Preprint. The final publication is available at www.springerlink.com. (DOI: 10.1007/s10055-011-0203-0)

Reducing the Gap Between Augmented Reality and 3D Modeling with Real-Time Depth Imaging

Svenja Kahn

Abstract Whereas 3D surface models are often used for augmented reality (e.g., for occlusion handling or model-based camera tracking), the creation and the use of such dense 3D models in augmented reality applications usually are two separated processes. The 3D surface models are often created in offline preparation steps, which makes it difficult to detect changes and to adapt the 3D model to these changes. This work presents a 3D change detection and model adjustment framework that combines AR techniques with real-time depth imaging to close the loop between dense 3D modeling and augmented reality. The proposed method detects the differences between a scene and a 3D model of the scene in real time. Then, the detected geometric differences are used to update the 3D model, thus bringing AR and 3D modeling closer together. The accuracy of the geometric difference detection depends on the depth measurement accuracy as well as on the accuracy of the intrinsic and extrinsic parameters. To evaluate the influence of these parameters, several experiments were conducted with simulated ground truth data. Furthermore, the evaluation shows the applicability of AR and depth image-based 3D modeling for model-based camera tracking.

Keywords 3D modeling · Augmented Reality · Computer vision · Tracking · Depth imaging · Analysis-by-synthesis

1 Introduction

Augmented reality (AR) applications combine real and virtual, are interactive in real time, and registered in 3D (Azuma 1997). Since Azuma first stated these characteristics of AR applications in 1997, Augmented Reality has matured remarkably (Zhou et al. 2008). However, an important bottleneck remains: The availability of dense 3D surface models. Such dense 3D models are of uppermost importance for two different augmented reality aspects. First, a 3D model is needed for a smooth and seamless integration of virtual objects into the camera images. Therefore, the virtual objects should be illuminated in a consistent way with the illumination of the real scene, they should cast shadows, and they should be occluded by parts of the real scene which are closer than the virtual object. Both occlusion handling and shadow calculation require knowledge about the 3D structure of the real scene (Haller 2004; Panagopoulos et al. 2009). Furthermore, dense 3D models are often used for model-based camera tracking, both for the camera pose initialization and for model-based frame-to-frame tracking (Lepetit and Fua 2005). The model-based estimation of the camera pose has the advantage that it overcomes the need to prepare the scene with fiducial markers (Gall et al. 2006).

Recently, depth cameras have been developed which acquire dense distance measurements in real time. Depth cameras can be used to create realistic AR applications with shadow mapping or occlusion culling even if no 3D model of the real scene exists (Franke et al. 2011). However, these approaches have several drawbacks. First, the user needs to move a bulky dual-camera setup comprising both a depth and a color camera (e.g., the Kinect has a width of 28 cm and the state-of-the-art SR4000 depth camera weights 500 g, without the additional color camera). A 3D model offers better device independence: It can also be used by mobile devices such as UMPCs, which offer powerful processors and built-in 2D cameras and which are thus suited for model-based AR, but which have no integrated depth measurement devices. Another disadvantage of real-time depth imaging without a 3D model is that the 2D camera and the depth camera capture the scene from different viewpoints. Thus due to occlusions, there is no complete mapping between the color pixels and the depth measurements (Lindner et al. 2007). Further artifacts arise in fast camera movements if the

S. Kahn
Fraunhofer IGD, Darmstadt, Germany
E-mail: svenja.kahn@igd.fraunhofer.de

color and the depth camera capture the images at slightly different points in time. Finally, a 3D model can provide more stable 2D-3D correspondences for model-based camera tracking than depth measurements that are captured on the fly. For example, depth measurement artifacts occur due to motion blur effects if the depth camera is moved quickly. In the 3D model creation process, these artifacts can easily be circumvented by slow camera movements. Therefore, if the user moves the camera quickly during the tracking, acquiring the 3D measurements from the depth camera on the fly would suffer from these artifacts, whereas the 3D model provides 3D surface information which is not influenced by this effect.

Currently, when it comes to dense 3D modeling, there is a gap between the 3D modeling phase and the use of the 3D models by AR applications (see Sect. 2). This work presents an augmented reality 3D modeling framework which reduces this gap by combining real-time depth imaging with AR (Sect. 3). In a first step, geometric differences between the real scene and the 3D model are detected in real time by aligning dense depth measurements of a real scene with a 3D model of this scene (Sect. 4). Then in a second step, the detected geometric differences are used to adapt the 3D model to the real scene, either manually or automatically (Sect. 5). The accuracy of the geometric difference detection depends on the depth measurement accuracy as well as on the accuracy of the intrinsic and extrinsic parameters. To evaluate the influence of these parameters, several experiments were conducted with simulated ground truth data. The results of these experiments are presented in Sect. 6. They quantify the influence of the camera tracking accuracy and the measurement noise on the AR difference detection method. Further experiments show the applicability of 3D model adjustment for model-based camera tracking.

2 State-of-the-art

The major challenge of using dense 3D models for AR is that the 3D models need to be adapted whenever some part of the scene changes (e.g., if new objects were added to the tracked scene or objects were moved to another position).

2.1 3D modeling for Augmented Reality

Most augmented reality techniques that make use of dense 3D models implicitly assume that such a 3D model already exists (Gall et al. 2006; Wuest 2008). Therefore, the reconstruction process is often decoupled from the augmented reality application and the reconstruction is done in an offline preparation phase before the AR application can be used (Wuest et al. 2007; Kahn et al. 2010a). This strict separation between the modeling process and the application of the created 3D model for AR causes two major problems: First, the occurrence of a change is often not obvious in the first place. When a 3D model is used for model-based camera tracking, objects that are part of the modeled scene are often displaced partially in the real scene. This can cause drift and instabilities in the camera tracking. However, it is not always obvious that the tracking inaccuracies are due to the fact that the real scene was changed. Furthermore, even when the user is aware that the 3D model does not fit the reality any more, it is often a difficult task to find out how the 3D model needs to be adapted such that it correctly models the real scene again.

Recently, several methods have been proposed that use augmented reality for 3D reconstruction with a 2D camera. Pan et al. (2009) reconstruct a textured 3D model of an object by rotating the object in front of the camera. Whereas this approach provides good results for textured objects which fulfill the condition that they can be rotated in front of the camera, this method requires a static camera position and thus cannot be used for the reconstruction of larger scenes. The system presented by Van den Hengel et al. (2009) relies on user input for the reconstruction of textured objects. It estimates the camera pose for each image of a previously recorded video sequence. To reconstruct a captured object, the user manually specifies the 2D vertices of flat polygons that form the contour of the object in the 2D image. Then the 3D positions of these vertices are reconstructed with simultaneous localization and mapping. Whereas this method is well suited for the reconstruction of objects that are composed from planar surfaces, it would be difficult to model scenes with arbitrary, non-planar surfaces. Bastian et al. (2010) presented a method that reconstructs the surface of an object with silhouette-based voxel carving. This approach is based on the precondition that the boundary of the object can be identified in the 2D image and that the camera can be moved around the object. Scenes that do not fulfill this condition cannot be reconstructed with this approach. For example, it is not possible to reconstruct the walls of a room because the projection of the wall onto the 2D camera image always covers the whole image and no object boundary gets projected onto the 2D image.

Structure from motion reconstruction algorithms combine 3D reconstruction with camera tracking methods (Bleser et al. 2007; Bleser 2009; Klein and Murray 2009). Sparse features are reconstructed online while the pose of the camera is tracked with camera pose estimation algorithms. These online reconstruction algorithms for sparse features provide a smooth integration of 3D modeling and AR applications. This is currently often not the case for dense 3D models, which (except by the method presented by Newcombe and Davison (2010)) usually need to be modeled offline. This is partly due to the fact that in contrast to sparse 3D features dense surfaces can not easily be reconstructed from

2D camera images in real time, especially not if the object surfaces are not well textured. However, currently real-time depth imaging cameras have been developed which can help to bridge this gap.

2.2 Real-time depth imaging

Time-of-flight cameras acquire dense 3D measurements in real time (Oggier et al. 2006; Kolb et al. 2009). They emit near-infrared modulated light that gets reflected by the scene and captured by the image sensor of the camera. For every pixel, the distance to the scene is calculated by the phase shift of the reflected light. The distances can be transformed to Cartesian coordinates, yielding a 3D measurement per pixel. Time-of-flight cameras capture 3D measurements as well as an intensity image that depicts the amount of light reflected onto each pixel.

Furthermore, recently the first low-cost depth camera for the mass market was released by PrimeSense respectively by Microsoft. This depth camera is part of the Kinect device. Initially targeted at the gaming market, the Kinect depth camera can be used to realize AR applications with real-time depth imaging as well. It has a higher resolution than state-of-the-art time-of-flight cameras (SwissRanger 3000: 176×144 pixel, Kinect: 640×480 pixel). The Kinect contains two cameras (a color camera and the depth camera) as well as an infrared projector that projects an infrared pattern onto the scene. This infrared pattern is detected by the depth camera (which is in fact an infrared camera) and used to calculate the distance of the scene. The measured depth and color images of the Kinect can be accessed with the OpenNI interface (OpenNI 2011).

Huhle et al. (2008) as well as Engelhard et al. (2011) presented methods that use a depth and a color camera for 3D reconstruction. Both assemble colored 3D point clouds by first aligning the 3D point clouds based on 2D features in the color images. Then, the alignment is refined with 3D-3D geometric registration. Both approaches can be used to reconstruct 3D scenes from scratch. However, they do not cover the issue how information about an existing 3D model (e.g., a triangle mesh that already models a part of the scene) can be incorporated into the reconstruction process. The incorporation of an existing triangle mesh in the reconstruction process requires a method that can detect where the existing mesh differs from the real geometry (geometric difference detection) and where it thus needs to be updated.

2.3 Geometric difference detection

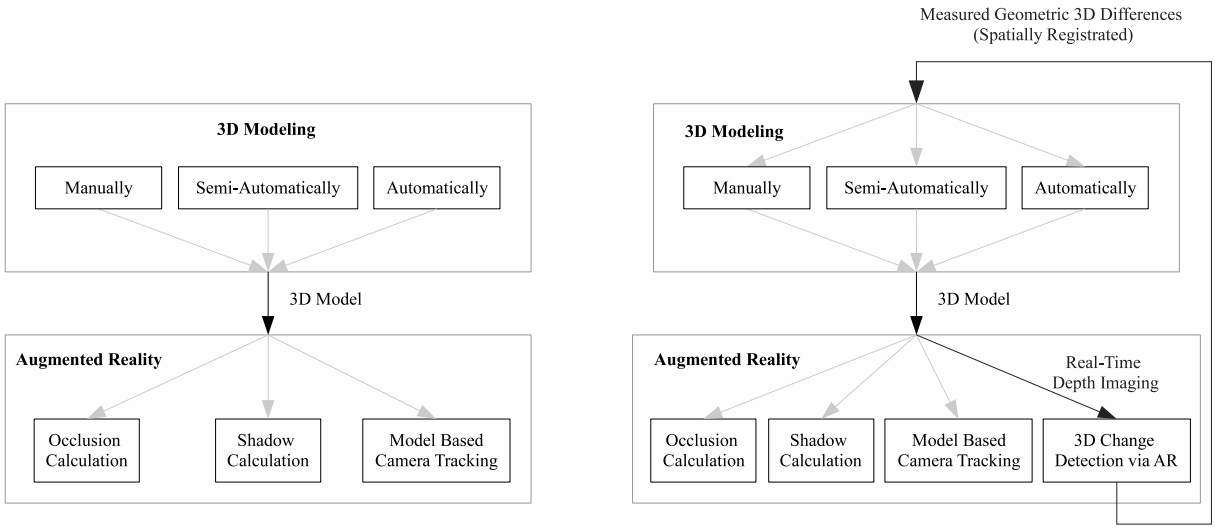
Kahn et al. (2010b) proposed the first 3D difference detection approach for a moving camera. This approach allows to move a camera freely in a scene and to automatically detect 3D differences in real time with a dense 3D discrepancy check. This paper extends the ideas that were first presented in (Kahn et al. 2010b). Previous solutions for discrepancy check were based on still 2D images: In their pioneering work, Georgel et al. (2007, 2009a,b) presented an augmented reality solution for discrepancy check in the context of construction planning. Their system allows engineers to superimpose still 2D images of a plant with the CAD model developed during the planning phase. Whereas this augmentation of still 2D images with the 3D model is very useful to visually compare the 3D model and the real scene, it is limited to the 2D information contained in the images and provides no possibility to automatically compare the 3D data of the model with the 3D geometry of the real scene. Webel et al. (2007) presented a system for AR discrepancy check with which the 3D positions of single points in the 3D model and the real scene can be compared: A laser pointer is used to depict a point on the surface of the real scene. The 3D coordinate of the point is reconstructed by triangulation with a stereo camera system. Whereas this approach allows the comparison of single 3D points, it is not suited for dense 3D difference detection.

Bosché et al. (2006) transformed the 3D data of a CAD model and 3D data measured with a time-of-flight camera into a common voxel occupancy grid. However, they used a static camera position and thus did not need to solve the registration problem. In recent publications, Bosché addressed the registration problem in the context of object recognition (Bosché 2008, 2010). He used manually specified 3D correspondences between a laser scan of a construction site and a 3D model of the construction site to transform both data sets into a common coordinate system. As the manual selection of 3D correspondences is an offline step (and for the task of discrepancy check, 3D-3D correspondences cannot easily be extracted automatically as the 3D model might differ from the real scene), this approach is not feasible for difference detection with an arbitrary moving camera.

3 Concept overview

Most frameworks for model-based augmented reality use a sequential one-way process for the 3D modeling task (see Fig. 1(a)). First a 3D model is created in a preparation step. Then, this 3D model is used for the realization of augmented reality applications. These conventional approaches do not offer the possibility to use AR for the 3D modeling step itself.

This limitation is overcome by the proposed framework (Fig. 1(b)). Similar to the work presented by Van den Hengel et al. (2009) and Bastian et al. (2010), it closes the loop between 3D reconstruction and augmented reality by employing AR for the 3D modeling process. In contrast to pre-



(a) Conventional framework (b) Proposed framework

Fig. 1 (a) Conventional framework: The 3D model is used as input for the AR application, but not vice versa. (b)Proposed framework: AR is used to detect geometric differences between the 3D model and the real scene. Then the measured geometric differences are fed back into the 3D modeling pipeline and are used to update the 3D model.

vious approaches, it furthermore combines AR with real-time depth imaging to detect geometric differences between the real scene and a 3D model of the scene. Thus, the 3D modeling is supported by the AR components of the framework. Vice versa, the reconstructed 3D model can be used for model-based AR applications. With the proposed framework, the user can reconstruct or adjust a 3D model of the scene with a depth-color camera setup. After the AR-based model adjustment step, AR applications that need 3D information about the scene can be realized with a single color camera, as the 3D information can be accessed from the 3D model.

The basic idea for 3D difference detection was first described in Kahn et al. (2010b). In this paper, it is extended for 3D modeling by AR. For the difference detection, the pose of a depth camera is estimated by calculating the pose of a rigidly coupled 2D camera with camera tracking algorithms. The tracked camera pose is then used to register the real depth measurements and the 3D model in a common coordinate system. By rendering the 3D model from the point of view of the depth camera, a simulated depth image can be generated which is compared to the real depth measurements to detect geometric differences (see Sect. 4).

In a next step, the measured geometric differences are fed back into the 3D modeling pipeline where they are used to update the 3D model either manually or semi-automatically (see Sect. 5). This update step benefits from the fact that the previous augmented reality difference detection registers the measured depth values and the 3D model in a common coordinate system. This eases the model adjustment task, both for the user and for (semi-)automatic model adjustment. Continuously updating the 3D model would require very complex algorithms for real-time 3D model processing. Furthermore, these algorithms would have to assure that pose estimation errors or measurement inaccuracies do not distort previously modeled parts. The approach presented in this paper is tailored to semi-automatic model adjustment: Rather than updating the 3D model fully automatically with each single depth measurements, the user takes a few snapshots of a 3D scene. Then the 3D model is automatically updated according to the depth measurements of these snapshots. The AR visualization of the geometric differences supports the user in identifying the viewpoints for new depth images and in judging the accuracy of the current camera pose estimation (which influences the accuracy of the 3D modeling, see Sect. 6).

4 Geometric difference detection

To detect geometric differences between a real scene and a 3D model of the scene, a color camera (used for the camera tracking) is rigidly coupled with a depth camera (for the 3D imaging). This section describes why both a 2D and a depth image are needed, how to calibrate the cameras, and how to use camera tracking in combination with analysis-by-synthesis to geometrically compare the real scene and the 3D model.

4.1 Combining 2D imaging with real-time depth imaging

By combining camera tracking based on 2D images with real-time depth imaging, dense 3D measurements can be acquired and registered with the coordinate system of the tracked scene in real time. This would not be possible if either one of these images (the 2D image or the depth image) was used alone: Whereas the color images of custom 2D cameras can be used to track the camera position, 2D cameras cannot capture dense depth images in real time (e.g., at untextured parts of a scene). Vice versa, depth cameras acquire dense depth images in real time but their pose cannot be tracked robustly with 2D image-based camera tracking algorithms. An alternative possibility to register the measured depth images with the 3D model would be to use geometric alignment, i.e., iteratively approach the 3D points with the Iterative Closest Point algorithm (Besl and McKay 1992). However, this approach would be computationally expensive. Furthermore, differences between the real scene and the 3D model would result in wrong registrations, which should be avoided for geometric difference detection tasks because here the 3D model can differ from the real scene.

For these reasons, two cameras are used: a custom 2D color camera and a depth camera. This combination of 2D imaging with 3D imaging is not restricted to specific devices. Whereas in Kahn et al. (2010b) a time-of-flight depth camera and a color camera were rigidly coupled on a camera rig, the Kinect camera of Microsoft (which contains both a depth and a color camera) can be used as well. Both systems have in common that the depth and the color camera are rigidly coupled and placed at different positions. Thus, the intrinsic parameters of both cameras as well as the relative transformation $(\Delta R, \Delta t)$ between the cameras need to be calculated. In contrast to the Kinect depth camera, the time-of-flight camera captures not only the depth image but also an intensity image that depicts the amount of light reprojected at each pixel. This intensity image can also be used to track the pose of the time-of-flight camera without a color camera. However, it has a much lower resolution than custom color cameras (typically 176×144 pixels) (Oggier et al. 2006).

4.2 Camera calibration

For a setup that combines a time-of-flight camera with a color camera, the MultiCameraCalibration tool (Schiller et al. 2008) calculates the relative transformation between the time-of-flight camera and the color camera as well as the intrinsic parameters of both cameras.

The main challenge for the calibration of a Kinect depth camera is that the 2D checkerboard patterns that are otherwise used for the calibration are not visible in the depth images. Therefore, the calibration of Kinect depth cameras requires a trick which is based on the insight that the depth camera is in fact an infrared camera (see Section 2.2). Instead of the depth image, the raw infrared image is acquired from the OpenNI interface. The infrared projector is covered with opaque tape to remove the infrared pattern from the image, and the chessboard pattern is illuminated with an external infrared lamp. This way the chessboard becomes visible in the infrared images of the depth camera. Then the KinectAutoCalibration can be used to calculate the intrinsic parameters as well as the relative transformation between the color and the depth camera of the Kinect (Engelhard 2010).

4.3 Camera tracking

After the 3D model was reconstructed or adjusted to the real scene, it can be used for model-based camera tracking. However, in the 3D model reconstruction phase, model-based camera tracking can be unstable: During the adjustment process, the 3D model does not yet correspond to the real geometry. This can result in errors or inaccuracies of the estimated camera pose. This is why marker tracking and structure from motion are used for the camera pose estimation in the 3D model adjustment phase. The camera pose is initialized with a marker whose coordinates are specified in the coordinate system of the given 3D model. Thus, the world coordinate system is identical to the model coordinate system. Marker-based camera tracking has the drawback that the camera pose can only be calculated if markers are visible in the current camera image. Therefore, we combine markerless camera tracking with structure from motion 3D reconstruction (Bleser et al. 2007). While the camera is moved, this approach detects characteristic image features with a Shi-Tomasi corner detector (Shi and Tomasi 1994) and tracks these 2D image features with a Lucas Kanade tracker (Wuest 2008). Then, the 3D positions of the tracked features are reconstructed online via triangulation (Bleser et al. 2007). Finally, the 2D-3D correspondences between the 2D image coordinates of the tracked features and their reconstructed 3D coordinates are used to calculate the camera pose if no marker is visible.

The pose (R_D, t_D) of the depth camera is calculated from the tracked pose of the color camera (R_C, t_C) with Eq. (1). Here $(\Delta R, \Delta t)$ is the relative transformation between the two cameras, which was calculated in the offline calibration step.

$$\begin{aligned} R_D &= R_C \cdot \Delta R \\ t_D &= R_C \cdot t_C + \Delta t \end{aligned} \tag{1}$$

4.4 Difference calculation via analysis-by-synthesis

To compare the 3D data acquired by the depth camera and the 3D model, the discrepancy between the 3D measurement

and the corresponding 3D position in the 3D model is calculated for each pixel via analysis-by-synthesis: First, the 3D model is rendered from the current camera pose with the intrinsic and extrinsic parameters of the depth camera. Then, a synthetic 3D image is calculated from the depth buffer of the graphics card. If P is the 4x4 projection matrix used for the rendering of the 3D model, each depth value is converted to the camera coordinate system with P^{-1}. After this conversion, the geometric differences between the 3D model and the real measurements are calculated pixelwise by comparing the depth value of the synthetic depth image and the depth measurement of the depth camera at the same pixel.

5 Model adjustment

The difference detection approach described in this paper contributes to geometric modeling in two different ways: On the one hand, differences are visualized to ease the model adaption task for the user. Furthermore, the 3D model can be adapted algorithmically such that its surface better corresponds to the measured scene.

5.1 Geometric difference visualization

To visualize the discrepancies, the camera image is augmented with a semi-transparent RGBA image whose colors represent the 3D differences. Red pixels show that the real scene is closer than its counterpart in the 3D model, at yellow pixels the real scene is farther away, and blue pixels show parts of the scene that do not exist in the 3D model. If the depth camera could not acquire a depth value at a certain pixel, this pixel is colored in green.

To visualize the degree of the discrepancies, the transparency α of each pixel in the difference visualization image is set such that pixels visualizing close distances have a higher transparency than pixels at positions where there is a large discrepancy between the 3D model and the real measurements: $\alpha = (d_{measured} - d_{3Dmodel}) \cdot o$. Here, $d_{measured}$ and $d_{3Dmodel}$ are the depth values. The opacity factor o can be set by the user to change the transparency of the whole discrepancy visualization image.

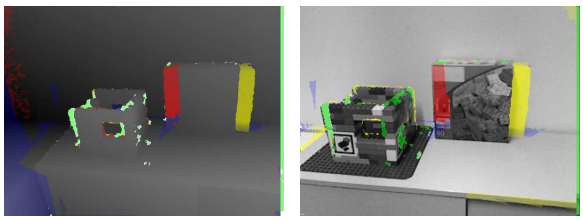
The image of the color camera provides more details about the scene than the grayscale-encoded depth image. However, if the color image is augmented, the differences are not well visible anymore because the colors of the 2D camera interfere with the colors of the difference visualization. Therefore, the color image is converted to a grayscale image. This way it is possible to visualize the (grayscale) details of the scene and the detected differences at the same time. The depth measurements of the depth camera need to be mapped onto the color image. Thus, the raw depth values are first transformed to 3D points and then projected onto the 2D image of the color camera. Equation (2) transforms the 1D depth value d_{cam} of a pixel (p_x, p_y) in the 2D image coordinate system of the depth camera to a 3D point $p_{CCS(D)}$ in the depth camera coordinate system CCS(D). The horizontal respectively vertical focal length is denoted by (f_x, f_y) and the principal point by (c_x, c_y). In a next step, the 3D points are transformed to the world coordinate system with Eq. (3).

$$p_{CCS(D)} = \begin{pmatrix} (p_x - c_x) * \frac{1}{f_x} * d_{cam} \\ (p_y - c_y) * \frac{1}{f_y} * d_{cam} \\ d_{cam} \end{pmatrix} \tag{2}$$

$$p_{WCS} = (R_D)^{-1} \cdot (p_{CCS(D)} - t_D) \tag{3}$$

Finally, Eq. (4) is used to project 3D points from the world coordinate system to 2D coordinates $p_{ICS(C)}$ in the camera image of the color camera. Here, K_C is the camera calibration matrix of the color camera which contains the intrinsic parameters.

$$p_{ICS(C)} = K_C((R_C \cdot p_{WCS}) + t_C) \tag{4}$$



(a) Depth image augmented with differences (b) Grayscale image augmented with differences

Fig. 2 Difference visualization (Kinect): in comparison to the 3D model, the box in the real scene was moved to the left.

Figure 2 shows the difference visualization of a box which that moved to the side. This displacement would cause problems for model-based camera tracking approaches because the box is well textured and thus many features are detected on the real box. If the 3D model does not match the position of the real object, wrong 3D positions would be inferred from the 3D model. With the approach proposed in this paper, the user can use the difference detection and visualization either to move the box back to the modeled position or to model the new box position in the 3D model (Fig. 3).

(a) Depth and color image (b) Colored 3D triangle mesh with virtual car

Fig. 3 (a) Depth and color image acquired by the Kinect. The black pixels in the depth image show pixels where the Kinect could not acquire depth values. (b) Colored 3D mesh created by mapping the colors to the depth image and by converting the depth values to the world coordinate system. The square markers on the brick model are used to initialize and to track the camera pose. The positions of the depth and the color camera are represented by the blue and orange camera symbols.

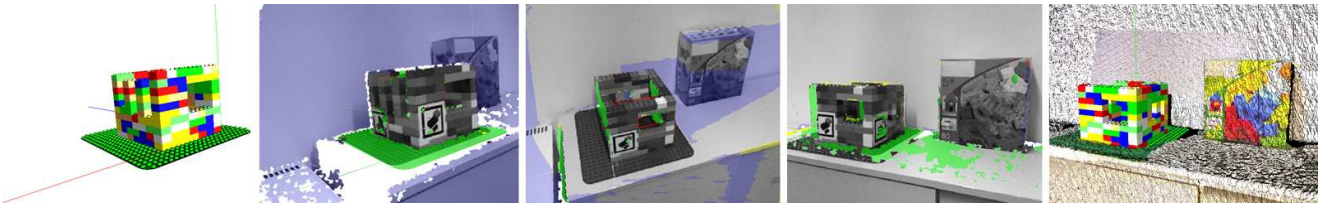


Fig. 4 Geometric model adjustment (Kinect): First only the 3D model of the bricks is known (parts of the scene which are not modeled are colored in blue). By adding submeshes for missing parts, the 3D model is gradually amended. Right: Adapted 3D model after three submeshes were added.

5.2 Geometric model adjustment

This section describes how the 3D model is geometrically adjusted. This is accomplished by inserting triangle meshes in the 3D model which cover the parts of the surface where the real scene was measured closer to the camera than the surface of the 3D model or where no 3D information was available beforehand (see Fig. 4). The described approach does not imply restrictions on the 3D model but is feasible for any kind of 3D model which can be rendered (e.g., polygonal meshes, implicit surfaces, or CAD models). The only precondition is that it is possible to store an additional triangle mesh in the used data format.

To adapt the 3D model, a triangle submesh is created which contains all the triangles of the depth measurements whose 3D measurements belong to pixels where a change is visualized. Thus when the user wants to adapt the 3D model to the depth measurements of the current viewpoint, he can change the threshold o (see Sect. 5.1) to increase or decrease the subparts of the measured depth image which are added to the scene. This triangle mesh is then added to the 3D model on the fly: The new triangle mesh is added as a new geometry node to the scenegraph system OpenSG (OpenSG 2011) which is used for the rendering of the 3D model.

To add 3D measurements of the depth camera to the geometry of the 3D model, the depth measurements need to be transformed to the coordinate system of the 3D model. As noted in Sect. 4.3, this corresponds to the world coordinate system. Therefore, the depth measurements are transformed to 3D points in the model coordinate system with Eqs. 2 and 3.

If the 3D model is a polygonal vertex mesh, the difference detection approach described in this paper can as well be used to remove parts of the 3D model where the time-of-flight camera measures distances that are farther away than the surface of the 3D model. In contrast to the previously described surface insertion technique, this modifies the vertices and the triangle structure of the 3D model. To remove the correct vertices of the 3D model, all vertices $v_i \in V$ (and the polygons adjacent to these vertices) need to be removed which fulfill the following two conditions:

1. v_i gets projected on one of the pixels where a difference was detected by the analysis-by-synthesis approach.
2. v_i is not hidden by another surface of the 3D model. This condition prevents the removal of hidden surfaces, for example the back side of the 3D model.

The vertices and triangles that fulfill these conditions are identified with a shader. The 3D model is rendered offscreen with a pixel buffer. To detect the visible vertices and triangles, each triangle is drawn with a different color. Then, the

rendered color is a mapping to the triangle which is visible at this pixel.

6 Quantitative evaluation

The accuracy of the 3D difference detection depends on the accuracy of the relative transformation between the two cameras, the accuracy of the camera pose estimation of the 2D camera, and the accuracy of the intrinsic calibrations of the 2D camera and the depth camera. When one of these parameters is inaccurately estimated, there is an offset between the camera pose used for the analysis-by-synthesis approach and the real camera pose. To quantify the influence of these parameters, we evaluated the difference detection and the 3D modeling with a simulation providing ground truth data for both the camera pose and the 3D geometry of the scene.

6.1 Extrinsic and intrinsic parameters

In the first setup, we used a 3D model of the toy brick scene to create simulated depth and color images while varying the parameters of the virtual cameras. For the evaluation, the difference detection was applied on a depth image which was simulated with the correct parameters (reference image) in combination with a depth image which was generated with the modified extrinsic or intrinsic parameters (evaluation image).

Figure 5 visualizes the results of this evaluation. The top row shows the differences between the reference and the evaluation image which occur due to the errors in the extrinsic and intrinsic camera parameters. The plots beneath show how the residual increases subject to the intrinsic and extrinsic inaccuracies. Here, the residual is the average of the pixelwise difference of both depth images. Each parameter was evaluated independently. Please note that for the evaluation of the rotation offsets, the camera was rotated locally such that its position remained constant. If the extrinsic of the camera is denoted with (R,t), changing the rotation R while keeping the translation parameter t constant would also result in a changed camera position because the rotation R is applied before t. The distance of the camera to the scene was about one meter, and the width of the brick model is 22 cm. The x-axis of the depth camera points to the right, the y-axis to the top and the z-axis points toward the brick model.

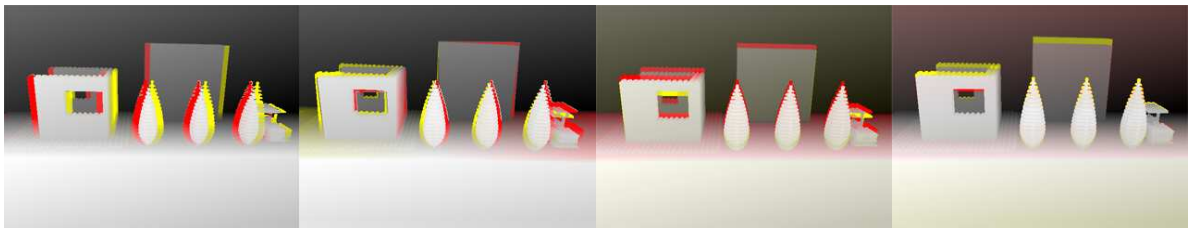
The evaluation shows that for this scene translational errors have rather moderate effects on the accuracy of the difference detection. This is mainly due to the fact that the scene contains flat walls. Therefore, the changes in the depth measurements are not very large when the camera is moved. However, the calculation accuracy depends strongly on an accurate estimation of the camera rotation. Small errors in the camera rotation cause large discrepancies between the reference and the evaluation depth image. In view of the intrinsic parameters, an inaccurate estimation of the principal point causes much larger errors than inaccuracies of the focal length.

6.2 Depth measurement noise

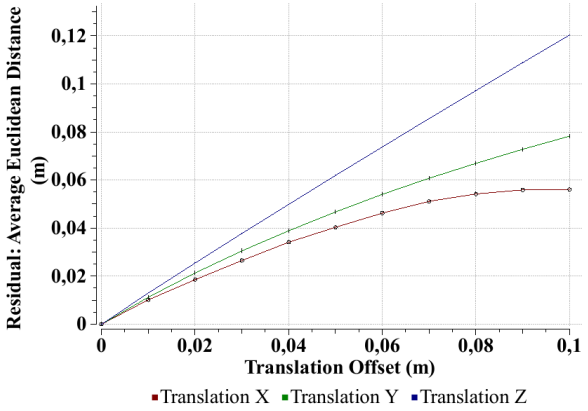
The depth measurements of depth cameras are affected by random noise as well as by systematic errors (e.g., due to reflections of specular surfaces). The systematic errors are different for each depth measuring technology and cannot be simulated easily. However, we evaluated the effects of measurement noise: Therefore, Gaussian noise was added to the depth values. In accordance with the specification of SwissRanger depth cameras, the standard deviation of this random noise was set to 1% of the distance to the camera (MesaImaging 2011). The right column visualizes the results of tests that were conducted with noisy depth images. If the pose of the depth camera is very accurately estimated, the accuracy of the calculation results is primarily influenced by the random measurement noise. However, when the pose error increases (especially the rotational error), the difference detection accuracy is primarily reduced by the errors that are caused by the camera pose estimation.

6.3 Inaccuracies of the 3D model

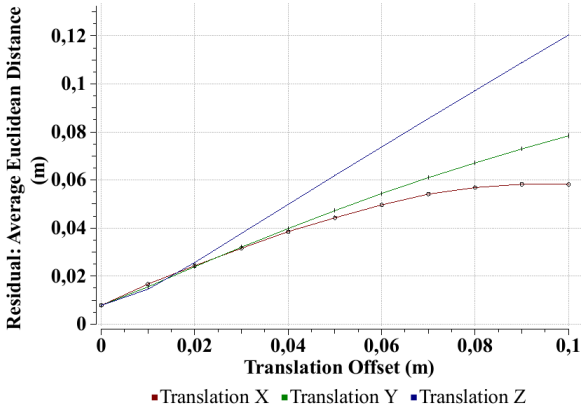
3D models often only approximate real objects. For example, polygonal meshes approximate curved objects with planar triangle surfaces. We used a standard VRML sphere with a radius of one meter to evaluate this effect. Figure 6 visualizes the polygonal sphere mesh as well as the differences that are detected when depth images of the sphere are taken from a distance of three meters to the sphere with rotational and translational offsets. Furthermore, Fig. 7 shows the residual plots of experiments in which the camera was positioned exactly at the center of the sphere. If the sphere model would be smoothly curved, the rotational offset would always be 0. However, the polygonal approximation of the sphere causes differences between the depth images. Adding noise to the depth images reveals a particular interesting effect: The residual is larger than with either the noise or the approximation-based differences alone, but not as much as the sum of both effects. In this case, the noise in the depth values smoothes the polygonal approximation of curved surfaces, thus reducing the error caused by the triangle approximation.



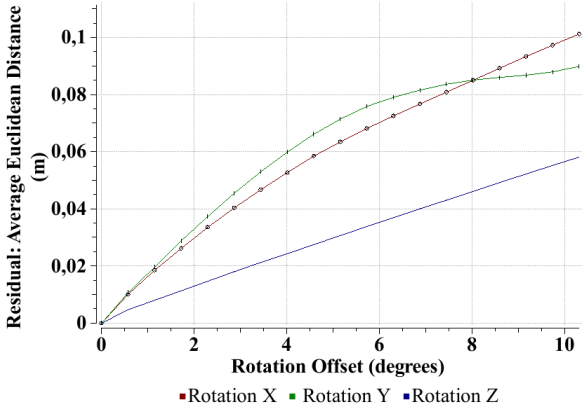
(a) From left to right: Inaccurate translation (0.02m), rotation (2.3deg), principal point (10 pixel), focal length (50 pixel).



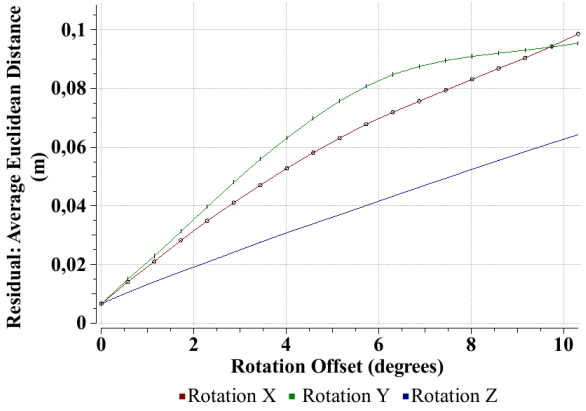
(b) Residual caused by camera translation (without noise)



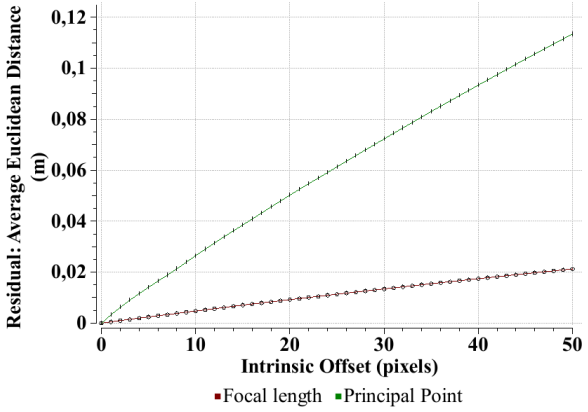
(c) Residual caused by camera translation (with noise)



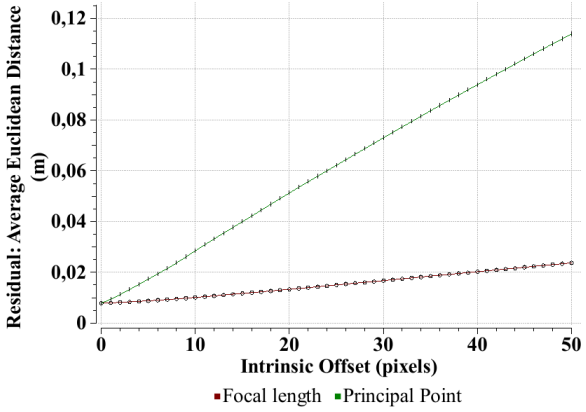
(d) Residual caused by camera rotation (without noise)



(e) Residual caused by camera translation (with noise)

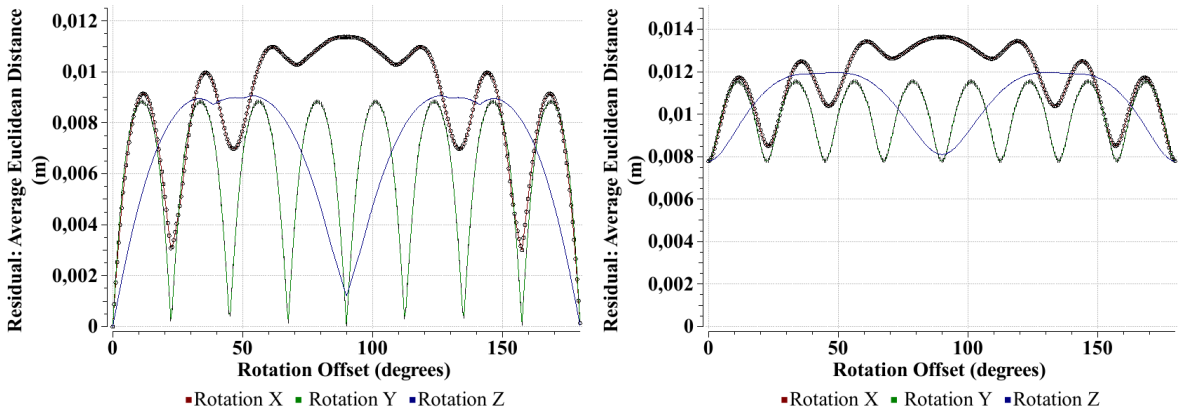


(f) Residual caused by incorrect intrinsic parameters (without noise)



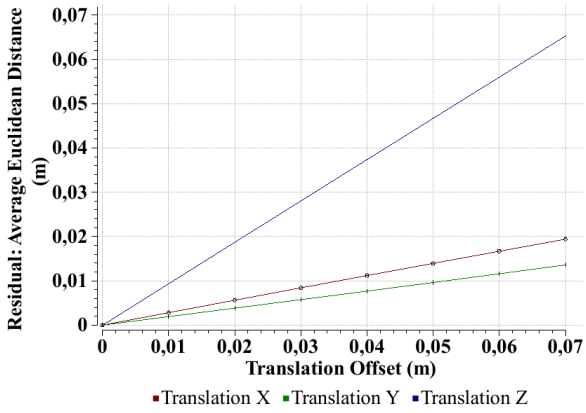
(g) Residual caused by incorrect intrinsic parameters (with noise)

Fig. 5 Influence of inaccurately estimated camera pose on the difference detection. ((b)-(e)): Inaccurate extrinsic parameters. (f)-(g)): Inaccurate intrinsic parameters. The residual is the average distance between the depth image and the 3D model.

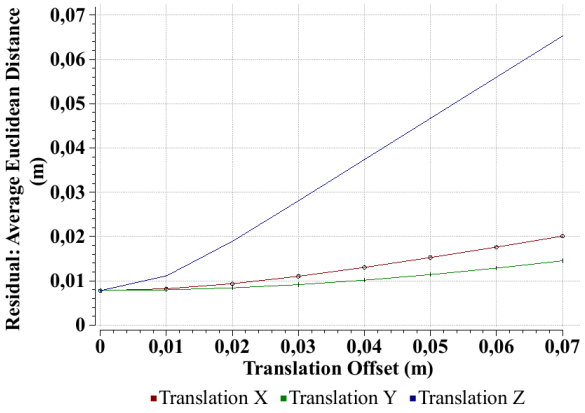


(a) Rotation: Without Noise

(b) Rotation: With Noise



(c) Translation: Without Noise



(d) Translation: With Noise

Fig. 7 Influence of approximated 3D meshes on difference detection.

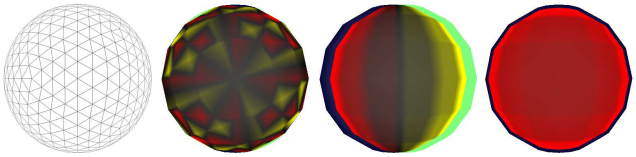


Fig. 6 From left to right: (a) Polygonal mesh of a 3D sphere (b) Difference visualization of rotational offset (c) Difference visualization of translational offset (sidewards) (c) Difference visualization of translational offset (distance)

6.4 Execution time

The execution time was measured with an Intel Core i7 processor with 3.07 Ghz and an NVidia GeForce GTX 470. The CPU-based part of the framework is implemented as a single core C++ implementation.

The geometric difference detection compares the real measurements with the 3D model (see Sects. 4.4 and 5.1). It is implemented as a GPU fragment shader. Table 1 shows the execution time of the geometric difference detection. If the 3D model consists of relatively few triangles, the execution time mainly results from the time it takes to copy the data to the graphics card and back to the CPU. This time is constant for a given image size. Even for large 3D models with more than 2 million triangles, the geometric difference detection is very fast. Thus, it can be used in real time in combination with online camera pose estimation. Please note that the time for the camera pose estimation (marker tracking or structure from motion) is not included in Table 1, as it depends on the structure and appearance of the tracked scene. The real-time structure from motion algorithm used in our framework was evaluated by Bleser et al. (2007).

Number of triangles	**Image size**		
	176×144	240×320	480×640
1.280	1ms	3ms	9ms
15.000	1ms	3ms	9ms
111.000	2ms	4ms	9ms
670.000	3ms	6ms	12ms
1.000.000	3ms	6ms	13ms
2.500.000	6ms	8ms	14ms

Table 1 Geometric difference detection: execution time (milliseconds)

The 3D model adjustment substep that adds triangles to the 3D model is implemented on the GPU. The processing time required to add a new triangle submesh depends only on the size of the new submesh (and thus the number of measurements in the depth image), not on the previous size of the 3D model. It is 9 ms for a depth image with 176×144 depth measurements, 22 ms for 320×240 depth measurements, and 80 ms for 640×480 depth measurements. Usually not the whole depth image, but only subparts of it need to be added to the 3D model. In this case, the processing time is reduced accordingly. The identification of the triangles that need to be removed from the 3D model is implemented as a fragment shader. The processing time of this shader corresponds approximately to the execution time of the difference detection shader, which is given in Table 1.

6.5 Model-based camera tracking

To evaluate the applicability of the presented approach for model-based camera tracking, a virtual 3D scene (for which ground truth data was known) was reconstructed with the presented approach. Then, the tracking accuracy of model-based camera tracking was evaluated for the reconstructed 3D model. The virtual 3D scene used for this evaluation corresponds to the real scene shown in Figs. 2, 3 and 4 and is visualized in Fig. 8. For this evaluation scenario, we assumed that initially a 3D model of the lego bricks and the sideboard was given. However, the box and the small car were not modeled yet. Thus, the user would take four snapshots of the box and the car, from which the surfaces of both objects were reconstructed (Gaussian noise was added to the simulated depth images). This 3D reconstruction step added 56.502 new triangles to the 3D model, which previously consisted of 201.670 triangles.

In a next step, the reconstructed 3D model was used for model-based camera tracking. The model-based pose estimation tracks KLT features in the 2D image and acquires the 3D coordinates of these features from the 3D model, see Bleser et al. (2005). The virtual camera was moved 360° around the object, which is a challenging camera path as it requires to continuously acquire new 2D-3D correspondences for the tracking. Figure 9(a) plots the Euclidean camera pose inaccuracy for the reconstructed 3D model and for a perfect 3D model of the scene. For the perfect 3D model, the average Euclidean distance of the calculated pose to the real camera pose was 3.3 cm, for the reconstructed 3D model 4.8 cm. To further evaluate the contribution of 3D model adjustment for model-based camera tracking, the box was moved 2 cm to the right. Figure 9(b) plots the camera pose error without and with a further 3D model adjustment step. Whereas an accuracy of 4.9 cm can be achieved with the 3D model which gets adjusted according to the replacement of the box, the average camera pose error without this adjustment is 16.5 cm. Figure 8 visualizes the estimated camera paths with and without the adjustment step.

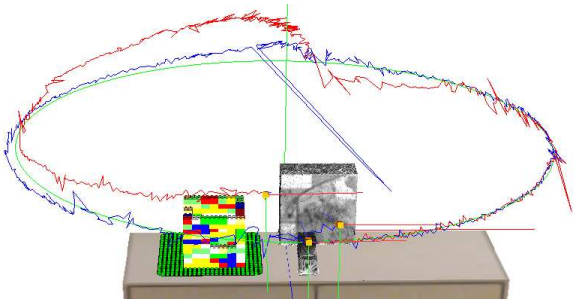


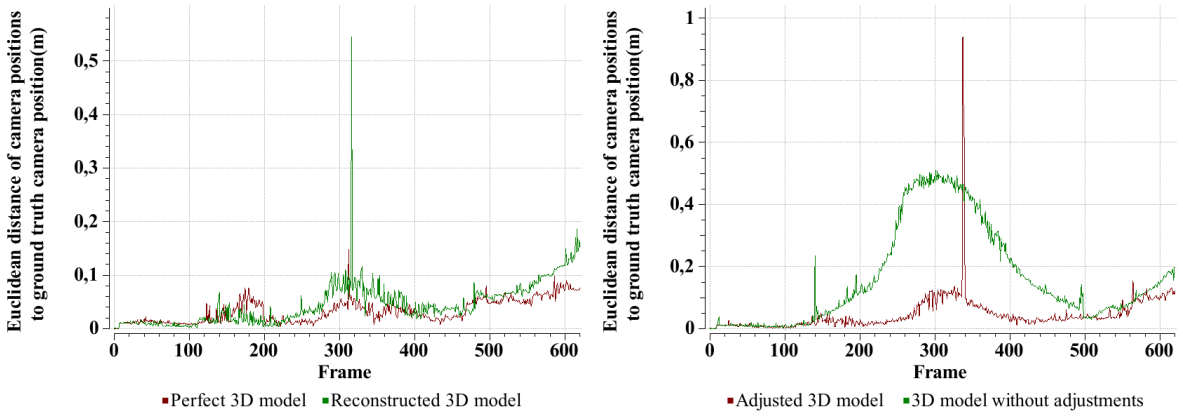
Fig. 8 Ground truth camera path (green) and tracked camera paths. Red: Without 3D model adjustment according to the changed box position. Blue: tracked camera pose after the 3D model adjustment step. Parts of the scene that were given as input to the reconstruction are colored, reconstructed parts are shown in gray.

7 Conclusion and future work

The framework presented in this paper reduces the gap between augmented reality and 3D modeling by supporting the 3D modeling task with augmented reality-based difference detection. The combination of real-time depth imaging with camera tracking is the basis for geometric difference detection between a real scene and a 3D model of this scene. In a model adjustment step, the detected differences can be used to adjust the 3D model such that it corresponds to the real scene. This is achieved by adding new submeshes to the 3D model or by removing triangles if geometric differences are detected.

In future work, the 3D model adjustment could be enhanced in several ways: Currently, the added triangle meshes have a uniform resolution. Thus, the number of triangles in the 3D model increases quite fast when new submeshes are added. The number of triangles can be significantly reduced by mesh decimation algorithms such as quadric edge collapse decimation (Garland and Heckbert 1997). Furthermore, the detection of geometric elements such as planar surfaces in the depth images could help to yield both smaller (in number of triangles) and more accurate surface structures. If a plane is detected, the 3D measurements can be adjusted such that they lie exactly on the plane and the geometry can be modeled by very few triangles.

The model adjustment could further be improved by edge stitching: In the current version, the triangle meshes are just added to the geometry the same way as they were measured. This causes saw-toothed edges that are not smoothly merged with the other triangles. This could be enhanced by adding additional triangles between the vertices at the border of the



(a) Tracked camera pose: perfect and reconstructed 3D model (b) Tracked camera pose: with and without 3D model adjustment

Fig. 9 Accuracy of camera pose estimation (model-based camera tracking).

new mesh and the closest vertices of the previously reconstructed triangle mesh. In view of the model adjustment itself, adjusting the position of triangles could be an alternative to the appending and removal of triangles. This would require to find a solution for the question: Under which conditions can we assume that an existing object has moved? And in which cases do we rather assume that a new object was added to or removed from the scene?

In the current implementation, the camera pose estimation process is separated from the geometric modeling. The registration between the depth image and the 3D model could be enhanced with an additional geometric alignment step such as the Iterative Closest Point algorithm (Besl and McKay 1992). Furthermore, a combined numerical optimization, which optimizes the poses as well as the 3D measurements, could help to find an optimum for the pose estimation and the geometric registration.

Acknowledgements This work was partially funded by the German BMBF project AVILUSplus (01IM08002).

References

Azuma RT (1997) A survey of augmented reality. Presence Teleoperators Virtual Environ 6:355–385

Bastian J, Ward B, Hill R, van den Hengel A, Dick A (2010) Interactive modelling for ar applications. In: 9th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp 199–205

Besl P, McKay N (1992) A method for registration of 3-D shapes. In: IEEE Trans Pattern Anal Match Intell, vol 14(2), pp 239–256

Bleser G (2009) Towards visual-inertial slam for mobile augmented reality. PhD thesis, TU Kaiserslautern

Bleser G, Pastamorv Y, Stricker D (2005) Real-time 3d camera tracking for industrial augmented reality applications. In: WSCG, pp 47–54

Bleser G, Becker M, Stricker D (2007) Real-time vision-based tracking and reconstruction. J Real Time Image Proc 2:161–175

Bosché F (2008) Automated recognition of 3d cad model objects in dense laser range point clouds. PhD thesis, University of Waterloo

Bosché F (2010) Automated recognition of 3D cad model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction. Elsevier J Adv Eng Inform 24(1):107–118

Bosché F, Teizer J, Haas CT, Caldas CH (2006) Integrating data from 3d cad and 3d cameras for real-time modeling. In: Proceedings of joint international conference on computing and decision making in civil and building engineering, pp 37–46

Engelhard N (2010) KinectAutoCalibration. https://github.com/NikolasE/KinectAutoCalibration

Engelhard N, Endres F, Hess J, Sturm J, Burgard W (2011) Real-time 3d visual slam with a hand-held rgb-d camera. In: Proceedings of the RGB-D workshop on 3D perception in robotics at the European robotics forum, Vasteras, Sweden

Franke T, Kahn S, Olbrich M, Jung Y (2011) Enhancing realism of mixed reality applications through real-time depth-imaging devices in x3d. In: Proceedings of the 16th international conference on 3D web technology, ACM, New York, NY, USA, Web3D '11, pp 71–79

Gall J, Rosenhahn B, Seidel HP (2006) Robust pose estimation with 3D textured models. In: Pacific-Rim symposium on image and video technology (PSIVT), pp 84–95

Garland M, Heckbert PS (1997) Surface simplification using quadric error metrics. In: Siggraph 1997, pp 209–216

Georgel P, Schroeder P, Benhimane S, Hinterstoisser S, Appel M, Navab N (2007) An industrial augmented reality solution for discrepancy check. In: ISMAR 2007: proceedings of the 6th IEEE and ACM international symposium on mixed and augmented reality, pp 1–4

Georgel P, Benhimane S, Sotke J, Navab N (2009a) Photo-based industrial augmented reality application using a single keyframe registration procedure. In: ISMAR 2009: Proceedings of the 8th IEEE and ACM international symposium on mixed and augmented reality, pp 187–188

Georgel P, Schroeder P, Navab N (2009b) Navigation tools for viewing augmented cad models. IEEE Comput Graph Appl 29(6):65–73

Haller M (2004) Photorealism or/and non-photorealism in augmented reality. In: Proceedings of the 2004 ACM SIGGRAPH international conference on virtual reality continuum and its applications in industry, VRCAI '04, pp 189–196

Van den Hengel A, Hill R, Ward B, Dick A (2009) In situ image-based modeling. In: Proceedings of the 2009 8th IEEE international symposium on mixed and augmented reality, ISMAR '09, pp 107–110

Huhle B, Jenke P, Straßer W (2008) On-the-fly scene acquisition with a handy multi-sensor system. Int J Intell Syst Technol Appl (IJISTA)

5(3/4):255–263
Kahn S, Wuest H, Fellner DW (2010a) Time-of-flight based scene reconstruction with a mesh processing tool for model based camera tracking. In: 5th international conference on computer vision theory and applications (VISAPP), vol 1, pp 302–309
Kahn S, Wuest H, Stricker D, Fellner DW (2010b) 3D discrepancy check and visualization via augmented reality. In: 9th IEEE international symposium on mixed and augmented reality (ISMAR), pp 241–242
Klein G, Murray D (2009) Parallel tracking and mapping on a camera phone. In: Proceedings of the eigth IEEE and ACM international symposium on mixed and augmented reality (ISMAR'09), Orlando, pp 83–86
Kolb A, Barth E, Koch R, Larsen R (2009) Time-of-flight sensors in computer graphics. In: Proceedings of the eurographics (state-of-the-art report), pp 119–134
Lepetit V, Fua P (2005) Monocular model-based 3D tracking of rigid objects: a survey. In: Foundations and trends in computer graphics and vision, vol 1, pp 1–89
Lindner M, Kolb A, Hartmann K (2007) Data-fusion of pmd-based distance-information and high-resolution rgb-images. In: Proceedings of the international symposium on signals, circuits and systems (ISSCS), session on algorithms for 3D TOF-cameras, vol 1, pp 121–124
MesaImaging (2011) Mesa imaging. Http://www.mesa-imaging.ch
Newcombe R, Davison A (2010) Live dense reconstruction with a single moving camera. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 1498–1505
Oggier T, Lustenberger F, Blanc N (2006) Miniature 3D ToF camera for real-time imaging. In: Perception and interactive technologies, pp 212–216
OpenNI (2011) Openni framework. Http://www.openni.org/
OpenSG (2011) OpenSG. http://www.opensg.org
Pan Q, Reitmayr G, Drummond T (2009) Proforma: probabilistic feature-based on-line rapid model acquisition. In: Proceedings of the 20th British machine vision conference (BMVC), p 11
Panagopoulos A, Samaras D, Paragios N (2009) Robust shadow and illumination estimation using a mixture model. In: IEEE conference on computer vision and pattern recognition (CVPR), pp 651 –658
Schiller I, Beder C, Koch R (2008) Calibration of a pmd-camera using a planar calibration pattern together with a multi-camera setup. In: The international archives of the photogrammetry, remote sensing and spatial information sciences, vol XXI. ISPRS Congress, pp 297–302
Shi J, Tomasi C (1994) Good features to track. In: IEEE conference on computer vision and pattern recognition (CVPR'94), pp 593–600
Webel S, Becker M, Stricker D, Wuest H (2007) Identifying differences between cad and physical mock-ups using ar. In: ISMAR 2007: Proceedings of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality, pp 281–282
Wuest H (2008) Efficient line and patch feature characterization and management for real-time camera tracking. PhD thesis, TU Darmstadt
Wuest H, Wientapper F, Stricker D (2007) Adaptable model-based tracking using analysis-by-synthesis techniques. In: Kropatsch W, Kampel M, Hanbury A (eds) Computer analysis of iImages and patterns, lecture notes in computer science, vol 4673, Springer, Berlin, pp 20–27
Zhou F, Duh HBL, Billinghurst M (2008) Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In: ISMAR 2008: IEEE / ACM international symposium on mixed and augmented reality, pp 193–202