High-Quality RGB-D Reconstruction via Multi-View Uncalibrated Photometric Stereo and Gradient-SDF — Supplementary Material

Lu Sang[1,2] Björn Häfner[1,2] Xingxing Zuo[1]
Daniel Cremers[1,2]
[1]Technical University of Munich
[2]Munich Center for Machine Learning
{lu.sang, bjoern.haefner, Xingxing.Zuo, cremers}@tum.de

This Supplement contains information on code, datasets, some more visualizations of results, and the mathematical details of the proposed algorithm.

1. Used Code and Datasets

The following table summarizes the code and datasets we use for evaluation and comparison. We compare with the methods whose code is publicly available and can be successfully compiled and run. Only for the work of Bylow el at. [8] we implement their method ourselves because their method has similarities with our proposed method, but the author has not published the code yet. Our code and recorded datasets will be made publicly available after the publication.

		code/data	year	link	license
[39]	TUM RGB-D Benchmark	dataset	2012	https://vision.in.tum.de/data/datasets/rgbd-dataset	CC BY 4.0
[22]	Intrinsic3D Dataset	dataset	2017	https://vision.in.tum.de/data/datasets/intrinsic3d	CC BY 4.0
[46]	multi-view stereo dataset	dataset	2015	http://graphics.stanford.edu/projects/vsfs/	CC BY-NC-SA 4.0
[34]	3D Scene Data	dataset	2014	https://qianyi.info/scenedata.html	-
[39]	TUM RGB-D Benchmark	code	2012	https://vision.in.tum.de/data/datasets/rgbd-dataset/tools	BSD-2
[36]	BAD SLAM	code	2019	https://github.com/ETH3D/badslam	BSD-3
[38]	Gradient-SDF	code	2022	https://github.com/c-sommer/gradient-sdf	BSD-3
[22]	intrinsic3d	code	2017	https://github.com/NVlabs/intrinsic3d	BSD-3
[40]	NeuS	code	2021	https://github.com/Totoro97/NeuS	MIT License
[42]	volSDF	code	2021	https://github.com/lioryariv/volsdf	MIT License

Table S2. Used datasets and code in our submission, together with reference, link, and license.

2. Visualization Results on TUM RGB-D Datasets

We evaluate our method on the TUM RGB-D datasets to validate the camera pose refinement. Notably we take lighting conditions into account. The qualitative Root Mean Square Error (RMSE) of the absolute trajectory error is shown in Table 1 of the main paper. Here we visualize in the Figure S1 the refined point cloud of more sequences and compare with two other baseline methods: badslam [36] and gradient-SDF [38]. For each sequence, we take 300 frames as input to initialize SDF with 2cm voxel size and the camera poses, then 30 keyframes are chosen using a sharpness detector [3] to avoid manual selection. An up-sampling is applied after 10 iterations.

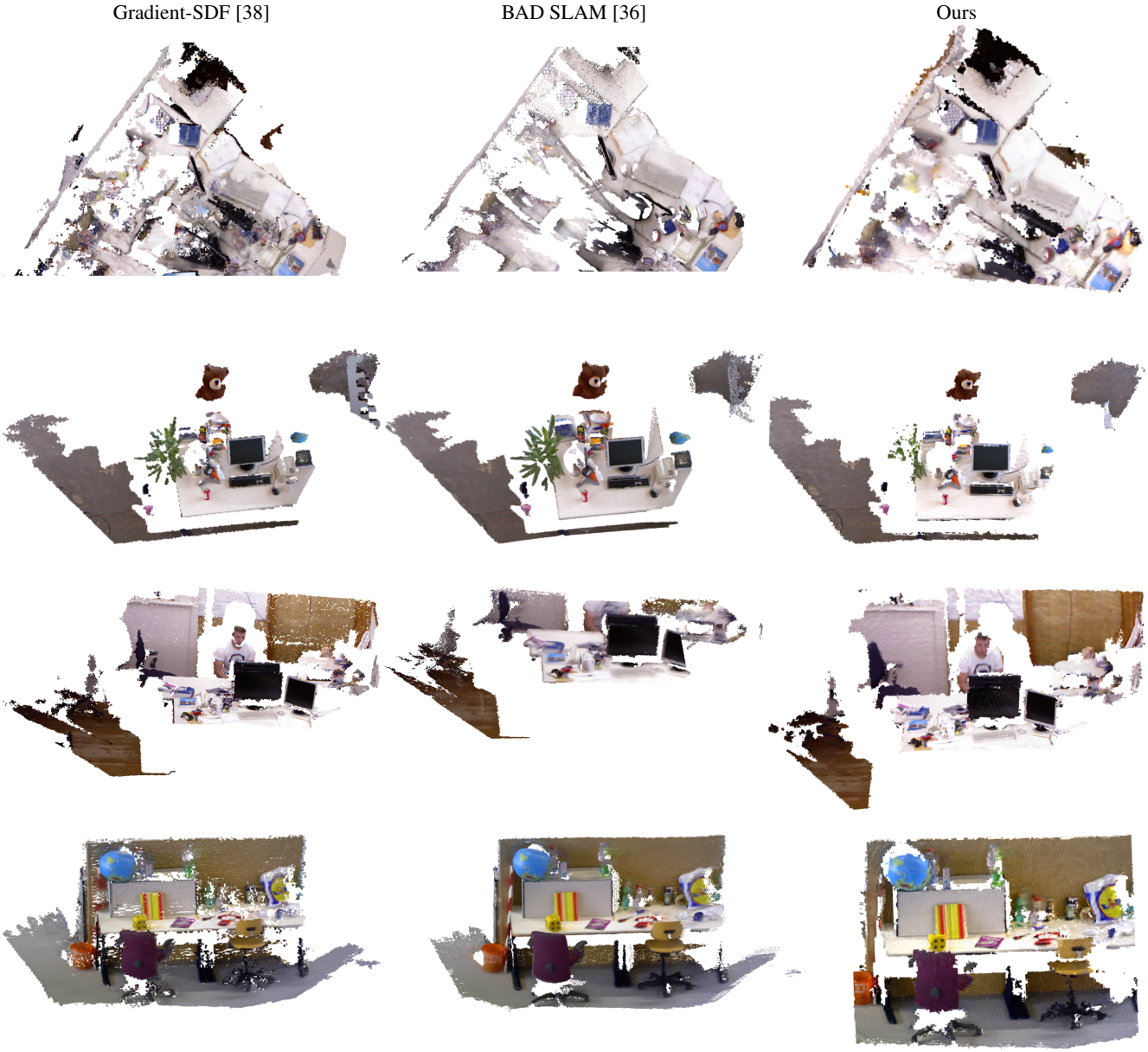

Figure S1. The reconstruction results on TUM RGB-D [39] sequences compare against two other camera tracking and refinement methods. While different methods perform better across sequences, our method achieves comparable results to the best-performed method. In addition, the texture is sharper and more precise.

3. Surface Geometry Refinement

3.1. Synthetic Data

We compare our results with two classical methods [8, 22] and two neural-rendering-based methods [40, 42]. Figure S2 shows the input RGB image with ground truth depth and noisy depth we use for validate methods. Our mesh accomplishes more detailed surface reconstruction and a more faithful albedo. Note that the work of Maier el at. [22] does not recover albedo but only recovers the gradient of its albedo. The color on the mesh is the average intensities of each voxel. The work [8] estimates the albedo but fails to recover it with good quality. With only RGB and camera poses as input, volSDF [42] and NeuS [40] can recover the mesh without color with a satisfactory quality but both works need train over 24 hours, and the resolutions are still not good.

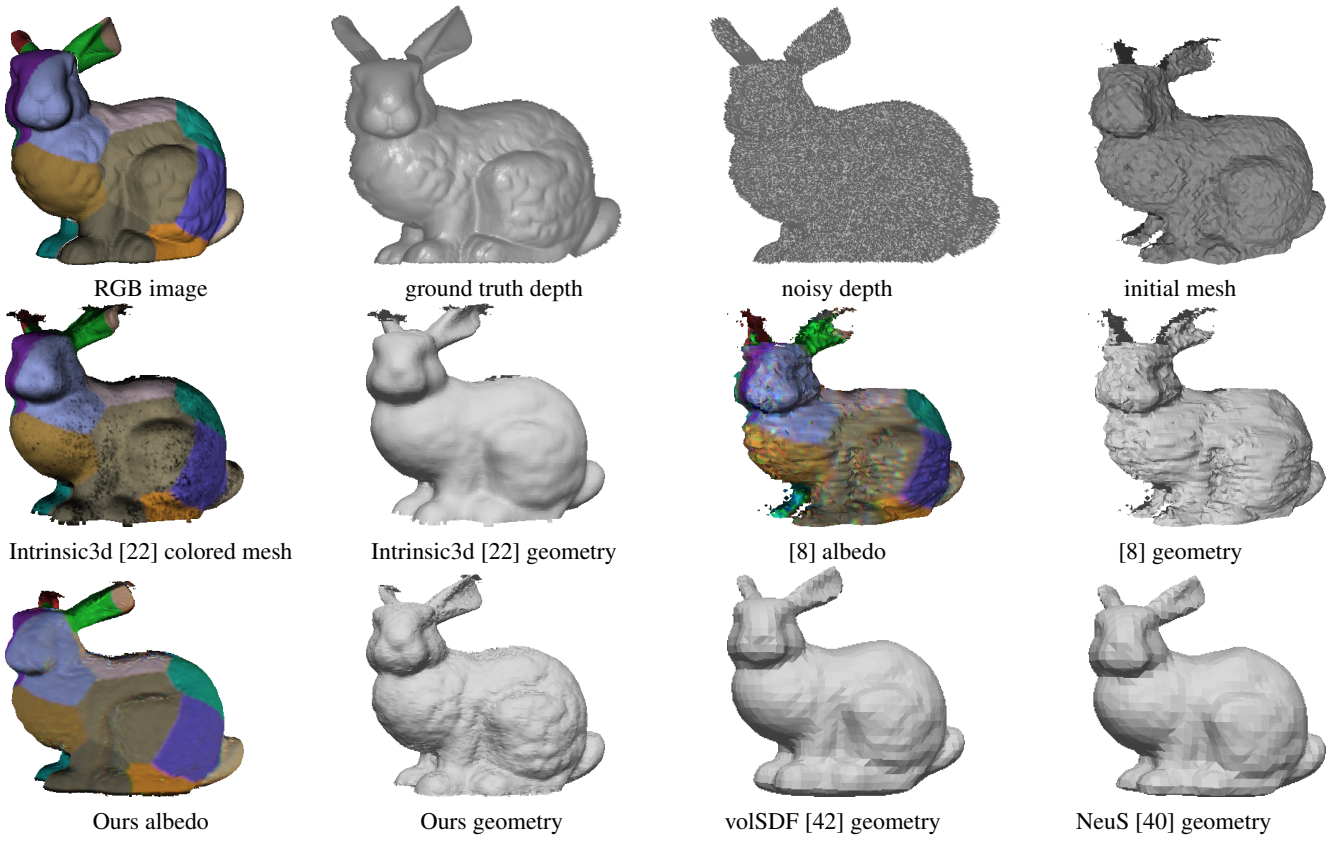

Figure S2. The rendered synthetic dataset bunny [33] with Kinect-like noise [16]. The initial mesh is the fused mesh after the camera tracking stage. The instrinsic3d [22] is over smoothed, and the mesh of [8] fails to deal with noisy depth. The work [42] and [40] does not recover albedo and the details are still missing.

3.2. Real-World Datasets

To measure the refined surface accuracy of our method quantitatively, we compare our results with a ground truth laser scan of the Socrates and Vase Multi-View Stereo dataset [46], these datasets offer the color images, and corresponding depth images, together with the pre-estimated camera poses. The depth images have been masked. All the outside areas are set to zero. In the error map, the green color indicates the minor error, from the yellow to red color transaction indicates a larger positive distance to the ground truth, and the blue direction indicates the negative distance to the ground truth. Figure S3 shows the comparison results of the related methods with our method. For the neural-rendering-based methods, volSDF [42] can only recover the blurry shape while the NeuS [40] fails to give any reconstruction surface, even though masks are provided. For classical methods, the work [8] suffers from the noisy depth, while intrinsic3d [22] can recover a good 3D model. However, the proposed method can recover the surface's fine scaled detail and achieve the smallest standard deviation error.

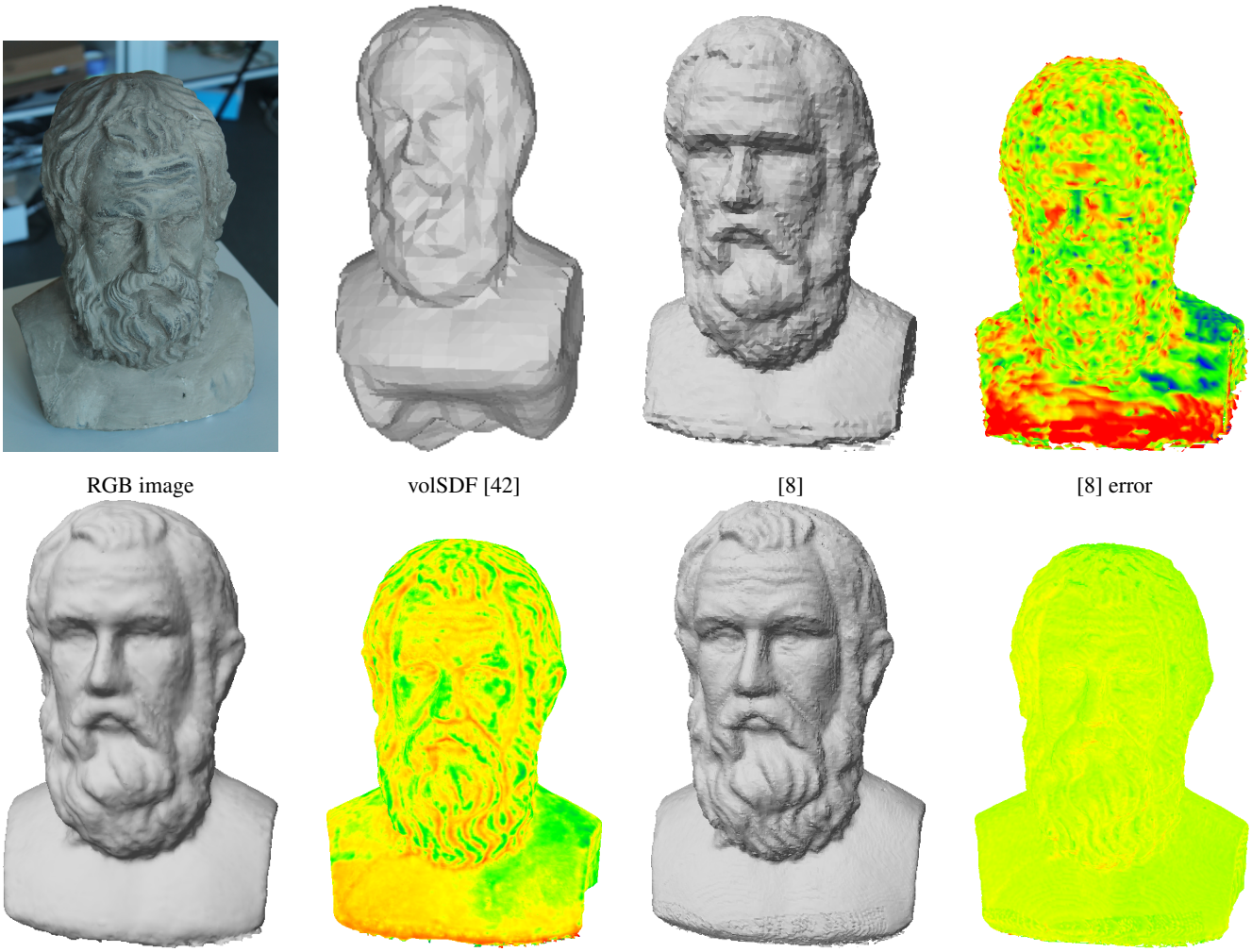

Figure S3. Comparison results with the ground truth laser model of multi-view dataset Sokrates [46]. While NeuS [40] fails to give any mesh, volSDF [42] gives only coarse estimated surface, the proposed method achieves error (standard deviation) of 1.2mm and Intrinsic3D [22] and [8] have error 2.1mm and 3.7mm respectively.

4. Visualization of Multi-view and Recorded Dataset

In this section, we demonstrate on Figure S4 more visualization results of the 3D scene dataset [34] and our own recorded dataset using the setup shown in Figure 3 in the paper. Both datasets have no camera poses offered.

Figure S4. The first two rows are the results of Lion dataset [22] and figure dataset [46], we track 200 frames and take 20 keyframes. After the refinement, the texture is more clear and the geometry detail is recovered. The last two rows are the recorded datasets. The refined mesh recovered clear recognizable pattern and the original color of the object.

5. Related Mathematics

List of Mathematical Symbols Here we list the mathematical symbols we used in the paper for a reference.

Symbol	Description	Symbol	Description
$\mathbf{I}_i$	i-th color image	$\mathbf{l}^s$	vector from point light source location to point $\mathbf{x}$
$\mathbf{p}$	continuous 2D image point in $\mathbb{R}^2$	$\mathbb{S}^2$	2D sphere
$\mathbf{x}$	continuous point in $\mathbb{R}^3$	R_i	rotation matrix of frame i
$\mathbf{p}(\mathbf{x})$	image points projected by space point $\mathbf{x}$	$\mathbf{t}_i$	translation vector of frame i
$\boldsymbol{\rho}(\mathbf{x})$	reflectance (albedo) of point $\mathbf{x}$	$\mathbf{o}$	origin of camera coordinates
$\mathbf{L}(\mathbf{i},\mathbf{x})$	point $\mathbf{x}$'s incoming radiance of direction $\mathbf{i}$	$d_{\mathcal{S}}(\mathbf{x})$	minimum distance from point $\mathbf{x}$ to the surface $\mathcal{S}$
$\mathbf{v}^j$	position of voxel j in $\mathbb{R}^3$	ψ^j	voxel j distance to the closest surface point
$\mathbf{g}^j$	voxel j gradient	$\mathbf{n}(\mathbf{x})$	surface normal at point $\mathbf{x}$
$\max(\cdot,\cdot)$	max operator, take the larger one	μ_s	anisotropy coefficient of point light source s
$\mathrm{SH}(\cdot)$	spherical harmonics function	∇	gradient operator
Ψ^s	point light source light intensity	l_i	lighting coefficient in SH model at i-th frame
$\mathcal{V}$	volume of the voxel grid	$\mathcal{I}$	input image set
ν_i^j	visibility indicator of voxel j at image i	$\mathcal{M}(\cdot,\cdot)$	image formation model

Table S3. Summary of used mathematical symbols.

6. Mathematical Details of Optimization

6.1. Image Formation Model

In this section, we illustrate the optimization details of the proposed method. The energy function is as described in equation (11). The optimization with robust estimator Φ is performed using the re-weighted least square method. In this section, we explain two proposed models in detail.

SH model From (5) and (6), note that $\mathbf{x}_i$ is the point $\mathbf{x}$ under i-th image coordinates. We use the fact that first order SH model is linear, so $\langle \mathbf{l}_i, \mathrm{SH}(R_i^\top \mathbf{x}^j)\rangle = \langle R_i \mathbf{l}_i, \mathrm{SH}(\mathbf{n}(\mathbf{x}^j))\rangle$. The total energy for SH model is

$$\begin{aligned}\min_{\{R_i,\mathbf{t}_i,\mathbf{l}_i\}_i,\{\boldsymbol{\rho}^j,\psi^j\}_j} \mathbf{E}(\boldsymbol{\rho}^j,\psi^j,R_i,\mathbf{t}_i,\mathbf{l}_i) = \sum_{i,j} \nu_i^j \Phi(\mathbf{I}_i(\pi(R_i^\top(\mathbf{x}^j-\mathbf{t}_i))) - \boldsymbol{\rho}^j \langle R_i\mathbf{l}_i, \mathrm{SH}(\mathbf{n}(\mathbf{x}^j))\rangle) \\ + |\left\|\nabla\psi^j\right\| - 1|^2 \,. \end{aligned} \tag{S1}$$

We directly optimize for $\hat{\mathbf{l}}_i = R_i\mathbf{l}_i$ instead in the real experiment.

PLS model From (7), the total energy for PLS model is

$$\begin{aligned}&\min_{\{R_i,\mathbf{t}_i,\Psi_i^s\}_i,\{\boldsymbol{\rho}^j,\psi^j\}_j} \mathbf{E}(\boldsymbol{\rho}^j,\psi^j,R_i,\mathbf{t}_i,\Psi_i^s) \\ &\quad= \sum_{i,j} \nu_i^j \Phi(\mathbf{I}_i(\pi(R_i^\top(\mathbf{x}^j-\mathbf{t}_i))) - \Psi_i^s \boldsymbol{\rho}^j \frac{\max(\langle R_i^\top \mathbf{n}(\mathbf{x}^j), -R_i^\top(\mathbf{x}^j-\mathbf{t}_i)\rangle, 0)}{\left\|R_i^\top(\mathbf{x}^j-\mathbf{t}_i)\right\|^3}) \\ &\quad+ |\left\|\nabla\psi^j\right\| - 1|^2 \,. \end{aligned} \tag{S2}$$

Optimization Let $\mathbf{r}_i^j = \mathbf{I}_i^j - \boldsymbol{\rho}^j \mathcal{M}(\mathbf{x}^j, \mathcal{X}_i)$ and $\mathbf{r}_e^j = \left\|\nabla\psi^j\right\| - 1$. For the kth iteration, we optimize

$$\min \sum_{i,j} w(\mathbf{r}_i^{j,(k)})(\mathbf{r}_i^j)^2 + (\mathbf{r}_e^j)^2 = 0 \,, \tag{S3}$$

where the weight function is defined as $w(\mathbf{r}) = \frac{\Phi(\mathbf{r})}{\mathbf{r}}$, and computed using current residuals.

Camera pose The camera pose is updated using 6DoF presentation. For $\omega_i \in SE(3)$, the rotation matrix R_i is updated by

$$R^{(k+1)} = R^{(k)} \exp(-\omega_i) \,. \tag{S4}$$

Written (S3) in vector form, we solve the linear system

$$(\mathbf{J}_c^\top W \mathbf{J}_c + \lambda_d I)\exp(\omega_i) = \mathbf{J}_c^\top W \mathbf{J}_c \tag{S5}$$

to solve the increment of the rotation matrix, $\mathbf{J}_c$ is the Jacobian matrix of R_i, and λ_d is the damping parameters and I is the diagonal of the $\mathbf{J}_c^\top W \mathbf{J}_c$. For SH model, $\mathbf{J}_c$ calculated as

$$\mathbf{J}_c = \frac{d\mathbf{r}}{d\omega_i} = [\nabla_x I, \nabla_y I]\left.\frac{d\pi}{d\mathbf{q}}\right|_{\mathbf{q}=R_i^\top(\mathbf{v}^j - \mathbf{g}_j\psi_j - \mathbf{t}_i)} \left[R_i^\top(\mathbf{v}^j - \mathbf{g}^j\psi^j - \mathbf{t}_i)\right]_\times , \tag{S6}$$

and for near field light is

$$\mathbf{J}_c = \frac{d\mathbf{r}}{d\omega_i} = [\nabla_x I, \nabla_y I]\left.\frac{d\pi}{d\mathbf{q}}\right|_{\mathbf{q}=R_i^\top(\mathbf{v}^j - \mathbf{g}^j\psi^j - \mathbf{t}_i)} \left[R_i^\top(\mathbf{v}^j - \mathbf{g}^j\psi^j - \mathbf{t}_i)\right]_\times - \left[\boldsymbol{\rho}^j \mathcal{M}_{\mathbf{v}^j, R_i, \mathbf{t}_i}()\right]_\times . \tag{S7}$$

Here π is the projection operator and $(\cdot)_\times$ is the twist matrix of the vector.

The translation vector is updated by $\mathbf{t}_i^{(k+1)} = \mathbf{t}_i^{(k)} - \Delta\mathbf{t}_i$, $\nabla\mathbf{t}_i$ is solved using the same step as $\exp(\omega_i)$. The Jacobian of $\mathbf{t}_i$ for LED model is

$$\frac{d\mathbf{r}}{d\mathbf{t}_i} = [\nabla_x I, \nabla_y I]\left.\frac{d\pi}{d\mathbf{q}}\right|_{\mathbf{q}=R_i^\top(\mathbf{v}^j - \mathbf{g}^j\psi^j - \mathbf{t}_i)} R_i^\top , \tag{S8}$$

and for the near field light the Jacobian is calculated using lagged attenuation term, that is we fix $\left\|(\mathbf{x}_i^j)^{(k)}\right\|^3$ from last iteration and treat it as a constant, then

$$\frac{d\mathbf{r}}{d\mathbf{t}_i} = [\nabla_x I, \nabla_y I]\left.\frac{d\pi}{d\mathbf{q}}\right|_{\mathbf{q}=R_i^\top(\mathbf{v}_j - \mathbf{g}^j\psi^j - \mathbf{t}_i)} R_i^\top - \frac{\boldsymbol{\rho}^j}{\left\|\mathbf{x}_i^{j(k)}\right\|^3}\mathbf{g}_i^\top . \tag{S9}$$

Voxel Distance As mentioned in the paper, the surface normal $\mathbf{n}(\mathbf{x}^j)$ is approximated by normalized voxel gradient $\mathbf{g}^j$, thus it is the function of voxel distance ψ^j. In practice, the derivative of distance is calculated using (forward or backward) finite difference, the ψ_x^j, ψ_y^j, ψ_z^j is the neighbor voxel in x, y, z direction respectively.

$$\mathbf{n}(\mathbf{x}^j) = \frac{\nabla\mathbf{g}^j}{\|\nabla\mathbf{g}^j\|} = \frac{(\psi^j - \psi_x^j, \psi^j - \psi_y^j, \psi^j - \psi_z^j)}{\left\|(\psi^j - \psi_x^j, \psi^j - \psi_y^j, \psi^j - \psi_z^j)\right\|} . \tag{S10}$$

Then for the natural light model, the Jacobian of ψ^j is

$$\mathbf{J}_{\psi^j} = [\nabla_x I, \nabla_y I]\left.\frac{d\pi}{d\mathbf{q}}\right|_{\mathbf{q}=R_i^\top(\mathbf{v}^j - \mathbf{g}^j\psi^j - \mathbf{t}_i)} R_i^\top(\frac{d\mathbf{g}^j}{d\psi}\psi - \mathbf{g}^j) - \boldsymbol{\rho}^j\mathbf{l}_i^\top\frac{d\,\mathrm{SH}(\mathbf{g}^j)}{d\mathbf{g}^j}\frac{d\mathbf{g}^j}{d\psi^j} , \tag{S11}$$

and for the near field light model, for simplicity, like for camera translation update, we fix the attenuation term from last update, and use the fact that $\left\|\mathbf{g}^j\right\| \approx 1$, then the Jacobian $\mathbf{j}_{\psi^j}$ is

$$\mathbf{J}_{\psi^j} = [\nabla_x I, \nabla_y I]\left.\frac{d\pi}{d\mathbf{q}}\right|_{\mathbf{q}=R_i^\top(\mathbf{v}^j - \mathbf{g}^j\psi^j - \mathbf{t}_i)} R_i^\top(\frac{d\mathbf{g}^j}{d\psi^j}\psi^j - \mathbf{g}^j) + \frac{\boldsymbol{\rho}^j}{\left\|\mathbf{x}_i^{j,(k)}\right\|^3}\left[(\mathbf{v}^j - 2\psi^j\mathbf{g} - \mathbf{t}_i)^\top\frac{d\mathbf{g}^j}{d\psi^j} - 1\right] , \tag{S12}$$

Then the distance increment $\nabla\psi^j$ is also solved by solving $\mathbf{J}_{\psi^j}^\top W\mathbf{J}_{\psi^j}\nabla\psi^j = \mathbf{J}_{\psi^j}^\top W\mathbf{r}$.

Reflectance and lighting For $\boldsymbol{\rho}^j$ and lighting $\mathbf{l}_i$, the Jacobian matrices are also computed by simple $\frac{d\mathbf{r}}{d\boldsymbol{\rho}^j}$, and $\frac{d\mathbf{r}}{d\mathbf{l}_i}$, they are relative straight forward to compute compare to distance and camera rotations, so we omit the detail here.

6.2. Camera Tracking

We use the SDF tracking energy to optimize the camera pose [38, 9]. To find the rigid body motion R and $\mathbf{t}$ for incoming point cloud with points $\mathbf{x}^k$ to the global shape $\mathcal{S}$. The energy is

$$\min_{R,\mathbf{t}} E(R,\mathbf{t}) = \sum_k w^k d_{\mathcal{S}}(R\mathbf{x}^k + \mathbf{t})^2 \,, \tag{S13}$$

note that k is point index in the current depth. For $w^k = \max(\min(1 + \frac{d^k}{T}, 1), 0)$ and

$$|d_{\mathcal{S}}(\mathbf{x})| = \min_{\mathbf{x}^s \in \mathcal{S}} \|\mathbf{x} - \mathbf{x}^s\| \,. \tag{S14}$$

In our case

$$d_{\mathcal{S}}(\mathbf{x}) = \psi^{j*} + (\mathbf{x} - \mathbf{v}^{j*})^\top \mathbf{g}^{j*} \,, \tag{S15}$$

$$\nabla d_{\mathcal{S}}(\mathbf{x}) = \mathbf{g}^{j*} \,, \tag{S16}$$

$$j* = \arg\min_j \left\|\mathbf{x} - \mathbf{v}^j\right\| \,. \tag{S17}$$

After the R, t is optimized, the distance for current global shape $\mathcal{S}$ can be updated using current point cloud by weighted average from starting frame until the current frame

$$\psi = \sum_i \frac{w_i d_i}{w_i} \tag{S18}$$

$$d_i = (\mathbf{x}^{k*} - R_i^\top(\mathbf{v} - \mathbf{t}_i))_z \tag{S19}$$

$$k* = \arg\min_k \left\|\mathbf{x}^k - R_i^\top(\mathbf{v} - \mathbf{t}_i)\right\| \,. \tag{S20}$$

References

[33] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, page 311–318, New York, NY, USA, 1994. Association for Computing Machinery.

[34] Qian-Yi Zhou and Vladlen Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Trans. Graph.*, 33(4), jul 2014.