Model-Free Authoring by Demonstration of Assembly Instructions in Augmented Reality

Ana Stanescu, Peter Mohr, Dieter Schmalstieg and Denis Kalkofen

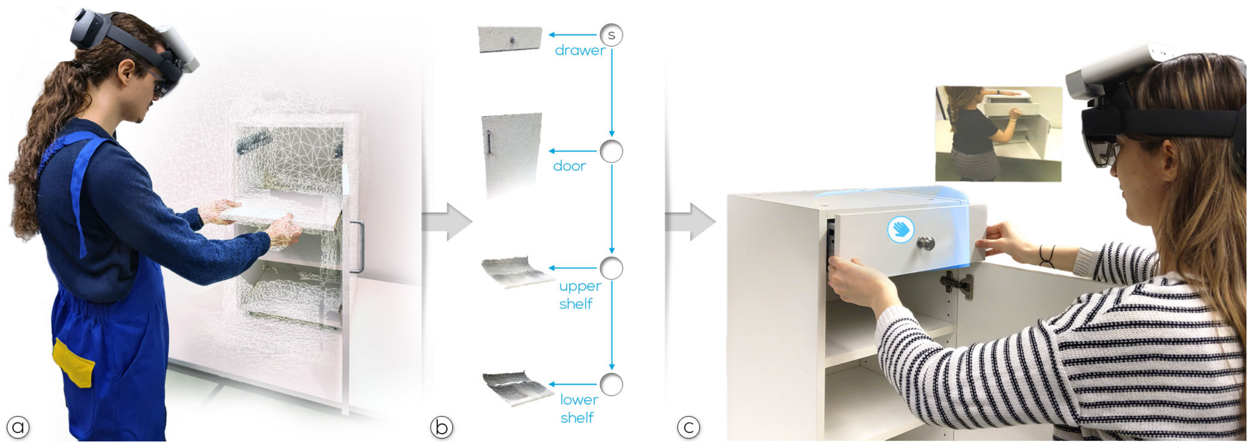

Fig. 1: (a) Our approach supports model-free authoring of step-by-step assembly instructions with no scene preparation from recording user demonstrations. (b) From the recordings, we build a 3D reconstruction, a video segmentation, and a partial assembly graph of the demonstrated procedures. (c) A user can follow the instructions, while being guided by AR situated visualizations that react to each step interactively.

Abstract— Among the most compelling applications of Augmented Reality are spatially registered tutorials. The effort of creating such instructions remains one of the obstacles precluding a wider use. We propose a system that is capable of extracting 3D instructions in a completely model-free manner from demonstrations, based on volumetric changes.The instructions are visualised later in an interactive Augmented Reality guidance application, on a mobile head-mounted display. We enable a technology that can be used by anyone in an ad-hoc tabletop setup for assemblies with rigid components.

1 Introduction

Many maintenance and construction tasks require following instructions on how to assemble and disassemble objects. It has been shown that delivering such instructions via Augmented Reality (AR) can reduce the cognitive load and, thus, the effort to complete the task [34]. The AR application enables delivering the equivalent of a traditional manual, but in a *situated* manner. In the AR application, the instructions are not statically printed on a piece of paper, but embedded in the user's 3D perception of the workspace.

While situated 3D instructions are compelling tools for end-users, creating them can be challenging. Authoring instructions commonly requires generating a database of 3D CAD objects first. The need for CAD modeling is a serious impediment to scalability. For example, a repair scenario would potentially require models of thousands of spare parts. Even if only a fraction of these parts is relevant, one usually does not know this subset in advance. Consequently, models for all possible occurrences are required. Arguably, model preparation is one of the important missing pieces standing in the way of a broader adoption of AR assembly tutorials. Thus, a *model-free* approach without object priors is much preferred.

Furthermore, the sensors deployed in the AR system can be used to monitor changes made by the user and assess the user's performance. Hence, interactive tutorials present the opportunity to automatically provide feedback (confirmations or warnings) accordingly. However, to date, AR instructions have been limited to showing only a few select objects, mostly consisting of easily detectable pieces [11, 37, 42]. These applications serve as a proof of concept, but lack a clear demonstration of scalability to complex assemblies with many parts. Our work makes an attempt towards better scalability by relying on real-time volumetric reconstruction using an RGB-D sensor for extracting part shapes directly during demonstration without prior modeling.

Thus, we propose the authoring of assembly sequences by mere demonstration. We focus on the problem of extracting information about the parts of a workpiece, i.e., an object that is being manipulated (e.g., assembled, disassembled, or reconfigured) by a human operator. Our system extracts information about an assembly tutorial directly from the user's actions, with no prerequisite information about the objects, and without scene preparation. We identify, from one or multiple sequential user demonstrations, recorded as RGB-D camera streams, (1) the *part geometry* of an object, (2) the *assembly graph* that encodes possible assembly sequences, and (3) video sequences that show the performed actions. The resulting 3D tutorial is subsequently visualized in 3D AR, using a Microsoft Hololens 2 in our current implementation. Please see Figure 1 shows an illustration of the proposed workflow.

Thus, we present the first approach to determine assembly instructions without prerequisite information about the objects' class and geometric structure. This is essential not only for authoring the physi-

- *Ana Stanescu and Peter Mohr are with Graz University of Technology. E-mail: {ana.stanescu|mohr}@icg.tugraz.at*
- *Denis Kalkofen is with Flinders University and Graz University of Technology. E-mail: kalkofen@icg.tugraz.at*
- *Dieter Schmalstieg is with VRVis GmbH and Graz University of Technology. E-mail: schmalstieg@tugraz.at*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

cal procedures, but also for detecting at runtime if users are correctly following the instructions or not. Hence, our main contribution is the first method for modeling multi-part assemblies purely from demonstrations, with no need for manual model editing or manual parameter tuning. We show our work's performance and versatility in an authoring-by-demonstration tool and in an AR guidance tool, which provides feedback on the user's performance in real-time, in addition to proactive instruction visualizations, based on the freshly extracted assembly information. This enables a large variety of objects and avoids clicking the "Go to the next step" button. In summary, our work makes the following technical contributions:

- We introduce the first approach for authoring multi-part assembly tutorials from user demonstrations without prior information.
- We introduce a new approach for topology change detection from volumetric scans, which is capable of providing real-time update rates in an assembly scenario.
- We evaluate the approach for volumetric change detection with an ablation study.

2 Related work

Our work is at the intersection of two areas: first, authoring and delivery of situated instructions in AR, and, second, methods for the reconstruction of dynamic scenes. In this section, we discuss previous work in these two areas and comment on similarities and dissimilarities to the approach presented in our paper.

2.1 Situated instructions

Previous work on delivering AR instructions has always assumed that object priors are available for the parts and the required manipulations. Oftentimes, the latter is assumed to be a linear sequence, although more sophisticated methods exist to model the entire assembly graph [17]. For a survey, see Lambert et al. [18]. We concentrate our discussion on the methods used to detect and track the parts in assembly applications. The detection can be roughly categorized into marker-based methods, methods that use natural visual features of the parts, and methods that operate on a volumetric representation (like we do).

Research that primarily focuses on investigating user interface aspects of AR instructions generally relies on instrumented scenes. Early work [29,45] used fiducials applied to parts, which can be problematic if parts are small or occluded. Even recent work, such as AuthorAR [40], which concentrates on versatile authoring, places fiducials on the parts and thus, introduces a complex preparation phase.

Other work relies on comparing natural visual features of the parts with prior models for detection and tracking. For example, Wu et al. [42] demonstrate a template matching approach (LINEMOD [13]) on RGB-D input, running at 9-17 Hz and operating on rather large, brightly colored parts. Similarly, Wang et al. [37] present an AR assembly system that combines a template matching approach on RGB video, enhanced with a probability model based on the assembly graph to disambiguate the parts. Yamaguchi et al. [44] use a method for contour segmentation and tracking (PWP3D [28]) together with a given assembly graph for extracting disassembly sequences from a video. They detect the presence of known parts in the video frames, followed by a region of interest (ROI) computation considering the contours of parts, hands, and tools. The recent work of Huang et al. [14] investigates the delivery of AR instructions that are adaptive to the user's performance. They rely on pre-registered regions of interest in a partially static environment combined with a neural network for detection of objects and object states that have been learned as a prior. Another recent approach by Chidambaram et al. [5] makes use of virtual tools manipulated with 3D controllers to author instructions, employing per-frame object detection to identify the present objects in the scene.

While feature-based model tracking is constrained by the necessity of having rich image features present within the object's image boundaries, template-based methods fail beyond a certain level of occlusion [13]. Besides, these methods assume that the object's detailed appearance and shape are known in advance.

In contrast to shape- and feature-based models, volumetric models are more suitable for representing the aggregation or combination of parts, since they are less constrained by a pre-determined shape or texture. A volume can be used as a generic representation of any intermediary model existing during an assembly procedure. Gupta et al. [11] realized that a volumetric representation can be used 1:1 for tracking the assembly of Duplo models, which consist of voxel-shaped blocks. A similar system was recently presented by Buttner et al. [4]. Both methods are only applicable to Duplo pieces. Miller et al. [21] developed another voxel-grid based approach for authoring block-based assemblies. Later approaches have been built up on top of LatticeFirst [1] to prototype an interactive error-detection capable system, also using toy building blocks.

Further AR instructions approaches that assume that 3D models are already available include the work by Makris et al. [20] and, more recently, Zogopoulos et al. [49]. Zhou et al. [46] and Güven et al. [16] propose state detection methods for AR technical support that assume that a user captures each state in advance to train machine learning models. Unlike these works, our method focuses on authoring AR instructions without 3D models or labeled data. The system proposed by Bhattscharya et al. [3] uses point clouds from a depth sensor to detect known parts from a database. Object segmentation is spatially and temporarily constrained by the absence of hands (users have to retract hands to trigger recognition). While these approaches use 3D models, Petersen et al. [27] aim to author instructions as video overlays from first-person-view human activity; they design a system for identifying significant events around the user's hand interactions, and later show the video snippets when the same user activity is detected. This is related to our work in its model-free authoring character. However, no 3D reconstruction or partial assembly graph is extracted.

Overall, we are not aware of any work that creates a volumetric model as a generic representation of an assembly process on the fly (i.e., model-free), as we do.

2.2 Reconstruction of dynamic scenes

In surveillance and robotics applications, large-scale reconstruction and change detection over longer periods is a recurring requirement. For example, construction engineers would like to visualize progress over time [50] or compare as-planned and as-built information [10, 25, 33]. These applications require change detection on 3D scenes, which can be determined from images [36] and volumetric models [7, 8, 12, 19]. Extracting the objects that have changed and re-detecting them (or other objects of the same types) are additional requirements that emerge in the context of the change detection problem [9].

For large scenes, obtaining a new scan and searching for changes is generally a time-consuming task that must be carried out offline. For example, the approach of Finman et al. [8] learns objects from changes in the surroundings over time. The scenario is that of a robot during multiple traversals of a large scene, and new objects are discovered by computing differences of point clouds. For each object, features are learned, so that the objects can be subsequently recognized in new scans. The approach is not aimed at real-time interaction, but at long-term object discovery. In contrast, Wasenmüller et al. [38] show that discrepancy checking can also work at millimeter-level precision and run in real-time for small and medium-sized objects. However, the requirement remains that both the scan of the current environment and the reference model must be fully available before comparison starts.

In contrast, SLAM using depth sensors makes it feasible to obtain dense reconstructions incrementally. The most popular representations used in dense SLAM are either volumes of a truncated signed distance function (TSDF) [15, 23] or collections of surfels [39].

While the original approaches for dense SLAM assumed a static scene, recent work attacks the problem of mapping for dynamic scenes. This area has similar goals as in surveillance or robotics, but is scaled down to room-size environments and operating incrementally. The ability to run in real-time and to reconstruct the scene *while* it is changing unlocks exciting new possibilities, but also incurs a steep increase in algorithmic complexity, making it necessary to enforce certain restrictions about the nature of the changes. These assumptions are not

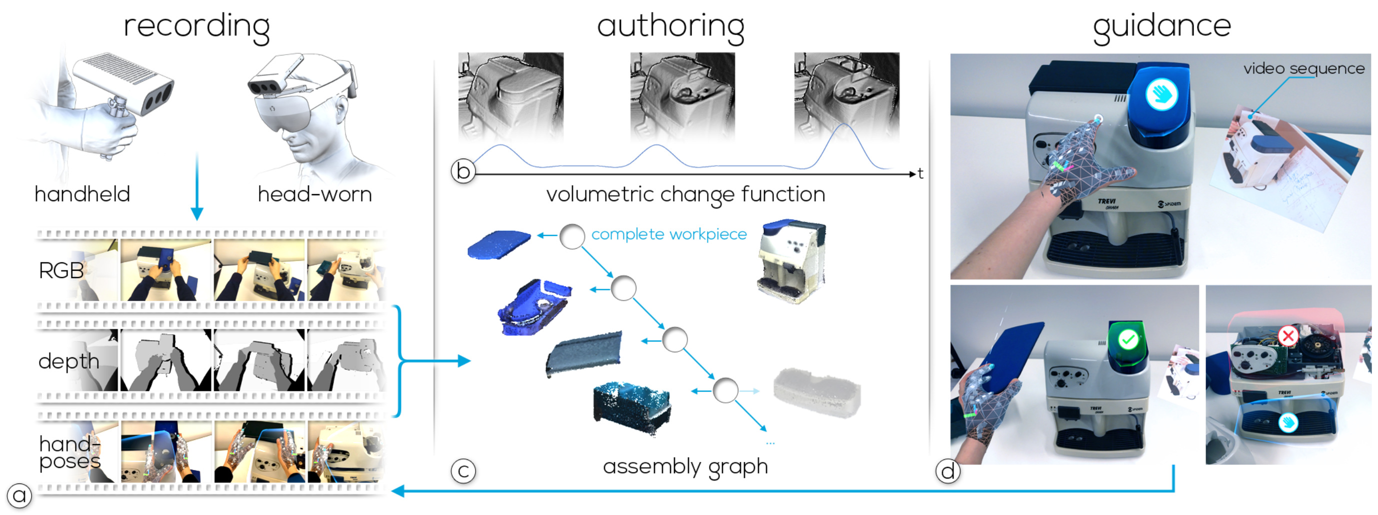

Fig. 2: (a) Our system uses input from an RGB-D camera for body and hand tracking as well as for SLAM-based camera tracking. (b) We analyze the volume that has been generated for tracking the camera to identify changes in the object's 3D reconstruction. (c) This analysis informs the extraction of 3D parts and the derivation of the sequence of instruction steps in the *authoring* mode. (d) We automatically generate visualizations that provide instructions in the *guidance* mode.

always explicitly stated in the literature, so we make an attempt of summarizing them here.

The first and most fundamental capability of SLAM for dynamic scenes is to *reconstruct a static background scene while ignoring any unwanted foreground objects*. Typical approaches detect foreground objects either using semantic segmentation [2] or geometric segmentation [24, 32], i.e., motion residuals. A more demanding capability is to *recover the shape and relative motion of one or multiple moving objects in addition to the background scene*. Moving object detection can either be done with purely geometric segmentation [30] or with the additional help of (expensive) semantic segmentation [31, 43]. Very recent work explores the use of an advanced space carving approach to enhance geometric segmentation [41].

However, all of these methods operate on a different – larger – scale compared to our scenario of a user assembling and manipulating a multi-part object. Typical foreground objects are sizable pieces of furniture, such as chairs. In contrast, assembly scenarios demand supporting object parts that are at least *one order of magnitude smaller*, and that are *representing potentially interlocking parts of an assembly*. These parts should be detected instantly once they are added, removed, or replaced. Previous work considering smaller foreground objects [31] relies on *learned categories* such as "cup" or "teddybear" and expects objects to be clearly seen and placed in an uncluttered environment.

An assembly scenario like ours does not fit well with these assumptions. The user is almost permanently interacting with the assembly, performing fast motion with the parts held in the hand, and causing heavy occlusions of the assembly. Approaches such as Co-Fusion [30] or RigidFusion [41] can segment and track novel dynamic objects, but only if they are slow-moving and not occluded. Consequently, we must design our approach to be more tolerant of these adverse conditions than previous work. We achieve such tolerance by *detecting topological changes with respect to previous scene states*, i.e., by considering the structure of the object at the points in time before and after the manipulation happened.

3 Overview

In this section, we give an overview of our system, highlighting the physical setup, the overall authoring procedure, and the underlying software components. The latter are further described in the following sections. Figure 2 provides an overview of the proposed workflow.

3.1 Physical setup

Our application relies on observing manipulations applied to a workpiece placed on top of a workbench from a single head-worn RGB-D camera, within a workspace of roughly 1 m^3 that is placed in front of the initial camera pose, on top of the workbench. We target a fully mobile setup, where the camera is incorporated into an AR head-mounted display (HMD). In our test setup, we relied on the Hololens 2, but used a high-resolution depth sensor (Kinect Azure) rather than the less capable built-in depth sensor of the HoloLens (Figure 2(a)). We fabricated a custom mount for the camera on top of the HoloLens. The Kinect is attached to a desktop PC and sends data via Wifi to the HoloLens.

While this configuration is fully operational, in its current iteration, it suffers from ergonomic drawbacks. For example, the Kinect must be tethered to a desktop computer and adds a weight of 440g to the headset. Therefore, we also support a configuration with minimal operator fatigue by mounting the Kinect on a tripod rather than on the HoloLens. Since the tripod is mobile, all results were created with the assumption of a moving camera, as would be required for fully mobile deployment. We did not consider a (stationary) multi-camera setup, which would likely improve reconstruction quality, but at the price of a much heavier infrastructure with correspondingly poorer scalability.

3.2 Authoring procedure

Our goal is to support authoring by demonstration. The user performs the required assembly steps in front of an RGB-D camera. Our approach is model-free, i.e., no prior model is given. Instead, 3D shapes of the object parts are collected during a demonstration along with the placement of the parts on the workpiece, and the order in which the parts are assembled is recorded as well. Thus, a single demonstration yields a linear list which constitutes a partial assembly graph. If multiple demonstrations are observed, and these performances deviate in their assembly order, a directed acyclic assembly graph is recovered automatically. Any path through this graph represents a possible assembly sequence. Finally, to better convey subtle activities not captured by the assembly graph (e.g., fastening of fixtures), we record detail views as indexed video sequences, which are cut from the RGB recording and associated with corresponding edges in the assembly graph. Figure 2(b) shows an illustration of the authoring.

3.3 Software components

The core function of our system is the analysis of changes to the volumetric reconstruction observed over time. We are interested *when* a change happened and *what* (i.e., which volumetric region) was changed. As we assume the manipulation process can be represented as an assembly graph, the **event detection** (Section 4) – the *when* question – is a prerequisite to the **content extraction** (Section 5) – the *what* question. More specifically, the difference between the reconstructions before and after the change event will reveal the 3D shape of the changed part. To provide detail about assembly actions, we further extract video sequences from the recordings, based on detected events.

We use this method both during authoring and during guidance. The **demonstration** mode lets us capture an assembly procedure and extract a description. The **guidance** mode (Section 6) leads a user through the procedure while displaying instructions and giving feedback based on the observed events (Figure 2(c)). In this mode, the AR system overlays instructions in the user's view. The instructions are interactive in the sense that they automatically respond to the user's activities. In particular, a visual cue is displayed at the position where the next part to be added to the assembly belongs to, and a video segment showing the required activity is shown as an inset. After detecting that the user has added a part, feedback (right or wrong step performed) is given.

4 Event Detection

Our method relies on identifying events over time. We refer to events as the frames when a user is adding or removing parts of an object. This is the core of our approach for both the authoring and for live guidance.

We detect events by analyzing a real-time scene reconstruction, based on a modified implementation of InfiniTAM [15]. Thus, the scene is stored as TSDF, represented in a volume of voxels v containing values $d(v)$ in the range $[-1,1]$.

4.1 Initialization

For observing the assembly sequence, we are only interested in the workpiece itself and not in the immutable background model (such as the bench on which the workpiece is placed). Therefore, we start by reconstructing a background model of the empty workspace. Afterwards, incoming depth frames are aligned to the background model via an iterative closest point (ICP) method. In doing so, we assume that a large portion of the pixels always shows the background, and relying on these pixels makes the camera tracking most stable. After the initialization converges, we disable further integration into the background by ignoring new depth values that are within a threshold (empirically set to 8.8 mm) from the background surface.

For change detection, we define a ROI in the workspace, in the shape of a cube that is axis-aligned to the voxel grid, with roughly the size of 1 m^3. After the background model is established, the user is expected to place the workpiece in the workspace. In the case of an assembly sequence, the user is prompted to place the first part to which additional parts are added in the assembly process. In the case of a disassembly sequence, the completed workpiece is placed in the ROI.

If a previous recording, and, thus, a 3D tutorial exists, the workpiece must be registered to the previous model reconstruction. This is the case either in guidance mode, when the AR instructions should be aligned with the workpiece to indicate the right placement of the next part, or when multiple demonstrations are being collected during authoring. We use the Open3D library [48] for registration in two steps, a fast global registration [47], followed by a colored ICP refinement [26].

Our implementation uses two tracking systems: (1) InfiniTAM (data from the Kinect Azure) and the (2) Hololens SLAM, each with its own world origin. The two coordinate systems are registered in the beginning of the pipeline using one of two methods. First, we can estimate the pose of a fiducial marker in either coordinate system, P_1 and P_2. We then compute the transformation from one world coordinate system to the other as $T_{1\rightarrow 2} = P_1 \cdot P_2^{-1}$. An alternative method uses the workpiece itself to register the coordinate systems. We automatically register the workpiece to the dense reconstruction of InfiniTAM as described above, then we manually register a virtual copy to the Hololens coordinate system with a pinching gesture.

Provided the alignment works as expected, instructions are registered precisely in the user's view, as can be seen for example in Figure 8, which shows portions of AR content recorded with the live capture video of the Hololens. The video-see-through stream delivered by the Hololens is a hardware feature which is nearly identical to what can be seen in optical see-through mode. The Open3D registration can fail if workpieces are poorly reconstructed. However, in our experiments, this was not an issue, and we find analyzing the accuracy of Open3D out of the scope of this paper.

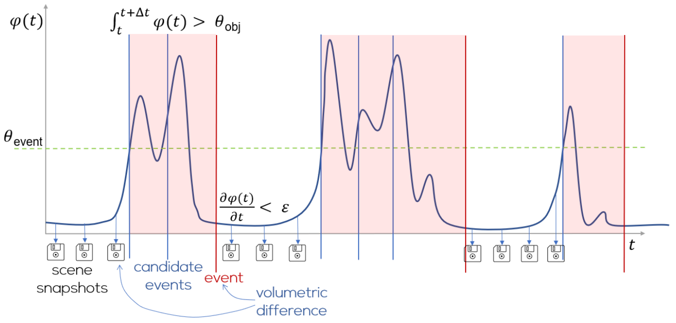

Fig. 3: The change detection relies on a change function that measures the volumetric difference between close frames. When the function forms a cluster of close peaks, an event is detected.

4.2 Candidate events

After initialization, we start observing the workpiece for change detection. Therefore, snapshots of the TSDF are saved periodically every k_0 frames. We set $k_0 = 38$ empirically, which corresponds to a 1.2 s interval at 30 Hz. We choose this interval because the integration usually takes a few seconds to converge. By spacing the snapshots, we ensure that sufficient change can occur in between, thereby reducing spurious detections. In Figure 3, the snapshots are visualized on the x-axis by "save" icons. In order to detect a change to the assembly of the workpiece, two questions arise:

1. At which point in time can we observe a pattern indicating a significant change in the reconstruction?
2. Which two snapshots can we use to characterize a state "before" and "after" the change, which let us identify a part that has been introduced or removed from the scene?

In order to address the first question, we use a *continuous change function* ϕ that is evaluated frequently, e.g., every $k_1 = 3$ frames. We use the change function directly on the TSDF reconstruction within the volume of interest. Our approach is inspired by Fehr et al. [7], but, unlike them, we apply it in real-time, on a smaller scale, and with a human operator present. An illustration of the change function can be seen in Figure 3. We define ϕ at time t as

$$\phi(t) = \sum_j [d(v_j(t)) - d(v_j(t-k_1)) < \varepsilon_1],$$

where $v_j(t)$ is the j-th voxel of the TSDF volume ($0 \le j < J$) taken at time t, ε_1 is a small constant, and $[.]$ is the Iverson bracket. We also apply a moving average with a window size of 3 to smooth the curve and eliminate noise, which is equivalent to a one-dimensional convolution $K * \phi(t)$ with a uniform kernel K. If ϕ is larger than a configurable threshold θ_{event}, a change event candidate is reported. After a change event, the detection is disabled for a cool-down period of 20 frames to suppress duplicate events.

Our modified InfiniTAM also keeps track of the empty space in our scene, so we can reason about seen and unseen space. This ability is important when an exploring camera discovers previously unseen portions of the workspace. In such a case, the event detection should not be triggered. Discovery of unseen space can also take place after a large part has been removed, revealing an area at a different depth that is not adjacent to the removed part in object space. In this case, suppressing event detection would lead to a false negative. Thus, we use a more conservative threshold for this criteria.

The candidate events, depicted with vertical blue lines in Figure 3, indicate that a significant change is happening. In the following, we will describe how to identify the optimal time to extract this change, answering the second question.

4.3 Temporal volumetric differences

We are interested in the new part, i.e., the portion of the volume occupied in the new state that was not occupied in the old state, (or, in the case of a disassembly, by the volume that was occupied and became free). Therefore, we need to choose the two moments in time, t^- and

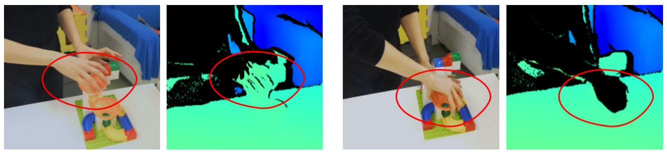

Fig. 4: Under- and over-segmentation of the user's hands – the body segmentation sometimes has challenges around the depth discontinuities of the body. While a too large mask is not a problem, an under-segmentation can lead to noise in the reconstruction, as small hand areas can be wrongfully integrated.

t^+, that enclose the peaks representing the addition or removal of the part, such that $t^- = t^+ - \Delta t$. We choose t^- as the closest snapshot before the detected candidate event. However, if this snapshot is very close to the candidate event, the second-to-last snapshot is considered instead. This heuristic ensures that the change curve has not yet started significantly increasing, and we capture the entire period of change.

Since volumetric integration of new depth information relies on an averaging scheme [15, 23], we search for t^+ forwards from t^- until the change curve becomes flat, i.e.,

$$\frac{\partial \phi(t^+)}{\partial t} < \varepsilon_2,$$

with the derivative approximated using backward differences. A small value indicates that the reconstruction has ended, so the shape can be extracted. Moreover, we want to make sure that the candidate events did not just capture some noise in the reconstruction, and that the volume of the change area is significant. Thus, we compute the integral

$$\int_{t^-}^{t^+} \phi(t) \mathrm{d}t,$$

which accumulates the change within a period of Δt frames. If this quantity is larger than the object threshold θ_{obj}, a change event is validated. After the validation, the detection is disabled for the same cool-down period of 20 frames as mentioned before. Since the integral takes into account a number of voxels that have changed during a time period, we make sure not to count the same voxel multiple times.

Note that we must assume that the reconstruction is incomplete in both the old and the new state, i.e., certain portions of the scene, such as the backside, have never been seen by the camera, or insufficient time was allotted for the reconstruction to fully converge to the true surface. Therefore, we need a volumetric difference that is robust to small inconsistencies. In the case of object addition, we achieve such robustness by copying the portion of the new volume at time t^+ that deviates by more than ε_1 from the old volume at time t^- into a difference volume $\Delta d_j(t)$; in case of a removal, we copy the part at time t^-:

$$\Delta d_j(t) \underset{j\in J}{=} \begin{cases} d(v_j(t^+)) \vee d(v_j(t^-)), & \text{if } |d(v_j(t^+)) - d(v_j(t^-))| > \varepsilon_1, \\ -1, & \text{otherwise.} \end{cases}$$

Since the integration is already disabled in locations that are very close to the surface of the background model, we expect that no parts of the background model contaminate the resulting difference volume. Consequently, the resulting difference $\Delta d(t)$ represents the reconstruction of the part that was added or removed at time t.

4.4 Threshold calibration

The addition or removal of parts results in a change curve with a peak when such an event happens. However, manually adjusting the sensitivity of detection for each specific workpiece would be tedious. Therefore, we auto-calibrate the event thresholds to a specific workpiece after adding the base piece. The thresholds θ_{event} and θ_{obj} are set to 1 % and 0.2 % of the base object size, respectively.

4.5 Human body segmentation

As we want to exclude the user's body from corrupting the SLAM map, we segment the body before detecting events. Pixels of the RGB-D camera output are selectively discarded via a body mask, which uses

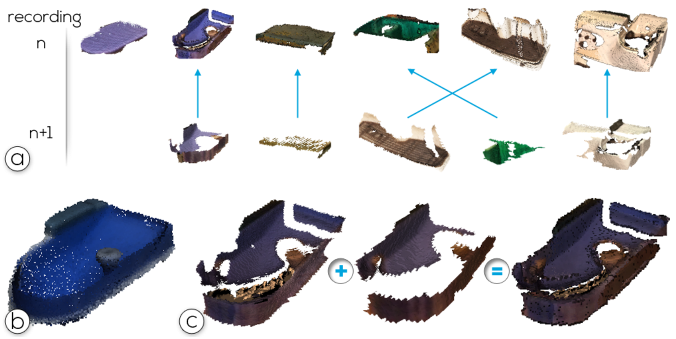

Fig. 5: Matching strategy for diverging sequences: (a) Two examples of disassembly sequences extracted from the Coffee Maker. The order of the removal of the drip tray and of the water tank are swapped, leading to branching in the assembly graph. (bottom row) Object part reconstruction results. (b) A piece of the Coffee Maker obtained by careful manual scanning. (c) Two partial reconstructions as obtained by our method, and the result of our merging strategy.

the hand tracking feature provided by the Hololens, when the Kinect is mounted on the HMD, and the Kinect body tracker, when mounted on a tripod at a sufficient distance so the user's body is mostly visible.

Unfortunately, both segmentations are not always pixel-accurate. If the user performs fast motions, the body mask may lag behind. Since we do not want to delay the volumetric processing to match this lag, we dilate the body mask with a 3×3 kernel to make it more conservative. The dilation eliminates most of the disturbances from the user's body appearing in the input frame, except for minor amounts of isolated pixels (see Figure 4). Since we use the SLAM map outside of the immediate vicinity of the workpiece only for camera tracking and never update it, we may safely ignore the problem of body tracking errors.

However, we need an additional safeguard to prevent noisy pixels from creeping into the reconstruction of the workpiece itself. These are primarily caused by the user's hands holding a part close to the workpiece during adding or removing the part. Therefore, we create additional filtering constraints with larger safety zones around the user's hand. We use axis-aligned bounding boxes of size $17 \times 17 \times 17$ cm, centered around the user's hand center, which is assumed to lie 15 cm away from the tracked wrist along the direction indicated by the lower arm. The voxels within the hand bounding volumes are disregarded during event detection. Thus, the small regions of the workpiece near the user's hands may be temporarily filtered out during workpiece manipulation, but the reconstruction converges swiftly in these areas as soon as the hands move away. Regarding the tracking performance of body parts with varying skin tones, we use tools provided by Microsoft. The Hololens 2 uses the depth sensor for hand tracking, which works independent of skin tones[1]. Microsoft also states about the Kinect Azure body segmentation that the training data was "ethically balanced (...) included varying height, body size, and skin tone."[2].

5 Content Extraction

Our approach extracts information from the author's demonstration at the moment when an assembly event occurs. This data is further filtered and processed to synthesize AR instructions.

Part geometry The outcome of the event detection is a series of part reconstructions, annotated with time and place where the part was added or removed. Before we can use a difference volume representing a detected part, it is further refined to eliminate any residual noise. First, we apply a 3D morphological opening on the TSDF representation of the part. Second, a triangular mesh is extracted via marching cubes. Third, over-tessellated areas of the mesh are simplified by vertex clustering. Fourth, a connected component analysis eliminates noisy sub-meshes with few triangles (we empirically set the threshold to 30 % of the entire mesh of the respective part).

[1] https://docs.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/research-mode
[2] https://github.com/microsoft/Azure-Kinect-Sensor-SDK/issues/1270

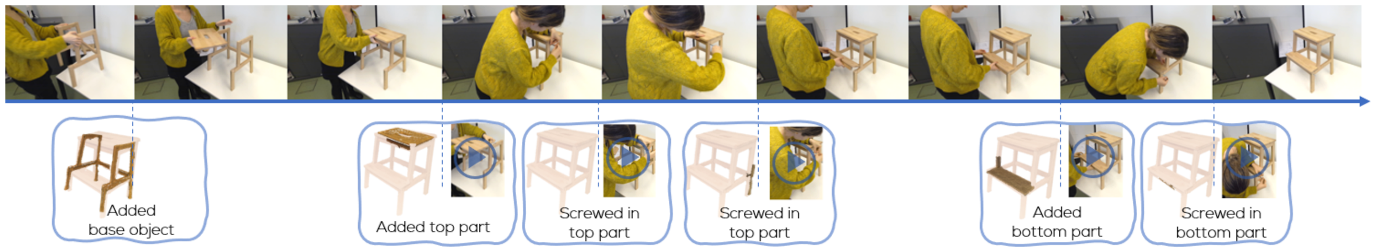

Fig. 6: Sequence containing some actions where the change in reconstruction is challenging to detect, but our systems still extracts video snippets of actions, representing the user screwing in a metal screw after adding object parts.

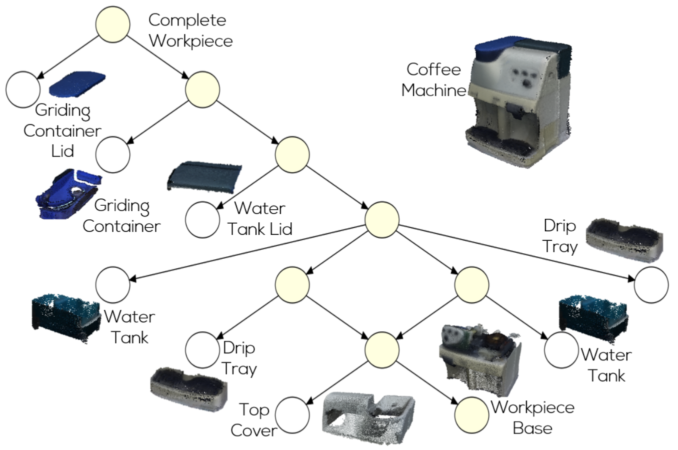

Fig. 7: Example obtained assembly graph of the Coffee Maker, after merging three different recordings, where the step order differs.

If only one demonstration session is recorded, the assembly graph (or, more precisely, the known portion of the graph) is just a linear sequence. Here, the part order is the essential information, but it is still helpful to record the timing and the location where the part is added or removed, if we want to synthesize instructions later.

If multiple recording sessions are available, they may diverge in the order in which parts are manipulated, thereby revealing additional branches of the assembly graph. We merge multiple sequences in the following manner: First, the initial states have to be registered, so that we can establish spatial correspondences between the recorded sequences. This registration happens in the same way as described in Section 4.1. Then, the individual parts need to be matched. The re-localization of InfiniTAM could be used in place of our registration method; however, it is optimized for a moving camera in a static scene and performs poorly when trying to register our workpieces.

Therefore, given two assembly sequences, each as a list of extracted parts, we compare every part in the first list with every part in the second list using a distance measure provided by the Open3D library [48]. The matching is performed step by step and relies on the spatial presence and layout of the pieces, quantified by a measure that accounts for the distance from each point in the first point cloud to each point in the second point cloud. If the matching score exceeds a threshold, the parts are considered as having the same label. Such matches are merged by concatenating their point clouds. We show an example of a part merge from multiple recordings, namely the grinding tank of the coffee maker in Figure 5. The labels are used to build the partial assembly graph. We can imagine the matching procedure in the shape of a bipartite graph, as shown in Figure 5. If there are intersecting edges, then the two assembly sequences have diverged, and we represent this as an intersection in the partial assembly graph.

The merging step also serves as a filter for false positives. If a part has only poor scores (large distances), no corresponding part has been found. Therefore, the part is either a false positive, or it has been discovered only in one recording, while the event detection for the part failed in the other recordings. To recover from such a condition, the authoring concludes with a feasibility check and prompts the operator to resolve any ambiguous cases.

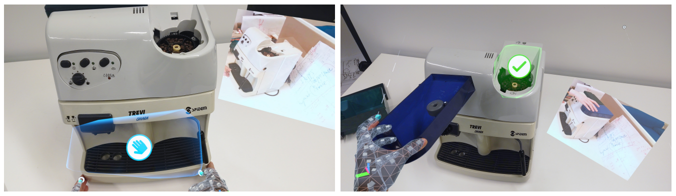

Fig. 8: The guidance mode visualization in first person view as shown on the Hololens 2: On the left, a part is in *todo* state, and on the right side, a part is in *correct* state, as a result of the interactive guidance detecting a correct assembly step.

To provide an overview visualization, we furthermore allow exporting the partial assembly graph to a format understood by the graph visualization software GraphViz [6]. An example of output sequences and the corresponding partial assembly graph can be seen in Figure 7.

The recordings from the guidance mode can be used as information sources as well. However, since it is not guaranteed that the user performs the correct steps during AR guidance, only the part reconstructions are considered, while the assembly graph is not modified.

Video sequences In addition to preparing the part geometry and the assembly graph, we also extract video segments showing each assembly action. We compile all frames between t^- and t^+ of each event into a short video loop in order to preserve subtle visual cues that contribute in important ways to the understanding of the activity [44]. When the event detection has challenges with extracting small parts or short actions, which are suppressed by the noise filter, the video loops extracted alongside the assembly graph fill in the gaps. The video frames are also cropped around the position of the object, projected into image space. For instance, assembling the stool in Figure 6 requires screwing in the top and bottom parts. While the screws are not detected as parts, videos of the screwing actions are retrieved. Figure 8 demonstrated how the extracted video sequences are visualized on a billboard during AR guidance.

6 Guidance

The guidance mode uses the extracted information to deliver real-time AR instructions. The visualization shows hints for each current object part, spatially registered to the object. The area is computed as the 3D convex hull of the extracted part model. The color of the highlight, along with a status icon, indicates the three possible states, as shown in Figure 8: blue (assembly to-do), green (step is correct), or red (step is incorrect). The icons are placed at the center of the bounding box of the 3D convex hull of a part reconstruction; the video snippet is played on a canvas next to the object. Along with this situated visualization, a floating video billboard is shown to the user next to the object [44] indicating the action that needs to be performed.

An error detection mechanism provides interactive feedback, based on the relative spatial layout of the parts. When a piece is added or removed, the extracted mesh is filtered and a spatial distance to the expected part is computed based on point clouds. If the parts are similar and an event is detected, the addition or removal is validated and the part enters a green (step is correct) state. Otherwise, an error message is displayed, and the part that needs to be handled remains in the blue (assembly to-do) layout.

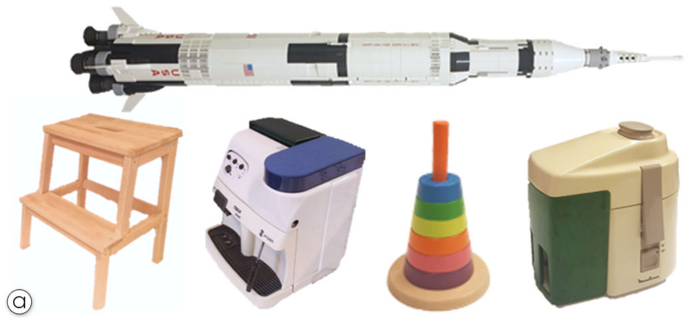
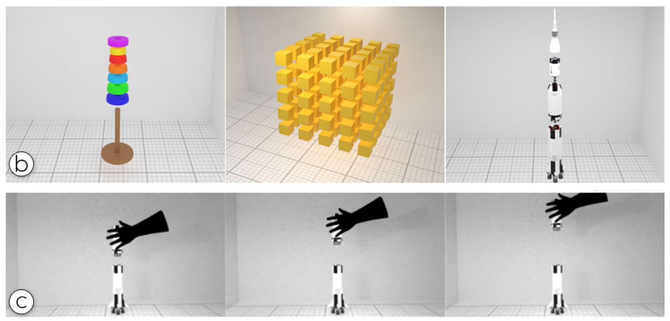
Fig. 9: Evaluation objects. (a) Real objects (top, left to right) Rocket, Stool, Coffee Maker, Tower, and Juicer. (b) We use three virtual datasets: a virtual stackable tower toy, a subdivided cube, and a Lego rocket. An exploded view of each object shows parts that we use in our evaluation. (c) To simulate the user interaction, we also use an animated virtual hand model. The hand contour creates a mask for perfect segmentation.

7 Evaluation

We present an evaluation that demonstrates the approach's robustness by testing with multiple objects and demonstrations, either captured with the Kinect or using synthetic scenes.

7.1 Experimental setup

In order to run qualitative experiments, we use a Kinect Azure camera, which we chose because of its high depth resolution (640×576). We let the camera warm up before we start the experiments to ensure more precision of the measurements [35]. The automatic white balance of the camera is disabled, and the lighting is manually set to a fixed value so that it remains consistent throughout the whole session, even if the camera position and the light sources are changing. We run most of the experiments at 30 Hz in "unbinned narrow field of view" mode, as this offers the highest quality of the depth maps [35], and the reconstruction heavily relies on the depth values.

The system runs on a desktop PC with an NVIDIA RTX 3080 Ti GPU, an Intel Core i7 CPU, and 32 GB of RAM, and streams to the HoloLens via WiFi. We use InfiniTAM v3 for mapping and tracking, with modifications, such as the aforementioned background-only tracking mode. The core functionality (event detection, volumetric differences, morphological operations etc.) is implemented in CUDA and shares the GPU with InfiniTAM.

For showing the capabilities of our system, we choose the following objects, as shown in Figure 9 (a):

- *Coffee Maker*: A coffee machine consisting of a base part and six removable parts, that can be built in a different order.
- *Stool*: A wooden stool that can be assembled by the user. The parts are thinner but larger.
- *Tower*: An example of a stackable "Tower of Hanoi" toy that is symmetric and has few features.
- *Rocket*: A Lego toy with little texture and many small bricks. However, we use groups of bricks as parts for our experiments.
- *Juicer*: An appliance with multiple parts of various sizes.

The objects only contain parts that can be either added or removed. In our evaluations, we do not include any events containing undetectable parts or more complex actions like pulling, screwing, etc.

The average runtime of the change detection algorithm with the aforementioned experimental setup for the captured data is 25.7 milliseconds per frame. The communication between our headset application (Unity) and the PC (C++) runs over UDP on WiFi, which has less than 5 milliseconds of latency. Some events trigger asynchronous processing in background threads. For initial registration, the user must wait an average of 3.6 s until the interactive mode starts. For error checking, after the event is detected, the user must wait an average of 0.8 s until the performed step is validated and the result is displayed (measured on the Coffee Maker). The AR rendering is always refreshed at full HMD frame rate; only the display of new results is delayed for a short period due to the event validation occurring after the change function peak. This accounts for the time needed for the reconstruction to converge.

Dataset	Mode	Voxelsize	TP	FP	FN	Precision	Recall
Coffee Maker	Disassembly	0.003	6	0	0	1.00	1.00
Stool	Disassembly	0.003	2	0	0	1.00	1.00
Tower	Disassembly	0.002	6	0	0	1.00	1.00
Rocket	Disassembly	0.003	4	0	0	1.00	1.00
Juicer	Disassembly	0.003	5	1	0	0.83	1.00
Coffee Maker	Assembly	0.003	4	0	2	1.00	0.66
Stool	Assembly	0.003	2	0	0	1.00	1.00
Tower	Assembly	0.002	5	0	1	1.00	0.83
Rocket	Assembly	0.003	3	0	0	1.00	1.00
Juicer	Assembly	0.003	5	0	0	1.00	1.00

Table 1: Precision and recall of the change detection function in identifying assembly or disassembly events for the used objects.

7.2 Quantitative results

We evaluate the function for change detection with recorded datasets containing our test objects. We measure the accuracy of the detection of assembly or disassembly events. For this evaluation, we manually label when the event happens by choosing the first frame where the part touches (or disconnects from) the workpiece. We consider a detection successful if the addition or removal of a part is captured between t^- and t^+, and no other event detection falls into that period. Results are shown in Table 1. Precision and Recall are computed with regards to the number of True Positives (TP), False Positives (FP), and False Negatives (FN), as Precision = TP/(TP + FP), Recall = TP/(TP + FN).

Figure 10 shows the change functions in disassembly (top) and assembly mode (bottom), respectively. When an event occurs, a pattern of two neighboring peaks usually encloses a valley. This reflects the behavior of the TSDF when a part is removed, namely, a sudden change, followed by the slow integration of the piece. The first peak can contain noise from the potentially unreliable hand mask. Other sources of noise include areas in the depth map that cannot be reconstructed due to reflections, or along depth discontinuities, where spurious pixels can appear. Moreover, slight deviations of the tracking can happen when larger parts of the background are occluded. The overall results show that our system is able to recover from all these conditions.

Our approach is able to correctly detect most events, but can occasionally deliver false negatives. For example, in the Coffee Maker assembly sequence, two events are missed. One of them is the water tray located on the bottom of the assembly, close to the workbench underneath. Since we disable integration within a short distance from the background surface, parts directly touching the background (the workbench) may not be fully reconstructed. In the case of the water tray, its flat bottom is truncated. Consequently, it is not recognized as a valid part that has been added.

7.3 Synthetic data

We chose to also evaluate our system on synthetic data, in order to obtain results in an environment where we can control the amount of noise. Synthetic data lets us measure system performance under idealized conditions (no noise, no reflective surfaces, no camera jitter). The datasets are generated by rendering frames of manually created animations. The animations have the same format as the captured data: a set of temporally ordered color and depth frames, with body parts masked out, and annotated with the positions of the wrists.

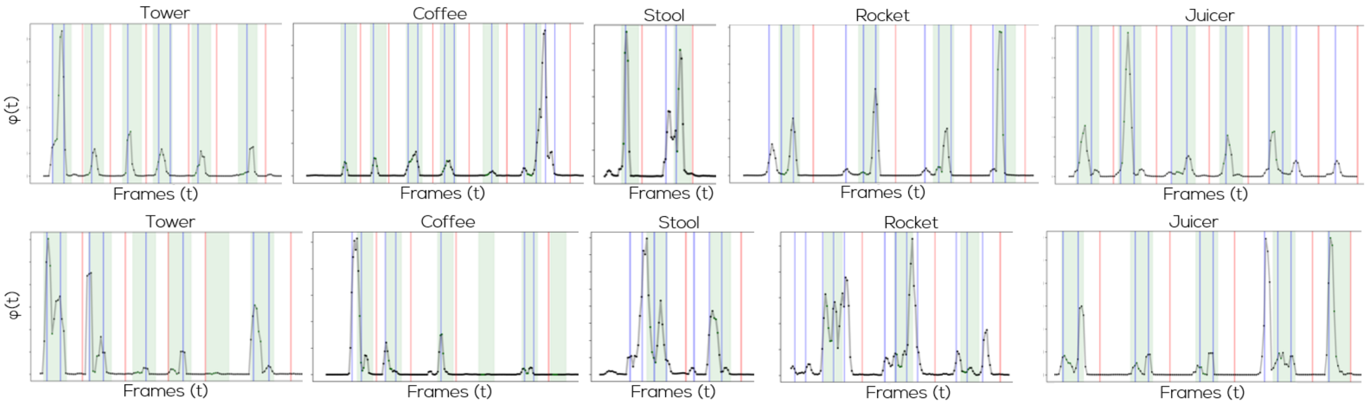

Fig. 10: Top: Unfiltered outputs of the change function during disassembly (from left to right) of Tower, Coffee Maker, Stool, Rocket, and Juicer, and outputs from the assembly mode on the bottom. The green highlights indicate the ground truth events, the blue lines mark the candidate events as detected by the system, and the red lines mark the validation event.

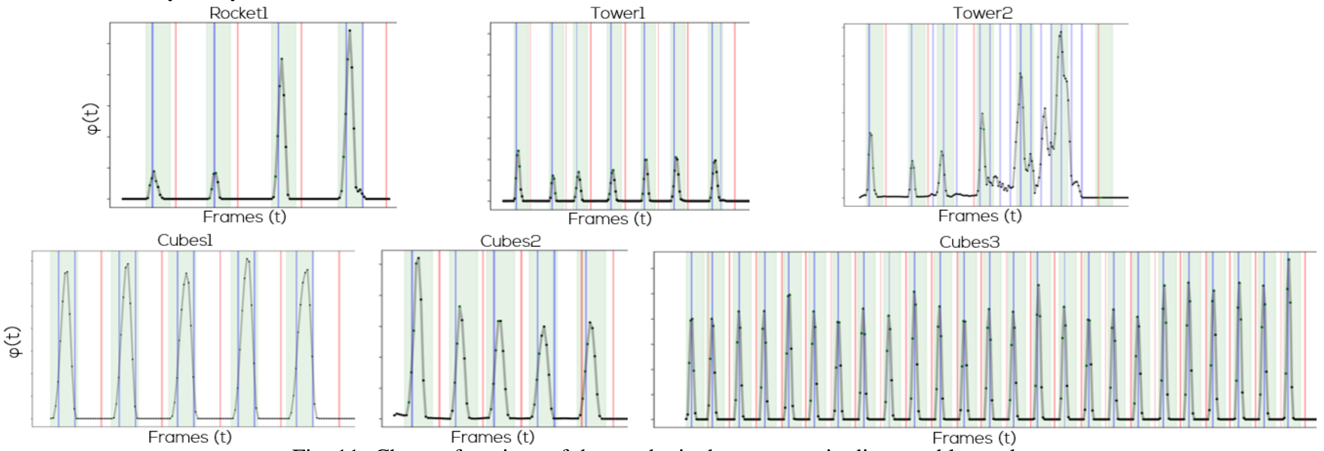

Fig. 11: Change functions of the synthetic datasets, ran in disassembly mode.

We chose to vary the following parameters: (1) type of camera movement, (2) voxel size, (3) part size, and (4) hand presence. The camera movement can be static, scanning (follow a circular arc path of 45° around the assembly during the initialization stage, after the base object has been placed) or continuous (follow the aforementioned path during the assembly procedure). We vary the voxel size as well as the part size to investigate the system's ability to extract small objects. Finally, hand presence offers insight on how occlusions caused by the hands impacts event detection and reconstruction quality. To this aim, we render a virtual hand into a mask suppressing depth pixels (Figure 9 (c)) to obtain results akin to a perfect hand segmentation.

We generate the synthetic data in Blender, where we render RGB-D frames for different camera trajectories. We also output the intrinsics of the virtual camera, since our system needs a calibrated camera to run. In Table 2 below, the tested scenarios and objects are shown.

We use three objects, as shown in Figure 9(b): The baseline is a simple cube with a side length of 30 cm, composed of 5×5 smaller cubes. In the "slices" configuration, the object is disassembled into five horizontal parts, while, in the "subparts" configuration, every small cube component from the top slice is removed one by one. The other two test objects are CAD versions of the Tower and the Rocket[3], which we also used as physical test objects. The rocket is disassembled in the same way as in the real scenario, by splitting it into four pieces. Hence, we are able to compare the test results on synthetic objects with their real-world counterparts, giving us an indication of the influence of real-world sensor noise.

The results of the evaluation can be found in Table 3. The system is able to extract most of the parts, even each small cube components of the Cubes3 dataset. The curves of the change function with the synthetic data shown in Figure 11 look cleaner, since there is no noise being integrated. In the output for Rocket1, one can see that the peaks are also proportional to the size of the removed pieces. The only noisy result is the dataset Tower2. Here, a continuous camera movement is present, and a very smooth, uniform wall as background is used for tracking. The ICP-based tracker fails to converge after some frames using these background voxels, causing the spikes in the second half of the run. The resulting erroneous camera poses cause the integration of noise, lowering precision and recall. For such challenging conditions, using visio-inertial tracking (e.g., the Hololens tracking) would likely yield more stable results. However, we used open-source InfiniTAM during authoring to have better control than possible with the closed-source Hololens SLAM.

We also experimented with a larger voxel size of 3 millimeters for Cubes3. However, this experiment results in the 25 top cubes not being detected, in contrast to all being detected for the 2 millimeter resolution.

Synthetic Dataset	Move	Voxelsize	Hands	Granularity
Cubes1	static	0.002	no	slices
Cubes2	scan	0.002	no	slices
Cubes3	static.	0.002	no	subparts
Tower1	static	0.002	no	all parts
Tower2	cont.	0.002	no	all parts
Rocket1	static	0.003	yes	large sub-assemblies

Table 2: Setup of the used synthetic datasets, with varying parameters in terms of camera movement, hand presence, part granularity and voxel size (in meters) during event detection.

Dataset	Mode	TP	FP	FN	Precision	Recall
Cubes1	Disassembly	5	0	0	1.00	1.00
Cubes2	Disassembly	5	0	0	1.00	1.00
Cubes3	Disassembly	25	0	0	1.00	1.00
Tower1	Disassembly	7	0	0	1.00	1.00
Tower2	Disassembly	3	1	4	0.75	0.43
Rocket1	Disassembly	4	0	0	1.00	1.00

Table 3: Precision and recall of the change detection function in identifying disassembly events for the synthetic objects.

[3]https://grabcad.com/library/lego-nasa-apollo-saturn-v-21309-1

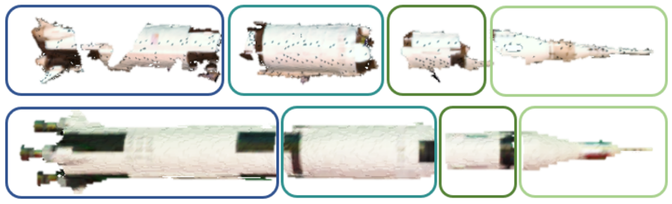
Fig. 12: Extracted pieces from the rocket object (top) captured by the RGB-D camera and (bottom) from a synthetically generated RGB-D dataset, Rocket1.

7.4 Qualitative results

Reconstruction quality Our part reconstruction does not necessarily aim to deliver complete geometric models. We extract the parts with visualization purposes in mind. Extracting a partial reconstruction is enough for an estimation of the location and of the rough size of a part, required for the visualization of spatially registered hints, as seen in Figure 8. Unsurprisingly, our results reveal that synthetic input leads to more complete reconstructions, even just from one view (static camera), after a disassembly procedure. For example, Figure 12 shows the extracted parts of the synthetic Rocket. Sample frames of this sequence can be seen in Figure 9 (c).

After performing the matching as described in Section 5, the parts are completed by new observations. In Figure 5 we show a part of the Coffee Maker that is completed after multiple runs: the ground truth (obtained by manual scanning), the part extracted by our method in two different runs in disassembly mode, and the merged models.

Public dataset We also run our system on a public dataset introduced by the Co-Fusion work of Rünz et al. [30], namely, the sequence that contains topological changes. The calibration step cannot be performed on this dataset, because no base object is available. As a workaround, we heuristically set a threshold θ_{obj} that accounts for a minimum changed volume of 40×40×20 millimeters, scaled by the voxel size. With these preparations, our system was able to detect all events and extract partial reconstructions and video snippets. However, since our use case requires the ability to handle heavy hand occlusion, we do not track assembled parts individually. Consequently, our reconstruction quality is less dense than the one obtained with Co-Fusion. Moreover, the hand segmentation is ignored; therefore, the reconstruction of the hand is also present in the scene (Figure 13).

Additionally, we also ran Co-Fusion on our own dataset, Coffee Maker Disassembly. This experiment did not deliver satisfactory results; individual pieces cannot be clearly segmented by Co-Fusion. The failure of Co-Fusion is likely caused by the faster motions, as well as to the parts being smaller, and to more hand occlusions. We ran the algorithm with its default parameters as provided in the source repository. After manually adjusting thresholds to allow smaller regions to be considered as parts, Co-Fusion was able to partly detect the base object, but not the individual parts.

8 Discussion

While our implementation demonstrates the feasibility of assembly sequence authoring and guidance, the quality of results is affected by a number of (known) limitations of the underlying technology, i.e., a time-of-flight camera feeding into volumetric fusion.

Concerning the workspace size, the system is capable of processing workpieces that fit atop a workbench, which is reasonable for many assembly tasks. However, the tabletop workspace does not benefit much from the ability to move the camera, as the user's mobility is limited by the need to stay close to the table. A more severe restriction comes from the inability to handle small parts, such as screws or fasteners, which are not reconstructed properly due to the limited spatial resolution of the RGB-D camera. Besides, black, transparent, or highly reflective objects pose challenges for time-of-flight sensors operating with infrared light.

The body segmentation itself has a significant influence on the reconstruction pipeline as well. If a very agile user frequently moves in and out of the camera's field of view, the body tracker may occasionally fail.

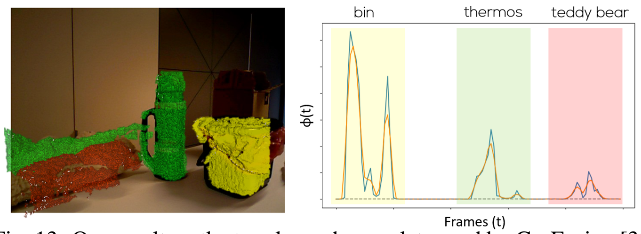

Fig. 13: Our result on the topology change data used by Co-Fusion [30]. Note that our system needs a hand segmentation, but this particular example was run directly on the dataset with no hand mask information, hence the hand reconstruction on the left side of the thermos flask. On the right side, the change function of this dataset shows three peak clusters that correspond to the addition of the three objects.

In this case, the reconstruction is not only polluted with a few noisy pixels, but with larger portions of the body, which can be larger than typical parts used in the assembly. If the pollution of the reconstructed volume becomes too significant, the tracking suffers, and artifacts may creep into the reconstruction of the parts. An overall higher framerate of both the camera and of the computing pipeline would likely reduce these problems without requiring changes to the algorithms.

Reconstruction by volumetric fusion tends to smooth edges, which can represent an additional challenge for small or thin parts, which rely on features at a scale that cannot tolerate the smoothing applied by the SLAM system. In unfortunate circumstances, edge smoothing combined with our noise filtering leads to small objects being suppressed entirely. A sensor with a higher resolution may ameliorate the issue.

We support only rigid parts, and the volumetric difference procedure cannot deal with parts that need to be deformed to fit other parts. For example, when building the Stool in Figure 9(a) and placing the top part, the user needs to slightly pull apart the sides in order for the top part to fit in place. This action can confuse the difference computation and result in hallucinated parts being extracted.

9 Conclusion and Future Work

We introduce a method for automatically extracting a multi-part reconstruction and a partial assembly graph driven by change event detection. We demonstrate that our approach has high practical value for instruction authoring and delivery with an AR application. It works with commodity hardware and lets operators perform authoring by demonstration with ease of use.

Currently, we support either an assembly or disassembly mode, which has to be chosen before using the system. In the future, this can be extended to support both operations in the same sequence, for example, when a group of parts is replaced during a repair procedure.

Our system supports authoring by an expert or interactive guidance for a user that follows the tutorial. Another possible application is to create an AR guidance mode that is a mixture between the two: The user would have the freedom to choose whether to enter authoring mode or guidance mode. A user could decide, during the guidance, that they want to add a new branch to the tutorial. This could be useful in the case when a user wants to build a new configuration of Lego blocks, or when an expert is specialized in a particular sub-assembly. In these cases, one may follow the guidance up to the desired level of completion and then switch to authoring mode.

Other interesting directions are support for annotations during user demonstration and guidance on how to hold parts in front of the camera to improve the reconstruction [22]. While our experiments did not indicate that the additional guidance is a critical aspect for novice users, the camera observing the parts from multiple viewpoints during manipulation could improve the reconstructions if the parts are large and visible enough to be tracked.

Acknowledgments

This work was enabled by the Austrian Science Fund FWF (grant no. P30694 and I5912) and the Competence Center VRVis, which is funded by BMK, BMDW, Styria, SFG, Tyrol and Vienna Business Agency in the scope of COMET - Competence Centers for Excellent Technologies (879730) which is managed by FFG.

References

[1] J. Alves, B. Marques, M. Oliveira, T. Araújo, P. Dias, and B. S. Santos. Comparing spatial and mobile augmented reality for guiding assembling procedures with task validation. In *IEEE International Conference on Autonomous Robot Systems and Competitions*, pp. 1–6, 2019. doi: 10.1109/ICARSC.2019.8733642

[2] B. Bescos, J. M. Facil, J. Civera, and J. Neira. Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3:4076–4083, 10 2018. doi: 10.1109/LRA.2018.2860039

[3] B. Bhattacharya and E. H. Winer. Augmented reality via expert demonstration authoring (AREDA). *Computers in Industry*, 105:61–79, 2019. doi: 10.1016/j.compind.2018.04.021

[4] S. Büttner, A. Peda, M. Heinz, and C. Röcker. Teaching by demonstrating–how smart assistive systems can learn from users. In *International Conference on Human-Computer Interaction*, pp. 153–163. Springer, 2020. doi: 10.1007/978-3-030-50344-4_12

[5] S. Chidambaram, H. Huang, F. He, X. Qian, A. M. Villanueva, T. S. Redick, W. Stuerzlinger, and K. Ramani. Processar: An augmented reality-based tool to create in-situ procedural 2D/3D AR instructions. In *Designing Interactive Systems Conference 2021*, pp. 234–249, 2021. doi: 10.1145/3461778.3462126

[6] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull. Graphviz and dynagraph—static and dynamic graph drawing tools. In *Graph drawing software*, pp. 127–148. Springer, 2004. doi: 10.1007/978-3-642-18638-7_6

[7] M. Fehr, F. Furrer, I. Dryanovski, J. Sturm, I. Gilitschenski, R. Siegwart, and C. Cadena. TSDF-based change detection for consistent long-term dense reconstruction and dynamic object discovery. In *IEEE International Conference on Robotics and Automation*, pp. 5237–5244. IEEE, 2017. doi: 10.1109/ICRA.2017.7989614

[8] R. Finman, T. Whelan, M. Kaess, and J. J. Leonard. Toward lifelong object segmentation from change detection in dense RGB-D maps. In *European Conference on Mobile Robots*, pp. 178–185. IEEE, 2013. doi: 10.1109/ECMR.2013.6698839

[9] F. Furrer, T. Novkovic, M. Fehr, A. Gawel, M. Grinvald, T. Sattler, R. Siegwart, and J. Nieto. Incremental object database: Building 3D models from multiple partial observations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6835–6842. IEEE, 2018. doi: 10.1109/IROS.2018.8594391

[10] M. Golparvar-Fard, F. Pena-Mora, and S. Savarese. Monitoring changes of 3D building elements from unordered photo collections. In *IEEE International Conference on Computer Vision Workshops*, pp. 249–256, 2011. doi: 10.1109/ICCVW.2011.6130250

[11] A. Gupta, D. Fox, B. Curless, and M. Cohen. Duplotrack: a real-time system for authoring and guiding duplo block assembly. In *Proceedings of ACM Symposium on User Interface Software and Technology*, pp. 389–402, 2012. doi: 10.1145/2380116.2380167

[12] M. Halber, Y. Shi, K. Xu, and T. Funkhouser. Rescan: Inductive instance segmentation for indoor RGBD scans. In *2019 IEEE/CVF International Conference on Computer Vision*, pp. 2541–2550, 2019. doi: 10.1109/ICCV.2019.00263

[13] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, 2011.

[14] G. Huang, X. Qian, T. Wang, F. Patel, M. Sreeram, Y. Cao, K. Ramani, and A. J. Quinn. AdapTutAR: An adaptive tutoring system for machine tasks in augmented reality. In *Proceedings of ACM Conference on Human Factors in Computing Systems*, pp. 1–15, 2021. doi: 10.1145/3411764.3445283

[15] O. Kahler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S. Torr, and D. W. Murray. Very High Frame Rate Volumetric Integration of Depth Images on Mobile Device. *IEEE Transactions on Visualization and Computer Graphics*, 22(11), 2015. doi: 10.1109/TVCG.2015.2459891

[16] S. G. Kaya, B. Zhou, R. R. Arora, N. Zheutlin, G. Vanloo, and E. K. Eyigoz. Dynamic content generation for augmented technical support. In *Proceedings International Symposium on Mixed and Augmented Reality*, pp. 441–446. IEEE, 2021. doi: 10.1109/ISMAR-Adjunct54149.2021.00101

[17] B. Kerbl, D. Kalkofen, M. Steinberger, and D. Schmalstieg. Interactive disassembly planning for complex objects. *Computer Graphics Forum*, 34(2):287–297, may 2015. doi: 10.1111/cgf.12560

[18] A. J. Lambert. Disassembly sequencing: a survey. *International Journal of Production Research*, 41(16):3721–3759, 2003. doi: 10.1080/0020754031000120078

[19] E. Langer, T. Patten, and M. Vincze. Robust and efficient object change detection by combining global semantic information and local geometric verification. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 8453–8460. IEEE, 2020. doi: 10.1109/IROS45743.2020.9341664

[20] S. Makris, G. Pintzos, L. Rentzos, and G. Chryssolouris. Assembly support using AR technology based on automatic sequence generation. *CIRP Annals*, 62(1):9–12, 2013. doi: 10.1016/j.cirp.2013.03.095

[21] A. Miller, B. White, E. Charbonneau, Z. Kanzler, and J. J. LaViola Jr. Interactive 3D model acquisition and tracking of building block structures. *IEEE Transactions on Visualization and Computer Graphics*, 18(4):651–659, 2012. doi: 10.1109/TVCG.2012.48

[22] P. Mohr, S. Mori, T. Langlotz, B. H. Thomas, D. Schmalstieg, and D. Kalkofen. Mixed reality light fields for interactive remote assistance. In *Proceedings of ACM Conference on Human Factors in Computing Systems*, p. 1–12. Association for Computing Machinery, 2020. doi: 10.1145/3313831.3376289

[23] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings International Symposium on Mixed and Augmented Reality*, pp. 127–136, 2011. doi: 10.1109/ISMAR.2011.6092378

[24] E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss. ReFusion: 3D reconstruction in dynamic environments for RGB-D cameras exploiting residuals. *IEEE International Conference on Intelligent Robots and Systems*, pp. 7855–7862, 5 2019. doi: 10.1109/IROS40897.2019.8967590

[25] E. Palazzolo and C. Stachniss. Fast image-based geometric change detection given a 3D model. In *2018 IEEE International Conference on Robotics and Automation*, pp. 6308–6315. IEEE, 2018. doi: 10.1109/ICRA.2018.8461019

[26] J. Park, Q.-Y. Zhou, and V. Koltun. Colored point cloud registration revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 143–152, 2017. doi: 10.1109/ICCV.2017.25

[27] N. Petersen, A. Pagani, and D. Stricker. Real-time modeling and tracking manual workflows from first-person vision. In *Proceedings International Symposium on Mixed and Augmented Reality*, pp. 117–124. IEEE, 2013. doi: 10.1109/ISMAR.2013.6671771

[28] V. A. Prisacariu and I. D. Reid. PWP3D: Real-time segmentation and tracking of 3D objects. *International Journal of Computer Vision*, 98(3):335–354, 2012. doi: 10.1007/s11263-011-0514-3

[29] D. Reiners, D. Stricker, G. Klinker, and S. Müller. Augmented reality for construction tasks: Doorlock assembly. In *Proc. of the International Workshop on AR: Placing artificial objects in real scenes*, pp. 31–46. AK Peters, Ltd., 1999.

[30] M. Rünz and L. Agapito. Co-fusion: Real-time segmentation, tracking and fusion of multiple objects. In *2017 IEEE International Conference on Robotics and Automation*, pp. 4471–4478, 2017. doi: 10.1109/ICRA.2017.7989518

[31] M. Rünz, M. Buffier, and L. Agapito. MaskFusion: Real-time recognition, tracking and reconstruction of multiple moving objects. *Proceedings International Symposium on Mixed and Augmented Reality*, pp. 10–20, 1 2019. doi: 10.1109/ISMAR.2018.00024

[32] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers. StaticFusion: Background reconstruction for dense RGB-D SLAM in dynamic environments. *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3849–3856, 9 2018. doi: 10.1109/ICRA.2018.8460681

[33] A. Taneja, L. Ballan, and M. Pollefeys. City-scale change detection in cadastral 3D models using images. In *Proceedings Conference on Computer Vision and Pattern Recognition*, pp. 113–120, 2013. doi: 10.1109/CVPR.2013.22

[34] A. Tang, C. Owen, F. Biocca, and W. Mou. Comparative effectiveness of augmented reality in object assembly. In *Proceedings of ACM Conference on Human Factors in Computing Systems*, pp. 73–80, 2003. doi: 10.1145/642611.642626

[35] M. Tölgyessy, M. Dekan, L. Chovanec, and P. Hubinskỳ. Evaluation of the Azure Kinect and its comparison to Kinect V1 and Kinect V2. *Sensors*, 21(2):413, 2021. doi: 10.3390/s21020413

[36] A. O. Ulusoy and J. L. Mundy. Image-based 4-d reconstruction using 3-d change detection. In *European Conference on Computer Vision*, pp. 31–45. Springer, 2014. doi: 10.1007/978-3-319-10578-9_3

[37] B. Wang, G. Wang, A. Sharf, Y. Li, F. Zhong, X. Qin, D. CohenOr, and B. Chen. Active assembly guidance with online video parsing. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces*, pp. 459–466. IEEE, 2018. doi: 10.1109/VR.2018.8446602

[38] O. Wasenmüller, M. Meyer, and D. Stricker. Augmented reality 3D discrepancy check in industrial applications. In *Proceedings International Symposium on Mixed and Augmented Reality*, pp. 125–134, 2016. doi: 10.1109/ISMAR.2016.15

[39] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016. doi: 10.1177/0278364916669237

[40] M. Whitlock, G. Fitzmaurice, T. Grossman, and J. Matejka. AuthAR: Concurrent authoring of tutorials for AR assembly guidance. In *Proceedings of Graphics Interface*, p. 431 – 439, 2020. doi: doi.org/10.20380/GI2020.43

[41] Y.-S. Wong, C. Li, M. Nießner, and N. J. Mitra. RigidFusion: RGB-D scene reconstruction with rigidly-moving objects. In *Computer Graphics Forum*, vol. 40, pp. 511–522. Wiley Online Library, 2021. doi: 10.1111/cgf.142651

[42] L.-C. Wu, I.-C. Lin, and M.-H. Tsai. Augmented reality instruction for object assembly based on markerless tracking. In *Proceedings ACM Symposium on Interactive 3D Graphics and Games*, pp. 95–102, 2016. doi: 10.1145/2856400.2856416

[43] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger. Mid-fusion: Octree-based object-level multi-instance dynamic SLAM. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May:5231–5237, 5 2019. doi: 10.1109/ICRA.2019.8794371

[44] M. Yamaguchi, S. Mori, P. Mohr, M. Tatzgern, A. Stanescu, H. Saito, and D. Kalkofen. Video-annotated augmented reality assembly tutorials. In *Proceedings of ACM Symposium on User Interface Software and Technology*, pp. 1010–1022, 2020. doi: 10.1145/3379337.3415819

[45] J. Zauner, M. Haller, A. Brandl, and W. Hartman. Authoring of a mixed reality assembly instructor for hierarchical structures. In *Proceedings International Symposium on Mixed and Augmented Reality*, pp. 237–246, 2003. doi: 10.1109/ISMAR.2003.1240707

[46] B. Zhou and S. Güven. Fine-grained visual recognition in mobile augmented reality for technical support. *IEEE Transactions on Visualization and Computer Graphics*, 26(12):3514–3523, 2020. doi: 10.1109/TVCG.2020.3023635

[47] Q.-Y. Zhou, J. Park, and V. Koltun. Fast global registration. In *European Conference on Computer Vision*, pp. 766–782. Springer, 2016. doi: 10.1007/978-3-319-46475-6_47

[48] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[49] V. Zogopoulos, E. Geurts, D. Gors, and S. Kauffmann. Authoring tool for automatic generation of augmented reality instruction sequence for manual operations. *Procedia CIRP*, 106:84–89, 2022. doi: 10.1016/j.procir.2022.02.159

[50] S. Zollmann, D. Kalkofen, C. Hoppe, S. Kluckner, H. Bischof, and G. Reitmayr. Interactive 4D overview and detail visualization in augmented reality. In *Proceedings International Symposium on Mixed and Augmented Reality*, pp. 167–176. IEEE, 2012. doi: 10.1109/ISMAR.2012.6402554