We sincerely thank all reviewers for their comments and for acknowledging that the *clear and strong motivation, straightforward yet powerful key insight, effectiveness on different baselines* (R1), *significantly reduced computational cost while achieving competitive performance* (R3), *the novel Token Reduction Supervision* (R2), *extensive and comprehensive evaluations* (R1, R3). We will incorporate all your feedback into the revised version.

[R1] How GTR or ITR affects the MVE? Token Reduction typically brings information loss, causing an accuracy drop, though it is slight. For MPVE, using an HRNet-W64 backbone, GTR typically causes a larger error increase of 2.6 mm (from 84.1 to 86.7), while ITP is 1.5 mm (from 86.7 to 88.2). This suggests GTR sacrifices more accuracy for efficiency, while ITP has less impact. We'll include more detailed discussions and visualizations in the revision.

[R1] Novelty of GTR. The GTR essentially captures the insight that using a compact representation (e.g., skeletal joints) together with a corresponding shape regressor can lead to significant improvement in efficiency, especially for structured shapes like human bodies. This insight is not explored in the relevant methods and might be beneficial to various Transformer-based tasks to enhance efficiency.

[R1] Can we regress SMPL parameters as GTR? We conducted this experiment in Tab 1. Although directly regressing SMPL parameters as improves efficiency, it greatly reduces accuracy compared to the proposed NSR for GTR. This is because regressing rotations from image features is more difficult than spatial positions, as verified in HybridIK.

Table 1. Comparison between different approaches for GTR. We test on FastMETRO with EfficientNet-b0 (Eb0) as a backbone.

Method	GFLOPs↓	Throughput↑	MPJPE↓	PAMPJPE↓
Parameter regression for GTR	1.2	998.3	77.3	53.2
NSR for GTR (Ours)	1.4	870.4	**63.2**	**43.9**

[R1] Can NSR be pre-trained? Not really. Because NSR relies on the learned features from the main Transformer rather than for training, it can not be pre-trained separately.

[R1] Report overall memory footprint. For inference, memory usage when processing one image is saved by **28.6%**; For training, the memory usage of transformer encoder and encoder-decoder structures is saved by **58.3%** and **27.4%** in a standard training scheme (Appendix B.2).

[R2] Other related works in token clustering for HMR. There are clear differences between our and the mentioned works. For TCFormer [1], we have *1) different architectures*. TCFormer is a complicated multi-stage method for token clustering, while ours only requires a single pass; *2) Different body representation*. There is no consideration of body representation in TCFormer while ours adopts NSR for GTR; *3) different performance.*. Ours surpasses TCFormer, as shown later. Moreover, both [2] and [3] are **NOT** for token reduction, and they even do **NOT** use Transformers. Their ideas are to learn the dense correspondence between vertices and the image in UV space. Nevertheless, we compared with TCFormer in Tab 2, where ours surpasses TCFormer on both two datasets. We will cite all suggested papers. Thanks for improving this submission.

Table 2. TCFormer [1] v.s. TORE (Ours) for HMR on Human3.6M.

Method	Throughput↑	3DPW		Human3.6M	
		MPJPE↓	PAMPJPE↓	MPJPE↓	PAMPJPE↓
TCFormer [1]	230.9	80.6	49.3	62.9	42.8
METRO+TORE (Ours)	210.1	75.5	46.6	**57.6**	37.1
FastMETRO+TORE (Ours)	249.2	**72.3**	**44.4**	59.6	**36.4**

[R2] More comparisons with other token reduction methods. We compared with PPT [4] that prunes tokens by locating human visual tokens with attention score; see Tab 3 where ours is more competitive. Note the comparison with TransFusion [5] is not applicable as it targets a different goal, i.e., multi-view pose estimation (only key points).

Table 3. PPT v.s. TORE (Ours). We test with FastMETRO-Eb0 on Human3.6M.

Method	GFLOPs↓	Throughput↑	MPJPE↓	PAMPJPE↓
PPT	1.6	862.1	68.4	46.2
Ours	1.6	870.4	**63.2**	**43.9**

Table 4. Influence of pruning rates. We test with FastMETRO-Eb0 on Human3.6M.

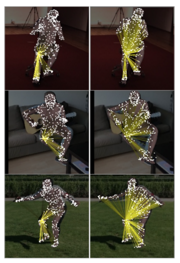
Metrics	No Pruning	0.2	0.5	0.75
PAMPJPE ↓	45.8	**43.9**	43.9	44.7
MPJPE ↓	69.2	**63.2**	64.2	65.3
GFLOPs ↓	7.1	1.6	1.4	1.2

[R2] "The throughout and flops look good, while MPVE/MPJPE/PMPJPES doesn't look amazing." A slight drop in accuracy has been widely accepted in the community when a method can significantly save computation costs; See the representatives like PPT and TCFormer. R1 and R3 also acknowledge this fact. Also note TORE exhibits higher accuracy in both body and hand mesh recovery with ResNet and Eb0 backbones, even with a massive efficiency improvement; See Tab 1, 2 and 8 of the main paper.

[R3] Practical impact of improvement in GFLOPs and Throughput. TORE effectively improves efficiency, leading to benefits like: (1) *Higher Parallelism.* A TORE-equipped model processes 870 images per second, while the number is 510 w/o TORE. (2) *Memory Saving.* The improvement allows for saving more memory costs by **32.6** from 13.2GB to 8.9GB compared with FM-EB0 (BS=32), loosening the requirements of GPU and devices.

[R3] How is the pruning rate determined? We empirically set = 20% based on extensive experiments. The influence of different pruning rates is shown in Tab 4.

[R3] More qualitative results of ablation studies. We show a visual comparison. W/ GTR+ITP, the joint-vertex attention is similar to the blending weights in SMPL which properly captures the shape structure. However, the model w/o GTR+ITP redundantly correlates local joints with distant vertices, leading to additional interaction costs. We will add more results in the revision.



[1] Not all tokens are equal:Human-centric ... token clustering transformer.

[2] 3D human mesh regression with dense correspondence.

[3] The best of both worlds:combining model-based ... body estimation.

[4] Transfusion:Cross-view fusion with ... for 3d human pose estimation.

[5] PPT:Token-pruned pose transformer for ... human pose estimation.