LiDAR-HMR: 3D Human Mesh Recovery from LiDAR

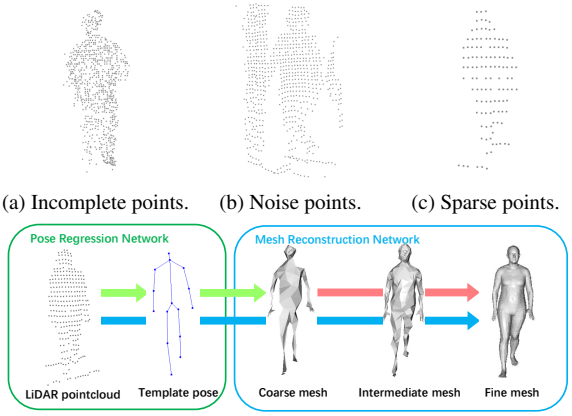
Bohao Fan * Wenzhao Zheng Jianjiang Feng Jie Zhou
Beijing National Research Center for Information Science and Technology, China
Department of Automation, Tsinghua University, China
fbh19@mails.tsinghua.edu.cn; wenzhao.zheng@outlook.com;
{jfeng, jzhou}@tsinghua.edu.cn

Abstract

In recent years, point cloud perception tasks have been garnering increasing attention. This paper presents the first attempt to estimate 3D human body mesh from sparse LiDAR point clouds. We found that the major challenge in estimating human pose and mesh from point clouds lies in the sparsity, noise, and incompletion of LiDAR point clouds. Facing these challenges, we propose an effective sparse-to-dense reconstruction scheme to reconstruct 3D human mesh. This involves estimating a sparse representation of a human (3D human pose) and gradually reconstructing the body mesh. To better leverage the 3D structural information of point clouds, we employ a cascaded graph transformer (graphormer) to introduce point cloud features during sparse-to-dense reconstruction. Experimental results on three publicly available databases demonstrate the effectiveness of the proposed approach. Code: https://github.com/soullessrobot/LiDAR-HMR/

1. Introduction

3D human pose estimation (HPE) and human mesh recovery (HMR) from un-constrained scenes has been a long-standing goal in computer vision. However, due to the difficulty in obtaining accurate 3D annotations, the use of 3D input, such as point clouds, for pose estimation primarily focuses on dense point clouds derived from depth maps [1, 2, 10, 16, 30] or weakly supervised methods [5, 37, 43, 47]. With the introduction of the Waymo [35] dataset and algorithms such as LPFormer [41], the feasibility of using point cloud data to perceive human pose information has been indicated. In this paper, we present the first attempt to estimate the human mesh, rather than joints, from sparse LiDAR observations. Along with 3D joint-based methods [41, 43, 47], HMR has many downstream tasks such as VR/AR, and computer graphics as a fundamental topic in computer vision. Furthermore, compared with HPE, HMR contains extra information on the shape and details of the human body, which extends to a wider range of applications such as human-computer interaction and identity recognition.



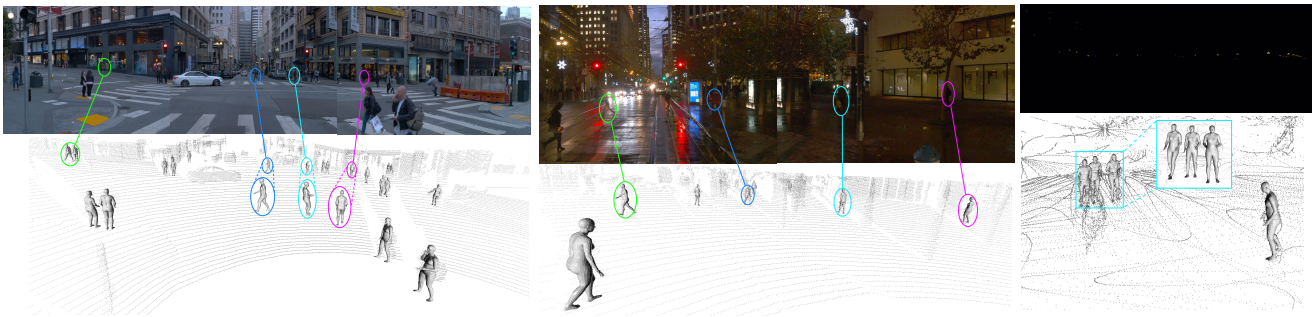
(a) Incomplete points. (b) Noise points. (c) Sparse points.

(d) Proposed pipeline.

Figure 1. (a)-(c): Three challenges exist in 3D human mesh reconstruction from LiDAR point cloud. (d) The proposed pipeline to overcome these challenges. Point cloud is fed into a pose regression network (PRN), to get an estimated template 3D pose (green arrows). Then template pose and the point cloud are input into the mesh reconstruction network (MRN) for coarse-to-fine reconstruction (red arrows). Different from that in [4], point cloud features run across the entire network (blue arrow).

Existing HMR methods based on point clouds mainly focus on dense point clouds derived from depth maps. These point clouds are typically complete and dense, and these models take the point clouds directly as input to estimate the SMPL pose and shape parameters. As illustrated in Figure 1, LiDAR point clouds have distinct characteristics: they are usually sparse, incomplete, and self-occluding, sometimes very noisy. These challenges make it very challenging to estimate human pose from these point clouds. We believe that prior information about the human body is necessary in such cases. Inspired by Pose2Mesh [4], we propose a sparse-to-dense reconstruction pipeline as illustrated in Figure 1d. We first introduce a pose regression network (PRN)

*Corresponding author.



(a) Day scene in Waymo. (b) Night scene in Waymo. (c) THU-MultiLiCa.

Figure 2. Three examples of LiDAR-HMR on multi-person scenes in Waymo [35] and THU-MultiLiCa [45] dataset. RGB images are not utilized but they are illustrated for better visualization. We utilize the annotated human position to gather local point clouds for mesh reconstruction. LiDAR-HMR can reconstruct accurate human meshes in different illumination conditions, especially for the scene illustrated in (c): very faint illumination.

as a backbone to extract template human 3D pose and corresponding point cloud features. Then, we utilize a mesh reconstruction network (MRN) which carries on these features to reconstruct a complete human mesh progressively.

Because the human body information contained in RGB images can be well represented by the estimated 2D/3D skeleton, previous methods [4, 25, 26, 29] did not consider the original image input information during the process of reconstructing human meshes. In contrast, estimated human poses from sparse point clouds are often not accurate enough, and the reliable 3D positional representation of the original point cloud can also provide local semantic information of the human body surface, which is difficult to describe by the human skeleton. Hence in our proposed pipeline, point cloud features run across the entire network as illustrated in Figure 1d. We propose a cascaded graph transformer structure to more efficiently utilize the local semantic information of the point cloud. During the process of progressively reconstructing the human mesh, the features of the point cloud are dynamically adjusted according to the resolution of the mesh, aiming to explore more local surface information of the point cloud.

Conventional point cloud-based algorithms [30, 41] for human pose estimation often voxelize the point cloud to obtain voxel-level 3D features. However, it is unintuitive to directly use these voxel-level features for surface reconstruction of the human body, and 3D CNN backbones include more redundant calculations. To address these issues, we propose a lightweight point cloud transformer backbone based on probabilistic modeling. This backbone significantly reduces calculations while achieving similar performance. Importantly, the extracted point cloud features can be directly used for subsequent human mesh reconstruction modules, which enables end-to-end multi-task training of the entire network. Fortunately, we found that this multi-task training strategy provides significant improvements for both pose estimation and human mesh recovery tasks.

As illustrated in Figure 2, the proposed method LiDAR-HMR can handle various types of outdoor scenes, and compared with the RGB-based mesh reconstruction method, it is not affected by illumination, which emphasizes the significance and application prospect of LiDAR-HMR. Experimental results on three public datasets demonstrate that LiDAR-HMR not only achieves the best performance in the task of human mesh recovery but also achieves superior performance in human pose estimation.

2. Related Work

In recent years, Human Pose Estimation (HPE) has emerged as a prominent research topic, encompassing various areas of study, including 2D HPE [3, 40], 3D HPE [13, 30, 31, 39, 41, 43, 46], and Human Mesh Recovery (HMR) [17, 20, 26, 33]. In this section, we focus on 3D HPE methods based on point cloud input. Additionally, we provide a summary and review of 3D HMR methods based on RGB image inputs, which are similar and offer valuable insights for reference.

3D HPE from point cloud. 3D HPE from depth images is a long-standing research topic [9, 11, 14, 19, 30, 34, 42]. However, depth map-based methods can only be used in indoor scenes, which limits their robustness and application scenarios. Recently, with the proposal of 3D human pose databases for point cloud scenes [6, 24, 35], some 3D pose estimation algorithms based on lidar point clouds have emerged. Zheng et al. [47] first proposed a multi-modal 3D HPE network to fuse RGB images and point clouds, using 2D labels as weak supervision for 3D pose estimation. In follow-up work, Weng et al. [37] used simulation data to train a transformer-based attitude estimation network and then proposed a symmetry loss to fine-tune with the actual LiDAR point cloud input. Effect. Ye et al. [41] proposed a multi-task structure, and used the task of object detection as pre-training. They used a transformer as a point cloud encoder to regress human poses and achieved state-of-the-art effects on the Waymo [35] database. These methods have

successfully demonstrated the feasibility of estimating human body information from sparse LiDAR point clouds.

3D Human mesh reconstruction. Human body mesh reconstruction can generally be divided into parameterized and non-parameterized methods. Most previous works [16, 17, 21, 23, 27, 36] used parameterized human body models, such as SMPL [28], and focused on using the SMPL parameter space as the regression target. Given pose and shape coefficients, SMPL is stable and practical for creating human meshes. However, as discussed in the literature [4, 22, 32, 44], accurately estimating the coefficients from the input data is not intuitive enough. Instead of regressing parametric coefficients, non-parametric methods [4, 25, 26] directly regress vertices from the image. In previous studies, Graph Convolutional Neural Network (GCNN) [4, 22] is one of the most popular options because it is able to model local interactions between adjacent vertices. However, it is less effective at capturing global features between vertices and body joints. To overcome this limitation, METRO [25] proposed a set of transformer-based architectures to model the global characteristics of vertices. However, compared with GNN-based methods [4, 22], it is less convenient to model local interactions. Subsequent work, Mesh Graphormer [26], uses a combination of graph convolution and transformer to obtain better results.

In previous RGB-based methods, as RGB features are often abstract semantic features and lack sufficient 3D representation, during the process of regression and fine-tuning mesh, the original image features cannot be well encoded into the human body mesh reconstruction process. Different from this, the point cloud itself contains sufficient 3D occupancy information and reliable human body surface information. We input the point cloud into the mesh reconstruction process and use transformers to gradually adjust the reconstructed mesh to conform to the observation characteristics of the point cloud itself.

3. Method

The proposed LiDAR-HMR can be divided into a pose regression network (PRN) and a mesh reconstruction network (MRN). PRN uses PointTransformer-v2 as a feature extractor to encode point clouds, and it decodes point features to reconstruct a template human pose. Given the template pose and per-point features, MRN utilizes point clouds to reconstruct fine human mesh progressively.

3.1. Pose Regression Network

Probabilistic modeling. Taking the input point cloud $\hat{P}$ with n points $(p_0, p_1, ..., p_{n-1})$, we model the problem of estimating 3D pose J with m keypoints $(j_0, ..., j_{m-1})$ from point cloud $\hat{P}$ as:

$$\max_{\mathrm{J}} P(J|p_0, p_2, ..., p_{n-1}). \tag{1}$$

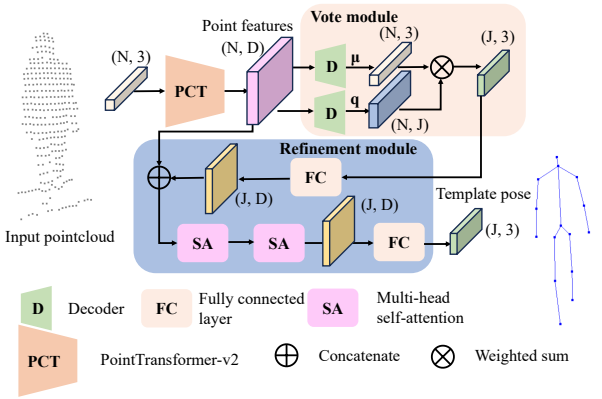


Figure 3. The overall structure of the proposed pose regression network (PRN). Input point clouds are encoded by PointTransformer-v2 and decoded into q and μ to get a predicted human pose. Then the predicted pose is fed into two layers of self-attention for refinement and completion. The shape of intermediate features is marked as (x, y). Specifically, N denotes the number of input points, J denotes the number of keypoints, and D denotes the fixed feature dimension for attention.

Without the connection relationship between key points, we have:

$$P(J|p_0, p_2, ..., p_{n-1}) = \prod_{k=0}^{m-1} P(j_k|p_0, p_2, ..., p_{n-1}). \tag{2}$$

With the hypothesis of the normal distribution and independent iso distribution, the probability distribution of j_k can be described as:

$$\begin{aligned} P(j_k|p_0, p_2, ..., p_{n-1}) &= \prod_{i=0}^{n-1} P(j_k|p_i) \\ &= \prod_{i=0}^{n-1} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(j_k-\mu_i)^2}{{\sigma_i}^2}}, \end{aligned} \tag{3}$$

where:

$$\begin{aligned} p_i &\in R_3, \\ d_i &\in R_3, \\ \mu_i &= p_i + d_i. \end{aligned} \tag{4}$$

d_i is pointwise offset, μ_i and σ_i are parameters. We use a Gaussian distribution to approximate the joint distribution:

$$\begin{aligned} P(j_k|p_0, p_2, ..., p_{n-1}) &= e^C \prod_{i=0}^{n-1} e^{-\frac{(j_k-\mu_{ik})^2}{{\sigma_{ik}}^2}} \\ &\approx e^C e^{\frac{(j_k-\hat{\mu}_k)^2}{{\hat{\sigma_k}}^2}}, \end{aligned} \tag{5}$$

$$\frac{(j_k-\hat{\mu}_k)^2}{{\hat{\sigma_k}}^2} = -\sum_{i=0}^{n-1} \frac{(j_k-\mu_{ik})^2}{{\sigma_{ik}}^2}, \tag{6}$$

where:

$$C = -\frac{n}{2} ln(2\pi) - \sum_{i=0}^{n-1} ln(\sigma_i). \tag{7}$$

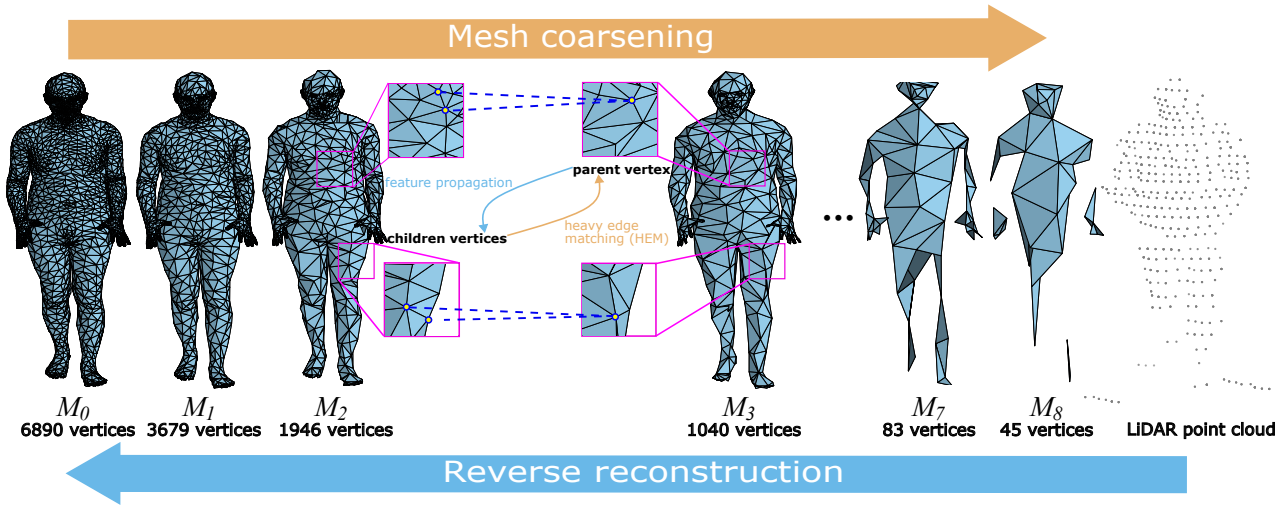


Figure 4. The mesh coarsening gradually generates multiple coarse graphs with heavy edge matching (HEM) following [4]. Specifically, we undergo reverse reconstruction with MRN which utilizes the parent-children relationship during coarsening. Vertex features are propagated following the parent-children edge to generate a higher-resolution mesh.

Let $q_{ik} = \frac{1}{\sigma_{ik}{}^2}$, we have:

$$\hat{\mu}_k = \frac{\sum_{i=0}^{n-1} q_{ik}\mu_{ik}}{\sum_{i=0}^{n-1} q_{ik}}. \tag{8}$$

Let $\sigma_{ik} \geq 1$ we get $q_{ik} \in (0, 1]$, it describes the confidence of p_i to vote for the position of keypoint j_k. Estimated model parameter $\hat{\mu} = (\hat{\mu}_0, \hat{\mu}_1, ..., \hat{\mu}_{m-1})$ denotes the estimated template pose J_0 with the maximized probability.

Estimate and complete human pose. The overall structure of PRN is shown in Figure 3. We utilize PointTransformer-v2 [38] as a per-point feature extractor and two decoders to regress μ_{ik} and q_{ik} respectively. This process is similar to the "vote" operation in VoteNet [7], hence we name it the vote module. We further find that due to the incomplete nature of the point cloud, one or more keypoints of the human body may not have been observed by the point cloud, leading to incomplete estimation of the template pose. To address this issue, we introduce a self-attention-based refinement module for completion and refinement, which consists of two self-attention layers. It is worth mentioning that the refinement module does not have any point cloud or corresponding feature inputs. Instead, it mainly relies on learned pose priors from the data.

Loss. We use the l2-loss to constraint the estimated pose J and groundtruth human pose J^{gt}:

$$L_J = ||J, J^{gt}||_2^2. \tag{9}$$

In order to directly constrain the per-point features, we constrain the output of vote module. Specifically, for each point p_i in the input point cloud, we can obtain the ground truth values $\check{\mu}_i$ and $\check{q}_i$ from the ground truth pose. Then we use L2 loss and cross-entropy loss to constrain them:

$$L_\mu = \sum_{i=1}^{N} ||\mu_i - \check{\mu}_i||_2^2, \tag{10}$$

$$L_q = \sum_{i=1}^{N} CrossEntropy(Q_i, \check{Q}_i), \tag{11}$$

where μ_i and Q_i is the estimated value, $Q_i = (q_{i1}, ..., q_{iJ})$, and $\check{Q}_i = (\check{q_{i1}}, ..., \check{q_{iJ}})$, J is the number of keypoints. The overall loss function for the proposed PRN is:

$$L_{PRN} = \lambda_J L_J + \lambda_p(L_\mu + L_q), \tag{12}$$

where λ_J and λ_p are hyper-parameters.

3.2. Mesh Reconstruction Network

As illustrated in Figure 4, we use Heavy Edge Matching to downsample the complete human body mesh, a total of 9 times, to obtain a set of human skeletal structures at different resolutions. The sparsest skeleton corresponds to 24 vertices, while the densest mesh possesses 6890 vertices. This process constitutes the transition from sparse to dense. MRN estimates the sparse vertices using the input template pose and point cloud features from PRN and reconstructs the complete human body mesh gradually. The overall structure of MRN is illustrated in Figure 5.

Graphormer for single resolution processing. The template pose is input into a fully connected layer that outputs sparse vertex features. Then we utilize the graphormer propagation module which consists of a point cloud-based graphormer [26] and a propagation module. The point cloud-based graphormer consists of a cascade of self-attention layers followed by a graph convolutional layer. It is utilized to process the relationships between the point cloud and the vertices of a mesh. The vertex and point cloud features are concatenated and input into a self-attention layer to introduce fine local point cloud features. Features of the mesh vertices after the self-attention module are then input into a graph convolutional layer to model the link relationship between vertices. It is noteworthy that the initial

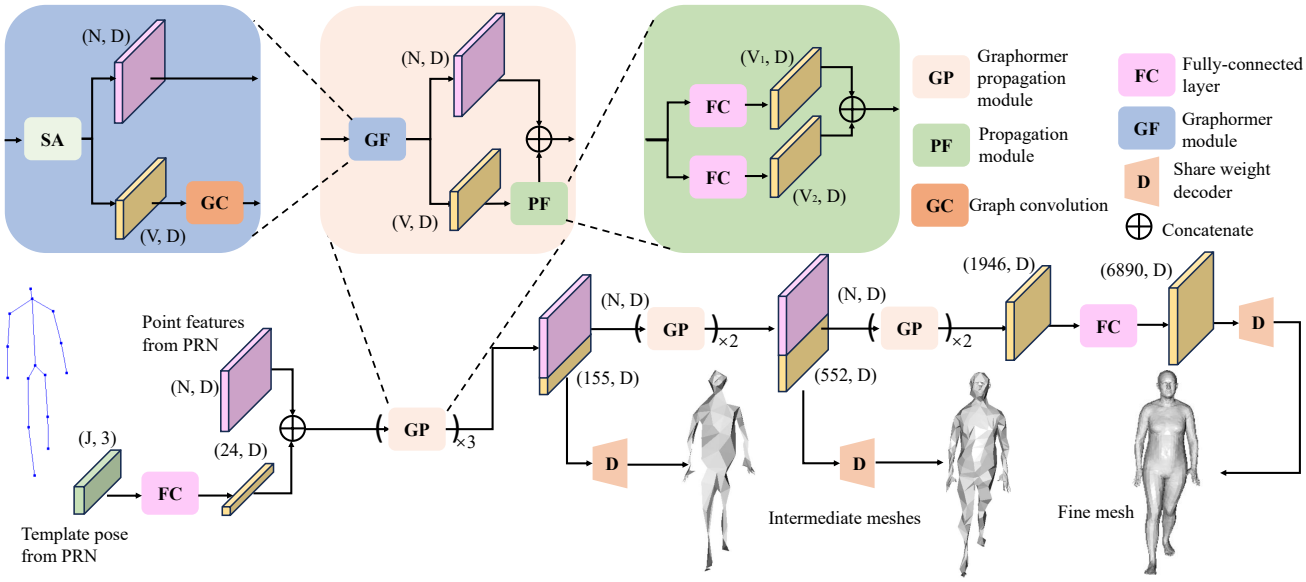


Figure 5. The overall structure of the proposed mesh reconstruction network (MRN). MRN receives point cloud features and estimated template pose from PRN and reconstructs the complete human mesh gradually. For each intermediate resolution, we utilize a point cloud-based graphormer [26] to introduce point cloud features during the reconstruction. Vertex features are inherited with a propagation module to model the parent-children relationship during the coarsening process. Finally, a fully connected layer is utilized to get the fine human mesh. The shape of intermediate features is marked as (x, y). Specifically, N denotes the number of input points, V denotes the number of vertices, and D denotes the fixed feature dimension for attention.

point cloud features are derived from the PRN, and the point cloud features are updated after the self-attention module.

Propagation module for inter-resolution conduction. To obtain higher-resolution vertex features, Pose2Mesh [4] uses an upsampling approach, which, in this random way, does not consider the correspondence between vertices from different resolution meshes. We found that the correspondence and connecting relationships remain unchanged during the point cloud upsampling process. Specifically, as illustrated in Figure 4, each parent vertex has 1-2 child vertices in the finer resolution. We model the point features between different resolution meshes based on this correspondence, assuming that the feature of a child vertex is obtained by a specific offset from its parent vertex feature. This is consistent with the inverse process of downsampling. We use two different fully connected layers to model these offsets between the relationships of different resolution meshes, which is the propagation module.

For the first seven resolutions, we use one graphormer module and one propagation module for upsampling. For the last two resolutions with 3679 and 6890 vertices correspondingly, due to the large numbers of vertices, using graphormer would result in excessive computational burden. Therefore, we use two layers of fully connected layers to obtain the fine human mesh. Notably, we use a unified decoder to decode all vertices features, enabling each resolution of mesh features to be decoded into the corresponding 3D coordinates. This ensures the consistency of the features learned by MRN at different resolutions. Furthermore, we can supervise not only the final mesh output but also every intermediate resolution.

Loss. Specifically, the final output mesh loss L_F is consistent with that in [4], which consists of vertex coordinate loss, joint coordinate loss, surface normal loss, and surface edge loss. In addition, we also apply a mesh loss based on intermediate resolution. Specifically, in intermediate resolution i (from 1 to 8), the vertex loss is defined as:

$$L_{vertex\_i} = ||M_i - \check{M}_i||_1, \tag{13}$$

where M_i is the estimated mesh and $\check{M}_i$ is the corresponding ground truth mesh. Then we can get the overall mesh loss for intermediate resolutions:

$$L_{inter} = \sum_{i=1}^{N} L_{vertex\_i}. \tag{14}$$

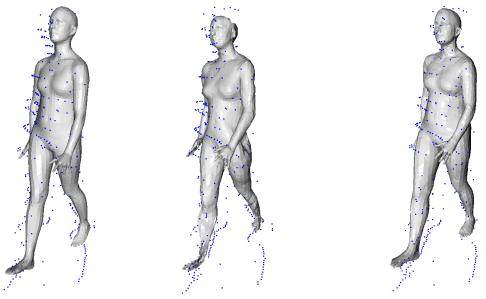
Finally, the entire network is trained end-to-end, so losses in PRN are also involved in training:

$$L_{MRN} = L_{PRN} + \lambda_F L_F + \lambda_i L_{inter}, \tag{15}$$

where λ_F and λ_i are hyper-parameters.

3.3. Implementation Details

The number of input points is sampled to 1024 and detailed information on network parameters can be found in the supplementary material. During training, we first train PRN alone with batch size 64 for 50 epochs to converge. LiDAR-HMR (PRN + MRN) is trained with batch size 8 for 30 epochs to converge. Network parameters are updated by Adam [18] and the learning rate is set to 5×10^{-4}.



Ground truth | MPVPE = 5.37cm MPERE = 0.135 | MPVPE = 7.92cm MPERE = 0.109

Figure 6. Two examples of estimated human meshes, and point clouds are illustrated in blue points. Meshes with low MPVPE may not have the best reconstruction quality. Therefore, we introduce the MPERE to measure the reconstruction results from another perspective.

4. Experiments

4.1. Datasets

Waymo open dataset [35] released the human keypoint annotation on the v1.3.2 dataset that contains LiDAR range images and associated camera images. We use v2.0 for training and validation. It possesses 14 keypoints annotation for each object. There are 8125 annotated human keypoints in the train set and 1873 in the test set.

Human-M3 dataset [8] is a multi-modal dataset that captures outdoor multi-person activities in four different scenes. It possesses 15 keypoints annotation for each object. There are 80103 annotated human keypoints in the train set and 8951 in the test set.

Due to the lack of 3D mesh annotation in Waymo and Human-M3 datasets, we used keypoints annotations and input point clouds to reconstruct the pseudo label of human mesh. This process is similar to Smplify-X [33], and more details are illustrated in the supplementary material.

SLOPER4D dataset [6] is an outdoor human motion capture dataset capturing several objects based on LiDAR point clouds and RGB images. Ground-truth meshes are obtained by motion capture devices. There are overall six motion fragments in the released part. As there is no manual assignment of train and test sets by the author, we selected a fragment as the test set. As a result, there are 24936 annotated human mesh in the train set and 8064 in the test set.

4.2. Metrics

For HPE, we utilize MPJPE [15] and PA-MPJPE [4, 12], and mean per vertex position error (MPVPE) for HMR. Besides, we propose a new metric mean per edge relative error (MPERE) for non-parametric methods evaluation. For non-parametric methods, position relationships between vertices are not fixed and the MPERE can represent the reconstruction quality of local details.

Specifically, for a predicted mesh M with edge length set $(l_1, ..., l_m)$, and ground-truth mesh $\check{M}$ with edge length set $(\check{l}_1, ..., \check{l}_m)$, the MPERE is defined as:

$$MPERE = \sum_{i=1}^{m} \frac{1}{m} \frac{||l_i - \check{l}_i||_1}{\check{l}_i}, \tag{16}$$

where m is the number of edges in the mesh.

MPERE measures the ratio of the length error to the ground-truth length of mesh edges and judges the reconstruction quality of short edges in dense parts more efficiently. As illustrated in Figure 6, sometimes MPVPE is not enough to measure the reconstruction quality. In this case, MPERE is needed as an additional measure.

4.3. Comparison to the State-of-the-art

For 3D HPE, we compare PRN with V2V-Posenet [30] and LPFormer [41]. For 3D HMR, we compare Pose2Mesh [4], SAHSR [16] and our-implemented Mesh Graphormer [26] with point cloud attention.

Specifically, Pose2Mesh requires a well-estimated 3D human skeleton, hence we utilize the estimated human pose by V2V-Posenet, which is called "V2V+P2M". For a fairer comparison, we also utilize our proposed PRN (well-trained) to concatenate with the MeshNet in the Pose2Mesh network for end-to-end training (the same as LiDAR-HMR), which is called "PRN+P2M".

The quantitative evaluation results are illustrated in Table 1. In particular, for **HPE**, PRN has achieved comparable results to state-of-the-art methods in different datasets, but with significantly lower computational requirements. They rely on the voxelization of point clouds and 3D CNN for extracting 3D features, which results in some computational redundancy and consumes more computational resources. Furthermore, the 3D features extracted by 3D CNN do not provide significant assistance in the subsequent human mesh recovery. Besides, the reconstruction performance of "V2V+P2M" is weaker than the comparative group "PRN+P2M". This also demonstrates the effectiveness of the proposed PRN network.

For **HMR**, Mesh Graphormer and SAHSR did not achieve satisfactory results. Both methods directly regress the human body mesh from point clouds without sparse-to-dense modeling. In particular, Mesh Graphormer is a non-parametric approach that lacks constraints on edge lengths, resulting in higher MPVPE and MPERE values than other methods. Given its excellent performance in estimating human mesh from RGB images, this also highlights the differences and challenges in 3D HMR from point clouds. SAHSR, on the other hand, is a parameterized method that faces difficulties in effectively modeling the relationship between point clouds and corresponding human meshes. For the more challenging datasets like Waymo and Human-M3,

Table 1. Evaluation results on SLOPER4D, Waymo and Human-M3 dataset. Metrics include MPJPE (cm), MPVPE (cm), MPERE, and computation cost while training when batch size is set to 1 (GFLOPs). The proposed methods are shown in blue and the best values are shown in bold.

		SLOPER4D			Waymo			Human-M3			
	Model	MPJPE	MPVPE	MPERE	MPJPE	MPVPE	MPERE	MPJPE	MPVPE	MPERE	GFLOPs
Pose	V2V-Posenet	**5.07**	-	-	7.03	-	-	8.30	-	-	61.803
	LPFormer	7.71	-	-	**6.39**	-	-	**7.75**	-	-	62.197
	PRN	5.70	-	-	6.78	-	-	8.22	-	-	**0.672**
Mesh	Graphormer	7.71	9.23	1.689	8.05	9.83	1.760	8.79	11.65	1.943	9.15
	SAHSR	7.26	8.12	**0.085**	9.66	11.68	0.163	10.55	13.15	0.291	**0.427**
	V2V+P2M	**5.07**	5.98	0.126	7.03	10.84	0.160	8.30	10.56	0.109	65.559
	PRN+P2M	5.66	6.53	0.132	8.74	9.04	0.150	8.06	8.96	0.091	4.428
	LiDAR-HMR	5.103	**5.189**	0.094	**6.28**	**8.24**	**0.119**	**7.76**	**8.95**	**0.088**	2.225

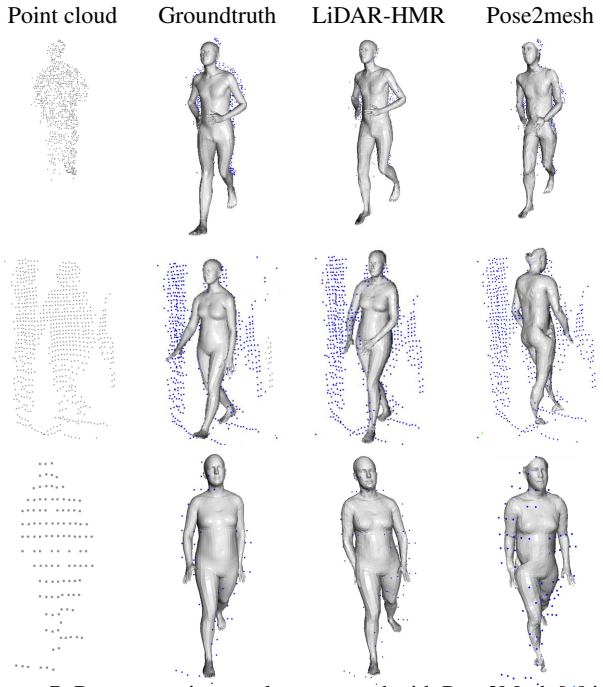


Figure 7. Reconstruction results compared with Pose2Mesh [4] in Waymo and SLOPER4D dataset. Meshes are visualized with the input point cloud (blue points) for better comparison. The three rows from top to bottom show examples corresponding to three different challenges: incomplete point clouds, noise, and sparse point clouds.

larger errors result in unsatisfactory MPERE scores. Comparing the above two methods highlights the effectiveness of the proposed sparse-to-dense reconstruction pipeline.

The **effect of MRN** is obvious, with the same backbone and training conditions, MRN outperforms the MeshNet proposed in the Pose2Mesh on three datasets while consuming fewer computational resources. Although both methods are based on a similar pipeline of sparse-to-dense modeling, MRN takes the point cloud as input to assist in the reconstruction. Additionally, constraints are applied to the intermediate reconstruction results, making the entire reconstruction process more rational and effective. In the autonomous driving scenario (Waymo), MRN achieves lower MPVPE and MPERE than the Pose2Mesh network by 0.8 cm and 0.031, accounting for 8.85% and 20.67% improvements, respectively. Furthermore, the MPJPE error is the best among all methods, not only outperforming the result of PRN but also surpassing LPFormer and V2V-Posenet, which is attributed to the multi-task training framework.

In general, the proposed LiDAR-HMR achieves the best results with fewer computing resources under three databases and different settings. Except for Figure 2, more visual results can be seen in supplementary materials

Table 2. Ablation evaluation of PRN on Waymo dataset. Metrics include MPJPE (cm), PA-MPJPE (cm), and computation cost while training when batch size is set to 1 (GFLOPs). The best values are shown in bold.

Group	MPJPE	PA-MPJPE	GFLOPs
no_vote	6.89	**5.07**	0.751
no_refine	7.05	5.62	0.475
no_all	10.14	7.31	**0.347**
no_voteloss	6.94	5.09	0.672
PRN	**6.78**	5.20	0.672

4.4. Ablation Study

As the Waymo dataset is the only dataset for autonomous driving, we conduct the ablation study on the Waymo dataset.

For PRN, we compare the following different settings: (1) Without the vote module, output features of PointTransformer-v2 are fed into a fully connected layer to directly output the human pose. (called "no_vote") (2) The refinement module is removed. (called "no_refine") (3) With the condition of "no_vote", the refinement module is removed. (called "no_all") (4) The loss term of L_μ and L_q are remove from L_{PRN} in equation 12 (called "no_voteloss").

Quantitative evaluation results are illustrated in Table 2. Compared with the "no_all" setting, the introduction

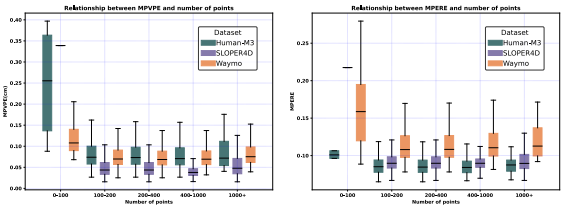
Table 3. Ablation evaluation of LiDAR-HMR on Waymo dataset. Metrics include MPJPE (cm), MPVPE (cm), MPERE, and computation cost while training when batch size is set to 1 (GFLOPs). The best values are shown in bold.

Group	MPJPE	MPVPE	MPERE	GFLOPs
no_pcd	7.05	9.05	0.145	**2.103**
upsample	6.49	8.34	0.127	2.164
no_PRN	42.59	28.03	0.306	2.225
no_pretrain	6.62	8.53	0.128	2.225
no_mid	6.53	8.89	0.123	2.212
edge_mid	6.31	8.31	0.165	2.225
LiDAR-HMR	**6.28**	**8.24**	**0.119**	2.225

of the vote module result in an MPJPE improvement of 3.09cm and a PA-MPJPE improvement of 1.69cm. Similarly, adding the refinement module can also bring an improvement of 3.25cm and 2.24cm respectively. It is worth noting that both the refinement module and the vote module can achieve good results when used alone. They obtain better human pose from the two aspects of "more accurate estimation" and "more accurate adjustment" respectively. Even so, the introduction of the vote module can still reduce the computing consumption of the network. The vote loss term in equation 12 has a slight benefit on the performance of PRN, and it is more natural and direct for the supervised backbone to learn point-by-point features.

For MRN, we compared the following different settings: (1) The point cloud is not fed into the reconstruction process of MRN, and the self-attention modules at each resolution only calculate the correspondence between vertex features. (called "no_pcd") (2) Different resolutions in the MRN network use upsampling (consistent with Pose2Mesh) instead of feature propagation. (called "upsample") (3) PRN no longer participates in end-to-end training, and the loss function does not include the pose estimation loss of PRN. (called "no_PRN") Or PRN is not pre-trained and initialized with random weights. (called "no_pretrain") (4) The loss function does not include the mesh reconstruction loss at intermediate resolutions (called "no_mid"), or the reconstruction loss at intermediate resolutions not only constrains the corner point positions but also introduces constraints on edge lengths (L_F term in Section 3.2) (called "edge_mid").

Quantitative evaluation results are illustrated in Table 3. The **effect of using point clouds** is obvious. Compared to setting "no_pcd", the MPVPE and MPERE decreased by 0.81 cm and 0.026 respectively, accounting for 8.95% and 17.93% reductions. This demonstrates the effectiveness of the proposed strategy of integrating point clouds into the reconstruction process. **End-to-end training** is essential for MRN. When the backbone does not participate in end-to-end training, the performance of MRN drops drastically, which indicates that the features learned from pose estimation pre-training cannot represent 3D mesh reconstruction.



(a) MPVPE (b) MPERE

Figure 8. Analysis of the relationship between performance and number of points in three public datasets.

Therefore, weight updating is important for the backbone of MRN training. Regarding the **upsampling strategy**, the proposed approach of feature propagation showed improvements over the original upsampling method, with MPVPE and MPERE decreasing by 0.1 cm and 0.008. As for **intermediate mesh losses**, it was observed that supervising the mesh at intermediate resolutions has a slight impact on the algorithm's performance. This may be attributed to providing the model with a more reasonable reconstruction process at intermediate stages, facilitating better learning. Interestingly, imposing constraints on the mesh length at intermediate resolutions was found to have a mildly negative effect on the model's performance, possibly due to excessive constraints during the intermediate reconstruction process leading to reduced freedom in the final mesh reconstruction.

The number of input points also affects the performance of LiDAR-HMR. As shown in Figure 8, the performance of LiDAR-HMR is relatively stable when the number of points exceeds 100. When the number of points is less than 100, the algorithm exhibited significant fluctuations in MPVPE and MPERE on the Human-M3 and Waymo datasets, respectively. The performance of LiDAR-HMR is indeed affected when input points are sparse, and problems under such circumstances are more challenging. We found that the three public datasets have a limited number of samples with sparse points (less than 100), which may be a contributing factor to this phenomenon as the learning process is not sufficiently comprehensive in handling such cases. This issue can be addressed through methods like extracting challenging samples. Limitations and Failure cases are illustrated in the supplementary material.

5. Conclusion

We present the first attempt for 3D human pose estimation and mesh recovery based on sparse LiDAR point clouds. We solve this problem by proposing a sparse-to-dense reconstruction pipeline. Taking advantage of the data characteristics of the point cloud, we introduce the input point cloud to assist the whole reconstruction process, which imposes constraints on the intermediate results of the reconstruction and achieves good results. We hope that our work will inspire subsequent work on sparse point clouds or multi-modal human perception tasks.