Splat-LOAM: Gaussian Splatting LiDAR Odometry and Mapping

Supplementary Material

In this supplementary material, we provide additional details on various components and design choices that were not fully elaborated in the main paper. These include the computation of the camera matrix, the rationale behind the bounding box computation, the incorporation of color features, the analytical derivation of our spherical rasterizer and extended tables on experimental analysis.

A. Camera Matrix

Using hard-coded field of views from the sensor's datasheet may lead to empty areas inside the image (i.e., when parts of the FoV contain no observable environment or, due to rounding errors). To solve these issues, we recompute the field of views, along with a camera matrix, for each input LiDAR point cloud.

Let $\{\mathbf{p}_q\}_{q=1}^{N}$ be a set of points expressed at the sensor origin. Let $\{\mathbf{v}_q = (\gamma, \theta, 1)^T = \psi(\mathbf{p}_q)\}_{q=1}^{N}$ be the same set of points expressed in spherical coordinates. From this representation, we can directly estimate the camera matrix $\mathbf{K}$ as follows. First, we compute the maximum horizontal and vertical angular values within the set:

$$\gamma_m = \min_{\gamma}\{\mathbf{v}_q\} \qquad \gamma_M = \max_{\gamma}\{\mathbf{v}_q\}, \tag{23}$$

$$\theta_m = \min_{\theta}\{\mathbf{v}_q\} \qquad \theta_M = \max_{\theta}\{\mathbf{v}_q\}. \tag{24}$$

Moreover, we compute the horizontal $\mathrm{FoV}_h = \gamma_M - \gamma_m$ and vertical $\mathrm{FoV}_h = \theta_M - \theta_m$ field of views and, provided an image size (H, W), we estimate the camera matrix as:

$$\mathbf{K}(\{\mathbf{p}_q\}) = \begin{pmatrix} -\frac{W-1}{\mathrm{FoV}_h} & 0 & \frac{W}{2}\left(1 + \frac{\gamma_M+\gamma_m}{\mathrm{FoV}_h}\right) \\ 0 & -\frac{H-1}{\mathrm{FoV}_v} & \frac{H}{2}\left(1 + \frac{\theta_M+\theta_m}{\mathrm{FoV}_v}\right) \\ 0 & 0 & 1 \end{pmatrix} \tag{25}$$

B. Bounding Box

In this section, we report a supplementary study concerning the computation of the tightly aligned bounding box on spherical images. Efficiently computing the tightly aligned bounding box for a splat on the view space requires solving a 4-th-order polynomial due to the complexity of the underlying manifold. While fixing the azimuth angle γ results in a planar surface in $\mathbb{R}^3$, fixing the altitude angle θ leads to a cone subspace in $\mathbb{R}^3$. To find the tightly aligned bounding box, we should search the spherical coordinates (γ, θ) that exactly intersect tangentially the splat space at a distance 3σ from the origin. Projecting the α-plane onto the splat frame results in a line, and the intersection condition can be solved via a linear equation. Projecting the γ-cone onto the splat's frame results in a parametric 2D conic equation. Enforcing two tangent solutions leads to a polynomial of 4th-degree. Given the small image sizes of LiDAR images and the relatively high cost of solving higher-order polynomials, we opt for an easier but less optimal solution. We relax the tight constraint and obtain an image-space bounding box by projecting the individual bounding box vertices. This typically results in a bounding box that includes more pixels but is faster in computation.

C. On color features

Modern LiDARs provide a multitude of information on the beam returns. Specifically, they provide details concerning the mean IR light level (ambient) and the returned intensity (intensity). Through this information, it is also possible to compute the *reflectivity* of the surface using the inverse square law for Lambertian objects in far fields. Throughout this study, we opted to omit the color information to focus on the geometric reconstruction capabilities of our approach. Moreover, we think incorporating intensity and reflectivity channels can pose a challenge due to the inherent nature of LiDARs. Both properties cannot be explicitly related to a portion of the space but rather from a combination of the surface and sensor position with respect to the former.

D. Rasterizer Details

In this section, we describe the process of rasterization over spherical images. First, we provide a detailed analysis of the rasterization process for a Gaussian primitive. Furthermore, we provide the analytical derivatives for the components of the process. Recall that a Gaussian primitive $\mathcal{G}$ is defined by its centroid $\boldsymbol{\mu} \in \mathbb{R}^3$, its covariance matrix decomposed as a rotation matrix $\mathbf{R} \in \mathbb{SO}(3)$ and a scaling matrix $\mathbf{S}$, and its opacity $o \in \mathbb{R}$. To obtain the homogeneous transform that maps points (α, β) from the splat-space to the sensor-space c, we decouple the axes of the rotation matrix $\mathbf{R} = (\mathbf{t}_\alpha, \mathbf{t}_\beta, \mathbf{t}_n)$ and the per-axis scaling parameters $\mathbf{s} = (s_\alpha, s_\beta)$, and assume $\mathbf{T}_w^c \in \mathbb{SE}(3)$ be the transform the world in camera frame. By concatenating $\mathbf{T}_w^c$ with Eq. (3), we obtain the following transform:

$$\mathbf{T}_{4\times 3} = \mathbf{T}_w^c \mathbf{H} = \begin{pmatrix} \mathbf{b}_\alpha & \mathbf{b}_\beta & \mathbf{b}_c \\ 0 & 0 & 1 \end{pmatrix}, \tag{26}$$

where $\mathbf{b}_c = \mathbf{R}_w^c \boldsymbol{\mu} + \mathbf{t}_w^c$. We omit the third column of $\mathbf{T}$, which is zeroed by construction.

D.1. Forward Process

In this section, we describe the rasterization process for a pixel $\mathbf{u} = (u, v)$. We assume that primitives are already pre-sorted. As described in Sec. 3.2.2, we compute the orthogonal planes in splat-space by pre-multiplying each plane by $\mathbf{T}$:

$$\mathbf{h}_\alpha = \mathbf{T}^T \mathbf{h}_x \qquad \mathbf{h}_\beta = \mathbf{T}^T \mathbf{h}_y, \tag{27}$$

and compute the intersection point $\hat{\mathbf{p}}$:

$$\hat{\mathbf{p}} = \mathbf{h}_\alpha \times \mathbf{h}_\beta \tag{28}$$

$$\hat{\mathbf{s}} = (\hat{s}_\alpha, \hat{s}_\beta) = \left(\frac{\hat{\mathbf{p}}_x}{\hat{\mathbf{p}}_z}, \frac{\hat{\mathbf{p}}_y}{\hat{\mathbf{p}}_z} \right)^T. \tag{29}$$

We use $\hat{\mathbf{s}}$ to estimate two quantities. First, we measure the Gaussian kernel at the intersection point $\mathcal{G}(\hat{\mathbf{s}})$ to compute the Gaussian density

$$\alpha = o\mathcal{G}(\hat{\mathbf{s}}), \tag{30}$$

and second, we compute the range as

$$\boldsymbol{\nu} = \hat{s}_\alpha \mathbf{b}_\alpha + s_\beta \mathbf{b}_\beta + \mathbf{b}_c \tag{31}$$

$$d = \|\boldsymbol{\nu}\|. \tag{32}$$

We follow Eq. (5), Eq. (6), and Eq. (7) to α-blend the sorted Gaussians and compute the pixel contributions.

D.2. Gradient Computation

From the rasterizer perspective, we assume to already have the per-pixel channel derivatives, namely the depth $\frac{\partial \mathcal{L}}{\partial d} \in \mathbb{R}$ and normal $\frac{\partial \mathcal{L}}{\partial \mathbf{n}} \in \mathbb{R}^3$. To improve the readability, each partial derivative also includes its dimension using the $\frac{\partial A}{\partial B}\big|_{\dim(A)\times\dim(B)}$ notation. Finally, we show the computation for the k-th Gaussian over the m Gaussians contributing to the pixel.
First, we derive the gradients w.r.t the density:

$$\left.\frac{\partial d}{\partial d_k}\right|_{1\times1} = d_k A_k - \frac{B_{d,k}}{1-\alpha_k} \tag{33}$$

$$\left.\frac{\partial \mathbf{n}}{\partial \mathbf{n}_k}\right|_{1\times3} = \mathbf{n}_k A_k - \frac{B_{\mathbf{n},k}}{1-\alpha_k} \tag{34}$$

$$\left.\frac{\partial \mathcal{L}}{\partial \alpha_k}\right|_{1\times1} = \frac{\partial \mathcal{L}_d}{\partial d}\frac{\partial d}{\partial \alpha_k} + \frac{\partial \mathcal{L}_n}{\partial \mathbf{n}}\frac{\partial \mathbf{n}}{\partial \mathbf{n}_k}, \tag{35}$$

where $A_k = \prod_{i=1}^{k-1}(1-\alpha_i)$, $B_{d,k} = \sum_{i>k} d_i \alpha_i A_i$, and $B_{\mathbf{n},k} = \sum_{i>k} \mathbf{n}_i \alpha_i A_i$. We leverage the sorting of the primitives to efficiently compute these values during the backpropagation of the gradients.

Furthermore, we propagate the gradients to the homogeneous transform matrix $\mathbf{T}$ from the intersection of planes $\hat{\mathbf{p}}_k$:

$$\begin{aligned}\left.\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{s}}_k}\right|_{1\times2} &= \frac{\partial \mathcal{L}}{\partial \alpha}\frac{\partial \alpha}{\partial \hat{\mathbf{s}}_k} + \frac{\partial \mathcal{L}}{\partial d_k}\frac{\partial d_k}{\partial \hat{\mathbf{s}}_k} \\ &= -\frac{\partial \mathcal{L}}{\partial \alpha}\alpha_k \hat{\mathbf{s}}_k^T + \frac{\partial \mathcal{L}}{\partial d_k}\frac{\alpha_k A_k}{d_k}\begin{pmatrix}\boldsymbol{\nu}^T \mathbf{b}_\alpha \\ \boldsymbol{\nu}^T \mathbf{b}_\beta\end{pmatrix}^T\end{aligned} \tag{36}$$

$$\begin{aligned}\left.\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{p}}_k}\right|_{1\times3} &= \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{s}}_k}\frac{\partial \hat{\mathbf{s}}_k}{\partial \hat{\mathbf{p}}_k} \\ &= \frac{1}{\hat{\mathbf{p}}_z}\begin{pmatrix}\partial\mathcal{L}/\partial\hat{\mathbf{s}}_\alpha \\ \partial\mathcal{L}/\partial\hat{\mathbf{s}}_\beta \\ -\frac{\hat{\mathbf{p}}_x \partial\mathcal{L}/\partial\hat{\mathbf{s}}_\alpha + \hat{\mathbf{p}}_y \partial\mathcal{L}/\partial\hat{\mathbf{s}}_\beta}{\hat{\mathbf{p}}_z}\end{pmatrix}^T.\end{aligned} \tag{37}$$

Thus, we can derive the gradients over the matrix $\mathbf{T}$. We keep the three accumulators $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_\alpha}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_\beta}$, and $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_c}$ decoupled to correctly integrate the contributions from each pixel:

$$\boldsymbol{\rho}_\alpha = \left(\frac{\partial \mathcal{L}}{\partial \mathbf{p}_k} \times \mathbf{h}_\alpha\right) \qquad \boldsymbol{\rho}_\beta = \left(\frac{\partial \mathcal{L}}{\partial \mathbf{p}_k} \times \mathbf{h}_\beta\right) \tag{38}$$

$$\begin{aligned}\left.\frac{\partial \mathcal{L}}{\partial \mathbf{b}_\alpha}\right|_{1\times3} &= \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{p}}_k}\frac{\partial \hat{\mathbf{p}}_k}{\partial \mathbf{b}_\alpha} + \frac{\partial \mathcal{L}}{\partial d_k}\frac{\partial d_k}{\partial \mathbf{b}_\alpha} \\ &= -\boldsymbol{\rho}_{\beta,1}\mathbf{h}_x + \boldsymbol{\rho}_{\alpha,1}\mathbf{h}_y + \frac{\mathcal{L}}{\partial d_k}\frac{\hat{s}_\alpha}{d_k}\boldsymbol{\nu}^T\end{aligned} \tag{39}$$

$$\begin{aligned}\left.\frac{\partial \mathcal{L}}{\partial \mathbf{b}_\beta}\right|_{1\times3} &= \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{p}}_k}\frac{\partial \hat{\mathbf{p}}_k}{\partial \mathbf{b}_\beta} + \frac{\partial \mathcal{L}}{\partial d_k}\frac{\partial d_k}{\partial \mathbf{b}_\beta} \\ &= -\boldsymbol{\rho}_{\beta,2}\mathbf{h}_x + \boldsymbol{\rho}_{\alpha,2}\mathbf{h}_y + \frac{\mathcal{L}}{\partial d_k}\frac{\hat{s}_\beta}{d_k}\boldsymbol{\nu}^T\end{aligned} \tag{40}$$

$$\begin{aligned}\left.\frac{\partial \mathcal{L}}{\partial \mathbf{b}_c}\right|_{1\times3} &= \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{p}}_k}\frac{\partial \hat{\mathbf{p}}_k}{\partial \mathbf{b}_c} + \frac{\partial \mathcal{L}}{\partial d_k}\frac{\partial d_k}{\partial \mathbf{b}_c} \\ &= -\boldsymbol{\rho}_{\beta,3}\mathbf{h}_x + \boldsymbol{\rho}_{\alpha,3}\mathbf{h}_y + \frac{\mathcal{L}}{\partial d_k}\frac{1}{d_k}\boldsymbol{\nu}^T,\end{aligned} \tag{41}$$

where $\boldsymbol{\rho}_{k,i}$ is the i-th element of $\boldsymbol{\rho}_k$.
Finally, we compute the gradients w.r.t. the Gaussian parameters.

$$\left.\frac{\partial \mathcal{L}}{\partial \mathbf{R}_k}\right|_{3\times3} = \left(s_\alpha \frac{\partial \mathcal{L}}{\partial \mathbf{b}_\alpha}^T \quad s_\beta \frac{\partial \mathcal{L}}{\partial \mathbf{b}_\beta}^T \quad \frac{\partial \mathcal{L}}{\partial \mathbf{n}_k}^T \right) \mathbf{R}_w^c \tag{42}$$

$$\left.\frac{\partial \mathcal{L}}{\partial \mathbf{S}_k}\right|_{1\times2} = \left(\frac{\partial \mathcal{L}}{\partial \mathbf{b}_\alpha}\mathbf{R}_w^c \mathbf{R}_{[1]} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}_\alpha}\mathbf{R}_w^c \mathbf{R}_{[1]} \right) \tag{43}$$

$$\left.\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k}\right|_{1\times3} = \frac{\partial \mathcal{L}}{\partial \mathbf{b}_c}\mathbf{R}_w^c, \tag{44}$$

$$\left.\frac{\partial \mathcal{L}}{\partial o_k}\right|_{1\times1} = \frac{\partial \mathcal{L}}{\partial \alpha_k}\exp\left(-\frac{1}{2}\mathbf{s}_k^T \mathbf{s}_k\right) \tag{45}$$

where $\mathbf{R}_{[i]}$ is the i-th column of $\mathbf{R}$.

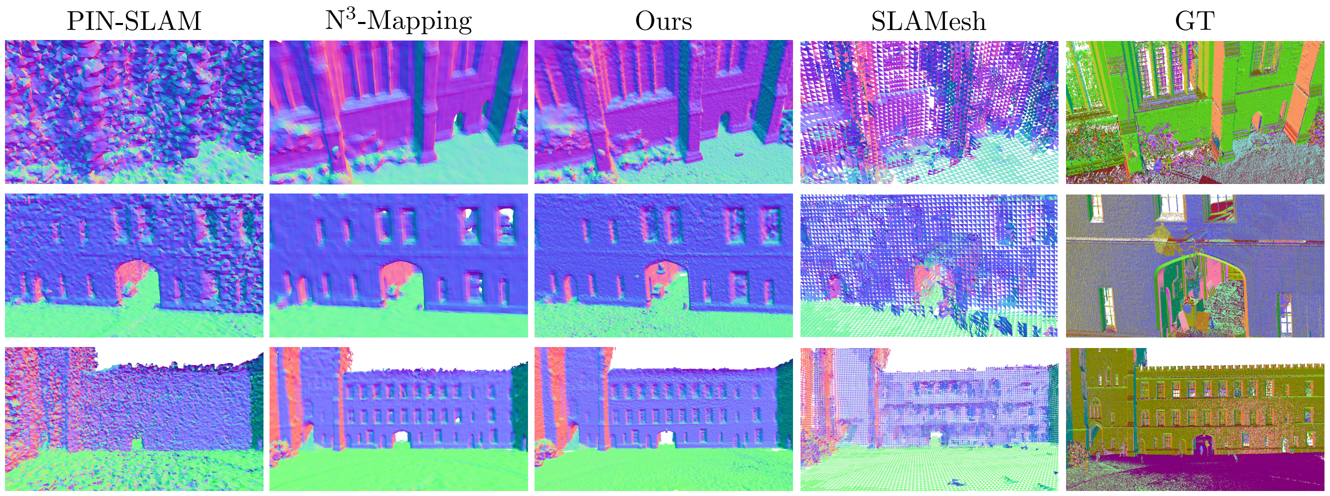

Figure D.1. **Qualitative Mapping Results.** The images show the mapping results for different pipelines in the quad-easy sequence. We also include the SLAMesh pipeline, which was evaluated on a self-estimated trajectory.

E. Additional Experiment Results

This section contains additional results that we did not include in the main paper due to lack of space. Specifically, Tab. B contains the RPE results that were used to generate Fig. 4. Moreover, the additional mapping evaluation metrics, namely Accuracy and Completeness for each dataset are reported. Specifically, the results of Newer College [50] are reported in Tab. E.2, the results of Oxford Spires [41] are reported in Tab. E.3 and the ones of Mai City [43] are reported in Tab. E.4.

Method	quad	math	keble	bodl	observ	pincio	spagna	campus	mai-1	mai-2	avg [%]
point2plane	0.0605	0.0103	0.1715	0.2110	0.1747	0.0074	0.0078	0.0083	fail	fail	8.14
SLAMesh	0.0043	0.0039	0.0060	0.0045	0.0043	fail	fail	fail	0.003	0.0028	0.41
LOAM	0.0035	0.0028	0.0047	0.0034	0.0055	0.0053	0.0022	0.0016	0.0045	0.0049	0.38
MAD-ICP	0.0036	0.0037	0.0036	0.0036	0.0035	0.0024	0.0008	0.0007	0.0032	0.004	0.29
PIN-SLAM	0.0041	0.0019	0.0038	0.0025	0.0028	0.0018	0.0013	0.0056	0.0026	0.0032	0.30
Ours	0.0057	0.0038	0.0078	0.0087	0.0052	0.0027	0.0033	0.0075	0.0069	0.0071	0.59

Table B. **Tracking RPE evaluation (%).** Averages in red are computed without failed sequences.

	quad-easy		math-easy	
Method	Acc ↓	Com ↓	Acc ↓	Com ↓
OpenVDB	11.45	4.38	11.69	10.55
VoxBlox	20.36	12.64	11.82	12.04
N^3-Mapping	6.32	9.75	fail	fail
PIN-SLAM	15.28	10.5	16.68	12.69
Ours	6.64	4.09	8.26	10.05

Table E.2. **Mapping results on the Newer College Dataset**. The table reports Accuracy and Completeness for each sequence.

	keble-02		bodleian-02		observatory-01	
Method	Acc ↓	Com ↓	Acc ↓	Com ↓	Acc ↓	Com ↓
OpenVDB	7.46	6.92	10.34	4.68	9.58	9.60
VoxBlox	15.81	14.25	18.92	11.56	15.09	15.15
N^3-Mapping	6.21	7.82	10.16	5.62	8.27	10.44
PIN-SLAM	13.73	9.94	14.34	7.14	16.91	12.07
Ours	6.18	8.69	10.87	4.33	9.35	11.76

Table E.3. **Mapping results on the Oxford Spires Dataset**. The table reports Accuracy and Completeness for each sequence.

	mai-01		mai-02	
Method	Acc ↓	Com ↓	Acc ↓	Com ↓
OpenVDB	3.04	3.63	3.01	3.51
VoxBlox	11.37	2.71	7.14	4.46
N^3-Mapping	2.62	2.66	2.75	2.76
PIN-SLAM	7.14	2.68	6.94	2.69
Ours	2.72	4.57	4.15	5.95

Table E.4. **Mapping results on the Mai City Dataset**. The table reports Accuracy and Completeness for each sequence.

F. Motion Distortion

Throughout the experiments, we noticed that Splat-LOAM is very sensitive to the motion distortion effect caused by the continual acquisition of LiDARs. Figure F.1 shows how the projective error over the estimated model changes while the sensor rotates during the acquisition, hindering both the registration and mapping phases. We plan to compensate for the motion distortion effect by simultaneously estimating the sensor pose and velocity.

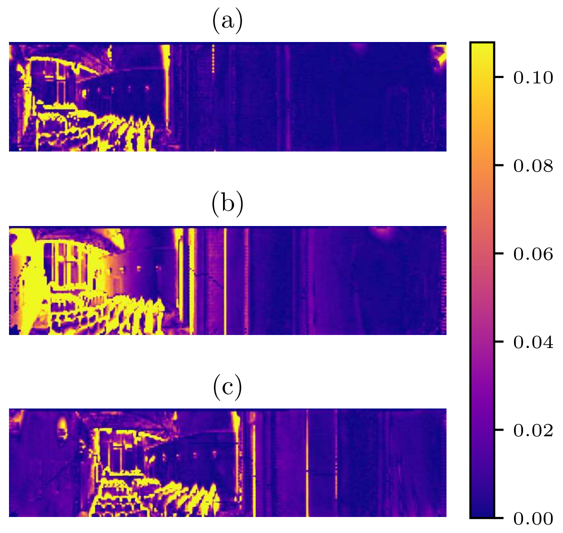

Figure F.1. **Effects of motion distortion during registration.** The images show the projective range error between our model and the incoming measurement **(a)** before a rotation, **(b)** during a rotation, and **(c)** after the rotation. The images are resized over the horizontal axis for visibility purposes. The error is expressed in meters.

G. Additional Qualitative Comparison

In this section, we show additional results over the meshing reconstruction. Figure D.1 includes the results we obtained using the work of Ruan *et al.* [32]. We remark that we did not include these results in the manuscript as we could not run the official implementation over the Ground Truth trajectory. Additionally, Figure G.1 shows the reconstruction results over the Oxford Spires dataset [41].

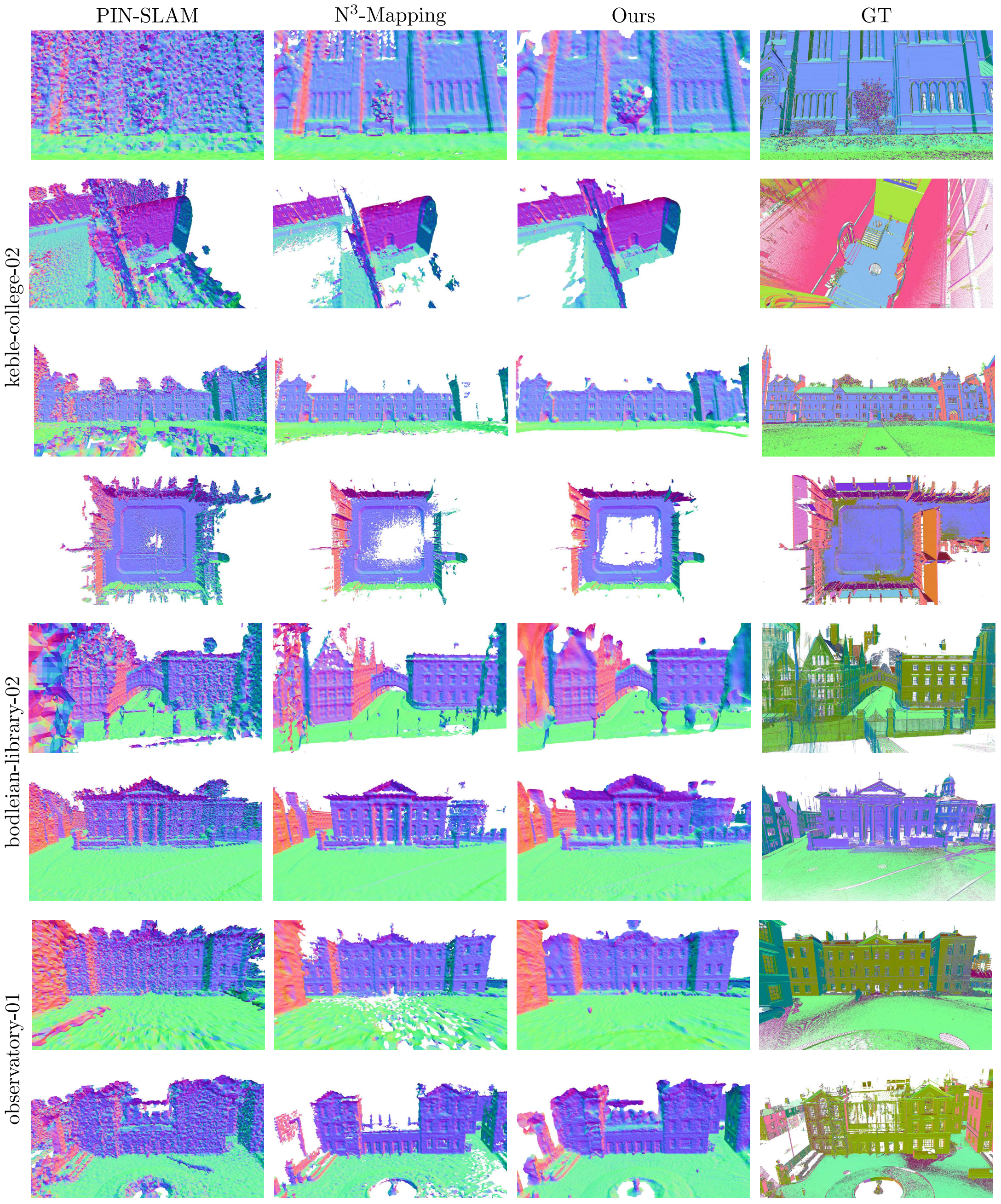

Figure G.1. **Qualitative Mapping Results.** The images show the mapping results for different pipelines in the Oxford Spires dataset [41].