} q_n$. Here q_v corresponds to the per-pixel posterior over vertices:

$$q_v = \frac{\exp(s_v)}{\sum_{k=1}^{V} \exp(s_k)} \tag{1}$$

We thus compare our proposed approach to simpler baselines: (i) decoding the pixel UV as the UV of the strongest vertex at any given pixel ('Nearest') and (ii) decoding the UV via argsoftmax over all vertices rather than those of the strongest triangle ('Global Average'):

$$\mathbf{u} = \sum_{m \in \mathcal{V}} \beta_m \mathbf{u}_m, \text{ where } \beta_m = \frac{q_m}{\sum_{n \in \mathcal{V}} q_n} \tag{2}$$

with $\mathcal{V}$ the set of all vertices. The evaluation indicates the merit of the smooth transition between vertices secured by barycentric interpolation on almost all DensePose metrics.

2.2. Occlusion Evaluation

To further validate the stability of our proposed method under occlusion, we also conducted evaluation on the object-occluded benchmark dataset: 3DPW-OCC[30, 35] and 3DOH50K[35]. **3DPW-OCC** refers to a new test-set with occluded sequences from the entire 3DPW dataset, while **3DOH** is a 3D human dataset with human activities occluded by objects, which provides 2D, 3D annotations and SMPL parameters. For a fair comparison following previous methods we train our model by including 3DOH without 3DPW (since some videos of 3DPW-OCC are from the training set).

From the evaluation results shown in Table 2, we see that the proposed MeshPose performs also very well in occluded scenarios by achieving state-of-the-art performance and outperforming approaches that are designed to deal with occlusion. Moreover, in Figure 3, we show that MeshPose achieves good mesh reconstructions even in cases with heavy object-occlusions. These results demonstrate the effectiveness of our proposed method that is able to deliver pixel-aligned 3D-mesh reconstructions and at the same time deal successfully with occlusion.

2.3. 2D Evaluation Benchmark

In Table 3, we expand the analysis from the main paper with additional evaluations with 2D metrics. For sake of completeness, we include LVD [4], though this method is not directly comparable as it has not been trained with in-the-wild datasets. We present the Average Precision (**AP**) and Average Recall (**AR**) for 2D COCO keypoints in instances where at least 80% of the keypoints are visible. This 80% threshold delineates a sub-dataset with instances that are almost fully visible, but yet remains sufficiently large to allow for meaningful conclusions. $\mathbf{AP}_M$ and $\mathbf{AP}_L$ measure medium (area between 32^2 and 96^2 pixels in the image) scale and large scale (area above 96^2 pixels in the image) object detection precision, while $\mathbf{AR}_M$ and $\mathbf{AR}_L$ assess recall for medium and large scale objects. Additionally, we provide the **AP** and **AR** metrics for the DensePose task.

Our approach surpasses the other methods in terms of 2D metrics, showcasing its strong 2D alignment.

2.4. Inference setup and mobile inference

For our desktop inference experiment, we assess the performance speed of the backbone network of each baseline methods using the original authors' official implementations. Each timing was evaluated on the same machine equipped with a Nvidia Tesla V100 GPU.

Our models are purely convolutional and thus run out-of-the-box on modern phones with accelerators. We exported the ONNX [26] versions of our models and computed their timings (FPS) on an iPhone-12 using the CoreML [3] backend, obtaining comparable timings to the GPU-desktop timings: 97, 99, and 153 FPS for Meshpose, MeshposeS, and MeshposeXS respectively.

3. Architecture and training details

In order to provide a comprehensive understanding of our method outlined in the main paper, we present additional details regarding the design of our pipeline and the training process. We begin by providing more details on the design of our backbones in Subsection 3.1. Following that, we include supplementary details concerning our multi-head decoders in Subsection 3.2. More precisely, we detail our vertex and visibility regression branch (3.2.1), our custom silhouette rendering - both introduced in Section 3.2.1 of the main paper (3.2.2) and our high-poly mesh upsampler presented

	AP	AP_{50}	AP_{75}	AP_M	AP_L	AR	AR_{50}	AR_{75}	AR_M	AR_L
DP - MBNet	53.54	91.66	58.29	54.75	53.65	64.09	95.68	75.39	54.89	64.73
SP - MBNet	54.08	93.01	59.41	55.94	54.32	64.46	96.03	76.10	56.24	65.03
DP - ResNet	57.31	93.06	65.60	57.72	57.49	67.51	96.30	80.29	57.87	68.17
SP - ResNet	58.87	93.05	68.34	60.10	58.90	68.73	96.26	82.08	60.14	69.33
DP - HRNet32	61.12	94.84	72.23	61.39	61.41	70.53	97.33	84.26	61.56	71.16
SP - HRNet32	61.24	94.63	72.61	61.68	61.54	70.52	97.10	84.53	61.77	71.13
DP - HRNet48	62.74	95.04	75.39	63.86	63.03	71.95	97.50	86.18	64.11	72.49
SP - HRNet48	**63.32**	**95.12**	**76.48**	**64.75**	**63.63**	**72.14**	**97.73**	**87.07**	**64.89**	**72.65**

(a) VertexPose (VP) vs DensePose (DP).

	AP	AP_{50}	AP_{75}	AP_M	AP_L	AR	AR_{50}	AR_{75}	AR_M	AR_L
Barycentric	**61.24**	94.63	**72.61**	**61.68**	**61.54**	**70.52**	97.10	**84.53**	**61.77**	**71.13**
Closest	59.69	**95.10**	70.08	61.39	60.01	68.80	**97.59**	83.24	61.49	69.31
Global Average	33.35	89.58	11.00	37.13	33.61	43.23	94.38	31.79	37.16	43.64

(b) UV aggregation strategy

Table 1. Analysis of VertexPose performance on the DensePose-COCO dataset. We evaluate the impact of the backbone choice and the UV decoding strategy.

Method	3DPW-OCC ↓			3DOH ↓	
	MPJPE	PAMPJPE	PVE	MPJPE	PAMPJPE
DOH [35]	-	72.2	-	-	58.5
CRMH [8]	-	78.9	-	-	-
VIBE [12]	-	65.9	-	-	-
SPIN [14]	95.6	60.8	121.6	104.3	68.3
HMR-EFT [9]	94.4	60.9	111.3	75.2	53.1
HybrIK [16]	90.8	58.8	111.9	40.4	31.2
PARE [13]	90.5	56.6	**107.9**	63.3	44.3
ROMP [29]	-	67.1	-	-	-
NIKI [17]	88.2	55.3	109.7	**38.9**	**29.2**
PLIKS [28]	**86.1**	53.2	-	51.5	39.3
MeshPose (HRNet32)	89.11	**51.81**	108.15	60.93	38.18

Table 2. Evaluation of the occlusion robustness of MeshPose compared to other state-of-the-art approaches in object-occluded benchmark datasets 3DPW-OCC and 3DOH.

in Section 3.2.3 of the main paper (3.2.3). Finally, in Subsection 3.3 we provide more details on our training strategy including datasets mixing, augmentations and scheduling.

3.1. Backbone architectures

Regarding our main backbone architecture of choice we employed the HRNet-32 model described in [31], as it is capable of producing a high-resolution feature map. Only the high-resolution, stride 4 output features of the last block are used. Since MeshPose is a multitasking system which outputs tensors of a large dimensionality, we have modified the architecture to have an output with more feature channels, without adding considerable overhead. More specifically, before the upsampling and the sum-based fusion of the feature maps from the last block, which have different stride and feature size, we project all of them to a fixed feature size of 256 instead of 32 that is used in the original HRNet-32 implementation. This modification only adds a small number of parameters, since it is applied only at the end of the backbone, but it removes the 32-channel bottleneck to accomodate the MeshPose tasks.

For the Resnet50 [6] and MobileNetV2 [27] variants we used dilated convolutions on their last block, which gives features with stride 16, and then we applied a decoder-net with separable-convolutions and skip-connections from the previous stages in order to produce the final feature map with stride 4. We found that this light-weight decoder-net has much less parameters (due to separable convolutions) compared to the fully deconvolutional layers that are usually employed in pose estimation [32].

3.2. Decoders

As explained in Section 3.2 of the main paper, we learn how to directly regress 3D vertex positions V_{reg}^{XYZ} for all vertices. In addition to the 3D positions, we also predict a visibility label $w \in [0, 1]$ for each vertex. The visibility w indicates whether we should rely on the VertexPose-based 2D position, V_{sp}^{XY} or fall back to the V_{reg}^{XY} value. A low w value implies that the corresponding vertex is occluded or out-of-crop which means that the VertexPose vertex is likely incorrect. The final MeshPose vertices are simply computed as a visibility-weighted average between both predictions: $V_{mp}^{XY} = V_{sp}^{XY} w + V_{reg}^{XY}(1-w)$. In this subsection, we detail how the regressed vertices V_{reg}^{XYZ} are obtained and how we supervise them together with their visibility predictions w.

	Method	2D COCO KeyPoints ↑ 80%						DensePose ↑	
		AP	AP_M	AP_L	AR	AR_M	AR_L	AP	AR
param	DaNet [34]	52.60	51.00	54.10	65.20	62.80	66.50	16.42	29.24
	HybrIK [16]	31.70	25.40	35.30	47.50	37.10	52.80	20.85	34.44
	PARE [13]	56.90	56.30	57.70	67.20	64.80	68.50	30.02	41.54
	CLIFF [18]	61.40	58.80	63.70	72.50	68.10	74.70	30.99	41.83
	PyMaf [33]	58.00	57.70	58.70	70.00	67.50	71.30	17.62	30.69
	LVD [4]	12.20	6.60	13.20	28.90	18.60	29.90	1.03	6.49
	NIKI [17]	48.50	44.80	51.10	61.50	55.70	64.50	25.11	37.39
n-param	Metro [19]	30.40	33.00	29.60	45.10	45.00	45.10	9.28	21.16
	Graphormer [20]	35.90	37.00	35.70	49.50	48.20	50.00	13.57	26.20
	FastMetro [2]	39.30	40.60	39.30	53.20	52.00	53.80	13.76	26.21
	PointHMR [11]	44.40	42.80	46.10	57.30	53.10	59.40	19.58	32.18
ours	MeshPose (HRNet32 [31])	**71.20**	**67.20**	**73.70**	**79.60**	**74.00**	**82.50**	**47.87**	**57.62**
	MeshPoseS (ResNet50 [6])	67.00	63.20	69.50	76.50	70.80	79.40	44.41	54.49
	MeshPoseXS (MBNet140 [27])	63.60	59.00	66.50	73.90	67.90	76.90	38.56	49.20

Table 3. Evaluation of 2D accuracy in COCO-DensePose for both 2D keypoint predictions and DensePose regression.

3.2.1 Coordinate and Visibility Regression

We draw inspiration from [28] and extract the last tensor $\mathbf{F} \in \mathbb{R}^{C,\frac{H}{4},\frac{W}{4}}$ of our CNN backbone with $C = 256$. As explained in the main paper, we use 1D convolutions to generate three feature tensors $\mathbf{P} = \{\mathbf{P}^x, \mathbf{P}^y, \mathbf{P}^z\} \in \mathbb{R}^{V\times 64}$, one for each dimension X, Y, Z and an extra visiblity weight $\mathbf{w} \in \mathbb{R}^V$, with V the number of vertices.

$$\mathbf{P}^x = f_x^{1D}(\psi_x(\text{avg}^{x,y}(\mathbf{F}))) \quad (3)$$

$$\mathbf{P}^y = f_y^{1D}(\psi_y(\text{avg}^{x,y}(\mathbf{F}))) \quad (4)$$

$$\mathbf{P}^z = f_z^{1D}(\psi_z(\text{avg}^{x,y}(\mathbf{F}))) \quad (5)$$

and

$$\mathbf{w} = \sigma(\text{avg}^z(f_z^{1D}(\psi_z(\text{avg}^{x,y}(\mathbf{F}))))) \quad (6)$$

We first apply an average pooling $\text{avg}^{x,y}$ across the spatial dimensions (x, y). Then, we apply two successive 1D convolutions ψ_i and f_i^{1D} along the indexed dimension i. The first convolution ψ_i expands the dimension from $C \times 1$ to $C \times C'$ then f_i^{1D} transforms the feature tensors to $V \times C'$ dimension. Finally, for the visibility weight $\mathbf{w}$, we average across the channel dimension avg^z, then apply a sigmoid σ activation function to map the values between 0 and 1.

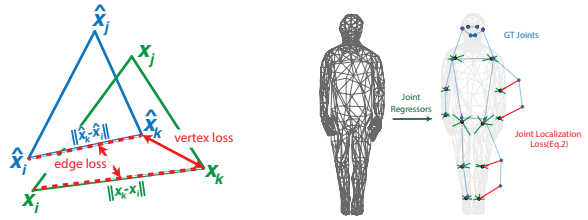
To obtain the 3D vertex positions from the learnt features $\{\mathbf{P}^x, \mathbf{P}^y, \mathbf{P}^z\}$, we apply argsoftmax over the $C' = 64$ channels. The resulting value of the argsoftmax will thus be between 0 and 64 and thus needs to be mapped to pixel positions. We map the range $[0, 64]$ to $[-W, 2W]$ for X, $[-H, 3H]$ for Y and $[-2W, 2W]$ for Z. The top left pixel on the image corresponds to pixel $[0, 0]$. The new range expands beyond the image boundary to predict out-of-crop vertices. We note that to accommodate for selfie-like images, we consider a larger range for Y ($[-H, 3H]$): this allows us to be able to predict the position of leg vertices that will often lie significantly below the crop.

3.2.2 Custom Silhouette Rendering

The learnt visibility weight $\mathbf{w}$ is partly supervised by the 3D localization and the edge losses (see Section 3.2.4 and Figure 5). We also use a binary-cross entropy loss L_W using the supervision of the mesh pseudo-ground truth. However, we also want to leverage the ground truth DensePose segmentation masks which provide a suitable signal to learn visibility with weak supervision. To achieve that, we introduce a novel silhouette rendering module by modifying the soft rasterization method of SoftRas [22] so that it incorporates the predicted vertex visibilities. More specifically, for each pixel i, we compute the silhouette I_s by:

$$I_s^i = \mathcal{A}_O(\{\mathcal{D}_j\}) = 1 - \prod_j (1 - w_j^i \mathcal{D}_j^i) \quad (7)$$

Here, as in [22], $\mathcal{D}_j$ denotes the influence of triangle f_j at pixel i and mostly depends on the distance of triangle j to pixel i. Contrary to [22], we also multiply the influence $\mathcal{D}_j$ by the visibility weight w_j^i of face j at pixel i. w_j^i is simply computed as the linear interpolation between the visibility weights of the three vertices of face j at pixel i. The newly added coefficient w_j^i modifies the initial rendering pipeline from [22] to ignore faces with low visibility weight.



(a) Edge and Vertex Loss (b) Joint Localization Loss

Figure 5. Regularization losses used for 3D mesh supervision.

We use two losses to supervise our rendered silhouette I_s. First, we use a simple L_2 loss between our rendered I_s and the corresponding ground truth from the DensePose annotation I_s^{DP}: $\mathcal{L}_{I_s}^1 = ||I_s - I_s^{DP}||_2$. The second loss we introduce is inspired from the boundary loss from [10]:

$$\mathcal{L}_{I_s}^2 = -\sum_{x,y} D(x,y) I_s(x,y) \tag{8}$$

with D the level set of the DensePose ground-truth segmentation boundary. More specifically, $D(x,y)$ is equal to the distance d between pixel (x,y) and the boundary ∂I_s^{DP} of the ground-truth segmentation mask - except for pixels (x,y) outside of the ground truth mask.

$$D(x,y) \begin{cases} d((x,y), \partial I_s^{DP}) & \text{if } I_s^{DP}(x,y) = 1 \\ 0 & \text{else.} \end{cases} \tag{9}$$

This loss only penalizes "too small" silhouettes that are not expanding over the whole ground truth mask. The final silhouette loss is the combination of the two introduced losses:

$$\mathcal{L}_{I_s} = 100 \cdot \mathcal{L}_{I_s}^1 + \mathcal{L}_{I_s}^2 \tag{10}$$

To also learn the visibility weight $\mathbf{w}$ for out-of-crop vertices, we render the silhouette on a larger crop and pad the ground truth mask accordingly. This helps the pipeline to assign low visibility weights to vertices that are not in the original crop.

3.2.3 High Poly Mesh Upsampler

As mentioned in the main paper, we predict the vertices of a low-polygon approximation (518 vertices) of our high resolution body mesh (6890 vertices). As pointed out in [19], working on a lower-resolution mesh both reduces memory usage while improving training stability by limiting correlated vertices. To generate this approximation, we used a custom-defined mesh to reduce the number of vertices in undesirable high curvature areas (hands, fingers).

To obtain the high-poly mesh from the low-poly mesh , which is the output of the network, we trained a multilayer perceptron upsampler following the ideas of [15, 19]. We employed two affine layers to progressively upsample the low-poly mesh to the high-poly variant (with an intermediate representation of 1723 vertices as in [19]). We applied an L1 loss between the ground-truth and the upsampled vertices at each stage, as well as between the GT joints and the joints that are estimated using a landmarker on the upsampled version. We trained the upsampler independently from the main network using the high-poly mesh pseudo GT annotations [25] from only COCO, Human 3.6M and MPI-INF-3DHP datasets to accurately cover a large pose distribution. Our method is thus agnostic to the high poly template and only the upsampler would need to be tuned for a different template to be used.

An additional improvement to the training of the upsampler, compared to previous approaches, is the addition of synthetic noise during the training to make the upsampler more robust to noise or to small errors that may be in the low-poly mesh. More specifically, with probability 0.5 we randomly added to 25% of the vertices spikes equal to 15% of their vertex position values (after applying root alignment to the mesh). In the other case, with probability 0.5 we added gaussian noise with standard deviation 5% to each vertex position.

For comparison, a nearest-point-on-triangle, non-learned upsampler produces slightly worse results (1mm difference for 3DPW MVE, 2 for DP AP).

3.3. Training

We used a combination of datasets and training signals in our model training. More specifically, we used a mixture of COCO and Human Mesh Reconstruction (HMR) datasets. First, we enriched the COCO (train2017) dataset with mesh pseudo-annotations and their DensePose annotations. For the HMR datasets, we combined Human3.6M [7], MPI-INF-3DHP [24] and 3DPW [30] training sets. For each dataset, we proceeded to image augmentation with (i) flipping , (ii) rotation (between $-45°$ and $45°$), (iii) scale (between 0.75 and 1.25) and (iv) color augmentations. We followed a $\sim 40/60$ split between COCO and HMR datasets. To achieve that, we upsampled each dataset by an associated factor to control the dataset mixture: 5 for COCO, 4 for Human3.6M, 1 for MPI-INF-3DHP and 2 for 3DPW. We trained with the Adam optimizer for 200 epochs with a mini-batch size of 96 and a learning rate of 1×10^{-3}, which is reduced by a factor of 10 after 100 epochs. The weights of the backbone of our systems are initialized with pretrained 2D pose estimation networks. We used Linux machines with 4 Nvidia Tesla V100 GPUs (16GB) for all of our experiments.

We aggregated the different losses based on the following weighted linear combination:

$$\begin{aligned} \mathcal{L} = \mathcal{L}_{BL} + L_{\text{consistency}} + 10 \cdot L_W + 0.1 \cdot \mathcal{L}_V \\ + \mathcal{L}_E + 0.1 \cdot \mathcal{L}_N + \mathcal{L}_J + \mathcal{L}_{I_s} \end{aligned} \tag{11}$$

with $\mathcal{L}_{BL}$ the barycentric cross-entropy loss (Section 3.1.2), $L_{\text{consistency}}$ the UV consistency loss (Section 3.1.2), L_W the visibility binary cross-entropy loss (Section 3.2.2 - Supplementary), $(\mathcal{L}_V, \mathcal{L}_E, \mathcal{L}_N)$ the vertex localization, the edge, the normal and the joint localization losses (Section 3.2.4) and $\mathcal{L}_{I_s}$ the silhouette loss (Section 3.2.2 - Supplementary).

We note that $(\mathcal{L}_V, \mathcal{L}_N)$ are given smaller weights to downscale the importance of the pseudo-ground truth. Our system however requires higher weight for $\mathcal{L}_E$ to remove mesh artefacts.

References

[1] Pexels. https://www.pexels.com/. Accessed: 2024-03-29. 2

[2] Junhyeong Cho, Kim Youwang, and Tae-Hyun Oh. Cross-attention of disentangled modalities for 3d human mesh recovery with transformers. In *ECCV*, 2022. 6

[3] Core ML. https://developer.apple.com/documentation/coreml. 4

[4] Enric Corona, Gerard Pons-Moll, Guillem Alenyà, and Francesc Moreno-Noguer. Learned vertex descent: A new direction for 3d human model fitting. *CVPR*, 2022. 4, 6

[5] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *CVPR*, 2018. 1, 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4, 5, 6

[7] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE TPAMI*, 36(7):1325–1339, 2013. 1, 7

[8] Wen Jiang, Nikos Kolotouros, Georgios Pavlakos, Xiaowei Zhou, and Kostas Daniilidis. Coherent reconstruction of multiple humans from a single image. In *CVPR*, 2020. 5

[9] Hanbyul Joo, Natalia Neverova, and Andrea Vedaldi. Exemplar fine-tuning for 3d human pose fitting towards in-the-wild 3d human pose estimation. In *3DV*, 2020. 5

[10] Hoel Kervadec, Jihene Bouchtiba, Christian Desrosiers, Eric Granger, Jose Dolz, and Ismail Ben Ayed. Boundary loss for highly unbalanced segmentation. In *Proceedings of The 2nd International Conference on Medical Imaging with Deep Learning*, pages 285–296. PMLR, 2019. 7

[11] Jeonghwan Kim, Mi-Gyeong Gwon, Hyunwoo Park, Hyukmin Kwon, Gi-Mun Um, and Wonjun Kim. Sampling is Matter: Point-guided 3d human mesh reconstruction. In *CVPR*, 2023. 6

[12] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *CVPR*, 2020. 5

[13] Muhammed Kocabas, Chun-Hao P. Huang, Otmar Hilliges, and Michael J. Black. PARE: Part attention regressor for 3D human body estimation. In *ICCV*, 2021. 5, 6

[14] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *ICCV*, 2019. 5

[15] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *CVPR*, 2019. 7

[16] Jiefeng Li, Chao Xu, Zhicun Chen, Siyuan Bian, Lixin Yang, and Cewu Lu. Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation. In *CVPR*, 2021. 5, 6

[17] Jiefeng Li, Siyuan Bian, Qi Liu, Jiasheng Tang, Fan Wang, and Cewu Lu. NIKI: Neural inverse kinematics with invertible neural networks for 3d human pose and shape estimation. In *CVPR*, 2023. 5, 6

[18] Zhihao Li, Jianzhuang Liu, Zhensong Zhang, Songcen Xu, and Youliang Yan. Cliff: Carrying location information in full frames into human pose and shape estimation. In *ECCV*, 2022. 6

[19] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers. In *CVPR*, 2021. 6, 7

[20] Kevin Lin, Lijuan Wang, and Zicheng Liu. Mesh graphormer. In *ICCV*, 2021. 6

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 1

[22] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. *ICCV*, 2019. 6

[23] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *TOG*, 34(6):1–16, 2015. 1

[24] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *3DV*, 2017. 7

[25] Gyeongsik Moon, Hongsuk Choi, and Kyoung Mu Lee. Neuralannot: Neural annotator for 3d human mesh training sets. In *CVPRW*, 2022. 7

[26] Open Neural Network Exchange (ONNX). https://github.com/onnx/onnx?tab=readme-ov-file. 4

[27] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 4, 5, 6

[28] Karthik Shetty, Annette Birkhold, Srikrishna Jaganathan, Norbert Strobel, Markus Kowarschik, Andreas Maier, and Bernhard Egger. Pliks: A pseudo-linear inverse kinematic solver for 3d human body estimation. In *CVPR*, 2023. 5, 6

[29] Yu Sun, Qian Bao, Wu Liu, Yili Fu, Michael J Black, and Tao Mei. Monocular, one-stage, regression of multiple 3d people. In *ICCV*, 2021. 5

[30] Timo von Marcard, Roberto Henschel, Michael J Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In *ECCV*, 2018. 1, 4, 7

[31] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui

Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE TPAMI*, 43(10): 3349–3364, 2020. 4, 5, 6

[32] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *ECCV*, 2018. 5

[33] Hongwen Zhang, Yating Tian, Xinchi Zhou, Wanli Ouyang, Yebin Liu, Limin Wang, and Zhenan Sun. Pymaf: 3d human pose and shape regression with pyramidal mesh alignment feedback loop. In *ICCV*, 2021. 6

[34] Hongwen Zhang, Jie Cao, Guo Lu, Wanli Ouyang, and Zhenan Sun. Learning 3d human shape and pose from dense body parts. *IEEE TPAMI*, 44(5):2610–2627, 2022. 6

[35] Tianshu Zhang, Buzhen Huang, and Yangang Wang. Object-occluded human shape and pose estimation from a single color image. In *CVPR*, 2020. 1, 2, 4, 5