2016 IEEE International Symposium on Mixed and Augmented Reality

Augmented Reality 3D Discrepancy Check in Industrial Applications

Oliver Wasenmüller* Marcel Meyer† Didier Stricker‡

DFKI - German Research Center for Artificial Intelligence

Abstract

Discrepancy check is a well-known task in industrial Augmented Reality (AR). In this paper we present a new approach consisting of three main contributions: First, we propose a new two-step depth mapping algorithm for RGB-D cameras, which fuses depth images with given camera pose in real-time into a consistent 3D model. In a rigorous evaluation with two public benchmarks we show that our mapping outperforms the state-of-the-art in accuracy. Second, we propose a semi-automatic alignment algorithm, which rapidly aligns a reference model to the reconstruction. Third, we propose an algorithm for 3D discrepancy check based on pre-computed distances. In a systematic evaluation we show the superior performance of our approach compared to state-of-the-art 3D discrepancy checks.

Index Terms: I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Depth cues; I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—3D/stereo scene analysis;

1 Introduction

Real-time 3D discrepancy check is required in many industrial applications such as prototyping, manufacturing and assembly control. With this detection one can verify with the help of a camera whether the 3D geometry of a real object exactly corresponds to a reference 3D model (e.g. CAD model). For example, after a building element or a technical installation was constructed, 3D discrepancy check can be used to check whether the constructed and installed elements comply to their 3D specification [15]. Like many other industrial applications [7] such a detection can be designed as an Augmented Reality (AR) system to increase the usability considerably. The detected discrepancies are directly augmented into the current camera view to enable an interactive system. This leads to the requirements of a precise real-time reconstruction of the scene and a real-time discrepancy check.

Many AR systems use standard 2D color cameras, which is sufficient for augmenting a scene. However, for the analysis of the scene geometry precise depth values are required. Thus, in industry often expensive laser scanners are used. While laser scanners offer a very high accuracy, they are only able to capture depth measurements for a single point or along a single scan line in time. To acquire a complete 3D point cloud time-consuming registration algorithms or mechanically rotating scan heads are required. Both approaches are not suitable for an interactive AR system. In this paper we make use of a RGB-D camera to capture a real object, since it provides dense depth and color images at a high frame rate. However, they suffer from a high level of noise and a limited resolution, leading to a rare distribution in industrial applications. Thus, we propose a new mapping approach to cope with the high sensor noise and achieve high quality reconstruction results. In order to enable the

*e-mail: oliver.wasenmueller@dfki.de
†e-mail: marcel.meyer@dfki.de
‡e-mail: didier.stricker@dfki.de

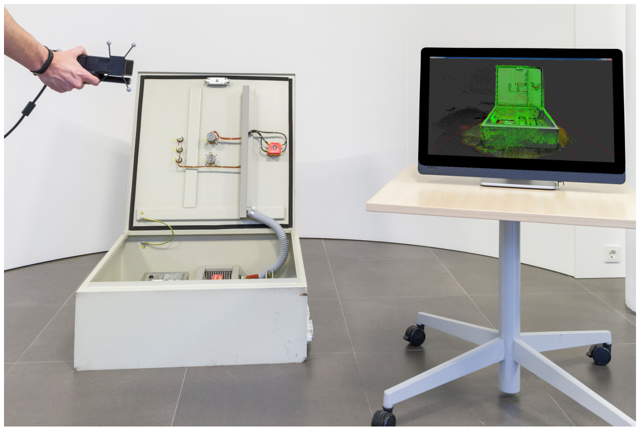


Figure 1: We present our new AR discrepancy check, which is able to capture scene geometry in real-time using an RGB-D camera. Discrepancies are detected while capturing with our new semi-automatic alignment and distance computation.

AR application the reconstruction has to be running online, meaning that continuous input can be processed, and in real-time. Our reconstruction approach can be divided into two main parts: First, we create a precise partial reconstruction from a subset of subsequent depth images, which removes the sensors noise. Second, we fuse all partial reconstructions to a globally consistent and accurate 3D model of the scene. With this approach we can achieve a much higher accuracy compared to the state-of-the-art [24, 29] and are not restricted to a fixed reconstruction volume.

In order to detect discrepancies between the reconstruction and a given reference, these two models must be aligned while capturing. In general, these models can have an arbitrary pose, where no assumptions can be made. In the state-of-the-art the alignment is done by marker detection in the scene [15] or by measuring corresponding points with an external coordinate measuring system [16, 28, 34]. In this paper we present a new semi-automatic approach for reference alignment to increase the usability of our system considerably. The alignment can again be subdivided into two main parts: First, we coarsely align the two models by computing corresponding points with the help of 3D feature descriptors. In a second step, we refine the alignment with variants of ICP. After alignment the discrepancies are detected by measuring the distances of all points in the reconstruction to the reference. In general, this is time-consuming and impedes an interactive discrepancy estimation. Thus, we propose a new approach to detect discrepancies by pre-computing distances in an elaborated way enabling an interactive visualization.

Summarized, our main contributions are:

- A new two-step online mapping algorithm, which is able to accurately reconstruct surfaces with an RGB-D camera like the Microsoft Kinect v2 in real-time and outperforms state-of-the-art methods in the reconstruction accuracy.
- A new semi-automatic approach to rapidly align a reference model with the reconstruction.
- A new and fast approach to detect discrepancies of the whole reconstruction to a given reference.

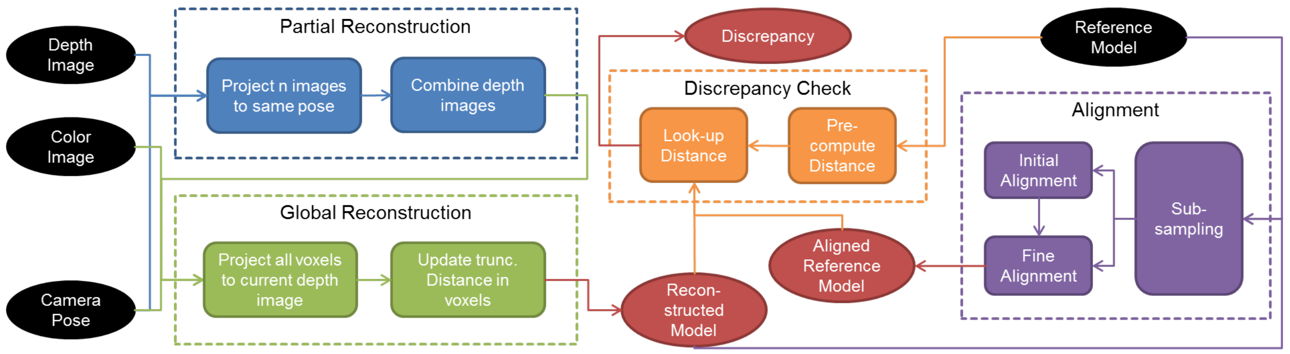


Figure 2: System overview of our new AR discrepancy check. As an input the system uses a reference model as well as depth and color images of the Kinect v2, whereas the camera pose is estimated by an external motion capture system. The current scene is reconstructed with our new online two-step approach consisting of partial and global reconstruction. In parallel the reference model is aligned to the reconstruction with our new semi-automatic approach. Discrepancies are estimated in real-time with our new discrepancy check approach.

- A systematic and rigorous evaluation of all algorithm parts with well-known publicly available benchmarks including real and synthetic data.

The complete pipeline is shown in Figure 2 and can be subdivided into three main parts: In Section 3 our two-step reconstruction approach is presented. Section 4 details our alignment process of reconstruction and reference model, whereas Section 5 describes our detection of discrepancies. All algorithm parts are evaluated in detail in Section 6. See our supplementary video for a short demo of the proposed system.

2 Related Work

The usage of Augmented Reality (AR) to support humans in industrial applications has a long history [7, 32]. Different kinds of discrepancy checks were already investigated in the relevant literature as introduced in this section. Discrepancy check is also sometimes referred to as *disparity estimation* or *difference detection* [15, 35].

The first AR systems for discrepancy check were based on 2D color images. Georgel et al. [8, 9] proposed a system for construction planning, where 3D models are superimposed into 2D images. However, with this augmentation a discrepancy check can only be performed visually and an analysis of the 3D geometry is impossible. Webel et al. [35] presented a system with a stereo camera, where single points in the scene can be marked with a laser pointer, triangulated and compared. Thus, only single points can be compared but no dense 3D model.

More advanced systems are based on 3D information captured by depth cameras [15, 28], camera-projector-systems [37] or laser scanners [11]. Simple approaches are limited to static camera positions [37] or demand manual alignment of single reconstructions [1]. Kahn et al. [15, 16, 17] presented the first system competing with our approach. They reconstruct the scene geometry in 3D in real-time with a Kinect v1 and the KinectFusion algorithm [24] coupled with an external motion capture system for a precise camera odometry. With this approach they capture good-looking dense 3D models in real-time, but suffer from the known drawbacks of KinectFusion like limited accuracy [4] or restriction to a limited reconstruction volume [29]. They align the reference model with the reconstruction with the help of an external coordinate measurement system. Disparities are detected by raycasting both the reconstruction and the reference model from the current camera pose to artificial depth images and comparing their per-pixel depth. This approach runs in real-time on the GPU, but considers only the currently captured pixels and suffers from a viewpoint dependency. The viewpoint dependency was resolved by Stahl [28] by computing closest points on the reference model for each pixel in the artificial depth image with a GPU shader. However, surfaces are not considered and still only points in the current view are compared.

3 Reconstruction

In order to compare a real object with a virtual 3D reference model we have to capture the 3D geometry of the real object. RGB-D cameras capture dense depth images representing the surface geometry, but suffer from a high level of noise and a limited resolution. For our proposed system we use the Microsoft Kinect v2 [21], which has a higher accuracy, precision and resolution than its predecessor [39] or competing devices. However, the quality is still too low, even when using advanced depth filtering technologies. Using depth images directly for discrepancy check leads to imprecise and incomplete results. Thus, we capture the complete real object out of a high number of images by reconstructing it, similar to Kahn et al. [16] and Stahl [28] (cp. Section 2).

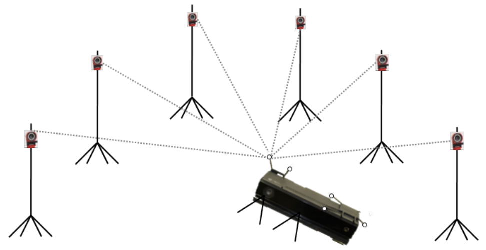
In the literature several algorithms for 3D reconstruction with RGB-D cameras were proposed. The 3D reconstruction problem can be subdivided into two main problems: The camera odometry (estimating the pose of the camera) and the mapping (fusing several depth images to one consistent global model). Some approaches try so solve these problems independently by first computing the camera pose [6, 12, 18, 36] and then fusing the depth images with the given pose [29, 30]. Other approaches, so-called SLAM (Simultaneous Localization And Mapping) algorithms, try to solve both problems simultaneously [24], since these can be treated as a convoluted problem. In this paper we use like Kahn et al. [16] and Stahl [28] an external system for the camera odometry to achieve high precision pose estimates. Thus, our contribution is a new two-step depth mapping, which achieves highly accurate results while keeping the real-time constraint as detailed in Section 3.2.

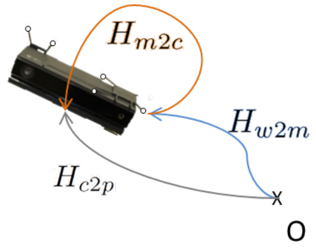
3.1 Camera Odometry

For a precise 3D reconstruction we must accurately compute the pose of the camera. The application of AR discrepancy check requires to retrieve the camera pose in real-time. The literature describes several approaches for real-time camera odometry based on the captured images [6]. However, even sophisticated approaches [12, 18] incorporate errors of multiple centimeters in the pose estimation, which is not feasible for our accuracy requirements.

Hence, we make use of a precise external coordinate measurement system like Kahn et al. [16] and Stahl [28] did. These systems are common in the context of industrial AR [7, 16]. Whereas [16, 28] use an robotic measurement arm, we utilize an active marker tracker system of OptiTrack [25] (see Figure 3b).This system is composed of twelve Flex 13 cameras with a resolution of 1280×1024 at 120 Hz and tracks passive spherical markers, which are rigidly attached to the Kinect as shown in Figure 3a. To detect the passive markers robustly and precisely, the motion capture cameras emit infrared light into a capture volume of up to 10×10

(a) Kinect v2 with Attached Markers


(b) Motion Capture System



(c) Hand-Eye Calibration

Figure 3: In our AR discrepancy check we use a (a) Microsoft Kinect v2 [21], whose pose is estimated with a (b) motion capture system [25]. In order to use this system a (c) hand-eye calibration is necessary.

meter. The system achieves a sub-millimeter and sub-degree precision. We verified that the infrared light of the Kinect and the motion capture system do not interfere. Compared to a robotic measurement arm [16, 28] our system is more flexible and has a much wider operating range. Nevertheless, the operating range is restricted to a (large but finite) capture volume. Although the system price is approx. € 15k, it is still cheaper than a robotic arm (approx. € 100k).

Since all external measurement systems are not able to track the camera centers pose directly, an additional calibration step is necessary. The motion capture system tracks only the pose of the rigid collection of the attached reflective markers. To use the depth images for the reconstruction, the pose of the camera center is required, which is not necessarily the pose of the reflective marker collection. Nonetheless, it exists a rigid transformation between these two poses (see Figure 3c) that can be estimated with a so-called hand eye calibration. Here, we follow the approach of Tsai and Lenz [31] and capture a fixed pattern (in our case a checkerboard) from n perspectives, while recording the depth images $C_{1:n}$ and the detected pose $H_{w2m,1:n}$ of the marker collection in world coordinates. The camera poses $H_{c2p,1:n}$ with respect to the pattern can be easily determined with the PnP algorithm from the images $C_{1:n}$. Since the position H_{w2p} of the pattern is fixed, one can build the equation system

$$H_{c2p,1:n} * H_{m2c} * H_{w2m,1:n} = H_{w2p} \tag{1}$$

for all n perspectives and solve it for the transformation H_{m2c}.

3.2 Depth Mapping

After retrieving the camera pose, the acquired depth images must be mapped into one consistent 3D model. The application of AR discrepancy check requires a high accuracy from the reconstructed model as well as a real-time performance. The challenge is to cope with the high level of noise of the Kinect v2 depth sensor. The random errors in the depth measurement rise up to multiple centimeters as shown in Figure 4. When projecting 120 depth images (this equals 4 seconds of capturing) with precisely estimated pose into world coordinates, a noisy point cloud is generated (see Figure 7a). Thus, sophisticated filter and fusion approaches are required to essentially reduce that noise.

In the corresponding literature several approaches with different runtime performances were proposed. In general, real-time approaches integrate the depth images into a voxel grid with an implicit surface representation. Doing this, they can cope with huge numbers of depth images in terms of runtime and noise. The most common approach is the KinectFusion algorithm [24], where the depth maps are fused into a dense voxel grid on the GPU. This algorithm was also used in the AR discrepancy check of Kahn et al. [16] and Stahl [28]. However, KinectFusion suffers from the restriction to a limited reconstruction volume, due to the limited available memory on the GPU. Whelan et al. [36] overcame this problem by dynamically shifting the dense voxel volume, whereas Steinbrücker et al. [29, 30] used an adaptive sparse voxel volume. Recently, Newcombe et al. [23] extended their algorithm to non-rigid scenes.

Another approach in the literature is to compute an optimal surface between a set of depth images by mathematical optimization. Sometimes this is also referred to as superresolution [5, 20]. In general, these approaches align several subsequent depth images and then compute an optimal surface by energy minimization. They deliver high quality results, but have a high runtime of several seconds. Another approach are joint bilateral filtering algorithms [19, 33], which require an aligned color image to increase the quality of a single depth image in real-time. However, in tests with real data these approaches did not perform better than the superresolution out of multiple depth images.

Our new idea is to split the depth mapping into a sophisticated two-step approach. First, a partial reconstruction from a small subset of the depth images is created like in the superresolution approaches without a voxel grid to achieve the maximum quality. Second, all partial reconstructions are integrated into an adaptive sparse voxel volume in a global reconstruction to cope with the huge number of points.

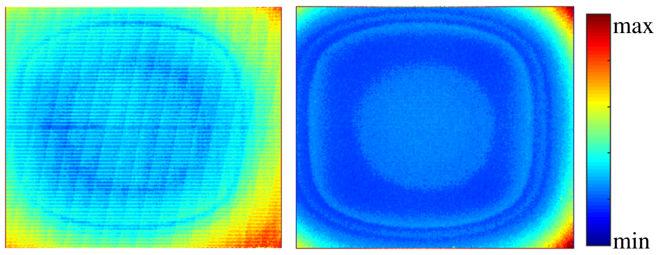
3.2.1 Partial Reconstruction

The input for the partial reconstruction are n ($n = 10$ in all experiments) subsequent depth images of the Kinect v2. Inspired by superresolution algorithms [5, 20], the goal is to fuse them into a high quality partial reconstruction. The main challenges are the runtime, since state-of-the-art methods require seconds, and the high noise level of the Kinect v2.

First, we analyze the noise properties of the Kinect v2 as visible in Figure 4. The outer pixels of the depth images have a high standard deviation as well as a high absolute deviation. Thus, they can not be treated as reliable. For our application of AR discrepancy check we require high quality results, where it is better to capture no geometry instead of wrong geometry. Consequently, we simply remove the outer pixels with a circular filter that removes all pixels more than 90% of the image height away from the image center. The lost image content comprehends no problem, since the camera center can be pointed to any interesting region by the user in our interactive AR application.

The next step is to reduce the random noise in the images. This incorporates high frequent noise like flying pixels and midrange noise on surfaces. Flying pixels are erroneous depth estimates which occur close to depth discontinuities. All time-of-flight cameras suffer from this problem. To remove these pixels, we check the one-ring neighborhood for similar values. More formally, a point B in the point cloud P is removed if

$$\begin{gathered} P \in \mathbb{R}^3 \quad B \in P \nexists A \in P \text{ with} \\ \|A - B\| < \varepsilon \wedge \|A_x - B_x\| < 1 \wedge \|A_y - B_y\| < 1, \end{gathered} \tag{2}$$



(a) offset (b) stdev

Figure 4: Evaluation of the per-pixel error in depth images of Kinect v2 [21]. Outer pixels have a much lower accuracy and precision.

where ε ($\varepsilon = 0.01m$ in the experiments) is the threshold defining the minimal distance of neighboring points. In order to remove the remaining noise the concepts of the superresolution algorithms [5, 20] are very well suited. However, we must considerably increase the runtime. First, subsequent depth images must be aligned. State-of-the-art methods minimize therefore the distance of two point clouds, e.g. with ICP [26]. This is runtime intensive and is difficult to be done in real-time. Indeed, alignment of depth images can be done in our system using camera pose data from the external motion capture system. As soon as the transformations between the depth images are known, they can be transformed within a millisecond in order to be congruent. After alignment the optimal surface must be found. State-of-the-art methods try to find a surface that minimizes the errors to all depth values. This is again computationally intense and difficult regarding real-time performance. Our idea is to compute a stack of n depth values for each pixel retrieved from the n aligned depth images. Then, the median value of the stack is used as the new depth pixel. The median of a stack can be computed very efficiently. Furthermore, the median is compared to mean more robust against outliers. Although this is a simple algorithm, it achieves comparable results to complex superresolution algorithms [5, 20]. But, the runtime is essentially lower and enables real-time performance as depicted in the evaluation in Section 6.1. Exemplary results of our new partial reconstruction are shown in Figure 6.

3.2.2 Global Reconstruction

The result of the partial reconstructions can be assumed to be almost free of high-frequent noise as visible in Figure 6. However, when combining all partial reconstructions, the number of points is too high to be treated in real-time. In addition, small systematic low-frequent errors like distortions remain in the partial reconstruction just as well as for Cui et al. [5]. Thus, we make use of a voxel grid to cope with the number of points and with small errors.

Following the works of [24, 29], the surface is stored implicitly as a truncated signed distance function in a 3D volume approximated by a finite number of voxels. At every voxel in the volume the function indicates how far away the closest surface is. Voxels in front of an object have a negative sign and voxels inside a positive one, whereas the zero crossing indicates the location of the surface. The signed distance representation has the benefit of being able to handle an arbitrary surface topology, in contrast to e.g. a surface mesh. As most space in the 3D voxel volume is far away from a surface, we make use of the approach of Steinbrücker et al. [29, 30]. The signed distance function is represented in an octree, where only cells close to the surface are allocated. As different parts of the scene will be reconstructed at different distances, we save geometry information at different resolutions of the voxel volume.

The signed distance function is incrementally updated with each new input partial reconstruction PR_t and its corresponding camera pose H_t. For each voxel v in the 3D volume the center point is transformed at time t to the current camera pose H_t by

$$v_t = H_t v. \tag{3}$$

The pixel location x of each voxel center v_t in the camera view at time t can be determined with the pinhole camera projection function π by

$$x_t = \pi(v_t). \tag{4}$$

Thus, for every voxel centers v_t the reconstructed point of the partial reconstruction PR_t can be determined using the inverse pinhole projection function $\pi^{-1}(x_t, PR_t(x_t))$ by

$$p_t = \pi^{-1}(\pi(v_t), PR_t(\pi(v_t))). \tag{5}$$

The signed distance λ at the voxel center v_t can be determined with a truncation threshold μ by

$$\lambda = \max\{\min\{\mu, |v_t - p_t|\}, -\mu\}. \tag{6}$$

The truncation threshold μ must be adjusted according to the expected noise in the reconstruction. Like [29] we set μ to the doubled voxel size.

In addition, we use a weighting function $w(\lambda)$ to model the reliability of the truncated distance. The function assigns a weight of one to all pixels in front of the captured surface and a linearly decreasing weight behind the surface [29]. The distance $D(v,t)$ and the weight $D(v,t)$ at time t are computed by

$$W(v,t) = w(\lambda) + W(v,t-1), \tag{7}$$

$$D(v,t) = \frac{W(v,t-1)D(v,t-1) + w(\lambda)\lambda}{W(v,t)}. \tag{8}$$

As a result we achieve high-quality reconstruction results in real-time, which outperform state-of-the-art algorithms as detailed in the evaluation in Section 6.1.

4 Alignment

In order to enable a discrepancy check we must align the reconstruction of the scene with a reference 3D model. In our scenario this includes several challenges: Since we establish an online discrepancy check, we must be able to align the two models as early as possible after starting capturing. Our system should be interactive, which requires short computation time for the alignment. In addition, our system must be robust and accurate, since detected discrepancies should not be originated by imprecise alignment. While aligning we must treat different kinds of noise, only partially overlapping scenes and - inherited from the application - discrepancies between the models. The state-of-the-art systems of Kahn et al. [16] and Stahl [28] require a manual alignment with an external coordinate measurement tool. Therefore, they measure predefined points on the real object and compute a transformation.

Our semi-automatic alignment can be divided into two parts: First, initial alignment that does not make any assumptions on the relative poses of the reconstruction and the reference model, but typically lacks the accuracy required for a precise discrepancy check. The second step compensates for that by doing a fine alignment with an iterative approach. Since we design an interactive AR system, we give the user the possibility to trigger both parts of the alignment independently. With this we can improve the alignment in case more points were reconstructed or compensate misalignments.

4.1 Initial Alignment

The result of the initial alignment is a transformation, which coarsely aligns the reconstruction and the reference model. Common approaches try to find features in both models, match them and estimate a rigid transformation out of the matches [14]. We are limited to geometric features, since our system should work with every kind of reference model, e.g. CAD models, laser scans, etc. Thus, we need features, that do not make use of any color or texture information. The feature extraction, description and matching works as follows: First, the feature extraction selects a subset of points from each point cloud, which are designated as features. These points are then described by an as unique as possible descriptor. For finding corresponding features, the descriptors of two clouds are compared. The technique is detailed in the following.

For feature extraction we use a simple downsampling with a voxel grid. Because model and scene are provided by different data sources downsampling is an instrument to make them more similar. This way of downsampling yields uniformly distributed data and provides a priori knowledge of the point cloud density without computing it via average nearest neighbor search. Thus, controlling the parameters for the feature descriptor is easier and can be set as linear dependence of the voxel grid resolution. Additionally, this is a way to get reasonable results despite the remaining noise in the reconstructed model (see Figure 7). Furthermore, downsampling is used as a method to reduce computational cost. In the evaluation in Section 6.2 we investigate the influence of the voxel grid resolution.

We ran several experiments with different kind of features, but at the end we make use of FPFH features [27] since they showed the best performance in our scenario. FPFH features goal is to generalize both the surface normals and the curvature estimates. This works as follows: The first step is to compute k-nearest neighbors q_i of each feature point p. A point pair (p, q_i) defines a reference frame consisting of the three unit vectors (u_i, v_i, w_i) centered on p. The unit vectors (u_i, v_i, w_i) are defined by

$$u_i = n_p, \qquad v_i = (p - q_i) \times u_i, \qquad w_i = u_i \times v_i. \tag{9}$$

Using this reference frame (u_i, v_i, w_i), the difference between the normals at n_p and n_q can be represented by

$$\alpha_i = v_i \cdot n_p \tag{10}$$

$$\phi_i = \frac{u_i \times (p - q_i)}{\|p - q_i\|} \tag{11}$$

$$\theta_i = \arctan\left(w_i \cdot n_p, u_i \cdot n_p\right) \tag{12}$$

The angles $\alpha_i, \phi_i, \theta_i$ are computed for all pairs (p, q_i) in the k-neighborhood of point p. These angles are binned into a so-called Simplified Point Feature Histogram (SPFH) by considering that each of them can fall into 5 distinct bins, and the final histogram encodes in each bin a unique combination of the distinct values for each of the angles. Then, for each point p the values of the SPFH of its k-neighbors q_i are weighted by their distance $w_i = p - q_i$ to produce the FPFH by

$$FPFH(p) = SPFH(p) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{w_i} \cdot SPFH(q_i). \tag{13}$$

For more details on FPFH we refer to [27].

As a last step, corresponding features in the two point clouds need to be detected. For this we follow Buch et al. [3] with their prerejective RANSAC. We choose a random polygon from the reconstruction point cloud and extract the k most similar points with respect to features similarity for each polygon vertex. For this matching step a cache is used to store correspondences, so matching is only done when a reconstruction point is chosen for the first time. In the next step a random correspondence for each vertex is chosen, such that a polygon consisting of reference points is implied. A transformation is only estimated if the reconstruction and reference polygons are congruent (i.e. if they can be brought to overlap up to a certain degree). In case of a discrepancy in the scene, which was caused by a translation and/or translation, simple feature matching would return matches where no transformation exists. However, checking for congruent polygons detects transformations of single points and discards them. The final transformation is only accepted as a pose estimation if afterwards enough reconstruction points are within a distance threshold to its euclidean nearest neighbors from the reference cloud model. The pose estimation can be used as a coarse alignment, but still requires improvement in a fine alignment (cp. Section 4.2) to be applicable for a discrepancy check. The quality of this coarse pose estimation is evaluated in Section 6.2.

4.2 Fine Alignment

The goal of the fine alignment is to determine a precise alignment between reconstruction and reference model given an initial coarse alignment (cp. Section 4.1). Therefore, we utilize variants of the Iterative Closest Point (ICP) algorithm [26] for this task. In its general formulation the algorithm proceeds as follows. For the reconstruction point cloud S and reference point cloud T, a function Φ representing the quality of alignment is to be minimized.

$$\min_{C \subset S \times T, t \in \mathbb{R}^3, R \in SO(3)} \Phi(C, R, t)$$
$$\text{with} \quad \Phi(C, R, t) = \sum_{(x,y) \in C} \mu(Rx + t, y) \tag{14}$$

This is a hard optimization problem, considering that the independent choice of C or t and R influence the optimal choice of the other under the function Φ. Therefore, the simplification taken by ICP is to alternate between optimizing the transformation, given by t and R, and the choice of corresponding points C in reconstruction and reference cloud. In a most simple form, often referred to as the ICP algorithm, μ is chosen as the squared euclidean distance of two points:

$$\mu(x, y) = \|x - y\|_2^2 \tag{15}$$

The optimization of the transformation is a least-squares solution of an overdetermined equation system. However, least squares minimization is very sensitive to far distant points. In our application the reconstruction is affected by small outliers due to remaining imprecisions in the reconstruction, as well as clutter caused by discrepancies between reconstruction and reference. The challenge is to cope with these distant points.

One way is to use iterative reweighting strategies, which compute a robust weight $w : \mathbb{R}_0^+ \mapsto [0, 1]$ on the distance of each point pair after matching. The resulting function μ for minimization is

$$\mu(x, y) = w(\|x - y\|) \cdot \|x - y\|^2, \tag{16}$$

where $w(x)$ is a weighting function. This function should neglect the influence of high distant points, whereas short distant points should have a huge impact. In our system we use a Tukey weight function

$$w_q(x) = \begin{cases} 0, & \text{if } x <= q \\ (1 - \frac{x}{q}^2)^2, & \text{otherwise} \end{cases}, \tag{17}$$

where q ($q = 0.1m$ in the experiments) is a threshold defining from which distance on points are neglected.

Another way to cope with distant points is Sparse Iterative Closest Point (SICP) proposed by Bouaziz et al. [2] using

$$\mu(x, y) = \|x - y\|_2^p, \text{ with } p \in [0, 1) \tag{18}$$

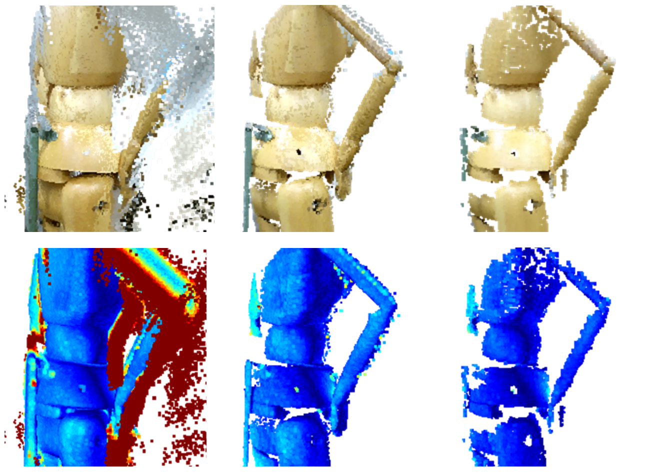

(a) Input Images (b) Partial Recon. (c) Global Recon.

Figure 5: Evaluation of our new two-step reconstruction approach using the Human (H2) scene of CoRBS [34]. (a) 100 input images are combined to (b) 10 partial reconstructions and afterwards to (c) one global reconstruction. The quality increases in each step.

for minimization. The term can be explained by looking at the convergence behavior of the p-th root with respect to p.

$$\lim_{p\to\infty} x^{1/p} = 1, x \in \mathbb{R}^+ \text{ and } \lim_{p\to\infty} 0^{1/p} = 0 \quad (19)$$

Consequently, for $p = 0$ the function Φ from Equation 14 simply returns the number of elements from C with vanishing distance. Thus, minimizing the function returns transformations such that a good match is found, however on a small subset. We evaluated the effect of the two approaches in Section 6.2.

The distance functions in Equation 15, 16 and 18 use a point-to-point distance. However, a point-to-surface distances lead to faster and more precise results [26], because of their higher precision in the near-field. The point-to-point distances between two points x and y are computed based on the connection vector $\psi(x,y) = x - y$. In order to consider point-to-surface distances the connection vector can be projected to normals by

$$\psi(x,y) = n^T \cdot (x - y). \quad (20)$$

For the AR application as well as the evaluation we use only point-to-surface distances. Using this distance measurement enables us also to downsample the point clouds with a voxel grid before fine alignment, since missing points are compensated by the projection of Equation 20. This speeds up the fine alignment, while the alignment results enhance. An evaluation on the voxel size is given in Section 6.2.

5 Discrepancy Check

After reconstructing the scene (Section 3) and aligning the reference model (Section 4), the last step in our application is to determine the local discrepancies between the reconstruction and the reference model. This task can be reduced to finding corresponding points in the two models and analyzing their distance. The challenge is again the real-time requirement of an AR application. A straightforward approach would be to compute for each point in the reconstruction the closest point in the reference model. This can be done relatively fast with kd-trees, but does not reach sufficient performance as shown in Section 6.3.

Golparvar-Fard et al. [10] store occupancies in a voxel grid enabling them to detect coarse differences, but a quantification of the discrepancies is not possible. Kahn et al. [15, 16] compare reconstruction and reference by raycasting them with the identical camera pose to an artificial depth image. Discrepancies are estimated by the per-pixel difference of the two depth images. This approach runs in real-time on the GPU, but contains major drawbacks. First of all, only the pixels visible in the current view are compared, leading to a partial comparison. This makes it hard to conclude statements about the complete real object. Furthermore, the approach is view-point dependent, since the per-pixel computation measures along the view direction and not along shortest distances. Stahl [28] resolved the viewpoint dependency with the help of a GPU shader. For every visible point the closest point in the reference model is computed. However, this incorporates only point-to-point distances, which are - especially for small differences - not precise. In addition, the limitation of partial comparison remains.

In this paper we propose a new approach for AR discrepancy check by using pre-computed distances. Since the reference model stays unchanged during runtime, the distance of a given point to the reference is fixed. Thus, we pre-compute a voxel volume, which stores in each voxel the distance of the voxel center to the reference surface. This has the advantage of precise distance measurements (compared to pure occupancies or point-to-point distances), is viewpoint independent and compares the complete reconstruction against the reference. On the other hand, the voxel volume must be created in advance and the access to the values must be fast. Therefore, we make use of a VDB voxel structure [22]. Basically, this is a data structure that combines concepts of B+ trees and space partitioning structures like voxels and is designed for storage of sparse volumetric data. To this end it divides space into a voxel grid and builds a tree on top. The voxel grid is spare, meaning that only voxels within a given band around the surface are allocated. The huge advantage of such a voxel grid is the fast and constant query time, which is independent of the number of voxels. A critical point is the voxel size, since for smaller voxels the memory consumption is higher, whereas for larger voxels the accuracy decreases. In Section 6.3 we analyze the influence of the voxel size. The voxel grid must be constructed only once and can be stored like a geometry file. Thus, for our AR discrepancy check the grid is loaded in the beginning and during reconstruction (Section 3) the distances of all

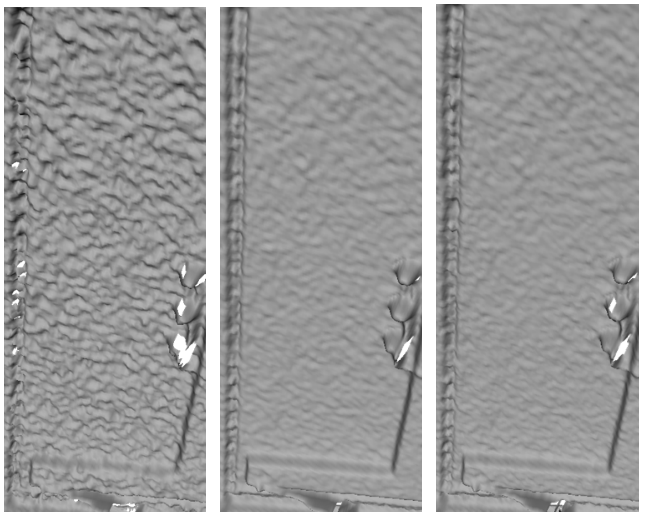

(a) Raw Input (b) Cui: ∼4500ms (c) Ours: ∼20ms

Figure 6: Partial Reconstruction: Close-up view of the Electrical Cabinet of the CoRBS dataset [34]. The raw input contains heavy noise introduced by the Kinect v2. The approaches of Cui et al. [5] and ours remove the noise considerably and archive comparable results, whereas ours is real-time capable.

Algorithm Part	Runtime
partial reconstruction *(combination of 10 images)*	0.02s
global reconstruction *(fusion of one partial reconstruction)*	0.01s
complete reconstruction of 10 images *(capture and partial + global recon.)*	0.35s
initial alignment *(depending on reconstruction size)*	0.1s - 1.5s
fine alignment *(depending on reconstruction size)*	0.002s - 0.01s
discrepancy query *(for one point)*	7.8e^{-8}s
discrepancy pre-computation *(for one point)*	10s

Table 1: Summary of the runtimes of each single algorithm part. We spent much effort on achieving fast runtimes (without loosing quality) in order to enable an interactive AR system.

points to the reference are queried from the grid at frame rate.

The discrepancies are visualized based on the queried distances using a color mapping (red-yellow-green). Since we can reconstruct scenes with an accuracy of around 0.01m (see Section 6.1), we colorize discrepancies up to that distance in green. Distances greater than 0.03m are treated as large discrepancy and thus colored in red, while all distances in between are visualized in yellow. Figure 9 shows an exemplary discrepancy visualization.

6 Evaluation

In this section we evaluate all parts of our new approach for AR discrepancy check. In order to obtain meaningful results we make use of two publicly available benchmarks, namely CoRBS [34] and ICL-NUIM [4, 13]. The CoRBS benchmark [34] was recently published and is the only dataset providing real image data together with ground truth trajectories for the camera and ground truth reconstructions of the scene. This is an ideal basis for the evaluation of our new system. The ground truth trajectory of CoRBS was acquired with an external motion capture system as we do it in our online AR application. The ground truth reconstructions of CoRBS can be used as a reference model, since they have high quality (similar to CAD) and are already aligned to the depth images. Thus, also the evaluation on the alignment is possible with this dataset. In addition, we recorded a new dataset of a modified Electrical Cabinet by adding four components. The data will also be published on the CoRBS website [34]. Since state-of-the-art reconstruction algorithms were mainly benchmarked with synthetic data in the past, we also use the synthetic ICL-NUIM benchmark [4, 13]. It provides two large-scale scenarios each with two trajectories.

All experiments in this paper were performed on a 64 bit Windows computer with a quad core CPU with $3.5GHz$ and $16GB$ main memory. In the following we discuss the qualitative results as well as the runtimes (see also Table 1) of each single algorithm part.

6.1 Reconstruction

First, we evaluate the reconstruction of objects with our system. As detailed in Section 3.2 we proposed a new online two-step approach consisting of a partial and a global reconstruction. Results of the partial reconstruction can be seen in Figure 6. The raw input image on the left contains heavy noise, whereas the algorithm of Cui et al. [5] and ours remove the noise with comparable results. For both algorithms we used the alignment of the external motion capture system to make the results comparable. Whereas the results are similar, the runtime differs essentially. Our new algorithms needs only around $20ms$ for the combination of $n = 10$ depth images, but

	KinectFusion		Steinbrücker		Ours	
	mean	RMSE	mean	RMSE	mean	RMSE
E5	0.017	0.026	0.020	0.029	0.012	0.019
D2	0.018	0.027	0.032	0.042	0.011	0.019
H2	0.015	0.025	0.019	0.031	0.010	0.021

Table 2: Mean distance and RMSE (in meters) of each reconstructed model to the ground truth surface using the CoRBS benchmark [34]. Identical image and pose data was used for all algorithms. Our two-step mapping outperforms state-of-the-art algorithms like KinectFusion [24] or Steinbrücker [30]. A visualization is given in Figure 7.

	Kintinuous*	DVO* SLAM	SUN3D SfM*	Choi*	Ours
LivingRoom1	0.22	0.21	0.09	0.04	0.008
LivingRoom2	0.14	0.06	0.07	0.07	0.008
Office1	0.13	0.11	0.13	0.03	0.008
Office2	0.13	0.10	0.09	0.04	0.007

* This algorithm uses an own pose estimation.

Table 3: Mean distance (in meters) of each reconstructed model to the ground truth surface using the ICL-NUIM benchmark [4, 13]. Note: Kintinuous [36], DVO SLAM [18], SUN3D SfM [38] and the approach of Choi [4] estimate also the camera trajectory, whereas ours uses the trajectory of an external motion capture system for evaluation as well as for our AR application.

	Steinbrücker		Ours	
	mean	RMSE	mean	RMSE
LivingRoom1	0.009	0.012	0.008	0.008
LivingRoom2	0.010	0.015	0.008	0.009
Office1	0.008	0.011	0.008	0.008
Office2	0.008	0.011	0.007	0.008

Table 4: Mean distance and RMSE (in meters) of each reconstructed model to the ground truth surface using the ICL-NUIM benchmark [4, 13]. Our new online two-step algorithm outperforms the state-of-the-art algorithm of Steinbrücker et al. [30]. Identical image and pose data was used for both algorithms.

state-of-the-art methods [5, 20] need seconds of runtime. Consequently, our partial reconstruction is real-time capable.

Next, we evaluate the reconstruction quality of our whole system. We compare our results on the ICL-NUIM benchmark [4, 13] against state-of-the-art algorithms [4, 18, 36, 38], which also estimate the camera pose. These algorithms can clearly be outperformed as visible in Table 3. This is not only an achievement of our new mapping but rather of using a high precision pose estimation like the external motion capture system. In the following experiments we always use identical camera poses and image input in order to investigate only the influence of the new depth mapping.

In Table 2 we compare our two-step mapping with real data on the CoRBS benchmark against KinectFusion [24] and Steinbrücker et al. [30]. Our algorithm has an essentially reduced mean and root mean squared error (RMSE) compared to the state-of-the-art. Especially the RMSE indicates that points with huge distance are reconstructed more accurately, which is of particular importance for our application. Figure 7 visualizes the accuracy improvements. KinectFusion is not able to reconstruct several parts of the scene, whereas Steinbrücker et al. contains a high noise level, since outliers are poorly treated. In contrast, the new two-step mapping is able to reconstruct almost all parts of the scene and simultaneously removing the heavy noise. Incorrect reconstructions are only gained if the sensor captures wrong depth measurements as visible on the black screen in the Desk scene. We also compared our new reconstruction approach on the synthetic ICL-NUIM benchmark. Table

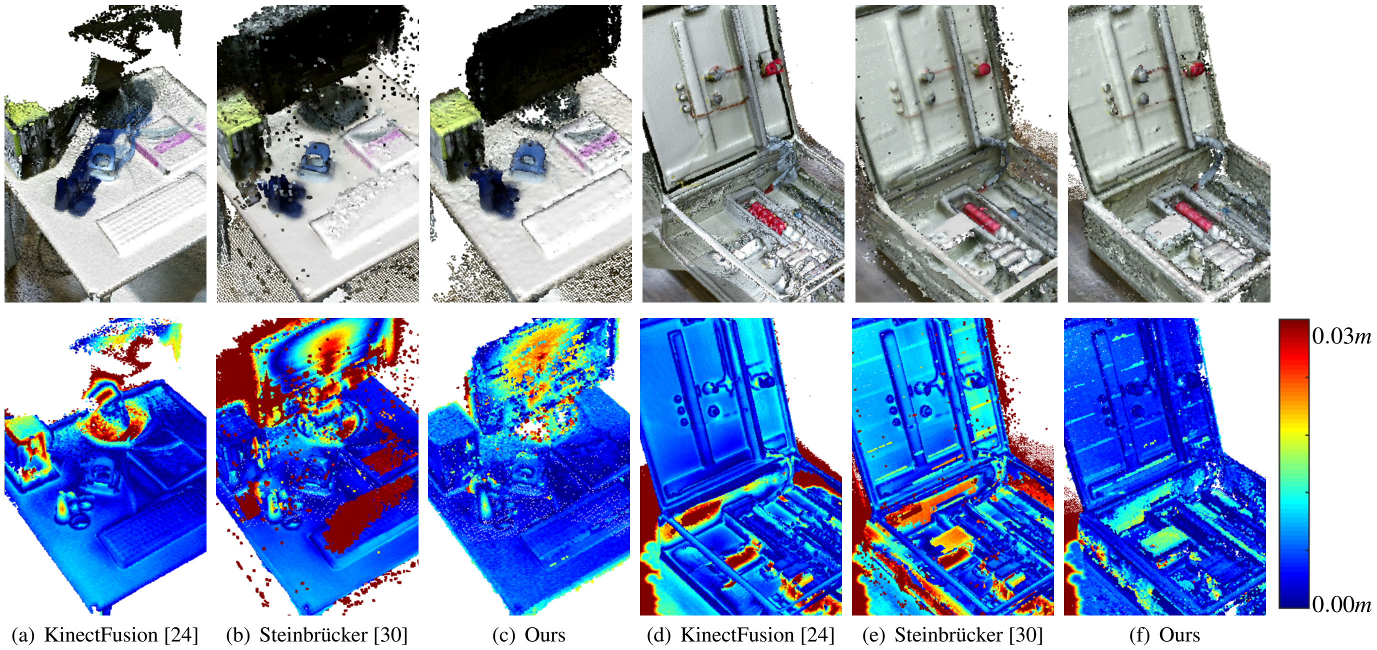


Figure 7: Evaluation of the mapping accuracy on the D2 (2380 frames) and E5 (2209 frames) sequences of the CoRBS benchmark [34]. Identical input images and camera poses are used for this evaluation. Our new two-step depth mapping is more accurate than the state-of-the-art algorithms KinectFusion [24] and Steinbrücker [30]. A quantitative evaluation is given in Table 2.

4 shows comparable results as for the real-world data. In addition, these reconstructions verify the large-scale capability of our system. KinectFusion [24] was not able to reconstruct that large scenes. For the accuracy analysis the ground truth alignment is used for reproducibility. Own alignments would obtain slightly better results, but the quintessence would not change. Our reconstruction (partial and global together) need around 350ms to process 10 images on our computer. This confirms the real-time performance. Summarized, we showed that our algorithm considerably pushes forward the state-of-the-art for large-scale online reconstruction in real-time. Thus, our system is also more precise than Kahn et al. [16] and Stahl [28], since they rely on KinectFusion.

6.2 Alignment

In this section we evaluate the accuracy of our new semi-automatic alignment of a reference model. As detailed in Section 4 our approach is composed into an initial and a fine alignment. We evaluated our algorithm on several scenes and image sequences. In this paper we show exemplified evaluation results using the Electrical Cabinet (E5) and Desk (D2) scene of the CoRBS benchmark.

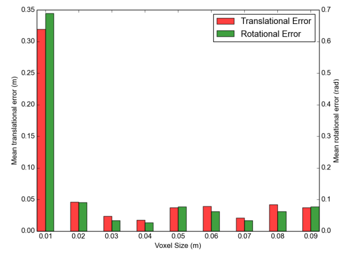
For the initial alignment a downsampling with a voxel grid is used as a feature extractor. The size of the grid influences the result as shown in Figure 8a,e. Whereas 0.01m voxel size leads to a misalignment, all other scales succeed. The lowest errors were achieved for 0.03m voxel size. Using this voxel size Figure 8b,f shows the translational and rotational error of the initial alignment at given time stamps. In the first 4s the alignment fails, since too few points are available. Afterwards the Electrical Cabinet can be aligned with on average 0.02m and the Desk with 0.04m translational error. This is sufficiently precise for the following fine alignment. In our experiment the initial alignment took between 0.1s and 1.5s depending on the number of points in the reconstruction. This is maintainable for an interactive AR system.

For the fine alignment again a downsampling per voxel grid is used, which influences the achieved accuracy. In our experiments a voxel size of 0.02m showed best results (cp. Figure 8c,g). Using this voxel size, Figure 8d,h shows the translational and rotational error of the fine alignment at given time stamps. The initial pose for ICP was taken from our initial alignment (cp. Section 4.1). However, for some datasets the Tukey weight showed better results, for some the Sparse ICP. Thus, we give the user the possibility to trigger both algorithms. At the end we are able to align the reference with less than 0.01m translational error. Please note, this error incorporates also the reconstruction error, since the reference is aligned against this cloud. The reconstruction error is also in the range of 0.01m following that the alignment is very precise. A visual check confirms this. In our experiment the fine alignment took between 0.002s and 0.01s depending on the initial alignment and on the number of points in the reconstruction. This is almost real-time and thus well suited for our interactive AR system.

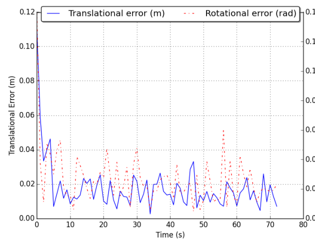
6.3 Discrepancy Check

In this section we evaluate our new discrepancy check using precomputed distances. As described in Section 5 we use a VDB voxel grid [22] storing in each voxel the distance of its center to the surface. Depending on the voxel size also the accuracy changes. We compare our VDB distance computation against Approximate Nearest Neighbors (ANN), which is also known to be fast. In order to determine the accuracy of the distance measurement, we select random points and compute the actual distance to the surface as well as make VDB and ANN queries. The query time of our approach is substantially faster for all accuracies as shown in Figure 10a. In our application an accuracy of around 0.001m is sufficient. In this range our VDB queries are approximately 15-times faster than ANN. With this we are able to determine the distances for all points in a complete reconstruction of an object at frame-rate. For example, the final reconstruction of the D2 sequence in CoRBS contains 295,000 points, leading to a total query time of around 23ms with our VDB queries (cp. ANN $\approx$ 340ms).

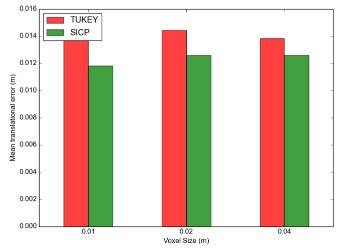
However, the fast query time is accompanied by a longer initial construction time as shown in Figure 10b. While the construction time of the ANN structure is almost constant for different accuracies, the construction time of our VDB structure increases exponentially with higher accuracies. For an accuracy of 0.001m the Desk



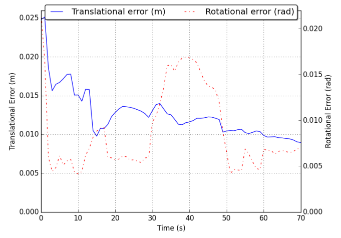
(a) Initial Alignment: Scale Influence



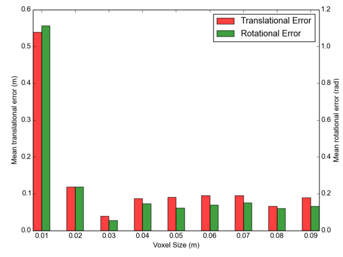
(b) Initial Alignment over Time



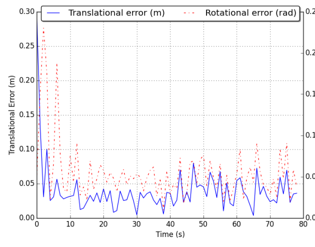
(c) Fine Alignment: Scale Influence



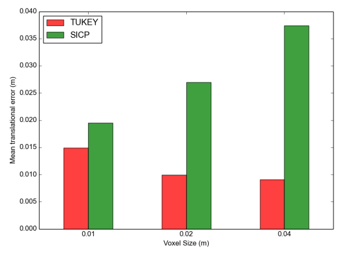
(d) Fine Alignment: Sparse ICP



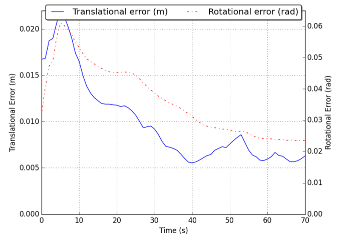
(e) Initial Alignment: Scale Influence



(f) Initial Alignment over Time



(g) Fine Alignment: Scale Influence



(h) Fine Alignment: Tukey Weights

Figure 8: Evaluation of the alignment with the CoRBS benchmark [34] using the Electrical Cabinet (E5) in the first row and the Desk (D2) in the second row. The statistics for other image sequences look similar.

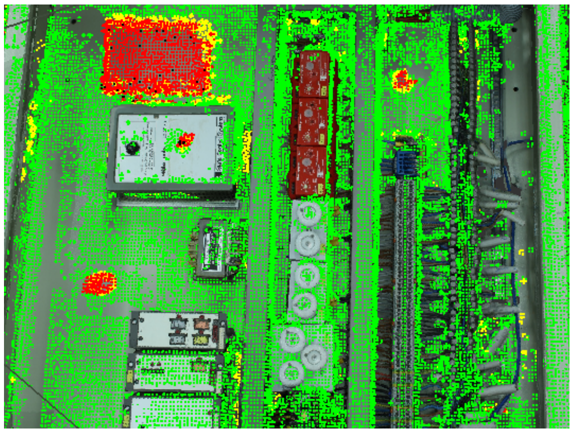
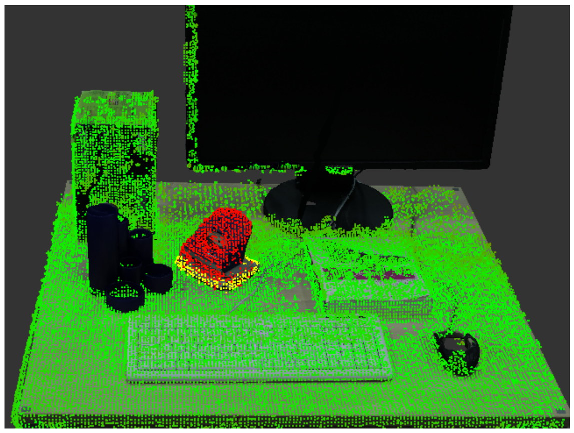
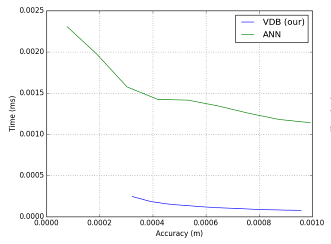
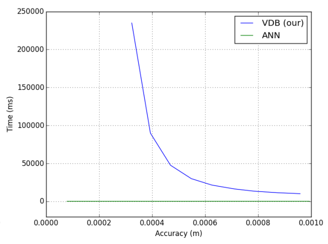



Figure 9: Exemplary visualization of discrepancies detected on a modified Electrical Cabinet (four discrepancies) and Desk (one discrepancy).



(a) Query Time



(b) Construction Time

Figure 10: Comparison of our VDB distance computation against ANN using the Desk scene of CoRBS [34]. The query time of our approach is 15-times faster enabling a real-time performance. However, our VDB grid requires a more time-consuming initial construction.

scene of CoRBS [34] took around $10s$. Since this construction must only be performed once, this can either be done after the program start or the grid can also be stored and loaded. Summarized, we proposed a new method for 3D discrepancy check, which is able to compute all point distances from the reconstruction to the reference cloud in real-time (see Figure 9).

7 Conclusion

In this paper we propose a new Augmented Reality discrepancy check using a RGB-D camera. With a new two-step reconstruction algorithm the system is able to capture the geometry of a given object with high accuracy. The algorithm is not restricted to a given volume and can handle online input images in real-time. We showed on public benchmarks that our two-step approach clearly outperforms state-of-the-art algorithms in matters of mapping accuracy. Furthermore, we propose a semi-automatic alignment algorithm, which is able to align a reference model fast and accurately. With this no manual alignment is necessary and the discrepancies can be directly detected. In addition, we propose a new approach to compute discrepancies based on pre-computed distances. Herewith, we are able to detect discrepancies at frame-rate for the whole reconstruction. At the end our system is able to detect discrepancies in the range of $\sim 0.01m$. For a short demo of the system, see the supplementary video.

Summarized, compared to existing approaches [16, 28, 35] our discrepancy check has the following advantages while fulfilling the constraints for an interactive AR system. We achieve a more accurate geometry capturing, since the mapping accuracy outperforms the state-of-the-art (see Section 6.1). The capture volume is not restricted by the geometry reconstruction (see Section 3.2) but solely the motion capture system is the limit. Furthermore, the alignment of a reference model is done semi-automatically (see Section 4). Discrepancies are detected for the whole reconstruction, instead of visible (or sparse) points only (see Section 5). In addition, discrepancies are determined based on point-to-surface distances without any view-point dependence (see Section 5).

Acknowledgements

This project was partially funded by the German Federal Ministry for Education and Research (BMBF) in the project ARVIDA (01IM13001J).

References

[1] F. Bosche, C. T. Haas, and B. Akinci. Automated recognition of 3D CAD objects in site laser scans for project 3D status visualization and performance control. *Journal of Computing in Civil Engineering*, 23(6):311–318, 2009.

[2] S. Bouaziz, A. Tagliasacchi, and M. Pauly. Sparse iterative closest point. In *Symposium on Geometry Processing*, 2013.

[3] A. G. Buch, D. Kraft, J.-K. Kamarainen, H. G. Petersen, and N. Kruger. Pose estimation using local structure-specific shape and appearance context. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2080–2087. IEEE, 2013.

[4] S. Choi, Q.-Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5556–5565. IEEE, 2015.

[5] Y. Cui, S. Schuon, S. Thrun, D. Stricker, and C. Theobalt. Algorithms for 3D shape scanning with a depth camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2013.

[6] Z. Fang and Y. Zhang. Experimental evaluation of RGB-D visual odometry methods. *International Journal of Advanced Robotic Systems*, 12:26, 2015.

[7] P. Fite-Georgel. Is there a reality in industrial augmented reality? In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 201–210. IEEE, 2011.

[8] P. Georgel, S. Benhimane, J. Sotke, and N. Navab. Photo-based industrial augmented reality application using a single keyframe registration procedure. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 187–188. IEEE, 2009.

[9] P. Georgel, P. Schroeder, S. Benhimane, S. Hinterstoisser, M. Appel, and N. Navab. An industrial augmented reality solution for discrepancy check. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 1–4. IEEE, 2007.

[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 (ICCV Workshops)*, pages 249–256. IEEE, 2011.

[11] C. Gordon and B. Akinci. Technology and process assessment of using ladar and embedded sensing for construction quality control. In *Construction Research Congress*, pages 5–7, 2005.

[12] D. Gutiérrez-Gómez, W. Mayol-Cuevas, and J. Guerrero. Inverse depth for accurate photometric and geometric error minimisation in RGB-D dense visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 83–89. IEEE, 2015.

[13] A. Handa, T. Whelan, J. McDonald, and A. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[14] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke. Registration with the point cloud library: A modular framework for aligning in 3D. *IEEE Robotics & Automation Magazine*, 22(4):110–124, 2015.

[15] S. Kahn. *Precise Depth Image Based Real-Time 3D Difference Detection*. PhD thesis, Technische Universität Darmstadt, 2014.

[16] S. Kahn, U. Bockholt, A. Kuijper, and D. W. Fellner. Towards precise real-time 3D difference detection for industrial applications. *Computers in Industry*, 64(9):1115–1128, 2013.

[17] S. Kahn, H. Wuest, D. Stricker, and D. W. Fellner. 3D discrepancy check via augmented reality. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 241–242. IEEE, 2010.

[18] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for RGB-D cameras. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3748–3754. IEEE, 2013.

[19] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. *ACM Transactions on Graphics (TOG)*, 2007.

[20] R. Maier, J. Stückler, and D. Cremers. Super-resolution keyframe fusion for 3D modeling with high-quality textures. In *International Conference on 3D Vision (3DV)*, pages 536–544. IEEE, 2015.

[21] Microsoft. Kinect v2. www.microsoft.com/en-us/kinectforwindows/.

[22] K. Museth. VDB: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics (TOG)*, 32(3):27, 2013.

[23] R. A. Newcombe, D. Fox, and S. M. Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[24] 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 *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[25] OptiTrack. Flex 13. www.optitrack.com/products/flex-13/.

[26] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *IEEE International Conference on 3D Digital Imaging and Modeling*, pages 145–152. IEEE, 2001.

[27] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3D registration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3217. IEEE, 2009.

[28] C. Stahl. Distortion-free 3D reconstruction and discrepancy check using an RGBD camera. Master Thesis, University of Kaiserslautern, Germany, 2013.

[29] F. Steinbrucker, C. Kerl, and D. Cremers. Large-scale multi-resolution surface reconstruction from RGB-D sequences. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3264–3271, 2013.

[30] F. Steinbrücker, J. Sturm, and D. Cremers. Volumetric 3D mapping in real-time on a cpu. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2021–2028. IEEE, 2014.

[31] R. Y. Tsai and R. K. Lenz. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation*, 5(3):345–358, 1989.

[32] X. Wang, S. Ong, and A. Nee. A comprehensive survey of augmented reality assembly research. *Advances in Manufacturing*, 2016.

[33] O. Wasenmüller, G. Bleser, and D. Stricker. Combined bilateral filter for enhanced real-time upsampling of depth images. *International Conference on Computer Vision Theory and Applications*, 2015.

[34] O. Wasenmüller, M. Meyer, and D. Stricker. CoRBS: Comprehensive RGB-D benchmark for slam using kinect v2. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016.

[35] S. Webel, M. Becker, D. Stricker, and H. Wuest. Identifying differences between CAD and physical mock-ups using AR. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 1–2. IEEE, 2007.

[36] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, Jul 2012.

[37] F. C. Wu and A. Dellinger. Gpu-based discrepancy check for 3D fabrication. In *International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2014.

[38] J. Xiao, A. Owens, and A. Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1625–1632, 2013.

[39] S. Zennaro, M. Munaro, S. Milani, P. Zanuttigh, A. Bernardi, S. Ghidoni, and E. Menegatti. Performance evaluation of the 1st and 2nd generation kinect for multimedia applications. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2015.