

VoxelKP: A Voxel-based Network Architecture for Human Keypoint Estimation in LiDAR Data

Supplementary Material

A. Technical Details

This section presents additional technical details of the network, loss functions, and metrics used.

A.1. Supplementary Network Details

The architecture of the stem module and prediction heads is presented in Figs. 7 and 8. The stem module includes CONV-BN-ReLU blocks with skip connections to extract low-level features. It contains one downsampling layer to obtain a smaller feature map. The model uses seven prediction heads. These heads predict: 1) the size of the bounding box, 2) the rotation of the bounding box, 3-5) the location of the box center and keypoints along the x, y, and z axes, 6) the visibility of keypoints, and 7) the Intersection over Union (IoU). Notably, we incorporate the IoU prediction to enhance performance, following [9].

A.2. Relationship to Prior Works

We implement the training pipeline for HPE on top of *OpenPCDet* [34], and we use sparse convolution operators from *sconv* [4]. Our *VoxelKP* fundamentally differs from the traditional LiDAR-based methods that predominantly focus on object detection or semantic segmentation tasks. We choose *VoxelNeXt* [2], a fully sparse network for 3D object detection, as the baseline architecture. To enhance the keypoint localization accuracy, our approach differs from *VoxelNeXt* in two aspects: 1) we employ a dual-branch solution where an additional context branch is used to preserve the global spatial details, and 2) we enhance spatial awareness by projecting absolute 3D coordinates to 2D BEVs. The effectiveness of the proposed modules is supported by the ablation results in Tab. 2. Technically, unlike previous point-voxel blocks such as *PVCNN* [19] and *PVT* [43], our spatial-context blocks are fully based on sparse voxels, without the need for voxelization and devoxelization within each building block. Essentially, the context branch captures per-voxel features to mitigate the loss of global context during successive convolutional blocks to ensure that each voxel retains a comprehensive understanding of its surroundings, thereby enhancing the accuracy and robustness of keypoint detection. In contrast to *PVT*'s box-attention strategy, which inefficiently handles dense tensors by repeatedly converting between dense and sparse representations, we take advantage of the fully sparse architecture for efficient processing.

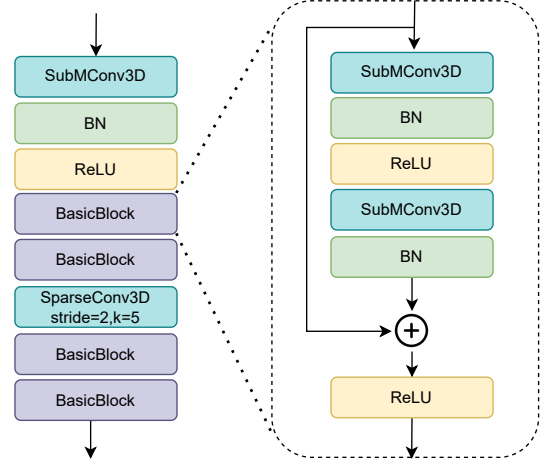


Figure 7. The architecture of the stem module.

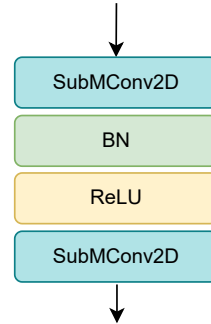


Figure 8. The architecture of prediction heads.

A.3. Losses

We use three types of losses in our work including the skeleton loss. Notably, the ground truth annotations are converted into the same sparse representation as the predictions for loss computation.

Heatmap Loss Our network outputs a set of heatmaps, one per class. This heatmap encoding allows our model to classify and localize objects in 3D space simultaneously. In the training phase, we assign positive heatmap indices based on ground truth annotations. Specifically, we identify the voxel closest to the annotated bounding box center and mark that voxel with a positive heatmap value. We supervise these heatmaps using an adapted focal loss function [2, 14, 18]. With the annotated and predicted heatmaps

I and \hat{I} , we have:

$$FL(I, \hat{I}) = \frac{-1}{N} \sum_{c=1}^C \sum_{v=1}^V \begin{cases} (1 - \hat{I})^\alpha \cdot \log(\hat{I}), & \text{if } I = 1 \\ \log(1 - \hat{I}) \cdot \hat{I}^\alpha \cdot (1 - I)^\beta, & \text{otherwise} \end{cases}, \quad (3)$$

where N , C , V are the batch size, number of channels, and number of voxels, respectively. α and β are the hyper-parameters to weigh each voxel. We use $\alpha = 2$ and $\beta = 4$ in this work, following [14].

L1 Regression Loss We adopt a simple L1 loss for other prediction heads of coordinates and keypoint visibilities. With the ground truth and predicted values Y and \hat{Y} , we have:

$$L1(Y, \hat{Y}) = \frac{1}{N} \sum_{c=1}^C \|Y - \hat{Y}\|_1. \quad (4)$$

A.4. Metrics

We use mean per-joint position error (MPJPE), pose estimation metric (PEM), and object keypoint similarity (OKS) to evaluate our method. Formally, let $\hat{Y} \in \mathbb{R}^{J \times 3}$ be the predicted keypoints of a human, $Y \in \mathbb{R}^{J \times 3}$ be the ground truth, and $v_j \in [0, 1]$ be the visibility of each joint j . The MPJPE metric is defined as:

$$\text{MPJPE}(Y, \hat{Y}) = \frac{1}{\sum_j v_j} \sum_{j \in [J]} v_j \|y_j - \hat{y}_j\|_2. \quad (5)$$

Note that MPJPE requires a one-to-one match between the keypoint predictions and the ground truth. Therefore, a Hungarian matching is performed to match the predicted and annotated keypoints before calculating the MPJPE.

PEM further takes into account the matching accuracy that is essentially a sum of the MPJPE over visible matched keypoints with a penalty for unmatched keypoints. Note that the unmatched keypoints include both the ground truth keypoints without matching predicted keypoints and the predicted keypoints without matching ground truth objects.

$$\text{PEM}(Y, \hat{Y}) = \frac{\sum_{i \in M} \|y_i - \hat{y}_i\|_2 + C|U|}{|M| + |U|}, \quad (6)$$

where M is a set of indices of matched keypoints, $|U|$ is a set of indices of unmatched keypoints, and $C = 0.25$ is a constant penalty for an unmatched keypoint.

Additionally, we include the classic metric of OKS in this work. The OKS metric is not computed per keypoint, it is a relative metric computed for each human body. In OKS, each ground truth object also has a scale s , which we define as the square root of the object segment area. OKS is computed as the arithmetic average across all labeled keypoints in an instance.

$$\text{OKS} = \frac{\sum_j e^{-\frac{d_j^2}{2s^2k_j^2}} v_j}{\sum_i v_i} \quad (7)$$

where d_j is the Euclidean distance between each corresponding ground truth and detected keypoint, k_j is a per-joint constant provided by COCO [17]. The reported OKS@KP is averaged over multiple OKS values, which are calculated for OKS thresholds starting at 0.50, increasing in steps of 0.05, and ending at 0.95.

B. Additional Results

B.1. The Fundamental Discrepancy

As mentioned in our introduction, in this experiment, we observe that when using a frozen pretrained detection model, simply extending the detection head with **trainable** keypoint heads may fail to converge during training. However, when we gradually unfreeze the model starting from the last layers, the training loss begins to decrease, as shown in Fig. 9. This behavior suggests that the features learned for object detection do not contain fine details beneficial for keypoint estimation.

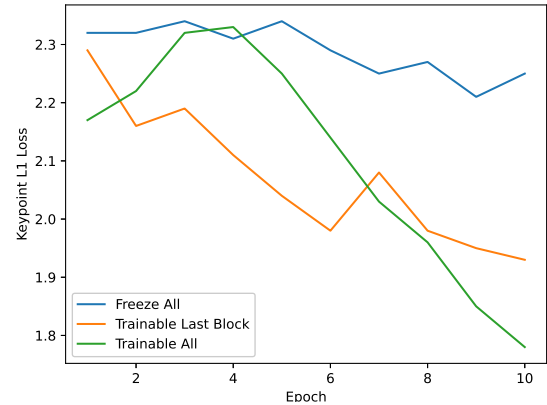


Figure 9. We show the training loss for the first 10 epochs under different settings for freezing the backbone.

B.2. Full Evaluation

We report the full spectrum of the evaluation, including MPJPE, OKS@AP, and PEM. The details for each metric can be found in Appendix A.4.

part	Head	Shoulders	Elbows	Wrists	Hips	Knees	Ankles	All
MPJPE	0.0570	0.0669	0.0948	0.1467	0.0670	0.0820	0.1084	0.0887
OKS@AP	0.6393	0.8917	0.7197	0.3791	0.9533	0.8586	0.7581	0.7300
PEM	0.1569	0.1563	0.1746	0.1987	0.1576	0.1660	0.1765	0.1695

Table 7. Full evaluation of *VoxelKP*.

B.3. Zero-shot evaluation

To the best of our knowledge, Waymo remains the only publicly available dataset for LiDAR-based human keypoint estimation. For a better understanding of the performance of the proposed method, we further perform zero-shot evaluation on *SLOPER4D*, a human motion dataset, by extracting keypoints from the SMPL model of the tracked human.

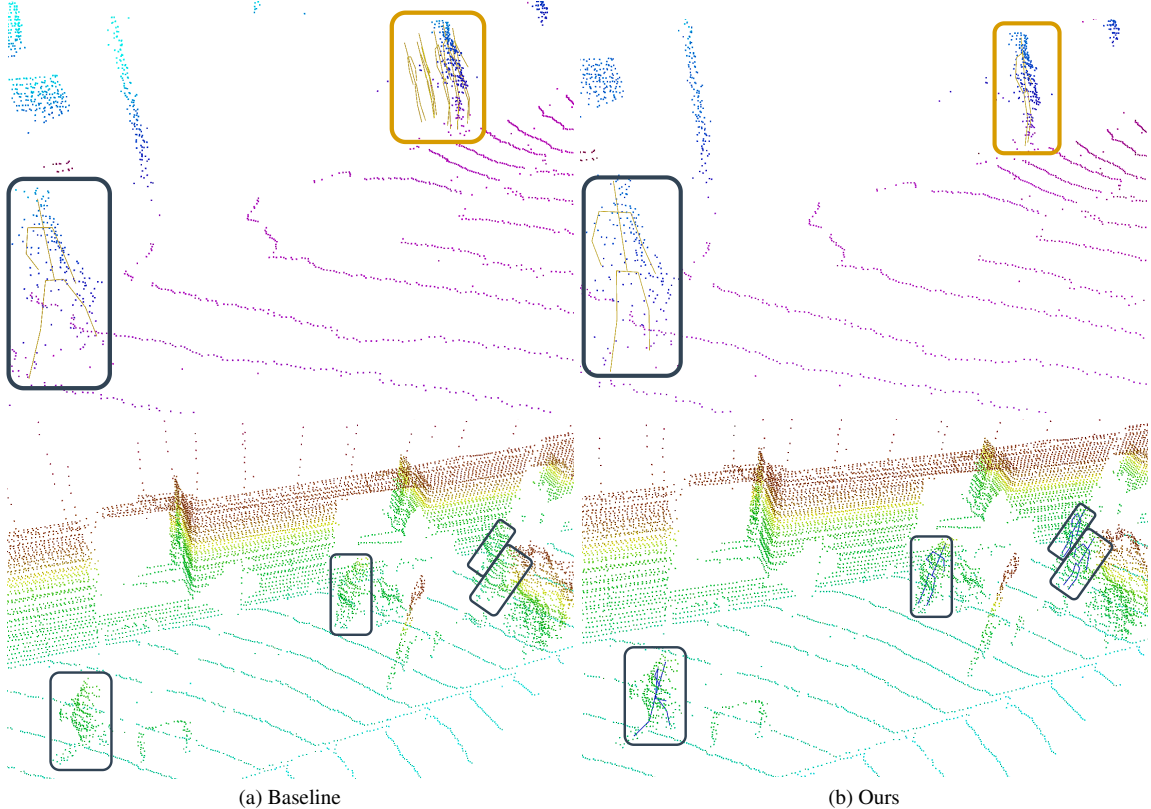


Figure 10. Additional visualization results. Our method detects the human objects that the baseline method fails to detect.

Since the dataset is not yet fully released by the time of publication, we use only the first 500 frames of three street-scene video sequences from the dataset. This setting is very challenging because 1) the annotation protocol is different from Waymo, and 2) the point clouds do not cover the full body.

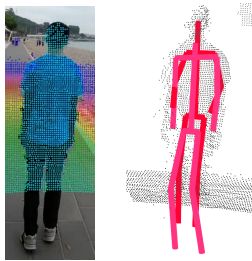


Figure 11. Visual demonstration of the results for the *SLOPER4D* dataset. Left: RGB input along with the captured points. Right: Pose estimation results, where the pink pose is GT and the red pose is from our inference.

B.4. Ablations

As reported in Sec. 3.2.1, each SSK module consists of two sparse 3D convolution branches: one uses $3 \times 3 \times 3$ kernels

No. Seq	003	008	009
Baseline	0.2034	0.2005	0.2090
VoxelKP	0.1819	0.1720	0.1603

Figure 12. Zero-shot results on *SLOPER4D* dataset. Due to the lack of other open-source methods for direct comparison, we compare our approach against our own baseline.

and another uses $5 \times 5 \times 5$ kernels. To assess whether further increasing the diversity of receptive fields improves performance, we conduct an ablation by adding a third branch with $7 \times 7 \times 7$ kernels. Our ablation shows that the use of three-way SSK lowers the performance. We also evaluate the effect of copy-and-paste augmentation. Our ablation indicates that this augmentation improves estimation accuracy.

	head	shoulders	elbows	wrists	hips	knees	ankles	all
SSK-three way	0.0513	0.0890	0.1410	0.1863	0.0771	0.1174	0.1848	0.1206
w/o augmentation	0.0647	0.0697	0.0982	0.1668	0.0693	0.1126	0.1828	0.1090
Ours	0.0570	0.0669	0.0948	0.1467	0.0670	0.0820	0.1084	0.0887

Table 8. Ablation study on the SSK configuration and copy-paste augmentation.