Supplementary Document for Paired-Point Lifting for Enhanced Privacy-Preserving Visual Localization

Chunghwan Lee
Hanyang University
jhlee612@hanyang.ac.kr

Jaihoon Kim
Samsung Electronics, KAIST
jh27kim@{gmail.com, kaist.ac.kr}

Chanhyeok Yun
Hanyang University
cksgurdbs@hanyang.ac.kr

Je Hyeong Hong*
Hanyang University
jhh37@hanyang.ac.kr

Abstract

This supplementary document illustrates for algorithmic details and concrete discussion on limitation of our method, and further experimental details from the main paper.

In addition to this document, we have also included a supplementary video comparing image reconstructions from different 3D representations (point cloud, original line cloud (OLC), PPL-based line cloud (PPL) and PPL with coplanar line rejection (PPL+)). Code is available at https://github.com/Fusroda-h/ppl.

1. Algorithmic details for PPL and PPL+

We provide algorithms for PPL-based line cloud sampling and PPL+ (PPL with coplanar line rejection) approach in terms of pseudocode. To start with PPL, the implementation of PPL is very straightforward. Given 3D points, it first permutes 3D points and divides into two sets. It returns 3D line direction vectors by subtracting between two 3D points selected from each set with the same index.

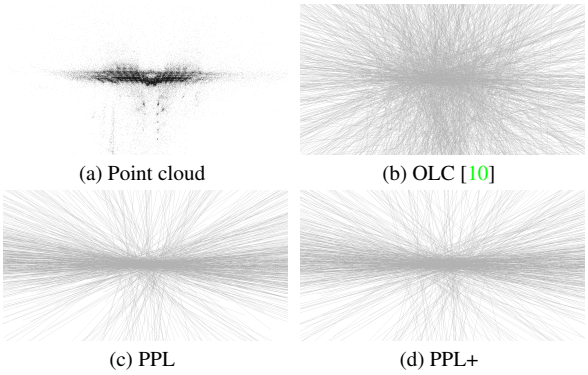
On the other hand, PPL+ starts by creating initial line directions with PPL method. Then, the algorithm finds the direction vectors from a 3D point to its nearest neighboring 3D points for every 3D points, which denoted as $\mathbf{nns}$. Note that $\mathbf{nns} \in \mathbb{R}^{N\times M\times 3}$ where N is the number of the 3D points and M is the number of the nearest-neighboring points which is set as the smaller one out of $N \times 0.01$ or 100 in our experiments. The algorithm then picks a random 3D points and its $\mathbf{nns}$ to detect the points which lie on the planar shape formed by PPL-based lines. It reserves the 3D points that do not align with the line cloud as the valid points, and repeats the process with the rest of the rejected points. The process is repeated until the number of the points that align with the line cloud is less than 10 or does not decrease for 50 iterations. We set the maximum number of the iteration as 1000 and the threshold angle between the line and the plane to be 20°. A detailed procedure is described in Algorithm 1.

2. Discussion on the limitation of PPL

In this section, we discuss the detailed explanation on limitation described in Sec. 3.4 from the main paper [5].

2.1. PPL-based lines may reveal scene geometry

As we shortly discussed in [5], Fig. 1 shows that PPL applied on planar scenes may hint existence of some structure. Some might guess the dense parallel lines might appear to be the wall. Nevertheless, the specifics of the scene and the whole scene that lines represent remain hidden, as the line cloud driven by PPL or PPL+ is still a line cloud. Therefore, it is still difficult to identify the nature of the geometry (e.g. a plane, corridor or an outer wall, etc.) just from line in complex line cloud maps.



(a) Point cloud (b) OLC [10] (c) PPL (d) PPL+

Figure 1. Comparison of line clouds in planar *Old Hospital*

2.2. Quasi-parallel lines worsen pose accuracy

We expand discussions on degenerate cases for PPL/PPL+ thanks to the reviewer's insight. For this purpose, we investigated the quasi-parallel nature of PPL by

*Corresponding author

comparing the mean angle between lines in a close-ended hallway of unit width and height with no ceiling/floor. We varied the hallway length and sampled 3D points with equal density and perturbation $N(\mathbf{0}, 0.1\mathrm{I}_3)$. Table 1 shows PPL yields more parallel lines as the scene becomes overly stretched. Also, it shows that the degeneracy of PPL+ (orthogonal-to-plane lines being parallel) can be avoided by choosing an appropriate (not strict) threshold.

Table 1. Mean angles between lines for different line clouds.

Hallway length (width=1)	1	10	100
OLC [10]	57.3°	57.3°	57.3°
PPL	56.7°	30.1°	6.4°
PPL+ (accept if $> 20°$ from plane)	55.8°	47.6°	49.0°
PPL+ (strict: accept only if $> 75°$)	54.4°	37.1°	18.9°

3. Further experimental details

In this section, we provide thorough details of the experimental motivation and details regarding settings and results.

Forthcoming sections include more details on data construction process, implementation of pose estimation (P3P and P6L) algorithms, procedures for quantitative/qualitative analysis presented in [5] and further discussions.

3.1. Details on dataset construction

For experiments, we utilized Cambridge [2, 3] and Energy Landscape [13] datasets. Among the Cambridge dataset [2, 3], we excluded "Street" and "Trinity Great Court" because they contained too many outliers, giving erroneous results like in Fig. 2. We constructed map and query datasets following the steps mentions in [10]. First, using the images provided, we ran COLMAP [9] pipeline to build sparse point clouds. This was required because the public dataset did not contain the information we needed such as SIFT descriptors. During the reconstruction, we confined to single camera, with intrinsic camera parameters assumed to be a simple radial. Next, we removed the information of query images from the point cloud. This includes correspondences between 2D and 3D points, SIFT descriptors of the query images and 2D points. Lastly, we ran bundle adjustment with camera extrinsic parameters fixed.

During the image localization experiments, we used ground truth images' intrinsic camera parameters rather than those generated during the reconstruction process. This is because COLMAP often failed to predict precise camera intrinsic parameters. Thus, we disregarded them and substituted to ground truth intrinsic parameters.

3.2. Details on experimental implementation

Detailed algorithm of two-peak finding (TPF) We know that the initial settings of OLC [10] and PPL are different, as our method holds two points in a single line. Therefore, naively adopting SPF [1] for our method casts doubt on whether the success of our method is attributable to the estimation of a deficient number of points. We present the peak-finding method with kernel density estimation (KDE) [6, 8] suitable for detecting N numbers of

Algorithm 1 Algorithmic procedures for Paired-Point Lifting (PPL) and PPL+. PPL+ has an additional stage of rejecting 3D lines being normal

Inputs: 3D points $\mathbf{X} \in \mathbb{R}^{N\times 3}$
procedure PPL($\mathtt{A}, \mathtt{B}$)
  for $i \leftarrow 1$ to N **do**
    $\mathtt{L}[i,:] \leftarrow \mathtt{Normalize}(\mathtt{A}[i,:] - \mathtt{B}[i,:])$
  end for
return $\{\mathtt{L},\ \mathtt{A},\ \mathtt{B}\}$
end procedure

$M \leftarrow 100$ ▷ Number of nearest neighbors
▷ Initialize matrix
$\mathtt{X}_\mathrm{A} \leftarrow$ *Empty Matrix* $\in \mathbb{R}^{N/2\times 3}$
$\mathtt{X}_\mathrm{B} \leftarrow$ *Empty Matrix* $\in \mathbb{R}^{N/2\times 3}$
$\mathtt{L}_\mathrm{closest} \leftarrow$ *Empty Matrix* $\in \mathbb{R}^{M\times 3}$

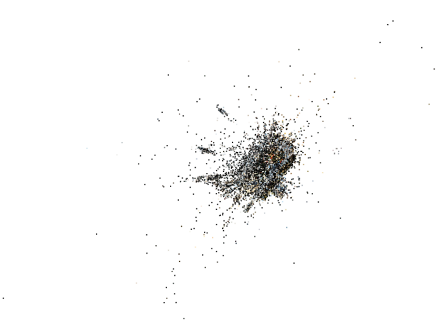
▷ Initialize line cloud
$\mathtt{X} \leftarrow \mathtt{Permute}(\mathtt{X})$
$\mathtt{L},\ \mathtt{A},\ \mathtt{B} \leftarrow \mathtt{PPL}(\mathtt{X}[:N/2,:],\ \mathtt{X}[N/2:,:])$

while $iter \leq \mathtt{MAX\_ITER}$ **do**
  ▷ Any 3D points presumed to be planar
  $\mathtt{L}_\mathrm{closest} \leftarrow \mathtt{Vector\ to\ M\ Closest\ Points}(\mathtt{X}[\mathtt{i},:], \mathtt{L}, \mathtt{X})$
  $[\mathtt{U}_\mathrm{plane}, \mathtt{S}_\mathrm{plane}, \mathtt{V}_\mathrm{plane}] \leftarrow \mathtt{SVD}(\mathtt{L}_\mathrm{closest})$
  ▷ Find the vector with the largest singular value
  $[\mathbf{v}_\mathrm{plane}] \leftarrow \mathtt{Representative\ Vector}(\mathtt{S}_\mathrm{plane}, \mathtt{V}_\mathrm{plane})$

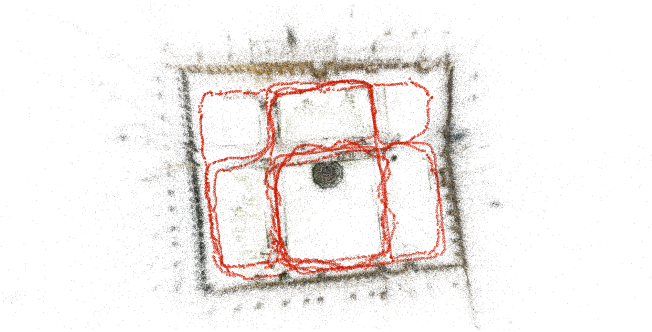
  $\mathtt{L}_\mathrm{orth} \leftarrow \mathtt{Cross\ Product}(\mathbf{v}_\mathrm{plane}, \mathtt{L}_\mathrm{closest})$
  $[\mathtt{U}_\mathrm{orth}, \mathtt{S}_\mathrm{orth}, \mathtt{V}_\mathrm{orth}] \leftarrow \mathtt{SVD}(\mathtt{L}_\mathrm{orth})$
  $[\mathbf{v}_\mathrm{orth}] \leftarrow \mathtt{Representative\ Vector}(\mathtt{S}_\mathrm{orth}, \mathtt{V}_\mathrm{orth})$

  $[\mathtt{C}] \leftarrow \mathtt{Dot\ Product}(\mathbf{v}_\mathrm{orth}, \mathtt{L}_\mathrm{closest})$
  ▷ Small dot product means planar points
  $[\mathtt{X}_\mathrm{in\_plane}, \mathtt{X}_\mathrm{out\_plane}] \leftarrow \mathtt{Get\ Planar\ Points}(\mathtt{C})$
  $\mathtt{X}_\mathrm{A}, \mathtt{X}_\mathrm{B} \leftarrow \mathtt{Add\ Non\ Planar\ Point}(\mathtt{X}_\mathrm{out\_plane})$

  if $\mathtt{X}_\mathrm{in\_plane} \leq 10\ \ OR\ \ !\mathtt{Decreasing}(\mathtt{X}_\mathrm{in\_plane})$ **then**
    break
  end if
  ▷ Perform PPL with existing planar 3D points
  $\mathtt{L},\ \mathtt{A},\ \mathtt{B} \leftarrow \mathtt{PPL}(\mathtt{X}_\mathrm{in\_plane\_A}, \mathtt{X}_\mathrm{in\_plane\_B}))$
  ▷ Update nns for in-plane points
  $iter \leftarrow iter + 1$
end while
▷ Return two sets of 3D points with their line directions
$\mathtt{L},\ \mathtt{A},\ \mathtt{B} \leftarrow \mathtt{PPL}(\mathtt{X}_\mathrm{A}, \mathtt{X}_\mathrm{B})$
Outputs: 3D lines $\{\mathtt{L},\ \mathtt{A},\ \mathtt{B}\}$

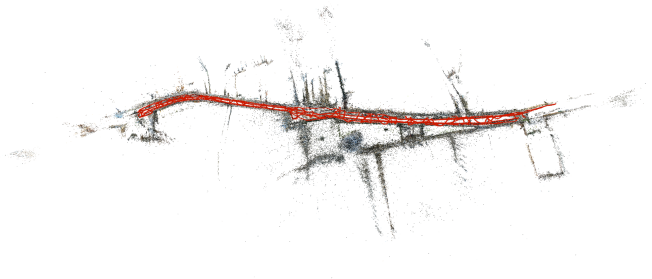


(a) 3D point cloud of the *Trinity Great Court* sequence in [3] reconstructed using COLMAP [9] contains outliers



(b) Bird's eye view of *Trinity Great Court* [3]

(c) 3D point cloud of the *Street* sequence in [3] reconstructed using COLMAP [9] contains potential outliers to the right



(d) Bird's eye view of *Street* [3]

Figure 2. Above illustrations of the 3D point clouds show both the *Trinity Great Court* and *Street* sequences reconstructed using COLMAP [9] contains a substantial amount of anomalous 3D points. We therefore excluded them from comparison

peaks. As denoted in [5], we find two peaks for the PPL-based line cloud, since the PPL-based line cloud consists of two 3D points.

TPF method depends on some hyper-parameters, such as the variance of the Gaussian kernel and the height of a local maximum, **h**. In Algorithm 2, the algorithm initially starts with **h = 0.2** and iterates until it finds two or more peaks or reaches **h = 10** where threshold height denoting minimum height between two peaks is 0.001. What usually happens is that, when two or more peaks are found, it returns the two highest peaks. If only one peak is found, it returns the peak along with second peak by adding a random noise to the first. If there is no peak present, we return two random floats $\in [-2, 2]$, which is the interval we empirically assume to make the good estimations. The last case is an extremely unlikely event and we have never encountered while conducting the experiments. Detailed algorithm chart is presented in Algorithm 2.

Discussion on increased 3D point errors for two-peak finding (TPF) During the experiments, we noticed that finding two or more peaks of a histogram is a non-trivial task. We make a further discussion to this problem that leads to high point errors for TPF shown in Table 1 from [5].

In Table 1 frpm [5], the reconstructed 3D point error actually increases substantially for TPF, which seems to contradict our motivation for finding a level playground for comparing SPF and TPF. However, the quality of the recovered images is improved when TPF is applied over SPF [1]. To justify our method to recover two points from PPL, we made further investigation showing that the peaks found by TPF also correspond with the human intuition by analyzing peak from the histogram of the closest-point-to-other-lines across 100 PPL-based lines. As shown in Table 2, when we asked people to pick two largest peaks from a histogram preprocessed by kernel density estimation, 93% of the times people picked the same peaks as 2-peak finding algorithm. The detailed experimental process is written below.

We manually analyzed a histogram of closest 3D points across 100 different PPL-based lines from *Apt1 luke*. From this we i) counted the number of notable peaks per histogram and ii) located two largest ones. We observed 2.55 peaks on average, meaning there are cases with more than two peaks due to complex nature of line distributions. The TPF algorithm performed quite accurate by correctly selecting the peaks 93% of the time. However, 59% of the time

Algorithm 2 Algorithmic procedures for two-peak finding (TPF).

Inputs: 3D point estimates $\beta \in \mathbb{R}^{\mathbb{N}}$
`MAX_H` $\leftarrow$ *maximum height of local maximum*
`MIN_H` $\leftarrow$ *minimum height of local maximum*

$X \leftarrow$ `Kernel Density Estimation`(β)
for *height* $\leftarrow$ `MAX_H` *to* `MIN_H` **do**
▷ Find peaks higher than the ***height*** from an array
  peaks $\leftarrow$ `Find Local Maxima`*(X, height)*
  if $peaks.size() \geq 2$ **then**
  ▷ Sort in descending order
    `Sort`*(peaks)*
    a, b $\leftarrow$ *peaks[0], peaks[1]*
  end if
end for
if $peaks.size() = 1$ **then**
  ▷ Estimate second peak by adding a random noise
  a, b $\leftarrow$ *peaks[0], peaks[0] + Random(-0.05, 0.05)*
else
  ▷ Estimate both peaks from random distribution
  a, b $\leftarrow$ *Random(-2, 2) + Random(-2, 2)*
end if
Outputs: Two peaks $\{a, b\}$

the actual peaks did not correspond to the location of original 3D points, meaning the issue briefly illustrated using two planes in Fig. 6 of [5] may be prevalent in real dataset.

In summary, we believe the complex nature of PPL-based lines make TPF and nontrivial task of finding multi-peaks to be erroneous, but TPF was robust and accurate algorithm we could implement to estimate two highest peaks of the given distribution of lines.

Average number of peaks	2.55
actual two peaks = two peaks found by ours	93%
actual two peaks = actual ground truth 3D points	59%

Table 2. Peak finding results verified through manual human inspection on *Apt1 luke* from Energy Landscape [13].

Image localization To compare the relocalization accuracy for point cloud, OLC, and PPL, we utilized the given code from [4] to estimate the absolute pose with P3P. [4] only provide generalized relative camera pose [12], and do not include an absolute pose estimation method suggested by [10] as P6L. Hence, we customized the code to estimate an absolute pose from 3D lines given a query image.

The problem setting for P6L is somewhat different than solving relative pose for two generalized camera models. In the P6L solver, the first camera, which is query image, is set as a pinhole camera and the second generalized camera is set as a line cloud. Therefore, we change the input for the estimation function to 3D lines from a line cloud and substituted score function discerning inliers as defined by [10, 11]. We also completed Levenberg-Marquardt(LM) refinement process for bundle adjustment of P6L. More details about the modifications from minimal solver from [12] to P6L [10] are explained in git code.

3.3. Detailed discussion on qualitative analysis

Image quality As mentioned in our submission [5], PPL has two descriptors per line, yielding 50% chance of swapped descriptors for each pair of points. Besides, point reconstruction errors are added when restoring point cloud from line cloud. In Fig. 6, we illustrated the effect of swapped features and point reconstruction errors. We added the noises on the locations of original point cloud to implement the errors. From the first row to the fourth row in the figure, each row shows the effect of swap in descriptors, while each columns shows the effect of noises added on the point locations. With the Fig. 6, we could see the synergistic effect of point reconstruction errors and swapped features. To indicate this effect, we showed the comparison between OLC [10] and 50% swapped point cloud with 2% noise in Fig. 5. For the comparison these to our methods, we also added the images from PPL and PPL+ in the same figure.

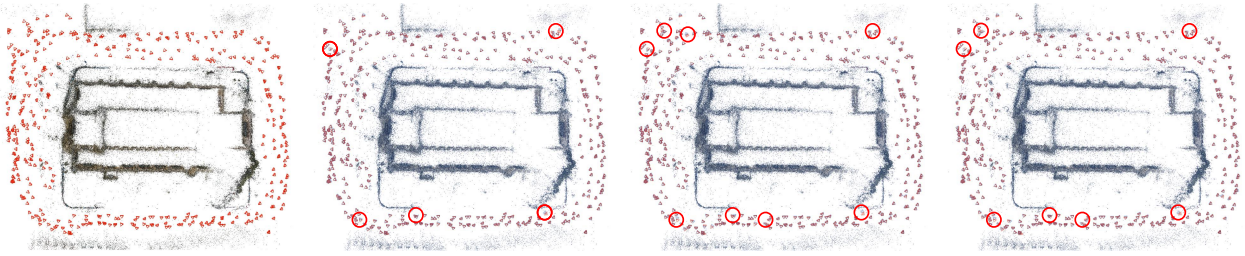
Pose estimation Visual localization is the fundamental problem of estimating camera location and orientation from input image with respect to a reference 3D scene. The qualitative comparison between ground truth camera poses and estimated camera poses from original 3D point cloud, OLC, and our proposed PPL are presented in Fig. 3. For clear visualization, we provide bird's eye view where every camera pose is visible. Ground truth camera poses are overlapped with camera poses estimated from each privacy preserving method. Red circles indicate poses with notable shifts from its ground truth poses.

Fig. 8, which was Fig 8 in [5], demonstrate image localization errors of original point cloud, OLC, PPL and PPL+. In general, image quality and pose estimation accuracy degrade as 3D points or lines get sparser. However, there are some cases which do not follow this tendency. We ran multiple tests to verify this and it showed that sparse 3D point cloud or line cloud occasionally outperform more dense ones. Such tendency is also observed in [10].

Scene geometry from InvSfM w/o RGB+SIFT Fig. 4 shows PPL further degrades scene geometry (compared to OLC [10]) when using InvSfM without RGB+SIFT.

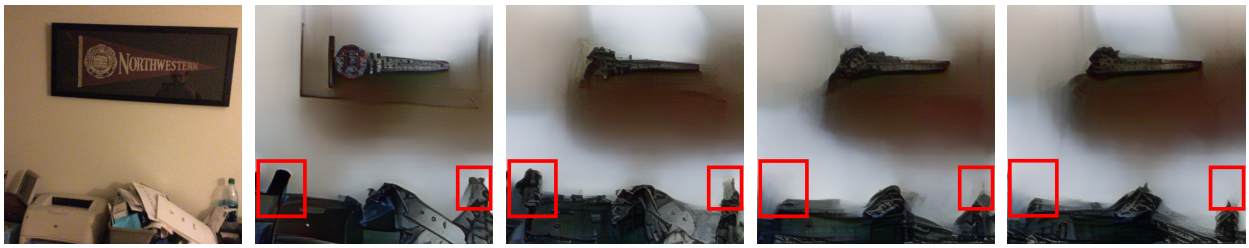
3.4. Details of quantitative analysis

Reconstructed point errors Reconstruction error is evaluated between the original 3D points and their recovered



(a) Ground truth (b) Original point cloud (P3P) (c) OLC [10] (P6L) (d) PPL (P6L)

Figure 3. Visualization of camera re-localization results using different scene representations (original 3D point cloud, original 3D line cloud [10] and proposed paired-point lifting (PPL)-based 3D line cloud). Query images are obtained from the St Mary's Church sequence in the Cambridge dataset [3]. Pseudo-ground truth camera poses obtained using COLMAP [9] are overlayed in red to each blue colored camera poses. Locations where camera poses significantly deviate from ground truths are marked in red circles. Note for (b) and (c), line cloud representations are used for pose estimation but point cloud is drawn just for the purpose of visualization. This indicates no significant difference between pose estimation results across different 3D scene representations.



(a) Ground truth (b) Point cloud (c) OLC [10] (d) PPL (e) PPL+

Figure 4. The results of recovered images without the information of RGB values an SIFT descriptor. Red marked square shows the scene details is disappeared in the revealed scene from PPL&PPL+.

ones. During this process, we disregard the obfuscation effect of feature descriptors—when SPF is applied, we compute error relative to the closer point out of two ground truth 3D points along the line. Similarly, when TPF method is applied, we compare the two estimated 3D points against two ground truth points and select the pairs with overall-minimum distance. This is performed to separate the impact of point estimation error from the impact of feature obfuscation in lowering image quality.

Besides, we present raw data from which Table 1 in [5] was analyzed. The graphs showing the distribution of the errors are not presented in the original paper due to the size limit. Fig. 7 shows that the ECDF curves of PPL(SPF), PPL(TPF), and PPL+(TPF) are shifted more to the right, meaning that the average error is larger for our methods. This further verifies that the large reconstructed 3D point errors of PPL and PPL+ specified in Table 1 of [5] is not due to few large errors. Table 1 of [5] provides overall reconstruction 3D point errors, quality of recovered images and pose estimation accuracy. Please note the results are shown after **averaging** across multiple sequences for each dataset.

Image quality assessment For line clouds (original and PPL-based), we use either 1-peak [1] or 2-peak (ours) finding algorithms to retrieve 3D point clouds. Then, we project the 3D point cloud to ground truth camera pose of input query image. The projection is discretized as a tensor and is fed into the InvSfM network [7] along with depths and descriptor information. The output (prediction) image is then compared against ground truth image using the PSNR, SSIM, and MAE metrics.

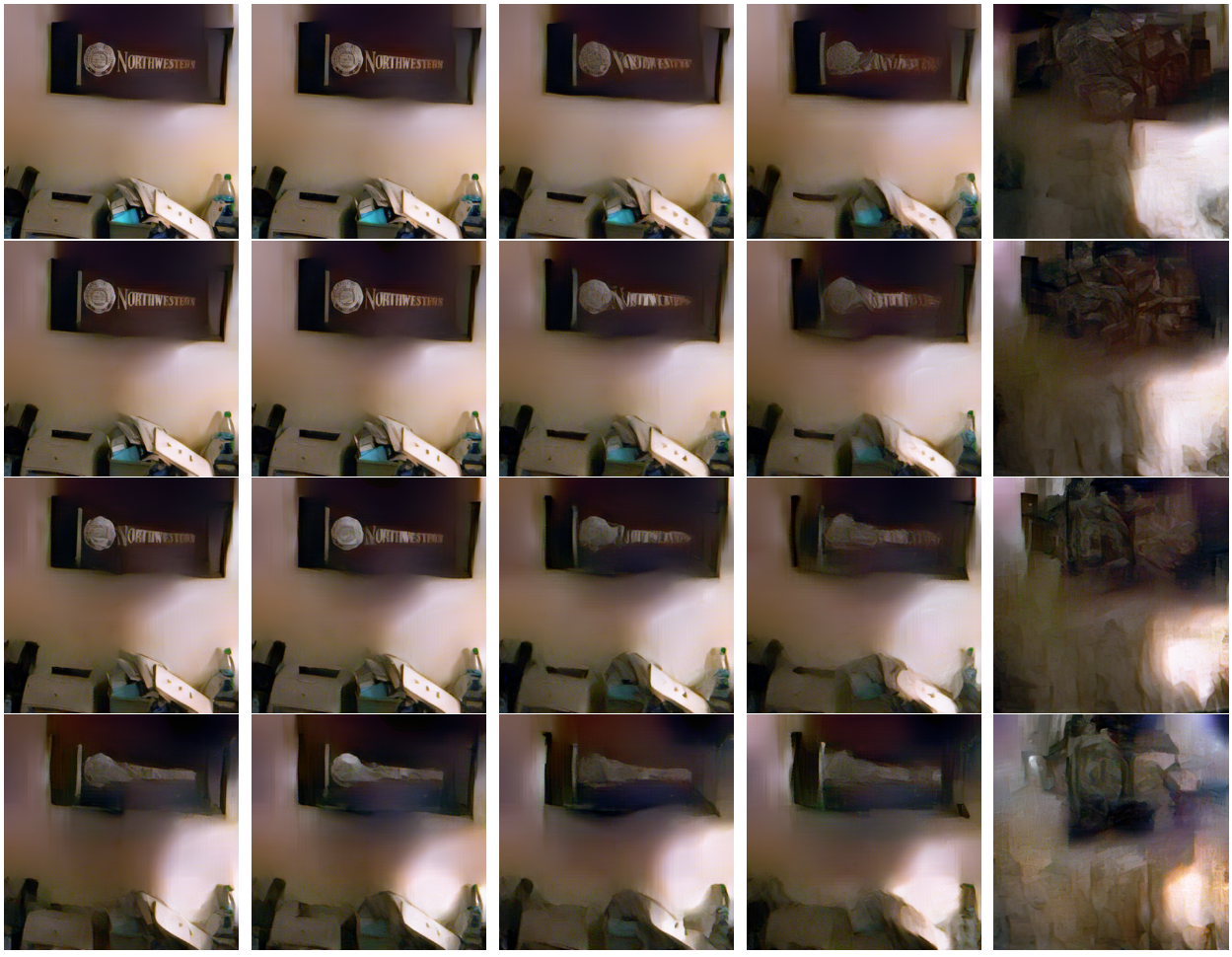
Pose estimation In Fig. 8, 9 we illustrate the empirical cumulative distribution function(ECDF) of camera pose errors of each data from Cambridge [3] and Energy Landscape [13]. The horizontal axis of the graph is limited to the half of the maximum error. However, for *Shop Facade"*, *Office2 5a*, *Office2 5b*, the axis is limited to 2% of the maximum error because these scene contain the points with large errors. Camera pose errors are calculated by comparing estimated pose with the ground truth pose which was generated using COLMAP [9] pipeline.

In terms of pose estimation, all of the privacy preserving methods such as OLC, PPL and PPL+ showed similar rotation errors and translation errors. This means that 50% sparsity in line cloud does not significantly harm pose estimation accuracy.

When finding mean and median errors of pose estimation, we first calculated mean and median errors from each dataset. Then, the final mean and median errors are calculated again from the previously acquired means and me-

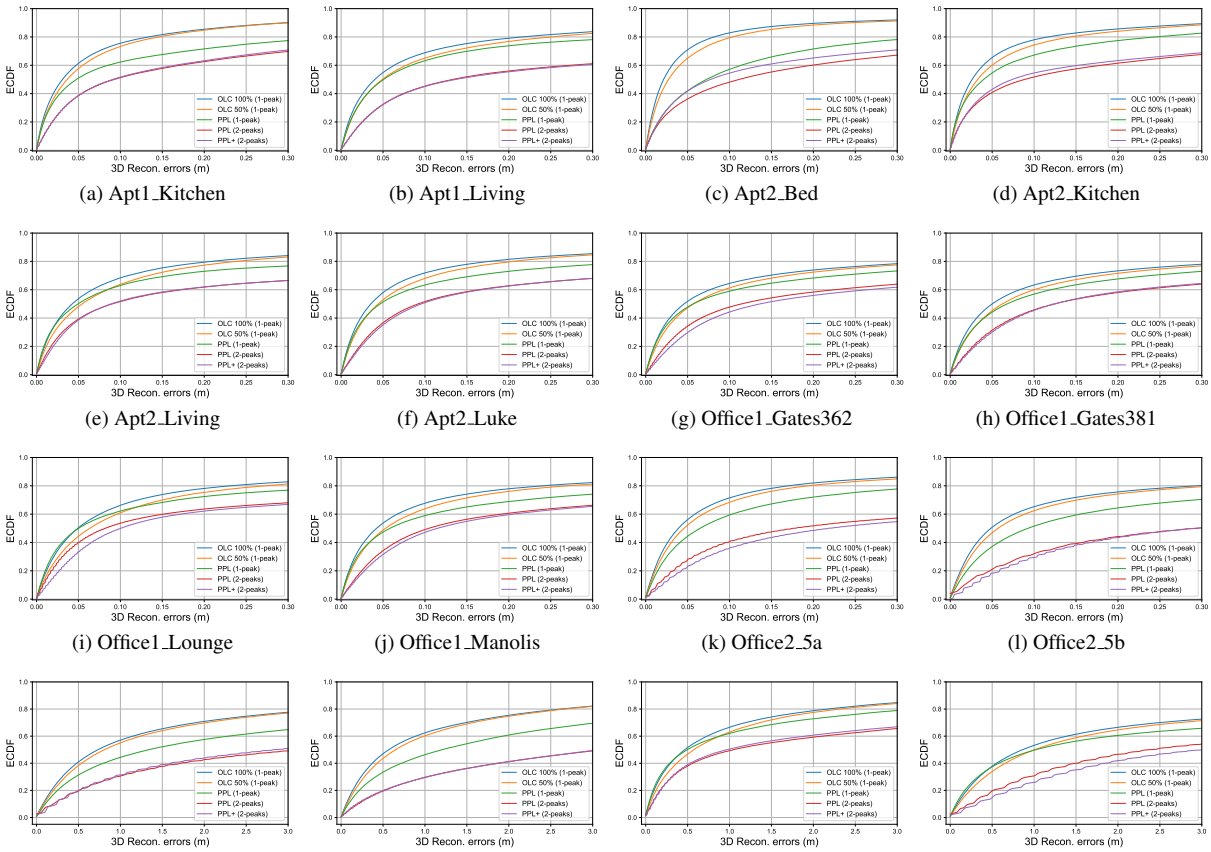
(a) OLC [10] (b) Noise 2%, swapped 50% (c) Ours (PPL), 2-peaks (d) Ours (PPL+), 2-peaks

Figure 5. Comparison between original OLC [10], point cloud with 2% of noise on the point location and 50% swapped descriptors, and our methods (PPL, PPL+). The second image shows the reconstructed image from point cloud with 2% of noise and 50% swapped descriptors, while the first, the third, and the fourth ones show the reconstruction from OLC, PPL, PPL+. The scene for the experiments is 'Apt2_luke' in Energy Landscape [13] dataset.



(a) Noise 0% (b) Noise 0.2% (c) Noise 1% (d) Noise 2% (e) Noise 20%

Figure 6. Effect of the combination of swapped features and point reconstruction errors on quality of reconstructed image. Each row shows reconstructed image with swapped descriptors in following order: 0%, 25%, 50%, 75%. To apply point reconstruction errors, we added noise on the point location of the original point cloud in following degrees: 0%, 0.2%, 1%, 2%, 20%. The scene for the experiments is *Apt2 luke* in Energy Landscape [13] dataset.



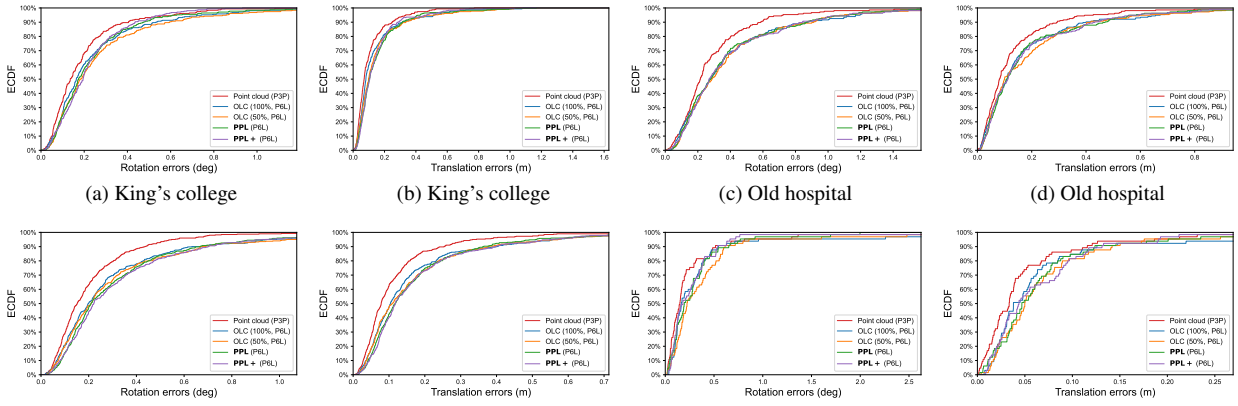
(a) Apt1_Kitchen (b) Apt1_Living (c) Apt2_Bed (d) Apt2_Kitchen

(e) Apt2_Living (f) Apt2_Luke (g) Office1_Gates362 (h) Office1_Gates381

(i) Office1_Lounge (j) Office1_Manolis (k) Office2_5a (l) Office2_5b

(m) King's college (n) Old Hospital (o) Shop Facade (p) St.Mary's Church

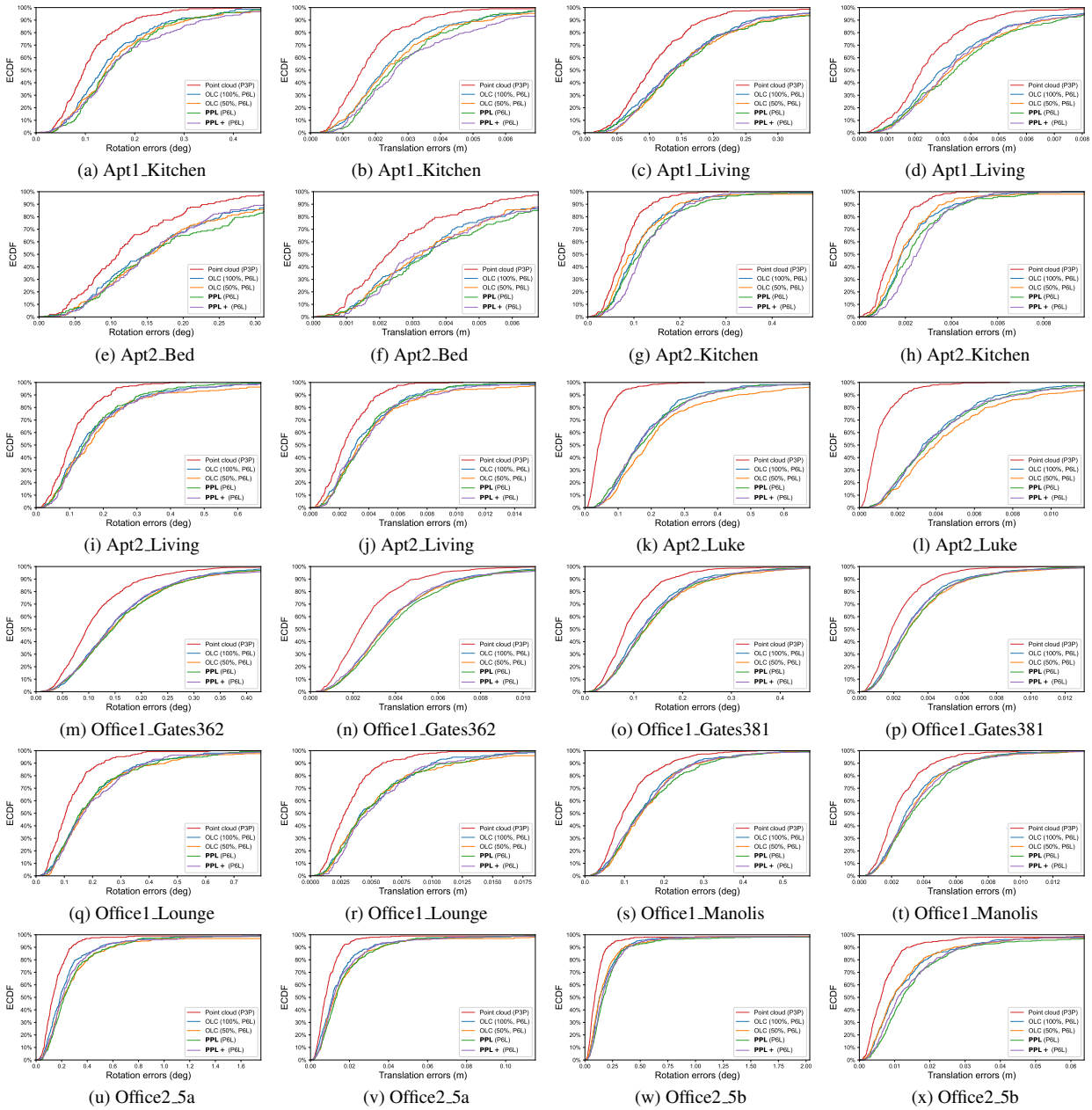
Figure 7. ECDF of reconstructed point errors across different datasets [3, 13] and 3D representations



(a) King's college (b) King's college (c) Old hospital (d) Old hospital

(e) St.Mary's Church (f) St.Mary's Church (g) Shop Facade (h) Shop Facade

Figure 8. ECDF of rotation & translation errors of estimated camera pose by each data from [3]



(a) Apt1_Kitchen (b) Apt1_Kitchen (c) Apt1_Living (d) Apt1_Living

(e) Apt2_Bed (f) Apt2_Bed (g) Apt2_Kitchen (h) Apt2_Kitchen

(i) Apt2_Living (j) Apt2_Living (k) Apt2_Luke (l) Apt2_Luke

(m) Office1_Gates362 (n) Office1_Gates362 (o) Office1_Gates381 (p) Office1_Gates381

(q) Office1_Lounge (r) Office1_Lounge (s) Office1_Manolis (t) Office1_Manolis

(u) Office2_5a (v) Office2_5a (w) Office2_5b (x) Office2_5b

Figure 9. ECDF of rotation & translation errors of estimated camera pose by each data from [13]

dians. In this way, we can derive more meaningful metric without being polluted from extremely skewed data.

To provide translation error metric in meters, we multiplied the translation error found using [4] by normalization coefficient. The coefficient is the mean of translation vectors after applying inverted extrinsic parameters of random image to every other image in the same dataset.

References

[1] Kunal Chelani, Fredrik Kahl, and Torsten Sattler. How privacy-preserving are line clouds? Recovering scene details from 3D lines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15663–15673, 2021. 2, 3, 5

[2] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6555–6564, 2017. 2

[3] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2938–2946, 2015. 2, 3, 5, 7

[4] Viktor Larsson. PoseLib - Minimal Solvers for Camera Pose Estimation. https://github.com/vlarsson/PoseLib, 2020. Accessed: 2022-10-30. 4, 8

[5] Chunghwan Lee, Jaihoon Kim, Chanhyuk Yun, and Je Hyeong Hong. Paired-point lifting for enhanced privacy-preserving visual localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17266–17275, 2023. 1, 2, 3, 4, 5

[6] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962. 2

[7] Francesco Pittaluga, Sanjeev J Koppal, Sing Bing Kang, and Sudipta N Sinha. Revealing scenes by inverting structure from motion reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 145–154, 2019. 5

[8] Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, pages 832–837, 1956. 2

[9] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 3, 5

[10] Pablo Speciale, Johannes L Schonberger, Sing Bing Kang, Sudipta N Sinha, and Marc Pollefeys. Privacy preserving image-based localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5493–5503, 2019. 1, 2, 4, 5, 6

[11] Pablo Speciale, Johannes L Schonberger, Sudipta N Sinha, and Marc Pollefeys. Privacy preserving image queries for camera localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pages 1486–1496, 2019. 4

[12] Henrik Stewénius, Magnus Oskarsson, Kalle Aström, and David Nistér. Solutions to minimal generalized relative pose problems, 2005. 4

[13] Julien Valentin, Angela Dai, Matthias Niessner, Pushmeet Kohli, Philip Torr, Shahram Izadi, and Cem Keskin. Learning to navigate the energy landscape. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 323–332, 2016. 2, 4, 5, 6, 7, 8