EP2P-Loc: End-to-End 3D Point to 2D Pixel Localization for Large-Scale Visual Localization (Supplementary Materials)

Minjung Kim Junseo Koo Gunhee Kim
Seoul National University
{minjung.kim, junseo.koo}@vision.snu.ac.kr, gunhee@snu.ac.kr
https://github.com/minnjung/EP2P-Loc

1. Details of EP2P-Loc

1.1. Feature aggregation for global matching

We extract 2D patch descriptors and 3D point descriptors using our image transformer and point transformer, respectively, described in Section 3.1 in the main paper. We then use NetVLAD [1] to aggregate these local descriptors into a global descriptor. The NetVLAD learns $\boldsymbol{C}$ cluster centers, denoted as $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_C \mid \boldsymbol{v}_c \in \mathbb{R}^{F_{dim}}\}$, and generates a $(F_{dim} \times C)$-dimensional VLAD descriptor $\boldsymbol{F}_{VLAD} = \{\boldsymbol{F}^1_{\text{vlad}}, \cdots, \boldsymbol{F}^C_{\text{vlad}}\}$, where F_{dim} is the dimension of the 2D patch descriptor and 3D point descriptor. We compress this $(F_{dim} \times C)$-dimensional VLAD descriptor into a 256-dim vector using a fully connected layer with an L2 normalization to produce the final global descriptor vector. Global descriptors are used to reduce search space from the 3D global reference map by retrieving relevant submaps from the database, which is called *global matching*. We set $F_{dim} = 128$ and $\boldsymbol{C} = 64$, and finally obtain a 256-dim global descriptor from 128-dim local descriptors.

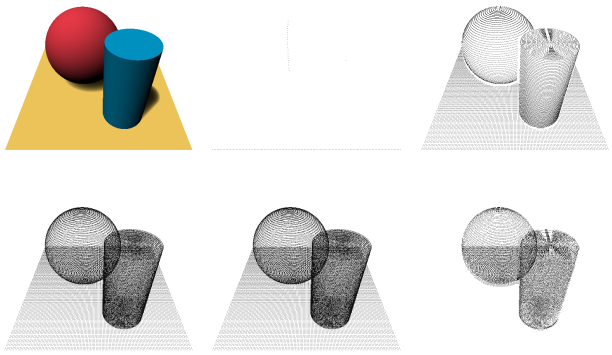
1.2. Invisible points removal algorithm

Our *Invisible 3D Point Removal (IPR)* algorithm aims to mitigate appearance discrepancy between the 2D image $I \in \mathbb{R}^{H \times W \times 3}$ and the corresponding positive 3D point cloud submap $P \in \mathbb{R}^{M \times 3}$ by removing invisible 3D points from the image before extracting point cloud features. The depth map $D \in \mathbb{R}^{H \times W \times 1}$ is generated by projecting the 3D points in P on the image I. We transform each 3D point (x_i, y_i, z_i) in P into the camera coordinate system, and use z-coordinate as a depth value as follow:

$$[x_i^c, y_i^c, z_i^c]^\top = R[x_i, y_i, z_i]^\top + t, \tag{1}$$

$$D_{\pi(x_i^c, y_i^c, z_i^c)} = z_i^c, \tag{2}$$

where (x_i^c, y_i^c, z_i^c) indicates the coordinates in the camera coordinate system. As in the main paper, rotation matrix R and translation vector t represent the camera pose $(R, t) \in SE(3)$, and π is a function that projects 3D points of the camera coordinate system to 2D points of the image coordinate system based on the camera intrinsics $\mathbf{K}$. We then apply min-pooling and max-pooling sequentially, both with the same kernel size s $(= 9)$, so that points for which the depth decreases after the two pooling layers are considered as occluded points. For every 3D point within the point cloud submap P, called a query point, our IPR algorithm employs min-pooling to locate nearby points, and then utilizes max-pooling to determine if these points encompass the queried point. This prevents 3D points from invisible objects from being projected onto other pixels and getting mixed together, which could happen due to the substantial amount of empty space within the 3D point cloud. Invisible and visible points are not guaranteed to have respectively minimum and maximum depth around them, so



(a) Originals (b) w/ one pooling (c) Results

Figure 1: Results of our IPR algorithm in a toy example. (a) The toy example image (top) and 3D point cloud (bottom). (b) Removal results using single min-pooling (top) and max-pooling (bottom). (c) Result of our IPR algorithm (top) and the set of removed 3D points (bottom).

(a) Point cloud (b) s=3 (c) s=5 (d) s=7 (e) s=9 (f) s=11

Figure 2: Results of our IPR algorithm using different kernel size s from 3 to 11 on the 2D-3D-S dataset [2]. The kernel size for both min-pooling and max-pooling is set to s.

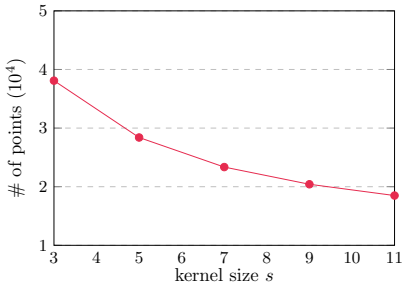


Figure 3: The average number of 3D points after applying our IPR algorithm with different kernel size s on the 2D-3D-S dataset [2]. The number of 3D points is obtained from the point cloud submaps in the database.

single min-pooling or max-pooling does not work well, as shown in Figure 1. The removal method using a single pooling layer does not properly remove invisible points. The min-pooling leaves only the locally closest points, and the max-pooling removes only the locally farthest points. Our IPR algorithm effectively detects the 3D points behind the surface by checking whether the closer points surround the point. As a result, our IPR algorithm can highly reduce the ratio of invisible points and helps to train efficiently in later steps. Note that the time complexity of the IPR algorithm is $O(M + HWs^2)$, but can be implemented as $O(M + HW)$ by using a sliding window, or $O(Ms^2)$ with a sparse pooling [8]. Invisible 3D points have no information that can be matched to 2D pixels in the image, which can result in incorrect 2D-3D correspondences. Therefore, our IPR algorithm can be used to learn which 3D points can be matched to 2D pixels.

Our IPR algorithm has two limitations. Firstly, our criteria for identifying invisible points to be surrounded by neighboring closer points may lead to the exclusion of local depth minima on concave surfaces or corners. In the toy example in Figure 1c, points positioned between the sphere and the plane also vanish. However, this primarily occurs with points situated backward rather than those belonging to foreground objects, and there isn't significant removal of edges within the 3D space. Secondly, our algorithm struggles to effectively eliminate occluded points when there is a substantial disparity in depth. Owing to perspective effects, the distance between 3D points on nearby surfaces appears greater than that on distant surfaces. If surfaces are closely situated, it might not be feasible to enclose distant points with those on the surface, and the distant points located behind the surface may not be removed well.

These two limitations of our IPR algorithm involve a trade-off depending on the kernel size s, as illustrated in Figure 2. As s decreases, fewer local minima are eliminated from the depth map, but the removal performance decreases even with a slightly closer surface; conversely, as s increases, more local minima are removed, but the algorithm is robust to closer surfaces. The scope of erroneously eliminated local minima and the upper limit on the distance between 3D points within a surface are proportional to s. As s increases, more 3D points are removed, as shown in Figure 3, there are $65,536$ points in each point cloud before removal, but as s changes between 3 and 11, the average number of points after removal changes from 3.81×10^4 to 1.85×10^4. Considering this trade-off and the number of points remaining after removal, we choose $s = 9$. Note that point density and image size can also affect the appropriate s, for example, sparser point clouds and larger images might necessitate a larger s. For our IPR algorithm, we work with point clouds containing $65,536$ points and generate depth maps with dimensions of 512×512 pixels.

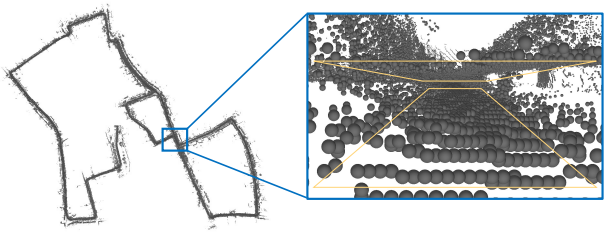


Figure 4: Accumulated point cloud map of the Oxford RobotCar dataset [16]. We utilize a sequence collected at 2014-06-26 09:31:18 and calculate poses using the INS pose sensor. Due to imprecise poses, the ground plane is represented in a different location upon revisitation.

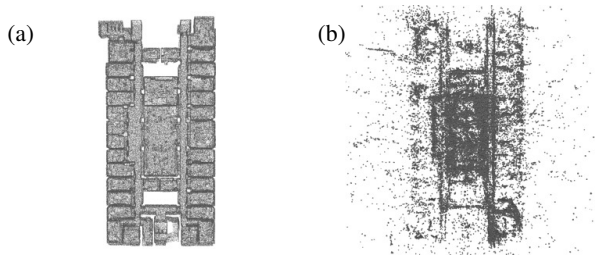


Figure 5: Global point cloud map of the 2D-3D-S dataset (Area 1) [2]. (a) Original global point cloud map. (b) Point cloud map built from SfM using 2D images in Area 1.

2. Benchmark datasets

2.1. Dataset selection

Poses for the large-scale outdoor datasets [5, 10, 16] are acquired through Inertial Navigation System (INS) or Global Positioning System (GPS), whereas the Oxford RobotCar dataset [16] employs visual odometry. The global pose can be calculated using the output from INS or GPS, but it is not locally consistent due to GPS errors and accumulation errors. To construct a large 3D point cloud map, we need to aggregate each LiDAR point cloud with its corresponding pose. Without using accurate LiDAR poses, the surface is duplicated due to inconsistent poses, as seen in Figure 4, which removes fine details from the point cloud. Consequently, the utilization of INS and GPS is unsuitable for tasks involving localization and registration using point clouds. Visual odometry, used in the image-to-point cloud registration task [15], provides a locally consistent relative pose between the point cloud and the corresponding image. However, due to accumulation errors inherent in visual odometry, the pose lacks temporal consistency, and as a result, revisiting the same place in different timestamps does not guarantee a similar pose. As a result, the visual odometry is also inadequate for the image-to-global point cloud localization.

2.2. Dataset preprocessing

We select Stanford 2D-3D-Semantic (2D-3D-S) [2] and KITTI [11] datasets to establish benchmarks. These datasets include database point clouds, query images, global poses of point clouds and images, and camera intrinsics. The 2D-3D-S dataset contains 6 areas scanned by the MatterPort camera, which gives RGB-D data. Each area contains RGB images, depth, surface normals, global XYZ images, and a global point cloud, where the global point cloud is identical to the global point cloud of S3DIS [3]. There are about 50 images taken at each camera location but at different angles, and a maximum of 20 images are selected to constrain the number of test images. The InLoc dataset [18], widely used in indoor visual localization task, collects training images exactly at 30° intervals. We sample RGB images for training and test so that the angles between the images taken at the same location do not differ by less than 30° and 15°, respectively, following the InLoc's setting. For the coarse-to-fine matching approach, we provide the global point cloud maps along with small submaps. We divide the global point cloud map of S3DIS into fixed-size submaps visible in the training image frames within 10m and downsample to have $65,536$ points. We follow the 3-fold cross-validation scheme of 2D-3D-S as shown in Table 1 in the main paper, which evaluates the performance of pose estimation for unseen places during training. In each Fold, the database for the test areas is constructed by collecting the training data for those areas that are not used during training. For example, the database for the Fold 3 consists of the training data from Areas 2, 4, and 5 combined.

The KITTI dataset is built with Velodyne HDL-64E, a 3D LiDAR, and we accumulate all 3D point clouds to obtain a global point cloud map. The query images are from Cam2 among 4 Point Gray Flea 2 cameras, the RGB camera installed at the center of the vehicle. There are 11 sequences numbered 00 to 10 that have ground-truth poses, and we split the sequences 00 to 08 for training and the sequences 09 to 10 for testing. Since the ground-truth poses from the KITTI dataset are relative poses to the camera pose from the first frame, we need to define a reference pose to use as the world coordinate system. We utilize the LiDAR pose from the first frame as the world coordinate system so that the z-axis is perpendicular to the ground as in the 2D-3D-S dataset. Images are collected every $2m$ from the trajectory without any special preprocessing. We construct a single 3D global point cloud map from each sequence in the dataset by accumulating 3D LiDAR data. Similar to the 2D-3D-S, we divide the global point cloud map into small submaps containing the 3D points in the corresponding image frame within $30m$, and downsample the collected point clouds to have $65,536$ points. There are samples of 2D-3D-S and KITTI in Figure 8 and Figure 9, respectively.

Model	Fold 1 (0.1, 1.0)	Fold 1 (0.25, 2.0)	Fold 1 (1.0, 5.0)	Fold 2 (0.1, 1.0)	Fold 2 (0.25, 2.0)	Fold 2 (1.0, 5.0)	Fold 3 (0.1, 1.0)	Fold 3 (0.25, 2.0)	Fold 3 (1.0, 5.0)	# of points	# of inliers	Runtime
EP2P-Loc w/o IPR	**88.12**	90.33	92.75	91.57	**94.12**	**94.89**	83.23	86.36	88.29	65.5K	3.12K	6.981s
EP2P-Loc w/ EPnP [13]	85.34	87.51	90.23	90.23	92.34	93.78	83.56	84.96	87.13	20.4K	0.32K	12.53s
EP2P-Loc w/ SfM [17]	82.18	84.88	89.23	87.43	90.11	92.31	81.97	84.23	86.09	4.10K	0.69K	9.53s
EP2P-Loc (Ours)	87.59	**91.65**	**92.89**	**92.47**	93.88	94.38	**85.24**	**87.38**	**89.43**	20.4K	2.73K	3.482s

Table 1: Ablation study on the 2D-3D-S dataset [2]. The threshold units are (m, °). The number of points is reported with the average value per point cloud submap in the training set. The number of inliers and runtime are reported with the average value per image in the test set, and runtime is measured with the overall pipeline including data loading to pose estimation.

Area	# of points: Original point cloud map	# of points: SfM [17]
1	44.0M	122K
2	47.3M	381K
3	18.7M	49K
4	43.5M	322K
5	78.6M	276K
6	41.4M	118K

Table 2: The number of points in the point clouds of the 2D-3D-S dataset [2], and the point clouds built from SfM [17] using the 2D images in the 2D-3D-S dataset.

2.3. Oxford RobotCar dataset

We also experiment with the Oxford RobotCar dataset [16] for further comparison with image-to-point cloud registration methods such as DeepI2P [15] and 2D3D-MatchNet [9]. Among the sequences in the Oxford RobotCar dataset, we use 40 sequences and split 35 for training and 5 for testing according to DeepI2P's setting. For each sequence, images and 3D point clouds are collected every $2m$. We use the image from the center camera of the Bumblebee XB3 stereo camera, and remove the lower 20% pixels to eliminate the vehicle in the image. We accumulate the 2D point clouds from the front LMS-151 for a $100m$ trajectory to create the 3D point cloud. The center of the point cloud is randomly selected at a location where the Chebyshev distance (*i.e.* L_∞) is within $10m$ from the camera location. We use padding and downsampling to ensure that each point cloud has $262,144$ points. For a fair comparison with image-to-point cloud registration methods, we take each image and point cloud pair as input, and evaluate the estimated relative pose between them. We show several samples in Figure 10.

3. Experiments

3.1. Results of image-to-point cloud registration

Similar to the KITTI dataset [11], we train on the Oxford RobotCar dataset [16] and compare the image-to-point cloud registration methods [9, 15] in Table 3. While we employ 40 sequences from the Oxford RobotCar dataset, all of them originate from the same geographical location with different illumination and weather conditions. This result shows that EP2P-Loc can localize robustly even when the timestamp changes significantly with showing illumination and weather changes.Furthermore, our approach surpasses the performance of previous methods.

3.2. Comparison with the 3D point cloud generated by Structure from Motion (SfM)

We evaluate EP2P-Loc with 3D point clouds created by the Structure from Motion (SfM), not LiDAR or RGB-D sensors. Using COLMAP [17], we generate a 3D reference map from the images in the 2D-3D-S dataset [2] of our benchmark datasets. We do not utilize any other information from the SfM, for example, 2D keypoint coordinates or 2D-3D correspondences. As shown in Figure 5, these point clouds are sparser and noisier than those generated by LiDAR or RGB-D sensors. The SfM calculates 3D points by triangulating 2D keypoints. However, when the keypoints are sparse, such as on a featureless wall, it may not be possible to triangulate the 3D points. Moreover, the presence of similar textures can lead to the triangulation of incorrect keypoint pairs that share resemblant features but are situated in different locations, resulting in noisy 3D points. As a result, the point cloud generated from the SfM only contains coarse geometry, and the walls are represented sparsely. We follow the same preprocessing as in the 2D-3D-S dataset, but downsample the point cloud submaps to have $4,096$ points due to the sparsity of the point clouds as shown in Table 2 and Figure 5.

Due to the sparsity of the point cloud constructed by the SfM, we retrieve the top-7 database point cloud submaps without using the IPR algorithm for evaluation. As demonstrated in Table 1, EP2P-Loc model trained with the SfM-derived point cloud map shows a performance discrepancy of less than 7% when compared to the model trained using the global point cloud map of the 2D-3D-S. This shows that our model can successfully localize by only utilizing 3D point information from the SfM model created from 2D images, which is also noisier and more than 200 times sparser than the original global point cloud. As a result, our model can localize regardless of whether the database consists of

Model	Oxford [16]		KITTI [11]	
	RTE (m)	RRE (°)	RTE (m)	RRE (°)
Direct Regression	5.02 ± 2.89	10.45 ± 16.03	4.94 ± 2.87	21.98 ± 31.97
Monodepth2 [12] + USIP [14]	33.2 ± 46.1	142.5 ± 139.5	30.4 ± 42.9	140.6 ± 157.8
Monodepth2 + GT-ICP [4, 7]	1.3 ± 1.5	6.4 ± 7.2	2.9 ± 2.5	12.4 ± 10.3
2D3D-MatchNet [9]	1.41	6.40	752.5 ± 6053.3	117.9 ± 52.1
Grid Cls. + PnP [13, 15]	1.91 ± 1.56	5.94 ± 10.72	3.22 ± 3.58	10.15 ± 13.74
DeepI2P [15]	1.65 ± 1.36	4.14 ± 4.90	3.28 ± 3.09	7.56 ± 7.63
Patch Cls. + PnP (Ours)	1.24 ± 1.17	4.23 ± 6.78	2.83 ± 2.89	6.32 ± 5.63
EP2P-Loc (Ours)	**0.87 ± 1.39**	**4.03 ± 5.32**	**1.32 ± 1.13**	**4.11 ± 5.46**

Table 3: Experimental results of image-to-point cloud registration on the Oxford [16] and the KITTI [11] dataset. We provide the 2D3D-MatchNet [9] results on the KITTI dataset of our implementation.

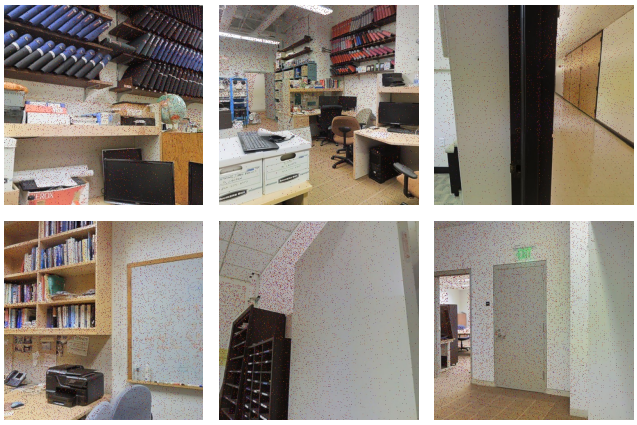


Figure 6: Visualization of the differentiable PnP solver [6] weights of each point in the point cloud. Red color has a higher value, and black color has a lower value.

RGB images, RGB-D images, or LiDAR sensor data.

3.3. Analysis of our 2D patch classification

In our EP2P-Loc, pose estimation is also possible by finding coarse correspondences using only 2D patch classification without 2D pixel coordinate calculation. Similar to DeepI2P, the correspondence is used as a 3D point with the center of the classified patch. As shown in Table 3, our Patch Cls. + PnP outperforms both DeepI2P [15] and Grid Cls. + PnP, and is not significantly different from EP2P-Loc. This shows that 2D pixel coordinate calculation only gives a finer correspondence, and our pipeline is better than grid classification from DeepI2P even without pixel coordinate calculation.

3.4. Visualization

We present qualitative results evaluated on our benchmark datasets. Figure 7 illustrates the results of some nearest neighbor searches using the global descriptor on the 2D-3D-S [2] evaluation set. The leftmost column represents the query image, and the next four results show the top 4 retrieved point cloud submaps of the database. The results in green indicate correct matches (true positive) and the ones in red indicate incorrect ones (false positive). As shown in the red-boxed examples in Figure 7, the failure cases are structurally similar point clouds in the evaluation set.

Figure 6 shows the visualizations of 2D-3D correspondences depicted by the weights used in the differentiable PnP solver. Each point is marked with a different color based on the weight value from our differentiable PnP solver module, with redder points having a higher weight. The 2D-3D correspondence is much less distributed over empty space in 3D space, and is concentrated in areas with clear geometry (3D features) or distinct texture (2D features), as shown in Figure 6. We put these weighted 2D-3D correspondences to the differentiable PnP solver for end-to-end training, which leads to the state-of-the-art performance of our method.

References

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5297–5307, 2016.

[2] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*, Feb. 2017.

[3] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1534–1543, 2016.

[4] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[5] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan,

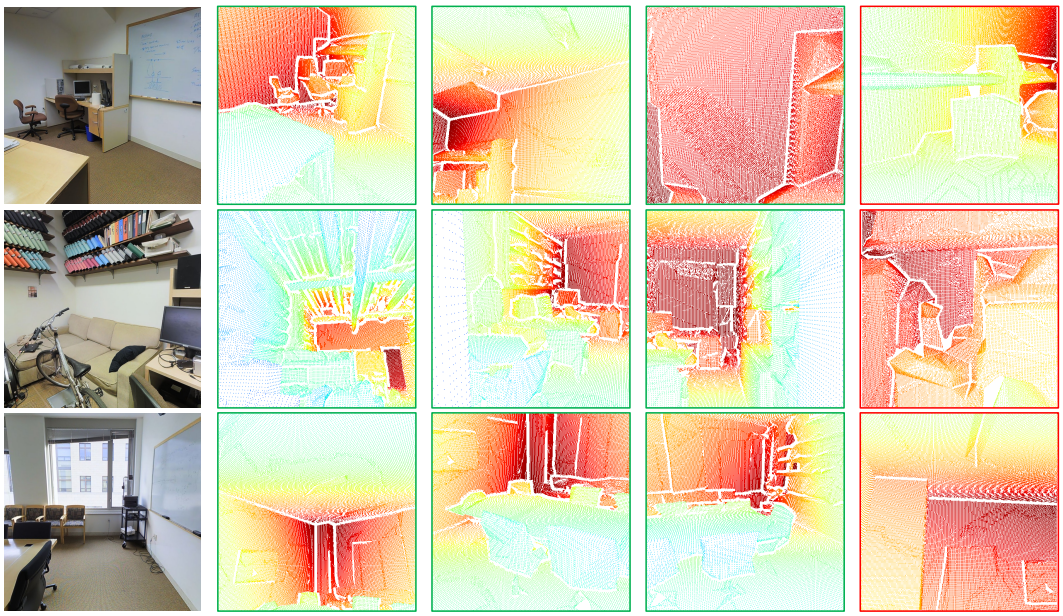


Figure 7: Results of our global matching. The first column represents the query image, and the following four columns represent the retrieved database point cloud. The green indicates the correctly retrieved point cloud, and the red indicates the incorrect one.

Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[6] Hansheng Chen, Pichao Wang, Fan Wang, Wei Tian, Lu Xiong, and Hao Li. Epro-pnp: Generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[7] Yang Chen and Gérard Medioni. Object modeling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992.

[8] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3075–3084, 2019.

[9] Mengdan Feng, Sixing Hu, Marcelo H Ang, and Gim Hee Lee. 2d3d-matchnet: Learning to match keypoints across 2d image and 3d point cloud. In *International Conference on Robotics and Automation (ICRA)*, pages 4790–4796. IEEE, 2019.

[10] Whye Kit Fong, Rohit Mohan, Juana Valeria Hurtado, Lubing Zhou, Holger Caesar, Oscar Beijbom, and Abhinav Valada. Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking. *arXiv preprint arXiv:2109.03805*, 2021.

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[12] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3828–3838, 2019.

[13] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision (IJCV)*, 81(2):155–166, 2009.

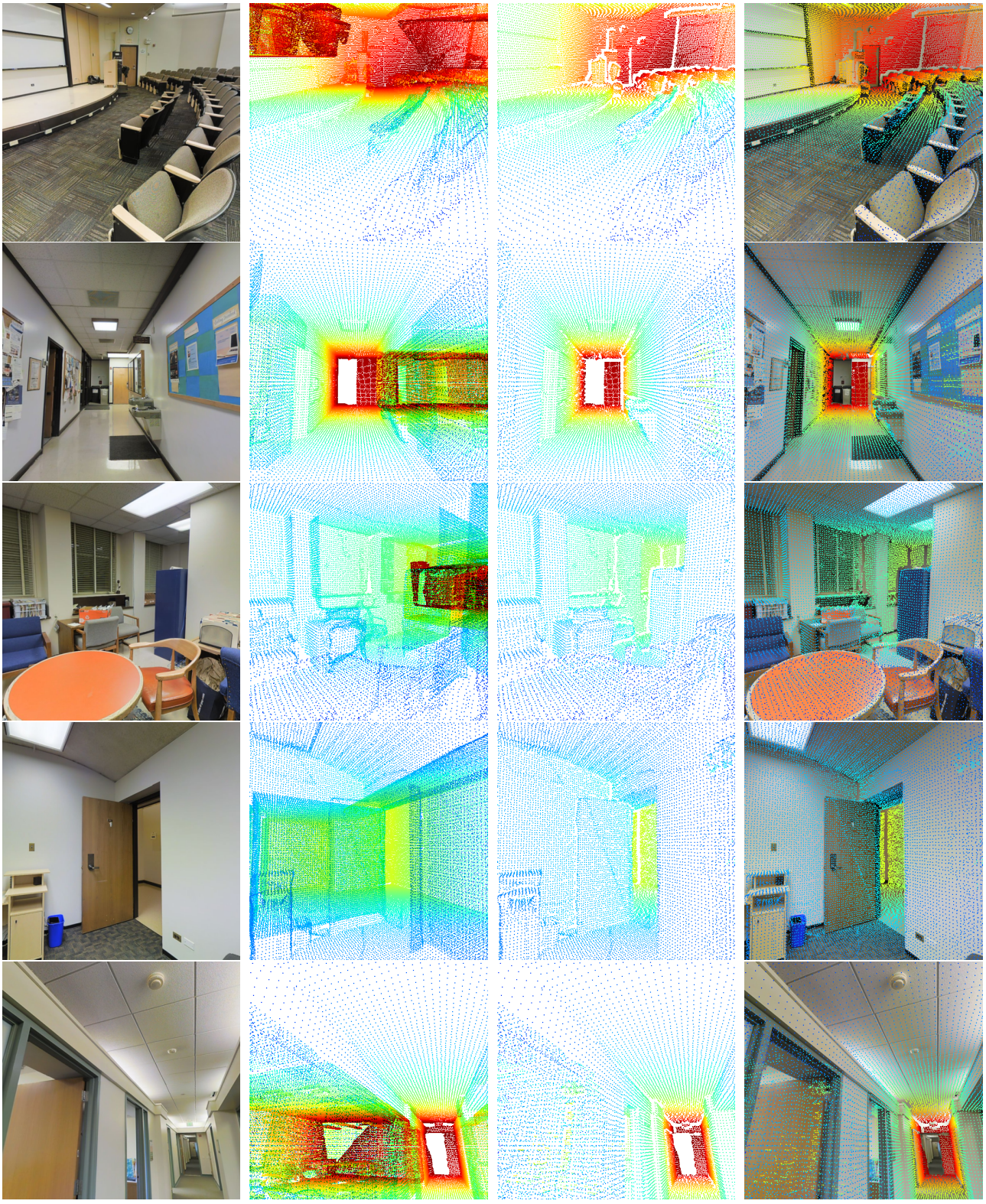
[14] Jiaxin Li and Gim Hee Lee. Usip: Unsupervised stable interest point detection from 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 361–370, 2019.

[15] Jiaxin Li and Gim Hee Lee. DeepI2P: Image-to-Point Cloud Registration via Deep Classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15960–15969, 2021.

[16] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017.

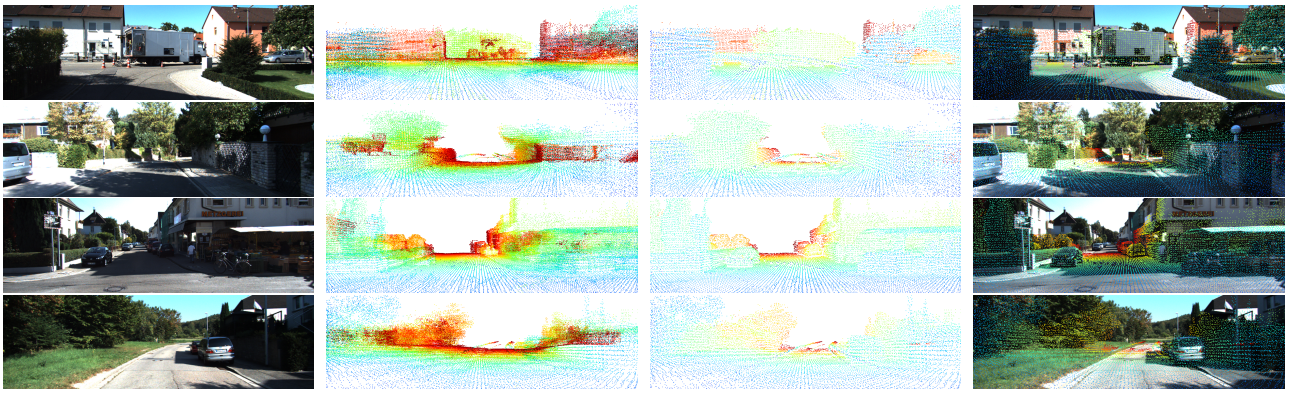
[17] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[18] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7199–7209, 2018.



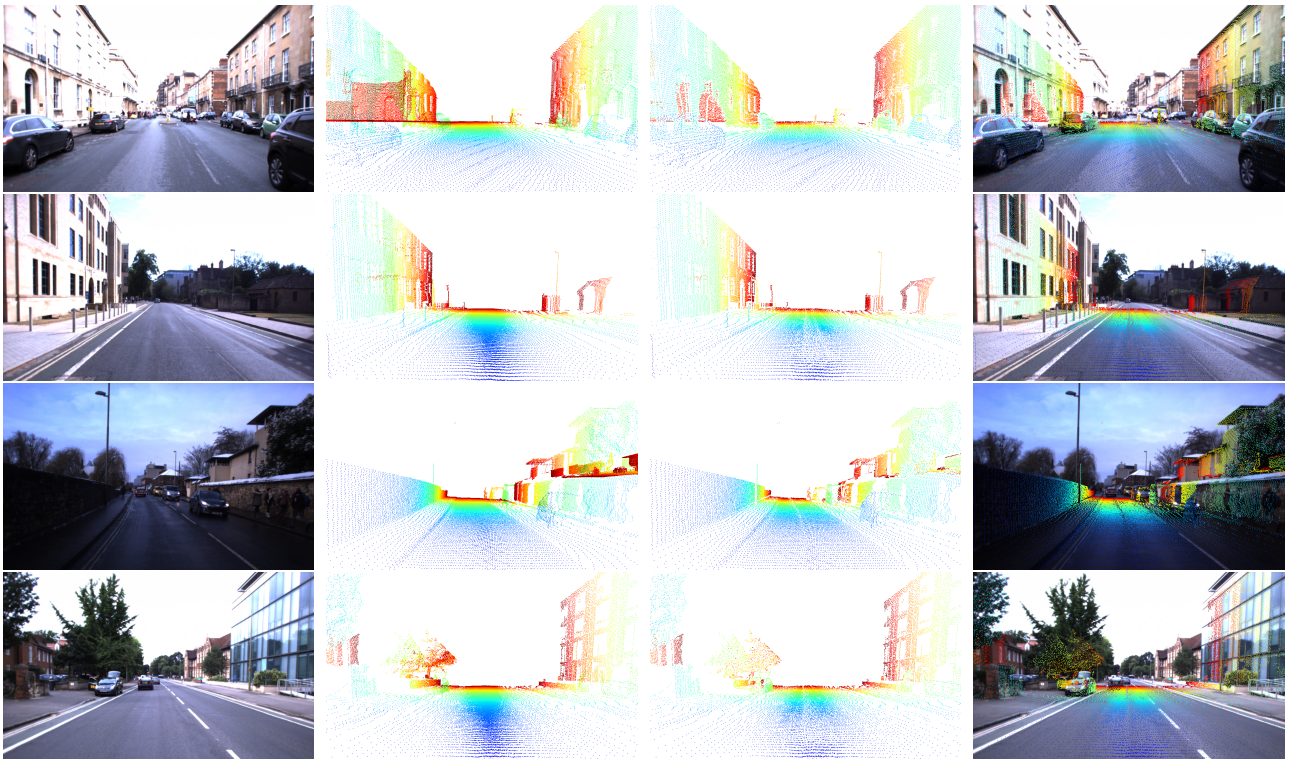
(a) (b) (c) (d)

Figure 8: Examples of 2D-3D-S dataset [2]. (a) RGB images. (b) Corresponding point cloud submaps. (c) Submaps after removing invisible points from (b). (d) Projected 3D points (c) into (a).



(a) (b) (c) (d)

Figure 9: Examples of KITTI dataset [11]. (a) RGB images. (b) Corresponding point clouds. (c) Point clouds after removing invisible points from (b). (d) Projected 3D points (c) into (a).



(a) (b) (c) (d)

Figure 10: Examples of Oxford RobotCar dataset [16]. (a) RGB images. (b) Corresponding point clouds. (c) Point clouds after removing invisible points from (b). (d) Projected 3D points (c) into (a).