8. Supplementary Material

In this supplementary material, we first explain the details about our modification on KPConv [33], then we analyze the contribution of rotation augmentation by an ablation study on 3DRotatedMatch dataset. We further conduct the ablative experiments on the detector loss as well as incorporating our detector with FCGF. Finally, we provide additional details about the experiments on 3DMatch, KITTI and ETH datasets and some further visualization results.

8.1. Implementation Details

Normalization term As explained in the main paper, the original formulation of KPConv is not invariant to point density. Thus we add a density normalization term, which sums up the number of the supporting points in the neighborhood, to ensure that the convolution is sparsity invariant. To demonstrate the effectiveness of the normalization term, we train networks with and without the normalization term in the same settings with what is described in the main paper, and report the registration results on 3DMatch dataset. During testing, instead of using voxel downsample, we use uniform downsample i.e. `uniform_down_sample` in Open3D [39] implementation, by which the density variations of input point clouds is enlarged. We evaluate the model on sample rate $= 15$, which leads to a similar average number of points for the point clouds in the test set. The result is shown in Table 6.

	Voxel	**Uniform**				
	5000	5000	2500	1000	500	250
w/o norm.	95.6	77.3	76.5	73.9	71.4	66.1
w norm.	95.8	91.9	91.8	91.2	90.2	89.4

Table 6: Feature matching recall at $\tau_1 = 10cm, \tau_2 = 5\%$. **Voxel** indicates voxel downsample with voxel size = $0.03m$, while **Uniform** indicates uniform downsample with sample rate = 15.

As shown in Table 6, the normalization term is effective to handle neighborhoods with different sparsity levels. When using a uniform downsample strategy, D3Feat with the normalization term is able to maintain a high matching quality, and outperform the model without the normalization term by a large margin.

Network architecture We adopt the architecture for segmentation tasks proposed in KPConv [33]. The number of channels in the encoder part are $(64, 128, 256, 512, 1024)$. Skip connections are used between the corresponding layers of the encoder part and the decoder part. The output features are processed by a 1×1 convolution to get the final 32 dimensional features. Other settings are the same with original paper [33].

8.2. Ablation on Rotation Invariance

In experiments, we find that a fully convolutional network is able to empirically achieve strong rotation invariance through low cost data augmentation. To demonstrate the effectiveness of simple data augmentation on the robustness to rotation, we further train the model without rotation augmentation and evaluate it on 3DRotatedMatch dataset. The result is shown in Table 7.

	5000	**2500**	**1000**	**500**	**250**
w/o aug.	5.5	5.4	4.9	5.4	5.5
w aug.	95.5	95.0	94.8	92.1	88.9

Table 7: Feature match recall on 3DRotatedMatch with and without rotation augmentation.

Without rotation augmentation, D3Feat fails on 3DRotatedMatch because the network cannot learn the rotation invariance from the data.

8.3. Ablation on Detector Loss

To better analyze the contribution of the proposed detector loss, we re-create a table below, which derives original results from Table 2, and the results from our model without the detector loss and performing on the predicted keypoints(w/o detector (pred)). The effect of detector loss can be seen from the comparison between w/o detector and D3Feat on both random (*rand*) or predicted (*pred)* points. It is shown that the detector loss not only strengthens the descriptor itself (given random keypoints), but also boosts the performance of the detector (given predicted keypoints). It is also noteworthy that only D3Feat(pred) improves the *Inlier Ratio* when reducing the number of points, which clearly indicates that the detector loss helps to better rank the keypoints regarding distinctiveness.

# Keypoints	**5000**	**2500**	**1000**	**500**	**250**
Feature Matching Recall (%)					
w/o detector(rand)	95.0	94.3	94.2	92.5	90.7
w/o detector(pred)	95.2	95.0	94.5	92.4	90.4
D3Feat(rand)	95.3	95.1	94.2	93.6	90.8
D3Feat(pred)	**95.8**	**95.6**	**94.6**	**94.3**	**93.3**
Inlier Ratio (%)					
w/o detector(rand)	40.7	39.9	35.2	31.0	25.1
w/o detector(pred)	**41.7**	40.3	38.7	36.6	33.2
D3Feat(rand)	40.6	38.3	33.3	28.6	23.5
D3Feat(pred)	40.7	**40.6**	**42.7**	**44.1**	**45.0**

Table 8: Ablation study on the proposed detector loss.

8.4. Ablation on Detector with FCGF

Since the detection scores are computed on top of the extracted dense descriptors, it is easy to incorporate our de-

tector with other dense feature description models, such as FCGF [2]. To demonstrate the usability of our method, we train the FCGF with the proposed joint learning method and evaluate it on the 3DMatch dataset, as shown in Table 9. The model FCGF + detector is trained using the proposed detector loss under the same setting with [2] for 100 epochs.

# Keypoints	5000	2500	1000	500	250
Registration Recall (%)					
FCGF[2]	**87.3**	85.8	85.8	81.0	73.0
FCGF + detector	86.7	**87.8**	**88.3**	**85.4**	**81.5**
Inlier Ratio (%)					
FCGF[2]	**56.9**	**54.5**	49.1	43.3	34.7
FCGF + detector	53.5	53.2	**53.6**	**53.2**	**51.0**

Table 9: Evaluation results of FCGF trained with the proposed detector loss.

The result shows that FCGF can indeed benefit from the proposed joint learning and maintain a high performance given a smaller number of points.

8.5. Runtime

To demonstrate the efficiency of our method, we compare the runtime of D3Feat with FCGF [2] on 3DMatch in Table 10. For a fair comparison, we use the same voxel size (2.5cm, roughly 20k points) with FCGF to measure the runtime of D3Feat including both detection and description. The performance difference mainly lies in the sparse convolution used by FCGF, which is time-consuming in hashing.

	CPU	GPU	Time(s)
FCGF[2]	Intel 10-core 3.0GHz(i7-6950)	Titan-X	0.36
D3Feat	Intel 4-core 4.0GHz(i7-4790K)	GTX1080	**0.13**

Table 10: Average runtime per fragment on 3DMatch test set.

8.6. Evaluation Metric for 3DMatch

Feature matching recall Feature matching recall is first proposed in [5], which measures the quality of features without using a RANSAC pipeline. Given two partially overlapped point cloud P and Q, and the descriptor network denoted as a non-linear function f mapping from input points to feature descriptors, the correspondence set for the fragments pairs is obtained by mutually nearest neighbor search in feature space,

$$\begin{aligned}\Omega = \{p_i \in P, q_j \in Q | f(p_i) = nn(f(q_j), f(P)),\\ f(q_j) = nn(f(p_i), f(Q))\},\end{aligned} \tag{16}$$

where $nn()$ denotes the nearest neighbor search based on the Euclidean distance. Finally the feature matching recall is defined as,

$$R = \frac{1}{|M|}\sum_{m=1}^{|M|} \mathbb{1}\Big(\Big[\frac{1}{|\Omega|}\sum_{(i,j)\in\Omega} \mathbb{1}(||p_i - T_m q_j|| < \tau_1)\Big] > \tau_2\Big) \tag{17}$$

where M is the set of point cloud fragment pairs which have more than 30% overlap, and T_m is the ground truth transformation between the fragment pair $m \in M$. τ_1 is the inlier distance threshold between a correspondence pair, and τ_2 is the inlier ratio threshold of the fragment pair. Following the setting of [36], the correspondence which have less than $\tau_1 = 10cm$ euclidean distance between their descriptors are seen as inliers, and the fragment pairs which have more than $\tau_2 = 5\%$ inlier correspondences will be counted as one match. The evaluation metric is based on the theoretical analysis that RANSAC need $k = 55258$ iterations to achieve 99.9% confidence of finding at least 3 correspondence with inlier ratio 5%.

Registration recall Registration recall [36] measures the quality of features within a reconstruction system, which firstly uses a robust local registration algorithm like RANSAC to estimate the rigid transformation between two point clouds, then calculate the RMSE of the ground truth correspondence under the estimated transformation. The ground truth correspondence set for fragments pair P and Q is given,

$$\Omega^* = \{p^* \in P, q^* \in Q\} \tag{18}$$

then the registration recall is defined as,

$$R = \frac{1}{|M|}\sum_{m=1}^{|M|} \mathbb{1}\Big(\sqrt{\frac{1}{|\Omega^*|}\sum_{(p^*,q^*)\in\Omega^*} ||p^* - \hat{T}_m q^*||^2} < 0.2\Big), \tag{19}$$

where $\hat{T}$ is the transformation matrix estimated by RANSAC. In our experiment, we run a maximum of 50,000 iterations on the initial correspondence set to estimate the transformation between fragments following [36].

8.7. Dataset Preprocessing

This section provides the steps to process the datasets including 3DMatch, 3DRotatedMatch, KITTI and ETH.

3DMatch For training set, we follow the steps in [36] to get fused point cloud fragments and corresponding poses. We find all the fragments pairs that have more than 30% overlap to build the training set. During training, we alternate between selecting the nearby fragment as the corresponding pair, or randomly selecting from all the overlapped fragments for fast convergence. For test set, we directly use the point cloud fragments and ground truth poses provided by the authors without performing any

preprocess to extract the dense feature and score map.

3DRotatedMatch Our model is inherently translation invariant because we are using the relative coordinates. So in order to test the robustness of our model to rotation, we create the 3DRotatedMatch test set following [4]. We rotate all the fragments in 3DMatch test set along all three axes with random sampled angle from a uniform distribution over $[0, 2\pi)$.

KITTI The training set of KITTI odometry dataset contains 11 sequences, we use sequence 0 to 5 for training, sequence 6 to 7 for validation and the last three for testing. Since GPS ground truth is noisy, we first use ICP to refine the alignment and then verify by whether enough correspondence pairs can be found. We select Lidar scan pairs with at least 10m intervals to obtain 1358 pairs for training, 180 pairs for validation and 555 for testing.

ETH For a fair comparison, we directly use the raw point clouds, the ground-truth transformations along with the overlap ratio provided by the authors of [11] to extract the features and evaluate the registration results.

8.8. Qualitative Visualization

We show some challenging registration results in Figure 6 and more visualizations of detected keypoints on 3DMatch, ETH, KITTI in Figure 7, 8, 9, respectively.

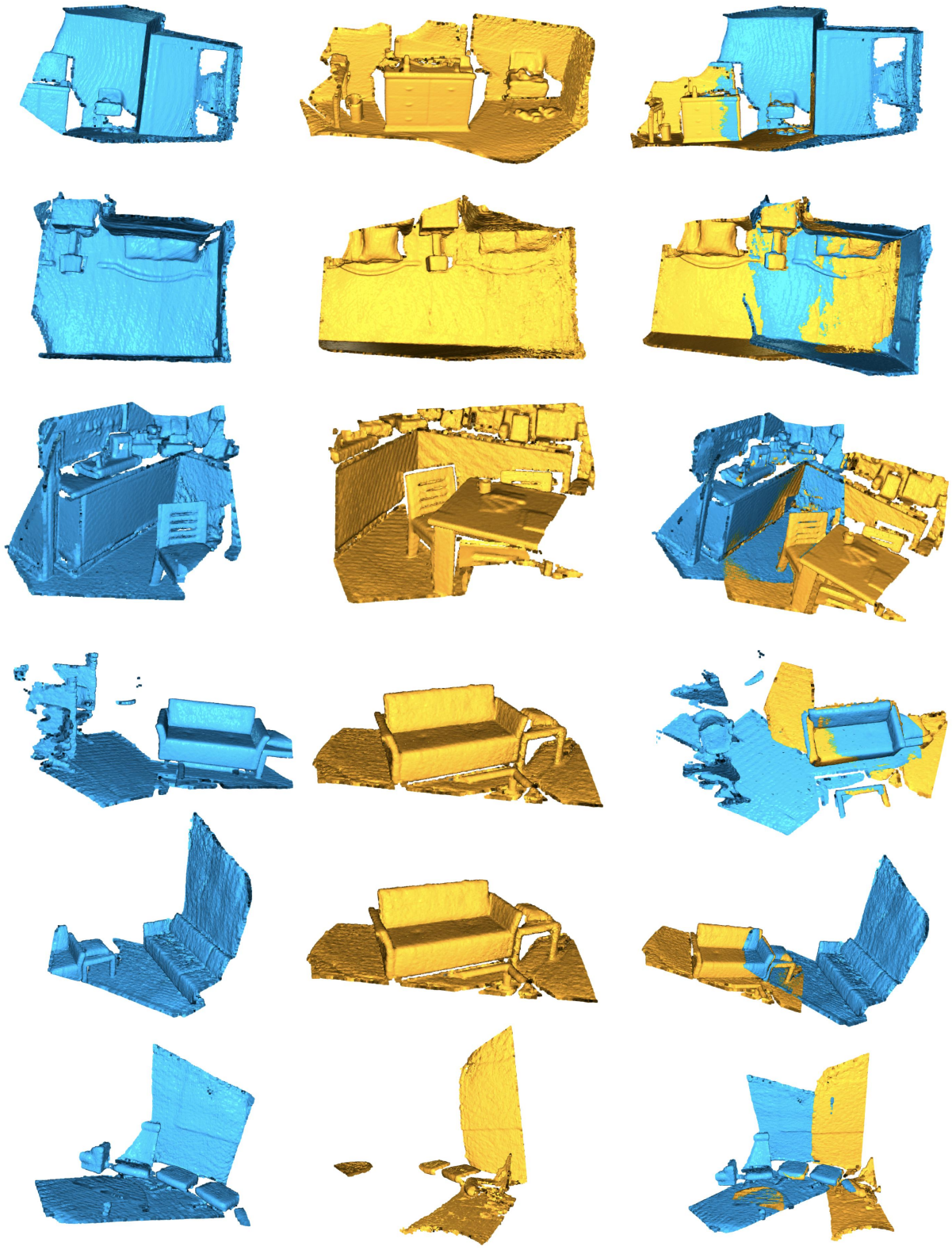

Figure 6: Qualitative results on the 3DMatch dataset. The first two columns are input point cloud fragments, and the third column presents the registration results. Best view with color and zoom-in.

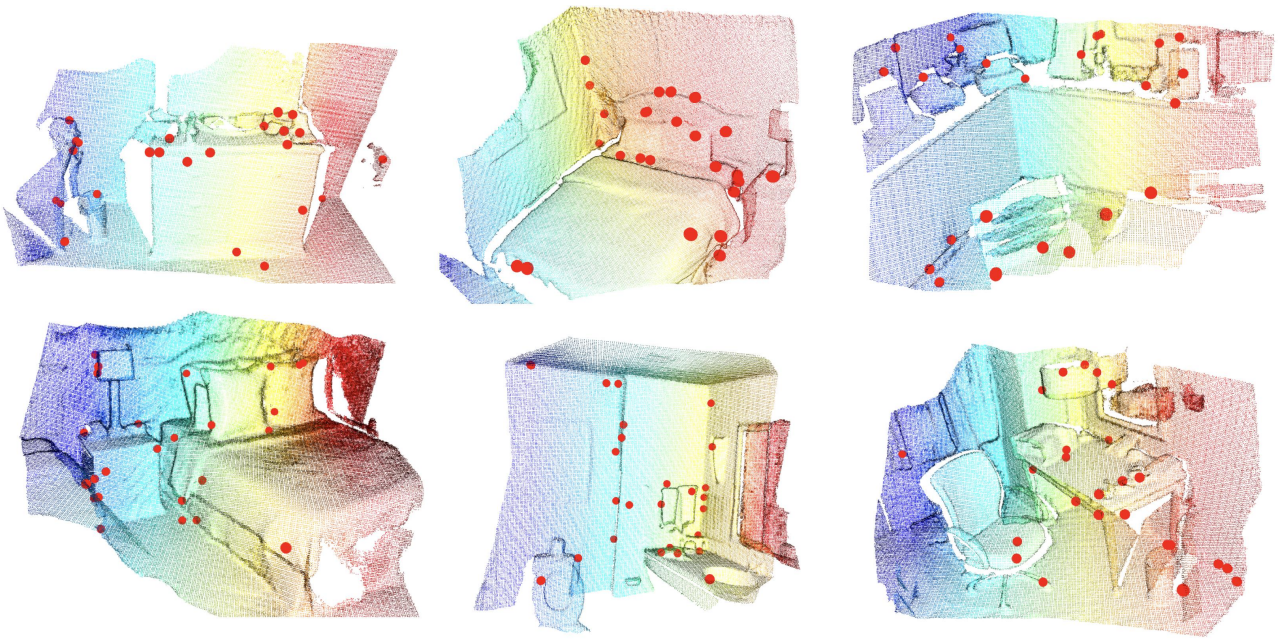

Figure 7: Visualization of keypoints on the 3DMatch dataset. Best view with color and zoom-in.

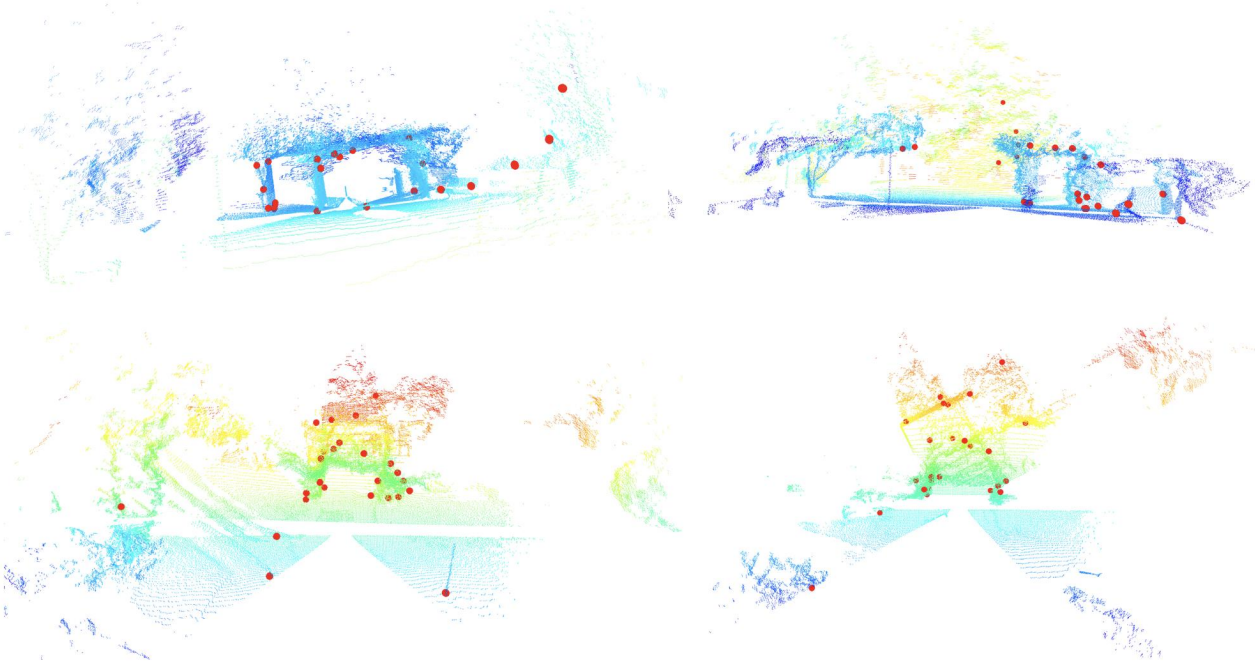

Figure 8: Visualization of keypoints on ETH dataset. Best view with color and zoom-in.

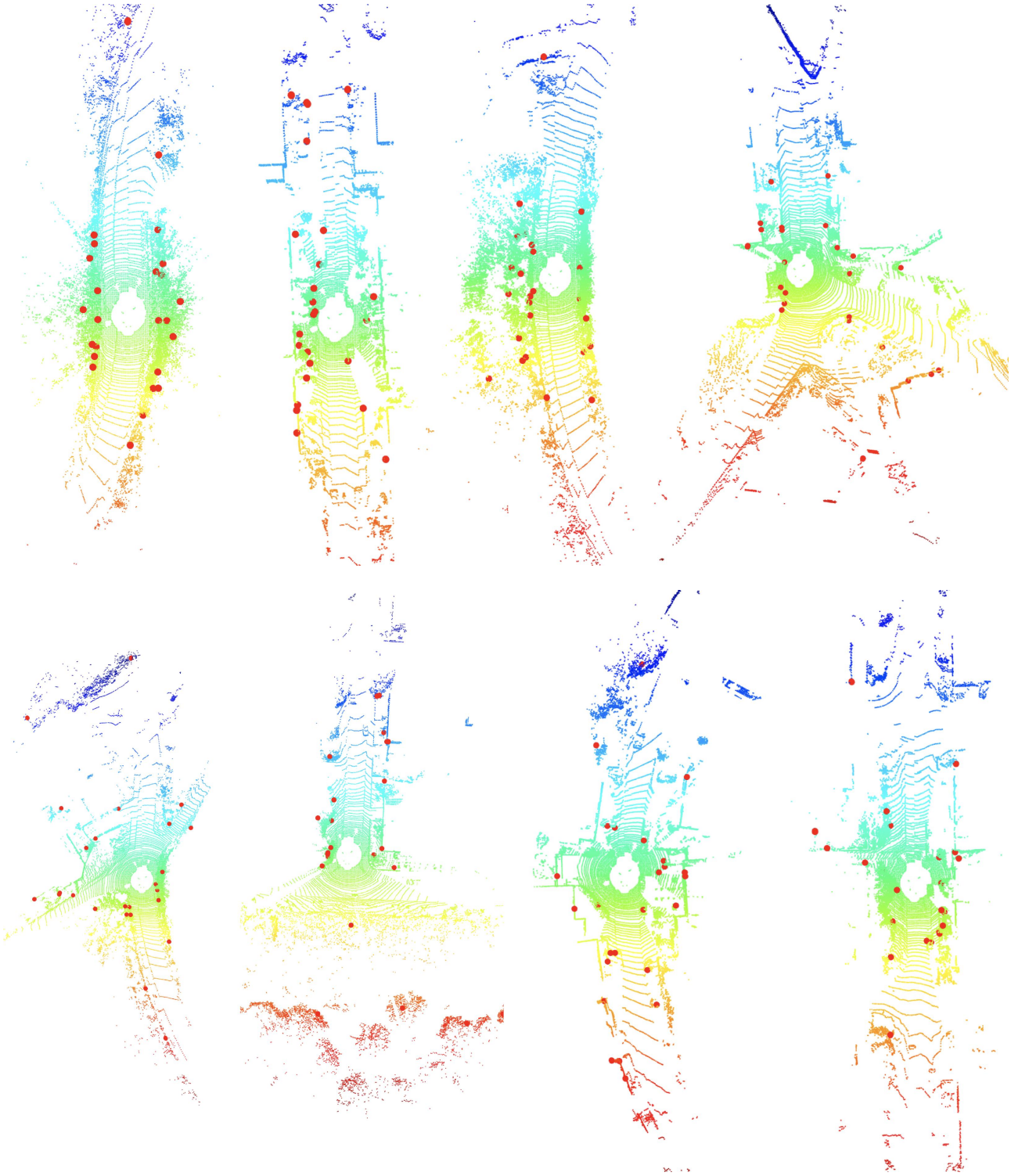

Figure 9: Visualization of keypoints on the KITTI dataset. Best view with color and zoom-in.