A Lightweight Graph Transformer Network for Human Mesh Reconstruction from 2D Human Pose

Ce Zheng
University of Central Florida
USA
cezheng@knights.ucf.edu

Matias Mendieta
University of Central Florida
USA
mendieta@knights.ucf.edu

Pu Wang
University of North Carolina at Charlotte
USA
pu.wang@uncc.edu

Aidong Lu
University of North Carolina at Charlotte
USA
aidong.lu@uncc.edu

Chen Chen
University of Central Florida
USA
chen.chen@crcv.ucf.edu

ABSTRACT

Existing deep learning-based human mesh reconstruction approaches have a tendency to build larger networks to achieve higher accuracy. Computational complexity and model size are often neglected, despite being key characteristics for practical use of human mesh reconstruction models (e.g. virtual try-on systems). In this paper, we present GTRS, a lightweight pose-based method that can reconstruct human mesh from 2D human pose. We propose a pose analysis module that uses graph transformers to exploit structured and implicit joint correlations, and a mesh regression module that combines the extracted pose feature with the mesh template to reconstruct the final human mesh. We demonstrate the efficiency and generalization of GTRS by extensive evaluations on the Human3.6M and 3DPW datasets. In particular, GTRS achieves better accuracy than the SOTA pose-based method Pose2Mesh while only using 10.2% of the parameters (Params) and 2.5% of the FLOPs on the challenging in-the-wild 3DPW dataset. Code is available at https://github.com/zczcwh/GTRS

ACM Reference Format:
Ce Zheng, Matias Mendieta, Pu Wang, Aidong Lu, and Chen Chen. 2022. A Lightweight Graph Transformer Network for Human Mesh Reconstruction from 2D Human Pose. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (ACM Multimedia '2022).* ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Analyzing and simulating humans from images is an essential task for computer vision. With the blooming of deep learning methods, human pose estimation (HPE) has been studied extensively and rapid progress has been made. In recent years, research in this domain has progressed beyond the estimation of 2D or 3D poses [45][42][27] with a basic keypoint structure, and the study of reconstructing the entire 3D mesh from a single image has attracted much interest. Mesh representation, which can provide rich human body information and have a better visualization, is more welcomed by real-world applications such as gaming, human-computer interaction, and virtual reality (VR). However, human mesh reconstruction from a single image is a challenging task due to depth ambiguity, occlusion, and complex human body articulation.

Two general approaches exist in the literature for performing mesh reconstruction. One is the direct image-based method, where the pipeline is trained end-to-end from input image to output mesh. The second is to employ an off-the-shelf 2D pose detector as the front end, and design a mesh reconstruction model using 2D poses as input. Most recent progress has been made in the first category, achieving promising performance. However, the performance gain has come at the cost of ever increasing computational requirements and complex models (for instance, METRO [22] requires 229M Params and 56.6G FLOPs). In real-world applications such as human-computer interaction, animated avatar, and VR gaming, the human mesh reconstruction task needs to be efficient and deployable on resource-constrained platforms like VR headsets.

While less studied, pose-based methods are alternative solutions for human mesh recovery with a few advantages for such applications. First, pose-based methods provide a modular design that can easily be incorporated with any off-the-shelf 2D pose detectors. With speed as the primary goal, fast pose detectors (e.g. [35, 38, 44]) can be deployed on a mobile device in real-time with impressive performance. Second, the input to pose-based methods (that is, the detected 2D pose) is extremely sparse data with the size of $J \times 2$, where J is the number of joints. Compared to the image input, it gives more flexibility to design a lightweight mesh reconstruction network to achieve computational and memory efficiency with competitive performance. Nonetheless, the existing methods (including the state-of-the-art pose-based method Pose2Mesh [5]) still incur substantial computational and memory overhead. The efficient design of the model is crucial for practical use, but has been almost entirely ignored in the literature.

To bridge this gap, we propose a **G**raph **T**ransformer network for human mesh **R**econ**S**truction from 2D human pose (GTRS), which

ACM Multimedia '2022, Oct 10–14, 2022, lisbon, portugal.

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

is **the first method** focusing on the efficiency. GTRS is a pose-based method designed to fully exploit joint correlations for pose and mesh feature representation while minimizing computational complexity and model size. The operational blocks of GTRS are designed with an intentional combination of graph neural networks and transformer operations. Recently, graph convolutional networks (GCNs) have shown promising advances in 3D HPE and mesh reconstruction tasks [62]. Human pose data is naturally formulated as a graph, and GCNs can extract useful information with relatively little compute and parameters. Therefore, we harness GCNs to form strong representations from these inherent structural priors on the front-end of GTRS block operations. After this, these representations are further refined with lightweight transformer structure to powerfully capture global dependencies via its self-attention mechanism. Thus, these combined operations form our designed graph transformer blocks, which we employ in a parallel fashion for comprehensively exploring human kinematic information from the 2D pose and modeling joint correlations in a lightweight manner.

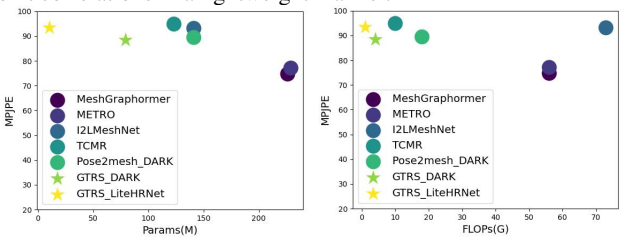


Figure 1: The trade-off between accuracy (MPJPE ↓) and model Params/FLOPs. All methods are evaluated on 3DPW dataset. GTRS (Ours) and Pose2Mesh [5] are pose-based methods. The reported Params/FLOPs include the corresponding front-end 2D pose detector (DARK [54] or LiteHRNet [53]). Others are image-based methods.

As an extremely lightweight pose-based method, GTRS uses only 7.9M parameters (Params) and 0.19G floating-point operations (FLOPs) without considering the front-end 2D pose detector. Compared to SOTA pose-based method Pose2Mesh [5], GTRS achieves better results while only requiring 10.2% of the Params and 2.5% of the FLOPs. When also considering the front-end 2D pose detector, GTRS also shows a significant reduction in Params and FLOPs compared to image-based methods in Fig. 1 (e.g. 6.9% Params and 1.2% FLOPs compared to I2LMeshNet [34]). More discussions are provided in Sec. 4.3.

Our contributions are summarized as follows:

- Observing that existing methods mainly pursue higher accuracy while ignoring computational and memory cost, we present a lightweight pose-based method, GTRS, for efficient human mesh reconstruction from the 2D pose. We hope our work can inspire more research on the efficiency of human mesh reconstruction.
- We introduce our pose analysis module with a parallel design to facilitate improved utilization of human kinematic information. Within this module, we propose our graph transformer blocks with fixed and learnable adjacency matrices to simultaneously explore diverse structured and implicit human joint correlations.
- GTRS achieves competitive results compared to previous methods with much fewer parameters and less computational cost on Human3.6M and 3DPW datasets.

2 RELATED WORK

Human Mesh Reconstruction: Recovering human mesh from images is a challenging task that has attracted much attention in recent years. Without requiring additional devices such as depth sensors or inertial measurement units, Human Mesh Reconstruction (HMR) from images makes it more efficient and convenient. The majority of previous works [19, 22, 23] utilize parametric human model such as SMPL [30], ADAM [48], STAR [37] to reconstruct human mesh by training a network to regress model parameters.

As one of the most popular volumetric models, the SMPL [30] model has been widely used in HMR, e.g., [2, 14, 15, 52]. Pavlakos et al. [41], and Omran et al. [36] regress SMPL parameters to reconstruct 3D human mesh. SPIN [19] revisits optimization approaches within the neural networks that initializes an iterative optimization process (SMPLify). Instead of predicting SMPL parameters, Zhu et al. [60] combine the SMPL model with a hierarchical mesh deformation framework to enhance the flexibility of free-form 3D deformation. Kocabas et al. [18] include the large-scale motion capture dataset AMASS [31] for adversarial training of their SMPL-based method named VIBE (Video Inference for Body Pose and Shape Estimation).

Compared to previous methods that recover human mesh directly from images, [5] estimates SMPL parameters from predicted 2D poses and achieves impressive performance. By applying off-the-shelf 2D pose detectors such as AlphaPose [8] and HRNet [46], the well-estimated 2D pose can be obtained. Then, a CNN-based PoseNet and MeshNet are proposed to exploit the human mesh topology to recover human mesh based on the input 2D pose. GTRS also follows this pose-based pipeline to reconstruct human mesh.

Graph Convolution Networks: Recently, graph convolution networks (GCNs) have been widely adopted in 3D human pose estimation (3D HPE) [6, 26, 56, 57, 62] because of the intuitive modeling of human joints as a graph structure and potential ability to better capture human kinematics. Following this trend, GCNs have also gained much attention in human mesh reconstruction [5, 20, 24]. Kolotouros et al. [20] regress the locations of the SMPL mesh vertices using a GCN architecture. Pose2Mesh [5] employs a GCN to regress SMPL parameters from estimated 2D and 3D pose.

Vision Transformer: Transformer architecture is developing rapidly in the field of computer vision. Recent works have demonstrated the powerful global representation ability of transformer attention mechanism in various vision tasks such as object detection [3, 61], image classification [7, 28], segmentation [59], human pose estimation [57, 58], etc. Lin et al. [22] combine CNNs with transformer networks in their method, named METRO, to regress mesh vertices from a single image.

MeshGraphormer [23] is a close related work, which also uses GCN with transformer architecture. However, it is *an image-based method* that injects GCN with fixed adjacency matrix into the transformer block between multi-head attention and multilayer perceptron (MLP). *As a pose-based method*, our GTRS utilizes GCNs to model features with prior knowledge, then applies transformers to further explore global dependencies. Moreover, GTRS adopts a paralleled design which enables different graph transformer blocks to explore diverse structured and implicit human kinematic information by using fixed and learnable adjacency metrices. Compared

to MeshGraphormer [23] that requires 226.5M Params and 56.6G FLOPs, GTRS shows significant computational and memory cost reduction. GTRS is more friendly to deploy on mobile devices since it only requires 9.7M Params and 0.89G FLOPs (4.29% and 1.57% of MeshGraphormer).

3 METHODOLOGY

3.1 Baseline

In order to achieve model efficiency, an intuitive solution is to utilize the existing lightweight architecture such as MobileNetV2 [43] for the 3D human mesh recovery from the 2D pose input. We refer to this simple design as our baseline. However, these CNN-based lightweight architectures are designed to process an image-like input (with the shape of $[C, H, W]$ where C is channel, H is the height, and W is the width). First we embed the 2D pose input of J joints given by $X_{in} \in \mathbb{R}^{J\times 2}$ to $X_{baseline} \in \mathbb{R}^{C\times H\times W}$, then apply MobileNetV2 [43] to model mesh features. Finally, the output mesh parameter $Y \in \mathbb{R}^{6890\times 3}$ can be estimated after feature embedding. However, CNNs do not provide a natural modeling of the graph-like 2D pose input (sparse and meaningfully structured), leaving some representational strength and efficiency on the table.

In light of the limitations of the baseline, we present our proposed GTRS architecture and elaborate the design components in the following.

3.2 Overview of GTRS

The vision transformer architecture is designed to capture the global dependencies cross all patches via self-attention mechanism given the input size of $[C, D]$ where C is the number of patches and D is the embedding dimension. Given the 2D pose input $X_{in} \in \mathbb{R}^{J\times 2}$, the joint correlations can be exploited when modeling human pose and mesh features through transformer architecture. Therefore, we design a lightweight transformer architecture which is more suitable for modeling 3D human pose and mesh given the 2D pose input rather than a lightweight CNN architecture. The results in Section 4.3 have proved this claim.

The overall architecture of GTRS is illustrated in Fig. 2. The input is the estimated 2D pose obtained by off-the-shelf 2D pose detector such as HRNet [45], which can be denoted as $X_{in} \in \mathbb{R}^{J\times 2}$, where J is the number of joints. A feature embedding layer embeds input $X_{in} \in \mathbb{R}^{J\times 2}$ to $X_{pose} \in \mathbb{R}^{J\times D}$ with a high feature dimension D. Then, a Pose Analysis Module (PAM) returns modelled feature $X^{'}_{pose} \in \mathbb{R}^{J\times D}$. Next, the mesh template $M_{temp} \in \mathbb{R}^{6890\times 3}$ (from [19]) is used to provide initial human mesh information, which is embedded to mesh template feature $X_{temp} \in \mathbb{R}^{T\times D}$ where T is the channel number. Then, we feed $X^{'}_{pose}$ and X_{temp} to Mesh Regression Module (MRM), and the output would be $X_{out} \in \mathbb{R}^{(J+T)\times D}$. Finally, the estimated mesh parameter $Y \in \mathbb{R}^{6890\times 3}$ can be obtained after the regression head.

3.3 Preliminaries of GTRS

GCN: The GCN was introduced by [17]. A graph is defined as $\mathcal{G} = \{v, \varepsilon\}$, where v is a set of N nodes and ε is a set of edges. We use GCN to model 2D pose feature, for the input feature $X \in \mathbb{R}^{J\times D}$, where J is the number of joints and D is the dimension of input feature. Given the adjacency matrix $A \in \mathbb{R}^{J\times J}$ based on the joints connectivity, the output $X' \in \mathbb{R}^{J\times D'}$ of one GCN layer can be represented as:

$$X' = \sigma(AXW) \tag{1}$$

where $\sigma(\cdot)$ is the activation function for network non-linearity, and $W \in \mathbb{R}^{D\times D'}$ is the learnable weight matrix which changes the feature dimension from D to D'. We use the Gaussian Error Linear Unit (GELU) [10] as activation function in this work.

Transformer: Multi-Head Self-Attention Layer (MHA) is the core function of the transformer blocks, which was proposed by Vaswani et al. [49]. The input $X \in \mathbb{R}^{J\times D}$ is first mapped to three matrices: query matrix Q, key matrix K and value matrix V by three linear transformation:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V. \tag{2}$$

where W_Q, W_K and $W_V \in \mathbb{R}^{D\times D}$.

The scaled dot product attention can be described as the following mapping function:

$$\text{Attention}(Q, K, V) = \text{Softmax}(QK^\top/\sqrt{d})V. \tag{3}$$

where $\frac{1}{\sqrt{d}}$ is the scaling factor for appropriate normalization to prevent extremely small gradients.

Next, the MHA utilizes multiple heads to model the information jointly from various representation subspaces with different positions. Each head applies scaled dot-product attention in parallel. The MSA output will be the concatenation of h attention head outputs.

$$\text{MSA}(Q, K, V) = \text{Concat}(H_1, H_2, \ldots, H_h)W_{out} \tag{4}$$

$$\text{where} \quad H_i = \text{Attention}(Q_i, K_i, V_i), i \in [1, ..., h] \tag{5}$$

W_{out} is a linear projection $\in \mathbb{R}^{D\times D}$.

3.4 Pose Analysis Module in GTRS

In the PAM, we utilize graph transformers to improve the structured and implicit correlations based on the human kinematic information. We follow [62] to build our GCN blocks. Then a transformer block is followed to model global dependencies. Our graph transformer block is illustrated in Fig. 3 (b). Different from previous transformer architecture that stacks multiple transformer encoders, we form graph transformer blocks parallelly as shown in Fig. 3 (a). The transformer embedding dimension is set to be a small number to make the network lightweight. Because GCNs maintain a strong relationship with the input graph structure, we begin each block with such operations to inject structural priors before performing the transformer's self-attention. Based on the human kinematic configuration illustrated in Fig. 4 (a), the actual adjacency matrix can be obtained as shown in Fig. 4 (b). Among these paralleled graph transformer blocks, only one block utilizes the actual adjacency matrix, which means this adjacency matrix is fixed and would not be updated. This graph transformer block is maintained to model pose features with structured human kinematic information using the fixed adjacency matrix. The rest of the graph transformer blocks are responsible to capture implicit correlations from the training data by applying the learnable adjacency matrices. The different patterns of correlations can be discovered through learnable adjacency matrices as shown in Fig. 4 (c). The parallel design allows for various structural biases to

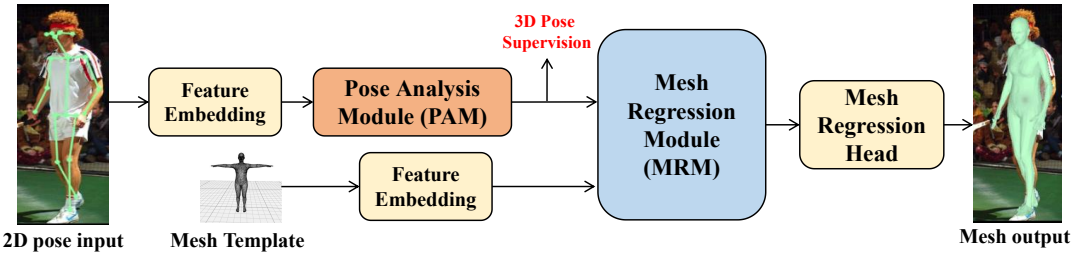


Figure 2: Overview of the proposed GTRS architecture. Given the image, 2D human pose is first detected by an off-the-shelf 2D pose detector. Then, the Pose Analysis Module outputs the pose feature and intermediate 3D pose which is supervised by the ground truth 3D pose. Next, the pose feature is modeled with template mesh feature in the Mesh Regression Module. Finally, a regression head will output human mesh parameters for reconstruction. The mesh template is provided by [19] and the mesh template figure is from [22]. A more detailed illustration of the GTRS architecture is provided in the appendix.

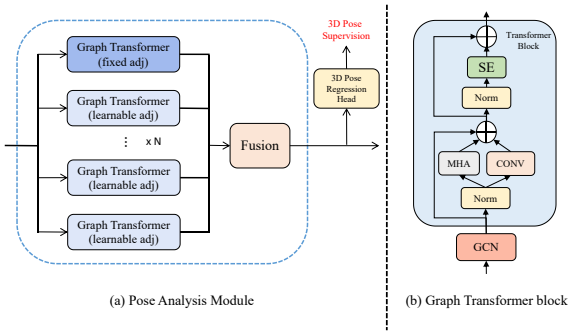


Figure 3: The Pose Analysis Module architecture is shown in (a), which consists of paralleled graph transformer blocks. 'fixed adj' means a fixed adjacency matrix is used in this graph transformer block, and 'learnable adj' means a learnable adjacency matrix is used. (b) is the architecture of one graph transformer block.

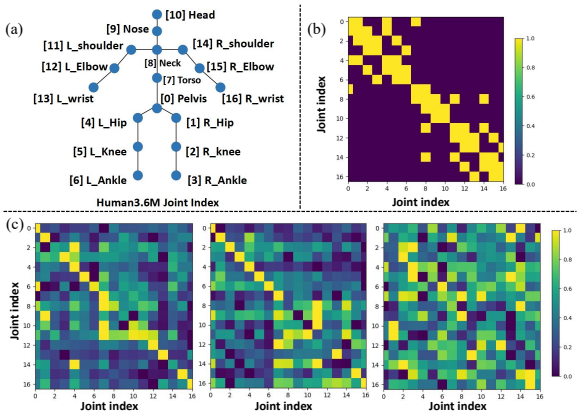


Figure 4: The normalized adjacency matrices used in PAM. (a) is the joint index of Human3.6M dataset. (b) is a fixed adjacency matrix directly from the joint connectivity to model structured correlations. (c) shows learnable adjacency matrices that learned from training data to capture implicit correlations. In PAM, we apply a fixed adjacency matrix and multiple learnable adjacency matrices with a paralleled design to allow the network to explore a diverse set of structured and implicit correlations.

be applied simultaneously. Not only the kinematic correlations can be provided by the fixed adjacency matrix (based on the body joints structure), but also the unexpected correlations beyond our prior knowledge can be discovered by the learnable adjacency matrices in PAM. Next, a fusion block, which is a convolutional layer, will fuse all paralleled features together to a feature that maintains the same size as the input.

Our transformer encoder layer is different from the original transformer encoder in [7]. The structure of our transformer encoder is illustrated in Fig. 3 (b). First, we add one convolutional branch parallel to the MHA branch, which is a point-wise convolution with the 1×1 convolutional filters as described in [11]. The reason we use this pointwise convolution is to create linear combinations of the input channels (J joint channels) while maintaining a low computational cost. Then, instead of the MLP layer, we use an SE block [12] which is also computationally lightweight. The SE block is designed to recalibrate channel-wise feature responses by modeling channel interdependencies. The output of our transformer blocks given the input $X_{in} \in \mathbb{R}^{J \times D}$ can be represented as follows:

$$X' = \text{MSA}(\text{Norm}(X_{in})) + \text{CONV}(\text{Norm}(X_{in})) + X_{in} \quad (6)$$

$$X_{out} = \text{SE}(\text{Norm}(X')) + X' \quad (7)$$

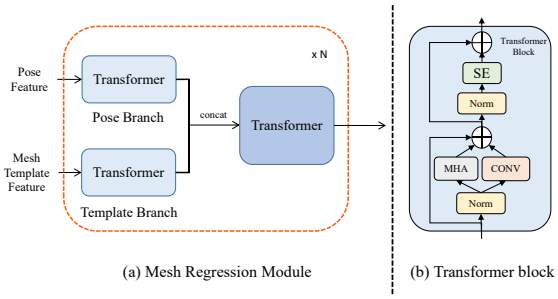
where $\text{CONV}(\cdot)$ is the convolutional block, $\text{Norm}(\cdot)$ is the normalization operator, and $\text{SE}(\cdot)$ denotes the SE block.

Besides the output pose features X_{out}, a 3D pose regression head (implemented as an MLP) is used to output a 3D pose prediction $X_{3DPose} \in \mathbb{R}^{J \times 3}$. This supervision on the 3D pose ensures the PAM can exploit pose features well even if the detected 2D pose is noisy (such as missing joints).

3.5 Mesh Regression Module in GTRS

The computational complexity of our transformer structure (as in our PAM) is $O(nD^2)$ for the input size of $X \in \mathbb{R}^{n \times D}$ and n is much less than D. To maintain a lightweight network, the transformer blocks with small embedding dimension D (consistent with the embedding dimension in PAM) are used in MRM. Considering that small embedding dimension D may not have enough representation ability, we introduce the mesh template which provides rich human mesh information for a better regression. The effectiveness of adding the mesh template has been verified in Sec. 4.4. We embed the original mesh template to the mesh template feature $X_{temp} \in \mathbb{R}^{T \times D}$ to reduce the computational cost.

Due to different scales of the pose feature $X'_{pose} \in \mathbb{R}^{J \times D}$ and the mesh template feature $X_{temp} \in \mathbb{R}^{T \times D}$, we design a dual-branch block structure where two separate transformers are applied first



(a) Mesh Regression Module (b) Transformer block

Figure 5: The Mesh Regression Module architecture is shown in (a), which is consists of dual-branch transformer blocks. (b) is the architecture of one transformer block.

to model these two features and then fuse information by a fusion transformer. The output is $X_{fusion} \in \mathbb{R}^{(J+T)\times D}$, which can be split to $X_{B\_pose} \in \mathbb{R}^{J\times D}$ and $X_{B\_temp} \in \mathbb{R}^{T\times D}$ as the input of next block.

Finally, a regression head MLP will upsample the feature $X_{out} \in \mathbb{R}^{(J+T)\times D}$ to the final mesh output.

3.6 Loss functions

For the PAM, we add a linear layer to regress the intermediate 3D human pose (the final 3D human pose is obtained from the final mesh) to get a better pose feature. The pose feature is fed into the MRM. To train the PAM, we apply an $L1$ distance loss between the predicted indeterminate 3D pose $J_{3D\_int} \in \mathbb{R}^{K\times 3}$ and the ground truth 3D pose $\hat{J}_{3D} \in \mathbb{R}^{K\times 3}$, where K is the number of joints. The indeterminate 3D joint loss is defined as follows:

$$\mathcal{L}_{J\_int} = \frac{1}{K}\sum_{i=1}^{K} \|J_{3D\_int} - \hat{J}_{3D}\|_1, \tag{8}$$

After pretraining the PAM, we train the entire pipeline includes the PAM and the MRM. The following loss functions are used.

Mesh Vertex Loss: We use an $L1$ distance loss between the predicted 3D mesh vertices coordinates $V_{3D} \in \mathbb{R}^{M\times 3}$ and the ground truth 3D mesh vertices coordinates $\hat{V}_{3D} \in \mathbb{R}^{M\times 3}$, where M is the number of vertices. The mesh vertex loss is defined as follows:

$$\mathcal{L}_{Vertex} = \frac{1}{M}\sum_{i=1}^{M} \|V_{3D} - \hat{V}_{3D}\|_1. \tag{9}$$

3D Joint Coordinate Loss: After estimating 3D mesh vertices coordinates $V_{3D} \in \mathbb{R}^{M\times 3}$, we use the model defined joint regression matrix $R \in \mathbb{R}^{K\times M}$ to calculate 3D pose based on the estimated mesh, where K is the number of joints and M is the number of vertices. We apply an $L1$ loss between the regressed 3D pose from estimated mesh RV and the ground truth 3D pose $\hat{J}_{3D} \in \mathbb{R}^{K\times 3}$.

$$\mathcal{L}_{Joint} = \frac{1}{K}\sum_{i=1}^{K} \|RV - \hat{J}_{3D}\|_1 \tag{10}$$

Surface Normal Loss: Following [5, 51], we supervise normal vectors of the estimated mesh surface with the ground truth unit normal vector. We apply this surface normal loss to improve surface smoothness and local detail as in [51]. The surface normal loss is defined as follows:

$$\mathcal{L}_{Normal} = \sum_{f}\sum_{\{i,j\}\subset f} \left| \langle \frac{v_i - v_j}{\|v_i - v_j\|_2}, n_f^* \rangle \right|, \tag{11}$$

where f denotes a triangle face in the human mesh and n_f^* denotes a ground truth unit normal vector of f. The v_i and v_j denotes the ith and jth vertices in f, respectively.

Surface Edge Loss: Following [5, 51], we use a edge length consistency loss between the predicted edges and ground truth edges. The surface edge loss aims to improve the smoothness of hands, feet, and a mouth which is defined as

$$\mathcal{L}_{Edge} = \sum_{f}\sum_{\{i,j\}\subset f} \left| \|v_i - v_j\|_2 - \|\hat{v_i} - \hat{v_j}\|_2 \right|, \tag{12}$$

where f denotes a triangle face in the human mesh. The v_i and v_j denotes the ith and jth vertices in f, respectively.

Based on the four type of loss functions, our **overall loss** is written as

$$\mathcal{L} = \lambda_v \mathcal{L}_{Vertex} + \lambda_j \mathcal{L}_{Joint} + \lambda_n \mathcal{L}_{Normal} + \lambda_e \mathcal{L}_{Edge}, \tag{13}$$

where $\lambda_v = 1$, $\lambda_j = 0.01$, $\lambda_n = 0.01$, and $\lambda_e = 0.01$.

4 EXPERIMENTS

4.1 Datasets and Evaluation Metrics

Human3.6M is one of the most widely used large-scale indoor dataset for 3D HPE and mesh reconstruction [13]. There are 3.6M video frames recorded by 11 professional actors with performing 17 actions. The ground truth 3D pose annotations were captured by an accurate marker-based motion capture system, but no ground truth 3D mesh annotations were provided. The previous works [15, 19, 20] used pseudo-ground truth mesh provided by Mosh [29]. However, they are no longer accessible due to license issues. Now we use the pseudo-ground truth mesh generated by [5]. Following previous works [5, 15, 19, 20], we select 5 subjects (S1, S5, S6, S7, S8) for training and 2 subjects for testing (S9, S11).

3DPW is an in-the-wild dataset [50] that contains 60 video sequences (51K video frames) captured in the outdoor environment. The ground truth 3D pose and mesh annotations are provided. We only use its defined test set for evaluation following [5, 19].

We also use **MSCOCO** [25] and **MuCo-3DHP** [33] for mixed training following [5, 19, 34]. For evaluation metrics, we use the three standard metrics below. The unit for these metrics is millimeters (mm).

MPJPE: Mean-Per-Joint-Position-Error is used to evaluate the estimated 3D human pose. It is computed as the mean Euclidean distance between the estimated joints and the ground truth joints.

PA-MPJPE: is the MPJPE after Procrustes Analysis [9]. The rigid alignment using procrustes analysis is performed the estimated 3D pose, then compute MPJPE with the ground truth 3D pose. P-MPJPE aims to measure the errors of the reconstructed structure without considering translations and rotations.

MPVE: Mean-Per-Vertex-Error is used to evaluate the estimated 3D mesh vertices. It is computed as the mean Euclidean distance between the estimated and the ground truth mesh vertices.

Table 1: Performance comparison with SOTA methods on Human3.6M. "†" denotes using GT (ground truth) 2D pose as input.

	Methods		Human3.6M	
			MPJPE↓	PA-MPJPE↓
Image Based	HMR [15]	CVPR 2018	88.0	56.8
	GraphCMR [20]	CVPR 2019	-	50.1
	SPIN [19]	ICCV 2019	-	41.1
	METRO [22]	CVPR 2021	54.0	36.7
	MeshGraphormer [23]	ICCV 2021	**51.2**	**34.5**
	PyMAF [55]	ICCV 2021	57.7	40.5
Video Based	VIBE [18]	CVPR 2020	65.6	41.4
	TCMR [4]	CVPR 2021	62.3	41.1
Pose Based (detected by [46])	Pose2Mesh [5]	ECCV 2020	64.9	47.0
	Baseline	-	68.3	50.0
	GTRS	-	64.3	45.4
Pose Based (GT 2D)	Pose2Mesh [5]†	ECCV 2020	51.3	34.9
	Baseline†	-	57.5	39.6
	GTRS†	-	**50.3**	**30.4**

4.2 Implementation Details

We implemented GTRS with Pytorch [40] using two NVIDIA RTX 3090 GPUs. GTRS can be trained in an end-to-end manner, but we first pretrain the PAM, then train the entire GTRS after loading those weights for better performance. In the PAM, there are 6 graph transformer blocks (1 with fixed adjacency matrix and 5 with learnable adjacency matrix) and the embedding dimension is 128. We use Adam [16] optimizer with a learning rate 1×10^{-4} to pretrain the PAM for 120 epochs. In MRM, the embedding dimension is the same as in PAM, which is 128, and the layer number N is 4. The mesh template $M_{temp} \in \mathbb{R}^{6890\times3}$ is the mean SMPL parameters provided by [19]. We also use Adam [16] optimizer with a learning rate 1×10^{-4} to train the GTRS for 180 epochs. The total training time would be less than one day.

4.3 Comparison with state-of-the-art results

Evaluation on Human3.6M: Table 1 compares GTRS with previous SOTA image/video-based and pose-based methods on Human3.6M test set. We follow the same setting as Pose2Mesh [5] and [15, 19, 20] for a fair comparison; that is, we only use the Human3.6M training set for training and PA-MPJPE is only measured on the frontal camera set. Same as Pose2Mesh [5], [46] is used as the 2D pose detector. Within the similar total number of parameters and FLOPs, the performance boost from our baseline to GTRS demonstrates that transformer architecture is more suitable than CNN-based architecture when considering model efficiency for pose-based human mesh recovery. GTRS achieves better performances than Pose2Mesh [5] (MPJPE decreases 0.6 and PA-MPJPE decreases 1.6). Compared with video-based methods, GTRS still achieves similar MPJPE (within 2 points) as a lightweight model. When using ground truth 2D pose as input, GTRS outperforms previous methods both in terms of MPJPE and PA-MPJPE. This results indicates the lower bound of GTRS. As more accurate and robust 2D human pose detectors are proposed, they can be plugged-in to GTRS and close the gap towards this lower bound.

Evaluation on 3DPW: Table 2 compares GTRS with previous SOTA image/video-based and pose-based methods on 3DPW test set. The training sets include Human3.6M, COCO, MPII [1], UP3D [21], MPI-INF-3DHP [32], and AMASS [31]. Each method uses a different combination of these datasets. For GTRS, we only use Human3.6M [13], COCO [25], and MuCo [33] as the training sets. Following Pose2Mesh [5], we use DARK [54] as the 2D pose detector for 3DPW evaluation. Comparing our baseline with GTRS under similar total number of parameters and FLOPs, we can draw the same conclusion that transformer architecture is more suitable than CNN-based architecture when considering model efficiency for pose-based human mesh recovery. Note here METRO [22] and MeshGraphormer [23] used 3DPW training set while others did not. Among those methods without using 3DPW training set, GTRS achieves comparable results with much less computational cost.

Table 2: Performance comparison with SOTA methods on 3DPW dataset. * indicates 3DPW training set is used during training. "†" denotes using GT 2D pose as input.

	Methods		3DPW dataset		
			MPJPE↓	PA-MPJPE↓	MPVE↓
Image Based	HMR [15]	CVPR 2018	-	81.3	-
	GraphCMR [20]	CVPR 2019	-	70.2	-
	SPIN [19]	ICCV 2019	-	59.2	116.4
	I2LMeshNet [34]	ECCV 2020	93.2	57.7	-
	PyMAF [55]	ICCV 2021	92.8	58.9	110.1
	MeshGraphormer* [23]	ICCV 2021	**74.7**	**45.6**	**87.7**
Video Based	VIBE [18]	CVPR 2020	93.5	56.5	113.4
	VIBE* [18]	CVPR 2021	82.9	51.9	99.1
	TCMR [4]	CVPR 2021	95.0	55.8	111.5
Pose Based (detected by [54])	Pose2Mesh [5]	ECCV 2020	88.9	58.3	106.3
	Baseline	-	95.6	61.3	129.8
	GTRS	-	88.5	58.9	106.2
Pose Based (GT 2D)	Pose2Mesh [5]†	ECCV 2020	65.1	34.6	-
	Baseline†	-	67.7	36.1	70.3
	GTRS†	-	**53.8**	**34.5**	**61.6**

When using ground truth 2D pose input, the performance of GTRS can be improved significantly. GTRS yields the lowest MPJPE of 53.8, PA-MPJPE of 34.5, and MPVPE of 61.6 (more than 24% reduction compared with MeshGraphormer [23]). 3DPW dataset is a challenging in-the-wild dataset, distinct from the lab-controlled Human3.6M dataset. Occlusions and atypical human postures in these in-the-wild cases are the biggest challenges for accurate 3D human mesh recovery. Due to the difficulties of obtaining sufficient training data with accurate mesh annotations, these issues are difficult to address directly in 3D pose estimation. However, it is much easier to acquire sufficient training data with accurate 2D pose annotations. By simply plugging-in a robust and more accurate 2D pose detector in the future, GTRS can further improve the performance and approach the lower bound. Therefore, GTRS has the potential to continue staying relevant as an effective approach for 3D mesh reconstruction as 2D pose estimators inevitably improve.

Model Size and FLOPs Comparison: Previous human mesh reconstruction methods did not pay much attention to model efficiency. These methods mainly pursued higher accuracy without considering computation and memory. Table 3 reports the Params and FLOPs comparison between previous methods and GTRS. Pose-based methods can easily select a lightweight 2D pose detector to reduce the computational burden. But for most image/video-based methods, a very large feature extractor is needed for extracting features (e.g. ResNet-50 is used by [4, 18, 19]). GTRS only requires 7.9M Params (10.2% of the Pose2Mesh [5]) and 0.19G FLOPs (2.5% of the Pose2Mesh [5]) while achieving better results than Pose2Mesh [5] when using the same 2D pose input detected by DARK [54]. When compared with our baseline, GTRS improves the performance while maintaining lower memory and computational cost. When we use Lite-HRNet [53] as 2D pose detector, the overall Params is 9.7M and FLOPs is 0.89G. GTRS achieves close results with significant Params and FLOPs reduction (6.9% and 1.2% of I2LMeshNet [34], 16.5% and 9.2% of VIBE [18], 7.9% and 8.6% of TCMR [4]). It also can be observed that METRO [22] and MeshGraphormer [22] demanded an extremely large number

Table 3: The Params and FLOPs comparison on 3DPW dataset. * indicates 3DPW training set is used during training.

	Methods		Feature extractor		Proposed model		Overall		MPJPE
			Params (M)	FLOPs (G)	Params (M)	FLOPs (G)	Params (M)	FLOPs (G)	on 3DPW
Image Based	I2LMeshNet [34]	ECCV2020	-	-	-	-	140.5	73.2	93.2
	METRO* [22]	CVPR2021	-	-	-	-	229.2	56.6	77.1
	MeshGraphormer* [23]	ICCV2021	-	-	-	-	226.5	56.6	**74.7**
Video Based	VIBE [18]	CVPR2020	25.6	8.2	33.0	1.3	58.6	9.6	93.5
	TCMR [4]	CVPR2021	25.6	8.2	97.3	2.1	122.9	10.3	95.0
	Methods		2D pose detector		Proposed model		Overall		MPJPE
			Param (M)	FLOPs (G)	Param (M)	FLOPs (G)	Param (M)	FLOPs (G)	on 3DPW
Pose Based	Pose2Mesh with DARK [54]	ECCV2020	63.6	3.6	77.1	7.5	140.7	11.1	88.9
	Baseline with DARK [54]	-	63.6	3.6	8.3	0.32	71.9	3.9	97.7
	GTRS with DARK [54]	-	63.6	3.6	**7.9**	**0.19**	71.5	3.8	88.5
	Pose2Mesh with LiteHRNet [53]	ECCV2020	**1.8**	**0.7**	77.1	7.5	78.9	8.2	-
	Baseline with LiteHRNet [53]	-	**1.8**	**0.7**	8.3	0.32	11.1	1.1	103.4
	GTRS with LiteHRNet [53]	-	**1.8**	**0.7**	**7.9**	**0.19**	**9.7**	**0.89**	93.5

Table 4: Ablation study on different components in PAM.

Architecture in PAM			
graph transformer blocks with fixed adj	graph transformer blocks with learnable adj	MPJPE↓	PA-MPJPE↓
1	0	68.0	50.3
1	1	66.9	48.9
1	3	65.1	48.2
1	5	**64.3**	**47.5**
1	7	64.6	47.6
6	0	65.3	48.4
0	6	66.4	48.8
pure transformer blocks			
6		67.2	49.5

of Params and FLOPs to achieve the SOTA results; they also used the 3DPW training set while other methods did not. Apart from that, METRO [22] and MeshGraphormer [22] were trained in 5 days on 8 NVIDIA V100 GPUs, which is incredibly time and resource consuming. GTRS, on the other hand, can be trained in less than one day on two NVIDIA RTX 3090 GPUs. **To summarize, GTRS is much more time and resource efficient compared to image-based methods and SOTA pose-based method.**

4.4 Ablation Study

We conduct the ablation study on Human3.6M dataset (training on S1, S5, S6, S7, S8, and testing on S9 and S11) and report the accuracy using MPJPE and PA-MPJPE.

Effectiveness of Using Graph Transformer Blocks: We investigate the use of graph transformer blocks in the PAM in Table 4. All blocks are applied in parallel as shown in Fig 3. When we only use one fixed graph transformer block, the MPJPE is 68.0 and PA-MPJPE is 50.3. By adding learnable graph transformer blocks, and thereby enabling the exploration of implicit correlations, the performance improves. The network achieves the best results (MPJPE is 64.3 and PA-MPJPE is 47.5) when using one graph transformer block with a fixed adjacency matrix and five graph transformer blocks with learnable adjacency matrices,

Next, to verify that the observed performance increase is not solely a matter of additional blocks, we also investigate fixing all adjacent matrices in the six graph transformer blocks. This enforces the graph transformer blocks only to learn with structured correlations of human kinematics. The MPJPE is 65.3 and PA-MPJPE is 48.4. On the contrary, we then try setting all adjacent matrices in the six graph transformer blocks to be learnable during the training. This allows the graph transformer blocks to learn implicit correlations of human kinematics, and results in a MPJPE of 66.4 and PA-MPJPE of 48.8. Lastly, we apply six pure transformer blocks (without any GCNs), which gives a MPJPE of 67.2 and PA-MPJPE of 49.5. All these results are worse than applying six graph transformer blocks (one with fixed adjacency matrix and five with learnable adjacency matrices), which verifies the effectiveness of using graph transformer blocks with both fixed and learnable adjacency matrices.

Impact of 2D Pose Detectors: In Table 5, we analyze the impact of the quality of 2D pose on the final mesh performance. When using ground-truth 2D pose as input, GTRS can get 50.3 of MPJPE and 30.4 of PA-MPJPE, which outperform the Pose2Mesh [5] results (51.3 of MPJPE and 34.9 of PA-MPJPE). Here, we see that accuracy of GTRS can be improved when using more precise 2D pose inputs. We also evaluate the performance of using different 2D pose detectors. Incorporated with [47], the MPJPE of GTRS is 64.3 and PA-MPJPE is 47.5, but the Params is 34M and FLOPs is 6.4G. When switching to a more lightweight 2D pose detector [53], the entire pipeline is more computational and memory efficient (Params is 9.7M and FLOPs is 0.9G) while preserving the accuracy.

Impact of 2D Pose Input during Inference: For testing in-the-wild images, the quality of the 2D pose would be affected by various fluctuations based on the input image. We evaluate the robustness of our trained model against potential perturbations since GTRS relies on the input pose.

First, we evaluate if GTRS can still perform well when input joints are missing. During inference, we set a drop probability for the joints and the results are shown in Fig. 7 (a). In PAM, we output the intermediate 3D pose for a 3D pose supervision, which enables GTRS can still achieve acceptable results given a large drop rate.

Second, we evaluate the robustness of GTRS to noisy 2D pose input. Specifically, we add Gaussian noise $\mathcal{N}(0, \sigma^2)$ to the input 2d pose to simulate in-the-wild 2D pose input during inference. We do not retrain the model (which is trained using GT pose), instead, we directly evaluate the performance of noisy pose input on the trained model. The results are shown in Fig. 7 (b). GTRS consistently outperforms Pose2Mesh [5], demonstrating that GTRS is more robust to in-the-wild inference.

Qualitative Results: Fig. 6 shows the qualitative results of GTRS on in-the-wild images from COCO dataset that GTRS can reconstruct acceptable human meshes. More qualitative results are in appendix.

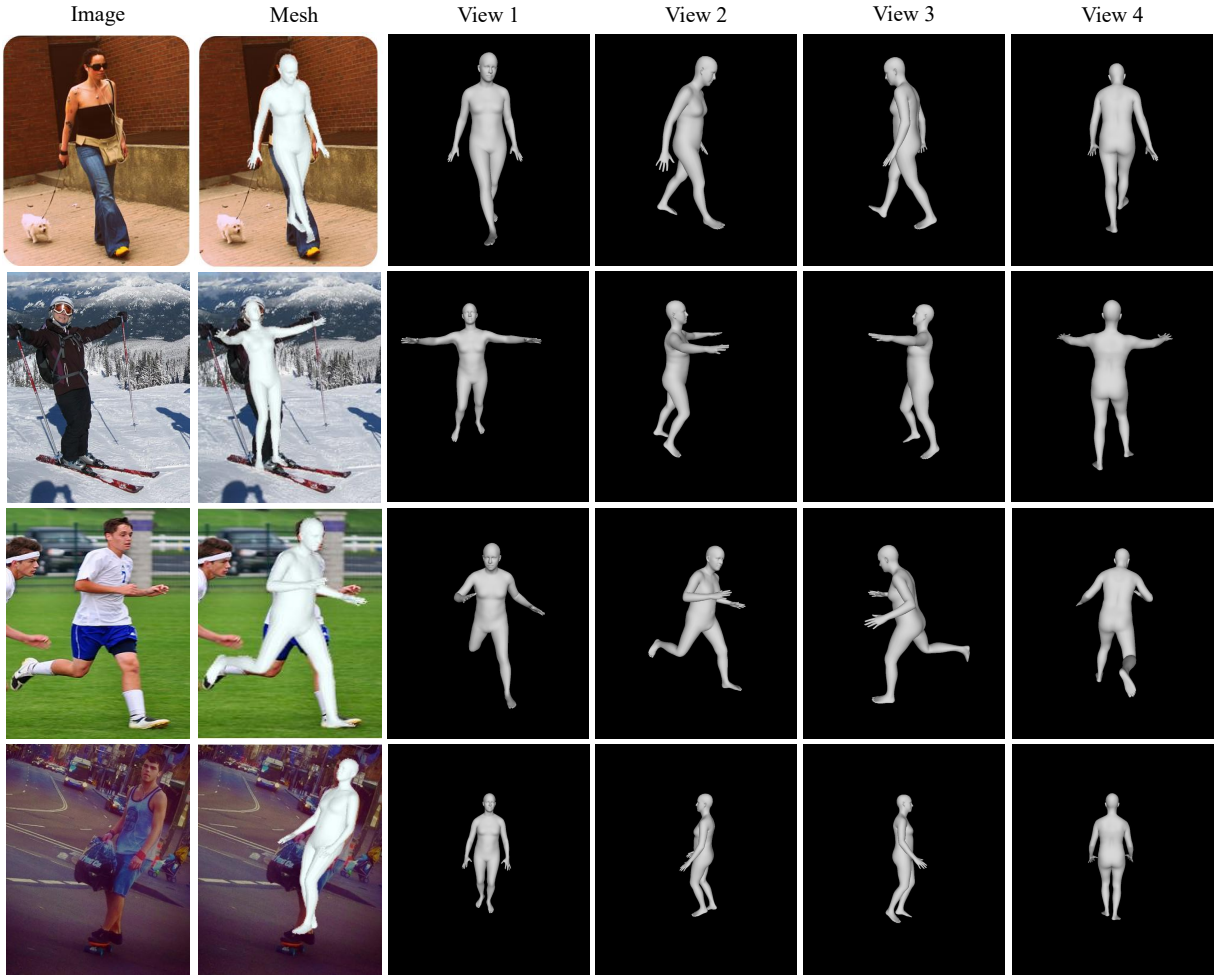
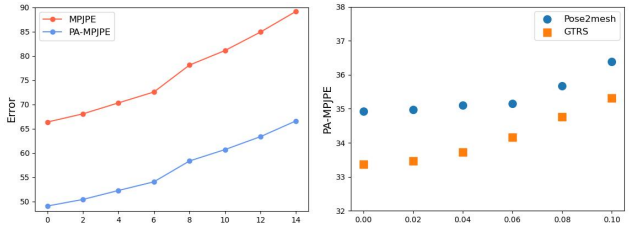


Figure 6: Qualitative results of the proposed GTRS. Images are taken from the in-the-wild COCO [25] dataset.

Table 5: Ablation study on different 2D pose input.

Methods	Input	MPJPE↓	PA-MPJPE↓
Pose2Mesh [5]	GT 2D pose	51.3	34.9
GTRS (Ours)	GT 2D pose	50.3	30.4
	Estimated 2D pose by [47]	64.3	47.5
	Estimated 2D pose by LiteHRNet [53]	66.9	48.7



(a) Drop rate (%) (b) Variance

Figure 7: (a) Impact of missing joints during inference. Each joint has a drop probability to simulate missing joints by the 2D pose detector during inference. (b) Impact of noisy 2D pose input during inference. Various degrees of Gaussian noise (i.e. variance σ^2) are added to the input 2d pose to simulate in-the-wild 2D pose input during inference. $\sigma^2 = 0$ means GT pose provided in the Human3.6M dataset.

5 CONCLUSION AND DISCUSSION

We present a lightweight pose-based method, GTRS, for human mesh reconstruction from 2D human pose that reduces Params and FLOPs significantly. A PAM is introduced to exploit structured and implicit joint correlations by using paralleled graph transformers blocks. Then, a MRM is able to combine the extracted pose feature with the mesh template efficiently to reconstruct the human mesh.

Despite GTRS achieving competitive performance, as a pose-based approach, GTRS may not be able to recover varied human body shapes using only 2D human poses as input. Although image-based methods have the potential to reconstruct a more accurate human mesh, pose-based methods are still worth investigating due to their flexibility and lightweight design. In the future, we intend to include another branch that extracts human shape features from images to improve reconstruction capability while keeping the model structure lightweight.

Acknowledgement. This work is supported by the National Science Foundation under Grant No. 1910844

A APPENDIX

A.1 Detailed overview of the proposed GTRS architecture

The detailed overview of GTRS is illustrated in Fig. 8. Given the input image, 2D human pose $X_{in\_2D} \in \mathbb{R}^{17\times2}$ is first detected by an off-the-shelf 2D pose detector, then a feature embedding layer (an MLP layer) embeds input 2D pose to $X_{pose} \in \mathbb{R}^{17\times128}$.

In the pose analysis module, the pose feature $X_{pose} \in \mathbb{R}^{17\times128}$ goes through each parallel graph transformer block without changing the size. A fusion block, consists of convolutional layers, is designed to aggregate these pose features back to $X^{'}_{pose} \in \mathbb{R}^{17\times128}$.

Before mesh regression module, the mesh template $M_{temp} \in \mathbb{R}^{6890\times3}$ is embedded to the template feature $X_{temp} \in \mathbb{R}^{15\times128}$. In the mesh regression module, the pose feature $X^{'}_{pose}$ and template feature X_{temp} are modeled separately, then concatenated to the mesh feature $X_{mesh} \in \mathbb{R}^{32\times128}$ for a large transformer modeling. The output would be $X_{out} \in \mathbb{R}^{32\times128}$.

Finally, the estimated mesh parameter $Y \in \mathbb{R}^{6890\times3}$ can be obtained by the regression head (an MLP layer) for human mesh reconstruction.

A.2 Design options of transformer block

Table 6: Ablation study on different design options of transformer block. Results are evaluated on Human3.6M dataset.

	MPJPE↓	PA-MPJPE↓
Original	65.2	47.7
Add CONV	64.5	**47.5**
Add CONV and replace MLP by SE	**64.3**	**47.5**

Our transformer encoder layer is different from the original transformer encoder in [7]. The structure of our transformer encoder is illustrated in Fig. 9 (c). If we use the original transformer block in Fig. 9 (a), the MPJPE is 65.2 and PA-MPJPE is 47.7 as shown in Table 6. Compared with the original transformer blocks, we add one convolutional branch parallel to the MHA branch as shown in Fig. 9 (b). We use this pointwise convolution to create linear combinations of the joint channels while maintaining a low computational cost. The performance has improved since MPJPE is decreased to 64.5 and PA-MPJPE is decreased to 47.5. Then, we replace the MLP with a lightweight SE block [12] as shown in Fig. 9 (c) and achieve the best performance (MPJPE is 64.3 and PA-MPJPE is 47.5).

A.3 Different locations to inject GCN.

In GTRS, we utilize GCNs to maintain a strong joint relationship based on human kinematic information. We inject structural priors provided by GCNs before performing the transformer's multi-head self-attention (MSA) as shown in Fig. 10 (a). We also investigate other locations to inject GCNs. The results are reported in Table 7. When GCN is between the MSA and SE block as illustrated in Fig. 10 (b), the MPJPE is 64.9 and PA-MPJPE is 48.2. When GCN is behind the SE block in Fig. 10 (c), the MPJPE is 65.2 and PA-MPJPE is 48.3. We observe that GCN before the MHA achieves the best performance (MPJPE is 64.3 and PA-MPJPE is 47.5), indicating that GCNs do provide structural priors to help final mesh regression.

Table 7: Ablation study on different locations to inject GCN. Results are evaluated on Human3.6M dataset.

	MPJPE↓	PA-MPJPE↓
GCN before MHA	**64.3**	**47.5**
GCN between MHA and SE	64.9	48.2
GCN after SE	65.2	48.3

A.4 Impact of different backbones

As a pose-based method, GTRS can easily choose a lightweight 2D pose detector to reduce the computational burden. But for most image/video-based methods, a very large feature extractor is needed for extracting features (e.g. ResNet-50 is used by [4, 18, 19]). We retrain the I2LMeshNet [34] with a small backbone (ResNet18), the results are shown in Table 8. Although selecting a small backbone can reduce the total Params and FLOPs, the performance also drops. GTRS with DARK [54] is much more time and resource efficient compared to I2LMeshNet.

Table 8: Impact of different backbones. Results are evaluated on 3DPW dataset.

	Backbone	total Params(M)	FLOPs(G)	MPJPE	PA-MPJPE
I2LMeshNet	ResNet50	140.5	73.2	93.2	57.7
I2LMeshNet (small)	ResNet18	94.6	68.3	101.8	62.9
GTRS with DARK [54]		71.5	3.8	88.5	58.9

A.5 Impact of different losses

We apply multiple loss introduced in Section 3.6 when training GTRS. Here we evaluate the impact of different losses combination in Table.

Table 9: Comparison of the inference speed. The frame per second (fps) is obtained by using batch size 1 on a single GPU/CPU.

	Human3.6M MPJPE	Human3.6M MPVE
Vertex Loss only	71.9	86.3
Vertex Loss + 3D joint Loss	65.3	83.4
Vertex Loss + 3D joint Loss + Normal Loss	64.9	82.4
Vertex Loss + 3D joint Loss + Normal Loss + Edge Loss	64.3	82.0

Without 3D joint supervision, the MPJPE is 71.9 and MPVE is 86.3 (2D pose is detected by [42]). After adding 3D joint supervision, the performances are boosted to 65.3 of MPJPE and 83.4 of MPVE. The normal loss and edge loss can further improve the performance slightly as shown in the table below.

A.6 Evaluation on the inference speed

Table 10: Comparison of the inference speed. The frame per second (fps) is obtained by using batch size 1 on a single GPU/CPU.

	FPS on GPU			FPS on CPU		
	2D pose detection	3D mesh regression	overall	2D pose detection	3D mesh regression	overall
METRO [22]	-	-	15.65	-	-	1.81
GTRS	133.73	32.41	**26.17**	19.87	24.11	**10.86**

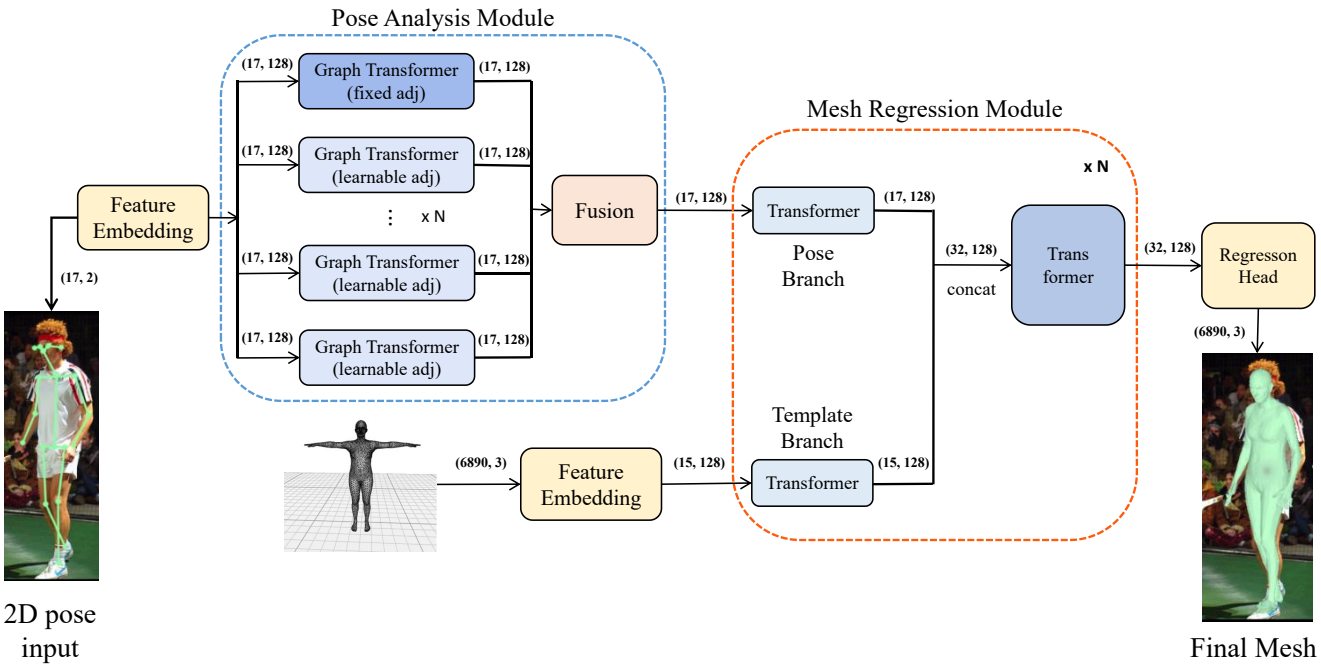


Figure 8: Overview of the proposed GTRS architecture. The mesh template figure is from [22].

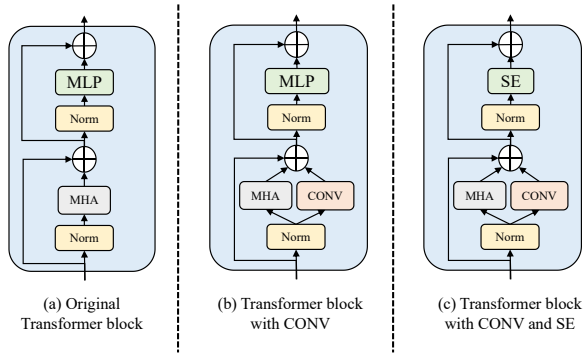


Figure 9: Three design choices for building our proposed transformer block. (a) is the original transformer block from [7]. (b) is the transformer block with adding a convolutional branch parallel to the MHA branch. (c) is the transformer block that replaces MLP by SE block based on (b).

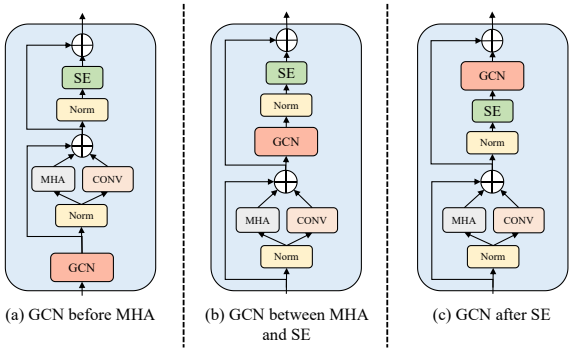


Figure 10: Three different locations to inject GCN to transformer blocks. (a) GCN is in front of the MHA block. (b) GCN is between the MHA and SE block. (c) GCN is behind the SE block. Results are evaluated on Human3.6M dataset.

We compare the inference speed between GTRS (end-to-end) and the state-of-the-art image-based mesh reconstruction method METRO[22]. The frame per second (fps) is reported in Table 7. We use a single NVIDIA RTX 3090 GPU and an AMD Ryzen 3970X 32-Core Processor CPU for testing. Our proposed GTRS is a pose-based method, which means any off-the-shelf 2D pose detector can be easily adopted. Here we use the lightweight OpenPose [39] as the 2D pose detector. The fps numbers of lightweight OpenPose on the testing GPU and CPU are 133.73 and 19.87, respectively. For GTRS, the fps numbers on the GPU and CPU are 32.41 and 24.11, respectively. Overall, GTRS can achieve 26.27 fps on GPU and 10.86 fps on CPU, which is **much faster** than METRO (the fps is 15.65 on GPU and 1.81 on CPU) on the same computing hardware. Compared to METRO, GTRS gains more advantages on resource-constrained devices since it is significantly more computational efficient.

A.7 More qualitative results of GTRS

In Fig. 11, we show the qualitative results of GTRS on in-the-wild images. We observe that GTRS achieves acceptable performance by reconstructing reasonable human mesh on these challenging in-the-wild cases. However, there are still some failure cases of our method due to challenging pose and heavy occlusion as shown in Fig. 12.

We also compare with image-based method I2LMeshNet [34] in Fig. 13. Our GTRS is comparable with I2LMeshNet while reducing memory and computational cost significantly (only 6.9 % Params and 1.2 % FLOPs).

Figure 11: Qualitative results of the proposed GTRS on in-the-wild images. Images are taken from MSCOCO[25].

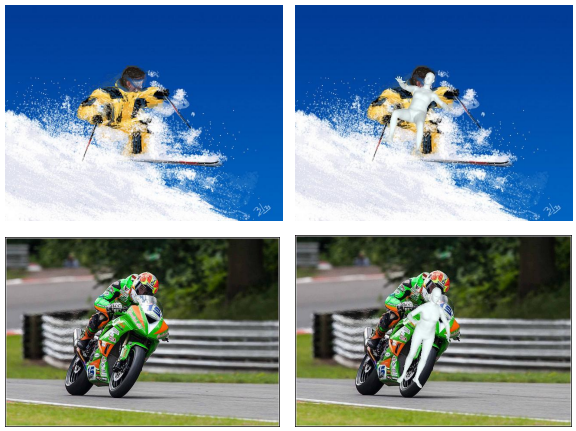


Figure 12: Failure cases due to challenging pose and heavy occlusion.

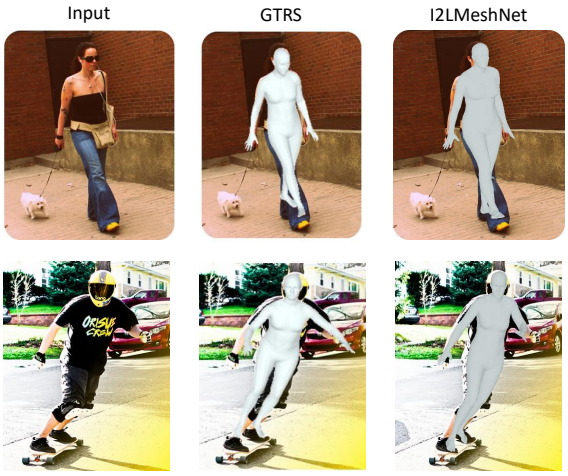


Figure 13: Qualitative comparison with image-based method I2LMeshNet.

REFERENCES

[1] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2014. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*. 3686–3693.

[2] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. 2016. Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image. In *ECCV*.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European Conference on Computer Vision*. Springer, 213–229.

[4] Hongsuk Choi, Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. 2021. Beyond Static Features for Temporally Consistent 3D Human Pose and Shape From a Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1964–1973.

[5] Hongsuk Choi, Gyeongsik Moon, and Kyoung Mu Lee. 2020. Pose2Mesh: Graph Convolutional Network for 3D Human Pose and Mesh Recovery from a 2D Human Pose. In *ECCV*.

[6] H. Ci, C. Wang, X. Ma, and Y. Wang. 2019. Optimizing Network Structure for 3D Human Pose Estimation. In *ICCV*.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR* (2021).

[8] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. 2017. Rmpe: Regional multi-person pose estimation. In *ICCV*.

[9] James W Grice and Kimberly K Assad. 2009. Generalized procrustes analysis: a tool for exploring aggregates and persons. *Applied Multivariate Research* 13, 1 (2009), 93–112.

[10] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415* (2016).

[11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[12] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

[13] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. 2014. 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 (jul 2014), 1325–1339.

[14] Wen Jiang, Nikos Kolotouros, Georgios Pavlakos, Xiaowei Zhou, and Kostas Daniilidis. 2020. Coherent Reconstruction of Multiple Humans From a Single Image. In *CVPR*.

[15] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. 2018. End-to-end Recovery of Human Shape and Pose. In *CVPR*.

[16] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

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

[19] N. Kolotouros, G. Pavlakos, M. Black, and K. Daniilidis. 2019. Learning to Reconstruct 3D Human Pose and Shape via Model-Fitting in the Loop. In *ICCV*.

[20] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. 2019. Convolutional Mesh Regression for Single-Image Human Shape Reconstruction. In *CVPR*.

[21] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J Black, and Peter V Gehler. 2017. Unite the people: Closing the loop between 3d and 2d human representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6050–6059.

[22] Kevin Lin, Lijuan Wang, and Zicheng Liu. 2021. End-to-end human pose and mesh reconstruction with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1954–1963.

[23] Kevin Lin, Lijuan Wang, and Zicheng Liu. 2021. Mesh Graphormer. In *ICCV*.

[24] Kevin Lin, Lijuan Wang, and Zicheng Liu. 2021. Mesh Graphormer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 12939–12948.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[26] Kenkun Liu, Rongqi Ding, Zhiming Zou, Le Wang, and Wei Tang. 2020. A Comprehensive Study of Weight Sharing in Graph Networks for 3D Human Pose Estimation. In *ECCV*.

[27] Ruixu Liu, Ju Shen, He Wang, Chen Chen, Sen-ching Cheung, and Vijayan Asari. 2020. Attention mechanism exploits temporal contexts: Real-time 3d human pose reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5064–5073.

[28] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *International Conference on Computer Vision (ICCV)* (2021).

[29] Matthew Loper, Naureen Mahmood, and Michael J Black. 2014. MoSh: Motion and shape capture from sparse markers. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 1–13.

[30] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM TOG* (2015).

[31] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. 2019. AMASS: Archive of Motion Capture as Surface Shapes. In *ICCV*.

[32] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. 2017. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *2017 international conference on 3D vision (3DV)*. IEEE, 506–516.

[33] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Srinath Sridhar, Gerard Pons-Moll, and Christian Theobalt. 2018. Single-Shot Multi-Person 3D Pose Estimation From Monocular RGB. In *3D Vision (3DV), 2018 Sixth International Conference on*. IEEE. http://gvv.mpi-inf.mpg.de/projects/SingleShotMultiPerson

[34] Gyeongsik Moon and Kyoung Mu Lee. 2020. I2L-MeshNet: Image-to-Lixel Prediction Network for Accurate 3D Human Pose and Mesh Estimation from a Single RGB Image. In *ECCV*.

[35] Christopher Neff, Aneri Sheth, Steven Furgurson, John Middleton, and Hamed Tabkhi. 2021. EfficientHRNet: efficient and scalable high-resolution networks for real-time multi-person 2D human pose estimation. *Journal of Real-Time Image Processing* 18, 4 (2021).

[36] Mohamed Omran, Christoph Lassner, Gerard Pons-Moll, Peter V. Gehler, and Bernt Schiele. 2018. Neural Body Fitting: Unifying Deep Learning and Model-Based Human Pose and Shape Estimation. In *3DV*.

[37] Ahmed A A Osman, Timo Bolkart, and Michael J. Black. 2020. STAR: A Spare Trained Articulated Human Body Regressor. In *ECCV*.

[38] Daniil Osokin. 2018. Real-time 2d multi-person pose estimation on CPU: Lightweight OpenPose. *arXiv preprint arXiv:1811.12004* (2018).

[39] Daniil Osokin. 2018. Real-time 2D Multi-Person Pose Estimation on CPU: Lightweight OpenPose. In *arXiv preprint arXiv:1811.12004*.

[40] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

[41] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. 2018. Learning to Estimate 3D Human Pose and Shape from a Single Color Image. In *CVPR*.

[42] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 2019. 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*. 7753–7762.

[43] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[44] Xuan Shen, Geng Yuan, Wei Niu, Xiaolong Ma, Jiexiong Guan, Zhengang Li, Bin Ren, and Yanzhi Wang. 2021. Towards Fast and Accurate Multi-Person Pose Estimation on Mobile Devices. *arXiv preprint arXiv:2106.15304* (2021).

[45] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. 2019. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5693–5703.

[46] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. 2019. Deep high-resolution representation learning for human pose estimation. In *CVPR*.

[47] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. 2018. Integral human pose regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 529–545.

[48] Matt Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. 2017. Total Capture: 3D Human Pose Estimation Fusing Video and Inertial Sensors. In *BMVC*.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[50] Timo von Marcard, Roberto Henschel, Michael Black, Bodo Rosenhahn, and Gerard Pons-Moll. 2018. Recovering Accurate 3D Human Pose in The Wild Using IMUs and a Moving Camera. In *European Conference on Computer Vision (ECCV)*.

[51] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. 2018. Pixel2mesh: Generating 3d mesh models from single rgb images. In

Proceedings of the European Conference on Computer Vision (ECCV). 52–67.

[52] Xiangyu Xu, Hao Chen, Francesc Moreno-Noguer, László A Jeni, and Fernando De la Torre. 2020. 3D Human Shape and Pose from a Single Low-Resolution Image with Self-Supervised Learning. In *ECCV*.

[53] Changqian Yu, Bin Xiao, Changxin Gao, Lu Yuan, Lei Zhang, Nong Sang, and Jingdong Wang. 2021. Lite-HRNet: A Lightweight High-Resolution Network. In *CVPR*.

[54] Feng Zhang, Xiatian Zhu, Hanbin Dai, Mao Ye, and Ce Zhu. 2020. Distribution-Aware Coordinate Representation for Human Pose Estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[55] Hongwen Zhang, Yating Tian, Xinchi Zhou, Wanli Ouyang, Yebin Liu, Limin Wang, and Zhenan Sun. 2021. PyMAF: 3D Human Pose and Shape Regression with Pyramidal Mesh Alignment Feedback Loop. In *Proceedings of the IEEE International Conference on Computer Vision*.

[56] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N. Metaxas. 2019. Semantic Graph Convolutional Networks for 3D Human Pose Regression. In *CVPR*.

[57] Weixi Zhao, Yunjie Tian, Qixiang Ye, Jianbin Jiao, and Weiqiang Wang. 2021. GraFormer: Graph Convolution Transformer for 3D Pose Estimation. *arXiv preprint arXiv:2109.08364* (2021).

[58] Ce Zheng, Sijie Zhu, Matias Mendieta, Taojiannan Yang, Chen Chen, and Zhengming Ding. 2021. 3D Human Pose Estimation With Spatial and Temporal Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 11656–11665.

[59] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. 2021. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6881–6890.

[60] Hao Zhu, Xinxin Zuo, Sen Wang, Xun Cao, and Ruigang Yang. 2019. Detailed human shape estimation from a single image by hierarchical mesh deformation. In *CVPR*.

[61] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2020. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159* (2020).

[62] Zhiming Zou and Wei Tang. 2021. Modulated Graph Convolutional Network for 3D Human Pose Estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 11477–11487.