OmniKinect: Real-Time Dense Volumetric Data Acquisition and Applications

Bernhard Kainz, Stefan Hauswiesner, Gerhard Reitmayr, Markus Steinberger,
Raphael Grasset, Lukas Gruber, Eduardo Veas, Denis Kalkofen, Hartmut Seichter,
and Dieter Schmalstieg
Graz University of Technology, Institute for Computer Graphics and Vision
Inffeldgasse 16, A-8010 Graz, Austria
<*author-last-name*>@icg.tugraz.at

ABSTRACT

Real-time three-dimensional acquisition of real-world scenes has many important applications in computer graphics, computer vision and human-computer interaction. Inexpensive depth sensors such as the Microsoft Kinect allow to leverage the development of such applications. However, this technology is still relatively recent, and no detailed studies on its scalability to dense and view-independent acquisition have been reported. This paper addresses the question of what can be done with a larger number of Kinects used simultaneously. We describe an interference-reducing physical setup, a calibration procedure and an extension to the KinectFusion algorithm, which allows to produce high quality volumetric reconstructions from multiple Kinects whilst overcoming systematic errors in the depth measurements. We also report on enhancing image based visual hull rendering by depth measurements, and compare the results to KinectFusion. Our system provides practical insight into achievable spatial and radial range and into bandwidth requirements for depth data acquisition. Finally, we present a number of practical applications of our system.

Categories and Subject Descriptors

C.0 [**Computer Systems Organization**]: System architectures; D.4.7 [**Organization and Design**]: Interactive systems; C.4 [**Performance of Systems**]: Performance attributes

Keywords

depth sensors; 4D reconstruction; Microsoft Kinect

1. INTRODUCTION

The Microsoft Kinect has profoundly changed the possibilities of sensing for games or Virtual Reality applications. Previously, inexpensive and thus scalable sensing technology was primarily based on affordable digital video cameras, making computationally expensive image processing necessary. With a Kinect device, direct depth sensing and video capture immediately deliver rich information on the scene structure without intensive processing at a competitive price. Not surprisingly, many researchers have taken advantage of this opportunity, and we see a proliferation of research projects that rely on this technology.

A single Kinect can deliver enough information to let a user control a character in a video game with body movements or resolve real time occlusion of video-see through Augmented Reality (AR) applications. However, the scope of potential applications is ultimately limited by the view-dependent nature of the sensor and the fact that it samples surfaces at a finite resolution. Scalability by using data from multiple Kinects is therefore a prime research topic for improving the quality of acquisition and reconstruction with such a device.

For applications that demand fully view-independent and dense surface reconstruction of dynamic scenes in real time, multiple Kinects can be clustered and their output combined. This approach is still inexpensive, at least for professional applications. However, building such a setup is not necessarily trivial, as a number of conceptual and technical challenges in the system design must be overcome. In this paper, we report on the construction and evaluation of OmniKinect, a system using multiple Kinects for real-time dense volumetric acquisition. We describe the challenges in making such a system work in practice. In particular, we describe the efforts taken in making our hardware setup flexible and scalable. We also report on an improvement technique to allow real-time fusion of sensor readings from multiple Kinects, and on an image-based rendering approach enhanced by depth information. Both algorithms are versatile tools for the processing of combined RGB-D data. Finally, we illustrate our work with several application examples, which previously would have been very hard or impossible to achieve.

2. RELATED WORK

Multi-view stereo systems: Before affordable depth sensors became available, systems with multiple video cameras were used to reconstruct and render dynamic 3D scenes. Early approaches extended the concept of feature-based stereo matching to multiple cameras [5]. GPU-based implementations work by, for example, using plane sweep algorithms [22] to achieve interactive frame rates.

In many applications, only a defined foreground object is of interest. If the foreground object or the background can

VRST'12, December 10–12, 2012, Toronto, Ontario, Canada.

be segmented in the multiple camera images, shape-from-silhouette algorithms can be used to reconstruct a coarse hull of the object fast [2]. GPU-based implementations can speed up the process considerably [10]. For applications that intend to render the reconstructed object without an explicit 3D model, image-based visual hull (IBVH) rendering [13] circumvents the explicit reconstruction of a 3D mesh or voxel grid by directly computing a depth map of the scene object from the desired viewpoint. A similar approach is used for 3D TV based on camera arrays [14].

Single and multi-depth sensor systems: The capabilities of the Microsoft Kinect have been already explored for a variety of applications. For example, a single Kinect already enables high-quality real-time human pose estimation [18]. Kinect-based body scanning [19] also enables virtual try-on applications at low costs. Newcombe et al. have shown with their work on KinectFusion [15] that dense volumetric reconstructions can be created in real time. Because the Kinect is so inexpensive, combining multiple devices has also been investigated for different research projects. For example, Wilson *et.al.* [21] and Berger *et.al.* [1] use up to four depth sensors to monitor a room. Both ensure that the reconstruction light patterns do not overlap, to avoid interferences of the structures light patterns emitted by multiple sensors. Maimone and Fuchs [11] propose advanced hole filling and meshing techniques to use a multi-Kinect setup for telepresence applications. Other approaches use different modulation frequencies per camera [8, 6], which is not possible when using Kinect depth sensors.

The problem of overlapping reconstruction light pattern has been solved by Maimone *et.al.* [12] and Butler *et.al.* [3] with a similar approach. Letting the whole RGB-D camera vibrate at a relatively high frequency blurs the light pattern for other, concurrently capturing sensors. The rigid connection of the vibrating sensor and the light pattern supports a clear reconstruction without interferences from other Kinects. In OmniKinect, we altered this approach slightly to gain more flexibility, as detailed in Section 3.

The FreeCam system [9] combines color cameras and depth cameras in a system for free-viewpoint rendering. They use a multi-camera rig instead of a full capturing room.

3. OMNIKINECT SYSTEM

The OmniKinect system provides a way to capture, record and stream information using a multiple Kinect sensors infrastructure, for both static or dynamic sensors. We propose a hardware setup and a list of software tools that can be used for a large number of applications. Our software tools include a set of basic capturing tools (record, filter, export) and a set of high level software components (tracking, visual hull rendering), which have been optimized for this specific system.

Setup overview: Our basic setup consists of an extensible, ceiling mounted aluminum frame with rigidly fixed vertical rods at regular distances. We have attached Kinect for Windows devices to the rods with stiffened foot joints. To reduce interferences between the Kinects, the rods are equipped with vibrators. In contrast to previous work [12, 3], we do not mount the vibrators directly onto the Kinects but on the supporting structure. This has various advantages: First, we do not have to disassemble the Kinects and demount their foots to mount the vibrators at a center position and ensure a stiff mounting. We have also tried a mounting on top of the Kinect, which revealed to be hard to control and to mount because of the bent shape of the Kinect and which produces much more image blurring than in our setup. Second, we can adjust and fine-tune the vibration amplitude by the position of the Kinect. Since the rods are not mounted on the floor, they vibrate at a higher amplitude near to their end/bottom, where the vibrator is mounted. The vibrator frequency can be controlled by an adjustable power supply.

Currently, our setup uses eight vibrating rods. Additional rods can be inserted with just a few simple steps in less than five minutes. To reduce clutter and to allow defined lighting conditions, we have surrounded the setup by two layers of curtains. We have measured the light filter effect of each curtain and can adjust the layers to reduce the incoming light by approximately 25% (one layer) or 50% (two layers). Figure 1(a) shows a schematic illustration of our setup and Figure 1(b) shows the current implementation of this setup.

Mounting: We use 1400mm long, 40×40mm *profile 8* pieces from *item Industrietechnik GmbH* (`http://www.item24.de`) as vertical rods on a 3450×3600mm ceiling mounted frame. The rods are mounted by using right-angled butt-fastening with T-slot nuts in the ceiling mount, to allow a rigid but slightly moveable connection for the vibration. Figure 1(a) shows the details for this approach. It is also possible to move one additional non-vibrating Kinect freely, as the reconstruction patterns of all mounted Kinects are blurred out by the Maimone/Butler method.

Vibrators and frequency control: Maimone *et al.* [12] do not give a lot of details for their choice of vibrators, hence we have experimented in the same way as Butler *et al.* [3] with different engines and offset weights to gain the optimal result. We finally chose an Igarashi *N2738-51* $12V$ motor with max. $14800rpm$ (idle running), $0.90Ncm$ torque and max. $11.8W$ output, a grub screw shaft connector, a $10g$ item "T-slot nut 8 St M8" as offset weight and hot glue as safety fixation. The engine's shaft connector is screwed into the thread of the T-nut. The vibrators are mounted with hot glue and tape to the vertical rods.

All vibrators are driven by a parallel circuit at slightly different frequencies, due to different cable lengths. In the final setup, we operate the vibrators between 7200 and $10200rpm$, which corresponds to a vibration frequency for each motor between 120 and $170Hz$. The final frequency adjustment has to be done manually, to reduce randomly occurring resonances with the mounting and therefore blurry RGB images for certain Kinects. Usually, $150Hz$ produces no disturbing resonances for our setup. Note that we use a higher frequency than proposed in [3], due to our vibrator mounting. We cannot measure the absorption caused by the vertical rods. Therefore we assume that the actual Kinect vibration in our setup is approximately at the same frequency as proposed by Butler *et al.* [3] $(60 - 80Hz)$.

Display device: To allow real-time visual feedback for various applications, we use a large TV LCD screen display, which can be freely positioned within our setup.

Control unit: As a control unit, we use an off-the-shelf PC with an ASUS Sabertooth X58 mainboard (two on-board USB 2.0 controllers and one USB 3.0 controller), an Intel Core i7 980X processor, 16 GB RAM, an NVIDA Quadro 6000 graphics card and four additional VIA USB 3.0 controllers. We also use powered USB extenders. Note that for the setup of multiple Kinects, only the number of physical

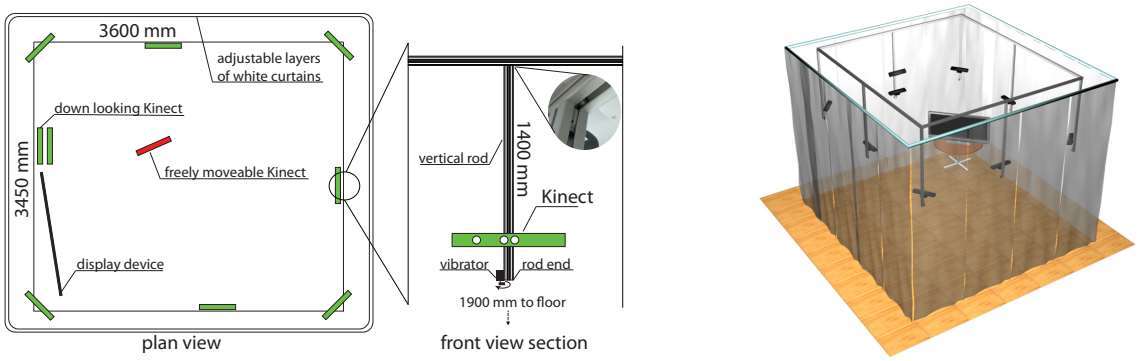

Figure 1: Plan views (a) and 3D overview (b) of our OmniKinect setup. In (a), vibrating Kinects are marked green and not vibrating Kinects red.

USB controller chips is important and not the number of USB ports. With this system, we can successfully operate 12 Kinects if only the RGB or the depth stream is used and 7 Kinects if both streams are used. This is mainly due to the limited bandwidth of the mainboard's south-bridge controller. As driver, we can use either the Microsoft Kinect SDK Driver, or, as for most of our example applications, the OpenNI (`http://openni.org`) PrimeSense Driver.

Calibration: We considered intrinsic, extrinsic, and depth calibration parameters.

As intrinsic camera parameters, we use the values given in the Microsoft Kinect SDK. However, for most of our target applications, we also need an initial extrinsic camera calibration for each Kinect.

We obtain extrinsic camera parameters using a $1300mm$ high calibration target, shown in Figure 2 with StbTracker [20] targets. Each of the target's 3 sides ($400 \times 500mm$) has 4 marker. The back-projection error for this target is in the sub-pixel range and can therefore be neglected.

The external calibration is computed on the RGB camera image. The depth image is transformed into the coordinate system of the RGB image by using the static transformation given by the OpenNI or Microsoft Kinect SDK.

Additionally, we provide a method to overcome depth inaccuracies between the Kinects. Each Kinect generates slightly different depth values, which can lead to holes or overlaps for the resulting reconstructions. For applications that require highly accurate multi-depth measurements, we perform an additional inter-Kinect depth calibration step using a red sphere ($\varnothing 120mm$), which is moved through the reconstruction volume (Figure 2 (c)). This sphere is segmented in the RGB image stream, and by mapping the depth stream onto the RGB stream also its depth values are obtained. Relying on the extrinsic and intrinsic calibration of the statically mounted cameras, the sphere's position in the room is triangulated. For each Kinect, the depth error d_{err} for one measurement is computed, taking the difference between the Kinect's estimated depth value d_{in} and the depth value obtained by the triangulated position of the sphere. After estimating depth errors for the entire calibration sequence, we fit a three dimensional polynomial function f_{err} (with the pixel coordinates p_x, p_y, and d_{in} as input) to the depth errors:

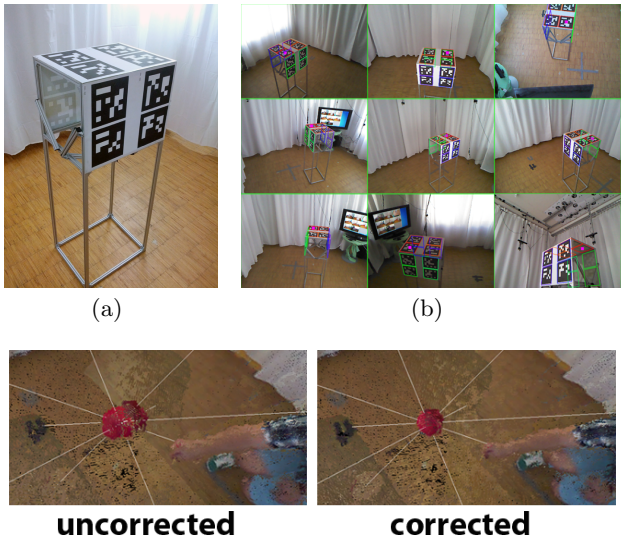

Figure 2: The StbTracker calibration target ($1300 \times 400 \times 500mm$) to gain initial extrinsic camera parameters (a) and the initial calibration view (b) showing the coordinate frame center for one example configuration using nine Kinects. (c) shows a point cloud rendering of the depth calibration target before (left) and after (right) correction with the camera viewing rays (white lines).

$$f_{err}(p_x, p_y, d_{in}) = \sum_{x,y,z} p_x^x \cdot p_y^y \cdot d_{in}^z \cdot c_{x,y,z} \stackrel{!}{=} d_{err}, \qquad (1)$$

with the exponents x, y, z running from zero to a maximum degree of four. According to our measurements, a maximum exponent of 1 for x, 1 for y, and 3 for depth works already sufficiently well and does not suffer from over-fitting. The polynomial coefficients $c_{x,y,z}$ are obtained by an SVD decomposition and stored in a calibration file. Using the coefficients the estimated depth error for unseen inputs can be computed efficiently. Adding this estimate to the reported

depth value of the Kinect yields the corrected depth value $d_{out} = d_{in} + d_{err}$. Using a single 3D polynomial guarantees a smooth transaction between neighboring locations.

Note that this approach needs to be carried out only once, even if the setup is changed slightly, as the depth calibration is obtained for each individual Kinect and it is only parametrized by the local pixel coordinates and the depth estimate. In principle, it would be possible to mount such spheres statically in our setup and to evaluate their position continuously. However, this does not seem to be necessary, as the depth error is not time varying. Additionally, such a setup would significantly reduce the available leeway.

Recording and playback: We use the OpenNI recorder and player feature to record or play back a sequence for each Kinect. Thereby, the maximum movement speed of objects is limited by the capture rate of the depth sensor, which is currently 30 frames per second.

Costs: Including all Kinects, the PC, the mounting and the motors, our whole system can be built with less than 5000 USD.

4. HIGH LEVEL COMPONENTS

4.1 OmniKinect fusion

Our experiments have shown that a straight forward fusion of depth maps from different sources is not possible, mainly due to the variations in the depth map accuracy between different Kinects. The user could use our extended polynomial calibration method from Section 3. However, this method might not always be feasible because of sparsely overlapping field-of-views within the sensor arrangement and the generally high time-effort for the calibration procedure. Therefore, we have extended the KinectFusion algorithm presented by Newcombe *et al.* [15] to work properly also with simultaneous uncorrected input streams from multiple Kinects. This approach uses only an initial extrinsic pose estimation of the cameras. We introduce an additional step for the algorithm as shown in Figure 3, which uses a smoothed histogram volume of truncated signed distance functions (TSDF) [4] to filter outlier measurements of the signed distance field before a temporal smoothing. In this way, persistent outliers due to variations in the registered pose or depth accuracies are removed, yielding a more robust estimate of the surface generating a complete and accurate reconstruction of the observed volume.

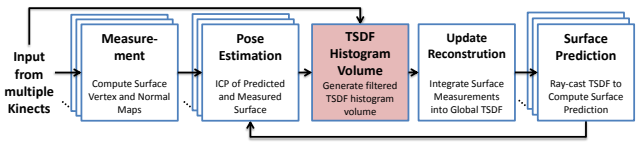

Figure 3: Overall system workflow for the modified KinectFusion algorithm [15] to support multiple simultaneous Kinects with different inaccuracies. The additional step is marked as red center square.

In our technique, we introduce a new volume consisting of a discrete histogram for the TSDF per voxel. We calculate the TSDF similar to Newcombe *et al.* with a ramp length of $\pm\mu$. The true surface is assumed to lie within this interval. Therefore, the TSDF $f_{R_k}(v,d)$ at a voxel v for device d with a distance η to the measured depth is given as

$$f_{R_k}(v,d) = \Psi(\eta) = \begin{cases} \min(1, \frac{\eta}{\mu}) & \text{if} \quad \eta \geq -\mu \\ null & \text{otherwise.} \end{cases} \tag{2}$$

$\Psi(\eta)$ lies in the interval $[-1,1]$ and is positive in free space, negative behind the surface. Instead of a direct weighted integration into the TSDF volume, we count the values of $\Psi(\eta)$ from each depth map at a voxel v in one histogram $\Theta(v)$ per voxel and increment the corresponding histogram bin with λ:

$$\begin{array}{lcl} \Theta_i(v) & = & \Theta_i(v) + \lambda, \lambda = 1 \\ i & = & \left\lfloor (f_{R_k}(v,d) + 1) * \frac{K}{2} + 0.5 \right\rfloor . \end{array} \tag{3}$$

The number of bins K can be freely chosen and defines another level of under-sampling for the TSDF. To allow a good coverage of zero-crossings, it is sufficient to choose this value as a small odd number. During our experiments, $K = 5, 7, 9, 11$ provided good results. Because of this discretization step, the histogram transformation behaves like an infinite impulse response (IIR) filter. The more bins are used for the histogram, the better edges and details are represented.

After the TSDFs of all input devices have been evaluated, each histogram bin is filtered separately with a three-dimensional bilateral filter to obtain a discontinuity preserving TSDF histogram volume with reduced noise per layer θ_i of $\Theta(v)$ between the measurements of the separate Kinects:

$$\theta_i(v) = \frac{1}{W} \sum_{q \in \mathcal{V}} \mathcal{N}_{\sigma_s}(\|v-q\|_2) \mathcal{N}_{\sigma_r}(\|\theta_i(v) - \theta_i(q)\|_2) \theta_i(q), \tag{4}$$

for each voxel $v = (x,y,z)^T$ in the volume domain $v \in \mathcal{V} \in \mathbb{R}^3$, $\mathcal{N}_\sigma = exp(-t^2\sigma^{-2})$ (spatial domain $\mathcal{N}_{\sigma_s}$ and intesity range domain $\mathcal{N}_{\sigma_r}$) and W as a normalization constant. Note, that this filter is very similar to the bilateral depth map filter of [17] in $\mathbb{R}^2$, but performed in $\mathbb{R}^3$ at this point. A subsequent maximum search per voxel histogram gives a robust estimate of the underlying TSDF volume value because it acts like a majority vote decision or IIR filter.

The histogram bins $\Theta_i(v)$ are normalized to the upper limit of the used data structure for the filter step. The actual value of the histogram bins does not influence the following steps, because we transform the index of the maximum of these histograms back into a regular TSDF volume ψ, using weighted averaging for this integration. Thereby, we use the inverse transform to Equation 3:

$$\begin{array}{lcl} \psi(v) & = & -1 - \frac{1-2\alpha}{K}, \text{with } K \neq 0 \text{ and} \\ \alpha & = & \arg\max_i(\theta_i(v)) \end{array} \tag{5}$$

We integrate that value into the final TSDF volume $F_{R_k}(v)$ with weighted averaging by storing a weight $W_k(v)$ with each value:

$$\begin{array}{lcl} F_{R_k}(v) & = & \frac{W_{k-1}(v) F_{k-1}(v) + W_{R_k}(v) \psi(v)}{W_{k-1}(v) + W_{R_k}(v)} \\ W_k(v) & = & W_{k-1}(v) + W_{R_k}(v) \end{array} \tag{6}$$

where $W_{R_k}(v) = 1$. Originally, this weight is proportional to $\cos(\tau)/R_k(x)$, where τ is the angle between the associated pixel ray direction and the surface normal measurement. Newcombe *et al.* [15] showed that a simple average

with $W_{R_k}(v) = 1$ provides good results. Our tests showed the same good performance with $W_{R_k}(v) = 1$. A similar weight can be used to increase the count in a histogram bin in Equation 3 by setting $\lambda = \lfloor 1 + |\cos(\tau)/R_k(x)| + 0.5 \rfloor$. With this approach, the tracking and reconstruction performance can be improved for certain scenes.

Note, that we do not reproduce the exact values of the input TSDF from Equation 3 in Equation 5. We are only interested in the majority vote for a zero-crossing of the different input devices. Therefore, exact numerical values are not necessary. The second volume implements a temporal smoothing, filtering temporal noise in the depth values.

With our method the standard KinectFusion *pose estimation* step can overcome calibration inaccuracies and disagreements of the depth sensors by adjusting the cameras' positions and orientations to positions in space, where most Kinects agree in their measurements. Furthermore, the vibration of the Kinects does not impair the transformation of the depth values into the static frame of reference because the vibration frequency is much higher than the image acquisition rate and possibly remaining artifacts are filtered out by integration over time. Figure 5 in Section 5.1 shows a comparison between a straight forward application of the original KinectFusion algorithm and OmniKinect fusion.

4.2 3D video and free viewpoint rendering

Capturing and rendering 3D videos is an important component of a variety of applications. For example, in 3D teleconferencing each participant needs to be rendered as seen from the observer's viewpoint. For digital entertainment and virtual try-on applications, a user wants to see herself immediately on a screen, immersed in a virtual world and augmented with virtual objects. To demonstrate the utility of the suggested system for these applications, we have implemented a free viewpoint rendering algorithm to display the 3D scene interactively.

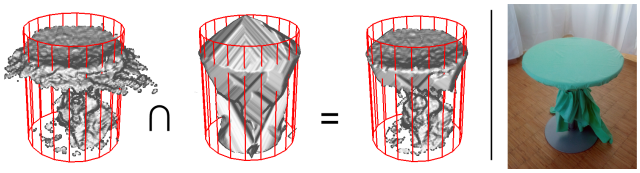

Figure 4: Free-viewpoint rendering of a static round table. From left to right: Point cloud rendering directly from Kinect depth maps suffers from low-quality edges. Visual hulls, on the other hand, have sharp edges, but not enough concavities. By intersecting the two surface representations, we are able to achieve a much more desirable result. The red cylinders indicate the ground truth diameter of the table on the right.

The image-based visual hull (IBVH) algorithm reconstructs and renders 3D objects from silhouette images. It is very efficient compared to stereo matching techniques. Additionally, it has the advantage of avoiding an explicit intermediate data representation, such as a voxel grid (as computed by KinectFusion) or mesh. It derives output depth maps directly from silhouette images and thus only computes surface regions that are actually visible. It can therefore provide pure dense image synthesis faster than with KinectFusion.

The IBVH algorithm produces depth maps of the object with clear edges, watertight topology, and no noise, even when using cheap cameras. However, it does not incorporate the depth data that is available in our system: it was designed to work with standard color cameras. Therefore, it fails to reconstruct some of the concavities.

In contrast to color images, depth maps from Microsoft Kinect are rather noisy and suffer from occlusions at depth discontinuities. At the same time, they convey more information about the shape of the scene. To get the advantages from both depth map-based rendering and image-based visual hull rendering, we combine the strengths of both approaches as follows.

Silhouette-carved point clouds: A prerequisite for visual hulls are silhouette images. To obtain silhouette images, the scene needs to be captured without foreground objects before the visualization starts. We use background subtraction based on color- and depth values to segment foreground objects. This method is more robust than relying on color or depth alone.

Our first method to incorporate visual hull information into point-based rendering is to carve the point clouds on the image planes by only considering depth values that are inside the silhouettes. Silhouettes are calculated by using binary foreground masks, which can be generated by using the aforementioned background subtraction. During our experiments we observed that some of the depth values are outliers even though they are inside their respective silhouettes.

Visual hull-carved point clouds: The visual hull of an object is a conservative surface estimate: It contains the whole object plus space that does not belong to the object. To remove low-quality depth values that caused noise in the silhouette-carved approach, we restrict the point cloud from all Kinects to the 3D space that is covered by the visual hull.

To do so, we perform IBVH rendering followed by point splatting to get both surface estimates. Then, all point splats are culled against the visual hull surface at that pixel. The result can be seen in Figure 4: concavities are reconstructed while sharp and precise edges are preserved.

5. EVALUATION AND RESULTS

5.1 OmniKinect fusion performance

To evaluate the performance of the OmniKinect fusion algorithm from Section 4.1, we have recorded several static scenes and compared their reconstruction results to a direct adaptation of the KinectFusion algorithm. The direct method integrates the TSDF values of each Kinect into a common volume with atomic operations. Figure 5 shows a comparison of horizontal and vertical slices through the TSDF volumes of these two approaches with a selected scene, which contains a simple box. Another qualitative comparison is given in Figure 6, using a reconstruction of the resulting zero-level set of the same round table object as shown in in Figure 4. Both reconstructions used the same parameters with $\mu = 0.1[m]$ and a TDFS size of 256^3 for the surface predictions, which were dumped after 10 frames. The corrugated regions in Figure 5 are the curtains around the setup.

Our additional computations introduce an average overhead of $3 - 9ms$ for the TSDF histogram evaluation per Kinect and $7 - 16ms$ for the histogram filtering, depending on the volume size and number of used histogram bins. A detailed run-time analysis for the original KinectFusion

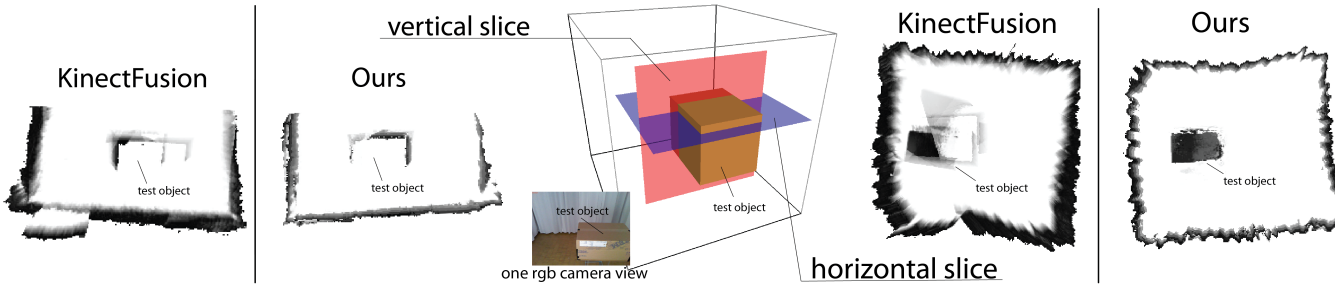

Figure 5: Overview over one of the test scenes using a large brown cardboard box and a comparison between cutting planes of the TSDF of a straight forward application of the KinectFusion algorithm to our approach.

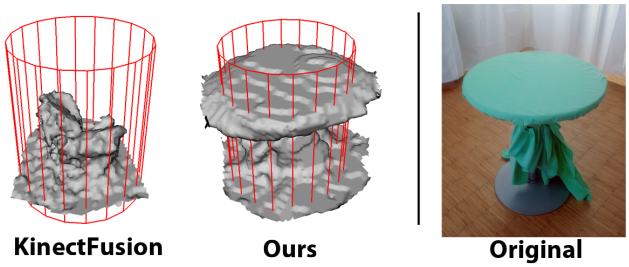

Figure 6: A qualitative comparison of the zero-level set resulting from the direct application of KinectFusion and our approach with the same object as shown in Figure 4 after 10 frames of recording time. For the direct approach (left), no clear zero-crossing can be extracted for the table surface. Similar to Figure 4, the red cylinders indicate the ground truth diameter of the table on the right.

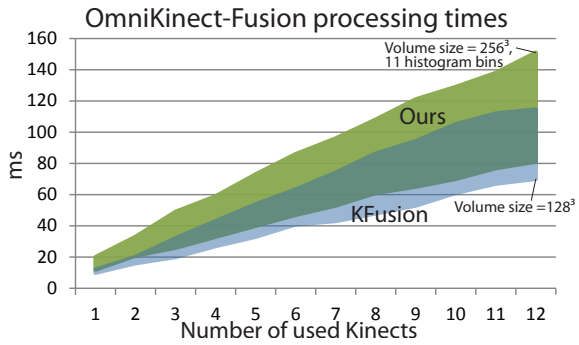

Figure 7: Evaluation of different configurations of OmniKinect fusion (green) compared to a straight forward implementation of the KinectFusion algorithm for multiple Kinects (blue). The KinectFusion range covers tests with TSDF volume sizes between 128^3 and 256^3. The tests for our method cover the same TSDF volume sizes. Additionally histograms with 5, 7, 9 and 11 bins have been measured for these volume sizes.

algorithm with multiple Kinect support and our system is given in Figure 7. We have performed our tests on a Nvidia GTX680 and average the overall processing times over 50 full computation cycles.

5.2 Free viewpoint rendering performance

To evaluate the quality of the OmniKinect setup, we captured an object of known size: a table with circular top (see Figure 4). We then reconstructed and rendered the object with a varying number of sensors with two methods: visual hulls and point cloud rendering.

Figure 8 shows the results. For two Kinects, the point cloud that is rendered from the depth data is already quite good when compared to the visual hull. One, two or even three silhouette images do not contain enough information to reconstruct such a surface in a meaningful way.

For a larger number of sensors, the rendering quality of the visual hull improves. It surpasses the quality of point cloud rendering at the edges of the object. The reason for this is that the Kinect cameras can be calibrated intrinsically and extrinsically very exactly, which directly translates to precise visual hull edges.

IBVH avoids explicit data representations and is output-driven: for every output pixel exactly one surface intersection is computed. Invisible parts or backsides of the object are not computed, which makes it very efficient. Point splatting is also a very efficient algorithm because modern GPUs are built for fast geometry transformations. The required scatter operation is also significantly faster to compute with recent CUDA versions. As a result, the visual-hull carved point cloud algorithm takes around 40 ms to compute on an Nvidia Quadro 6000 at a resolution of 1000 × 1000 pixels using 7 Kinects.

6. APPLICATIONS

In this section, we present selected application examples that leverage the OmniKinect system.

Full body scanning: An obvious application for our OmniKinect-fusion algorithm is detailed smooth body capturing. The user or dummy is placed in the center of the room and the zero-level set of the TSDF volume is extracted as soon as sufficient integration quality has been reached. During our tests, we have achieved good results already after 5-10 frames. Figure 9 shows an example for that use case.

Consistent photometric registration: Photo-realistic AR requires a geometric and a photometric model of the real world, so that virtual objects can be embedded consistently with the real environment. For example, virtual and real objects should mutually occlude each other, and cast shadows upon each other. Using the OmniKinect fusion detailed in section 4.1, high quality geometric information of the real scene can be obtained almost instantaneously. To

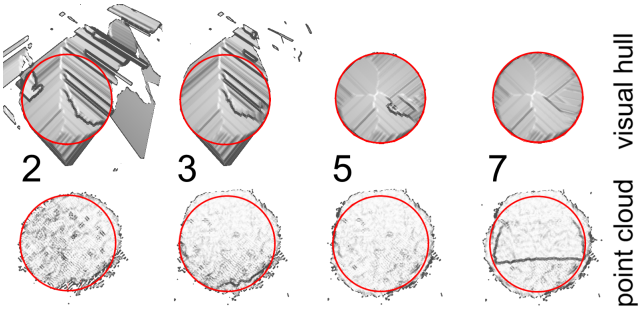

Figure 8: The top row shows visual hull rendering, the bottom row point cloud rendering for a varying number of Kinects. The red circle indicates the ground truth diameter of the reconstructed object.

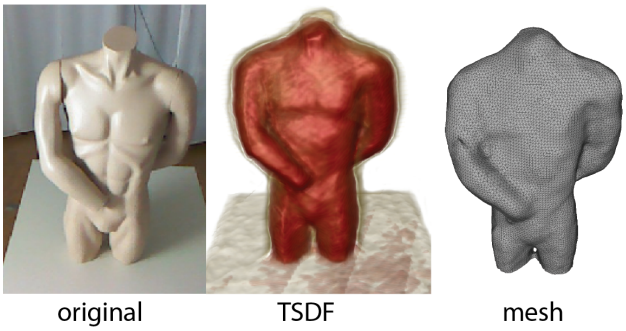

Figure 9: Example of a dummy body-scan application using OmniKinect fusion after ten frames integration time. A volume rendering of the resulting TSDF volume is shown in the center and the resulting mesh of the zero-level set at the right.

acquire an estimate of the real-world illumination in real time, previous approaches have relied on special light probe devices, such as a mirror ball or an omni-directional camera. OmniKinect's high quality volumetric model allows to use the scene itself as a light probe, and infer the current illumination from observations of the scene.

As proof of concept, we implemented a photometric registration application (Figure 10). For the light estimation, we simultaneously observe the reflection of the incident light on the scene geometry from multiple views. This allows measuring all outgoing light from the scene, which can be interpreted as a hemisphere of illumination directions. The final result is a combination of all light estimates from each camera view. For efficient real time computation, the illumination is represented using Spherical Harmonics.

X-ray visualization in Augmented Reality: Live volumetric representations of a real scene allow advanced X-ray visualizations in AR. Rather than simply combining virtual and real objects, we can leverage the volumetric representation to modify the appearance of a real object with respect to other real objects.

For example, X-ray visualization can make a real object translucent and show another real object occluded by the first one. Because the whole working volume is available in real time, both the occluder and the occluded can be dynamic, animated objects such as moving human beings.

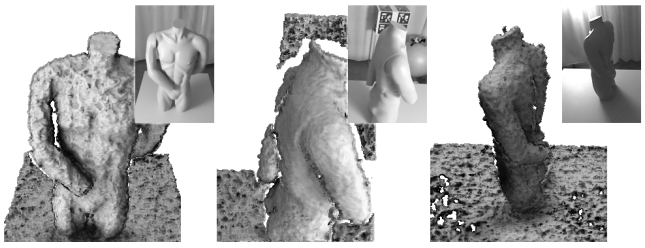

Figure 10: The three images show the reconstruction of the scene lit by the estimated environment light. For better comparison the input images of the camera are added as small inserts in the upper right corner. Note that the light situation is the same for all three images.

Technically, any combination of real and virtual views can be rendered to separate frame buffers, and then blended in a final step relying on a G-buffer approach [7]. Multiple objects can be extracted from the OmniKinect reconstruction using an appropriate segmentation method, for example by giving a bounding volume for a static object or by comparing a scene with and without a particular object. The object can then be rendered separately, and treated differently in the compositing, for example by modulating transparency.

Figure 11: X-Ray visualizations: see-through-me demo by segmenting a person (left), see inside the box with free-hand Kinect (middle), by comparing minimal differences in two scenes and using OmniKinect to see behind an object (right).

3D Magic Book: The OmniKinect setup can create a 3D AR Magic Book. Similar to 3D Live [16], we can record movement sequences and play them back in an AR environment. As our system is simpler to use, it can be easily deployed and provide a way to create a personalized 3D photo-book, a three-dimensional extension of online photo-book creation tools. The accompanying video presents an example of playing an AR book of a Karate training. Each of the sequences has been exported separately in an offline process, converted to a point cloud and saved to a compressed format supporting animation.

3D Magic Mirror: OmniKinect combined with a 3D autostereoscopic display can be used for a Magic Mirror metaphor where the user interacts with the reconstructed sequence (see Figure 12). Especially in sports that require specific postures the trainee can directly observe the scene in full 3D. Furthermore, the OmniKinect fusion reconstruction can be used to create a real-time differential 3D reconstruction to analyze diverging movements.

Figure 12: MagicMirror using the OmniKinect for generating a virtual trainer.

7. CONCLUSIONS

We have presented an easy-to-build, flexible, and affordable system for real-time 3D reconstruction and free-viewpoint rendering of real-world scenes. While conceptually simple, the calibration and fusion of data from multiple depth sensors requires careful design and deployment of the hardware setup and advanced algorithms for data processing and analysis. We have successfully improved the KinectFusion algorithm to accommodate multiple sensors simultaneously. Using this foundation, it is easy to implement depth segmentation algorithms and geometrically aware AR for dense dynamic scenes.

For 3D video capture and free-viewpoint rendering of moving and deforming objects, we introduced visual-hull carved point cloud rendering. It combines the advantages of image-based visual hull and point cloud rendering: precise edges and support for concave objects.

As future work we will conduct more thorough quantitative and qualitative evaluations, and investigate more applications. Currently, we are also working on an automatic marker-less online depth value correction, which will ease the initial calibration procedure significantly.

Acknowledgments: This work was supported by the Christian Doppler Laboratory for Handheld Augmented Reality, the Austrian Science Fund (FWF): P23329 and P24021, and the Austrian Research Promotion Agency (FFG) under the BRIDGE program, project 822702 (NARKISSOS).

8. REFERENCES

[1] K. Berger, K. Ruhl, C. Brümmer, Y. Schröder, A. Scholz, and M. Magnor. Markerless motion capture using multiple color-depth sensors. In *Proc. VMV 2011*, pages 317–324, Oct. 2011.

[2] E. Boyer. A hybrid approach for computing visual hulls of complex objects. In *Proc. IEEE CVPR*, pages 695–701, 2003.

[3] A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim. Shake'n'sense: reducing interference for overlapping structured light depth cameras. In *Proc. CHI '12*, pages 1933–1936, 2012.

[4] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH '96*, pages 303–312, 1996.

[5] U. Dhond and J. Aggarwal. Structure from stereo-a review. *IEEE Trans. on Systems, Man and Cybernetics*, 19(6):1489 –1510, nov/dec 1989.

[6] L. Guan, J.-S. Franco, and M. Pollefeys. 3D Object Reconstruction with Heterogeneous Sensor Data. In *Proc. 3DPVT*, 2008.

[7] D. Kalkofen, E. Mendez, and D. Schmalstieg. Comprehensible visualization for augmented reality. *IEEE TVCG*, 15(2):193–204, 2009.

[8] Y. M. Kim, D. Chan, C. Theobalt, and S. Thrun. Design and calibration of a multi-view TOF sensor fusion system. In *Proc. IEEE CVPR Workshops*, pages 1 –7, June 2008.

[9] C. Kuster, T. Popa, C. Zach, C. Gotsman, and M. Gross. Freecam: A hybrid camera system for interactive free-viewpoint video. In *Proc. VMV*, 2011.

[10] M. Li. *Towards Real-Time Novel View Synthesis Using Visual Hulls*. PhD thesis, Universität des Saarlandes, 2004.

[11] A. Maimone and H. Fuchs. Encumbrance-free telepresence system with real-time 3D capture and display using commodity depth cameras. In *Proc. ISMAR '11*, pages 137–146, 2011.

[12] A. Maimone and H. Fuchs. Reducing interference between multiple structured light depth sensors using motion. In *Proc. IEEE VR*, pages 51–54, 2012.

[13] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *Proc. ACM SIGGRAPH '00*, pages 369–374, 2000.

[14] W. Matusik and H. Pfister. 3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *Proc. SIGGRAPH '04*, pages 814–824, 2004.

[15] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc IEEE ISMAR '11*, pages 127–136, 2011.

[16] S. Prince, A. D. Cheok, F. Farbiz, T. Williamson, N. Johnson, M. Billinghurst, and H. Kato. 3D Live: Real Time Captured Content for Mixed Reality. In *Proc. IEEE ISMAR '02*, page 7, 2002.

[17] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. In *Proc. IEEE CVPR '11*, pages 3017–3024, 2011.

[18] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proc. IEEE CVPR'11*, 2011.

[19] J. Tong, J. Zhou, L. Liu, Z. Pan, and H. Yan. Scanning 3d full human bodies using kinects. *IEEE TVCG*, pages 643–650, 2012.

[20] D. Wagner, T. Langlotz, and D. Schmalstieg. Robust und unobtrusive marker tracking on mobile phones. In *Proc. IEEE ISMAR'08*, pages 121–124, 2008.

[21] A. D. Wilson and H. Benko. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proc. ACM UIST '10*, pages 273–282, 2010.

[22] R. Yang, G. Welch, and G. Bishop. Real-time consensus-based scene reconstruction using commodity graphics hardware. In *Proc. Computer Graphics and Applications 2002*, pages 225 – 234, 2002.