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

Jian Shi, Peter Wonka

King Abdullah University of Science and Technology

{jian.shi, peter.wonka}@kaust.edu.sa

Abstract

We present VoxelKP, a novel fully sparse network architecture tailored for human keypoint estimation in LiDAR data. The key challenge is that objects are distributed sparsely in 3D space, while human keypoint detection requires detailed local information wherever humans are present. We propose four novel ideas in this paper. First, we propose sparse selective kernels to capture multi-scale context. Second, we introduce sparse box-attention to focus on learning spatial correlations between keypoints within each human instance. Third, we incorporate a spatial encoding to leverage absolute 3D coordinates when projecting 3D voxels to a 2D grid encoding a bird's eye view. Finally, we propose hybrid feature learning to combine the processing of per-voxel features with sparse convolution. We evaluate our method on the Waymo dataset and achieve an improvement of 27% on the MPJPE metric compared to the state-of-the-art, HUM3DIL, trained on the same data, and 12% against the state-of-the-art, GC-KPL, pretrained on a $25\times$ larger dataset. To the best of our knowledge, VoxelKP is the first single-staged, fully sparse network that is specifically designed for addressing the challenging task of 3D keypoint estimation from LiDAR data, achieving state-of-the-art performances. Our code is available at https: //github.com/shijianjian/VoxelKP.

1. Introduction

Human pose estimation is a critical area of research with applications spanning computer vision, robotics, human-computer interaction, and augmented/virtual reality. Previous works [21, 30, 33] are mostly based on 2D images and videos. Compared to regular RGB input, LiDAR sensors provide detailed 3D structural information by measuring the distance to objects using laser light. Apart from its robustness under occlusion and illumination changes, LiDAR also offers privacy protection as it can not retain facial details. In recent years, significant progress has been made in 3D ob-

ject detection from LiDAR point clouds, with methods like PointRCNN [24], Part-A2 [25], and PV-RCNN [26] achieving impressive results, while human pose estimation from LiDAR is still an open research problem with much room for improvement. Typically, object detection methods focus on capturing objects scattered sparsely across the 3D space while the keypoints tend to be distributed densely within localized regions around the human body. This fundamental discrepancy in the context captured by existing detectors limits their suitability for precise 3D keypoint prediction due to the lack of fine-grained spatial information. To address this gap, we aim to extend the success of 3D object detection to 3D keypoint estimation for Lidar point cloud data by introducing novel components to preserve fine-grained spatial information.

This work identifies the importance of learning local dense features to capture the intricate spatial relationships between keypoints for precise human pose estimation. For this purpose, we introduce the *VoxelKP* architecture. *VoxelKP* is a novel, fully sparse neural network tailored specifically for human keypoint estimation within LiDAR point clouds. It combines local feature extraction and global context modeling to achieve accurate human pose prediction from LiDAR scans. To be specific, we introduce four key components that play a pivotal role in enhancing local feature learning for keypoint estimation:

- **Sparse Selective Kernel (SSK) Modules**: These modules are designed to selectively aggregate multi-scale 3D features efficiently extracted at sparse voxel locations. By employing various receptive field kernels and a selection mechanism, the SSK modules significantly improve spatial context. This is crucial for the accurate estimation of keypoints, as it allows the model to understand the spatial relationships between keypoint locations.

- **Sparse Box-Attention Modules**: Our approach incorporates localized box-based self-attention to partition the sparse voxel space into non-overlapping box regions. This strategy enables the model to capture dependencies between voxels within each box. By doing so, it extracts fine-grained local features necessary for resolving

densely distributed keypoints. This focused modeling of intricate spatial relationships between keypoints is instrumental in achieving precise human pose estimation.

- **Spatially Aware Multi-Scale BEV Fusion**: To retain the 3D spatial relationships between keypoints, we introduce a spatially aware multi-scale bird's eye view (BEV) fusion technique. This innovative approach encapsulates 3D spatial information into 2D representations, thereby improving the accuracy of keypoint estimation. It ensures that the model considers spatial information when predicting keypoints, enhancing the overall performance.
- **Hybrid Feature Learning** In addition to the above components, we propose the use of hybrid feature learning. We combine the results of two parallel branches in the architecture: per-voxel computations using MLPs and sparse convolutions that process voxel-neighborhoods.

To the best of our knowledge, *VoxelKP* is the first single-staged, fully sparse network that is specifically designed for addressing the challenging task of 3D keypoint estimation from LiDAR data, achieving 27% on the MPJPE metric compared to the current state-of-the-art trained on the same data.

2. Related Work

2.1. Deep Learning on Point Clouds

Many neural network architectures have been adapted for processing point clouds. Earlier methods like VoxNet [20] applied 3D CNNs to voxel grids for object classification. PointNet [22] was one of the first works to operate directly on point clouds using MLPs and max pooling to extract global features of entire scenes represented by point clouds. Follow-up works like PointNet++ [23] introduced hierarchical and localized feature learning. Meanwhile, another branch of works such as PointCNN [14] and KPConv [32] introduced novel convolutional operators for learning features on the unordered point clouds, overcoming the limitations of typical convolutions for this irregular data type.

Typical LiDAR-generated point clouds contain more than $100,000$ points, making point-by-point computations overwhelming due to the massive data scale. Voxel-Net [41] proposed a voxel feature encoding (VFE) layer as a workaround for the high computational and memory issues brought by point-by-point computations. Meanwhile, sparse and submanifold sparse convolution operations [4] exploit sparsity in the voxel grid to reduce computations. SECOND [35] introduced an efficient sparse convolutional approach that benefits from the sparse operations. Following SECOND, subsequent works like Point-Pillars [11], 3DSSD [36], PV-RCNN [26], CenterPoint [38] further advanced sparse convolutional detection on point clouds, introducing ideas like pillar encoding for faster detection, multi-scale detection stacks with anchor boxes,

shared voxel encoders, and detecting small objects by center points. VoxelNeXt [2] further demonstrates a fully sparse voxel-based method without sparse-to-dense conversion or NMS post-processing. However, these approaches are targeted at improving bounding box localization accuracy, which does not require fine-grained spatial features for precise keypoint estimation tasks. Instead, We propose *VoxelKP*, a novel sparse convolutional architecture tailored for learning discriminative local features from sparse LiDAR data for accurate human pose estimation.

2.2. Human Pose Estimation on Point Clouds

Human pose estimation has been extensively studied in images, with methods like DeepPose [33], Stacked Hourglass [21], and HRNet [30] achieving high accuracy on benchmarks like COCO-wholebody [8]. However, compared to RGB images, point clouds provide explicit 3D structural information about the shape and depth of objects. Shotton *et al.* [27] pioneered point cloud human pose estimation from a single depth image. Recent works such as [19, 42] proposed a deep learning-based 3D human pose estimation from depth images. Waymo [31] has released keypoint annotations for LiDAR-collected point cloud scenes, while only 3% of the frames are annotated with keypoint human poses. Due to the scarcity of the keypoint annotations within LiDAR point cloud data, many works have taken semi-supervised or weak-supervised approaches to compensate for the limited availability of labeled 3D pose data. For example, some works [39, 40] took a multi-modal approach to utilize the enriched image annotations to assist the recognition from point clouds. Weng *et al.* [34] proposed an unsupervised approach that generates pseudo ground truth without using annotated keypoint data, along with a fine-tuning approach that pretrains the model with synthetic data then fine-tunes on the training set. A concurrent work [37] adopted a fine-tuning strategy that used a frozen backbone pretrained on a large-scale dataset as a feature extractor, achieving plausible performance. In general, multi-person pose estimation from sorely point clouds remains relatively unexplored due to the lack of ground-truth 3D human pose annotations. This work proposes a single-staged keypoint estimation method with only LiDAR point clouds, achieving comparable performances without extra training data.

3. Method

LiDAR point clouds typically contain sparsely distributed objects that occupy only small regions of the full 3D space. While the distribution of humans in space is sparse, in contrast, human keypoints require dense information wherever a human is present. To handle this density variation, we aim to improve feature learning in the regions where keypoints need to be located and detailed information is required. In

this section, we first present the formulation of the task, then introduce the key components proposed in our network, and finally elaborate on the details of the network architecture.

3.1. Problem Formulation

Given a 3D point cloud scanned by LiDAR sensors, our goal is to estimate the 3D locations of K keypoints that represent the human pose. Let the input point cloud P be $\mathbb{R}^{N \times C}$ where N is the number of points and C is the number of features (*e.g.* x, y, z, intensity, elongation). We use a sparse voxel representation to represent point clouds, which consists of two separate tensors: one feature tensor $\mathbb{R}^{V \times C}$ and one index tensor $\mathbb{R}^{V \times 4}$ where V is the number of non-empty voxels and 4 dimensions are used for batch sample index and the three coordinates of each voxel. We define the ground truth pose for the ith human as a set of 3D keypoint locations $G_i = \{g_i^1, g_i^2, ..., g_i^K\}$ where $g_i^k \in \mathbb{R}^3$ is the location of the kth keypoint in the global coordinate frame. The set of K keypoints corresponds to anatomical joints of interest such as shoulders, elbows, wrists, hips, knees, and ankles. Our objective is to predict the 3D keypoint locations from the input point cloud, i.e. to learn a function F such that $\hat{G} = F(P)$, where $\hat{G} \in \mathbb{R}^{M \times K \times 3}$ is the tensor of predicted 3D keypoint locations of M humans.

3.2. Key Components

See Fig. 4 for our final architecture *VoxelKP*. First, the scene is voxelized into a sparse 3D grid. Then the sparse grid goes through multiple 3D blocks to extract multi-scale 3D sparse features, followed by a projection into a sparse 2D grid, and 2D blocks. Finally, multiple prediction heads output the keypoints. The proposed architecture contains four key components for enhancing the spatial localization accuracy of keypoints. Specifically, we employ spatially aware *sparse selective kernel modules* and *sparse box-attention modules* in our network to improve the representational power to encode and localize the fine-grained keypoint features. In addition, we use a *spatially aware multi-scale BEV fusion* method to encode the spatial information, along with a multi-scale fusion to understand the context across varying densities. Lastly, we use a *hybrid feature learning* approach to capture both fine-grained per-voxel details and relatively coarse-grained local neighborhood information.

3.2.1 Sparse Selective Kernel Module

Inspired by [5, 13], we propose the sparse selective kernel (SSK) module that selectively aggregates multi-scale features to improve spatial context. The SSK modules perform spatial attention on a 3D sparse voxel space, where the attention specializes the receptive field at each position using a data-driven kernel selection. As demonstrated in Fig. 1,
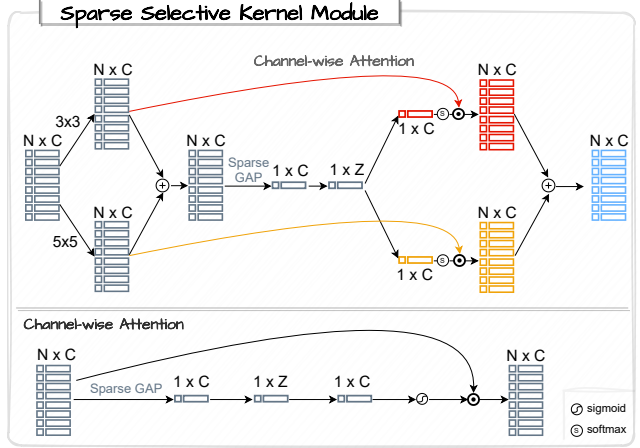

Figure 1. Sparse selective kernel module with one sample input. The SSK module selects the best kernels from different receptive fields with a softmax-based channel-wise attention mechanism.

we first generate a set of sparse 3D submanifold convolution kernels with varied receptive field sizes of $3 \times 3 \times 3$ and $5 \times 5 \times 5$. A submanifold convolution computes output values only if the convolution kernel is centered on a non-empty voxel, i.e., the number of non-empty voxels remains the same. These operations are applied to sparsely sampled voxel locations, extracting multi-scale features while remaining efficient. Next, the features from each kernel are fed into a selection module that compresses the spatial dimension by a global average pooling (GAP), and then a feature squeeze and expansion are applied. In our implementation, Z is 25% of C. The resulting tensor would then be used to weigh the features after a softmax activation. We denote a voxel position as p, the set of voxel positions within a voxel grid as P_s, and the feature corresponding to voxel position p as f_p. The sparse GAP $\bar{F}_s$ can be obtained by:

$$\bar{P}_s = \{(x_p, y_p, z_p) | p \in P_s\},$$
$$\bar{F}_s = \{\frac{1}{|S_{\bar{P}}|} \sum_{p \in S_{\bar{P}}} f_p | \bar{p} \in \bar{P}_s\}, \quad (1)$$

where $|S_{\bar{P}}|$ is the number of valid voxels for the sample s in a batch. This produces channel-wise attention weights, allowing the network to selectively emphasize or suppress each kernel's features. The multi-scale local features can then be obtained by combining weighted features from all kernels through averaging.

3.2.2 Sparse Box-Attention Module

We apply box-based self-attention. Unlike the previous works that tried to capture a wider range of global features with self-attention methods for segmentation tasks [9, 10], we focus on local feature extraction to resolve the densely
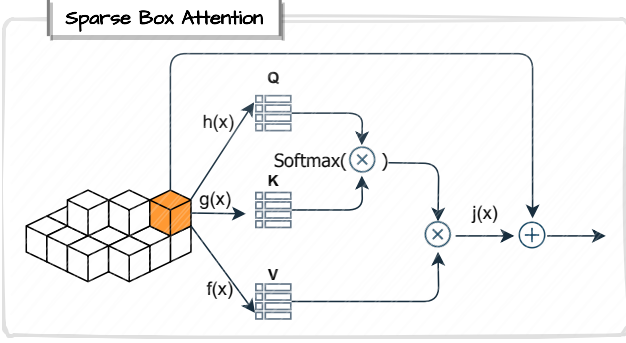
Figure 2. Sparse box-attention module. This attention mechanism selects the voxel features that correspond to one box partition referring to the index tensor and then performs self-attention on the selected voxels. The functions f,g,h, and j are linear layers.

distributed keypoints in local regions. The key idea is to partition the sparse 3D voxel space into non-overlapping boxes. Within each local box, we apply self-attention to capture dependencies between the voxels inside the box. The features in each box go through a linear layer for the queries Q, keys K, and values V, where $Q, K, V \in \mathbb{R}^{k_b \times h \times d}$ and k_b, h, d are the number of valid voxels in the b-th box, attention heads, and feature dimensions. Since we are using sparse tensor representations, each box partition may contain a varying number of voxels. Referring to [9], we then compute the attention map by the following equation:

$$\text{Attention}_{i,h} = \sum_{j=1}^{k_b} \text{softmax}(Q_{i,h} \cdot K_{j,h}) \times V_{j,h}. \quad (2)$$

We then further apply an additional projection layer on the obtained attention map, as shown in Fig. 2.

3.2.3 Spatially Aware Multi-Scale BEV Fusion

Compressing features into bird's eye view (BEV) maps is a common practice for object detection [1, 35]. For a sparse 3D voxel grid of size $C \times X \times Y \times Z$, we use C to denote the number of features per voxel, X and Y as the spatial extent in the ground plane, and Z as the up axis. Starting with a sparse 3D voxel grid, previous works such as [2] simply ignore the height information by summing the features of all voxels that share the same position on the ground plane (the same x and y coordinates). However, different from object detection tasks, height information is essential for keypoint estimation tasks to precisely locate each keypoint. A reasonable approach is to directly deploy 3D feature maps. Unfortunately, this direct 3D approach does not lead to a decent performance as training does not converge well, as shown in Tab. 4. We, therefore, propose a simple *spatially aware multi-scale BEV fusion* approach for fusing features

from multiple encoder layers in a way that retains spatial information, as illustrated in Fig. 3.

Height Encoding Transforming 3D data into BEV is often used in 3D object detection and segmentation tasks, for reducing the dimensionality of point clouds and making them more manageable for processing. An object detection method may project the 3D voxel grid to a 2D BEV representation by adding features from voxels that share the same x and y position, losing the information about which height a feature was taken from. Instead, we use a height encoding method. Specifically, we compress the height dimension to 1 using convolution kernels of size $(1, 1, h)$ where h is the height of each 3D voxel grid. Meanwhile, we increase the number of resulting channels to retain more spatial details and features from the 3D representation. This provides a richer representation for the 2D regression heads to work with.

Multi-scale Feature Fusion After obtaining z multi-scale height-encoded BEV maps from the last few stages of the network, we then fuse those feature maps to create a feature map that contains multi-scale features. Unlike working with dense tensors, the direct interpolation of the feature maps in the sparse case is computationally complex, as it requires specialized algorithms to efficiently navigate through the predominantly empty voxels to find and interpolate the adjacent non-empty voxels. Instead, we directly modify the feature position of the sparse tensor by multiplying the voxel position by its scale r. To avoid overlapping feature position of $(p_x * 2^r, p_y * 2^r)$ $r \in \{0, 1, 2...\}$ dur-
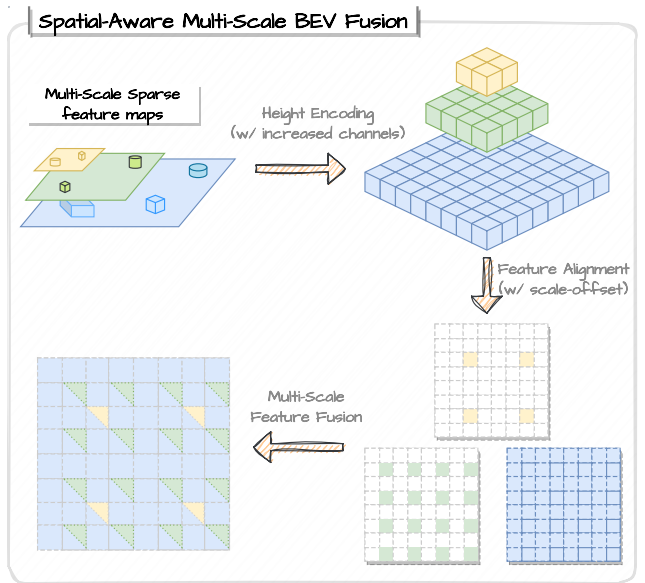

Figure 3. Spatially aware multi-scale BEV Fusion module. Note that we use a dense representation for a better visual illustration of the method.
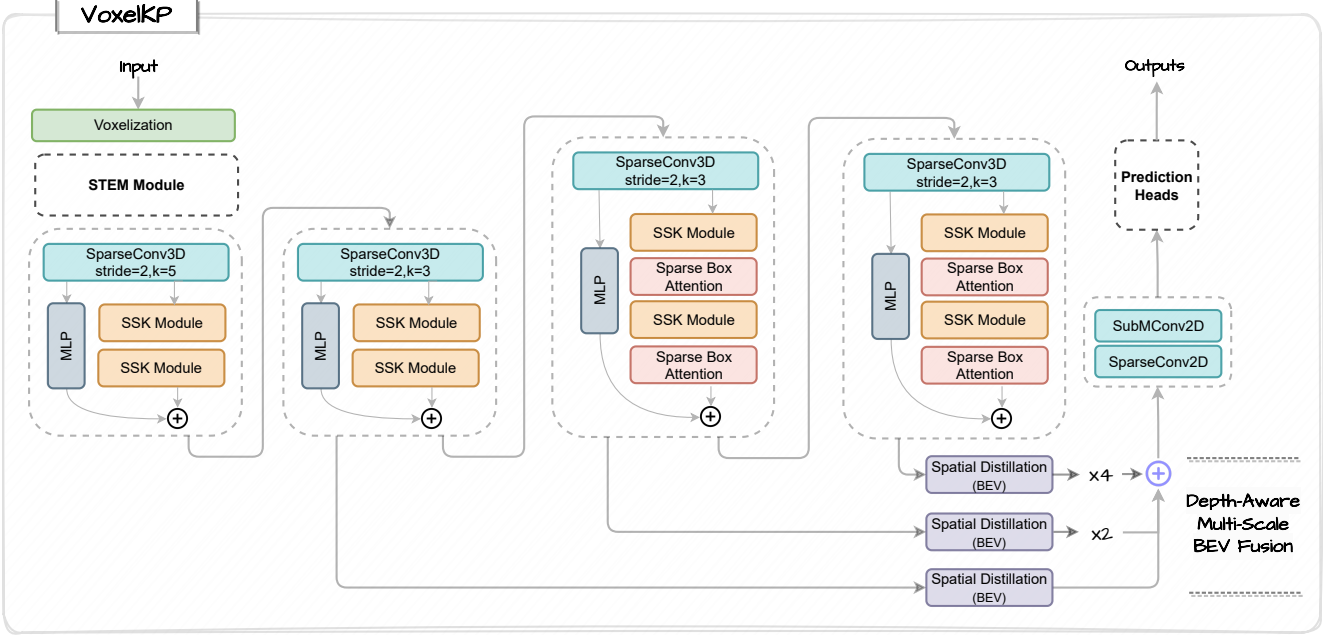
Figure 4. The overall architecture of the *VoxelKP*.

ing the scale multiplication, we align the xy-plane positions (p_x, p_y) using scale offsets $(p_x * 2^r + r, p_y * 2^r + r)$.

By stacking the r-scaled feature maps together, we obtain a multi-scale 3D feature map with a height of r. To obtain a BEV feature map, instead of collapsing with $1 \times 1 \times r$ convolutions, we simply apply an intuitive scaling for each scale of the feature map. The scaling factor is proportional to the height (scale) of the 3D feature map. The compressed sparse features $\bar{F}_c$ and their positions $\bar{P}_c$ are obtained as:

$$\bar{F} = \{f_p \cdot \hat{r}_p | p \in P_c\}, \quad \bar{P}_c = \{(x_p, y_p) | p \in P_c\},$$
$$\bar{F}_c = \{\sum_{p \in S_{\bar{p}}} \bar{f}_p \, | \bar{p} \in \bar{P}_c\}, \tag{3}$$

where $\bar{F}$ contains the scaled features by the scale offsets and $S_{\bar{p}} = \{p | x_p = x_{\bar{p}}, y_p = y_{\bar{p}}, p \in P_c\}$ contains voxels that are put onto the same 2D position $\bar{p}$. $\hat{r}_p$ is the normalized height position of each individual feature.

3.2.4 Hybrid Feature Learning

The convolutional operations focus on understanding spatial hierarchies and local geometric structures to extract local neighborhood information. Concurrently, inspired by the previous point-voxel networks [17, 26], we include an MLP branch for each stage. The integration of an MLP branch alongside a convolutional branch is a strategic approach to capture both fine-grained per-voxel details and relatively coarse-grained local neighborhood information. Each MLP branch is composed of three sequential blocks,

each consisting of a linear layer, batch normalization, and a ReLU activation function. The number of channels in each linear layer is set to match the channels of the incoming tensor. We then merge the output features from the MLP and convolutional branches through element-wise summation to create hybrid features of the per-voxel and per-neighborhood information. This hybrid feature learning approach is deployed to retain and process fine details across the voxel space, which is critical for the accurate localization of keypoints.

3.3. Network Architecture

We propose a single-stage, fully sparse neural network, designed for human pose estimation within LiDAR point clouds. The architecture is demonstrated in Figure 4. The input is a point cloud $\mathbb{R}^{N \times C}$ where N is the number of points and C is the number of features (*e.g.* x, y, z, intensity). We voxelize the point cloud into a sparse voxel representation. Our method consists of an input stem network and four stages with gradually decreased feature map size, where each stage reduces the spatial shape of the sparse voxel space by a factor of two. The input stem network is a simple stack of convolution layers, as shown in Appendix A.1, to extract low-level features from the voxelized point cloud. Next, we apply our proposed SSK modules in our next two stages to better capture the multi-scale local features. We further include window-based self-attention modules for our last two blocks to emphasize local-region features. Note that we do not increase the number of channels for the last three stages. For each stage, we further

5

Method		Dataset	Description	MPJPE cm.
With Extra Training Data				
Zheng *et al*. [40]	(CVPR 22)	Internal dataset + Waymo v.?	Trained on $155,182$ objects from internal data. Generated pseudo labels from 2D image labels.	10.80
GC-KPL [34]	(CVPR 23)	Waymo v.?	Pre-trained on synthetic data. Fine-tuned on ground truth	11.27
		Waymo v.?	Pre-trained on $200,000$ Waymo objects. Fine-tuned on ground truth	10.10
Without Extra Training Data				
HUM3DIL [39]	(CoRL 22)	Waymo v.1.3.2	Randomly initialized	12.21
VoxelKP		Waymo v.1.4.2	Randomly initialized	**8.87**

Table 1. Benchmark results. The numbers in the table are taken from their corresponding papers aside from *HUM3DIL*, which is taken from *GC-KP*L paper. It is unclear about the exact training dataset used for *Zheng* et al. and *GC-KPL*. Waymo v1.3.2 and Waymo v1.4.2 share the same data for keypoint estimation task.

include a side MLP branch for learning hybrid features. We then convert the resulting 3D feature maps from the last three blocks to 2D spatial-encoded BEV representations. Note that we increase the number of channels for the BEV representation to compensate for the information loss of the BEV conversion. These 2D features are further refined with 2D convolutions to aggregate spatial context. In the end, we obtain the estimated keypoints $Y_{kp} \in \mathbb{R}^{K \times 3}$ and the corresponding predicted visibilities $Y_{kp} \in \mathbb{R}^{K}$, where K is the number of keypoints.

4. Experiments

4.1. Implementation Details

Dataset We use the Waymo v1.4.2 dataset [31]. During the training, we merged "Pedestrian" and "Cyclist" classes together as a "Human" class. Note that there are only $8,125$ human examples with keypoint annotations whilst over 1 million bounding box annotations. We therefore removed the points inside those bounding boxes without keypoint annotations. Each human object is labeled with 14 3D keypoints (nose, left/right shoulders, left/right elbows, left/right wrists, left/right hips, left/right knees, and left/right ankles, head).

Network The architecture of the network is composed of a stem module followed by four stages, with output channels set to 64, 128, 256, 256, and 256, respectively. Given the high resolution (*e.g.* $1504 \times 1504 \times 61$) of the voxelized point cloud input, we employ larger sparse convolution kernels (kernel size $k = 5$) for the downsampling block in both the stem module and the initial stage. For the subsequent three stages, we revert to a smaller kernel size ($k = 3$). To compensate for the information loss in the BEV projection, we increased the channels from 256 to 384 during this process.

Training We use the point cloud range of the Waymo dataset as $(150.4m, 150.4m, 6m)$ and we transform them into voxel representations by a voxel size of

$(0.1m, 0.1m, 0.1m)$. We directly use the global keypoint locations without any encoding. Due to the limited number of training samples, we first apply a ground truth sampling technique [3, 35] to concatenate target objects from other frames into the sampled frames. Next, we apply global augmentations on the whole point cloud, including random flips on the x and y axes, random scale of the range of $[0.95, 1.05]$, and random rotation ranged from $[-\pi/4, \pi/4]$. Additionally, we apply local augmentations on each annotated object, including the random scale of the range of $[0.95, 1.05]$, random rotation ranged from $[-\pi/20, \pi/20]$, random frustum dropout [6] with an intensity range from $[0., 0.2]$, and random noise around the object. Our model is trained using AdamW [18] optimizer plus OneCycle [29] learning rate scheduler to mitigate overfitting [28]. Specifically, we use a learning rate of 0.003, weight decay of 0.01, and 0.9 momentum. Aside from the regular regression loss and heatmap loss, we include a skeleton regularization loss to make the model aware of the spatial relationships of keypoints. The details of the used loss functions can be found in Appendix A.2.

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

Table 2. Full evaluation of *VoxelKP*.

4.2. Benchmark Methods

There is a limited number of relevant research for this task. Most of the prior works utilize additional training data beyond the 3D keypoint data within the Waymo dataset. To provide a fair comparison, we need to consider approaches
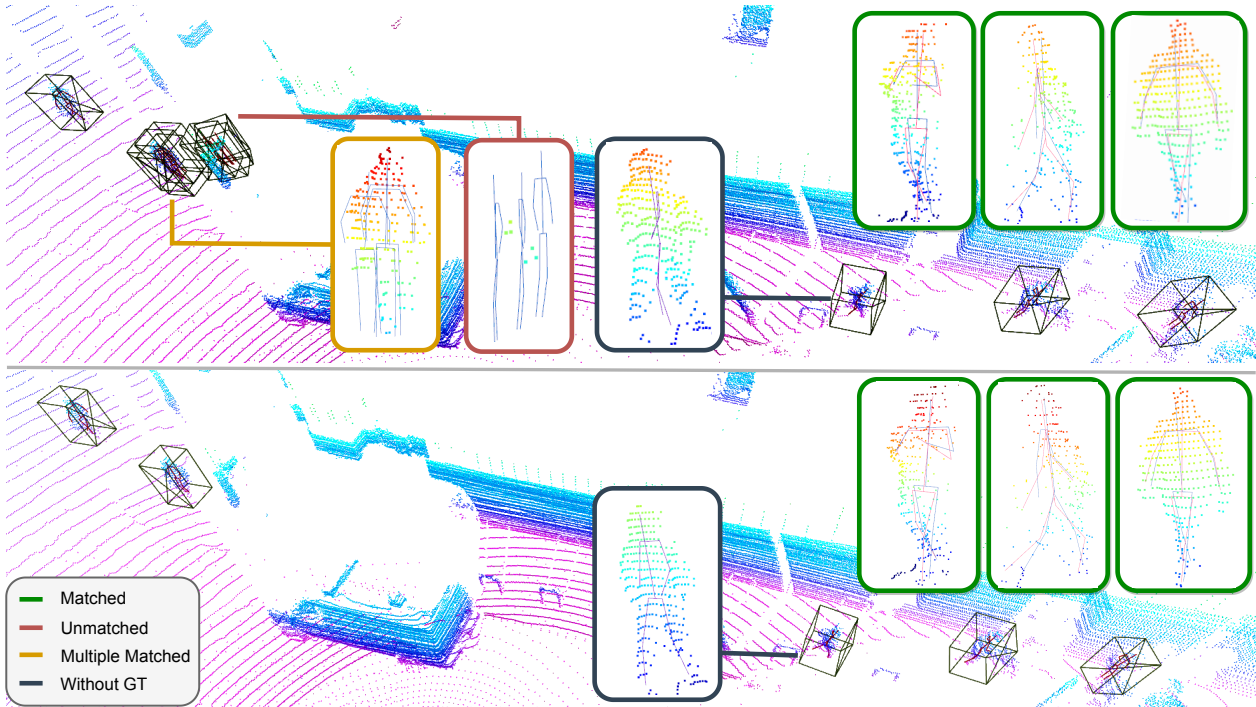
Figure 5. A visual demonstration of our baseline model (top) and the proposed *VoxelKP* (bottom). Our *VoxelKP* offers improved keypoint estimation with precise locations and fewer false positives. The insets are color-coded according to the legend in the figure. In the green-colored insets, a comparison with the ground truth is shown, with ground truth in red and predictions in blue.

that use extra data and those that rely solely on Waymo ground truth separately. Zheng *et al.* [40] adopted a pseudo-label generation approach to provide stronger supervision. It utilizes an internal dataset as training data and uses the Waymo dataset for evaluation. *GC-KPL* [34] pre-trains its backbone model with extra synthetic or real-world data, then fine-tunes the model with the full Waymo training set. Given the reliance on extra data in these methods, we consider the LiDAR-only version of *HUM3DIL* [39] as our primary competitor. *HUM3DIL* shares the exact same training data as our approach, allowing a direct comparison of techniques.

4.3. Results

Previous methods like *GC-KPL* use a subset of the validation data for evaluation, while we evaluate our method with the full validation set for better reproducibility. We report MPJPE on matched keypoints for our benchmark, following prior works. As shown in Tab. 1, we outperform the baseline *HUM3DIL* by approximately 27% in MPJPE. Our approach achieves state-of-the-art results among methods trained solely on Waymo ground truth. We also surpass the approaches leveraging extra synthetic data, beating *Zheng* et al. with synthetic pseudo labels by around 18% and *GC-KPL* with synthetic point clouds by about 21%. We achieve better performances as the SOTA *GC-KPL* ap-

proach which is pre-trained on $200,000$ real-world samples by about 12%. Overall, we demonstrate significant improvements over both the baseline solely using Waymo 3D keypoint data, as well as other techniques relying on extra data. A visual demonstration is presented in Fig. 5.

In addition, we report the full spectrum of the evaluation in Tab. 2, including MPJPE, OKS@AP, and PEM. The details for each metric can be found in Appendix A.3.

5. Ablations

We demonstrate the effectiveness of each proposed component in Tab. 3. We use the architecture of *VoxelNext* [2] as the baseline model, then gradually update the baseline model with the proposed components. We start with *VoxelNext* for two reasons: 1) it is one of the state-of-the-art point cloud object detection models with a fully sparse architecture design, and 2) it provides a good balance between computational costs and performance. We report the MPJPE for our ablations. The results indicate all the individual components can contribute to improving keypoint estimation. Next, we further present the ablation studies to show the alternative design choices of the individual component.

Spatially Aware BEV The use of the BEV representation significantly simplifies the detection problem by collapsing the 3D voxel space into a 2D feature map. This ablation

Components				MPJPE								PEM
Spatial BEV	SSK	Attention	Hybrid Feat.	head	shoulders	elbows	wrists	hips	knees	ankles	**all**	**all**
				0.0631	0.1486	0.2313	0.2395	0.1142	0.1431	0.1932	0.1612	0.2350
✓				0.0721	0.0995	0.1456	0.1904	0.0870	0.1277	0.1928	0.1304	0.2069
✓	✓			0.0603	0.0848	0.1232	0.1715	0.0759	0.1084	0.1608	0.1118	0.1889
✓	✓	✓		0.0558	0.0604	0.0903	0.1679	0.0620	0.1091	0.1834	0.1039	0.1791
✓	✓	✓	✓	0.0570	0.0669	0.0948	0.1467	0.0670	0.0820	0.1084	0.0887	0.1695

Table 3. Overall ablation for the effectiveness of each component.

evaluates the effectiveness of the proposed spatially aware BEV module. We first evaluate the direct use of a naïve 3D representations, followed by experiments with the spatially aware BEV. The findings, as shown in Tab. 4, indicate that our spatially aware BEV yields superior performance. The direct deployment of the 3D representation results in severe overfitting and, therefore, low performance. In addition, we also show that increasing the number of channels during the BEV projection can effectively improve the model performances, by compensating for information loss during projection. Overall, our spatially aware BEV strikes a balance that retains spatial acuity beyond basic BEV for resolving keypoint relationships while avoiding the complexity of full 3D convolutions.

	cp.	head	shoulders	elbows	wrists	hips	knees	ankles	all
3D	-	2.4620	2.4559	2.4492	2.4449	2.4394	2.4264	2.419	2.4422
Ours	✗	0.0688	0.0714	0.0982	0.1657	0.0723	0.1029	0.1595	0.1053
Ours	✓	0.0570	0.0669	0.0948	0.1467	0.0670	0.0820	0.1084	0.0887

Table 4. Ablation study for the spatially aware BEV module. *Cp.* denotes if to expand the number of channels to compensate for the information loss during the 2D projection.

Different Attention Mechanism This ablation study assesses the effectiveness of the box-attention mechanism within point cloud processing. Recent advancements, such as the stratified self-attention [9], focus on aggregating long-range contextual information, particularly beneficial for segmentation tasks. However, for keypoint estimation tasks, capturing global dependencies is less crucial. Instead, our approach utilizes local box-attention, which concentrates on adjacent local regions. The results, as presented in Tab. 5, demonstrate that local box-attention outperforms other methods. Interestingly, we found that the stratified attention mechanism could slightly impair performance. We suspect that the box-based approach concentrates on areas most relevant to each keypoint location, whereas long-range attention may cause the network to overlook local, dense details. As a result, the box-based attention mechanism allows efficient modeling of local keypoint distributions, without excessive computation or over-smoothing from global aggregation.

	head	shoulders	elbows	wrists	hips	knees	ankles	all
w/o	0.0659	0.0956	0.1405	0.1855	0.0831	0.1077	0.1515	0.1181
stratified	0.0650	0.0911	0.1347	0.1995	0.0819	0.1245	0.1919	0.1266
box	0.0570	0.0669	0.0948	0.1467	0.0670	0.0820	0.1084	0.0887

Table 5. Different self-attention methods. *w/o* denotes no attention applied.

6. Conclusion

In this work, we proposed a new 3D fully sparse neural network for estimating dense human poses from point clouds. Our method combines several novel components including sparse selective kernel layers, box-attention layers, spatially aware multi-scale BEV fusion, and hybrid feature learning to accurately predict human body keypoints. Experiments on the Waymo dataset demonstrate the advantages of our approach compared to prior art and we demonstrate improved performance compared to other approaches trained on the same data as well as other approaches trained with additional data.

Despite these advancements, we further identify certain areas for future exploration and improvement. As mentioned above, this work used a small volume of training data, but it could benefit from a larger-scale dataset. While we focus on single-frame point clouds, future work could leverage temporal information across sequences of LiDAR point clouds. Additionally, instead of the straightforward estimation of keypoints, future work may adopt inverse kinematics to include physical constraints on human body movement. Aside from refining estimated keypoint locations, this may especially be useful to handle real-world challenges such as occlusion within motion.

References

[1] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017. 4

[2] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Voxelnext: Fully sparse voxelnet for 3d object detection and tracking. In *Proceedings of the IEEE/CVF Con-*

ference on Computer Vision and Pattern Recognition, 2023. 2, 4, 7, 1

[3] Zehui Chen, Zhenyu Li, Shiquan Zhang, Liangji Fang, Qinhong Jiang, and Feng Zhao. Autoalignv2: Deformable feature aggregation for dynamic multi-modal 3d object detection. *ECCV*, 2022. 6

[4] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018. 2

[5] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 3

[6] Jordan SK Hu and Steven L Waslander. Pattern-aware data augmentation for lidar 3d object detection. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2703–2710. IEEE, 2021. 6

[7] Yihan Hu, Zhuangzhuang Ding, Runzhou Ge, Wenxin Shao, Li Huang, Kun Li, and Qiang Liu. Afdetv2: Rethinking the necessity of the second stage for object detection from point clouds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 969–979, 2022. 1

[8] Sheng Jin, Lumin Xu, Jin Xu, Can Wang, Wentao Liu, Chen Qian, Wanli Ouyang, and Ping Luo. Whole-body human pose estimation in the wild. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 2

[9] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8500–8509, 2022. 3, 4, 8

[10] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for lidar-based 3d recognition. In *CVPR*, 2023. 3

[11] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019. 2

[12] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, pages 734–750, 2018. 1

[13] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 510–519, 2019. 3

[14] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018. 2

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 3

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1

[17] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Pointvoxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, 2019. 5

[18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[19] Xiaoxuan Ma, Jiajun Su, Chunyu Wang, Hai Ci, and Yizhou Wang. Context modeling in 3d human pose estimation: A unified perspective. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6238–6247, 2021. 2

[20] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 922–928. IEEE, 2015. 2

[21] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, pages 483–499. Springer, 2016. 1, 2

[22] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2

[23] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2

[24] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019. 1

[25] Shaoshuai Shi, Zhe Wang, Xiaogang Wang, and Hongsheng Li. Part-aˆ 2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *arXiv preprint arXiv:1907.03670*, 2(3), 2019. 1

[26] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Pointvoxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10529–10538, 2020. 1, 2, 5

[27] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *CVPR 2011*, pages 1297–1304. Ieee, 2011. 2

[28] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018. 6

[29] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning*

for multi-domain operations applications, pages 369–386. SPIE, 2019. 6

[30] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019. 1, 2

[31] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 6

[32] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019. 2

[33] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014. 1, 2

[34] Zhenzhen Weng, Alexander S Gorban, Jingwei Ji, Mahyar Najibi, Yin Zhou, and Dragomir Anguelov. 3d human keypoints estimation from point clouds in the wild without human labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1158–1167, 2023. 2, 6, 7

[35] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2, 4, 6

[36] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11040–11048, 2020. 2

[37] Dongqiangzi Ye, Yufei Xie, Weijia Chen, Zixiang Zhou, and Hassan Foroosh. Lpformer: Lidar pose estimation transformer with multi-task network. *arXiv preprint arXiv:2306.12525*, 2023. 2

[38] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021. 2

[39] Andrei Zanfir, Mihai Zanfir, Alex Gorban, Jingwei Ji, Yin Zhou, Dragomir Anguelov, and Cristian Sminchisescu. Hum3dil: Semi-supervised multi-modal 3d humanpose estimation for autonomous driving. In *Conference on Robot Learning*, pages 1114–1124. PMLR, 2023. 2, 6, 7

[40] Jingxiao Zheng, Xinwei Shi, Alexander Gorban, Junhua Mao, Yang Song, Charles R Qi, Ting Liu, Visesh Chari, Andre Cornman, Yin Zhou, et al. Multi-modal 3d human pose estimation with 2d weak supervision in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4478–4487, 2022. 2, 6, 7

[41] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[42] Yufan Zhou, Haiwei Dong, and Abdulmotaleb El Saddik. Learning to estimate 3d human pose from point cloud. *IEEE Sensors Journal*, 20(20):12334–12342, 2020. 2

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 are presented in Figs. 6 and 7. 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 [7].
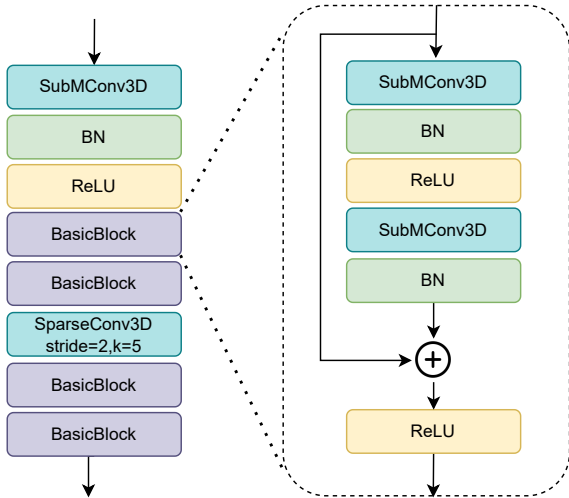

Figure 6. The architecture of the stem module.

A.2. 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
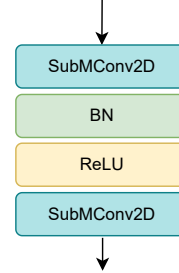

Figure 7. The architecture of prediction heads.

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, 12, 16]. With the annotated and predicted heatmaps I and $\hat{I}$, we have:

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

$$(4)$$

where N, C, V are the batch size, number of channels, and number of voxels, respectively. α and β are the hyperparameters to weigh each voxel. We use $\alpha = 2$ and $\beta = 4$ in this work, following [12].

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:

$$L1(Y, \hat{Y}) = \frac{1}{N} \sum_{c=1}^{C} ||Y - \hat{Y}||_1. \qquad (5)$$

Skeleton Regularization We propose to use a skeleton loss to encode prior information about the relative positioning of keypoints. For this purpose, we include bone length regularization in the loss function. This term computes the distance between the ground truth bone length and the predicted bone length. Specifically, given the ground truth keypoint locations Y and predicted keypoint locations $\hat{Y}$, we first compute the skeleton bone lengths $BL(Y)$ and $BL(\hat{Y})$ by calculating the Euclidean distance between connected keypoint pairs. The skeleton loss is then calculated as the Huber loss $h(\cdot)$ between the predicted bone lengths $BL(\hat{Y})$ and ground truth bone lengths $BL(Y)$, resulting in:

$$SK(\hat{Y}, Y) = h(BL(\hat{Y}), BL(Y)). \qquad (6)$$

This enforces the model to predict keypoint locations that respect the biomechanical constraints of bone lengths in the human skeleton. Matching the distribution of predicted bone lengths to the ground truth, ensures awareness of the spatial relationships between different joints. The skeleton loss penalizes predicted keypoints that violate the physical constraints of bone lengths, acting as a strong prior for plausible human poses.

A.3. 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:

$$\text{MPJPE}(Y, \hat{Y}) = \frac{1}{\sum_j v_j} \sum_{j \in [J]} v_j ||y_j - \hat{y}_j||_2. \quad (7)$$

Note that MPJPE requires a one-to-one match between the keypoints predictions and ground truth. Therefore, a Hun-garian 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.

$$\text{PEM}(Y, \hat{Y}) = \frac{\sum_{i \in M} ||y_j - \hat{y}_j||_2 + C|U|}{|M| + |U|}, \quad (8)$$

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
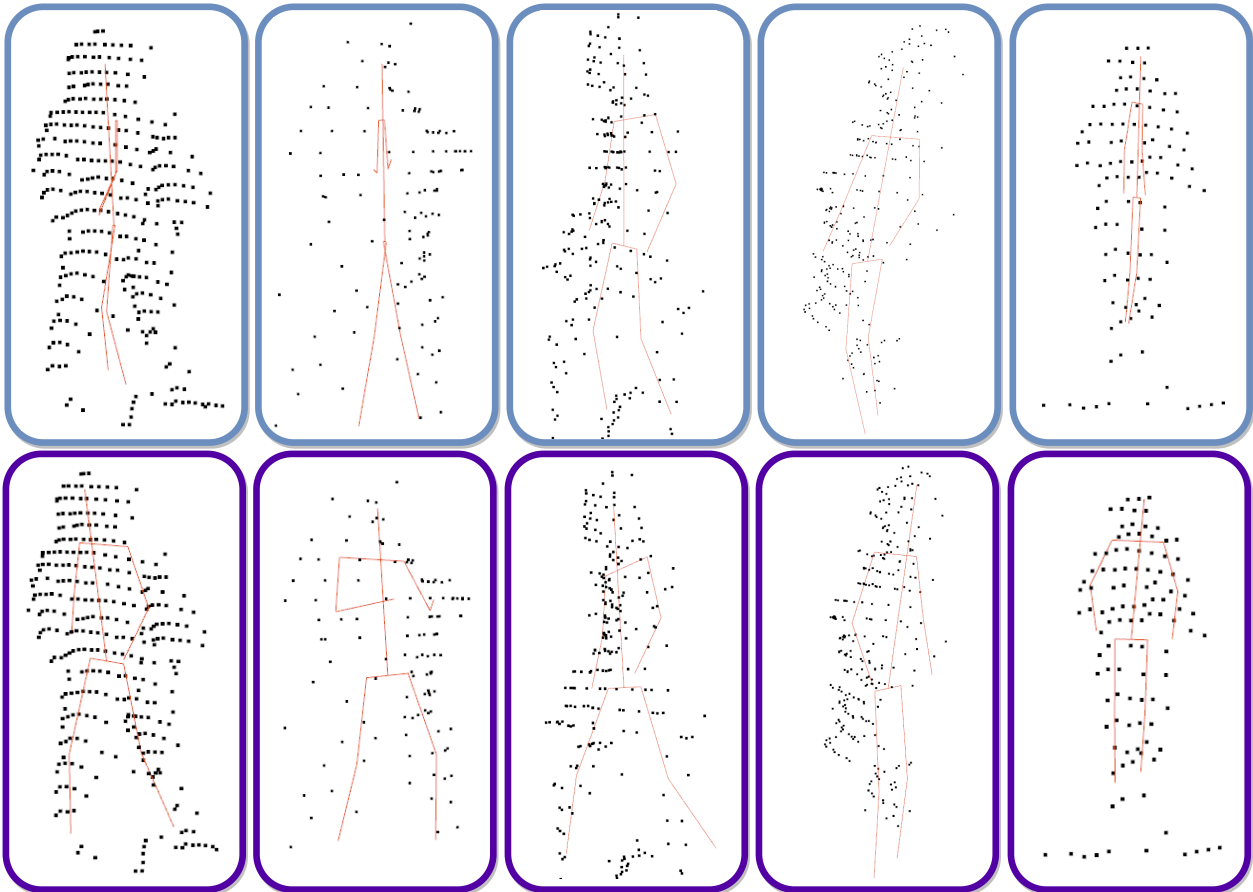

Figure 8. A visual demonstration of the baseline model (top row) and the proposed *VoxelKP* (bottom row) on matched human objects.
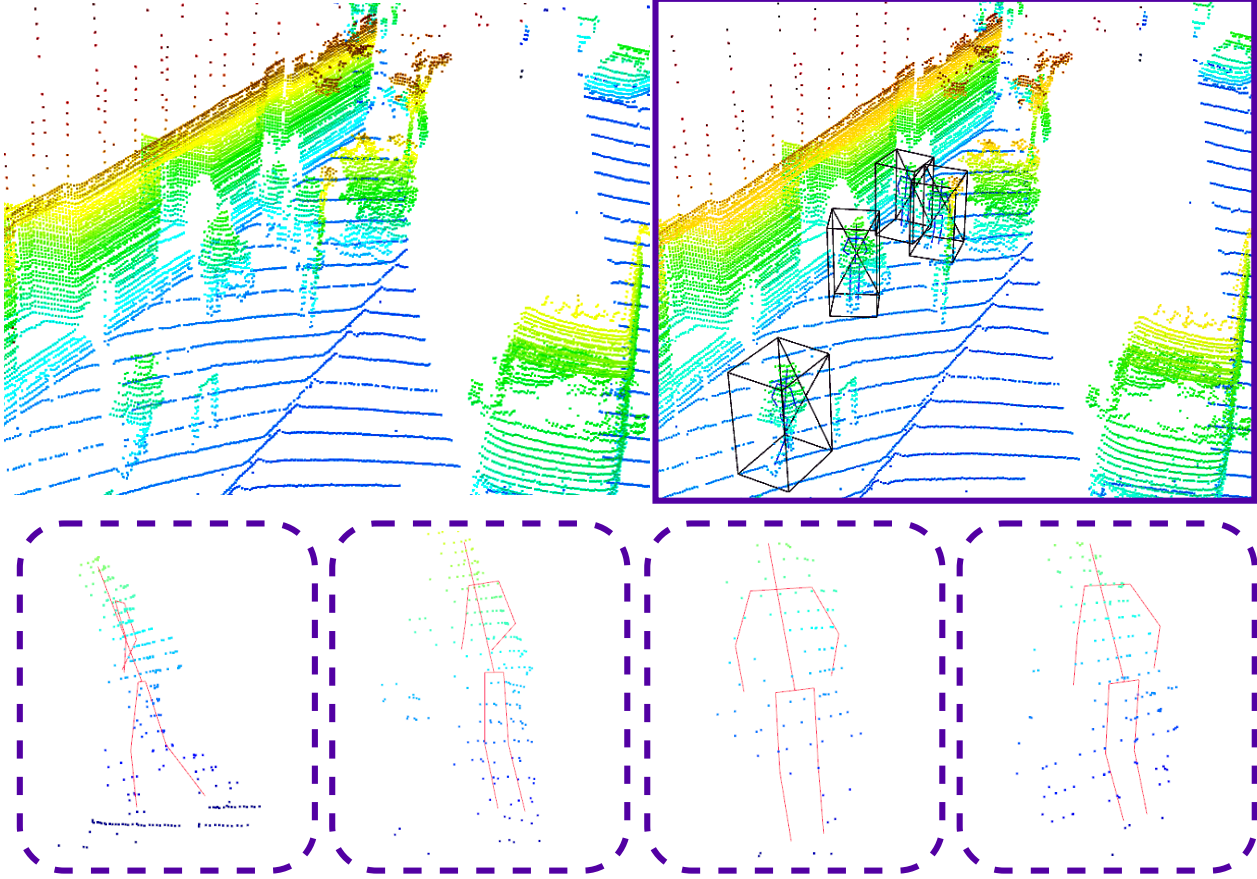
2

Figure 9. A visual demonstration of the baseline model (top left) and the proposed *VoxelKP* (top right). Our method detects the human objects that the baseline method fails to. The bottom line shows object-level keypoint localization performance.

in an instance.

$$\text{OKS} = \frac{\sum_j e^{-\frac{d_j^2}{2s^2 k_j^2}} v_j}{\sum_i v_j} \tag{9}$$

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 [15]. 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. Visual Results

We present additional visual results in this section. Majorly, we show our method locates the keypoints with better precision in Fig. 8, and can detect the human objects better than the baseline in Fig. 9.