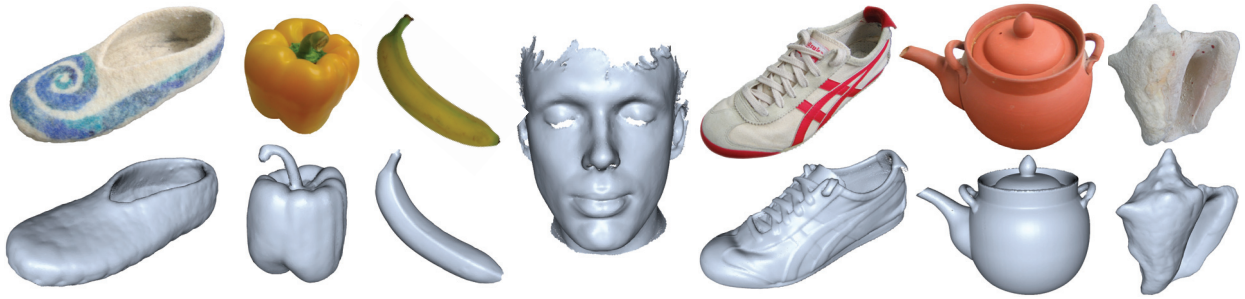
In-hand Scanning with Online Loop Closure

Thibaut Weise[1] Thomas Wismer[1] Bastian Leibe[2] Luc Van Gool[1,3]
[1]Computer Vision Laboratory ETH Zurich, Switzerland
[2] UMIC Research Centre RWTH Aachen, Germany
[3]ESAT/PSI-VISICS IBBT KU Leuven, Belgium
{weise,leibe,vangool}@vision.ee.ethz.ch thomas.wismer@awtec.ch



Abstract

We present a complete 3D in-hand scanning system that allows users to scan objects by simply turning them freely in front of a real-time 3D range scanner. The 3D object model is reconstructed online as a point cloud by registering and integrating the incoming 3D patches with the online 3D model. The accumulation of registration errors leads to the well-known loop closure problem. We address this issue already during the scanning session by distorting the object as rigidly as possible. Scanning errors are removed by explicitly handling outliers. As a result of our proposed online modeling and error handling procedure, the online model is of sufficiently high quality to serve as the final model. Thus, no additional post-processing is required which might lead to artifacts in the model reconstruction. We demonstrate our approach on several difficult real-world objects and quantitatively evaluate the resulting modeling accuracy.

1. Introduction

3D scanning and modeling of real-world objects plays an important role in many areas such as cultural heritage, architecture, and entertainment. However, 3D modeling is often a time-consuming and costly process, preventing a more wide-spread use. In archeology, for example, only the most precious objects are typically scanned instead of the bulk of the finds. In-hand scanning may alleviate that problem as it simplifies and speeds up the scanning process, thus reducing time, cost, and man-power.

3D scanning generally consists of the following steps: 1) recording a set of 3D range scans from an object, 2) aligning the individual surface patches, and 3) integrating these surfaces into one seamless 3D mesh. In case the user is not satisfied with the result, additional scans might become necessary, making this a rather cumbersome process.

With an in-hand scanning setup [23], the user simply turns the object in front of a real-time 3D scanner, and the 3D model of the object is reconstructed and displayed online. The user can thus directly control the coverage and quality of the reconstruction by turning the object appropriately until he/she is satisfied with the result, making the scanning process fast and intuitive. Current in-hand scanning approaches register the incoming 3D patches sequentially [23, 15, 30], which leads to the well-known loop closure problem: when performing a full scan around the object, the accumulation of registration errors leads to an offset at the scanning borders, resulting in visible artifacts (Fig. 1). Current approaches therefore use an additional offline optimization step to compensate for this problem and remove accumulated artifacts. As a result, however, the online model is only a preview model which may well deviate from the final offline reconstructed model. Thus, an important advantage of in-hand scanning systems, the direct and reliable feedback, is lost.

In this work we propose an in-hand scanning system that truly follows the *WYSIWYG* principle, yielding an online reconstructed model to serve as the final result. Instead of ignoring loop closure cases, our approach explicitly detects and compensates for them on-the-fly by deforming the online model appropriately. As our goal is to avoid the need for post-processing altogether, the online model needs to be accurate and robust. Our approach addresses this by handling outliers using visibility constraints. We experimentally show that our online model building method is similar in performance to offline methods and that reconstruction quality is visibly improved through online loop closing.

2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops
978-1-4244-4441-0/09/$25.00 ©2009 IEEE

Related Work. Due to the vast amount of research in 3D scanning and modeling, we only discuss the work most relevant to our in-hand scanning system. A broader perspective of the complete 3D modeling pipeline is given in [3]. For interactive applications, real-time, motion-insensitive range scanner are necessary to provide input data. Such systems have been proposed based on laser scanners [12, 27], time-of-flight cameras [21], and structured light [24, 31, 18, 29, 16]. Structured-light systems have the advantage that they provide a full 2D depth map per frame at a higher accuracy than current time-of-flight cameras.

In order to build a 3D model from depth scans, the individual 3D surface patches need to be registered. ICP [4, 6] and its variants [24, 9] are the technique of choice for this geometric registration when an approximate alignment is available. For in-hand scanning systems, this is typically the case due to the high temporal resolution of the scanner. The ICP registration can additionally be extended to texture [3, 30], which is however not yet used in our system.

Interactive in-hand 3D modeling systems have recently been demonstrated by [23, 15, 30]. In those systems, the input depth scans are registered sequentially and are integrated into a simple preview model. Outliers are handled by aggressive data pruning, but non-detected outliers will remain in the model. In order to remove the resulting accumulation errors and loop-closure artifacts, in-hand scanning systems typically perform an offline global registration optimization step when all scans have been collected [22, 14, 19]. After offline registration all input scans are integrated and converted into a seamless mesh [7, 13, 10].

In contrast, we address the loop closure problem using an idea inspired by online loop closing methods employed in robotic localization and mapping [2]. As in the standard in-hand scanning pipeline, we register depth scans sequentially to build up an online model. However, we explicitly try to detect loop closure cases and, once such a case has been found, we deform the online model such that the discontinuous surface borders fit together and the accumulated error is distributed over the entire online model. Many different methods for constraint-based shape deformation have been proposed in the past [1, 25, 26, 5]. Here, we use a graph-based space deformation method similar to [26] due to its efficiency and as-rigid-as-possible deformation.

[27, 8] also propose a system that returns the online model as the final result. Their approach employs implicit surfaces for the complete modeling pipeline. In contrast to our method, however, they do not attempt to solve the loop closure problem and do not explicitly handle outliers.

2. Online Model Building

In this paper, we propose an online model building approach that reconstructs a 3D model of an object from a sequence of depth scans. This task poses several requirements

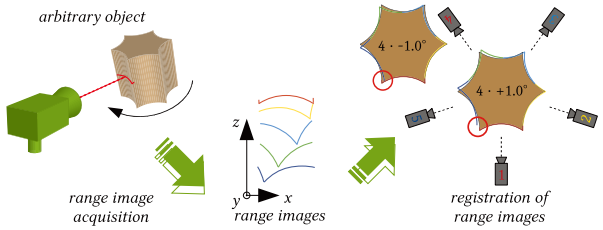


Figure 1. Visualization of the loop closure problem: five range scans of a rotating gear wheel are registered. A pairwise alignment error of only $\pm 1^\circ$ results in a considerable surface discrepancy where the model actually should close (indicated by red circles).

on the underlying representation. As the input data is noisy, several measurements need to be integrated for each surface part for accurate reconstruction. Since the input may also contain outliers, we need to be able to to differentiate between true surface points and erroneous parts. Last but not least, the representation needs to be compact and easy to modify in order to facilitate online loop closure.

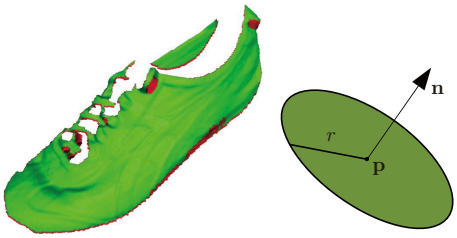
We address those requirements by quantizing the surface and representing the model as a set of small oriented discs or *surface elements* (*surfels*) whose size is directly dependent on the scanning resolution. This quantized representation allows us to easily integrate multiple depth measurements and to update the surface estimate when additional measurements become available. In addition, we collect visibility statistics for each surfel to determine from which orientations it has been observed. This allows us to differentiate between likely true surface points which have been observed from many different directions and possible outliers for which this is not yet the case.

In the following, we describe the input data our approach is based on (Sec. 2.1) and the basic surfel representation (Sec. 2.2). In order to bring the online surfel model into correspondence with the current scan, we perform rigid registration with a fast ICP algorithm (Sec. 2.3). After successful registration, the new scan is then integrated with the surfel model, as described in Sec. 2.4. Sec. 3 then presents our extension to explicitly handle the loop closure problem.

2.1. Data Acquisition

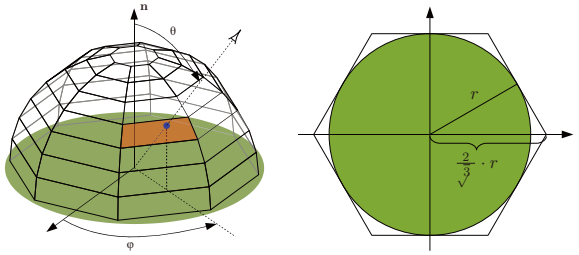
The in-hand scanning system is built on top of our real-time structured-light range sensor [29], running at 30 fps. This setup delivers one depth map D_t (VGA resolution) per frame t, containing for each pixel u_k a depth value d_k^* (in mm) corresponding to a 3D position $\mathbf{p}_k^*$.

Input Data Processing. Morphological operators are used to remove small isolated geometry patches that are likely outliers. Normals $\mathbf{n}_k^*$ are calculated using the method described in [20]. In order to penalize potentially inaccurate vertices, each depth value is assigned an input confidence value $c_k^* \in [0, 1]$, which is initialized to $c_k^* = 0$ at a depth map discontinuity, otherwise $c_k^* = 1$. The confidence values are then spread out by a diffusion process [28]. Fig. 2(a) depicts an input scan colored according to confidence.



(a) (b)

Figure 2. a) Input depth map of a shoe. Green represents high data confidence, red represents low data confidence. b) Model surfel: a surfel is described by its position $\mathbf{p}$, normal $\mathbf{n}$ and radius r.



(a) (b)

Figure 3. a) Surfel visibility confidence: a pair of a polar angle θ and an azimuth angle φ characterizes a view direction for each surfel. The bookkeeping of the view direction from which the surfel has already been observed is done with a binary histogram (64 bins). b) Hexagonal surfel approximation for efficient rendering.

2.2. Surfel Representation

We adopt an explicit surface representation similar to [11] by representing the model surface M_t as set of N surfels (see Fig. 2(b)), each having a position $\mathbf{p}_i$, normal vector $\mathbf{n}_i$, radius r_i and visibility confidence v_i. The use of surfels instead of a triangle mesh has the crucial advantage that the unstructured set of surfels can easily be kept consistent throughout any modifications. In contrast, for a triangle mesh, considerable efforts are needed to guarantee the integrity and quality of the mesh topology after adding, updating, or removing any vertices.

Visibility Confidence. In order to estimate its reliability, each surfel is assigned a visibility confidence $v_i \in [0, 64]$. A surfel is assumed to be correct if its position is confirmed by several observations from different directions. This is unlikely to occur for wrong surfels due to outliers. When a new surfel is created, a spherical coordinate system is generated with the estimated surfel normal as the first base vector and an arbitrary vector lying in the surfel plane as second base vector. View directions are then characterized by polar and azimuth angles. The view directions a surfel has been seen from are recorded in a two-dimensional binary histogram (see Fig. 3(a)). For our setup, a surfel has high visibility confidence if it has been observed in at least 6 bins.

Visualization. A surfel is efficiently visualized as a hexagon (Fig. 3(b)) by rendering four triangles, which are directly calculated on the GPU from $\mathbf{p}_i$, $\mathbf{n}_i$ and r_i.

2.3. Registration

We apply fast ICP [24] to align the online model M_t with the input data of the current depth map D_t. All surfels of M_t are iteratively projected into D_t and the optimal rigid transformation $[\mathbf{R}_t, \mathbf{t}_t]$ is calculated by minimizing the point-surface distance:

$$E_{p2s} = \sum_{i=1}^{N} \| w_i \, \mathbf{n}_i^{*\top} \left(\mathbf{R}_t \mathbf{p}_i + \mathbf{t}_t - \mathbf{p}_i^* \right) \|_2^2 \qquad (1)$$

where $\mathbf{p}_i^*$ is the corresponding point found by projection; $\mathbf{n}_i^*$ is the associated normal; and w_i is a correspondence-specific weighting term based on normal compatibility and dynamic distance threshold [15]. For point and normal look-up in the depth map we use bilinear interpolation.

Registration Failure. In order to check for registration failure, a virtual depth map $\hat{D}_t$ is generated of the current model M_t using the rigid transformation $[\mathbf{R}_t, \mathbf{t}_t]$ and the intrinsics of the range sensor. Each pixel u_k is checked whether it is an inlier or outlier based on a maximum distance threshold on the absolute depth difference between virtual rendering and input scan ($2mm$ in our system setup). If the total ratio $\frac{outliers}{inliers+outliers} < t_{reg} = 0.05$, the registration is successful. We do not differentiate between scan occluding model, or model occluding scan. While the first may be valid, we still count these as outliers for robustness. Due to the interactive nature of the system, the user can rotate the object such that registration succeeds again.

2.4. Integration

After successful registration, the current depthmap D_t is used to update the online model M_t. This is done using three basic operations: *surfel update*, *surfel addition* and *surfel removal*. Surfels that are in correspondence with the input scan are updated by integrating the depth measurements. New surfels are created for parts of the scan that are not explained by the current model. Surfels that are not confident and in conflict with the current scan are removed.

Surfel Update. For each surfel, $\mathbf{p}_i$ and $\mathbf{n}_i$ are transformed using the current rigid transformation:

$$\mathbf{p}_i' = \mathbf{R}_t \mathbf{p}_i + \mathbf{t}_t \qquad (2)$$

and accordingly for $\mathbf{n}_i'$. The z component $d_i' = \mathbf{p}_i'(z)$ is the depth of the surfel. It is compared against the depth d_i^*, normal $\mathbf{n}_i^*$ and confidence c_i^* of the input scan D_t, as well as the depth $\hat{d}_i$ and $\hat{v}_i$ from the virtual depth map $\hat{D}_t$. There are 4 different update cases:

(1) If the normal $\mathbf{n}_i'$ deviates by more than $t_{away} = 80°$ from the principal axis of the range sensor or if the input scan has low confidence ($c_i^* < c_{min} = 0.8$), the surfel is not updated, as this indicates that it is not visible or not in correspondence with the input scan.

(2) $|d_i' - d_i^*| < d_{max}$: We assume that surfel and input scan are in correspondence if their distance is less than a

threshold $d_{max} = 5mm$. In this case, the surfel position and normal are updated with the new measurements by computing a running average.

(3) $d'_i - d^*_i < -d_{max}$: The input scan is behind the model surfel, which results in a visibility conflict. In case the surfel has a low visibility confidence, the surfel is assumed to be an outlier and is replaced by a new surfel at the scanner depth. Otherwise, the scan is considered the outlier.

(4) $d'_i - d^*_i > d_{max}$: The input scan occludes the model surfel. By itself this is not necessarily a visibility conflict, as the scan might observe a different surface. We try to detect this by checking for self-occlusions using the model acquired so far. If no self-occlusion is found, this is treated as visibility conflict and is handled as in case 3. However, given a self-occlusion of the model, $d'_i - \hat{d}_i > d_{max}$, the model surfel is removed if the surfel has low confidence, is occluded by high-confidence surfels ($\hat{v}_i > v_{min}$) that are in correspondence, and if the surfel should be seen from this viewpoint (based on the visibility confidence histogram). This test may be slightly inaccurate, but it is an efficient strategy for removing outliers located inside the object.

Surfel Addition. After all surfels have been updated, surfels are added in those parts where the scanner depth map is not covered by model surfels. New surfels are only introduced where the input data is valid and confident ($c^*_k \geq c_{min}$). After each surfel addition and update, the surfel visibility confidence histogram is updated accordingly.

Surfel Removal. Model surfels that are not confident are removed if they are in conflict with the input scan, as explained above. During scanning, many correct surfels, as well as some possible outliers, are added. Correct surfels are re-observed from many view directions and thus reach a high confidence value. In contrast, outliers are unlikely to be seen again and will be deleted if a conflict appears. In order to keep the model lightweight, we additionally apply an erosive strategy: every surfel that has not been updated within the last $t_{starve} = 30$ frames and has not yet reached a minimum confidence value $v_{starve} = 3$ is removed.

Surface Growing. Surfels in cavities will only be visible from few view directions and never attain a high visibility confidence. Hence, the erosive strategy will always remove these surfels. Thus, a second strategy is employed once the coarse object structure has been successfully acquired (determined by the user) and no more unsolved loop closure problems remain (see Sec. 3). The model is cleaned up, *i.e.* all surfels having a confidence below v_{min} are deleted, and the lost parts are re-scanned with surfel removal switched off. In order to avoid outliers, surfels are only added if there are already some model surfels close-by with similar orientations. This strategy is referred to as *surface growing* [11]. In detail, a potential surfel is only added, if it is at the border of a model surfel and if their normals do not deviate by more than $45°$.

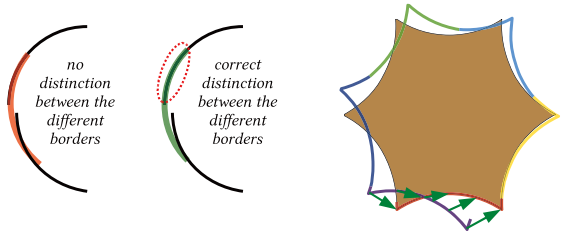


Figure 4. Solution to the loop closure problem: When a loop closure case is detected, our algorithm proceeds by only registering and integrating new input data with one of the *'meeting'* borders. Once the overlap of the involved surface patches is sufficiently large, the model is closed by an as-rigid-as-possible deformation.

Radius Estimation. Instead of fixing the radius of each surfel to some constant value, we adapt the surfel radius according to the theoretical accuracy limit of the input device, as the reconstructed model cannot be more accurate than the resolution of the scanner. The radius for each surfel is estimated conservatively to cover one pixel of the input data using the following equation:

$$r_i = \frac{1}{\sqrt{2}} \frac{d'_i / f}{\mathbf{n}'_i(z)} \tag{3}$$

where f is the focal length of the range sensor. The surfel radius is only updated if this result is smaller than the current estimate. With this update strategy, it is possible to increase the level of detail in the model by bringing the object closer to the camera.

2.5. In-hand Scanning

The interface of our in-hand scanning system is similar to [23]. The user holds the object in front of the scanner, and the reconstructed online model is displayed in real-time. The scan is overlayed over the model to facilitate reinitialization. The model surfels are colored according to the visibility confidence such that the user can see which parts are not yet confident. Depending on the application, the final model can be converted to a watertight triangle mesh after the scanning session using the freely available software from [17]. This step also fills the parts that cannot be acquired due to limitations of the scanner hardware.

3. Online Loop Closure

The online model building method from the previous section can be used for in-hand scanning as it stands. However, as soon as the scanned object has been turned for almost a full rotation, the growing model surface reaches one of its older borders from the other side. In such a case, the accumulated registration error may have grown to a degree that the growing surface does not coincide with its own border anymore and no proper alignment can be found. This is referred to as the *loop closure problem*, visualized in Fig. 1.

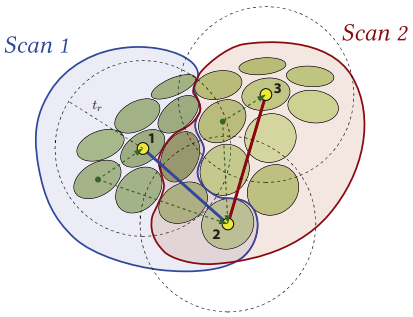


Figure 5. Topology graph. In this example, three surfels act as topology graph nodes, indicated by the yellow bullets. Node 1 and 2 are connected, since both have been observed in scan 1. The same holds for nodes 2 and 3 in scan 2. However, node 1 and 3 have never been seen together, so they are not connected. Every surfel is properly represented since all surfels are attached to at least one node (indicated by green dashed arrows).

In the following, we present an algorithmic extension that enables our system to automatically detect and close loops, thus removing discontinuities at the model boundaries.

Fig. 4 visualizes the main idea of our approach. We solve the loop closure problem by discriminating between the different model surface borders. When a loop closure case is detected, the input data is only registered and integrated with one of the borders. Once the overlap of the two involved surface patches is sufficiently large, the input data is registered to all borders individually, and the loop is closed by an as-rigid-as-possible deformation, bending the overlapping surface parts onto each other.

At the core of our approach lies a sparse *topology graph*. This graph fulfills two important functions. It helps discriminate between the different borders and it contributes connectivity information that is used for the actual deformation. In the following, we describe the details of this representation and how it is used for expressing and updating connectivity information (Sec. 3.1). Sec. 3.2 then shows how the basic registration and integration methods are adapted to include the topology graph, and Sec. 3.3 finally presents the as-rigid-as-possible deformation for loop closure.

3.1. Topology Graph

Whenever two surfels have been seen together in a scan, the scan data places a constraint on the relative positions of these two surfels. The topology graph is used to encode those constraints in a sparse manner. It is laid over the surface consisting of thousands of surfels and is represented by a set of S nodes (a subset of the surfels). Two nodes are connected by an undirected edge whenever they have been seen together in any of the original scans (see Fig. 5).

Each surfel (and thus also each node) stores a list of nodes it has been observed with. We say it is *'attached'* to them. Additionally, each surfel stores which attached nodes are closer than a given threshold t_r. Trivially, this set of nodes to which a surfel is *'attached in range'* is a subset

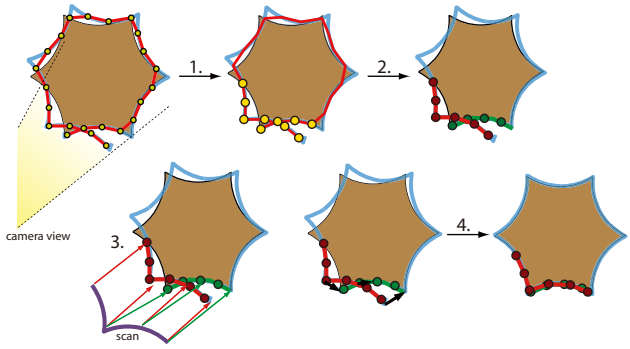


Figure 6. Steps of the loop closure procedure: 1) Determine the currently visible nodes. 2) Intersect the visible nodes with the topology graph and decompose into connected components. 3) Register each component individually to the input scan. 4) Deform the model surface if components have sufficient overlap.

of the nodes to which it is attached. A surfel is properly represented in the topology graph if it has at least one node *attached in range*. The threshold t_r implicitly defines the density of the topology graph and is set in our experiments to $t_r = 15mm$. After every modification of the surfel cloud, the topology graph has to be updated as well. The required tasks for the possible model modifications are:

Surfel Addition. The new surfel is attached to all currently visible nodes. Furthermore, it has to be determined which of those nodes are closer than t_r to the surfel. In case no node is closer than t_r, the surfel is added as a new node.

Node Addition. The new node is added and all visible nodes, as well as all visible surfels, are attached. All surfels closer than t_r are additionally *attached in range*. Node addition is dependent on the surfel processing order.

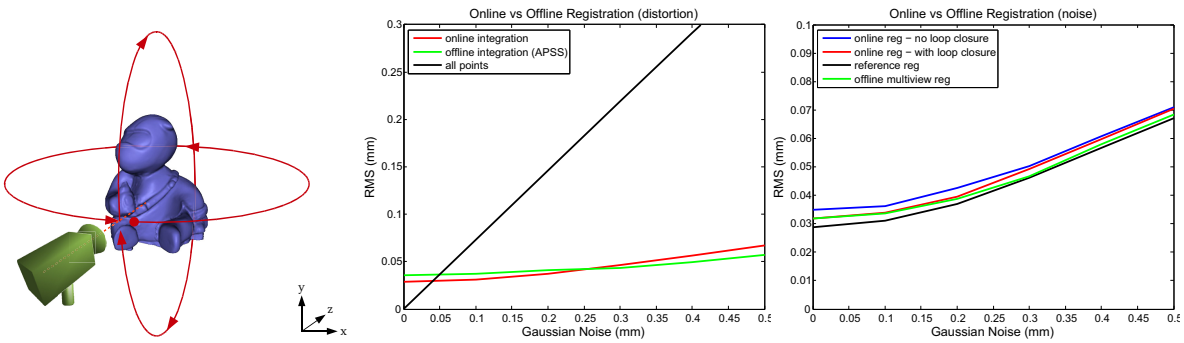
Node Deletion. A node is deleted when the corresponding surfel is removed. Prior to surfel deletion, all nodes and surfels are disconnected from that node.

Surfel/Node Update. Whenever surfel properties (position, normal, or radius) are updated, the surfel is attached to all currently visible nodes. Whenever a node needs to be deleted, it might happen that there are surfels that are attached to that single node only and will not be properly represented any more after the node deletion. Such surfels with no attachments are called *loose*. As soon as a loose surfel can be updated (thus it is visible once more), the surfel is integrated again into the topology graph.

The complexity for the topology graph update is $O(NS)$, though $S << N$ and all operations can be efficiently implemented using bitsets.

3.2. Component Registration & Integration

The online model building method needs to be adapted to properly include the topology graph. At each frame, we determine the currently visible nodes (subset of visible surfels) by checking for each node if its distance to the scan is below $d_{vis} = 5mm$. The nodes are divided into a



(a) (b) (c)

Figure 7. a) Synthetic rendering of *boris* model. b) Comparison of integration algorithms: online integration performs similarly to offline integration [10]. Both introduce a small error for 0 Gaussian noise compared to the baseline (using all input scan vertices), but are successful in handling the noise of the input scans. c) Comparison of registration algorithms depending on the amount of Gaussian noise on the simulated input scan. All methods perform similarly, though online loop closure is always better than no loop closure.

set of L connected components by intersecting the set of visible nodes with the topology graph (see Fig. 6). Only the largest connected component (main component) is used for registration and integration, *i.e.* only surfels are taken into account that are attached to any of the nodes in this main component. Thus, when looping around the object, the other surface border will create a second unconnected component. Surfels belonging to that other component are ignored. This generalizes to any number of components. Thus, on its own, the main component will grow over the other border surfaces, creating the necessary overlap for the model deformation described in the next section.

3.3. As-rigid-as-possible Deformation

The full procedure for loop closure is visualized in Fig. 6. Once the overlap between the model components and the input scan has become large enough (each component explains more than 70% of the depth scan), the overlapping borders are combined. This is done by registering each of the L components individually to the current input scan. The resulting constraints on each surfel are then used to deform the complete model as-rigidly-as-possible in order to connect the separate components. Once the model is deformed, the components are joined.

Deformation Using the topology graph we employ an as-rigid-as-possible deformation similar to the method described by [26]. This space deformation method is general enough that it can be applied to any object, while providing natural feature preservation and efficiency. A space deformation is represented by a collection of rigid transformations organized in the topology graph structure.

Each graph node $\mathbf{g}_j$ is assigned a transformation $[\mathbf{R}_j, \mathbf{t}_j]$, inducing a transformation of the nearby space:

$$\tilde{\mathbf{p}} = \mathbf{R}_j (\mathbf{p} - \mathbf{g}_j) + \mathbf{g}_j + \mathbf{t}_j. \quad (4)$$

In the space between graph nodes, the influence of individual graph nodes is smoothly blended, similar to skeleton-subspace deformation from character animation. The deformed position $\tilde{\mathbf{p}}_i$ of each surfel i is calculated as weighted sums of its counterparts after application of the deformation graph transformations:

$$\tilde{\mathbf{p}}_i = \sum_{j=1}^{S} w_j(\mathbf{p}_i) [\mathbf{R}_j (\mathbf{p}_i - \mathbf{g}_j) + \mathbf{g}_j + \mathbf{t}_j] \quad (5)$$

where the weights $w_j(\mathbf{p}_i)$ depend on the distance of the surfel to the k-nearest nodes. The weights for each surfel are precomputed according to

$$w_j(\mathbf{p}_i) = (1 - \|\mathbf{p}_i - \mathbf{g}_j\|_2 / d_{max})^2 \quad (6)$$

and then normalized to sum to one. d_{max} is the distance to the $k+1$-nearest node. The experiments in [26] indicate that setting $k=4$ is sufficient.

The unknowns of the optimization problem are the rotation matrices and translation vectors of each node. Rigid registration of each component yields a rotation matrix $\mathbf{R}^\ell$ and translation vector $\mathbf{t}^\ell$. This rigid transformation gives a positional constraint for each visible surfel i of subpart ℓ on how it is registered to the input data:

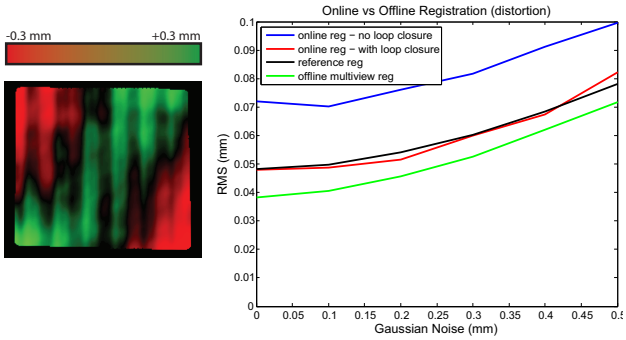
$$\mathbf{p}_i^\ell = \mathbf{R}^\ell \mathbf{p}_i + \mathbf{t}^\ell \quad (7)$$

The deformation is formulated as an optimization problem consisting of a positional energy term E_{pos} and a regularization term E_{reg}:

$$E_{pos} = \sum_{\ell=1}^{L} \sum_{i \in H(\ell)} \|\mathbf{p}_i^\ell - \tilde{\mathbf{p}}_i\|_2^2 \quad (8)$$

$$E_{reg} = \sum_{j=1}^{S} \sum_{n \in \Omega(j)} \|\mathbf{R}_j (\mathbf{g}_n - \mathbf{g}_j) + \mathbf{g}_j + \mathbf{t}_j - (\mathbf{g}_n + \mathbf{t}_n)\|_2^2$$

where L is the number of components, $H(\ell)$ is the set of visible surfels in component ℓ, and $\Omega(j)$ is the set of connected nodes of node j. The regularization term makes the deformation smooth and as-rigid-as-possible by penalizing the difference between the actual transformation of a node and the transformation defined by a connected node. The optimal registration is found by minimizing a weighted sum of the energy terms:



(a) (b)

Figure 8. a) Measured depth distortion of our range scanner. b) Comparison of registration algorithms when scanner specific depth distortion is applied additionally to the noise. Online loop closure significantly improves the results.

$$\min_{\mathbf{R}_1,\mathbf{t}_1 \ldots \mathbf{R}_S,\mathbf{t}_S} E_{pos} + w_{reg} E_{reg} \tag{9}$$

where $w_{reg} = 0.1$ in our system. Eq. 9 is non-linear in terms of the $7S$ unknowns that define the rotation matrices and translation vectors of each node. Note that we represent $\mathbf{R}_j$ using quaternions. The non-linear optimization is very sparse and can be solved using Gauss-Newton iterations in conjunction with an efficient sparse linear solver.Once the optimal deformation is found, the surfels are deformed and both surfels and topology graph are updated.

4. Experimental Results

Some objects reconstructed with our in-hand scanning system are shown on the front page of this paper (rendered surfel clouds). All parameters have been set experimentally and left unchanged throughout all experiments. The in-hand scanning system in action is best seen in the supplementary video. Most of the presented algorithms have been implemented on the GPU. For a typical hand-sized object consisting of about $100K$ surfels, the system runs at 20 fps. Online loop closure takes 2-3 seconds, but is typically only performed once or twice per scanning session and thus does not disturb the interaction.

4.1. Experiments on Synthetic Data

In order to quantitatively evaluate our in-hand scanning system, we performed experiments using synthetic data. For this purpose, we created a virtual scanning sequence of the *boris* object from [30], rotating once around the x-axis and once around the y-axis (see Fig. 7(a)). The sequence consists of 125 depth scans. For evaluation, we register the reconstructed point clouds with the original model using ICP and calculate the RMS error in mm between reconstruction and original. The size of *boris* is approximately $100mm$ in all dimensions. The virtual distance from the range sensor is $1000mm$ with approximately $1mm$ lateral resolution, resulting in approximately $8K$ vertices per scan.

Integration. In order to evaluate the quality of the integration, we disable registration and use the reference transformations given by the synthetic rendering instead. As integration methods we compare simply taking all vertices from the input scans, the offline method from [10] (which produced the best results of the offline integration methods we tested), and our proposed online modeling method. Fig. 7(b) shows the quality of the integration procedures for different noise levels of the input data. As expected, when simply using all points the error increases linearly with the noise level. Both the online and the offline method perform similarly, introducing a slight error ($0.035mm$), but handling noise well. Thus, using the online model does not degrade the quality of the integration here. Note that the noise level of our scanner is typically in the range of $0.2-0.4mm$.

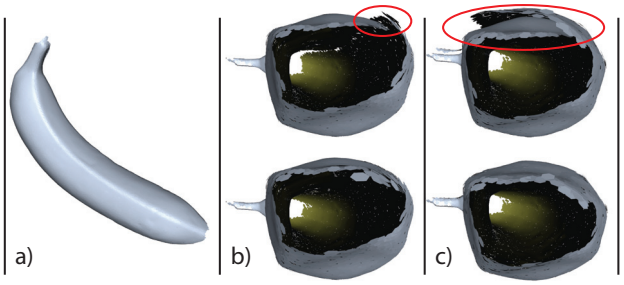


Figure 9. a) Reconstructed banana. b) Registration without loop closure (banana cross-section): the loop closure error (top) is only compensated for locally (bottom) by *averaging out* the error at the border overlap. c) Registration with loop closure: the loop closure error (top) is compensated for globally, spreading the accumulated error over the full model. Note in particular the distortion of the left and bottom surface parts.

Registration. The registration quality is evaluated by comparing the reconstruction using 1) the reference transformations, 2) sequential registration with offline multiview optimization [23], 3) online registration without loop closure and 4) online registration with loop closure. All methods are used in conjunction with our proposed integration method.

In a first experiment, we only add Gaussian noise to the input data. Fig. 7(c) shows that all methods perform similarly and that using online loop closure does not degrade the results. Note that white noise on its own seems to only introduce a minimal loop closure discontinuity.

In a second experiment, we also model the distortion of our scanning setup in addition to white noise to better reflect real-world conditions. For this, we recorded a planar surface with our scanner and fit a plane to the smoothed data. The deviation of the smoothed data to the planar surface is used to generate a distortion map (see Fig. 8(a)), modeling pattern artifacts and calibration errors of our setup (We verified that the error does not come from the planar surface itself). Fig. 8(b) shows that using online loop closure significantly improves the reconstruction results compared to ignoring the loop closure problem. Offline optimization is still better than online optimization, but cannot be seamlessly embedded into the interaction.

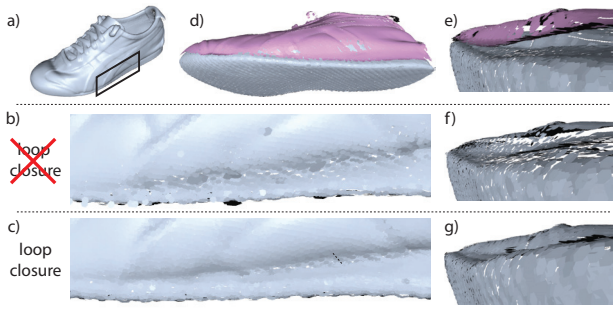


Figure 10. a) Reconstructed shoe. b) Close-up view without loop closure. c) Close-up view with loop closure. d) View of the two unconnected surface parts leading to the loop closure problem. e) The surface discontinuity is prominent in the zoomed view. f) No loop closure *averages out* the discontinuity resulting in a noisy and distorted reconstructed geometry. g) Online loop closure correctly compensates for the discontinuity.

4.2. Experiments on Real Data

Fig 9 and Fig 10 demonstrate the benefit of using online loop closure on real data. Looking at the cross-section of the reconstructed banana in Fig 9, we can observe that without loop closure, the accumulated registration error is only compensated for locally by the integration procedure, resulting in an artificially strong bending of the surface. In comparison, using loop closure, the error is spread globally by distorting the entire model. The effect of locally *averaging out* the error and the resulting artifacts are shown in the reconstructed shoe in Fig 10. The loop closure problem appears at the border of the sole, where the meeting surface boundaries generate a discontinuity. With loop closure, the surface boundaries are correctly *stitched*, whereas without loop closure the two surfaces *average out*, resulting in a noisy and distorted geometry.

5. Conclusion

We have presented a complete in-hand scanning system that returns an online reconstructed 3D model of sufficient accuracy to serve as the final result. The loop closure problem that arises during scanning is detected and compensated for on-the-fly by deforming the model appropriately. Thus, no additional post-processing is required.

The online loop closure distributes the accumulated registration error evenly by distorting the model. It assumes, however, that all registration errors are fairly small, and it cannot compensate for single completely erroneous registrations. Similarly, the accumulated error needs to be small, as the overlap between surface borders is measured directly in the input scans. However, this is no problem in our setup, as our range scanner generally provides sufficient accuracy.

The in-hand scanning system currently only uses geometry. In future work, we plan to also incorporate texture into the online model and to extend the method to allow for deforming objects, such as for the scanning of a human hand.

References

[1] B. Allen, B. Curless, and Z. Popović. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Trans. Graph.*, 22:587–594, 2003.

[2] A. Angeli, S. Doncieux, J.-A. Meyer, and D. Filliat. Real-time visual loop-closure detection. In *ICRA '08*, 2008.

[3] F. Bernardini and H. Rushmeier. The 3D model acquisition pipeline. *Comp. Graph. Forum*, 21:149–172, 2002.

[4] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *PAMI*, 14:239–258, 1992.

[5] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *Trans. Vis. Comp. Graph.*, 14:213–230, 2008.

[6] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *IVC*, 10:145–155, 1992.

[7] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH '96*, 1996.

[8] J.-D. Deschenes, P. Lambert, and P. Hebert. Interactive modeling with automatic online compression. In *3DPVT '06*, 2006.

[9] N. Gelfand, S. Rusinkiewicz, L. Ikemoto, and M. Levoy. Geometrically stable sampling for the ICP algorithm. In *3DIM*, 2003.

[10] G. Guennebaud and M. Gross. Algebraic point set surfaces. *ACM Trans. Graph.*, 26:23, 2007.

[11] M. Habbecke and L. Kobbelt. A surface-growing approach to multi-view stereo reconstruction. In *CVPR '07*, 2007.

[12] A. Hilton and J. Illingworth. Geometric fusion for a hand-held 3d sensor. *MVA*, 12:44 – 51, 2000.

[13] A. Hilton, A. Stoddart, J. Illingworth, and T. Windeatt. Implicit surface-based geometric fusion. *CVIU*, 69(3):273 – 291, 1998.

[14] D. F. Huber and M. Hebert. Fully automatic registration of multiple 3D data sets. *IVC*, 21:637–650, 2003.

[15] T. Jaeggli, T. Koninckx, and L. V. Gool. Online 3d acquisition and model integration. In *Pro. Cam.*, 2003.

[16] H. Kawasaki, R. Furukawa, R. Sagawa, and Y. Yagi. Dynamic scene shape reconstruction using a single structured light pattern. In *CVPR '08*, 2008.

[17] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *SGP '06*, 2006.

[18] T. P. Koninckx, T. Jaeggli, and L. V. Gool. Adaptive scanning for online 3d model acquisition. In *CVPRW '04*, 2004.

[19] S. Krishnan, P. Y. Lee, J. B. Moore, and S. Venkatasubramanian. Global registration of multiple 3D point sets via optimization-on-a-manifold. In *SGP '05*, 2005.

[20] N. Max. Weights for computing vertex normals from facet normals. *J. Graph. Tools*, 4:1–6, 1999.

[21] MESA. http://www.mesa-imaging.ch/.

[22] K. Pulli. Multiview registration for large data sets. In *3DIM*, 1999.

[23] S. Rusinkiewicz, O. A. Hall-Holt, and M. Levoy. Real-time 3D model acquisition. *ACM Trans. Graph.*, 21:438–446, 2002.

[24] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *3DIM '01*, 2001.

[25] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *SGP '07*, 2007.

[26] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26:80, 2007.

[27] D. Tubic, P. Hébert, J.-D. Deschênes, and D. Laurendeau. A unified representation for interactive 3D modeling. In *3DPVT '04*, 2004.

[28] J. Weickert. *Anisotropic Diffusion in Image Processing*. Teubner, 1998.

[29] T. Weise, B. Leibe, and L. V. Gool. Fast 3d scanning with automatic motion compensation. In *CVPR '08*, 2007.

[30] T. Weise, B. Leibe, and L. V. Gool. Accurate and robust registration for in-hand modeling. In *CVPR '08*, 2008.

[31] S. Zhang and P. Huang. High-resolution, real-time 3d shape acquisition. In *CVPRW '04*, 2004.