Comprehensive Use of Curvature For Robust And Accurate Online Surface Reconstruction

Damien Lefloch, Markus Kluge, Hamed Sarbolandi, Tim Weyrich, and Andreas Kolb

Abstract—Interactive real-time scene acquisition from hand-held depth cameras has recently developed much momentum, enabling applications in ad-hoc object acquisition, augmented reality and other fields. A key challenge to online reconstruction remains error accumulation in the reconstructed camera trajectory, due to drift-inducing instabilities in the range scan alignments of the underlying iterative-closest-point (ICP) algorithm. Various strategies have been proposed to mitigate that drift, including SIFT-based pre-alignment, color-based weighting of ICP pairs, stronger weighting of edge features, and so on. In our work, we focus on surface curvature as a feature that is detectable on range scans alone and hence does not depend on accurate multi-sensor alignment. In contrast to previous work that took curvature into consideration, however, we treat curvature as an independent quantity that we consistently incorporate into every stage of the real-time reconstruction pipeline, including densely curvature-weighted ICP, range image fusion, local surface reconstruction, and rendering. Using multiple benchmark sequences, and in direct comparison to other state-of-the-art online acquisition systems, we show that our approach significantly reduces drift, both when analyzing individual pipeline stages in isolation, as well as seen across the online reconstruction pipeline as a whole.

Index Terms—3D Reconstruction, Curvature, Depth Fusion, Camera Tracking, Differential Geometry.

1 Introduction

INTERACTIVE real-time scene acquisition from hand-held depth cameras has recently developed much momentum [1], [2], enabling applications in ad-hoc object acquisition, augmented reality and other fields. Despite various improvements in real-time performance [3], scalability [4], [5], [6], [7], [8] and treatment of dynamic content [6], a key challenge to online reconstruction remains instabilities in the reconstructed camera trajectory due to imprecision in the range scan alignments of the underlying iterative-closest-point (ICP) algorithm [9], [10], which are particularly severe where the acquired object misses sufficiently salient geometric features to latch on to [11]. These errors accumulate over time, leading to distortions across larger scales in the final reconstruction.

While such drift may partially be mitigated through global offline relaxation [1], [12], [13], the need for a global post-process defeats many of the benefits of an online acquisition system. Various strategies have hence been proposed to minimize registration error already during the online registration stage, including color feature-based pre-alignment [14], color-based weighting of ICP pairs [15], [16], stronger weighting of edge features [17], and so on.

Our work, too, aims at minimization of registration error, by focusing on surface curvature as a reliable feature that is detectable on range scans alone and hence does not depend on accurate multi-sensor alignment. Unlike previous work that took curvature or related measures into consideration, however, we treat curvature as an independent quantity that we consistently incorporate into every stage of the real-time reconstruction pipeline, including densely curvature-weighted ICP, range image fusion, local surface reconstruction, and rendering, while maintaining real-time rates even for very large scenes.

Conceptually and technically our approach comprises the following features and contributions:

- we present the first online reconstruction design to systematically incorporate curvature as an independent surface attribute into the end-to-end reconstruction pipeline; key innovations are:
- an ICP variant that considers curvature for both dense correspondence finding and weighting for increased stability,
- a method to efficiently blend curvatures in the fusion stage,

and, with respect to the underlying point-based fusion framework [6] that we extend,

- fast and high-quality, curvature-aware local surface reconstruction directly from an *index map* [6],
- extension of the index-map approach to mitigate the impact of point collisions and to significantly speed up operations on the model point cloud.

In addition, we present a new benchmark data set that provides ground truth camera poses and geometry using Kinect and Kinect 2 cameras.

Using multiple benchmark sequences, and in direct comparison to other state-of-the-art online acquisition systems, we show that our approach significantly reduces drift, both when analyzing individual pipeline stages in

- *T. Weyrich is with the Department of Computer Science at University College London, Gower Street, WC1 6BT, London, United Kingdom. E-mail: t.weyrich@cs.ucl.ac.uk*
- *All others are with the Computer Graphics Group at University of Siegen, Hoelderlinstrasse 3, Siegen, 57076, Germany. E-mail: (given name).(surname)@uni-siegen.de*

isolation, as well as seen across the reconstruction pipeline as a whole. All data sets, camera poses, ground truth geometries, and reconstructions of our method are provided under http://www.cg.informatik.uni-siegen.de/3d-reconstruction/low-feature-benchmark.

2 Online Surface Reconstruction

Our design follows the established overall structure for online reconstruction systems shown in Figure 1. This general structure is equally shared by the first in-hand scanners [1], [3], Newcombe et al.'s KinectFusion [2], and various later improved systems for online 3D reconstruction from range images ([6], [8], [17], [18], amongst others), with differences in algorithmic details and in data representations underlying the individual pipeline stages. In this section, we will provide a brief overview over this system structure while motivating our design decisions in the context of previous work.

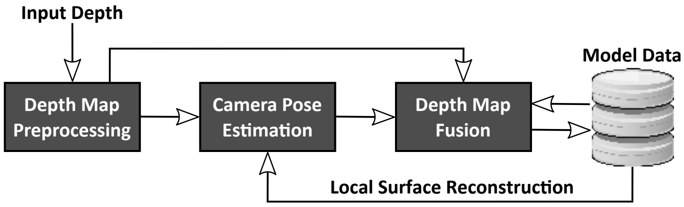


Fig. 1: Overview of 3D reconstruction framework.

Online 3D reconstruction typically assumes a hand-held sensor that captures a dense stream of depth images at video rates; some projector-based in-hand scanners move the object instead [1], [3], but the general principle remains the same. After initial *depth-map preprocessing*, the system has to *estimate camera pose* relative to previously acquired geometry (*local surface reconstruction*), before incorporating the newly acquired range image in a *depth map fusion* stage. The whole system is designed to operate in real time, including immediate visual feedback (*rendering*) of the partial reconstruction so far, creating a rapid feedback loop involving the user who guides the camera to gather geometry data where it is most needed.

Previous work showed continual improvement in accuracy, through algorithmic improvements but also through improved camera technology and processing speed. One problem, however, remains common to such systems: drift in the recovered camera trajectory, due to geometry-dependent instabilities in the camera pose estimation. Our work addresses this by systematically incorporating curvature as an additional surface attribute into the reconstruction pipeline.

Benefits of including curvature may not be immediately obvious, as one might argue that curvature was simply a function of surface shape, which is already being reconstructed. Also, derivatives of (noisy) real-world measurements are generally considered amplifying noise, which would render curvature a potentially unreliable quantity. As we will show, however, consistently incorporating curvature throughout the end-to-end reconstruction pipeline leads to significant reduction of drift. The remainder of this section outlines our respective extensions to the online reconstruction pipeline. Other enhancements orthogonal to our approach, such as incorporation of sensor uncertainty [19], [20], [21], [22], use of additional data sources beyond range images [15], [16], [23], [24], simplifying assumptions on structures in the scene [25], or non-rigid alignment [26], [27], could generally be of additional use but are outside the scope of this paper.

Depth Map Preprocessing Any depth reading is affected by noise, potential outliers, and subject to artifacts around depth discontinuities and where material properties affect the readings, requiring suitable pre-filtering of the input depth maps. In addition, most scanning pipelines estimate surface normals directly from the (filtered) input range image, on the grounds that extracting normal information later in the pipeline would be subject to ambiguous surface orientation [28], and that filtering in the camera domain is suited to reduce sensor-specific artifacts, leading to more robust normals for use in data association [2]. Depending on the model representation, there are further advantages to carrying out normal reconstruction only once and then explicitly storing it in the model representation, rather than having to repeatedly recompute it on the fly [6].

We find that all of these arguments equally apply to second-order surface properties, so we derive principal curvatures directly from the range image and store them alongside position and normal information. See Section 3 for more details.

Camera Pose Estimation is at the core of what makes hand-held online scanning possible. Incoming depth maps are continuously registered with the partial reconstruction of the object, the *model* acquired so far, to determine the camera's relative position to all previous observations. Any potential drift will occur at this stage, through inaccuracies in the depth map alignment. As we will show, however, improvements of the other pipeline stages indirectly reduce drift as well.

This depth map alignment generally makes use of the Iterative-Closest-Point (ICP) framework. In broad strokes, ICP consists of two stages: *data association*, where points on the incoming range image are paired with points on the model, and *minimization of alignment error*, as determined by an error metric that acts on the point pairs. (See [29] for a more detailed discussion and analysis of the design space for ICP implementations.)

Previous work has analyzed convergence rates and robustness of ICP, exploring alternative pairing strategies and error metrics [11], [12], [15], [29]. Godin et al. [15] introduce the closest-compatible point strategy that takes surface properties beyond simple point proximity into account during data association. They focus on surface color but stress generality of the approach; Pulli [12] demonstrates the benefits of considering compatibility of normals. Others report improved convergence when considering compatibility of image intensity and their gradients for pairing [16], [30].

In our work we show that naturally extending this strategy to incorporate compatibility of local curvature improves results even further. We facilitate this by maintaining in our model representation a continuously updated account of surface curvatures extracted from the input depth maps; see *local surface reconstruction* and *depth map fusion* for more detail.

Beyond this simple compatibility criterion, (implicitly) paying attention to high-curvature regions has proved valuable in previous work: Gelfand et al. [11] show that normal-space sampling [29], i.e., sub-sampling of the surface so that the corresponding normal directions are distributed as evenly as possible, creates point pairs that lead to a much-improved numerical condition of the ICP minimization.

While neither Gelfand et al. [11] nor Rusinkiewicz and Levoy [29] look at curvature itself, we observe that regions of higher curvature generally correspond to a larger spread of normal directions. We hence suggest that *putting more emphasis on high-curvature regions should similarly improve condition*. Also in contrast to normal-space sampling, which operates in the context of sparse point correspondences, we do not implement such emphasis through varying sampling density; instead, in the dense-ICP setting amenable to modern GPUs, we implement it as *weighting* within the error metric for alignment optimization.

Zhou and Koltun [17] recently presented a system that extracts contour cues from range images to stabilize alignment and thus reduces drift. While generally characterized by high principal curvature, these contours, however, are treated as a discrete feature that can be present or not, resulting in a bi-level weighting scheme. In contrast, our system uses continuous weights and is still able to exploit curved features that would not qualify as contour.

A purely feature-based approach has been presented by Johnson and Hebert [31], who compute local *spin images* across depth maps, using their signatures to match corresponding features in different depth maps. As any feature-based approach [32], this has merits if the geometry exhibits a sufficiently dense set of unique surface characteristics. In contrast, our method is based on ICP, which does not rely on unique local features and still converges even in the absence of high-curvature regions, as long as sufficient large-scale characteristics of the surface shapes exist.

Local Surface Reconstruction Data Association requires identifying individual (corresponding) points on the surface of the so-far accumulated model. While early works explored various strategies to construct correspondences between incoming and partially reconstructed model surfaces, the ICP community eventually identified "projective" pairing of incoming and model points as leading to far superior convergence times [29]. For each point on the incoming range map, this involves casting a ray along the depth sensor's lines of sight onto the model, and taking the intersection point as a candidate for pairing. Regardless of the underlying model representation, such ray-surface intersections require *local surface reconstruction* in the vicinity of that ray, e.g., ray-surface intersection if the model is explicitly represented as a triangle mesh, ray-casting of an (implicit) volumetric model representation [2], or some form of "point-based rendering" of surface information into the camera plane [1], [3], [6], [18]. In general, these operations borrow from surface rendering in computer graphics; however, the quantities being sampled (or rendered) depend on the attributes required for data association, typically comprising position, normal, and sometimes color.

In our work, we expand upon the local surface reconstruction by Keller et al. [6], which in its original formulation leads to a piecewise-linear local reconstruction, not unlike the approximations by other previous works [3]. In contrast, we consider full curvature information when determining local ray-surface intersections, and we will show how the resulting higher-quality surface reconstruction noticeably contributes to the overall drift reduction.

Depth Map Fusion While early online reconstruction systems display more or less raw input data during the online phase ([1], [3]), leaving data fusion into a single model to a post-process of global alignment [12] and volumetric fusion [28], Newcombe et al. [2] showed that a real-time implementation of Curless and Levoy's volumetric fusion approach [28] is possible.

These methods, however, require continual conversion between range-map and volumetric representations, and operate with a fixed spatial resolution. Keller et al. [6] present a purely point-based framework that allows for real-time fusion including adaptive resolution, without the need for frequent data conversion.

Our implementation follows the framework by Keller et al., as it allows for the most natural extension to support and analyze the use of curvature throughout the reconstruction pipeline. Similar to their depth map fusion that accumulates normal information independent from positional information, we introduce yet another independent information channel to accumulate curvature and present a method to efficiently blend curvature information during fusion (Sec. 6).

In order to maintain real-time rates, we extend Keller et al.'s intermediate, screen-space, index map representation, originally only used for data association: our *deep index map* supports incremental screen-space updates during fusion, enabling online rendering directly off that representation.

Rendering Many previous systems either offer lower-quality visual feedback, sufficient to guide the user toward regions where sensor data is missing [1], or perform comparatively expensive ray-casting on a volumetric representation [2], [8]. Weise et al. [3] use a local surface reconstruction approach for data association that is equally suitable for (point-based) rendering. We, too, simply capitalize on our high-quality local surface reconstruction approach that works directly on our internal model representation and can equally be used for high-fidelity, curvature-aware, rendering (Section 5). We further use a simple Phong illumination model coupled with a fast approximation of ambient occlusion known as Screen-Space Ambient Occlusion (SSAO) [33] for added realism in the visual feedback.

3 Depth Map Preprocessing

In the preprocessing stage, we extract point attribute maps from the range image data, following and extending conventions used by Newcombe et al. [2] and Keller et al. [6]. See Tab. 1 for a complete list of conventions used.

After outlier removal, depth map values $\mathcal{D}^t(\boldsymbol{u})$ are transformed into 3D positions in camera coordinates, using the inverse intrinsic camera matrix $\boldsymbol{K}^{-1}$, and stored in a *vertex map* $\mathcal{V}^t(\boldsymbol{u})$, with t the input frame index and pixel coordinates $\boldsymbol{u} = (x, y)^\top$ within the camera image. The *normal map* $\mathcal{N}^t$ is extracted from bilinearly filtered depths, a *point-radius map* $\mathcal{R}^t$ is obtained from local point neighborhood sizes, and, new in our system, a *curvature map* $\mathcal{K}^t$ is derived from $\mathcal{N}^t$ (see Section 3.1). Furthermore, following again previous work, we assign a confidence value $\mathcal{W}^t(\boldsymbol{u})$ for each input frame pixel $\boldsymbol{u}$. This value accounts for the radially decreasing quality of range image values (with fall-offs specific to each

Kinect model) and for the reduction of depth quality due to motion blur. The latter is estimated from the relative transformation $\boldsymbol{T}^{t\rightarrow(t-1)}$ between adjacent camera poses at times $t-1$ and t (see also Sec. 4.2).

3.1 Curvature Estimation

The curvature map encodes directions and values of principal curvature at each surface point: $\mathcal{K}^t = \{\hat{\boldsymbol{e}}_1, \kappa_1, \kappa_2\}$ stores the first principal direction $\hat{\boldsymbol{e}}_1$ and both curvature values κ_1, κ_2; the second principal direction $\hat{\boldsymbol{e}}_2$ is implicitly given as $\hat{\boldsymbol{e}}_2 = \hat{\boldsymbol{n}} \times \hat{\boldsymbol{e}}_1$.

While surface curvature is well defined on G^2-continuous surfaces, various competing approximations exist for discretized surface representations [34], [35], and many of them are applicable to point-sampled geometry. We tested several approaches for their robustness in our application scenario, i.e., on Kinect depth maps.

The eigen decomposition proposed by Pauly et al. [36] not only yields normal estimates, but also a notion of curvature. Even though a rather robust estimate, however, the approach is not scale-invariant if applied to projectively unevenly sampled geometry.

An alternative class of approximations performs a local surface fit and uses the curvature of the fitted surface as an estimate for the input vertex. Goldfeather and Interrante's adjacent-normal cubic approximation method [37], amongst the most robust curvature estimators in literature, uses a high-order surface polynomial fit (typically order 3) that takes into account the normal information of adjacent vertices in its formulation; the method, however, involves a larger (7×7) linear-least squares fit that, even when performed on the GPU, does not meet our real-time constraint.

We finally settled on the chord-and-normal-vectors (CAN) approach of Zhang et al. [38], which is comparatively robust also in the case of projectively unevenly sampled geometry and still computationally efficient. CAN estimation initially fits circles to the current oriented vertex $(\mathcal{V}^t(\boldsymbol{u}), \mathcal{N}^t(\boldsymbol{u}))$ and each oriented vertex $(\mathcal{V}^t(\boldsymbol{u}'), \mathcal{N}^t(\boldsymbol{u}'))$ in the selected neighborhood [38]. A principal curvature compatible with the fitted circles is then determined as an approximate, minimum least-squares solution that only requires solving a 3×3 linear system and is thus predestined for a GPU implementation. A more robust curvature estimator by Chen et al. [39] would still require solution of a 6×6 system, which for current GPU models would no longer be possible in real time.

$\hat{\boldsymbol{x}} \in \mathbb{S}^2$	normalized vector ($\hat{\boldsymbol{x}} = \boldsymbol{x} / \Vert\boldsymbol{x}\Vert$)
$t \in \mathbb{N}$	input frame ID
$i, j \in \mathbb{N}$	point indices
$l \in \mathbb{N}$	iteration index
$\boldsymbol{K} \in \mathbb{P}^3$	intrinsic camera matrix
$\boldsymbol{T}^{t\rightarrow(t-1)} \in \mathbb{SE}^3$	(rigid) relative transformation between consecutive camera poses
$\boldsymbol{u} = (x, y)^\top \in \mathbb{R}^2$	pixel position in input frame
$\mathcal{D}^t(\boldsymbol{u}) \in \mathbb{R}$	input depth map
$\mathcal{W}^t(\boldsymbol{u}) \in \mathbb{R}$	reliability map
$\mathcal{V}^t(\boldsymbol{u}) \in \mathbb{R}^3$	*vertex map* of object points corresponding to $\boldsymbol{u}$
$\mathcal{N}^t = \{\hat{\boldsymbol{n}}\}$	normal map (per-point attribute)
$\mathcal{R}^t \in \mathbb{R}$	point-radius map (per-point attribute)
$\mathcal{K}^t = \{\hat{\boldsymbol{e}}_1, \kappa_1, \kappa_2\}$	curvature map (per-point attribute)
$\boldsymbol{p}_i^t$	i-th input point in frame t
$\hat{\boldsymbol{n}} \in \mathbb{S}^2$	normal vector
$\kappa_1, \kappa_2 \in \mathbb{R}$	first and second principal curvature
$\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2 \in \mathbb{R}^3$	corresponding directions of curvature ($\langle\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2\rangle = 0$)
$\mathcal{I}^t$	index map

TABLE 1: List of Conventions

Fig. 2 compares the quality of Zhang et al. [38] with Goldfeather and Interrante [37] for both an input depth map and an accumulated model. It can be seen, that Zhang's method delivers stable results that are only slightly worse than Goldfeather and Interrante's, mainly for the second main curvature κ_2. (Note that under ideal conditions, we would expect $\kappa_2 = 0$ along straight edges and $\kappa_2 \neq 0$ for parabolic and saddle points.)

4 Camera Pose Estimation

Similar to the pose estimation proposed by Newcombe et al. [2], our camera pose estimation uses a hierarchical model-to-frame variant of the iterative-closest-point (ICP) registration [9] and is based on the point-to-plane error metric [12].

This common iterative framework alternates between *data association* (i.e., establishing correspondences between frame t's input points $\boldsymbol{p}_i^t = \mathcal{V}^t(\boldsymbol{u}_i)$ and corresponding points $\boldsymbol{p}_{\mathcal{M}}^* = \mathcal{V}_{\mathcal{M}}(\boldsymbol{u}^*)$ of the model acquired until frame $t-1$), and *minimization* of an error term $E(\boldsymbol{T}^{t\rightarrow(t-1)})$ that expresses the level of mismatch within point pairs under the estimated relative transformation $\boldsymbol{T}^{t\rightarrow(t-1)}$.

Our main enhancements of this framework are twofold: the correspondence-finding stage additionally considers curvature (on both the input map and the model); furthermore, we introduce a curvature-dependent weighting scheme into the error term $E(\boldsymbol{T}^{t\rightarrow(t-1)})$, which significantly increases the robustness of the convergence, and thus minimizes drift. In the following, we describe these extensions in detail.

4.1 Data Association

At the beginning of each iteration l, and given the latest estimate of the relative transformation $\boldsymbol{T}_{(l)}^{t\rightarrow(t-1)}$ with $T_{(0)}^{k\rightarrow(k-1)} := [\mathbf{I}_{3\times3}|\mathbf{0}]$, selection and matching are performed simultaneously, starting with the full set of input points.

Each input point $\boldsymbol{p}_i^t = \mathcal{V}^t(\boldsymbol{u}_i)$, including its geometric entities $\{\hat{\boldsymbol{n}}_i^t, \hat{\boldsymbol{e}}_{1,i}^t, \hat{\boldsymbol{e}}_{2,i}^t, \kappa_{1,i}^t, \kappa_{2,i}^t\}$, is transformed into the model reference $\boldsymbol{p}_j^{t-1} = \boldsymbol{T}_{(l)}^{t\rightarrow(t-1)}\boldsymbol{p}_i^t$ (and analogously for vectors $\hat{\boldsymbol{n}}_i^t, \hat{\boldsymbol{e}}_{1,i}^t$ and $\hat{\boldsymbol{e}}_{2,i}^t$). Then, we draw the set of neighboring model points $\mathcal{H}(\boldsymbol{p}_j^{t-1})$ from a 5×5 pixel window of a local surface reconstruction around the projection of $\boldsymbol{p}_j^{(t-1)}$ under $\boldsymbol{T}_{(l)}^{t\rightarrow(t-1)}$. Following general practice in point correspondence search [15], points in $\mathcal{H}(\boldsymbol{p}_j^{(t-1)})$ whose position and normal significantly differ from $\boldsymbol{p}_j^{(t-1)}$ are discarded. (More precisely, potential correspondences are rejected if $\Vert\boldsymbol{T}_{(l)}^{t\rightarrow(t-1)}\mathcal{V}^t(\boldsymbol{u}) - \mathcal{V}_{\mathcal{M}}(\boldsymbol{u}^*)\Vert \geq \theta_{\text{dist}}$ or $\angle(\boldsymbol{R}_{(l)}^{t\rightarrow(t-1)}\mathcal{N}^t(\boldsymbol{u}), \mathcal{N}_{\mathcal{M}}(\boldsymbol{u}^*)) \geq \theta_{\text{angle}}$, like Newcombe et al. [2].)

We now look for the model point $\boldsymbol{p}_{\mathcal{M}}^* \in \mathcal{H}(\boldsymbol{p}_j^{(t-1)})$ that best matches position, *Darboux frame* $\{\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2, \hat{\boldsymbol{n}}\}$, and the respective curvature values κ_1, κ_2, by minimizing a weighted sum of dissimilarity measures of positions (D_p), normals (D_n), and curvature (D_c),

$$\boldsymbol{p}_{\mathcal{M}}^* = \underset{\boldsymbol{p}_{\mathcal{M}} \in \mathcal{H}(\boldsymbol{p}_j^{(t-1)})}{\arg\min} \; \lambda_p D_p + \lambda_n D_n + \lambda_c D_c \,, \qquad (1)$$

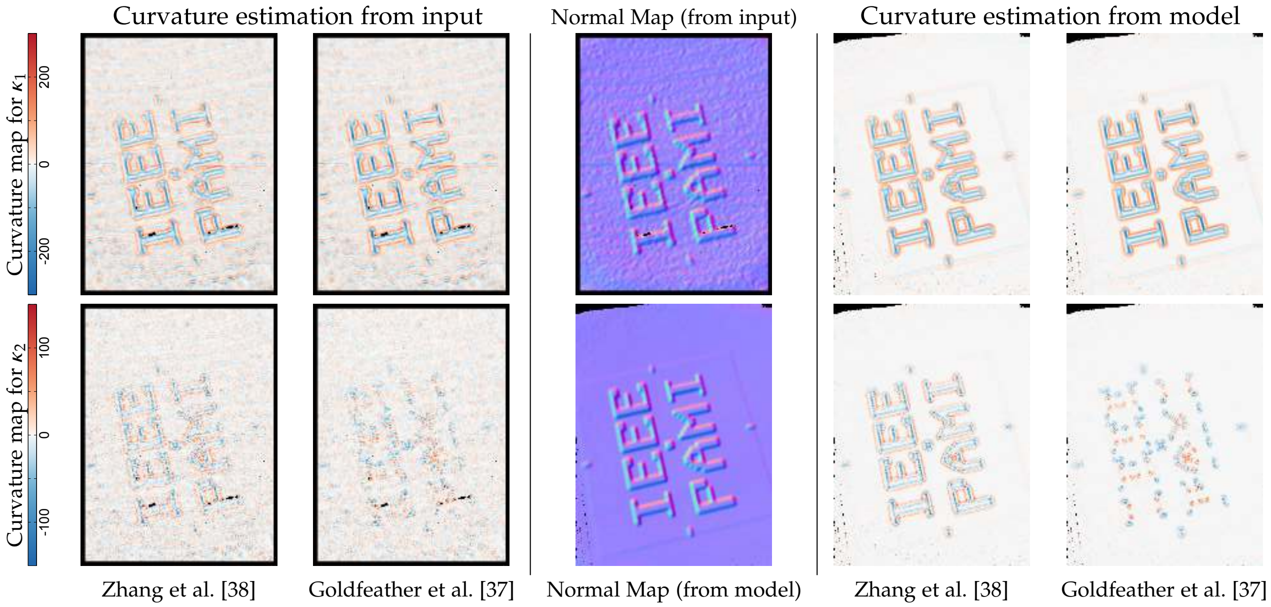


Fig. 2: Example curvature estimation for frame 361 of our **legoPAMI**$_{\text{SL}}$ data set, using the methods by Zhang et al. [38], and Goldfeather et al. [37], respectively. *Left:* using the input frame given by the range camera; *Right:* using the current reconstructed surface model. Curvature maps on the *top* and *bottom* show κ_1, and κ_2, respectively; corresponding input normal maps convey noise level.

with

$$D_p = \frac{\left\|\boldsymbol{p}_{\mathcal{M}} - \boldsymbol{p}_j^{(t-1)}\right\|_2}{R}, \quad D_n = 1 - \langle \hat{\boldsymbol{n}}_{\mathcal{M}}, \hat{\boldsymbol{n}}_j^{(t-1)} \rangle, \quad \text{and}$$

$$D_c = \begin{cases} \frac{\left|\kappa_{1,\mathcal{M}} - \kappa^t_{1,i}\right| + \left|\kappa_{2,\mathcal{M}} - \kappa^t_{2,i}\right|}{\kappa^{\max}_{\mathcal{M}}}, & \text{if } \left|\kappa^t_{1,i} - \kappa^t_{2,i}\right| < \theta_\kappa, \\ \frac{\left\|Q_{\mathcal{M}} - Q_j^{(t-1)}\right\|_2}{\kappa^{\max}_{\mathcal{M}}}, & \text{otherwise}, \end{cases} \tag{2}$$

where R is the maximum radius of the neighborhood search $\mathcal{H}(\boldsymbol{p}_j^{(t-1)})$, $\kappa^{\max}_{\mathcal{M}} = \max\{|\kappa_{1,\mathcal{M}}|, |\kappa_{2,\mathcal{M}}|\}$ and $\boldsymbol{Q} = \boldsymbol{C}\,\mathrm{diag}(\kappa_1, \kappa_2, 0)\,\boldsymbol{C}^\top$ with $\boldsymbol{C} = (\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2, \hat{\boldsymbol{n}})$ representing the transformation into the tangent space, scaled by the principal curvatures. The division of D_p and D_c, by R and $\kappa^{\max}_{\mathcal{M}}$, respectively, normalizes the components, leading to a relative error measure; all of our results use $\lambda_{\{p,n,c\}} = \frac{1}{3}$.

As the principal curvature directions are formally undefined for $\kappa_1 = \kappa_2$ (e.g., for planes or spheres) and numerically unstable for $\kappa_1 \approx \kappa_2$ (particularly in the presence of noise), D_c determines curvature dissimilarity independent from curvature directions when κ_1 and κ_2 are similar, i.e., within a threshold θ_κ; we used $\theta_\kappa = 15\ \mathrm{m}^{-1}$ for all experiments.

Furthermore, in the case of $\kappa_1 \neq \kappa_2$, the Darboux frame is unique up to inversion around $\hat{\boldsymbol{n}}$, thus we rotate $Q_j^{(t-1)}$ by π around $\hat{\boldsymbol{n}}$ if $\langle \hat{\boldsymbol{e}}_{1,\mathcal{M}}, \hat{\boldsymbol{e}}_{1,j}^{(t-1)} \rangle < 0$ to ensure compatible alignment before applying Eq. 2. We evaluate $\left\|Q_{\mathcal{M}} - Q_j^{(t-1)}\right\|_2$ via SVD, exploiting the matrix 2-norm equality $\|\boldsymbol{A}\|_2 = \sigma_{\max}(\boldsymbol{A})$.

4.2 Minimization

The point-to-plane error metric can be expressed as

$$E(\boldsymbol{T}^{t\to(t-1)}) = \sum_{\boldsymbol{u}\in\mathcal{S}} \langle \boldsymbol{T}_l^{t\to(t-1)} \mathcal{V}^t(\boldsymbol{u}) - \mathcal{V}_{\mathcal{M}}(\boldsymbol{u}^*), \mathcal{N}_{\mathcal{M}}(\boldsymbol{u}^*) \rangle^2, \tag{3}$$

with $\mathcal{S}$ the subset of all input map points for which a valid correspondence has been found, and with $\boldsymbol{u}^*$ being, again, each $\boldsymbol{p}^*_{\mathcal{M}}$'s projection into the previous frame. Some previous ICP works extend this least-squares minimization by a set of per-correspondence weights $w(\boldsymbol{u}^*)$, so that

$$E(\boldsymbol{T}^{t\to(t-1)}) = \sum_{\boldsymbol{u}\in\mathcal{S}} w(\boldsymbol{u}^*) \langle \boldsymbol{T}_l^{t\to(t-1)} \mathcal{V}^t(\boldsymbol{u}) - \mathcal{V}_{\mathcal{M}}(\boldsymbol{u}^*), \mathcal{N}_{\mathcal{M}}(\boldsymbol{u}^*) \rangle^2. \tag{4}$$

Zhou and Koltun [17] choose $w(\boldsymbol{u}^*)$ depending on whether a point is a contour point. In contrast, instead of using bi-level weights only, our $w(\boldsymbol{u}^*)$ continuously depends on the curvature information, thus leading to an adaptive, curvature-related weight.

We propose the following curvature weight scheme which is based on the maximum absolute principal curvature $\kappa^{\max}_{\mathcal{M}}$:

$$w(\boldsymbol{u}^*) = \frac{1}{[\boldsymbol{p}^*_{\mathcal{M}}]_z^2} \left(w'_M(\boldsymbol{u}^*) + \exp\left(-\frac{1}{2} \left[\frac{\lambda}{\kappa^{*,\max}_{\mathcal{M}}(\boldsymbol{u}^*)} \right]^2 \right) \right). \tag{5}$$

with $w'_M(\boldsymbol{u}^*) = c_{\mathcal{M}}(\boldsymbol{u}^*)/256$ derived from the model point's confidence counter $c_{\mathcal{M}}$ (see [6]). The denominator $[\boldsymbol{p}^*_{\mathcal{M}}]_z^2$ regularizes against noise (correlated with distance) since curvature computation is not reliable enough on data with low signal-to-noise ratio, with $[\boldsymbol{p}^*_{\mathcal{M}}]_z$ the z-Cartesian distance of the model point in meters. λ is a control parameter of the curvature-based weight $w(\boldsymbol{u}^*)$ and regulates its influence; for larger λ, the weight increasingly depends on the point's depth and confidence counter only.

5 Local Surface Reconstruction

As we will show, a key ingredient toward improved tracking robustness is paying particular attention to high accuracy (under real-time conditions) of the local surface reconstruction that data association, and with it pose error minimization, rely upon. To that end, we intersect the viewing ray with the second-order surface patches defined by the model point's orientation and curvature (see Sec. 5.1) and apply an elliptically-weighted blending scheme for all patch intersections resulting in the finally reconstructed surface point (see Sec. 5.2). Algorithm 1 summarizes our local surface reconstruction procedure.

Algorithm 1: Model maps generation using the updated index map (see Sec. 5.2).

```
Input: $\mathcal{M}$ (model), $\mathcal{I}^t$ (index map), $\varepsilon_d$ (surface thickness)
Output: $\mathcal{V}_\mathcal{M}$ (model vertex map), $\mathcal{N}_\mathcal{M}$ (model normal map), $\mathcal{K}_\mathcal{M}$ (model curvature map)
foreach pixel $\boldsymbol{u}$ in model map in parallel do
    $\boldsymbol{r}$ = generate ray for $\boldsymbol{u}$
    $\mathcal{P}$ = stable points in the vicinity of $\boldsymbol{r}$ using index map $\mathcal{I}^t$
    $z_\text{front} = -\inf$

    // Identify intersection points on closest surface
    $\mathcal{L} \leftarrow \emptyset$    // $\mathcal{L}_i.\boldsymbol{v}$: vertex, $\mathcal{L}_i.\hat{\boldsymbol{n}}$: normal, $\mathcal{L}_i.w$: weight
    foreach $\boldsymbol{q} \in \mathcal{P}$ do
        $(\boldsymbol{v}_q, \hat{\boldsymbol{n}}_q, \mathcal{K}_q) = \boldsymbol{r} \bigcap$ surface patch at $\boldsymbol{q}$  // see Sec. 5.1
        if $z_{front} - \varepsilon_d < \boldsymbol{v}_q.z$ then
            $w_q = \exp(-\frac{1}{2}\left(\left|\boldsymbol{q} - \boldsymbol{v}_q\right| / \mathcal{R}^t(\boldsymbol{u})\right)^2)$  // blend weight
            $\mathcal{L}$.append$(\boldsymbol{v}_q, \hat{\boldsymbol{n}}_q, w_q)$
        // Identify closest stable intersection point
        if $z_{front} < \boldsymbol{v}_q.z$ then
            $z_\text{front} = \boldsymbol{v}_q.z$
            $\mathcal{K}_\mathcal{M}(\boldsymbol{u}) \leftarrow \mathcal{K}_\mathcal{M}(\boldsymbol{q})$

    // Model map output w.r.t. points on closest surface
    $\mathcal{V}_\mathcal{M}(\boldsymbol{u}) \leftarrow \boldsymbol{0}$    // initialize model map vertex position
    $\mathcal{N}_\mathcal{M}(\boldsymbol{u}) \leftarrow \boldsymbol{0}$    // initialize model map normal
    $w_\text{valid} = 0$    // initialize sum of blend weights
    foreach $(\boldsymbol{v}, \hat{\boldsymbol{n}}, w) \in \mathcal{L}$ do
        if $z_{front} - \varepsilon_d < \boldsymbol{v}.z$ then
            $w_\text{valid} \leftarrow w_\text{valid} + w$
            $\mathcal{V}_\mathcal{M}(\boldsymbol{u}) \leftarrow \mathcal{V}_\mathcal{M}(\boldsymbol{u}) + w\boldsymbol{v}$
            $\mathcal{N}_\mathcal{M}(\boldsymbol{u}) \leftarrow \mathcal{N}_\mathcal{M}(\boldsymbol{u}) + w\hat{\boldsymbol{n}}$

    // Normalize result
    $\mathcal{V}_\mathcal{M}(\boldsymbol{u}) \leftarrow \mathcal{V}_\mathcal{M}(\boldsymbol{u}) / w_\text{valid}$
    $\mathcal{N}_\mathcal{M}(\boldsymbol{u}) \leftarrow$ normalize$(\mathcal{N}_\mathcal{M}(\boldsymbol{u}))$
    return
```

5.1 Quadratic Surface Patch Intersection

The main idea to incorporate the curvature information into the local surface reconstruction is to replace the standard ray-plane intersection used by previous works [3], [6] by an intersection with a higher-order surface with the same curvature as the one stored in the model point under consideration.

Knowing the Darboux frame $\{\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2, \hat{\boldsymbol{n}}\}$ and the curvature amplitudes $\{\kappa_1, \kappa_2\}$ for a given model point $\boldsymbol{p}$, we define an explicit quadratic surface parameterized over the $\hat{\boldsymbol{e}}_1$-$\hat{\boldsymbol{e}}_2$ tangent plane in local coordinates as:

$$F(x, y) = \tfrac{1}{2}\kappa_1 x^2 + \tfrac{1}{2}\kappa_2 y^2. \tag{6}$$

The local coordinate of the quadratic is represented by the Darboux frame centered at the current point position. We compute the intersection in the local coordinate frame, therefore we transform the ray $\boldsymbol{r}(\alpha) = \boldsymbol{q} + \alpha \boldsymbol{d}$ into local coordinates of the Darboux frame $\boldsymbol{r}'(\alpha) = \left(q'_x, q'_y, q'_z\right)^\top + \alpha\left(d'_x, d'_y, d'_z\right)^\top$. Some basic calculus reveals, that the intersection of $\boldsymbol{r}(\alpha)$ with the quadratic surface patch from Eq. 6 leads to this simple quadratic equation:

$$\frac{1}{2}\alpha^2 A + \alpha B + C \ = \ 0\,, \tag{7}$$

$$\text{where} \quad A = \kappa_1 d'^2_x + \kappa_2 d'^2_y, \quad B = \kappa_1 d'_x q'_x + \kappa_2 d'_y q'_y - d'_z\,,$$
$$\text{and} \quad C = \frac{1}{2}(\kappa_1 q'^2_x + \kappa_2 q'^2_y) - q'_z\,.$$

Solving this intersection equation usually leads to two solutions where we take the one closest to the related model point $\boldsymbol{p}$. The weight for this intersection point is calculated using a (non-normalized) Gaussian distribution, scaled by the model point's radius $\mathcal{R}^t(\boldsymbol{u})$ (see Alg. 1).

The resulting intersection point $\boldsymbol{s}' = \left(s'_x, s'_y, s'_z\right)^\top$ is expressed in the local Darboux frame coordinates. We additionally compute the associated surface normal as:

$$\hat{\boldsymbol{n}}_q = \frac{\boldsymbol{n}}{\|\boldsymbol{n}\|} \quad \text{with} \quad \boldsymbol{n} = \begin{pmatrix} 1 \\ 0 \\ \kappa_1 s'_x \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ \kappa_2 s'_y \end{pmatrix}, \tag{8}$$

which is analytically deduced from Eq. 6. The intersection point and its normal are finally back-transformed to camera coordinates.

5.2 Blending of Quadratic Surface Intersection Points

Since we compute a set of local surface intersections for each model-map pixel $\boldsymbol{u}$, we can easily compute a fast weighted average of all intersection points belonging to the first surface shell in our point-based model representation. As model points and their respective quadratic surface patches are not perfectly aligned in depth, due to noise and quantization, we blend points whose depth values fall into a *depth tolerance threshold* ε_d in order to identify all model points contributing to the model map at $\boldsymbol{u}$.

Our approach shows strong similarity to point-based rendering techniques, especially to differential point rendering [40], as well as elliptical weighted average (EWA) splatting [41], of which even curvature-rendering variants exist [42]. A key difference to previous works in that field is the order in which surface samples are collected, which eliminates the costly need for a dedicated normalization render step: rather than accumulating splat contributions for all pixels by each model point individually before dividing each pixel by the sum of relative weights accumulated so far, our index map (see Sec. 7) provides direct access to all model points contributing[1] to each pixel, allowing for local (and trivially parallel) evaluation of each surface intersection, rather than employing the costly distribute-and-gather process of traditional EWA splatting.

For each pixel, our parallel implementation identifies the intersection point at distance z_front closest to the camera, selects all intersection points lying within the given surface thickness ε_d, and blends the intersection points including their normals in order to get the final model map entries $\mathcal{V}_\mathcal{M}(\boldsymbol{u}), \mathcal{N}_\mathcal{M}(\boldsymbol{u})$. As blending curvature information is computationally more complex (see Sec. 6), the final curvature information is taken from the closest intersection point. Algorithm 1 describes this procedure in detail.

Fig. 3 shows a comparison of the model map quality for two different representation for two sequences, applying the

[1] Note that we artificially bound screen-space size of model points, as in a real-time acquisition setting, such cases would mean that the current camera data is of much higher quality than the coarse model points in question. This allows us to collect all points contributing to a pixel within a fixed-size window.

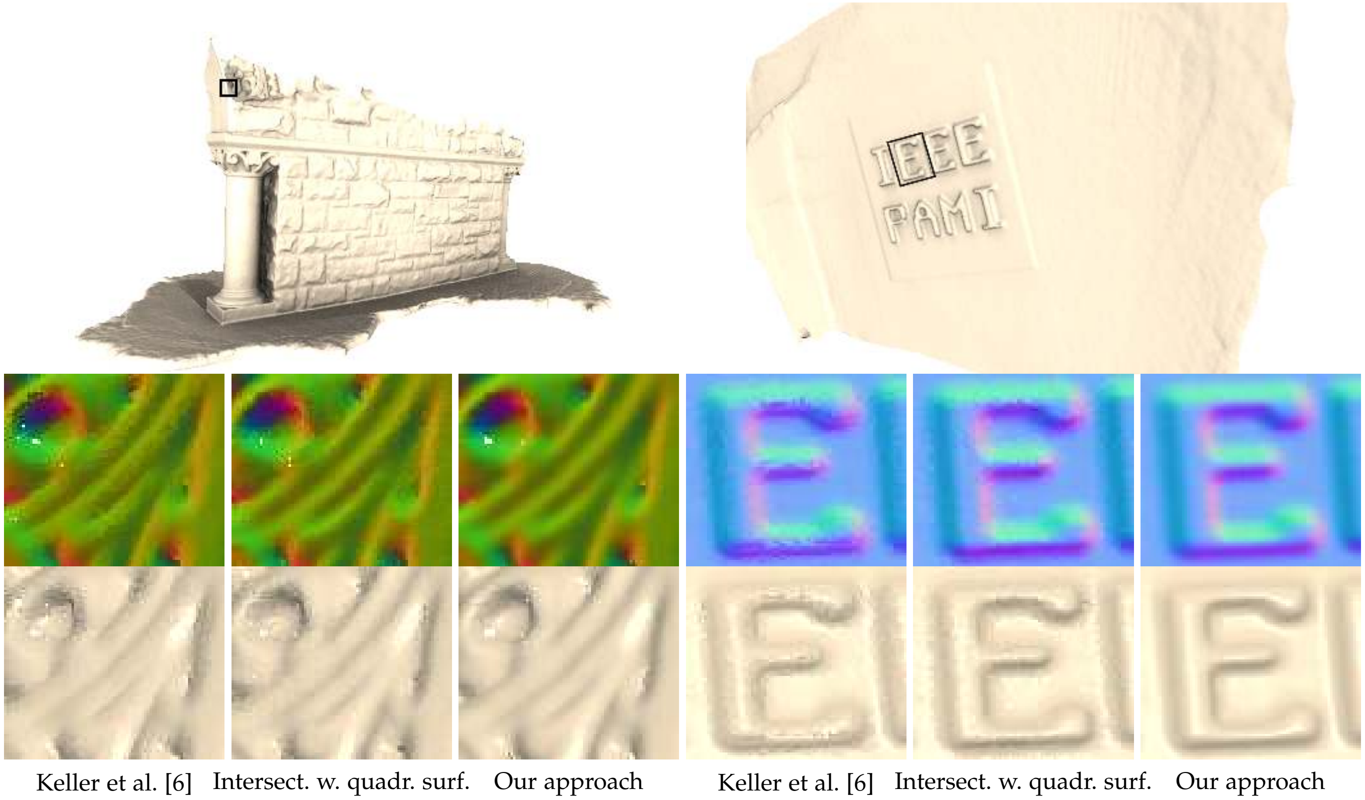


Fig. 3: Comparing model map quality for two different scenes. The first sub-column refers to the simple splatting proposed in [6]), the second to our quadratic surface intersection based on curvature information (see Sec. 5.1) and the third one is our blending scheme using our quadratic surface intersection (see Sec. 5.2).

splatting from Keller et al. [6] (left column), replacing the ray-plane intersection in the splatting with the intersection scheme for quadratic surfaces described in Sec. 5.1 (middle column), and the blending scheme for intersections with the quadratic surface explained in Sec. 5.2 (right column). It can be seen, that the model map quality already increases when curvature information is used, but the blending further mitigates discontinuities at splat boundaries. For all our experiments, we set the depth tolerance ε_d to 5 mm.

6 Depth Map Fusion

Conceptually, point-based data fusion follows Keller et al. [6] by accumulating geometric point attributes independently, thus, avoiding costly re-computation of normals and curvature from a local neighborhood of points.

We, too, use simple convex combination to accumulate an input point's position $\boldsymbol{p}_i$ into the position $\boldsymbol{p}_\mathcal{M}$ of its associated model point:

$$\boldsymbol{p}_\mathcal{M} \leftarrow \frac{c_i \boldsymbol{p}_i + c_\mathcal{M} \boldsymbol{p}_\mathcal{M}}{c_i + c_\mathcal{M}}, \quad c_\mathcal{M} \leftarrow c_i + c_\mathcal{M}, \tag{9}$$

where c_i and $c_\mathcal{M}$ are the input weight of the point, and the confidence counter of the model point (the accumulated weight of the input points), respectively.

Unlike Keller et al., however, who process normals $\hat{\boldsymbol{n}}_i$ and $\hat{\boldsymbol{n}}_\mathcal{M}$ analogously to point positions, we now have to blend the full curvature information of the input point $\{\hat{\boldsymbol{n}}_i, \hat{\boldsymbol{e}}_{1,i}, \hat{\boldsymbol{e}}_{2,i}, \kappa_{1,i}, \kappa_{2,i}\}$ with a model point's $\{\hat{\boldsymbol{n}}_\mathcal{M}, \hat{\boldsymbol{e}}_{1,\mathcal{M}}, \hat{\boldsymbol{e}}_{2,\mathcal{M}}, \kappa_{1,\mathcal{M}}, \kappa_{2,\mathcal{M}}\}$, which cannot be done linearly.

Our approach fuses the Darboux frames using a fractional rotation (slerp) of the model Darboux frame towards the input frame using the unique 3D rotation matrix $\boldsymbol{R}(\hat{\boldsymbol{a}}, \phi)$ described by a rotation axis $\hat{\boldsymbol{a}}$ and an angle ϕ, transforming between the frame $\boldsymbol{C}_\mathcal{M} = (\hat{\boldsymbol{e}}_{1,\mathcal{M}}, \hat{\boldsymbol{e}}_{2,\mathcal{M}}, \hat{\boldsymbol{n}}_\mathcal{M})$ and $\boldsymbol{C}_i = (\hat{\boldsymbol{e}}_{1,i}, \hat{\boldsymbol{e}}_{2,i}, \hat{\boldsymbol{n}}_i)$:

$$\begin{aligned} &\boldsymbol{C}_\mathcal{M} \leftarrow \boldsymbol{R}(\hat{\boldsymbol{a}}, \alpha\phi)\boldsymbol{C}_\mathcal{M} \\ &\text{with} \quad \boldsymbol{R}(\hat{\boldsymbol{a}}, \phi) = \boldsymbol{C}_i \boldsymbol{C}_\mathcal{M}^\top \quad \text{and} \quad \alpha = \frac{c_i}{c_i + c_\mathcal{M}}. \end{aligned} \tag{10}$$

As described in Sec. 4.1, the Darboux frame is unique up to inversion around $\hat{\boldsymbol{n}}$, thus input frames are suitably inverted in order to ensure fractional rotation along the shorter path. Finally, the curvature amplitudes are accumulated analogously to Eq. 9.

7 Deep Index Map

The previous sections primarily focused on quality improvements that lead to drift reduction; equally important, however, is real-time processing that operates at the camera's native frame rate. This is not the least as there is a direct relationship between throughput of range images and reduction of drift: aligning more range maps yields more data per surface area and thus less measurement uncertainty; it also implies shorter baselines between consecutive frames, which in many scenarios translates to higher stability of the range-map alignment, that is, camera pose estimation.

In order to efficiently handle models of up to several million points including their associated attributes, our entire reconstruction pipeline is implemented on the GPU using CUDA. Such an implementation has to support various operations at once, including efficient spatial addressing for *data association*, *local surface reconstruction*, *point attribute manipulation* during fusion, and efficient removal of *outliers* and *invalid model points* due to moving objects in the scene.

Keller et al. [6] enable efficient spatial access to the unordered point cloud by introducing a simple screen-space data structure, the *index map*, in the data association and fusion stages. Rather than rendering a dense surface reconstruction from the camera's perspective to determine all camera viewing ray-surface intersections, they render only pixel-sized points that encode vertex indices rather than colors in the output map. Thus, model points that project close to a given camera pixel $\boldsymbol{u}^*$ can easily be identified by looking up the index map in the vicinity of $\boldsymbol{u}^*$. The remainder describes our extensions to their index map that improve performance at all pipeline stages.

7.1 Point Collisions

Depending on viewing distance and model resolution, multiple model points may map to the same pixel. In order to more reliably determine a suitable match, Keller et al. reduce the risk of collisions by 4×4-upsampling their index map.

In our experiments, however, we often found that much of the 16-fold increased pixel space of the upsampled index map remains unused while collisions are still frequent. The effect is similar to what is observed in memory cache design: even with uniformly random-distributed data, collisions are extremely likely.

Borrowing from cache design, which addresses this problem through the *set associative cache* that trades cache address space for additional storage to resolve collisions, we *reduce* the amount of upsampling and (partially) resolve collisions by storing multiple point indices per pixel.

Concretely, our *deep index map* stores indices for up to two stable and one unstable points at each pixel position, which allows us to reduce the upsampling to 2×2 while still losing fewer points to collisions than Keller et al.'s approach at 4×4. Furthermore, by allocating separate capacity for storing stable and unstable points we effectively eliminate situations where unstable points occlude stable model points, a case that can hamper depth map fusion, for instance in the presence of dense outliers due to a misregistered camera frame or moving object.

Note that parts of the model seen from a far distance, or under an oblique angle, create more collisions than can be held by the deep index map. In practice, however, such cases tend to occur in spatial regions where the camera data is deemed too unreliable to be fused into the model.

7.2 Screen-Space Updates

Any point update in the fusion stage, be it merging of an input point (which could have a point change its position in screen-space) or removal due to point expiration or free-space violation, triggers removal of one of the original model points. In Keller et al.'s implementation this required copy-intensive model point cloud array compaction in every frame.

In contrast, we eliminate the need for costly point copy operations by first marking all required deletions within the deep index map itself, in a dedicated per-pixel "removal index" field. New, incoming points are either merged with existing model points or held in a queue for later insertion. Once all removals are known, the index buffer's removal index plane is sorted (in parallel) to obtain a list of freed positions in the model point list, which is where new model points from the insertion queue are inserted; excess insertion points are appended to the model list, unused freed positions remain tagged as free for future use.

By furthermore also logging newly created vertex indices in the deep index map, we keep the map fully up to date during fusion, which allows the final rendering stage to retrieve the latest model version directly from that map, without the need to one more time iterate over millions of points for rendering.

In summary, our deep index map improves and speeds up critical operations throughout the pipeline, which helps us maintain real-time rates with the curvature-enhanced online reconstruction pipeline.

8 Results

Since inception of the ICP algorithm [9], [10], copious design variants have been implemented. Interplay of the different design decisions is nontrivial, and the best holistic analysis of the ICP design space available [29] predates real-time, dense ICP implementations. This, and the sheer variety of implementations in existence, makes it difficult to determine the authoritative baseline our design should be compared against. We believe, however, to have found a meaningful set of comparisons and evaluations that expose the inherent benefits of our approach.

We decided to compare our approach to the following state-of-the-art techniques in the context of KinectFusion-like surface reconstruction, due to a combination of their availability, their proven high quality, and their input-output compatibility with our implementation.

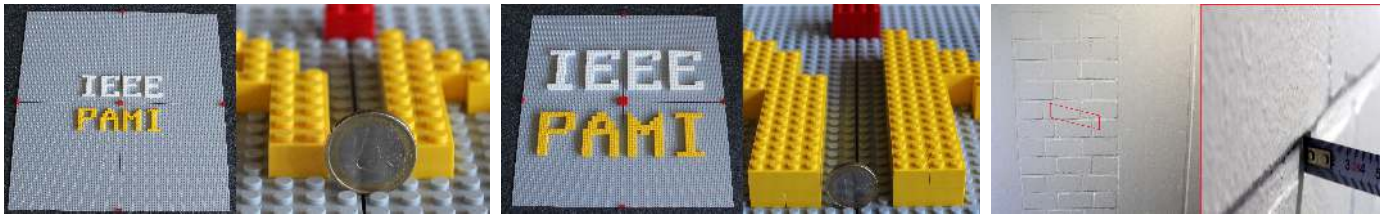
Kell13: The point-based approach given by Keller et al. [6]. We use the authors' implementation.

Nies13: Niessner et al. [8] voxel-based hashing technique. We use the authors' implementation (http://graphics.stanford.edu/~niessner/niessner2013hashing.html); we always use the finest grid resolution of 4 mm.

Sera15: Serafin et al. introduce Dense Normal Based Point Cloud Registration (NICP) [30], [43]. Like our approach, their method builds upon derivatives for improved reconstruction. We employ their publicly available implementation, using the configuration file as provided for the data set by Pomerleau et al.; only the camera parameters were updated appropriately.

To have their computationally costly CPU implementation of NICP run in real time, Serafin et al. suggest to work on images of one quarter of the original size. However, to provide a fairer quality comparison, we run NICP off-line, at full resolution (640×480, or 512×424, respectively); consequently, the presented results would currently not be achievable within an online system.

Ours: Our approach is based on curvature as an attribute throughout the reconstruction pipeline, and on a high-quality local surface reconstruction generation. For



(a) LEGO™ small scale (b) LEGO™ big scale (c) Brick wall

Fig. 4: Objects used to create each of our scenery. A 1€ coin (ø = 2.325 cm) is used to visualize the object scale.

further analysis, we partially disable some of our innovations, to observe their effect on tracking quality (see Sec. 8.3 and the supplemental material).

Various other approaches for online fusion have been proposed, using conceptually orthogonal ways to address drift, including off-line and online optimization of camera poses and surface reconstruction [3], [18], [44]. In this paper, however, we look specifically at the merits of comprehensive use of curvature information throughout the reconstruction pipeline. We hence do not directly compare to algorithms that employ complementary approaches that conceivable could be combined with our proposed methodology.

We present results using several reference data sets with and without geometric and/or camera pose ground truth:

Lego-PAMI-TT: The letter sequence "IEEE PAMI" constructed out of LEGO™ pieces on a 80×80 cm^2 ground plate is acquired using a Kinect. The camera is fixed at about 80 cm height above the scene, which resides on a precisely controllable turntable. The scene's controlled rotation—one 360° revolution in the course of 1601 frames—yields ground truth (relative) camera poses where the first and the last positions coincide.

We use two different scales of the "IEEE PAMI" scene: **Lego-PAMI-TT**$^{\times1}$ and **Lego-PAMI-TT**$^{\times2}$ with a single and a double block width (uniform scale factor of 2), respectively (see Fig. 4).

We access geometry ground truth with high precision, using the LEGO™ Designer software (http://ldd.lego.com/en-us/). However, the LEGO™ knobs (cylindrical connectors) are at or below the depth resolution limit of current Kinect range cameras.

Lego-PAMI-Free: The two "IEEE PAMI" scenes are acquired with a free-hand uncontrolled camera motion at about 50 – 100 cm distance from the target and acquiring some 1,100 frames. Here, ground truth is available for geometry only, not for camera pose.

Stone-Wall: This data set by Zhou and Koltun [13] comprises some 2,700 input frames of a wall with approx. 5.8 m × 2.8 m × 0.7 m size, acquired with Asus Xtion Pro Live range camera and including a prominent loop closure. (www.stanford.edu/~qianyizh/projects/scenedata.html).

Brick-Wall: This scene comprises of a nearly planar wall with very thin depth features (≤ 4 mm) only present at the wall's brick interstices (see Fig. 4). The wall was acquired using a hand guided Kinect at a distance of approx. 50 – 100 cm, yielding some 800 depth frames. The scenery covers approximately 1.80×1.70m.

Neither ground truth of camera poses nor geometry are available. For improved visualization we convert the images to gray-scale, yielding a better impression on the fine wall structures.

Racing-Car-R3: Wasenmüller et al.'s time-of-flight sequence [45] comes with a high-quality ground truth mesh that allows for direct evaluation of reconstruction error.

mit_76-417b: The dataset by Xiao et al. [46] offers a long-distance structured-light sweep of a large-scale open office space, which is suitable to demonstrate performance for very large datasets, in terms of both memory efficiency and camera drift over time.

The three scenes **Lego-PAMI-TT**, **Lego-PAMI-Free**, and **Brick-Wall** have been acquired using the first Kinect version, based on a structured light (SL) technique, as well as with a new Kinect based on the time-of-flight (ToF) principle. Subscripts are used to distinguish between the two Kinect types, e.g., **Lego-PAMI-TT**$_{\text{SL}}$ and **Lego-PAMI-TT**$_{\text{ToF}}$. The Kinect-SL and the Kinect-ToF have quite different quality levels for depth, noise, and other error sources; see Sarbolandi et al. [47] for a detailed discussion.

Furthermore, we provide evaluation results on TUM data sets [48] in the supplementary material.

8.1 Qualitative Evaluation

We first demonstrate the robustness of our method in a qualitative way by visually comparing its reconstruction results to state-of-the-art methods.

Fig. 5 shows a reconstruction comparison using all **Lego-PAMI-Free** data sets. Even for the double sized LEGO™ scene (col. 1 and 2) some of the state-of-the-art methods, like **Nies13** have severe difficulties in tracking the camera motion. In general, the tracking is more robust for Kinect-ToF data sets than for the Kinect-SL ones, which is most likely due to its better quality in depth resolution and noise [47]. Col. 3 and 4 in Fig. 5 demonstrate the robustness of our curvature-enhanced tracking method. Virtually all state-of-the-art methods completely fail to retrieve a valid camera motion and/or an appropriate reconstruction for the small scale scenery **Lego-PAMI-Free**$^{\times1}_{\text{SL/ToF}}$, whereas our method yields robust reconstruction even for the Kinect-SL with its lower depth quality.

Fig. 6, left, shows the reconstruction of the **Brick-Wall**$_{\text{ToF}}$. Note how well our method is able to robustly reconstruct the subtle wall structure. Nevertheless, our method is still not able to properly reconstruct the Kinect-SL acquired scenery **Brick-Wall**$_{\text{SL}}$ (see Sec. 9).

For completeness, Fig. 7 compares our approach against the offline global optimizer **Zhou13** [13]. Our approach shows no apparent drift, i.e., when returning to the initial camera pose, our method seamlessly completes the model.

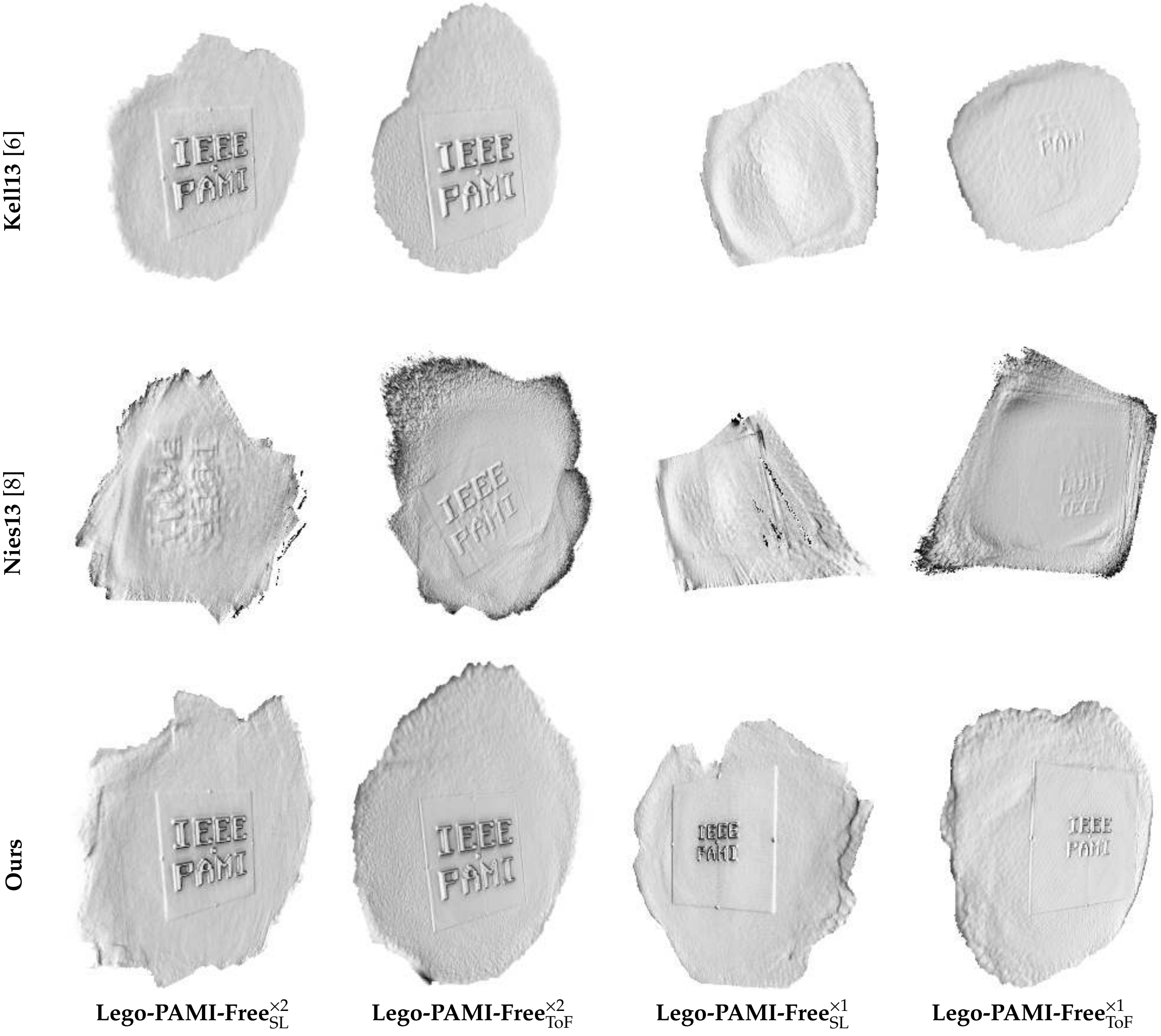

Kell13 [6]
Nies13 [8]
Ours
Lego-PAMI-Free$^{\times2}_{\mathrm{SL}}$
Lego-PAMI-Free$^{\times2}_{\mathrm{ToF}}$
Lego-PAMI-Free$^{\times1}_{\mathrm{SL}}$
Lego-PAMI-Free$^{\times1}_{\mathrm{ToF}}$

Fig. 5: Comparison of reconstructions for the **Lego-PAMI-Free** sequences.

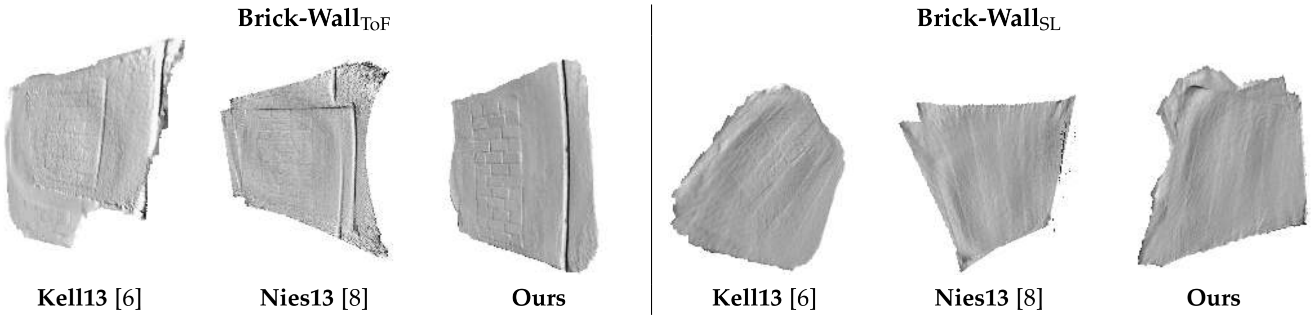


Fig. 6: Comparison of reconstructions for the **Brick-Wall**$_{\mathrm{ToF}}$ (three leftmost images) and for the **Brick-Wall**$_{\mathrm{SL}}$ (three rightmost images) sequence.

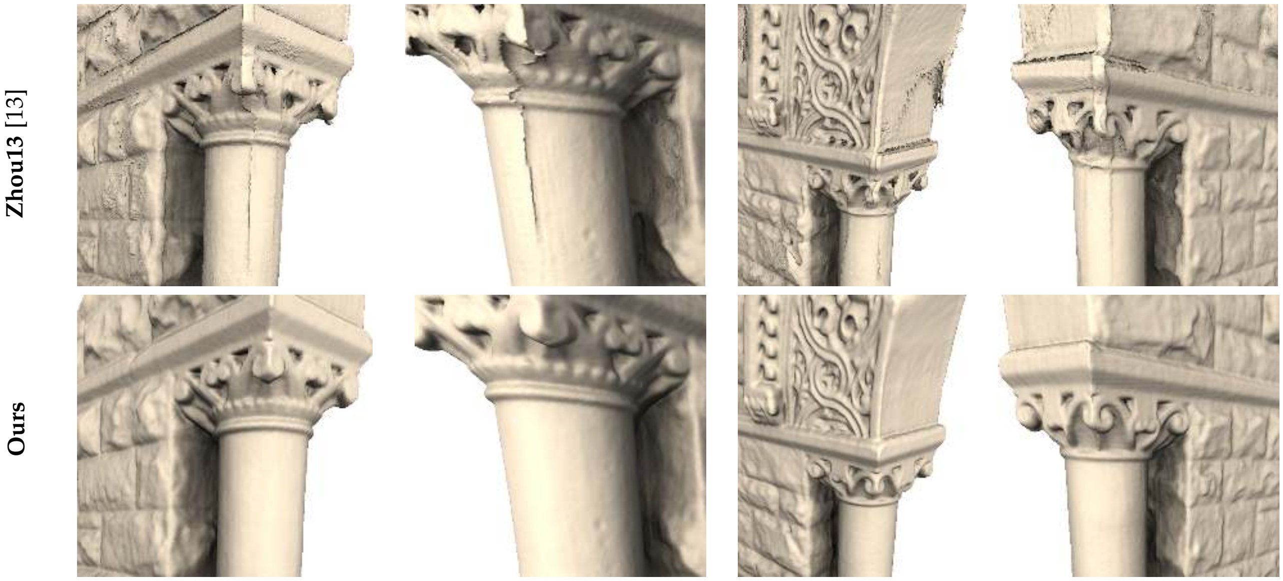


Fig. 7: Comparison of reconstructions for the **Stone-Wall**$_{\text{SL}}$ sequence, reconstructed using the offline global optimizer from Zhou and Koltun [13] (top row) and our method (bottom row). Images on the left side refer to the right column of the stonewall and the images on the right side to the left column of the stonewall, where the acquisition starts and ends.

We want to emphasize here that our method leads to an artifact-free reconstruction due to a valid ego motion. However, a one-to-one comparison of visual results is not possible since both methods use different model representation (*TSDF over a uniform voxel grid* and *point-based*).

8.2 Quantitative Ground Truth Evaluations

Our first series of experiments considers scenarios in which a reliable ground truth exists for comparison.

8.2.1 Camera Tracking

We evaluate camera tracking accuracy using the turntable dataset **Lego-PAMI-TT**, which provides a measure of the robustness of the compared methods. Due to the rigid acquisition setup, the camera pose estimated in the ICP should ideally result in an equidistantly sampled, perfect circle, with a constantly rotated optical axis. Therefore, we generate reference poses in order to evaluate the estimated camera poses as follows:

1. RANSAC fit of camera poses to a plane, removing the influence of outliers,
2. RANSAC fit of a circle in the plane to the projected camera centers, and
3. projection of the initial camera pose to the circle as starting point for regularly sample the circle.

Using these reference camera poses, we calculate the *camera center error* as the Euclidean distance between the estimated ICP-pose and the corresponding reference point. In order to compute the *rotation angle error* we extract the angular argument of the rotational transformation matrix between the initial pose 0 and the current pose at t and compare it against the ideal angular offset $\Delta p = (360/1600)^\circ$ between neighboring frames: $\left|\text{angle}(\boldsymbol{R}_t \boldsymbol{R}_0^\top) - t\Delta p\right|$.

We quantitatively evaluate the reconstructed geometry for all **Lego-PAMI-TT** data sets by extracting the relevant scenery from reconstructed geometry, registering the reconstructed geometry to the ground truth LEGO™ model and, finally, computing distance errors using CloudCompare [49].

Tab. 2 presents plots of the camera center and the camera rotation angle errors, images of the geometric reconstruction error, and error statistics (mean, standard deviation, min and max) for the **Lego-PAMI-TT**$_{\text{SL/ToF}}$ data sets. (For **Lego-PAMI-TT**$^{\times 1}_{\text{ToF}}$, **Sera15** was unable to lock onto the geometry, producing an invalid trajectory of an almost static camera position; these results are hence excluded.)

As expected, the worse signal-to-noise ratio for smaller geometric features of the small scale datasets **Lego-PAMI-TT**$^{\times 1}$ decreases the stability of the ICP-based camera tracker, leading to larger camera center and rotation angle errors for all methods. Apparently, our proposed method is much more robust than the state-the-art methods, which have severe difficulties to retrieve the correct ego motion.

Comparing the small scale with the large-scale data set **Lego-PAMI-TT**$^{\times 2}_{\text{ToF}}$, two facts seem to be counter-intuitive. Firstly, the geometric error for the small scene is smaller than for the large scene, even though the tracking is less robust, and, secondly, our method yields slightly larger camera center and rotational angle errors for the large-scale data set **Lego-PAMI-TT**$^{\times 2}_{\text{ToF}}$ than state-the-art methods. The first aspect is explained by comparing a *single input frame* to the ground truth geometry. This results in less geometric error for the small scale (mean=0.590, SD=0.557, min=0, max=5.943) than for the big scale (mean=0.745, SD=0.766, min=0, max=6.659). This is most likely due to different camera error effects such as multi-path, flying pixel, etc. [47]. Furthermore, the averaging applied within any KinectFusion-like approach erases geometric errors due to erroneous tracking after some frames. We have no consistent explanation for the second aspect, i.e., the smaller rotation angle error in conjunction with an extreme camera center error for **Kell13**. However, our approach is the only one robust according to both, camera center and rotation angle error.

8.2.2 Impact of Noise

As our method is based on second-order derivatives, and because derivatives are known to be sensitive to noise, we conduct a systematic study on the influence of noise on the camera tracking accuracy. Our noise evaluation is based on the large-scale turntable data set **Lego-PAMI-TT**$^{\times 2}_{\mathrm{ToF}}$. As in Sec. 8.2.1, we use circle fitting to create reference poses.

We evaluate the estimated camera trajectories at different levels of (Gaussian) noise. Starting from a noise level typical for ToF cameras, we successively increase the noise's standard deviation by integer factors, through addition of normal-distributed noise to the original (noisy) depth values; for the principal point of the Kinect-ToF, the SD for measured depth values at 80 cm distance (**Lego-PAMI-TT**$^{\times 2}_{\mathrm{ToF}}$) is about 1.1 mm, see Sarbolandi et al. [47].

We compare our approach to **Kell13** [6], **Nies13** [8] and **Sera15** [30], [43]. **Kell13**, **Nies13**, and our method have in common that they use a bilateral prefilter to mitigate noise. **Sera15** does not prefilter and hence is more susceptible to noise. In order to ensure a meaningful comparison even toward higher noise levels, we additionally evaluate a version where we feed NICP with bilaterally prefiltered images (**Sera15 (modified)**).

In all of our experiments, we parameterize the bilateral filter with the same values: $\sigma_{\mathrm{D}} = 2.5$, $\sigma_{\mathrm{R}} = 0.03$, $r_{\mathrm{filter}} = 5$. For **Kell13**, we further enabled the use of positions from the bilaterally filtered map in their fusion stage (which otherwise would use positions from unfiltered data).

Fig. 8 shows the resulting mean error (top row) and SD (bottom row) for estimated camera centers (left) and rotational angles (right).

We generally observe that our method reliably produces lowest error at first, only to then suffer more than others as noise levels become very high. Depending on the error quantity considered, the cross-over point lies between five to twelve times of the natural noise level (SD 5.5–13.2 mm in our experiment); for a reference of scale, consider that the **Lego-PAMI-TT**$^{\times 2}_{\mathrm{ToF}}$ data set consists of LEGO™ double blocks with a height of 19.2 mm, without knobs.

We hence argue that under realistic imaging conditions, the benefits of incorporation of curvature tend to prevail; only in particularly high-noise scenarios, alternative approaches (in this case probably **Nies13**) should be preferred.

8.2.3 Surface Reconstruction Error

The **Racing-Car-R3**$_{\mathrm{ToF}}$ sequence by Wasenmüller et al. [45] provides a high-quality ground truth geometry for a complex object. We compare the reconstructions computed with **Kell13**, **Nies13**, and our approach to the ground truth geometry. We remove the floor manually so that only the relevant scenery remains, register the reconstructed geometry to the ground truth mesh and compute the distance error using CloudCompare [49].

For **Nies13**, we had to change the voxel size from 4 mm (default) to 6 mm; with the default settings, their method stopped adding depth data and produced a corrupted output mesh. For **Kell13**, we disabled the removal of dynamic parts of the scene. All other parameters were kept unchanged (default values).

Fig. 9 shows the absolute distance error [m] to the ground truth mesh. See Tab. 3 for mean and standard-deviation of the corresponding reconstruction errors.

Our method generally provides significantly lower reconstruction error than **Nies13**, while offering a more complete reconstruction (featuring fewer holes) than **Kell13**, which otherwise shows competitive reconstruction error.

8.3 Contributors to Robustness

We conducted an experiment in order to evaluate and justify the use of curvature in the individual stages of our reconstruction pipeline. We compare the following variants of our algorithm.

Base: Our reconstruction pipeline with deactivated curvature methods for ICP correspondences finding, ICP optimization, and local surface reconstruction, yielding reconstructions equivalent to Keller et al. [6].

ICP Weight: Base plus activated curvature based ICP weighting scheme (see Sec. 4.2).

Correspondence Finding: Base plus activated curvature based ICP correspondences finding (see Sec. 4.1).

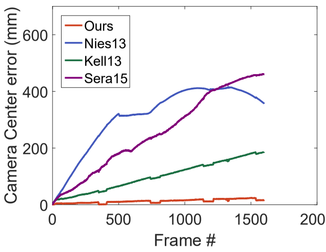
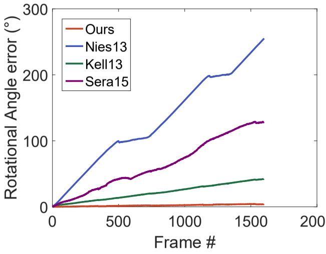
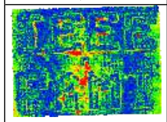
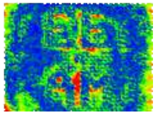
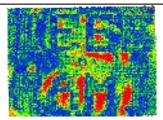
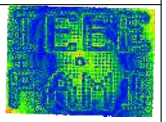
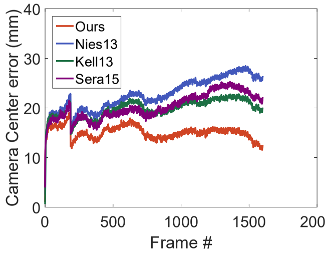
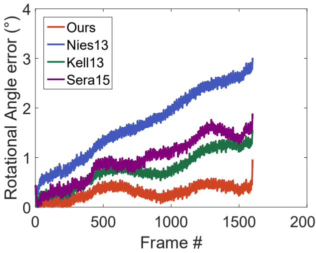
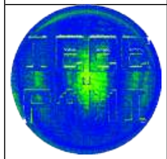
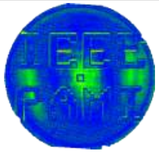
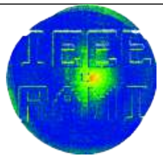
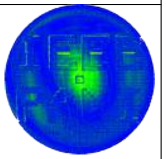
Local Reconstruction: Base plus activated enhanced local surface reconstruction using curvature information and blending (see Sec. 5).

Ours: Our approach incorporating all three curvature components.

We use the small-scale turntable data set **Lego-PAMI-TT**$^{\times 1}_{\mathrm{SL}}$, which consists of 1600 frames for a complete 360° turn (0.225° frame-to-frame rotation) and provides a camera pose reference. This data set is most challenging, as it has little depth variation. Furthermore, we increase the frame-to-frame motion by taking one every nth frame, $n \in \{5, 15, 25, 35, 45, 55, 65\}$, resulting in a total of seven experiments and five variants of the reconstruction pipeline.

Tab. 4 shows the camera center error statistics for all methods and experiments. We only evaluate results, if the camera tracking stage of the variant does not completely fail. More precisely, each of the methods that process the full frames of the turntable sequence **Lego-PAMI-TT**$^{\times 1}_{\mathrm{SL}}$ give a radius of the fitted circle equals to 26 ± 0.5 cm. Thus, experiment trajectories are rejected from the evaluation if their radius is not close enough to this radius. As can be seen from the results, weighting has a strong impact and for inter-frame rotation up to $45 \times 0.225° = 10.125°$ weighting alone has very similar results compared to our fully curvature equipped method. Furthermore, curvature-based correspondences finding as well as local surface reconstruction improve on the base algorithm, failing beyond an inter-frame rotation of $15 \times 0.225° = 3.375°$. Weighting alone exhibits strongly decreased robustness at an inter-frame rotation of $55 \times 0.225° = 12.375°$ and fails afterwards.

For completeness, our approach starts degenerating beyond an inter-frame rotation up to $66 \times 0.225° = 14.85°$, thus applying all curvature-based components clearly leads to improved robustness of the overall reconstruction system compared to an isolated curvature component based application.

	Error Type	Methods			
		Kell13	Nies13	Sera15	Ours
Lego-PAMI-TT$^{\times 1}_{\text{SL}}$	Center [mm] Mean, STD Min, Max	99.787, 52.101 4.866, 186.679	313.268, 111.632 5.323, 415.532	256.363, 138.939 1.826, 461.583	**12.748, 6.128** **0.186, 24.679**
	Rot. Angle (°) Mean, STD Min, Max	21.687, 11.984 0.409, 42.313	130.331, 68.766 0.197, 255.062	64.026, 39.040 0.044, 129.079	**2.177, 1.209** **0.002, 4.390**
					
	Reconst. [mm] Mean, STD Min, Max	0.803, 0.744 **0.000**, 4.995	0.899, 0.863 0.001, 4.728	0.979, 1.043 **0.000**, 8.549	**0.685, 0.602** **0.000, 3.547**
					
Lego-PAMI-TT$^{\times 2}_{\text{SL}}$	Center [mm] Mean, STD Min, Max	19.883, 1.785 **0.618**, 22.796	22.774, 3.182 2.550, 28.480	20.274, 2.609 3.963, 25.290	**15.065, 1.310** 1.068, **19.212**
	Rot. Angle (°) Mean, STD Min, Max	0.762, 0.342 **0.000**, 1.564	1.682, 0.704 0.004, 3.010	0.971, 0.446 0.004, 1.887	**0.305, 0.138** 0.001, **0.959**
					
	Reconst. [mm] Mean, STD Min, Max	1.012, 1.049 **0.000, 9.867**	1.061, 1.045 **0.000**, 10.244	1.103, 3.237 **0.000**, 314.765	**0.898, 0.906** **0.000**, 11.233
					

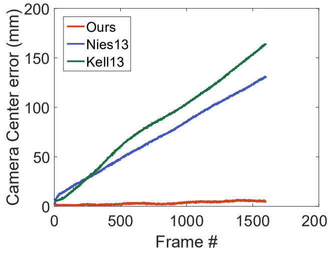
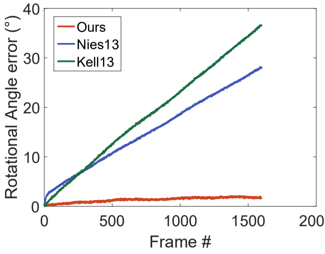
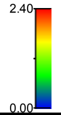
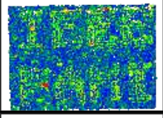
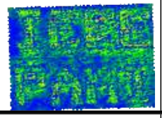
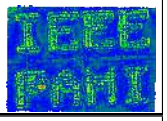
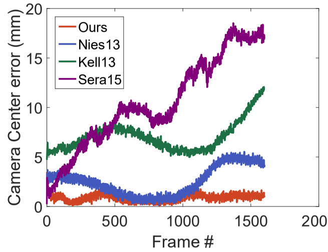
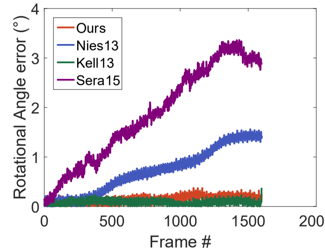
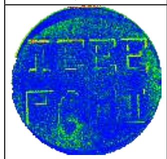
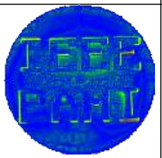
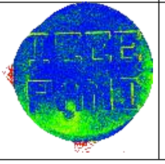
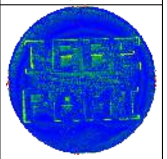
	Error Type	Methods			
		Kell13	Nies13	Sera15	Ours
Lego-PAMI-TT$^{\times 1}_{\text{ToF}}$	Center [mm] Mean, STD Min, Max	84.866, 45.723 5.608, 164.437	70.532, 35.280 1.772, 131.159	∅	**3.274, 1.619** **0.532, 6.911**
	Rot. Angle (°) Mean, STD Min, Max	18.760, 10.378 **0.015**, 36.669	15.397, 7.457 0.212, 28.145	∅	**1.342, 0.480** 0.086, **2.117**
					
	Reconst. [mm] Mean, STD Min, Max	0.466, 0.429 **0.000**, 5.030	0.445, 0.387 **0.000, 2.402**	∅	**0.398, 0.383** **0.000**, 3.183
				∅	
Lego-PAMI-TT$^{\times 2}_{\text{ToF}}$	Center [mm] Mean, STD Min, Max	7.069, 1.507 4.708, 12.080	2.520, 1.471 0.130, 5.534	10.504, 4.588 0.229, 18.524	**0.877, 0.386** **0.040, 3.453**
	Rot. Angle (°) Mean, STD Min, Max	**0.087, 0.057** **0.000**, 0.378	0.715, 0.461 **0.000**, 1.562	1.866, 0.966 0.141, 3.364	0.140, 0.077 **0.000, 0.375**
					
	Reconstr. [mm] Mean, STD Min, Max	0.725, **0.806** **0.000**, 11.710	0.726, 0.874 **0.000, 8.375**	1.183, 19.065 **0.000**, 3173.000	**0.637**, 0.807 **0.000**, 25.268
					

TABLE 2: Comparison of the ego-motion robustness for three different methods based on small (top row) and large (bottom row) scales of **Lego-PAMI-TT$_{\text{SL}}$** (left column) and **Lego-PAMI-TT$_{\text{ToF}}$** (right column) data sets. **Bold numbers** refer to the smallest error for each given error statistic. Reconstruction runs where a method's camera tracking failed completely are left out from the comparison (and marked as ∅).

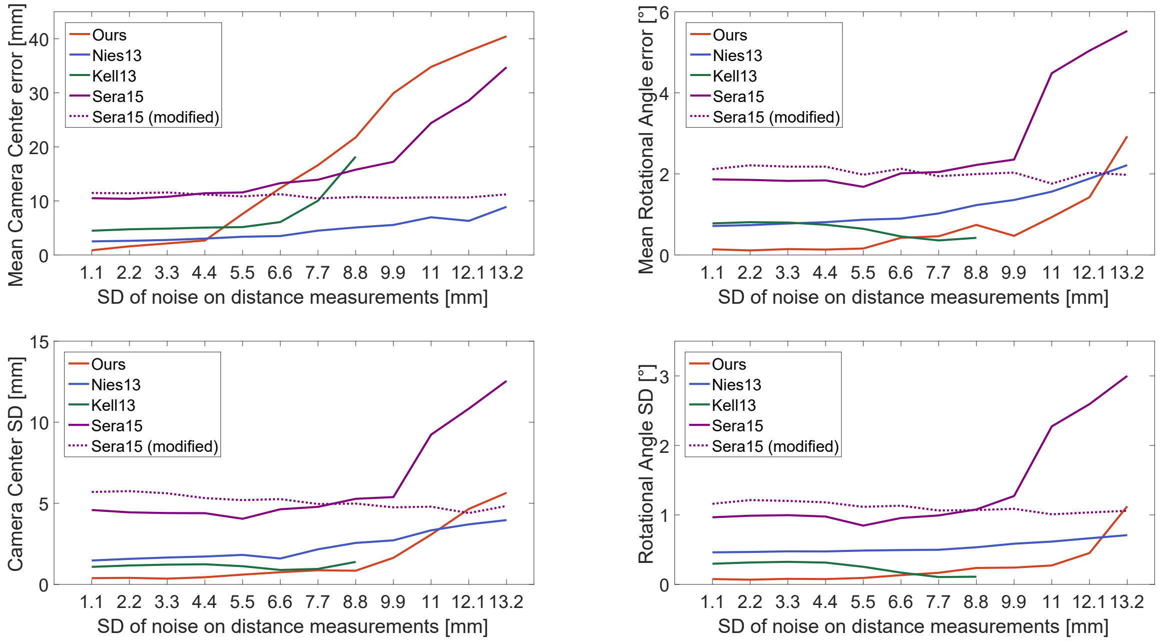
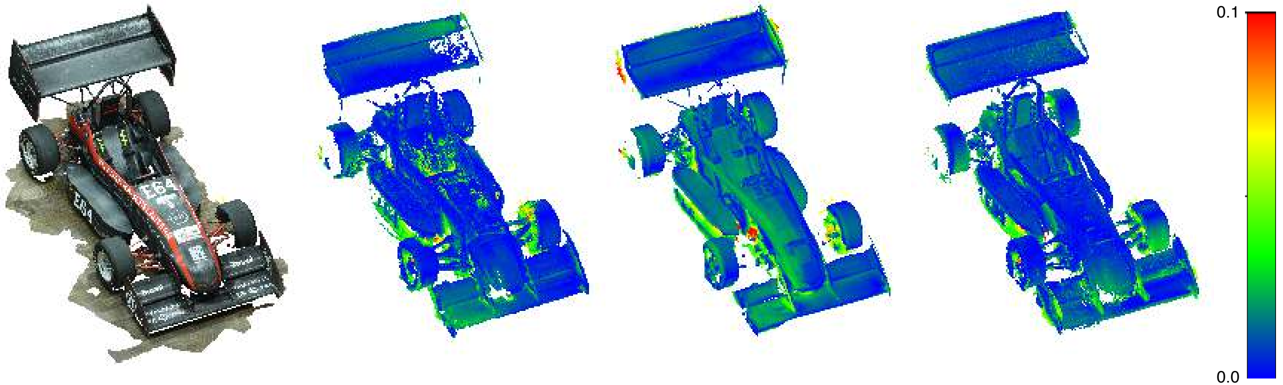


Fig. 8: Mean error (top row) and SD (bottom row) for estimated camera centers (left) and rotational angles (right) with increasing noise for the **Lego-PAMI-TT$^{\times 2}_{\mathrm{ToF}}$** scene. Data points where the camera tracking completely failed are omitted. In order to make the results comparable, we include the results for a modified version of **Sera15** using bilaterally prefiltered depth images.



Ground Truth **Kell13** [6] **Nies13** [8] **Ours**

Fig. 9: Comparison of reconstructions for the **Racing-Car-R3$_{\mathrm{ToF}}$** sequence by Wasenmüller et al. [45]. For every model point, the absolute distance error [m] to the ground truth mesh is visualized using the CloudCompare tool [49].

	Kell13	Nies13	Ours
Mean, SD	10.405, 12.757	25.371, 24.093	**9.250**, **9.743**
Min, Max	**0.000**, 155.180	**0.000**, 200.466	**0.000**, **153.895**

TABLE 3: Absolute distance error [mm] for the **Racing-Car-R3$_{\mathrm{ToF}}$** sequence [45]. For every model point, the absolute distance error to the ground truth mesh is calculated.

8.4 Scalability

We compare our approach to Keller et al. [6] and Niessner et al. [8] on the large scene **mit_76-417b$_{\mathrm{SL}}$** by Xiao et al. [46].

For **Nies13**, the voxel size was changed from 4 mm to 10 mm. With 4 mm, **Nies13** stopped adding depth data in the beginning of the scene. For **Kell13**, we disabled the removal of dynamic parts of the scene.

Fig. 10 shows the reconstructions for the **mit_76-417b$_{\mathrm{SL}}$** sequence by Xiao et al. [46]. Using CloudCompare [49], the reconstructions are aligned and rendered using EDL (Eye-Dome Lighting). **Kell13** lost ICP tracking and could not recover. See the inset for a reconstruction with ElasticFusion, as shown in their YouTube video.

On this challenging dataset that is prone to drift over large scales, our method generally performs as well as, or better than, **Nies13**. As far as visible in the video, ElasticFusion, which employs non-rigid surface deformation, preserves rectilinearity of the office isles better; we note that their deformation scheme is a design decision orthogonal to our incorporation of curvature, and it is conceivable that both could be combined.

Lastly, note that our reconstruction consists of 24,630,119 surfels, while the **mit_76-417b$_{\mathrm{SL}}$** sequence consists of 11,158 frames. This demonstrates the space-efficiency of the point-based-fusion scheme, where each input frame on average contributes only 2,207 points to the final model.

8.5 Performance

All our tests were performed with an Intel Core i7-4790 with an NVIDIA GeForce GTX 980 Ti with 6GB VRAM. Tab. 5 states the timings for all processing modules of our system. Apparently, due to our implementation improvements for handling large point data sets (see Sec. 7) the proposed approach scales very well with respect to the size of the scene and the number of points in the final model.

9 Discussion

We present a novel real-time, point-based reconstruction framework for robust surface extraction using curvature as a an independent quantity. Our approach significantly reduces drift, thus improving camera tracking and reconstruction quality. Particularly, our method is able to robustly reconstruct scene with very low depth-feature information, not possible with state-of-the-art methods. Finally we build a new benchmark data set that provides ground truth camera poses and geometry using both Kinect cameras, supporting further research in the field.

Our approach is still limited in not being able to successfully reconstructed scenes with few depth features, when the input depth maps have a high noise level. Fig. 6, right, shows such a failure case for the **Brick-Wall**$_{\text{SL}}$ sequence, for which the brick structure is still discernible in the individual input frame of the Kinect-SL camera.

Finally, our method does not solve explicitly loop closure, only reduce the overall camera drift, thus problems could still occur for large-scale scenarios.

Drop	**Base** $\mu \pm \sigma$ Min-Max	**Base & ICP Weighting** $\mu \pm \sigma$ Min-Max	**Base & Corresp. Finding** $\mu \pm \sigma$ Min-Max	**Base & Local Reconstr.** $\mu \pm \sigma$ Min-Max	**Ours** $\mu \pm \sigma$ Min-Max
1:5	3.25 ± 0.59 0.06 – 4.00	0.62 ± 0.23 0.09 – 0.95	2.97 ± 0.55 **0.02** – 3.68	2.97 ± 0.53 0.05 – 3.65	**0.45** ± **0.13** 0.09 – **0.72**
1:15	2.78 ± 0.55 0.07 – 3.31	0.55 ± 0.17 **0.01** – 1.00	2.43 ± 0.48 0.06 – 2.91	2.35 ± 0.47 **0.01** – 2.86	**0.43** ± **0.14** 0.12 – **0.85**
1:25	∅	0.55 ± **0.15** 0.05 – 0.83	2.00 ± 0.48 0.05 – 2.51	1.91 ± 0.46 **0.02** – 2.42	**0.46** ± **0.15** 0.12 – **0.76**
1:35	∅	0.57 ± 0.17 0.12 – 0.83	2.07 ± 0.50 **0.03** – 2.58	∅	**0.49** ± **0.13** 0.16 – **0.81**
1:45	∅	0.59 ± 0.21 **0.13** – 0.91	∅	∅	**0.51** ± **0.15** 0.20 – **0.84**
1:55	∅	7.86 ± 1.36 0.97 – 8.56	∅	∅	**0.52** ± **0.15** **0.20** – **0.86**
1:65	∅	∅	∅	∅	**0.56** ± **0.18** **0.18** – **0.90**

TABLE 4: Camera center error statistics in cm for the robustness experiment based on the **Lego-PAMI-TT**$_{\text{SL}}^{\times 1}$ and applied to Kell13, improved version of Kell13 and our fully curvature enhanced pipeline. $1 : n$ indicates that every nth frame is used for reconstruction. **Bold numbers** indicate the lowest error in its category and ∅ refers to a complete failure of the camera tracker.

Acknowledgments

This work was funded by the German Research Foundation (DFG) as part of the research training group GRK 1564 *Imaging New Modalities*, and by the UK Engineering and Physical Sciences Research Council (grant EP/K023578/1).

References

[1] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3D model acquisition," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 21, no. 3, pp. 438–446, 2002.

[2] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: real-time dense surface mapping and tracking," in *Proc. IEEE Int. Symp. Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136.

[3] T. Weise, T. Wismer, B. Leibe, and L. Van Gool, "In-hand scanning with online loop closure," in *Proc. IEEE Int. Conf. on Computer Vision Workshops (ICCV Workshops)*. IEEE, 2009, pp. 1630–1637.

[4] H. Roth and M. Vona, "Moving volume KinectFusion," in *British Machine Vision Conf.*, 2012.

[5] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald, "Robust tracking for real-time dense rgb-d mapping with kintinuous," Computer Science and Artificial Intelligence Laboratory, MIT, Tech. Rep., 2012.

[6] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3D reconstruction in dynamic scenes using point-based fusion," in *Proc. Conf. Joint 3DIM/3DPVT (3DV)*, 2013, p. 8.

[7] J. Chen, D. Bautembach, and S. Izadi, "Scalable real-time volumetric surface reconstruction," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 32, no. 4, pp. 113:1–113:16, Jul. 2013.

[8] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Trans. on Graphics*, vol. 32, no. 6, p. 169, 2013.

[9] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Robotics-DL tentative*. International Society for Optics and Photonics, 1992, pp. 586–606.

[10] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992.

[11] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy, "Geometrically stable sampling for the ICP algorithm," in *Int. Conf. 3D Digital Imaging and Modeling (3DIM)*, Oct. 2003, pp. 260–267.

[12] K. Pulli, "Multiview registration for large data sets," in *Int. Conf. 3D Digital Imaging and Modeling (3DIM)*, 1999, pp. 160–168.

[13] Q.-Y. Zhou and V. Koltun, "Dense scene reconstruction with points of interest," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 32, no. 4, pp. 112:1–112:8, 2013.

[14] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments," *Int. J. Robotics Research*, vol. 31, pp. 647–663, Apr. 2012.

[15] G. Godin, M. Rioux, and R. Baribeau, "Three-dimensional registration using range and intensity information," *Proc. SPIE (Videometrics III)*, vol. 2350, pp. 279–290, 1994.

[16] S. Weik, "Registration of 3-d partial surface models using luminance and depth information," in *Proc. Int. Conf. on Recent Advances in 3-D Digital Imaging and Modeling*, May 1997, pp. 93–100.

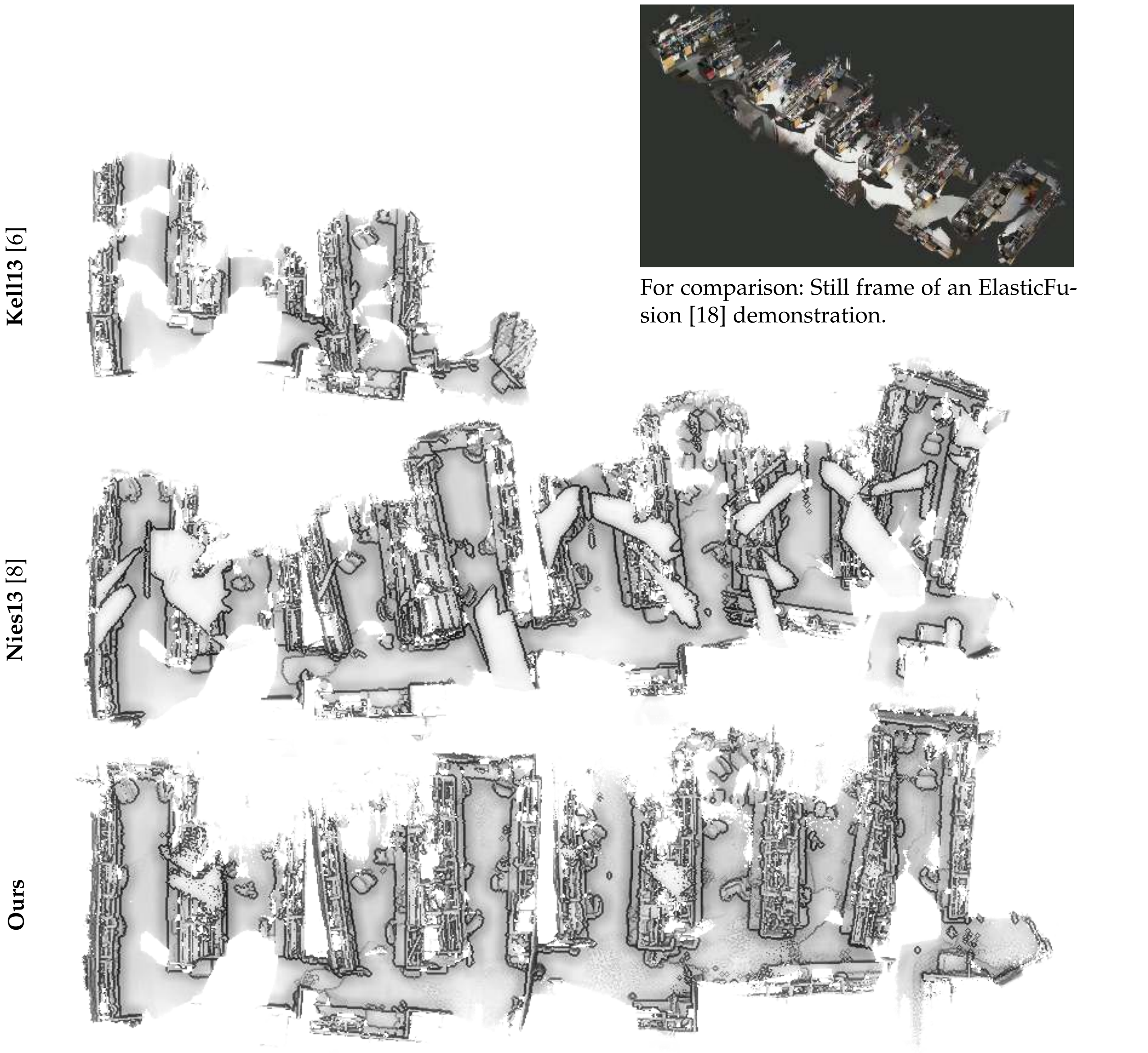
[17] Q.-Y. Zhou and V. Koltun, "Depth camera tracking with contour cues," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2015, pp. 632–638.

[18] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "Elasticfusion: Dense slam without a pose graph," in *Proceedings of Robotics: Science and Systems*, 2015.

[19] S. J. Julier, J. Uhlmann, and K., "Unscented filtering and nonlinear estimation," in *Proceedings of the IEEE*, vol. 92, no. 3, 2004, pp. 401–422.

[20] M. Sofka, G. Yang, and C. V. Stewart, "Simultaneous covariance driven correspondence (cdc) and transformation estimation in the expectation maximization framework," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[21] D. MacKinnon, V. Aitken, and F. Blais, "Review of measurement quality metrics for range imaging," *Journal of Electronic Imaging*, vol. 17, no. 3, pp. 033 003–033 003–14, 2008.



For comparison: Still frame of an ElasticFusion [18] demonstration.

Fig. 10: Comparison of reconstructions for the **mit_76-417b**$_{\text{SL}}$ sequence by Xiao et al. [46]. Comparison with ElasticFusion shows still frame of https://youtu.be/-dz_VauPjEU?t=3m52s.

Data-Set (# points)	**Brick-Wall**$_{\text{SL}}$ (487,682)	**Brick-Wall**$_{\text{ToF}}$ (466,775)	**Lego-PAMI-TT**$^{\times 1}_{\text{SL}}$ (268,618)	**Lego-PAMI-TT**$^{\times 1}_{\text{ToF}}$ (113,364)	**Lego-PAMI-TT**$^{\times 2}_{\text{SL}}$ (305,576)	**Lego-PAMI-TT**$^{\times 2}_{\text{ToF}}$ (121,853)	**Lego-PAMI-Free**$^{\times 1}_{\text{SL}}$ (989,535)	**Lego-PAMI-Free**$^{\times 1}_{\text{ToF}}$ (410,833)	**Lego-PAMI-Free**$^{\times 2}_{\text{SL}}$ (756,630)	**Lego-PAMI-Free**$^{\times 2}_{\text{ToF}}$ (756,630)	**Stone-Wall**$_{\text{SL}}$ (3,893,768)	**Racing-Car-R3**$_{\text{ToF}}$ (1,409,161)	**mit_76-417b**$_{\text{SL}}$ (24,630,119)
Processing Modules	$\mu \pm \sigma$ Mn-Mx	$\mu \pm \sigma$ Mn-Mx	$\mu \pm \sigma$ Mn-Mx	$\mu \pm \sigma$ Mn-Mx	$\mu \pm \sigma$ Mn-Mx	$\mu \pm \sigma$ Mn-Mx	$\mu \pm \sigma$ Mn-Mx	$\mu \pm \sigma$ Mn-Mx	$\mu \pm \sigma$ Mn-Mx	$\mu \pm \sigma$ Mn-Mx	$\mu \pm \sigma$ Mn-Mx	$\mu \pm \sigma$ Mn-Mx	$\mu \pm \sigma$ Mn-Mx
Preproc. Sec. 3	3.6 ± 0.0 3.6 – 3.8	2.5 ± 0.1 2.3 – 2.6	2.6 ± 0.1 2.3 – 2.6	1.2 ± 0.1 1.1 – 1.3	2.4 ± 0.0 2.4 – 2.7	1.3 ± 0.1 1.1 – 1.3	3.6 ± 0.0 3.6 – 3.9	2.5 ± 0.0 2.5 – 2.7	3.6 ± 0.0 3.6 – 3.8	2.4 ± 0.1 2.2 – 2.6	3.0 ± 0.4 2.0 – 3.4	2.2 ± 0.1 1.9 – 2.4	3.0 ± 0.2 2.1 – 3.4
Tracking Sec. 4	5.1 ± 2.0 0.7 – 14.7	3.9 ± 0.9 0.8 – 9.9	3.4 ± 0.3 0.6 – 4.4	3.0 ± 0.4 0.6 – 5.9	3.2 ± 0.3 0.7 – 4.5	2.2 ± 0.3 0.8 – 3.5	4.5 ± 0.7 0.8 – 9.5	4.3 ± 0.8 0.7 – 10.1	4.5 ± 0.6 0.7 – 6.2	3.9 ± 0.6 0.7 – 11.2	4.4 ± 0.6 0.7 – 8.9	4.1 ± 0.9 0.6 – 9.9	5.9 ± 1.5 0.8 – 17.8
Index Map Sec. 7	1.1 ± 0.0 0.8 – 1.1	0.7 ± 0.1 0.5 – 0.8	1.0 ± 0.1 0.7 – 1.1	0.6 ± 0.0 0.5 – 0.7	1.0 ± 0.0 0.8 – 1.1	0.6 ± 0.0 0.5 – 0.7	1.2 ± 0.1 0.8 – 1.3	0.8 ± 0.0 0.6 – 0.9	1.1 ± 0.1 0.8 – 1.3	0.8 ± 0.1 0.5 – 0.9	2.0 ± 0.6 0.8 – 2.8	1.1 ± 0.2 0.6 – 1.3	5.5 ± 1.8 0.8 – 7.1
Fusion Sec. 6	4.3 ± 0.2 1.4 – 4.8	3.4 ± 0.2 1.5 – 4.0	3.9 ± 0.2 1.5 – 4.8	2.8 ± 0.2 1.1 – 3.4	3.8 ± 0.2 1.4 – 4.2	2.8 ± 0.2 1.3 – 3.7	4.4 ± 0.2 1.6 – 5.1	3.5 ± 0.2 1.4 – 9.6	4.3 ± 0.2 1.5 – 5	3.5 ± 0.2 1.2 – 6.0	4.3 ± 0.2 1.2 – 5.8	4.0 ± 0.4 1.2 – 5.5	3.8 ± 0.9 1.4 – 13.9
$\mathcal{M}$ Maps Sec. 5	8.0 ± 0.8 4.6 – 9.3	4.8 ± 0.7 2.9 – 6.0	6.1 ± 0.6 3.3 – 6.8	2.9 ± 0.3 1.8 – 3.3	6.2 ± 0.8 3.3 – 7.2	3.1 ± 0.3 1.8 – 3.5	8.3 ± 0.8 4.4 – 10.7	5.2 ± 0.4 2.9 – 6.0	6.9 ± 0.5 4.4 – 8	5.7 ± 0.8 2.9 – 7.5	7.5 ± 1.6 3.4 – 11.7	5.3 ± 1.1 2.5 – 7.7	8.4 ± 2.1 3.0 – 16.3
Full	22.1 ± 2.1 11.5 – 31.5	15.3 ± 1.5 8.1 – 22	17 ± 1.0 8.8 – 18.9	10.6 ± 0.7 5.3 – 12.5	16.7 ± 0.9 8.9 – 18.5	10 ± 0.6 5.8 – 11.6	21.9 ± 1.4 11.5 – 27.4	16.3 ± 0.9 8.2 – 23	20.4 ± 0.9 11.3 – 22.5	16.4 ± 1.2 7.8 – 21.7	21.2 ± 2.4 8.9 – 28.4	16.6 ± 1.9 7.1 – 25.6	26.6 ± 3.7 9.8 – 38.9

TABLE 5: Timings of our complete reconstruction pipeline (all timings are in milliseconds). Preprocessing module refers to the computation of all input maps (vertex, normal and curvature maps).

[22] L. Maier-Hein, A. Franz, T. dos Santos, M. Schmidt, M. Fangerau, H. Meinzer, and J. Fitzpatrick, "Convergent iterative closest-point algorithm to accomodate anisotropic and inhomogenous localization error," *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34, no. 8, pp. 1520–1532, Aug 2012.

[23] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 2100–2106.

[24] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR)*, Nara, Japan, November 2007.

[25] R. F. Salas-Moreno, B. Glocken, P. H. Kelly, and A. J. Davison, "Dense planar SLAM," in *Proc. IEEE Int. Symp. Mixed and Augmented Reality (ISMAR)*, 2014, pp. 157–164.

[26] B. Brown and S. Rusinkiewicz, "Global non-rigid alignment of 3-D scans," *ACM Trans. on Graphics (Proc. SIGGRAPH)*, vol. 26, no. 3, 2007.

[27] Q.-Y. Zhou, S. Miller, and V. Koltun, "Elastic fragments for dense scene reconstruction," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2013, pp. 473–480.

[28] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. Comp. Graph. & Interact. Techn.*, 1996, pp. 303–312.

[29] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Int. Conf. 3D Digital Imaging and Modeling (3DIM)*. IEEE, 2001, pp. 145–152.

[30] J. Serafin and G. Grisetti, "Using augmented measurements to improve the convergence of icp," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014, pp. 566–577.

[31] A. E. Johnson and M. Hebert, "Surface registration by matching oriented points," in *Proc. Intl. Conf. Recent Advances in 3-D Digital Imaging and Modeling*, ser. NRC '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 121–128. [Online]. Available: http://dl.acm.org/citation.cfm?id=523428.825373

[32] T.-W. R. Lo and J. P. Siebert, "Local feature extraction and matching on range images: 2.5D SIFT," *Computer Vision and Image Understanding*, vol. 113, no. 2009, pp. 1235–1250, 2009.

[33] M. Mittring, "Finding next gen: Cryengine 2," in *ACM SIGGRAPH 2007 courses*. ACM, 2007, pp. 97–121.

[34] E. Magid, O. Soldea, and E. Rivlin, "A comparison of gaussian and mean curvature estimation methods on triangular meshes of range image data," *Computer Vision and Image Understanding*, vol. 107, no. 3, pp. 139–159, 2007.

[35] S. Nigam and V. Agrawal, "A review: Curvature approximation on triangular meshes," *Intl. J. Engineering Science and Innovative Technology (IJESIT)*, vol. 2, no. 3, pp. 330–339, May 2013.

[36] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *Proc. Visualization (VIS)*, 2002, pp. 163–170.

[37] J. Goldfeather and V. Interrante, "A novel cubic-order algorithm for approximating principal direction vectors," *ACM Trans. on Graphics*, vol. 23, no. 1, pp. 45–63, 2004.

[38] X. Zhang, H. Li, and Z. Cheng, "Curvature estimation of 3D point cloud surfaces through the fitting of normal section curvatures," in *Proc. AsiaGraph*, 2008, pp. 72–79.

[39] Z. Cheng and X. Zhang, "Estimating differential quantities from point cloud based on a linear fitting of normal vectors," *Science in China Series F: Information Sciences*, vol. 52, no. 3, pp. 431–444, 2009. [Online]. Available: http://dx.doi.org/10.1007/s11432-009-0061-5

[40] A. Kalaiah and A. Varshney, "Differential point rendering," in *Proceedings of the 12th Eurographics Workshop on Rendering*, London, UK, Jun. 2001, pp. 139–150.

[41] M. Zwicker, H. Pfister, J. V. Baar, and M. Gross, "EWA splatting," *IEEE Trans. on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 223–238, 2002.

[42] M. Botsch, M. Spernat, and L. Kobbelt, "Phong splatting," in *Proc. Eurographics Symposium on Point-Based Graphics 2004*, Zurich, Switzerland, jun 2004, pp. 25–32.

[43] J. Serafin and G. Grisetti, "Nicp: Dense normal based point cloud registration," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 742–749.

[44] N. Fioraio, J. Taylor, A. Fitzgibbon, L. D. Stefano, and S. Izadi, "Large-scale and drift-free surface reconstruction using online subvolume registration," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2015, pp. 4475–4483.

[45] O. Wasenmüller, M. Meyer, and D. Stricker, "CoRBS: Comprehensive rgb-d benchmark for slam using kinect v2," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, March 2016. [Online]. Available: http://corbs.dfki.uni-kl.de/

[46] J. Xiao, A. Owens, and A. Torralba, "Sun3d: A database of big spaces reconstructed using sfm and object labels," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1625–1632.

[47] H. Sarbolandi, D. Lefloch, and A. Kolb, "Kinect range sensing: Structured-light versus time-of-flight kinect," *Journal of Computer Vision and Image Understanding*, vol. 13, pp. 1–20, 2015.

[48] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[49] D. Girardeau-Montaut, "CloudCompare OpenSource Project," 2013. [Online]. Available: http://www.danielgm.net/cc/

Damien Lefloch received his Bachelor and Master degree in computer science from the University of Dijon, France in 2005 and 2007, respectively. He is currently working as a scientific employee with the Computer Graphics and Multimedia Systems Group at the University of Siegen. His research interests are 3D reconstruction, image processing, depth imaging and GPU programming.

Markus Kluge received his Bachelor and Master degree in computer science from the Dortmund University of Applied Sciences and Arts, Germany, in 2012 and 2015, respectively. He is currently working as a researcher with the Computer Graphics and Multimedia Systems Group at the University of Siegen. His research interests are 3D reconstruction, depth imaging, computer graphics and motion capture.

Hamed Sarbolandi received his Bachelor degree in Electrical and Electronics Engineering from Islamic Azad University, Iran in 2009 and Master degree in the Electrical Engineering from Tampere University of Technology, Finland in 2014. He is currently working as a researcher with the Computer Graphics and Multimedia Systems Group at the University of Siegen. His research interests are 3D cameras and depth imaging.

Tim Weyrich is Professor of Visual Computing at the Department of Computer Science, University College London, and Deputy Director of the UCL Centre for Digital Humanities. Previously, he was a Postdoctoral Teaching Fellow of Princeton University in the Princeton Computer Graphics Group, a post he took after receiving his PhD from ETH Zurich, Switzerland, in 2006. His research interests are appearance modeling and fabrication, point-based graphics, 3D reconstruction, cultural heritage analysis and digital humanities.

Andreas Kolb is the head of the Computer Graphics and Multimedia Systems Group, University of Siegen, Germany. He received his Ph.D. at the University of Erlangen, Germany, in 1995. He is spokesman of the Research Training Group *Imaging New Modalities*, funded by the German Research Foundation (DFG). His research interests are computer graphics and computer vision, including particle-based simulation and visualization, lightfields, real-time processing and visualization of sensor data.