Computer Vision and Image Understanding 115 (2011) 635–648

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Online loop closure for real-time interactive 3D scanning

Thibaut Weise [a,*], Thomas Wismer [b], Bastian Leibe [c], Luc Van Gool [b,d]

[a] Laboratoire d'informatique graphique et géométrique, EPFL Lausanne, Switzerland
[b] Computer Vision Laboratory, ETH Zurich, Switzerland
[c] UMIC Research Centre, RWTH Aachen, Germany
[d] ESAT/PSI-VISICS IBBT, KU Leuven, Belgium

ARTICLE INFO

Article history:
Received 15 February 2010
Accepted 1 November 2010
Available online 31 December 2010

Keywords:
3D modeling
3D scanning
Registration
Integration
Loop closure

ABSTRACT

We present a real-time interactive 3D scanning system that allows users to scan complete object geometry by turning the object around in front of a real-time 3D range scanner. The incoming 3D surface patches are registered and integrated into an online 3D point cloud. In contrast to previous systems the online reconstructed 3D model also serves as final result. Registration error accumulation which leads to the well-known loop closure problem is addressed already during the scanning session by distorting the object as rigidly as possible. Scanning errors are removed by explicitly handling outliers based on visibility constraints. Thus, no additional post-processing is required which otherwise might lead to artifacts in the model reconstruction. Both geometry and texture are used for registration which allows for a wide range of objects with different geometric and photometric properties to be scanned. We show the results of our modeling approach on several difficult real-world objects. Qualitative and quantitative results are given for both synthetic and real data demonstrating the importance of online loop closure and outlier handling for model reconstruction. We show that our real-time scanning system has comparable accuracy to offline methods with the additional benefit of immediate feedback and results.

1. Introduction

3D scanning and modeling of real-world objects is attractive for a variety of applications, including entertainment, engineering, cultural heritage, and architecture. However, 3D modeling is often a time-consuming and costly process, preventing a more widespread use. In archeology, for example, only the most precious objects are typically scanned instead of the bulk of the finds. Interactive modeling may alleviate that problem as it simplifies and speeds up the scanning process, thus reducing time, cost, and man-power.

Most scanning systems capture 3D surface patches from different viewpoints, e.g. using a turntable, and the different parts are stitched together in an offline process to form a coherent mesh. When scans are only registered and integrated offline, such a process risks leaving holes in the reconstruction in places where the scanner could not reach from the predetermined viewpoints. Moreover, the offline processing means that those holes may only be detected much later. Filling in the missing data then requires another scanning session, which is a tedious process and which may in some cases even be impossible, since the object is no longer accessible.

Interactive modeling is an appealing concept, since it circumvents this problem. With an interactive scanning setup [35], the user simply turns the object in front of a real-time 3D scanner (or alternatively the scanner is turned around the object), 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. Commercially available scanning systems such as [48] use optical targets or robotic arms to determine the relative position between individual input scans. In contrast, current interactive scanning approaches that do not use additional markers register the incoming 3D patches sequentially [35,25,46], 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 interactive scanning systems, the direct and reliable feedback, is lost.

In this work we propose an interactive scanning system that truly follows the *WYSIWYG* principle, yielding an online reconstructed model to serve as the final result. Instead of ignoring loop

* Corresponding author.
E-mail address: thibaut.weise@epfl.ch (T. Weise).

1077-3142/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2010.11.023

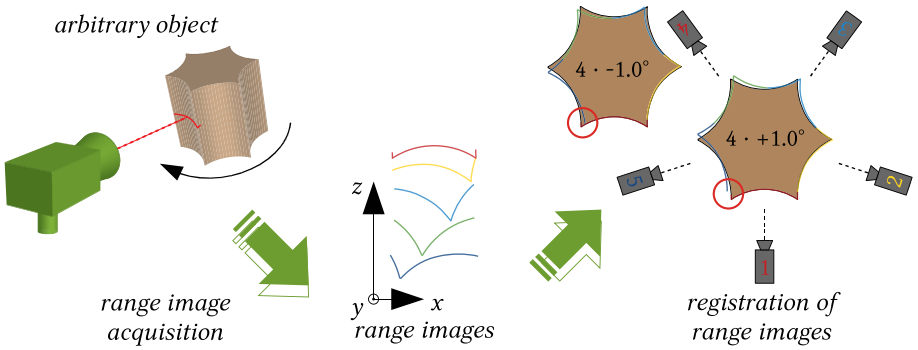


Fig. 1. Visualization of the loop closure problem: five range scans of a rotating gear wheel are registered. A pairwise alignment error of only ±1° results in a considerable surface discrepancy where the model actually should close (indicated by red circles). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

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. The interactive modeling should be applicable to a wide range of objects, yet geometric registration may fail for symmetric objects due to ambiguities. While our system does not reconstruct the texture of the final model, we exploit the additional color image that is acquired by current range scanners. By incorporating texture features into the registration we are able to break the geometric symmetries in most cases. We experimentally show that our online model building method is similar in accuracy to offline methods and that reconstruction quality is visibly improved through online loop closing.

2. Related work

A comprehensive overview on methods required for 3D modeling is given by Bernardini and Rushmeier [4], including registration methods based both on geometry and texture. For interactive applications, real-time, motion-insensitive range scanners are necessary to provide input data. Such systems have been proposed based on laser scanners [20,6,42], time-of-flight cameras [31,24,8], passive stereo [11], and structured light [36,49,29,45,27]. Structured-light systems have the advantage that they provide a dense 2D depth map per frame with higher accuracy than current time-of-flight and stereo cameras.

In order to build a 3D model from depth scans, the individual 3D surface patches need to be registered. A comprehensive overview about methods that estimate a coarse registration between two surface patches is given by Campbell and Flynn [9]. ICP [5,10] and its variants [36,14] are the technique of choice when an approximate alignment is already available. For interactive scanning systems, this is typically the case due to the high temporal resolution of the scanner. Texture may be used as an additional cue in ICP registration by either modifying the closest point search to include color [26,37], by adding feature correspondences [16,4,38,46], or by using optical flow [44,46].

Interactive 3D modeling systems have been demonstrated by Gionis and co-workers [15,20,35,25,42,13,46,33]. (Commercialized) systems by [15,20] consist of a hand-held scan device whose position is tracked by optical or mechanical means, and therefore the scan data is already in correct alignment and no registration is required. The main disadvantages of these systems are typically their high cost and bulkiness.

An interactive scanning system that returns the online model as the final result was proposed by Tubic and co-workers [42,13]. 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.

An interactive modeling approach that is solely based on video was introduced by Pan et al. [33]. For this purpose an online structure from motion algorithm is used for camera pose tracking, and the geometry is reconstructed using tetrahedron carving on the Delaunay tetrahedralization of the point cloud. No special hardware besides a simple webcam is required, yet the reconstruction quality is limited and requires many texture features that cover the entire object.

In the systems of [35,25,46], 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, interactive scanning systems typically perform an offline global registration optimization step when all scans have been collected [34,23,30]. After offline registration all input scans are integrated and converted into a seamless mesh [12,21,17].

In contrast, we address the loop closure problem using an idea inspired by online methods employed in robotic localization and mapping [2]. As in the standard interactive 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 [1,39,41,7]. Here, we use a graph-based space deformation method similar to [41] due to its efficiency and as-rigid-as-possible deformation. A first version of this work appeared in [47] which only used geometry for registration. In this paper we extend the previous method to also include texture features to be able to handle a wider variety of objects.

3. Online model building

We propose an online model building approach that reconstructs the 3D geometry of an object from a sequence of depth scans. This task poses several requirements 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 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.

3.1. Data acquisition

The interactive scanning system is built on top of our real-time structured-light range sensor [45], running at 30 fps. This setup delivers one depth map $\mathscr{D}_t$ (VGA resolution) per frame t, containing for each pixel u a depth value d_u^* (in mm) corresponding to a 3D position $\mathbf{p}_u^*$. An additional RGB color value $\mathbf{b}_u^*$ is given for each pixel from a separate texture image B_t.

3.1.1. Input data processing

Morphological operators remove small isolated patches that are likely outliers. Normals $\mathbf{n}_u^*$ are estimated at each scan pixel u using the method described in [32]. In order to penalize potentially inaccurate vertices, each depth value is assigned an input confidence value $c_u^* \in [0,1]$, which is initialized to $c_u^* = 0$ at a depth map discontinuity, otherwise $c_u^* = 1$. The confidence values are then spread out by a diffusion process [43]. Fig. 2a depicts an input scan of '*boris*' colored according to confidence (the '*boris*' model itself is shown in Fig. 13a).

In the following, we describe the basic surfel representation (Section 3.2). In order to bring the online surfel model into correspondence with the current scan, we perform rigid registration with the fast ICP algorithm (Section 3.3). After successful registration, the new scan is integrated with the surfel model, as described in Section 3.4. Registration and integration are then combined into a practical interactive scanning system (Section 3.5). An extension to also use texture features during modeling is given in Section 3.6. Section 4 then presents our approach to explicitly handle the loop closure problem.

3.2. Surfel representation

We adopt an explicit surface representation similar to [18] by representing the model surface $\mathscr{M}_t$ (integrated up to frame t) as a set of surfels s_i, with $i = 1 \ldots, N_t$, each surfel having a position $\mathbf{p}_i$, normal vector $\mathbf{n}_i$, radius r_i and visibility confidence v_i (see Fig. 2b). 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.

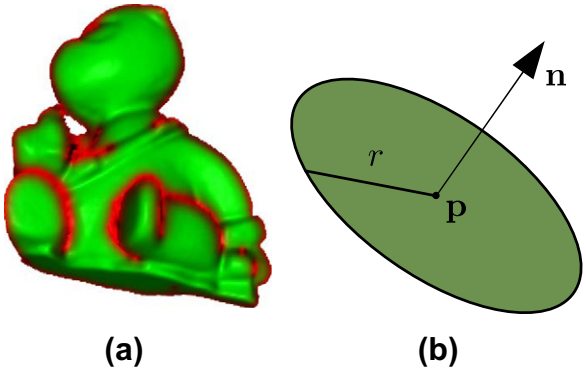


Fig. 2. (a) Input depth map of '*boris*'. Green represents high data confidence, red represents low data confidence. (b) Model surfel: a surfel is described by its position **p**, normal **n** and radius *r*. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.2.1. Visibility confidence

In order to estimate its reliability, each surfel is assigned a visibility confidence v_i. 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. 3a). For our setup, a surfel has high visibility confidence if it has been observed in at least 6 out of the 64 bins.

3.2.2. Visualization

A surfel is efficiently visualized as a hexagon (Fig. 3b) by rendering four triangles which are directly calculated on the GPU from $\mathbf{p}_i$, $\mathbf{n}_i$ and r_i using geometry shaders. This results in a significant memory and performance benefit as the triangles need not be stored and transferred to GPU memory.

3.3. Registration

We apply fast ICP [36] to align the online model M_t with the input data of the current depth map $\mathscr{D}_t$. All surfels of $\mathscr{M}_t$ are iteratively projected into $\mathscr{D}_t$, and the optimal rigid transformation $[\mathbf{R}_t, \mathbf{t}_t]$ is calculated by minimizing the point-to-plane distance:

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

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. Outlier correspondences which would otherwise distort the registration are removed by setting $w_i = 0$. We assume a correspondence is incorrect if the associated normals deviate by more than $t_{angle} = 60°$ as the corresponding points will likely refer to different surface patches. Likewise, correspondences which are further apart than a given threshold t_{dist} are also considered outliers. A suitable distance threshold can be found by setting t_{dist} to twice the mean distance of all correspondences.[1]

3.3.1. Registration failure

In order to check for registration failure, a virtual depth map $\widehat{\mathscr{D}}_t$ is generated of the current model $\mathscr{M}_t$ using the rigid transformation $[\mathbf{R}_t, \mathbf{t}_t]$ and the intrinsics of the real-time 3D scanner. Each pixel u 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 (2 mm 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 the cases *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.

[1] While the median would be more correct, we use the mean for efficiency reasons.

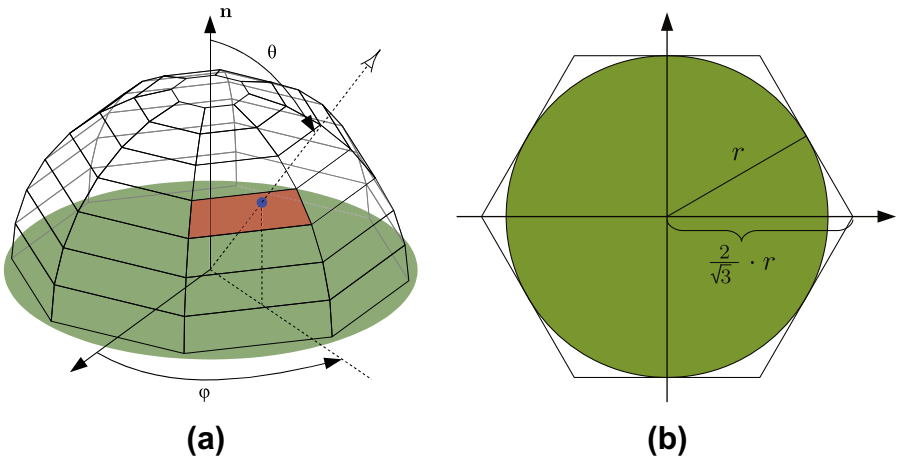


Fig. 3. (a) Surfel visibility confidence: a pair of a polar angle θ and an azimuth angle φ characterizes a view direction for each surfel. The bookkeeping of the view direction from which the surfel has already been observed is done with a binary histogram (64 bins). (b) Hexagonal surfel approximation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

3.4. Integration

After successful registration, the current depth map $\mathscr{D}_t$ is used to update the online model $\mathscr{M}_t$. 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.

3.4.1. Surfel update

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

$$\mathbf{p}'_i = \mathbf{R}_t \mathbf{p}_i + \mathbf{t}_t, \tag{2}$$

$$\mathbf{n}'_i = \mathbf{R}_t \mathbf{n}_i. \tag{3}$$

The z component $d'_i = \mathbf{p}'_i(z)$ is the surfel's depth. Each surfel is compared against the depth d^*_i, normal $\mathbf{n}^*_i$ and confidence c^*_i of the input scan $\mathscr{D}_t$, as well as the depth $\hat{d}_i$ and visibility confidence $\hat{v}_i$ from the virtual depth map $\widehat{\mathscr{D}}_t$. There are four 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 $d_{max} = 5$ mm. 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.

3.4.2. 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 ($c^*_k \geqslant c_{min}$). After each surfel addition and update, the surfel visibility confidence histogram is updated accordingly.

3.4.3. 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 is removed that has not been updated within the last $t_{starve} = 30$ frames and has not yet reached a minimum visibility confidence value $v_{starve} = 3$.

3.4.4. 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 and no more unsolved loop closure problems remain (see Section 4). 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* [18]. 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°. Currently, the user needs to determine whether the coarse object structure has been acquired and manually switch to *surface growing*, but for future work we would like to replace this by some automatic scheme.

3.4.5. 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:

$$r_i = \frac{1}{\sqrt{2}} \frac{d_i'/f}{\mathbf{n}_i'(z)} \tag{4}$$

where f is the focal length of the range sensor and $\mathbf{n}_i'(z)$ the z-component of the normal vector $\mathbf{n}_i'$. 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. Radius shrinking also occurs when surface parts that were initially observed in an oblique view are re-observed with more detail in a frontal view. Currently, the position update is independent of the radius update. However, with an increased level of detail, the recorded scan positions will be more accurate, and the surfel should therefore be replaced by the new scan position instead of being updated (as the running average includes the lower resolution positions). This is a trade-off between noise and accuracy. In future work, we plan to investigate what the best update strategy is in these cases.

3.5. Interactive scanning

In our current setup the user holds the object in front of a 3D scanner, and the reconstructed online model is displayed in real-time on the screen (see Fig. 4). The interface of our interactive scanning system is similar to [35], displaying scan and model in a single panel. Fig. 5 shows a screenshot of the interactive modeling user interface in action. The current scan is overlaid over the model to facilitate re-initialization. The model surfels are colored according to the visibility confidence such that the user can see which parts are not yet confident. The hand and background are currently removed by using a black glove and curtain which are not reconstructed by the scanner. In the future, we plan to integrate detection mechanisms to automatically remove these parts from the scans without posing constraints on the color, e.g. using hand tracking. Depending on the application, the final model may be converted to a watertight triangle mesh after the scanning session using the freely available Poisson surface reconstruction software [28]. This step also fills the parts that cannot be acquired due to limitations of the scanner hardware. If no watertight mesh is required, the method from [17] produces the best results for converting our online model to a triangle mesh.

3.6. Texture features

For many objects geometry-based registration and integration may result in ambiguities due to symmetries of the object. For example, a cylinder has two unconstrained degrees of freedom, namely translation along and rotation around the principal axis. In previous work [46] we presented a method for pair-wise scan registration that uses texture as an additional cue to break geometric symmetries. Here we show how the online modeling approach can be extended to also include texture features. Texture registration itself is an adaptation of [46] where the main difference is in the procedure how texture features are efficiently integrated into the surfel model. For this purpose we extend the surfel representation to optionally include a feature descriptor $\mathbf{f}_i$ for each surfel s_i.

3.6.1. Texture registration

Given the texture image B_t, we extract feature points using Harris corners [19] and SURF descriptors [3], with each feature descriptor covering a fixed size area of radius $d_{rad} = 6$ mm. Each feature point j consists of a 3D position $\mathbf{p}_j^*$ and feature descriptor $\mathbf{f}_j^*$. Given the set of feature surfels $\mathbf{f}_i$ and scan features $\mathbf{f}_j^*$ with their associated 3D positions, feature matching and a fast RANSAC-like method are employed to extract the optimal coarse rigid transformation between model and scan. The resulting feature correspondences are included into ICP (Eq. (1)) to break the symmetries (details are given in [46]).

3.6.2. Texture integration

After registration, each scan feature f_j may be integrated into the online model by simply updating the closest model surfel s_i. However, the same feature appearing in multiple scans would most likely update different surfels as the feature positions vary slightly due to noise. The online model would therefore quickly fill with many duplicate features. This case is prevented by prohibiting features on the model to be closer than a given threshold t_{feat}. To a certain extent the extracted SURF feature descriptors are viewpoint invariant [3]. Nevertheless, if the viewpoint changes too drastically, corresponding feature may not match anymore and registration may fail. The threshold t_{feat} is therefore only employed when the viewpoint between model feature i and scan feature j is below a given threshold t_{view}. Thus, a feature may be added multiple times but with different feature descriptors representing different viewpoints. Threshold t_{feat} and t_{view} pose a trade-off between robustness and performance. Setting $t_{feat} = 5$ and $t_{view} = 30°$ proved to be suitable for all our hand-sized examples. The complexity of registration and integration is $O(N_f^{(t)} K_f^{(t)})$, where $K_f^{(t)}$ is the number of scan features and $N_f^{(t)}$ the number of model features. Note that both $N_f^{(t)}$ and $K_f^{(t)}$ are much smaller then the total number of surfels. Fig. 6 shows the extracted features on a textured scan of a cylindri-

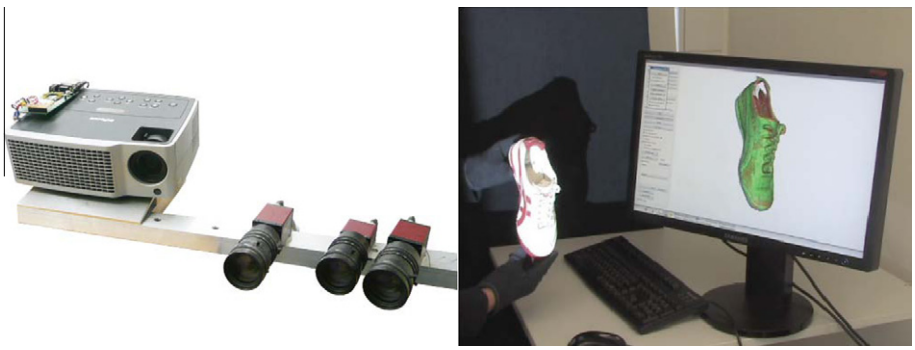


Fig. 4. Scanning of a '*shoe*'. left) real-time 3D scanner right) the interactive scanning system in action.

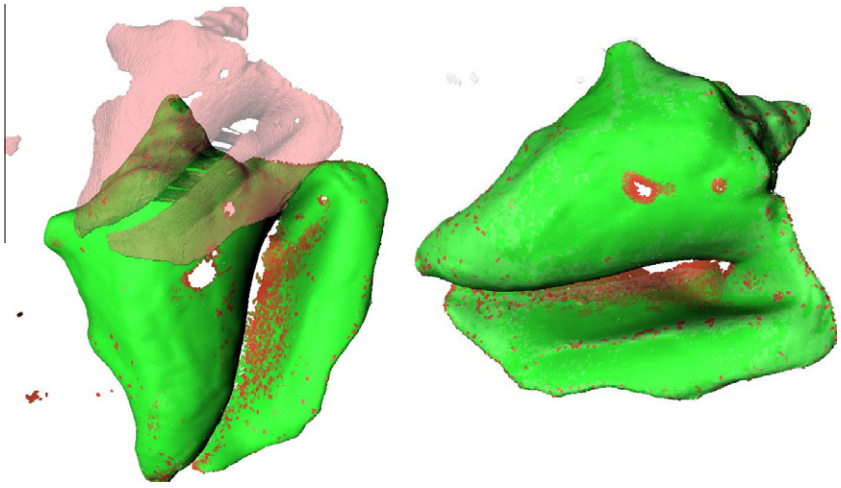


Fig. 5. User interface for the interactive scanning tool using an online model. The model surfels are colored according to confidence (red: low confidence, green: high confidence). The scan is overlaid over the model. Left: registration still unsuccessful. Right: registration and integration successful. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

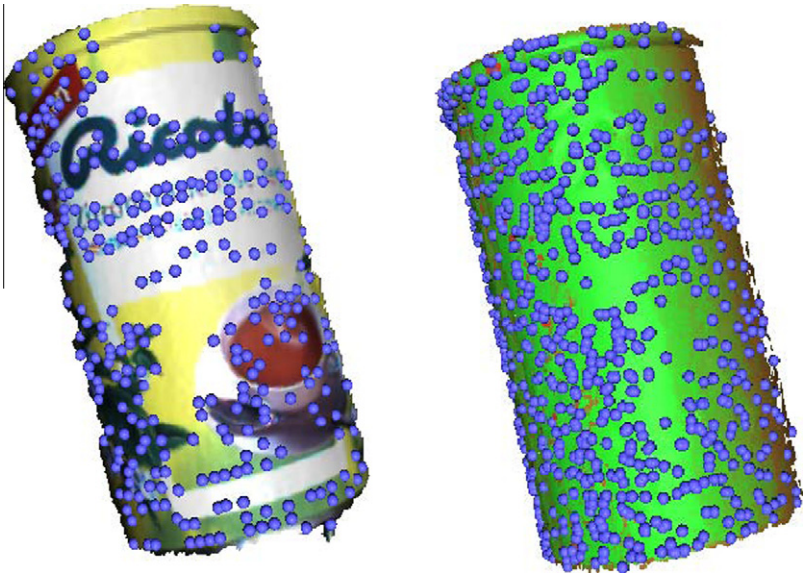


Fig. 6. Extracted feature points on the scan (left) and all integrated features on the model (right).

cal object. The same figure also shows the accumulated features on the online model after 30 frames.

4. Online loop closure

The online model building method from the previous section can be used for interactive 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. 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. 7 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. We use a graph-based space deformation method similar to [41] due to its efficiency and as-rigid-as-possible deformation.

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 (Section 4.1). Section 4.2 then shows how the basic registration and integration methods are adapted to include the topology graph, and Section 4.3 finally presents the as-rigid-as-possible deformation for loop closure.

4.1. Topology graph

Whenever two surfels have been seen together in a scan, the scan data places a constraint on the relative positions of these

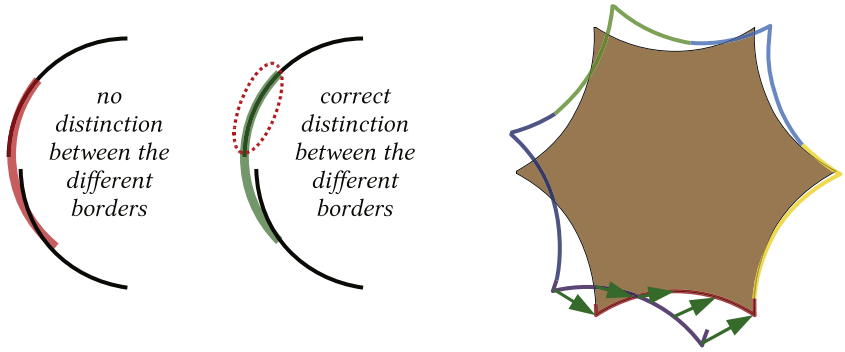


Fig. 7. Solution to the loop closure problem: When a loop closure case is detected, our algorithm proceeds by only registering and integrating new input data with one of the '*meeting*' borders. Once the overlap of the involved surface patches is sufficiently large, the model is closed by an as-rigid-as-possible deformation.

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 nodes $\mathbf{g}_j$, with $j = 1 \ldots, S$, which are 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. 8).

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 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 $t_r = 15$ mm. 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:

4.1.1. 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.

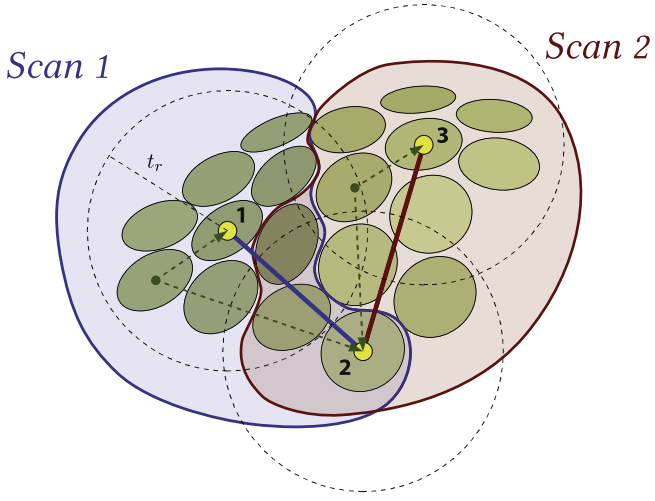


Fig. 8. Topology graph. In this example, three surfels act as topology graph nodes, indicated by the yellow bullets. Nodes 1 and 2 are connected, since both have been observed in scan 1. The same holds for nodes 2 and 3 in scan 2. However, node 1 and 3 have never been seen together, so they are not connected. Every surfel is properly represented since all surfels are attached to at least one node (indicated by green dashed arrows). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.1.2. 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*. This means that node addition is dependent on the surfel processing order.

4.1.3. 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.

4.1.4. 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 $O(N\ S)$, but $S \ll N$ and all operations can be efficiently implemented on the GPU using bitsets.

4.2. Component registration & integration

The online model building method needs to be adapted to properly include the topology graph, which is used to distinguish between the different borders of the model. Registration and integration is only performed with one of the borders until loop closure can be applied. 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 $d_{node} = 5$ mm. The nodes are divided into a set of L connected components by intersecting the set of visible nodes with the topology graph (see Fig. 9). 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 during registration and integration. 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.

4.3. As-rigid-as-possible deformation

The full procedure for loop closure is visualized in Fig. 9. Once the overlap between the model components and the input scan has be-

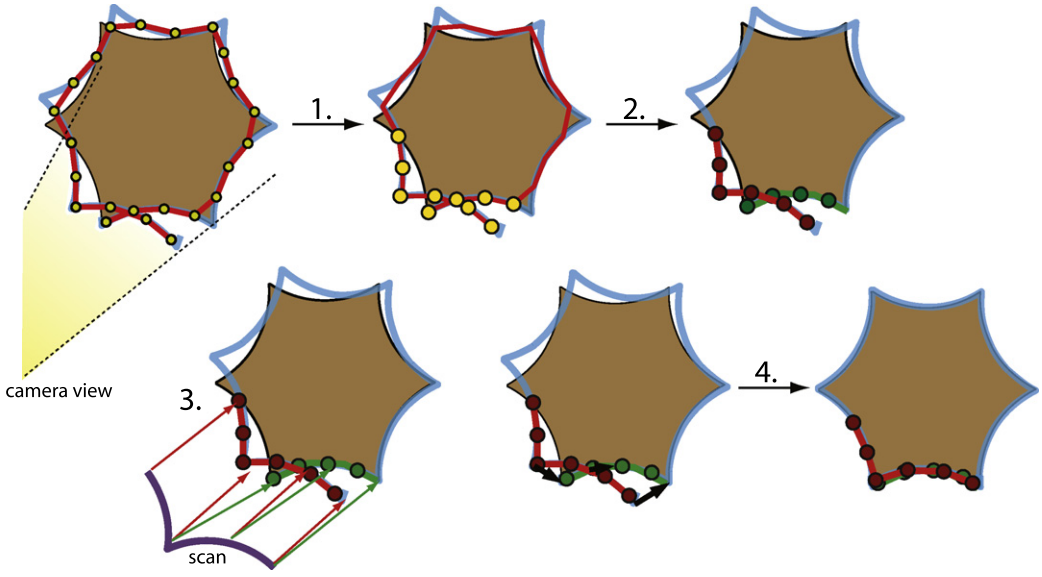


Fig. 9. Steps of the loop closure procedure: (1) Determine the currently visible nodes. (2) Intersect the visible nodes with the topology graph and decompose into connected components. (3) Register each component individually to the input scan. (4) Deform the model surface if components have sufficient overlap.

come 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 by updating the topology graph accordingly.

4.3.1. Deformation

For online loop closure we employ an as-rigid-as-possible deformation using the topology graph similar to the method described by Sumner et al. [41]. 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:

$$\tilde{\mathbf{p}} = \mathbf{R}_j(\mathbf{p} - \mathbf{g}_j) + \mathbf{g}_j + \mathbf{t}_j. \tag{5}$$

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 $\tilde{\mathbf{p}}_i$ and normal $\tilde{\mathbf{n}}_i$ of each surfel i are calculated as weighted sums of their counterparts after application of the deformation graph transformations:

$$\tilde{\mathbf{p}}_i = \sum_{j=1}^{S} w_j(\mathbf{p}_i)\left[\mathbf{R}_j(\mathbf{p}_i - \mathbf{g}_j) + \mathbf{g}_j + \mathbf{t}_j\right] \tag{6}$$

$$\tilde{\mathbf{n}}_i = \sum_{j=1}^{S} w_j(\mathbf{p}_i)\mathbf{R}_j^{-\top}\mathbf{n}_i \tag{7}$$

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:

$$w_j(\mathbf{p}_i) = (1 - \|\mathbf{p}_i - \mathbf{g}_j\|_2 / d_{max})^2 \tag{8}$$

and then normalized to sum to one. d_{max} is the distance to the $k+1$-nearest node. The experiments in [41] 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:

$$\mathbf{p}_i^\ell = \mathbf{R}^\ell \mathbf{p}_i + \mathbf{t}^\ell \tag{9}$$

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

$$E_{pos} = \sum_{\ell=1}^{L} \sum_{i \in H(\ell)} \|\mathbf{p}_i^\ell - \tilde{\mathbf{p}}_i\|_2^2 \tag{10}$$

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

where $\Omega(j)$ 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:

$$\min_{\mathbf{R}_1, \mathbf{t}_1 \ldots \mathbf{R}_S, \mathbf{t}_S} E_{pos} + w_{reg} E_{reg} \tag{11}$$

where $w_{reg} = 0.1$ in our system. Eq. (11) is non-linear in terms of the $7S$ unknowns that define the rotation matrices (quaternions) and translation vectors of each node. The non-linear optimization is sparse and can be solved efficiently using Gauss–Newton iterations in conjunction with an efficient sparse linear solver. Spurious local minima are avoided by initializing all node transformations by the optimal global rigid transformation that aligns all components simultaneously. This global transformation can be determined in closed-form using the absolute orientation algorithm by Horn [22]. Once the optimal deformation is found, the surfels are deformed and both surfels and topology graph are updated.

5. Experimental results

In this section both qualitative and quantitative experimental results are presented, demonstrating the accuracy and robustness of our interactive modeling approach. In the following, '*Online Sequential*' stands for using our online modeling method with loop closure turned off, while '*Online Loop Closure*' stands for loop closure turned on. We compare our approach to a previous method that uses scan-to-scan registration ('*Sequential*') with an additional global offline loop closure step ('*Offline Loop Closure*') [35,46]. Both '*Sequential*' and '*Offline Loop Closure*' may be employed together with *anchor* scans which are specifically designed to reduce error accumulation during registration [35,46].

Some objects reconstructed using our proposed modeling method are shown in Fig. 10. All parameters of our modeling methods have been set experimentally and are left unchanged throughout all experiments. The effect of using texture features for registration is shown in Fig. 11. Geometric symmetries lead to artifacts in the reconstruction, which can be seen as a *prolongation* of the can and the *additional* handles in the pot example. Texture features break the symmetries and the objects are correctly reconstructed.

Fig. 11. Modeling results for textured objects. The middle column shows the artifacts that arise due to symmetries when only geometry is used for registration. The right column shows the correctly reconstructed objects when texture features are used for registration.

5.1. Complexity and performance

The complexity of registration and integration is $O(N)$, texture registration and integration $O(N_f K_f)$ and topology graph update $O(NS)$, with N being the number of surfels in the model, S the number of nodes, N_f the number of model features and K_f the number of scan features. Thus, the complexity scales linearly with the number of surfels as $S \ll N$, yet the actual computation scales sublinearly, since only the visible surfels are processed at each step. Most of the presented algorithms have been implemented on the GPU. For a typical hand-sized object consisting of about 30 k vertices in each scan, the interactive modeling system runs at approximately 20 *fps*. Registration and integration of texture features requires an additional 25 ms, thus dropping performance to about 12 *fps*. For '*Offline Loop Closure*', the post-processing steps require additional computation time approximately proportional to the number of scans. For a typical modeling session with one to two thousand scans, this takes around 5–10 min. For '*Online Loop Closure*', the actual online loop closure optimization takes 2–3 sec, but is typically only performed once or twice per scanning session. The system locks during the optimization, but this does not disturb the interaction significantly. In the case of '*Online Loop Closure*' no additional post-processing is required.

5.2. Modeling accuracy on synthetic data

In order to quantitatively evaluate our interactive scanning systems, we performed experiments using synthetic data. For this purpose, we created a virtual scanning sequence of the '*boris*' object (see Fig. 13a), as well as the '*bunny*', '*armadillo*', '*dragon*', and '*buddha*' objects from [40] shown in Fig. 12, by rotating each object once around the x-axis and once around the y-axis. Each sequence consists of 142 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 all objects is scaled to approximately 150 mm in all dimensions. The virtual distance from the range sensor is 1000 mm with approximately 1 mm lateral resolution, resulting in approximately 15–20 k vertices per scan.

5.2.1. Integration

In order to evaluate the accuracy 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 reconstruction method '*Algebraic Point Set Surfaces*' (APSS) from [17], the scan integration method VRIP from [12], Poisson reconstruction from [28], and our proposed online modeling method. We set the parameters of all methods to produce about the same number of mesh vertices. Fig. 13b shows the accuracy 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. On the other hand, the online and the offline methods perform similarly, introducing a slight error for noise-free data, but generally handling noise well. APSS reconstruction produces the best results in our experiments, on average being always slightly better than our proposed online reconstruction method. VRIP reconstruction performs slightly worse than our proposed integration method for low noise levels, but performs better for higher noise levels. Poisson integration performs significantly worse,

Fig. 10. Modeling results for a number of objects using online modeling with online loop closure.

Fig. 12. The stanford repository objects used for synthetic data evaluation: '*bunny*', '*armadillo*', '*dragon*', and '*buddha*'.

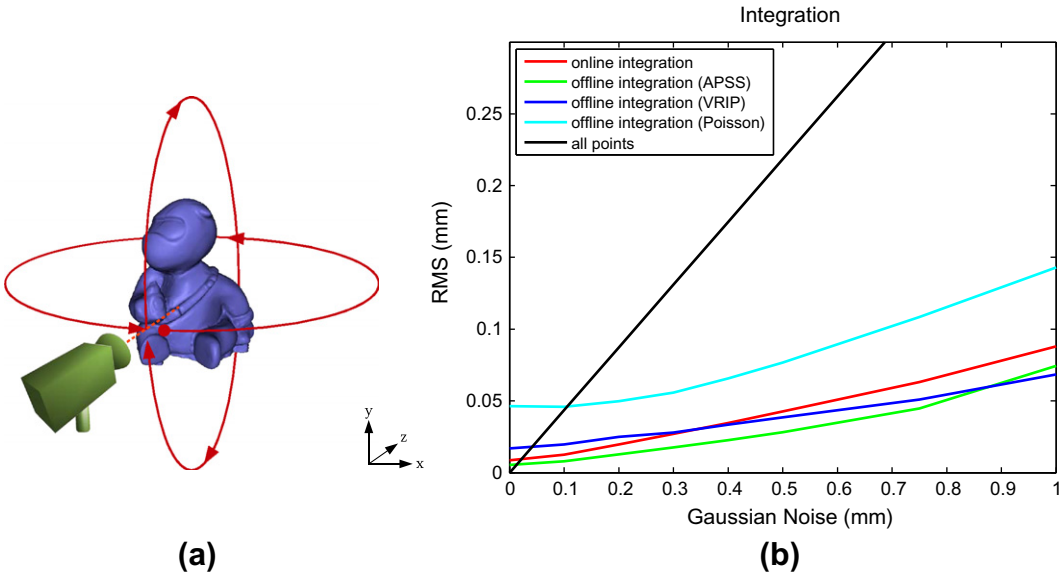


Fig. 13. (a) Synthetic rendering of '*boris*'. (b) Comparison of integration algorithms: online integration is competitive in performance with offline integration methods in terms of noise handling. All integration methods introduce a small error for zero Gaussian noise compared to the baseline (using all input scan vertices), but are successful in handling the noise of the input scans.

which can be explained by a slight geometric distortion induced by the reconstruction method. To conclude, our proposed reconstruction method performs competitively in terms of noise handling compared to the offline integration methods. Note that the noise level of our scanner is typically in the range of 0.1–0.5 mm.

5.2.2. Registration

Registration accuracy is evaluated by comparing the reconstructions using the reference transformations, '*Sequential*' registration without (1) or with (2) *anchor* scans, '*Offline Loop Closure*' on top of sequential registration, '*Online Sequential*', and '*Online Loop Closure*'. All methods are used in conjunction with our proposed integration method.

5.2.3. Gaussian noise

In a first experiment, we only add Gaussian noise to the input data. Fig. 14 shows that '*Online Loop Closure*' performs as well as '*Offline Loop Closure*', and is almost as accurate as the reference transformations. Using '*Sequential*' registration without *anchor* scans performs significantly worse. Note, however, that white noise on its own seems to only introduce a small loop closure discontinuity.

5.2.4. Distortion

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

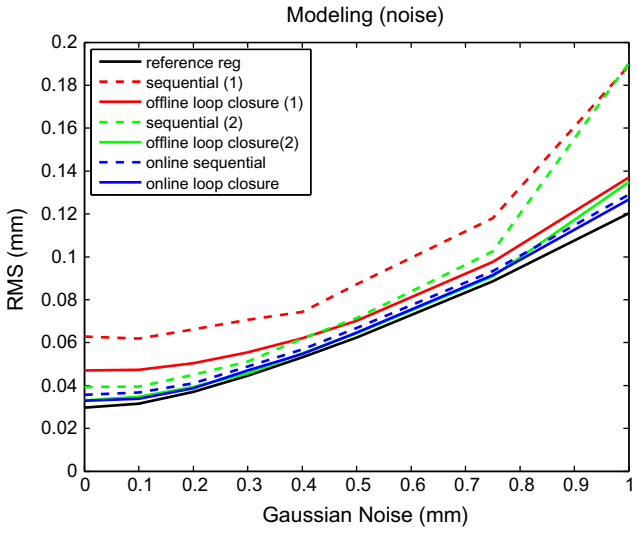


Fig. 14. Comparison of registration algorithms depending on the amount of Gaussian noise on the simulated input scan. '*Online Loop Closure*' performs as good as '*Offline Loop Closure*' with *anchor* scans, and both approach the accuracy of the reference transformations. Important to note is that '*Online Loop Closure*' is always better than '*Online Sequential*'.

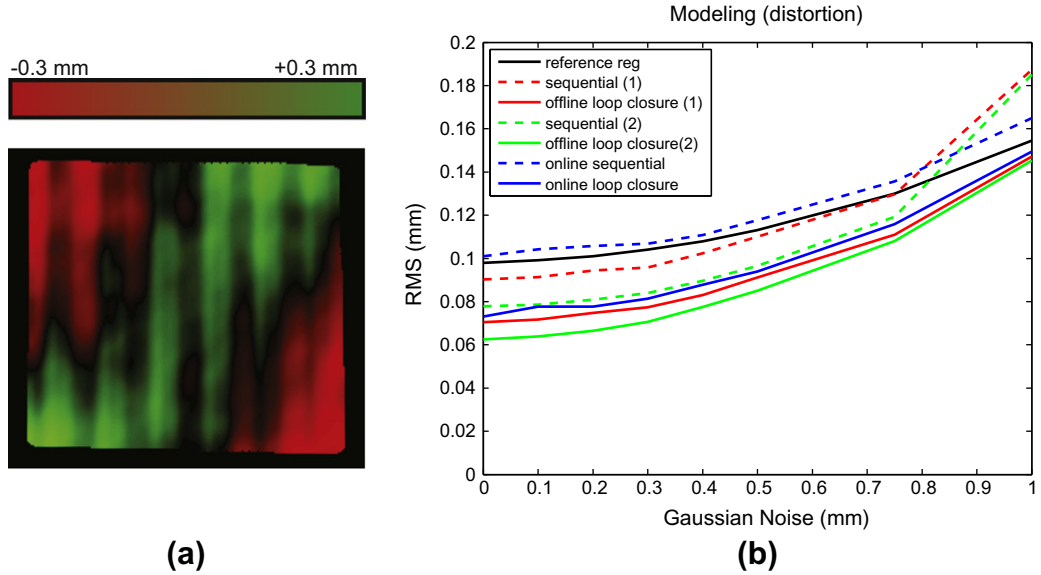


Fig. 15. (a) Measured depth distortion of our range scanner. (b) Comparison of registration algorithms when scanner specific depth distortion is applied additionally to the noise. '*Online loop closure*' significantly improves the results compared to ignoring loop closure, but does not yet achieve the performance of '*Offline loop closure*'.

the smoothed data to the planar surface is used to generate a distortion map (see Fig. 15a), modeling pattern artifacts and calibration errors of our setup (we verified that the error does not come from the planar surface itself). Fig. 15 shows that using '*Online Loop Closure*' significantly improves the reconstruction results compared to ignoring the loop closure problem. Fig. 16 shows this improvement for an exemplary reconstruction of '*boris*' with distortion and Gaussian noise ($\sigma = 0.3$ mm). Offline optimization is still better than online optimization, but cannot be seamlessly embedded into the interaction. Interestingly, '*Sequential*' registration with *anchor* scans is performing almost as well as '*Online Loop Closure*'. However, distortion is only one of the scanning error sources, and the following real world experiments show that '*Online Loop Closure*' performs significantly better than '*Sequential*' registration with *anchor* scans.

5.3. Modeling accuracy on real data

5.3.1. Integration

In this experiment, we evaluate the robustness the different integration methods with respect to outliers. The synthetic experiments only evaluated the accuracy with respect to noise, but outliers frequently appear in real data. We therefore qualitatively compare the integration results for an exemplary modeling session of the '*boris*' object, as shown in Fig. 17. On their own, both VRIP and APSS produce meshes with many outliers as these methods were not designed to explicitly handle outliers. By using morphological operators on triangle meshes, and by only keeping the largest surface, most outliers are removed. Nevertheless, as can be seen in the resulting meshes, artifacts remain (marked in red). While VRIP produces slightly more detailed results, our proposed online reconstruction method handles outliers much more gracefully. We have verified that for different objects the results are similar. Thus, due to efficient outlier handling in combination with the immediate feedback to the user, the online modeling method proves to be an efficient interactive modeling tool.

5.3.2. Online loop closure

Figs. 18 and 19 demonstrate the benefit of using '*Online Loop Closure*' on real data compared to having loop closure switched off. Looking at the cross-section of the reconstructed '*banana*' in Fig. 18, 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

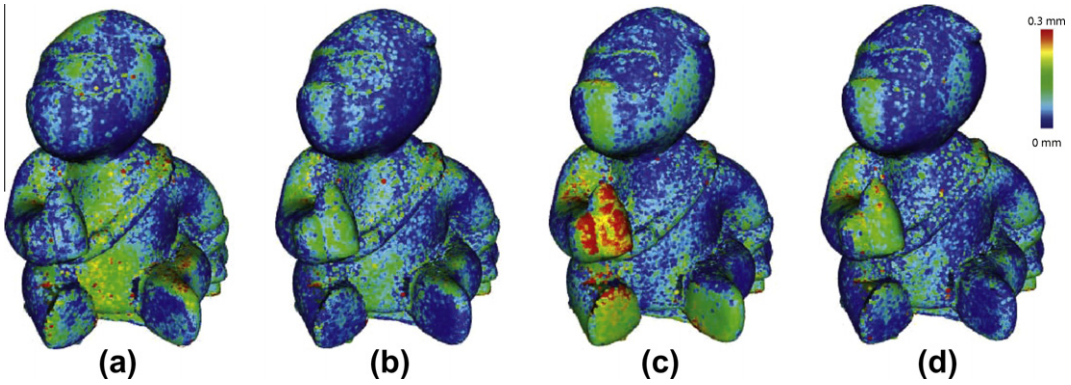


Fig. 16. Color coding of the reconstruction error for the '*boris*' object with synthetic distortion and Gaussian noise ($\sigma = 0.3$ mm): (a) '*Sequential*', (b) '*Offline loop closure*', (c) '*Online Sequential*', and (d) '*Online loop closure*'.

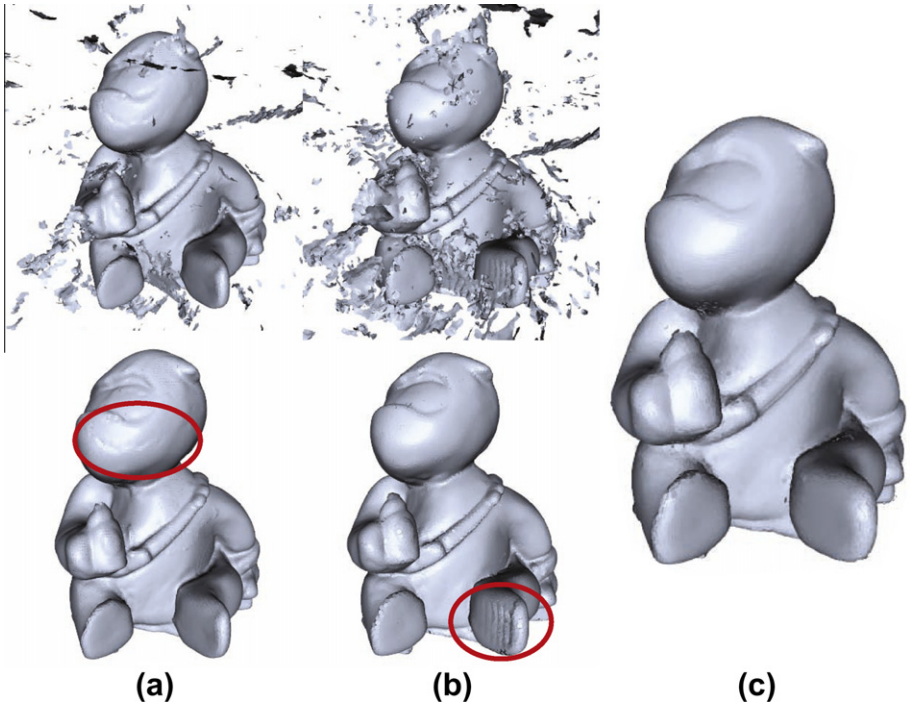


Fig. 17. (a) Integration with VRIP without (top) and with (bottom) cleanup. Artifacts are highlighted in red. (b) Integration using APSS. (c) Integration using our proposed online modeling method. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

error and the resulting artifacts are shown in the reconstructed '*shoe*' in Fig. 19. 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.

5.3.3. Quantitative results

We perform two experiments for a quantitative analysis of the modeling accuracy of our real-time modeling methods. In a first experiment, we estimate the RMS error of reconstructing the '*boris*' object. As the exact geometry is unknown, we perform four independent modeling sessions of '*boris*' and calculate the error between all reconstructed models. The individual reconstructed models are registered to each other using ICP and the RMS error is calculated for each alignment. We assume that each individual RMS error is a sample of the true RMS error, and thus estimate the true RMS error as the mean of all individual RMS errors. Table 1 shows the corresponding results. '*Sequential*' registration leads to the worst results, but using an online model for sequential registration already improves the results significantly. '*Online Loop Closure*' improves the results further, approaching the accuracy of the '*Offline Loop Closure*' method.

The previous experiment only measures precision, but does not provide an accuracy estimate. Thus, in a second experiment we measure the reconstruction accuracy by comparing the reconstructed model against ground truth data. For this purpose, we record four different modeling sessions of a '*lego*' construction as shown in Table 2, where for each modeling session the object is placed on a turntable and rotated from different starting positions.

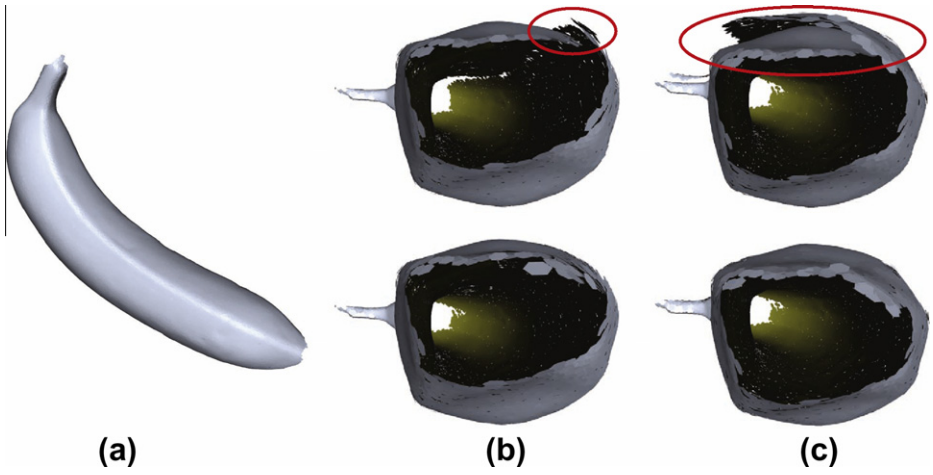


Fig. 18. (a) Reconstructed '*banana*'. (b) Registration without loop closure ('*banana*' cross-section): the loop closure error (top) is only compensated for locally (bottom) by *averaging out* the error at the border overlap. (c) Registration with loop closure: the loop closure error is compensated for globally, spreading the accumulated error over the full model. Note in particular the distortion of the left and bottom surface parts.

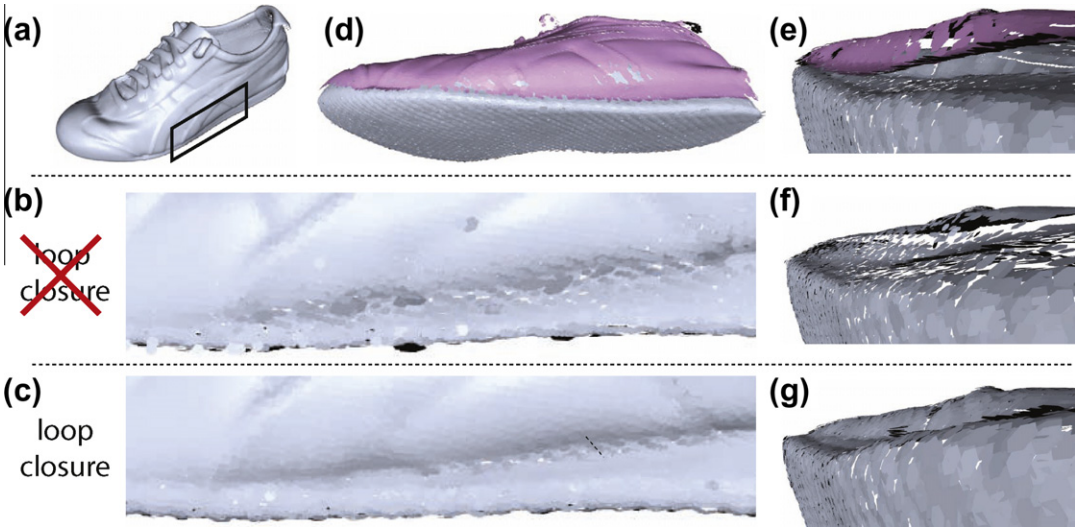
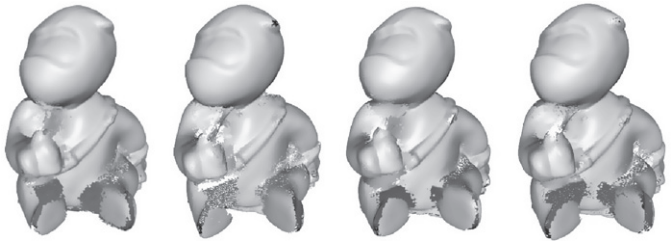


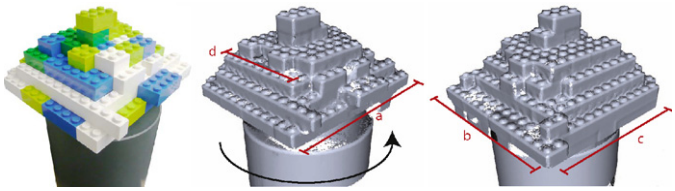
Fig. 19. (a) Reconstructed '*shoe*'. (b) Close-up view without loop closure. (c) Close-up view with loop closure. (d) View of the two unconnected surface parts leading to the loop closure problem. (e) The surface discontinuity is prominent in the zoomed view. (f) No loop closure *averages out* the discontinuity resulting in a noisy and distorted reconstructed geometry. (g) Online loop closure correctly compensates for the discontinuity.

Table 1
Modeling accuracy for *boris* object. The four different registration methods were applied to four independent modeling sessions of *boris*. The resulting point clouds of each method were registered with one another and the average RMS error $\bar{\mu}$ and its standard deviation $\sigma(\mu)$ were calculated. '*Offline Loop Closure*' produces the best results, but '*Online Loop Closure*' is competitive. '*Online Sequential*' registration is worse, but is still significantly better than simple '*Sequential*' registration without an online model.



Registration method	$\bar{\mu}$ (mm)	$\sigma(\mu)$ (mm)
'*Sequential*'	0.253	0.040
'*Offline Loop Closure*'	**0.088**	0.013
'*Online Sequential*'	0.145	0.041
'*Online Loop Closure*'	**0.106**	0.020

Table 2
Modeling accuracy for a '*lego*' construction. The different registration methods were applied to four independent modeling sessions of the object, where the object was turned on a turntable with different starting positions. The ground truth distances (a–d) were measured on the object using a gauge with an approximate error around 0.05 mm. For each distance, the RMS error (in mm) between the reconstructed distance and the ground truth distance is calculated. '*Offline Loop Closure*' performs best, but online modeling with '*Online Loop Closure*' is competitive. The large error for distance (d) can be explained due to smaller amount of geometry on that part, resulting in less influence during registration, and thus a significantly lower accuracy.



Registration Method (error in mm)	(a)	(b)	(c)	(d)	total
'*Sequential*'	0.44	0.91	0.50	0.97	0.74
'*Offline Loop Closure*'	0.11	0.16	0.13	0.84	**0.44**
'*Online Sequential*'	0.23	0.27	0.24	1.06	0.57
'*Online Loop Closure*'	0.05	0.12	0.16	1.00	**0.51**

For accuracy measurements, we compare four different distances on the reconstructions against the ground truth distances. Ground truth is estimated using a gauge with an approximate accuracy of 0.05 mm. The distances on the reconstruction are measured by manually selecting the corresponding surfels. A locally planar patch is fitted for each endpoint and the distance between the two patches is an estimate of the distance. As can be seen in Table 2 the results agree with the previous experiment: '*Offline Loop Closure*' performs best, but '*Online Loop Closure*' performs almost as well. '*Sequential*' registration performs by far the worst. To conclude, our proposed online modeling approach with online loop closure permits almost as accurate results as offline methods, but with the advantage of immediate feedback to the user.

6. Discussion and conclusion

We have presented a complete interactive scanning system that returns an online reconstructed 3D model that serves directly as the final result. In practice, the visibility confidence measure combined with surfel starvation, surfel growing, and the implicit user control permits graceful outlier handling, resulting in artifact-free models. In contrast, offline integration methods may produce artifacts in the reconstructed models, and these are only visible after the additional post-processing computation. 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. This makes the interactive modeling process fast and intuitive.

The online loop closure distributes the accumulated registration error evenly by distorting the model. It implicitly 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 overlap of the input scans. However, this restriction does not pose a problem in our setup, as our real-time 3D scanner generally provides sufficient accuracy.

Our approach currently has the following limitations that we plan to address in future work. We distinguish borders by finding connected components for the visible nodes. However, this might lead to cases where no loop closure will be performed, such as when rotating around an object with some surface patch being always visible. Hence, all nodes will be connected through a node on that surface patch, hindering a loop closure.

The surfels are currently updated using a weighted running average. We plan to investigate a more statistical representation that better handles the uncertainties of the input data including the higher accuracy when moving the object closer to the scanner.

The current implementation on the GPU poses restrictions on the maximum number of nodes and surfels. This is sufficient for hand-sized object, but further developments are required to handle arbitrary sized objects, in particular if the method is combined with a hand-held scanner to reconstruct much larger objects. A possible future direction would be an extension to deforming objects. For instance, the scanning of a human hand is an interesting research challenge, as it requires both fast deformation methods and an efficient surface correspondence detection mechanism.

References

[1] B. Allen, B. Curless, Z. Popović, The space of human body shapes: reconstruction and parameterization from range scans, ACM Transactions on Graphics 22 (2003) 587–594.
[2] A. Angeli, S. Doncieux, J.A. Meyer, D. Filliat, Real-time visual loop-closure detection, in: IEEE International Conference on Robotics and Automation (ICRA'08), 2008.
[3] H. Bay, T. Tuytelaars, L. Van Gool, SURF: Speeded up robust features, in: Europeen Conference on Computer Vision (ECCV '06), 2006.
[4] F. Bernardini, H. Rushmeier, The 3D model acquisition pipeline, Computer Graphics Forum 21 (2002) 149–172.
[5] P.J. Besl, N.D. McKay, A method for registration of 3-D shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (1992) 239–258.
[6] F. Blais, M. Picard, G. Godin, Accurate 3D acquisition of freely moving objects, in: 3D Data Processing, Visualization and Transmission (3DPVT '04), 2004.
[7] M. Botsch, O. Sorkine, On linear variational surface deformation methods, IEEE Transactions on Visualization and Computer Graphics 14 (2008) 213–230.
[8] B. Büttgen, T. Oggier, M. Lehmann, Ccd/cmos lock-in pixel for range imaging: Challenges, limitations and state-of-the-art, in: 1st range imaging day, 2005.
[9] R.J. Campbell, P.J. Flynn, A survey of free-form object representation and recognition techniques, Computer Vision and Image Understanding 81 (2001) 166–210.
[10] Y. Chen, G. Medioni, Object modelling by registration of multiple range images, International Journal of Image and Vision Computing 10 (1992) 145–155.
[11] N. Cornelis, L. Van Gool, Real-time connectivity constrained depth map computation using programmable graphics hardware, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05), 2005.
[12] B. Curless, M. Levoy, A volumetric method for building complex models from range images, in: SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, 1996.
[13] J.D. Deschênes, P. Lambert, P. Hebert, Interactive modeling with automatic online compression, in: 3D Data Processing, Visualization, and Transmission (3DPVT'06), 2006.
[14] N. Gelfand, S. Rusinkiewicz, L. Ikemoto, M. Levoy, Geometrically stable sampling for the ICP algorithm, in: 3-D Digital Imaging and Modeling (3DIM'03), 2003.
[15] F.F. Gionis, R.B. Fisher, A. Fitzgibbon, A. Gionis, M. Wright, D. Eggert, A hand-held optical surface scanner for environmental modeling and virtual reality, in: Virtual Reality World, 1996.
[16] G. Godin, M. Rioux, R. Baribeau, Three-dimensional registration using range and intensity information, in: Videometrics III, 1994.
[17] G. Guennebaud, M. Gross, Algebraic point set surfaces, ACM Transactions on Graphics 26 (2007) 23.
[18] M. Habbecke, L. Kobbelt, A surface-growing approach to multi-view stereo reconstruction, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07), 2007.
[19] C. Harris, M. Stephens, A combined corner and edge detection, in: Proceedings of The Fourth Alvey Vision Conference, 1988.
[20] A. Hilton, J. Illingworth, Geometric fusion for a hand-held 3d sensor, Machine Vision Applications 12 (2000) 44–51.
[21] A. Hilton, A.J. Stoddart, J. Illingworth, T. Windeatt, Implicit surface-based geometric fusion, Computer Vision and Image Understanding 69 (1998) 273–291.
[22] B. Horn, Closed-form solution of absolute orientation using unit quaternions, Journal of the Optical Society of America 4 (1987) 629–642.
[23] D. Huber, M. Hebert, Fully automatic registration of multiple 3D data sets, Image and Vision Computing 21 (2003) 637–650.
[24] G. Iddan, G. Yahav, 3d imaging in the studio (and elsewhere), SPIE Proceedings Series 4298 (2000) 48–55.
[25] T. Jaeggli, T. Koninckx, L. Van Gool, Online 3d acquisition and model integration, in: IEEE International Workshop on Projector-Camera Systems (ICCV '03), 2003.
[26] S.B. Kang, A.E. Johnson, Registration and integration of textured 3-D data, in: 3-D Digital Imaging and Modeling (3DIM '97), 1997.
[27] H. Kawasaki, R. Furukawa, R. Sagawa, Y. Yagi, Dynamic scene shape reconstruction using a single structured light pattern, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08), 2008.
[28] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Symposium on Geometry Processing (SGP '06), 2006.
[29] T. Koninckx, T. Jaeggli, L. Van Gool, Adaptive scanning for online 3D model acquisition, in: Workshop on Real-Time 3D Sensors and Their Use, 2004.
[30] S. Krishnan, P.Y. Lee, J.B. Moore, S. Venkatasubramanian, Global registration of multiple 3D point sets via optimization-on-a-manifold, in: Symposium on Geometry Processing (SGP'05), 2005.
[31] R. Lange, P. Seitz, A. Biber, R. Schwarte, Time-of-flight range imaging with a custom solid state image sensor, in: Laser Metrology and Inspection, 1999.
[32] N. Max, Weights for computing vertex normals from facet normals, Journal of Graphics Tools 4 (1999) 1–6.
[33] Q. Pan, G. Reitmayr, T. Drummond, ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition, in: British Machine Vision Conference (BMVC '09), 2009.
[34] K. Pulli, Multiview registration for large data sets, in: 3-D Digital Imaging and Modeling (3DIM '99), 1999.
[35] S. Rusinkiewicz, O. Hall-Holt, M. Levoy, Real-time 3d model acquisition, ACM Transactions on Graphics 21 (2002) 438–446.
[36] S. Rusinkiewicz, M. Levoy, Efficient variants of the ICP algorithm, in: 3-D Digital Imaging and Modeling (3DIM'01), 2001.
[37] C. Schütz, T. Jost, H. Hügli, Multi-feature matching algorithm for free-form 3d surface registration, in: International Conference on Pattern Recognition (ICPR'98), 1998.
[38] J.K. Seo, G.C. Sharp, S.W. Lee, Range data registration using photometric features, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005.
[39] O. Sorkine, M. Alexa, As-rigid-as-possible surface modeling, in: Symposium on Geometry Processing (SGP'07), 2007.
[40] Stanford Scanning Repository, 2009. <http://graphics.stanford.edu/data/3dscanrep/>.
[41] R.W. Sumner, J. Schmid, M. Pauly, Embedded deformation for shape manipulation, ACM Transactions on Graphics 26 (2007) 80.
[42] D. Tubic, P. Hébert, J.D. Deschênes, D. Laurendeau, A unified representation for interactive 3D modeling, in: 3D Data Processing, Visualization and Transmission (3DPVT '04), 2004.
[43] J. Weickert, Anisotropic Diffusion in Image Processing, Teubner, 1998.
[44] S. Weik, Registration of 3-d partial surface models using luminance and depth information, in: 3-D Digital Imaging and Modeling (3DIM '97), 1997.
[45] T. Weise, B. Leibe, L. Van Gool, Fast 3d scanning with automatic motion compensation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07), 2007.
[46] T. Weise, B. Leibe, L. Van Gool, Accurate and robust registration for in-hand modeling, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08), 2008.
[47] T. Weise, T. Wismer, B. Leibe, L. Van Gool, In-hand scanning with online loop closure, in: IEEE International Workshop on 3-D Digital Imaging and Modeling (3DIM '09), 2009.
[48] Z Corporation, <http://www.zcorp.com>, 2010.
[49] S. Zhang, P. Huang, High-resolution, real-time 3d shape acquisition, in: IEEE Conference on Computer Vision and Pattern Recognition Workshop, 2004.