Simultaneous Localization and Calibration: Self-Calibration of Consumer Depth Cameras

Qian-Yi Zhou[*] Vladlen Koltun[†]

Abstract

We describe an approach for simultaneous localization and calibration of a stream of range images. Our approach jointly optimizes the camera trajectory and a calibration function that corrects the camera's unknown nonlinear distortion. Experiments with real-world benchmark data and synthetic data show that our approach increases the accuracy of camera trajectories and geometric models estimated from range video produced by consumer-grade cameras.

1. Introduction

Consumer-grade range cameras are known to introduce distortion into the computed range images [8, 3, 11]. This distortion can be viewed as a function $d : \mathbb{R}^3 \to \mathbb{R}^3$ that shifts point measurements in the camera's measuring volume. This function has a complicated irregular shape that is specific to each camera. It is more complex than linear, radial, and decentering distortions commonly encountered in color cameras [2] and is not addressed by traditional camera calibration techniques. In practice, range distortion can lead to significant artifacts in reconstructed three-dimensional models of real-world scenes [11, 14].

To date, two approaches to dealing with range distortion have emerged. The first is to apply dedicated calibration to each camera and estimate a distortion model that can be used to correct subsequent range images produced by the camera. For example, Smisek et al. [8] acquire range images of a planar calibration target at multiple distances, fit planes to the images, and use the residuals to estimate the distortion function. Herrera et al. [3] describe a similar approach. Teichman et al. [11] show that a planar calibration target may not be necessary and that a distortion function can be estimated by acquiring and processing a dedicated range video sequence of an unstructured environment, as long as the environment has sufficient distinctive keypoints. The technique involves imaging the same objects at different distances and using near-field measurements to estimate the distortion for the far field; near-field readings are treated as ground truth and are never calibrated.

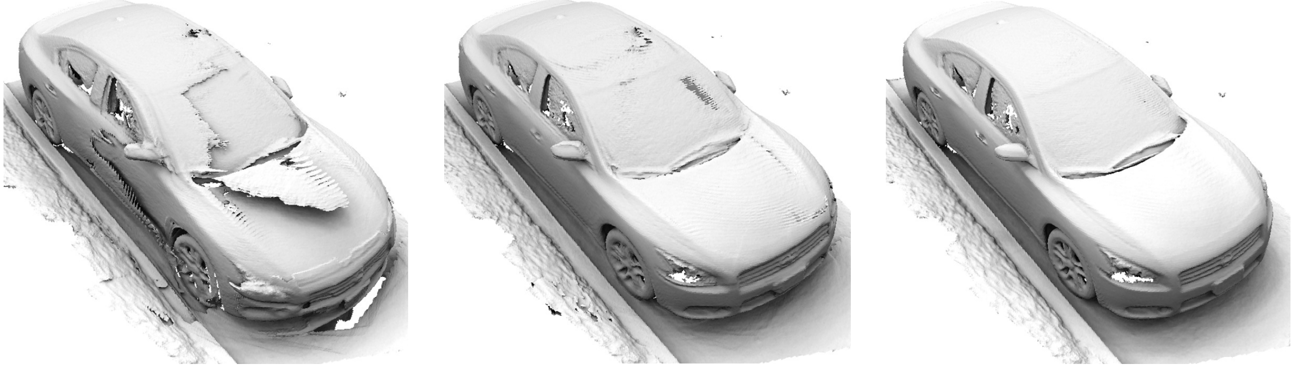
These calibration approaches all require using the camera to acquire a specialized video sequence. This prevents the application of these approaches to range data that has been acquired in the past by sensors that may not be readily available for calibration. It would also be difficult to apply these approaches to range cameras that have already been shipped, including over twenty million Kinect cameras that are currently in the homes of consumers.

The second approach to dealing with range distortion is to allow the acquired range data to deform nonrigidly when a scene model is reconstructed [14]. This approach does not estimate the distortion explicitly: it simply attempts to correct the unknown distortion by deforming the range images to each other. One disadvantage of this approach is that it interferes with reliable camera localization. Another disadvantage is that the nonrigid deformation can warp the reconstruction unnecessarily, since the technique does not explicitly distinguish true distortion correction from spurious warping.

In this paper, we introduce an approach to joint scene reconstruction and range camera calibration. Range distortion is estimated in tandem with camera localization and scene reconstruction. Our approach can be applied to legacy range video sequences and to uncalibrated cameras that had already been deployed. No separate calibration is needed, since it is performed *in situ* while the camera is being used to scan real-world objects and environments.

Our approach directly recovers a camera trajectory alongside the distortion model. This distinguishes it from reconstruction approaches that simply deform the input data without performing distortion estimation and camera localization [14]. Furthermore, factorizing the deformation into a rigid camera localization component and a static range distortion component allows us to substantially reduce the number of estimated parameters as compared to generic nonrigid deformation. This compact parameterization enables extremely efficient simultaneous localization and calibration (SLAC). As a result, our approach can reconstruct real-world scenes while estimating and correcting for range distortion in real time.

[*] Stanford University
[†] Adobe Research



(a) KinectFusion (b) Online SLAC (c) Offline SLAC

Figure 1. Reconstruction results for a scan acquired by looping around a car. (a) KinectFusion suffers from drift due to distortions in the data and is unable to cleanly close the loop. (b) The calibration performed by our online SLAC implementation reduces drift such that the loop is closed cleanly without explicit loop closure detection. (c) Our offline SLAC implementation includes explicit loop closure detection that further improves reconstruction quality.

Figure 1 shows a three-dimensional reconstruction produced by our approach. Online calibration performed by the system during scanning reduces drift and yields a qualitatively cleaner reconstruction. Quantitative results are provided in Section 4.

2. Simultaneous Localization and Calibration

In this section we describe the general SLAC framework. The input is a sequence $\{\mathcal{D}_i\}_{1\leq i\leq N}$ of depth images and a set $\{\mathcal{K}_{i,j}\}$ of correspondence sets. Each correspondence set $\mathcal{K}_{i,j}$ provides a set of correspondence pairs $\{\mathbf{p},\mathbf{q}\}$ such that $\mathbf{p}\in\mathcal{D}_i$ and $\mathbf{q}\in\mathcal{D}_j$. In other words, $\mathcal{K}_{i,j}$ is a set of pairwise correspondences between $\mathcal{D}_i$ and $\mathcal{D}_j$. In this section, we are assuming that the depth images $\{\mathcal{D}_i\}$ and the correspondences $\{\mathcal{K}_{i,j}\}$ are provided as input. Section 3 describes how to establish the correspondences $\{\mathcal{K}_{i,j}\}$ in an off-line setting in which the complete sequence $\{\mathcal{D}_i\}$ is provided and significant processing time is available, as well as an online setting in which the sequence $\{\mathcal{D}_i\}$ is streaming in at video rate and the correspondences $\{\mathcal{K}_{i,j}\}$ need to be established in real time.

Given $\{\mathcal{D}_i\}$ and $\{\mathcal{K}_{i,j}\}$, SLAC produces camera poses $\{\mathbf{T}_i\}$ and a calibration function $C(\cdot)$. The camera poses form a trajectory $\mathbf{T}=\{\mathbf{T}_i\}$. For every frame i, the pose $\mathbf{T}_i$ is a rigid transformation that maps the depth image $\mathcal{D}_i$ from its local coordinate frame to the global world frame, where $\mathcal{D}_i$ is aligned with other depth images from the input sequence. Throughout this paper, we will use homogeneous coordinates for three-dimensional points and transforms. Thus $\mathcal{D}_i\subset\mathbb{P}^3$ and $\mathbf{T}_i$ is a 4×4 transformation matrix. The calibration function is a mapping $C:\mathbb{P}^3\to\mathbb{P}^3$ that is applied in the local coordinate frame of each depth image $\mathcal{D}_i$ and corrects range distortion. The calibration function produced for the reconstruction shown in Figure 1(c) is visualized in Figure 2.

Since we are aiming to correct the static intrinsic distortion of the camera, the function $C(\cdot)$ is the same for all images $\mathcal{D}_i$. This substantially reduces the complexity of the optimization problem and enables a real-time implementation.

2.1. Objective

To compute $\mathbf{T}$ and C, we minimize an objective of the following form:

$$E(\mathbf{T},C)=E_a(\mathbf{T},C)+\lambda E_r(C). \tag{1}$$

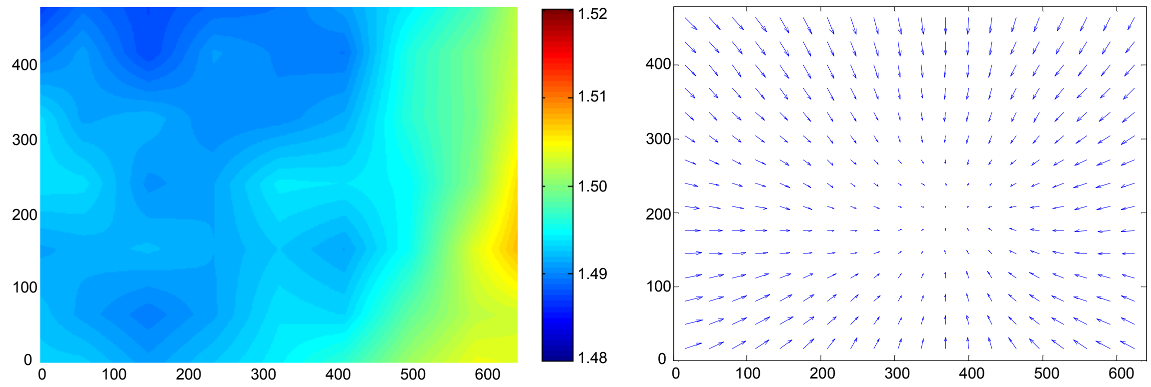
The alignment term $E_a(\mathbf{T},C)$ evaluates the alignment between depth images in the world frame. The regularization term $E_r(C)$ constrains the calibration function to be spatially smooth. The coefficient λ balances the strength of the two terms. Since the composition of the alignment term E_a grows with the number N of frames in the input sequence while the regularization term remains the same, we set $\lambda=N$.

The alignment term is based on the input correspondence sets $\{\mathcal{K}_{i,j}\}$. In essence, this term guides all corresponding pairs of points to align as closely as possible in the world frame. Alignment is measured in terms of the point-to-plane distance:

$$E_a(\mathbf{T},C)=\sum_{i,j}\sum_{(\mathbf{p},\mathbf{q})\in\mathcal{K}_{i,j}}\left((\mathbf{p}'-\mathbf{q}')^\top\mathbf{n}'_{\mathbf{p}}\right)^2. \tag{2}$$

Here the points $\mathbf{p}'$ and $\mathbf{q}'$ are points $\mathbf{p}$ and $\mathbf{q}$, transformed from their local coordinate frames to the world frame:

$$\begin{aligned}\mathbf{p}' &= \mathbf{T}_iC(\mathbf{p})\\ \mathbf{q}' &= \mathbf{T}_jC(\mathbf{q}).\end{aligned}$$



(a) Depth displacement

(b) Lateral displacement

Figure 2. The calibration function produced for the reconstruction shown in Figure 1(c). The calibration function is a displacement field over the camera's measuring volume. (a) visualizes the depth component of the displacement and (b) visualizes the lateral displacement. Each vector in (b) is scaled by a factor of 4. Both visualizations are for the slice of the measuring volume at 1.5m.

Note that the calibration function C is applied to each point in its local frame before the point is transformed by the appropriate camera pose to the world frame. The vector $\mathbf{n}'_{\mathbf{p}}$ is the normal of $\mathbf{p}'$ in the world frame.

To parameterize the function C, we take advantage of the spatial smoothness of range distortion [3, 11, 14]. Since the function is low-frequency, we can represent it at regularly spaced samples in $\mathbb{P}^3$ and interpolate the sampled values over the continuous space. We thus sample C over a regular lattice $\mathbf{V} = \{\mathbf{v}_l\} \subset \mathbb{P}^3$. The values of C over $\mathbf{V}$ are then interpolated for any point $\mathbf{p}$:

$$C(\mathbf{p}) = \sum_l \gamma_l(\mathbf{p}) C(\mathbf{v}_l), \tag{3}$$

where $\{\gamma_l(\mathbf{p})\}$ are trilinear interpolation coefficients. They are computed once for all input points and remain constant henceforth.

If our objective consisted of the alignment term by itself, nothing would prevent a degenerate solution in which the entire scene was collapsed to a single point in space. In this solution, the calibration function maps the entire measuring volume to a point, and all camera poses are arranged so that these points all align in the world frame. The value of the alignment term is minimized to zero in this solution, so a regularizer of some form is clearly needed. We use a shape-preserving regularizer inspired by elasticity theory [10, 14]:

$$E_r(C) = \sum_{\mathbf{v}\in\mathbf{V}} \sum_{\mathbf{u}\in\mathcal{N}_\mathbf{v}} \|C(\mathbf{u}) - {}^{\mathbf{v}}\mathbf{R}_{C(\mathbf{v})}\mathbf{u}\|^2, \tag{4}$$

where $\mathcal{N}(\mathbf{v})$ is the set of neighbors of $\mathbf{v}$ in $\mathbf{V}$ and the transform ${}^{\mathbf{v}}\mathbf{R}_{C(\mathbf{v})} \in SE(3)$ is a local linearization of C at $\mathbf{v}$.

2.2. Optimization

We minimize $E(\mathbf{T}, C)$ using the Gauss-Newton method. Let $\mathbf{x}$ be the vector of variables that includes all the parameters of $\mathbf{T}$ and C. The calibration function C is parameterized by the calibrated position $C(\mathbf{v})$ of each lattice point $\mathbf{v}$. The parameterization of the trajectory $\mathbf{T}$ is described below.

In iteration 0 the variables are initialized with the vector $\mathbf{x}^0$ that includes the camera poses from an initial rigid alignment of the input images $\{\mathcal{D}_i\}$ and a stationary function C that maps all the lattice points to themselves. Section 3 will describe how the initial rigid alignment is obtained.

In each subsequent iteration $k+1$, for $k \geq 0$, we locally linearize $\mathbf{T}_i$ around $\mathbf{T}_i^k$. Specifically, we parameterize $\mathbf{T}_i$ by a 6-vector $\xi_i = (\alpha_i, \beta_i, \gamma_i, a_i, b_i, c_i)$ that represents an incremental transformation relative to $\mathbf{T}_i^k$. Here (a_i, b_i, c_i) is a translation vector, which we will denote by $\mathbf{t}_i$, and $(\alpha_i, \beta_i, \gamma_i)$ can be interpreted as angular velocity, denoted by ω_i. $\mathbf{T}_i$ is approximated by a linear function of ξ_i:

$$\mathbf{T}_i \approx \begin{pmatrix} 1 & -\gamma_i & \beta_i & a_i \\ \gamma_i & 1 & -\alpha_i & b_i \\ -\beta_i & \alpha_i & 1 & c_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{T}_i^k. \tag{5}$$

The full parameter vector is updated in iteration $k+1$ as follows:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}. \tag{6}$$

Here $\Delta\mathbf{x}$ is a vector that collates $\{\xi_i\}$ and $\{\Delta C(\mathbf{v})\}$, where $\Delta C(\mathbf{v})$ is simply an additive increment to the vector $C^k(\mathbf{v})$. The adjustment $\Delta\mathbf{x}$ is computed by solving the following linear system:

$$\mathbf{J}_\mathbf{r}^\top \mathbf{J}_\mathbf{r} \Delta\mathbf{x} = -\mathbf{J}_\mathbf{r}^\top \mathbf{r}. \tag{7}$$

Here $\mathbf{r} = \mathbf{r}(\mathbf{x})$ is the residual vector and $\mathbf{J}_\mathbf{r} = \mathbf{J}_\mathbf{r}(\mathbf{x})$ is its Jacobian, both evaluated at $\mathbf{x}^k$. To compute the Jacobian, we compute the derivatives of the residuals of the two terms in the objective: the alignment term and the regularization term. The partial derivatives of the regularization residual are straightforward. To simplify the computation of the alignment residual derivatives, we freeze the effect of the calibration function C on the normals $\mathbf{n_p}$ and the transforms ${}^{\mathbf{v}}\mathbf{R}_{C(\mathbf{v})}$ within each Gauss-Newton iteration. These normals and transforms are updated to reflect the adjusted calibration function C between iterations. Consider an edge $\mathbf{e} = (\mathbf{p}, \mathbf{q})$ and consider the alignment residual for this edge:

$$r_a^{\mathbf{e}} = \left(\mathbf{T}_i C(\mathbf{p}) - \mathbf{T}_j C(\mathbf{q})\right)^{\top} \mathbf{T}_i \mathbf{n_p}. \tag{8}$$

Using Rodrigues' formula and simplifying, we get

$$\begin{aligned} \frac{\partial r_a^{\mathbf{e}}}{\partial \omega_i} &= \mathbf{T}_j^k C^k(\mathbf{q}) \times \mathbf{T}_i^k \mathbf{n_p}, & \frac{\partial r_a^{\mathbf{e}}}{\partial \mathbf{t}_i} &= \mathbf{T}_i^k \mathbf{n_p}, \\ \frac{\partial r_a^{\mathbf{e}}}{\partial \omega_j} &= \mathbf{T}_i^k \mathbf{n_p} \times \mathbf{T}_j^k C^k(\mathbf{q}), & \frac{\partial r_a^{\mathbf{e}}}{\partial \mathbf{t}_j} &= -\mathbf{T}_i^k \mathbf{n_p}, \\ \frac{\partial r_a^{\mathbf{e}}}{\partial \Delta C(\mathbf{v}_l)} &= \left(\gamma_l(\mathbf{p}) \mathbf{T}_i^k - \gamma_l(\mathbf{q}) \mathbf{T}_j^k\right)^{\top} \mathbf{T}_i^k \mathbf{n_p}. \end{aligned}$$

In iteration $k+1$, we evaluate $\mathbf{r}$ and $\mathbf{J}$ at $\mathbf{x}^k$ and solve the linear system (7) using sparse Cholesky factorization. The parameter vector $\mathbf{x}$ is updated according to equation (6). In particular, each camera pose $\mathbf{T}_i$ is updated by applying the incremental transformation ξ_i to $\mathbf{T}_i^k$ using equation (5) and then mapped back into the $SE(3)$ group. In the next iteration, we re-parameterize $\mathbf{T}_i$ around $\mathbf{T}_i^{k+1}$ and repeat.

3. Implementation

We have developed two implementations of the SLAC framework. As described in Section 2, the input to the framework is a set of depth images, a set of correspondences between depth images, and an initial camera trajectory used for initializing the optimization. Our implementations differ in how this input is provided. We begin by describing an offline implementation that incorporates loop closure constraints and then describe an online implementation that aims to reduce drift but does not perform explicit loop closure.

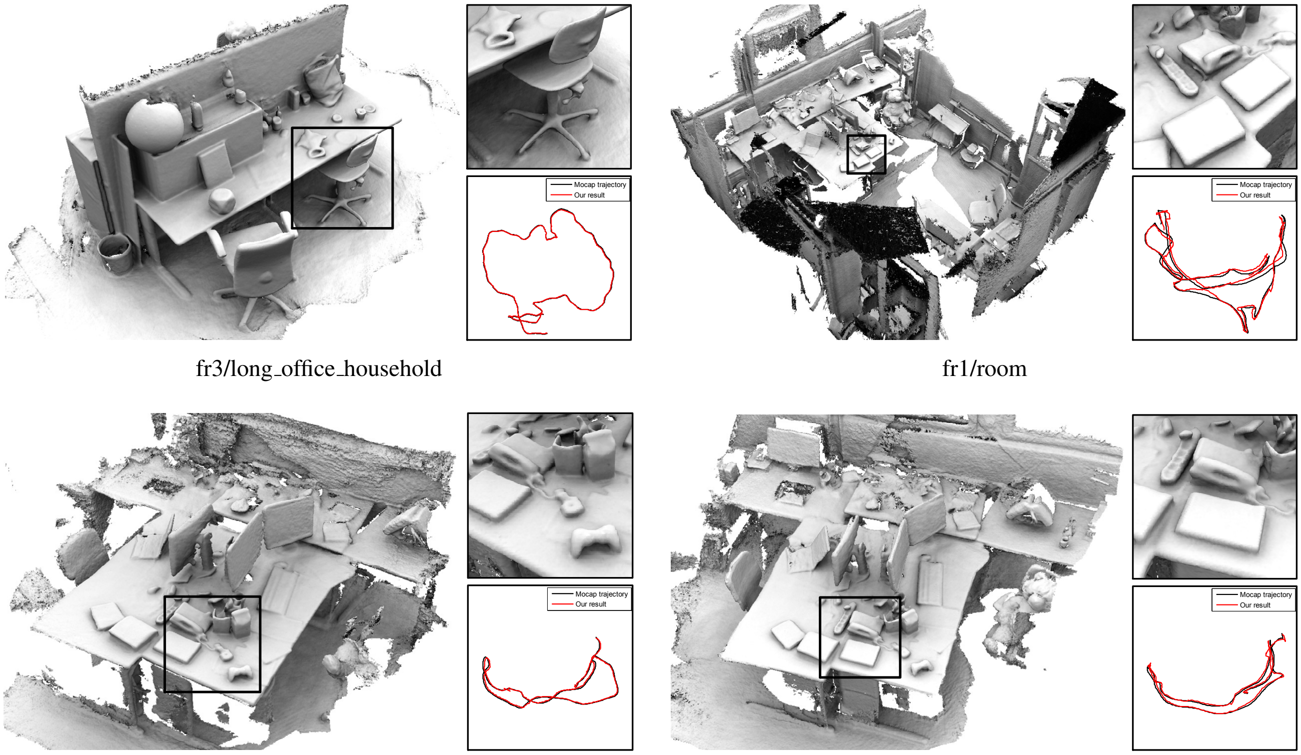
Offline system. Our offline implementation is based on the reconstruction system of Zhou et al. [14]. This system establishes dense correspondences throughout a sequence of range images using a loop closure module and computes an initial camera trajectory by performing rigid alignment. The range data is then deformed to satisfy the correspondence constraints.

We use the results of the first part of this pipeline, which provides dense correspondences and a rough initial camera trajectory. Then, instead of deforming the range data, we optimize the camera trajectory and the calibration function as described in Section 2. An experimental comparison of our approach with the approach of Zhou et al. is provided in Section 4. Our approach has a number of benefits compared to the prior approach. First, our method explicitly produces a camera pose for each individual frame, providing a camera trajectory that can be used in subsequent applications. Second, our optimization problem is considerably more compact and the optimization process is much faster.

In more detail, the objective formulated by Zhou et al. optimizes a separate control lattice for each scene fragment. Let m be the number of variables involved in specifying the state of a control lattice. For example, $m = 2187$ for an 8^3 lattice. The total number of variables optimized by Zhou et al. is km, where k is the number of scene fragments and is proportional to the length of the scan. For scans that are more than 10 minutes long, millions of variables are optimized, making the solver slow and memory intensive. In contrast, our method uses m variables for a single shared control lattice and 6 variables for a rigid body transformation optimized for each scene fragment: a total of $m + 6k$ variables. The number of variables is on the scale of thousands rather than millions. Our optimization thus takes minutes rather than hours. Moreover, this reduction in the scale of the optimization problem enables the use of a higher-resolution control lattice without incurring prohibitive computational penalties. Finally, by constraining the deformation to a single latent camera calibration function and thus reducing the scope for spurious warping of the data, our approach produces more metrically accurate reconstructions, as demonstrated in Section 4.

Online system. Our online implementation builds on the KinectFusion system of Newcombe et al. [5]. Our system is composed of a KinectFusion front-end and a SLAC back-end. The front-end maintains a TSDF volume $\mathbf{V}$, a camera pose trajectory $\mathbf{T}$, and a calibration model C. Once a depth image $\mathcal{D}_i$ comes in, the front-end first warps $\mathcal{D}_i$ using C and re-samples it as a depth image $\overline{\mathcal{D}}_i$. KinectFusion takes $\overline{\mathcal{D}}_i$ as input, registers it with $\mathbf{V}$ to get the camera pose $\mathbf{T}_i$. $\mathbf{T}_i$ is added into the camera pose trajectory and $\mathbf{T}_i\overline{\mathcal{D}}_i$ is integrated into $\mathbf{V}$. In order to provide data for the SLAC back-end, the front-end creates a new scene fragment every κ frames ($\kappa = 50$). A scene fragment is a three-dimensional model produced from the corresponding κ frames. To create these fragments, the front-end maintains a second TSDF volume $\mathbf{V}'$ that is used for integration of the short sequences of input frames that contribute to the current fragment. This second integration thread does not perform its own odometry, relying instead of the registration performed in the primary volume and reusing its results. The front-end extracts a point cloud from $\mathbf{V}'$ and resets this volume every κ frames.

The SLAC back-end is an independent thread running in parallel. It iterates through the optimization procedure to update the calibration model C. At the beginning of



fr3/long_office_household

fr1/room

fr1/desk

fr1/desk2

Figure 3. Scene reconstruction and trajectory estimation on benchmark data [9].

each iteration, it collects all the fragments $\widetilde{\mathcal{D}}_k$ extracted so far. To avoid additional computational overhead, correspondences are only established between temporally adjacent fragments. We skip ICP by directly computing relative transformations from $\mathbf{T}$ and using a standard kd-tree for nearest neighbor lookup. The back-end solves the optimization problem described in Section 2 and produces an updated calibration model that is immediately adopted by the front-end.

4. Results

Figure 1 shows reconstruction results for a two minute long scan collected with an Asus Xtion Pro Live camera that streams VGA-resolution range images at 30 fps. The operator scanned a car by looping around it. We reconstructed the car using the online KinectFusion system [5], our online SLAC implementation, and our offline SLAC implementation. The KinectFusion system is handicapped by distortion in the data and drifts to an extent that severely corrupts the reconstructed model when the loop is closed. In contrast, SLAC estimates a nonlinear calibration model for the camera and uses it to correct the distortion in the data. Consequently, camera drift is greatly reduced and the loop is closed without severe misalignments even without explicit loop closure.

Sequence	# of frames	SLAC	Zhou and Koltun	RGB-D SLAM	Extended KinFu
fr3/office	2,488	0.022	0.047	0.042	0.118
fr1/room	1,360	0.059	0.087	0.223	0.231
fr1/desk	591	0.022	0.026	0.034	0.059
fr1/desk2	631	0.035	0.037	0.061	0.048

Table 1. Evaluation of estimated camera pose trajectories on four sequences from the RGB-D SLAM benchmark. This table shows the RMSE (in meters) obtained on these sequences using SLAC, the approach of Zhou and Koltun [12], RGB-D SLAM [1], and Extended KinectFusion [7].

To evaluate our approach on legacy data, we tested the offline SLAC implementation on four scenes from the RGB-D SLAM benchmark [9]. The results are shown in Figure 3. Our approach creates globally consistent scenes with high-fidelity local details. Since the benchmark datasets provide ground-truth camera trajectories, we can quantitatively evaluate the trajectories produced by our approach and compare them to the results of alternative approaches using the absolute translational root mean square error (RMSE) measure [9]. As shown in Table 1, our approach produces the most accurate trajectory for all four sequences.

In addition, we applied offline SLAC to the sitting area

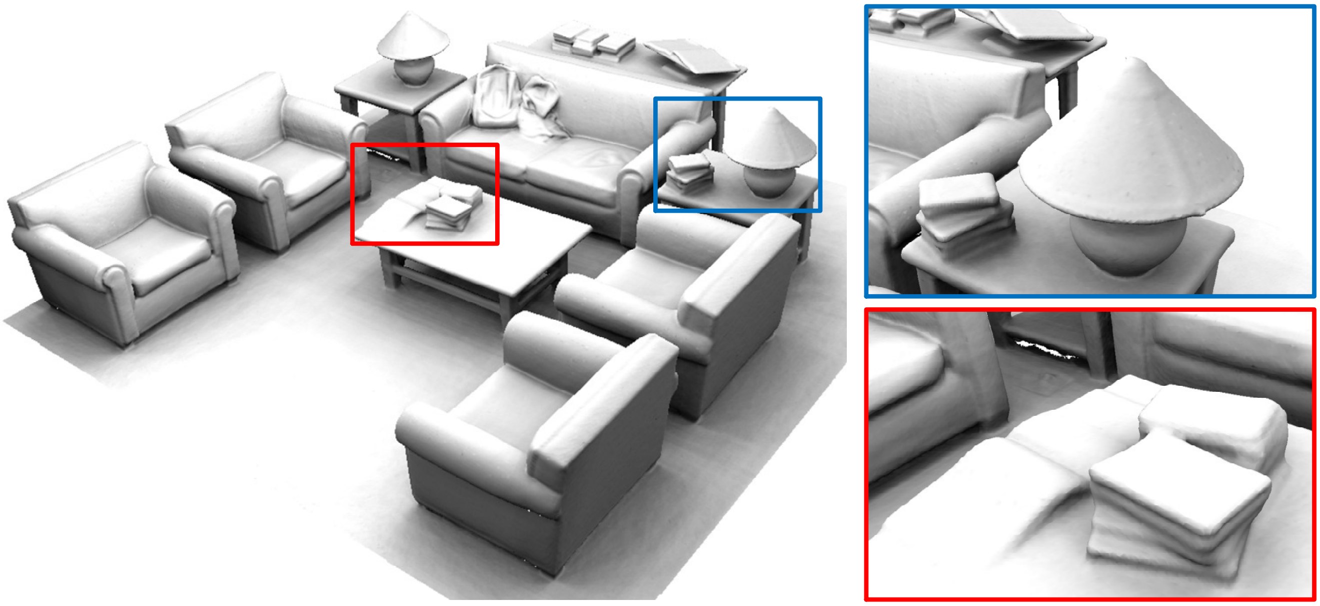


Figure 4. Reconstruction of the sitting area sequence from Zhou et al. [14].

sequence of Zhou et al. [14]. The reconstruction produced by SLAC is shown in Figure 4. It is qualitatively identical to the reconstruction produced by the elastic fragments approach of Zhou et al., which can be seen in their paper. However, the SLAC reconstruction is optimized much faster, in 19 minutes for an 8^3 control lattice and in 37 minutes for a 16^3 control lattice. This is in contrast to the elastic fragments optimization, which takes 7 hours and 30 minutes for an 8^3 lattice resolution. Running times were measured on a workstation with an Intel i7 3.2GHz CPU, 24GB of RAM, and an NVIDIA GeForce GTX 690 graphics card.

To evaluate the effect of calibration in a controlled setting, we performed experiments with the synthetic datasets of Zhou et al. [14]. These are rendered range videos of two synthetic sculptures, acquired along known camera trajectories. The sculptures are six meters tall. The synthesized range images are combined with an error model that simulates the data produced by real-world range cameras. Specifically, the error model combines standard disparity-based quantization [4], a high-frequency noise model [6], and a model of low-frequency distortion estimated on a real PrimeSense sensor [11]. We use these range videos as input to our offline SLAC implementation and test it with two resolutions for the estimated calibration function: 8^3, which is the resolution used by Zhou et al. for their control lattices, and 16^3, which is the default resolution for our system.

We compare the reconstructions produced by our approach to two alternatives: the first is the reconstruction produced by the elastic fragments approach, the second is the reconstruction produced by integrating the simulated data along the ground truth camera trajectory. The reconstructed models are shown in Figure 5. We can quantitatively analyze the accuracy of the different reconstructions by computing the cumulative distribution of point-to-plane distances from points on each reconstructed model to their nearest neighbors on the ground truth shape. These cumulative distributions are also shown in Figure 5. For both datasets, SLAC reconstructions are more accurate than the reconstructions produced by the elastic fragments approach, at either calibration resolution. We hypothesize that this is due to the uncontrolled nature of the deformation performed by the elastic fragments approach. Our optimization is considerably more constrained and produces metrically more accurate results. Furthermore, the results produced by SLAC with the higher calibration resolution are closer to the ground truth than the results of integrating the input data along the ground truth camera trajectory. This demonstrates that real-world camera distortion is sufficiently strong to corrupt reconstruction results even with perfect localization, and that our approach can ameliorate this problem.

5. Discussion

We presented an approach to simultaneous localization and calibration of range video. This approach can be viewed as a development of our elastic fragments pipeline [14]. Our new formulation factorizes the nonrigid deformation into a rigid camera localization component and a latent nonrigid calibration component. This factorization drastically reduces the problem size and enables significant acceleration, reducing optimization times from hours to minutes. Unlike the elastic fragments approach and other nonrigid registration approaches, SLAC produces an optimized camera

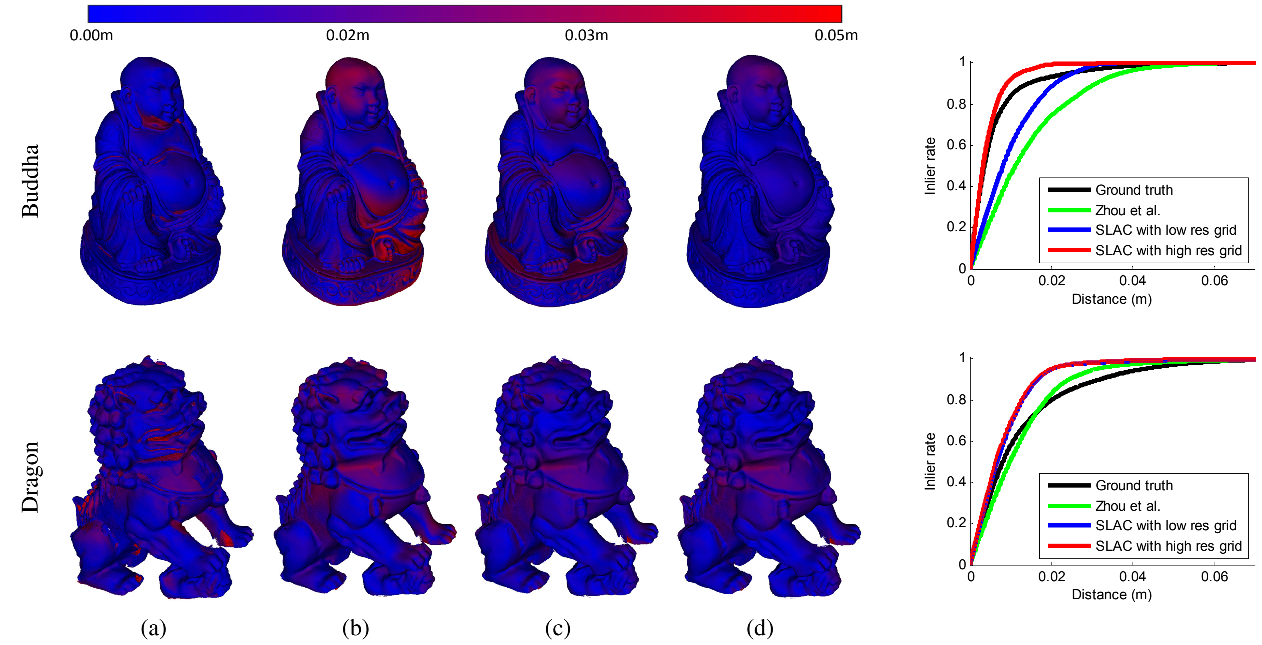


Figure 5. Experiments with synthetic data: (a) volumetric integration along the ground-truth camera trajectory, (b) the approach of Zhou et al. [14], (c) SLAC with an 8^3 control lattice, and (d) SLAC with a 16^3 control lattice. The graphs on the right show the cumulative distributions of distances from the reconstructed models to the ground-truth surface. Our approach is more accurate than the approach of Zhou et al., at either lattice resolution. Our approach also outperforms integration along the ground-truth camera trajectory. See text for analysis.

trajectory that can be used in subsequent processing stages such as color mapping [13].

The SLAC formulation enables an online implementation and we have demonstrated results produced by such an implementation. Our current online implementation is quite limited. First, although camera poses are optimized in tandem with the calibration function, the optimized camera poses are not used by the current online system: only the optimized calibration function affects the generated model in real time. Second, the optimized calibration function is not applied retroactively: it does not affect previously integrated frames. Producing a more advanced online implementation is one task for future work.

References

[1] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *ICRA*, 2012.

[2] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[3] D. Herrera C., J. Kannala, and J. Heikkilä. Joint depth and color camera calibration with distortion correction. *PAMI*, 34(10), 2012.

[4] K. Konolige and P. Mihelich. *Technical description of Kinect calibration*. 2012. http://wiki.ros.org/kinect_calibration/technical.

[5] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[6] C. V. Nguyen, S. Izadi, and D. Lovell. Modeling Kinect sensor noise for improved 3D reconstruction and tracking. In *3DIMPVT*, 2012.

[7] H. Roth and M. Vona. Moving volume KinectFusion. In *BMVC*, 2012.

[8] J. Smisek, M. Jancosek, and T. Pajdla. 3D with Kinect. In *ICCV Workshops*, 2011.

[9] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IROS*, 2012.

[10] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3), 2007.

[11] A. Teichman, S. Miller, and S. Thrun. Unsupervised intrinsic calibration of depth sensors via SLAM. In *RSS*, 2013.

[12] Q.-Y. Zhou and V. Koltun. Dense scene reconstruction with points of interest. *ACM Trans. Graph.*, 32(4), 2013.

[13] Q.-Y. Zhou and V. Koltun. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Trans. Graph.*, 33(4), 2014.

[14] Q.-Y. Zhou, S. Miller, and V. Koltun. Elastic fragments for dense scene reconstruction. In *ICCV*, 2013.