Vis Comput (2018) 34:1507–1523
https://doi.org/10.1007/s00371-017-1425-2

ORIGINAL ARTICLE

Reconstructing three-dimensional models of objects using a Kinect sensor

Chin-Hung Teng[1,2] · **Kai-Yuan Chuo**[2] · **Chen-Yuan Hsieh**[2]

Published online: 9 August 2017

Abstract Advanced sensor technology has allowed us to acquire three-dimensional (3D) information from a scene using a low-cost RGB-D sensor such as Kinect. Although this sensor can recover the 3D structure of a scene, it cannot distinguish a target object from the background. In view of this, we incorporate an interactive 3D segmentation algorithm with a well-known Kinect scene reconstruction system, the KinectFusion, to effectively extract an object from the scene, and hence obtain a 3D point cloud of the object. With this system, a user can freely move the Kinect sensor to reconstruct the scene and then select the foreground/background seeds from the reconstructed point cloud. The system can take over the following tasks to complete the 3D reconstruction of the selected object. The advantage of this system is that users need not select the foreground/background seeds very carefully, which greatly reduces the operational complexity. Moreover, previous segmentation results are inherited to the next phase as new foreground/background seeds, which minimizes the required user intervention. With a simple seed selection, the point cloud of the selected object can be gradually recovered when a user moves the sensor to different viewpoints. Several experiments were conducted, and the results confirmed the effectiveness of the proposed system. The 3D structures of objects with complex shapes are well reconstructed by our system.

Keywords 3D reconstruction · Kinect sensor · RGB-D sensor · Lazy Snapping · KinectFusion

✉ Chin-Hung Teng
chteng@saturn.yzu.edu.tw

Kai-Yuan Chuo
s1026428@mail.yzu.edu.tw

Chen-Yuan Hsieh
s976424@mail.yzu.edu.tw

[1] Innovation Center for Big Data and Digital Convergence, Yuan Ze University, 135 Yuan-Tung Road, Chung-Li, Taiwan

[2] Department of Information Communication, Yuan Ze University, 135 Yuan-Tung Road, Chung-Li, Taiwan

1 Introduction

The popularity of the Kinect sensor has triggered many interesting applications that are not limited to skeleton extraction for games. Because it is a depth sensor, it can generate continuous depth images of a scene and is thus able to reconstruct the 3D structure of the scene based on the obtained depth information. In the field of 3D reconstruction, Kinect reconstruction is neither the most accurate nor the cheapest. Its accuracy cannot be compared with that of expensive laser-based approaches such as 3D laser scanners and range finders. Its cost is also higher than that of Structure from Motion (SfM) techniques. However, it provides a solution for amateur users who want to reconstruct a 3D model of an object with acceptable accuracy at an affordable price. Kinect can produce dense and more accurate 3D reconstruction than SfM with a much lower price than laser-based devices.

Although Kinect can be used to recover the 3D structure of a scene, directly employing the raw measurements of Kinect in reconstruction leads to poor results. The raw measurements are quite noisy, and a sophisticated approach is necessary to overcome this problem. For example, Song et al. [36] developed an approach to measure the confidence values of depth measurements from Kinect sensor. With the confidence value, the reconstruction accuracy may be further improved. Of the approaches for Kinect reconstruction, KinectFusion [20,27] provides excellent scene

reconstruction in real time. It solves the problem of noisy measurements by first applying a bilateral filter [37] to the depth images to remove measurement outliers. Subsequently, a truncated signed distanced function (TSDF) is maintained on a volumetric space to average or smooth the obtained depth information. This effectively reduces the influence of noisy depth information, leading to smooth and satisfactory surface reconstruction. Although KinectFusion can provide a good 3D reconstruction of a scene, it cannot distinguish a target object from the others present. To achieve this, a 3D point segmentation algorithm is also necessary.

Similar to image segmentation, automatic 3D scene segmentation or understanding is a very difficult issue. Instead of full automation, if our goal is to extract the 3D model of an object in a scene, then interactive segmentation provides a feasible approach to achieve this goal. With the help of user operations, we can select and delete the points in the background, preserving the points on the target objects for subsequent model creation. However, deleting points in 3D space is a tedious work, especially for complex backgrounds. This work is even more labor intensive compared with background removal in image segmentation since we need to choose suitable viewpoints to select 3D points in the foreground or background.

In view of this, we integrate a user-friendly interactive segmentation algorithm with a Kinect reconstruction technique to complete an object 3D model reconstruction system, aiming to maximally reduce the required user intervention in the model reconstruction. We employ KinectFusion for scene reconstruction because of its real-time and high-accuracy surface reconstruction. In 3D object segmentation, we employ Lazy Snapping [21], an graph-cut segmentation algorithm that was originally developed for images and that has been proved to work well in 2D image segmentation, to reduce the required user effort in separating the foreground and background.

Lazy Snapping is a graph-based approach, and its advantages lie in the easy selection of the foreground and background seeds. Users need to only roughly select the seeds of the foreground and background. The algorithm can complete the remaining segmentation tasks. It avoids the tedious work of interactive segmentation at the boundary of foreground/background, and hence its user operation is simple and friendly. The upper row of Fig. 1 illustrates this point. By simply drawing on the image to select the foreground (red line) and background (blue line) seeds, the foreground can be easily extracted. If the 3D point cloud can be segmented in a similar way, as shown in the bottom row of Fig. 1, the task of extracting target object is greatly simplified. Another advantage of employing this seed-based approach is that the segmentation results at the previous time instant can be inherited to the next time instant as the new seeds, thus further reducing the user intervention in seed selection. Users need only to perform seed selection once or a few times if the user continuously moves the Kinect sensor in space.

However, to apply Lazy Snapping we need to define nodes and edges in this graph-based algorithm. Two mechanisms were developed in our system: a point-based mechanism and a cell-based mechanism. In the point-based approach, each 3D point serves as a node in the graph. Edge definition is not straightforward, as we need to find adjacent nodes, which is not as easy in 3D space in comparison with the image segmentation case where the surrounding pixels are just the adjacent nodes. To find the adjacent nodes, we employ a Kd-tree search scheme on the reconstructed point cloud to improve the search efficiency. Moreover, since the point cloud in KinectFusion is generated with respect to the camera coordinate system, we need to align the point cloud generated from each time instant so that we can propagate the segmentation results from time to time. In our system, we solve this issue by applying an iterative closest point (ICP) algorithm [42] to align the current point cloud with the new point cloud generated by KinectFusion. Although the point-based approach requires more processing in edge definition and point cloud alignment, it can produce a more dense and complete point cloud for the target object.

In contrast with the point-based approach, the cell-based approach defines nodes based on the cells of the TSDF. Thus, edge definition in the cell-based approach is much simpler since the TSDF cells have explicit adjacent relations, similar to an image. Moreover, since each cell is defined with a globally defined world coordinate system, cell alignment is also unnecessary. Thus, the cell-based approach is more efficient than the point-based approach. However, because each cell has a volumetric space, its space resolution is smaller than that of the point-based approach. To produce a denser point cloud, a ray-casting approach needs to be applied to the cell-based segmentation results. The results of applying the two approaches on 3D object model reconstruction will be discussed and compared in our experimental section.

Another issue of combining KinectFusion with Lazy Snapping is the computational efficiency since Lazy Snapping cannot be used in real time. To preserve the real-time feature of KinectFusion, we implement Lazy Snapping in another thread so that KinectFusion can still operate smoothly. The segmentation results will be reflected to the main thread when Lazy Snapping finishes its task. Subsequently, the need seeds are transferred to Lazy Snapping to activate the next segmentation process. With the above implementations, our system operates as follows. First, a user moves the Kinect to acquire the 3D information of the scene. Next, the user selects the foreground and back-

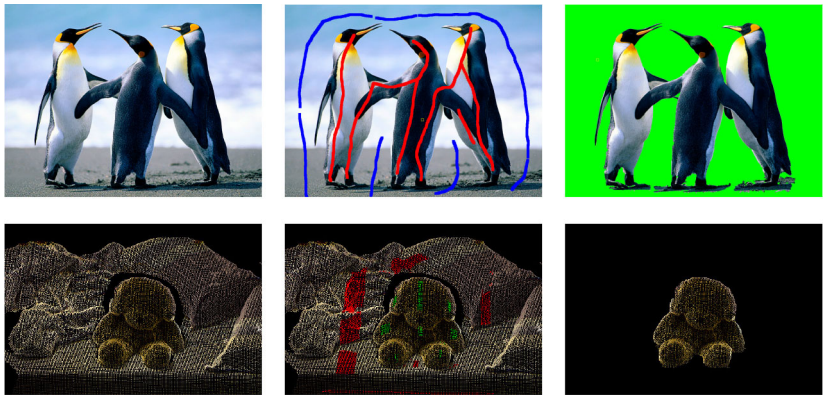


Fig. 1 Graph-cut image segmentation and 3D point cloud segmentation. From the *left* to the *right* are the input data, seed selection, and final segmentation

ground seeds from the scene and then the system extracts the 3D data of the object from the scene. Subsequently, the user moves the Kinect to acquire more data. Meanwhile, previously extracted 3D points serve as the new seeds for subsequent segmentation. Thus, after the first segmentation, the system automatically segments the object when the Kinect acquires more data. In this way, the model of an object is dynamically updated and we gradually obtain a complete model of the object.

To facilitate the operation of the proposed system, an Arduino module with BlueTooth XBee is also included in this system to replace traditional mouse operations. The Arduino module is attached to the Kinect sensor to allow users to move the sensor and simultaneously select the foreground and background seeds by clicking the buttons on the Arduino board. This further improves the usability of the proposed system since the users need not simultaneously operate the mouse and the Kinect.

In summary, the major contribution of this paper is that we incorporate the KinectFusion and Lazy Snapping to develop a real-time and interactive object 3D model reconstruction system. Two mechanisms for graph construction, the point- and cell-based approaches are devised, and their performance is evaluated. An Arduino module with BlueTooth XBee is also included in the system to facilitate the operation of the system. With these features, a user can hold the Kinect to quickly reconstruct the 3D model of a target object with little user operations. In the following sections, we will describe our system in detail.

The remainder of this paper is organized as follows. In the following, a related work is presented to further address the position of our system with respect to other researches. In Sect. 3, the framework of our system is introduced. Later, the key points of KinectFusion are described. An interactive segmentation algorithm is also discussed in this section. Section 4 presents the experimental results of applying our system in 3D object model reconstruction for several cases. Finally, we conclude in Sect. 5.

2 Related work

Creating 3D models of objects plays a very important role for a variety of 3D-based applications such as virtual reality (VR), augmented reality (AR), games, and animation. Approaches for creating 3D object models can be roughly categorized into two types: modeling using powerful 3D software such as 3Ds Max or Maya and reconstructing the object model using specially designed instruments such as 3D scanners.

Although 3Ds Max and Maya have the advantage of creating models that do not exist in the real world, they are typically very expensive and require specialized skills to operate them. They cannot be applied by non-expert users to create a simple model of an existing object for simple applications. From this viewpoint, the approach for reconstructing a model using an instrument is easier and more straightforward.

Generally, the approaches for sensing 3D information can be classified as active and passive. Active method indicates that we reconstruct the 3D structure of a scene by actively measuring the 3D information of an object. 3D laser scanners and time-of-flight techniques [19] are representatives of an active method where a laser beam or LED light is emitted and projected onto the object to acquire 3D data. Typically, 3D laser scanners can produce very accurate 3D measurements, but they are also very expensive devices. In comparison with active approaches, passive approaches achieve 3D reconstruction by the light reflected from an object.

Typical passive approaches include binocular stereo [5, 9,14], photometric stereo [8,18], structure from motion [29,34], and shape from contour [43] or shape from silhouette [10,33]. In general, passive approaches require more sophisticated algorithms to produce satisfactory reconstruction results. For example, Sahillioğlu and Yemez [33] develop a coarse-to-fine surface reconstruction system based on mesh deformation. The system can produce watertight surface models of complex objects from the silhouette of the object and range data. Dibra et al. [10] introduced a method for

human body shape estimation from the silhouette. This approach employs the very hot convolutional neural networks to achieve the task. Sometimes, the passive approach may require some manual operations to achieve better results and this leads to interactive modeling. For instance, VideoTrace [39] is an image-based interactive modeling approach. In this system, a user must go through the image sequence and draw some polygons on the image to indicate the faces of an object. Later, the 3D positions of points on the polygons are determined and the polygonal mesh representing the object is recovered. PhotoSketch [41] is another image-based interactive modeling system that can be used create the model of a building from urban scenes.

Recently, because of the popularity of Kinect sensors in the field of gaming, 3D reconstruction using a Kinect sensor has received much attention. Kinect is actually a structured light approach [1,15,30], which actively projects a specific pattern onto the environment and then obtains depth data by analyzing the deformation of the pattern in the captured image. There are two cameras in a Kinect sensor: an ordinary RGB camera and an infrared camera. The infrared camera can capture the image composed of a number of dots projected by the Kinect infrared projector. By comparing the captured infrared image with a benchmark, a depth image of the scene is generated and hence Kinect is an RGB-D sensor. Because of the low-cost and moderate accuracy of Kinect, some researchers tried to exploit it in various computer vision applications. Han et al. [16] gave a review about using Kinect in the fields of object tracking and recognition, human activity analysis, hand gesture analysis, and indoor 3D mapping and indicated the merits of using RGB-D sensor such as Kinect in these fields. In particular, this paper also addressed several works relevant to KinectFusion, including the comparison of KinectFusion with high-end ground truth generation techniques [25], the adaption of KinectFusion to more realistic situations such as large lighting variations [23,24], allowing camera roam free [31], and unbounded environment [40]. Recently, Kinect is also employed in the field of underwater 3D mapping to replace expensive LIDAR technology [2].

Kinect can also be applied in the SLAM applications [11–13,17]. Engelhard et al. [13] used a Kinect sensor to implement a real-time visual SLAM. This approach is developed on the basis of SURF features [3]. A number of SURF features are detected and matched in the image sequence, and a 3D feature map is created from these features. The feature map is later used in camera tracking via an ICP algorithm [42]. Finally, a complete 3D model of the scene is reconstructed.

Endres et al. [12] developed a SLAM system and evaluated the accuracy, robustness, and processing time for three feature detection techniques, namely SIFT [22], SURF, and ORB [32], and concluded that their system can robustly tackle difficult data in common indoor environments.

Henry et al. [17] utilized the depth and color information provided by a Kinect sensor to develop an RGB-D ICP algorithm. This algorithm uses SIFT for feature detection and matching and employs GPU to improve computational efficiency. The detected features are first filtered by a RANSAC algorithm to remove outliers. Later, an initial translation and rotation between the features in each frame are estimated and refined by the ICP algorithm to acquire more accurate camera tracking. Finally, a complete 3D model of the scene is constructed. Du et al. [11] later enhanced the algorithm by including a concept called visibility conflicts. This technique can tackle some visual errors so as to improve the stability of the RGB-D ICP algorithm. Moreover, Du et al.'s system has the functionality of self-checking, which can remind users which parts of the reconstructed 3D data are incomplete or have errors, allowing the users to enhance data acquisition for those parts.

Unlike previous approaches that use color and depth data for 3D reconstruction, Newcombe et al. [27] developed a tool called KinectFusion, which employs only the depth data for 3D scene reconstruction. The reason for using only the depth data is that image color is easily influenced by ambient light and is therefore less robust than depth data. In KinectFusion, the space is partitioned into a number of cells and a truncated signed distance function (TSDF) is assigned for each cell. The TSDF records the distance from the surface of an object to the cell; thus, the surface of the scene can be estimated by the zero level set of the function. KinectFusion employs the GPU to process the depth data in parallel and can achieve real-time 3D reconstruction of the scene.

Song et al. [35] also developed a system for scene surface reconstruction using a Kinect sensor. However, this system is quite similar to KinectFusion. Both employ volumetric representation and TSDF for surface reconstruction.

Previous approaches can all be used to reconstruct the 3D model of a scene. However, they cannot distinguish an object in the scene from other objects. That is, if we want to reconstruct the 3D model of an object in the scene, we need to segment the reconstructed scene to extract the desired object. Izadi et al. [20] provided a simple approach to extract the 3D model of a specific object in a scene on the basis of KinectFusion. Their idea is based on detecting the difference in the reconstructed 3D scene model at different times. Hence, if we want to acquire the 3D model of an object, we can remove the object after the scene has been reconstructed. By comparing the difference between the scene before and after removing the object, the piece of 3D information for this object can be extracted.

The above approach is straightforward, but it cannot be applied to stationary objects or if we want to reconstruct only a portion of an object. Chen et al. [6] proposed a simple

mechanism to extract an object from a reconstructed surface based on a Kinect sensor. They employed a maximum and a minimum threshold distance to separate the object from the background. When the distance between the reconstructed point and the camera exceeds the threshold range, the point is automatically cut to preserve the target object. Because the depth image from Kinect is noisy, the reconstructed point cloud may contain some holes, which leads to large triangles when converting to a polygonal mesh model. Moreover, the recovered polygonal mesh model may have sharp features and an unsmooth surface. Chen et al. applied a bicubic spline function to smoothen the model. Actually, point cloud segmentation is not the major concern of Chen et al.'s research. Their system may require some trial and error to adjust the thresholds so that the target object can be well extracted. It also needs to very carefully set up the viewpoints to remove the background points especially at vicinity where the foreground and the background surfaces meet. Moreover, this approach may not work well if an object has concave shape. Figure 2 shows an example of point cloud extraction using the min/max thresholds. Obviously, if the thresholds or the viewpoints are not well set, the extracted point cloud may have holes or the background points may not be well removed such as the points in the table in Fig. 2.

Another approach that is suitable for automatic indoor scene 3D reconstruction and understanding was proposed by Chen et al. [7]. This system was developed based on automatic semantic modeling, where the 3D scene is first reconstructed using a Kinect sensor and then the system extracts objects in the scene by comparing them with models in a database. The spatial relationship between each object is calculated and is exploited to create a more accurate scene structure. Finally, a semantic 3D model is recovered for the scene. However, because model creation is achieved by comparing the reconstructed surface with existing models, the system is not suitable for objects that are not in the database. Moreover, the extracted model is the most similar object in the database, not the physical object itself. Morana [26] devised another system for 3D scene reconstruction using Kinect. Unlike Chen et al.'s approach, Morana employed superquadrics to model the objects and hence model matching is replaced by parameter optimization of superquadrics. Because of the feature of superquadrics composition, this system is not applicable to objects with irregular shapes.

The software Skanect allows users to reconstruct a model using a Kinect sensor. However, this software requires users to manually cut the undesired part and thus is more suitable for objects with clean backgrounds. For complex backgrounds, users need to carefully change the viewpoints and select the surface that is not part of the target object, and hence the operation is sometimes tedious. Compared with the previous approaches, our system can create the model of a target object with acceptable accuracy and fewer user interventions and is therefore more user-friendly. SemanticPaint [38] is an amazing interactive 3D scene reconstruction and understanding technique. By combining different techniques, including KinectFusion, online learning, speech recognition, and mean field inference, this system can distinguish different objects from the reconstructed scene in real time. Actually, because this system is developed based on an online decision forest training, it may lead to false labeling for objects with similar shapes. Hence, it sometimes requires interactive corrections to amend the false labeling. In addition, the system requires a user to reach out and touch the object to initialize the labeling of a new object; thus, it may not be friendly for remote or untouchable objects. In contrast to interactive systems, Pan and Taubin [28] developed a fully automatic system for point cloud segmentation. However, the point cloud is from multi-view reconstruction and the system assumed that the target object is at the center of the input images. In addition, the system cannot be run in real time. Compared with previous approaches, our approach is a real-time system and because we focus on target object extraction, we do not have the issue of false labeling. Moreover, because we select foreground/background seeds from the screen, we do not need to physically touch the target object. We summarize the advantage/disadvantage of previous approaches in Table 1 for comparison.

3 Proposed system

The system diagram of our system is depicted in Fig. 3. The entire process is divided into two parts. The first part is scene reconstruction. Here, the 3D information of the scene, including the object we want to reconstruct, is constructed. This part is mainly based on KinectFusion [27].

The second part of our system is object extraction. Here, we apply an interactive segmentation algorithm, Lazy Snapping [21], to extract the 3D data of the desired object. Note that the KinectFusion is run in the GPU and therefore can achieve real-time 3D scene reconstruction. On the other hand, Lazy Snapping requires some user operations to select the foreground and background seeds. In addition, it cannot operate in real time. To avoid destroying the real-time characteristic of KinectFusion, the entire process of object extraction is run in a different thread and the results are updated each time a segmentation is completed.

As discussed in Sect. 1, we have two schemes for selecting the nodes for Lazy Snapping segmentation: the point- and the cell-based approaches. For the cell-based approach, the 3D positions of the TSDF cells are directly used to construct the graph, while for the point-based approach, we employ the reconstructed 3D points that are generated by a predicting surface process as the nodes of segmentation. This point is

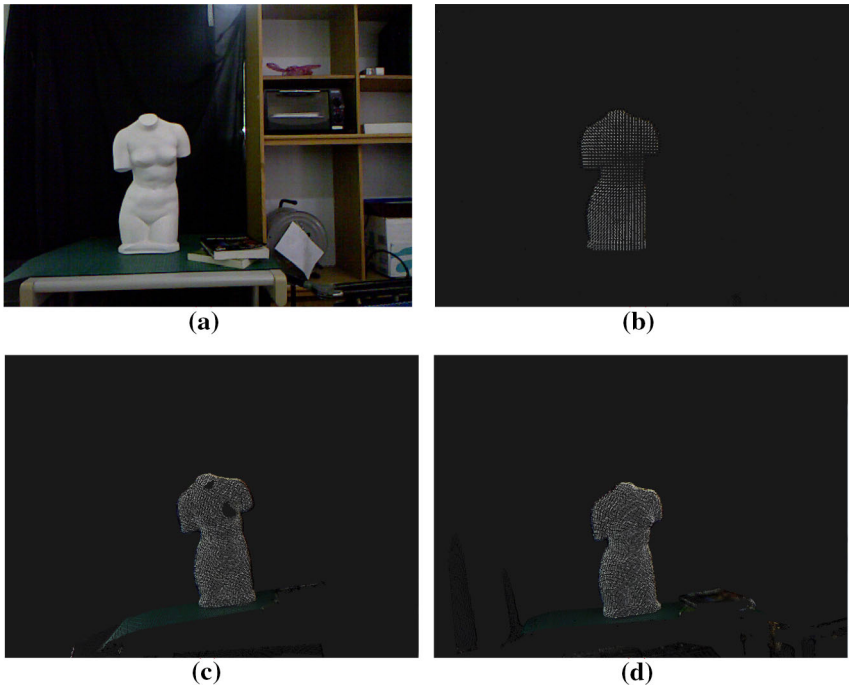


Fig. 2 Point cloud segmentation using our system and the approach of min/max thresholds for two different conditions. **a** One of the input RGB images. **b** Extracted point cloud (the statue) using our system. **c**, **d** The extracted point cloud using min/max thresholds. The point cloud may contain holes or background points if the viewpoints and the thresholds are not carefully chosen

Table 1 A summary of related point cloud segmentation techniques

Techniques	Advantages	Disadvantages
Chen et al. [6]	Real time	Need to carefully adjust viewpoints and the min/max thresholds to extract the foreground
Chen et al. [7]	Automatic semantic modeling	Need to match the objects with those in a database
Morana [26]	Automatic modeling	Only suitable for the objects with superquadric shapes
Pan and Taubin [28]	Automatic segmentation	Offline process, objects should be at the center of images
SemanticPaint [38]	Real time, user-friendly for room-scale 3D scene reconstruction and labeling	Need to touch the objects, sometimes the objects with similar appearance may be falsely labeled
Proposed system	Real time, user-friendly for target object 3D reconstruction	Only suitable for small-scale objects

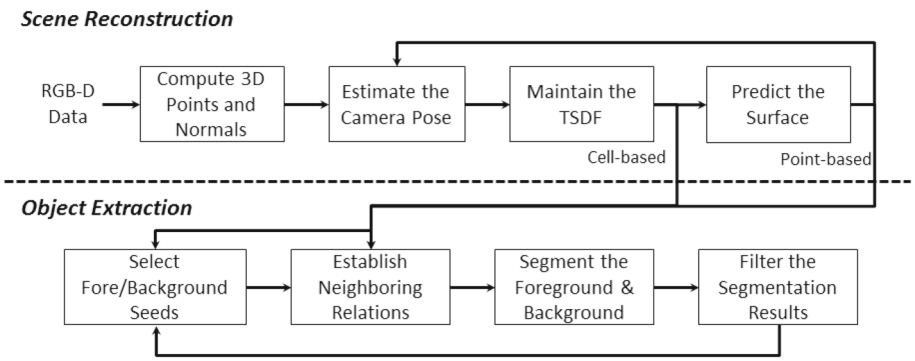


Fig. 3 System diagram of the proposed system

addressed in Fig. 3, which shows one branch from the module of ''Maintain the TSDF" and another branch from the "Predict the Surface". Although we plot the two branches in the diagram simultaneously, one of them is sufficient to complete the segmentation task, depending on the method used.

Although our system requires user interaction in selecting the foreground/ background seeds, this process need not be activated each time we perform the segmentation. Previously obtained segmentation results can serve as the new foreground and background seeds to continuously update the segmentation results. This point is illustrated by the feed-

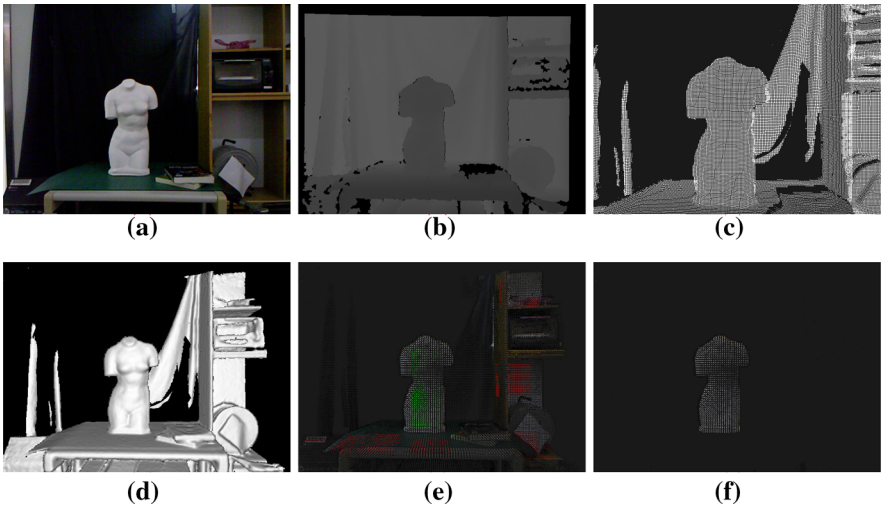


Fig. 4 Some intermediate results of proposed system. **a** The input *color* image. **b** The corresponding depth image. **c** The computed 3D points from the depth image. **d** Surface predicted by KinectFusion. **e** Foreground (*green points*) and background (*red points*) seed selection. **f** Final point cloud segmentation

back path from the final module of object extraction to the first module of seed selection in Fig. 3. With this mechanism, the points on the object can be gradually reconstructed when we move the sensor around the object after the initial seed selection.

To further understand the overall process, we have shown some intermediate results in Fig. 4. Figure 4a, b shows one frame of input RGB and depth image. Figure 4c is the computed point cloud from the current depth image. The predicted surface of KinectFusion is displayed in Fig. 4d. Figure 4e, f is the results of seed selection and final point cloud segmentation. In the following, the procedures of scene reconstruction and object extraction of our system are discussed in detail.

3.1 Scene reconstruction

Our scene reconstruction is finished by KinectFusion. The basic idea of KinectFusion is to maintain a TSDF on a volumetric space. To obtain the TSDF, the space is partitioned into a number of cells with each cell recording a TSDF value. This is illustrated in Fig. 5, where the 3D space is simplified to a planar diagram for concept illustration. The surface of an object is located at the zero level set of the TSDF. To construct and maintain the TSDF, several steps as depicted in Fig. 3 are employed to process the depth data generated by Kinect. These steps are described in the following subsections.

3.1.1 Computing 3D points and normals

When the Kinect sensor generates the depth data for a scene, the data are processed to update the TSDF values. The first step to achieve this is to transform the depth data into a number of 3D points and the associated normals. These 3D points and normals can be used to estimate the camera pose of the current frame.

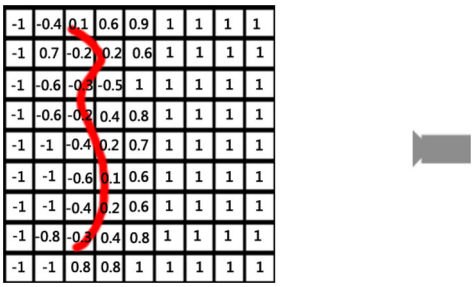


Fig. 5 A 2D illustration of the TSDF. The *red curve represents* the position of a surface

Let $R_k(\mathbf{u})$ denote the depth data generated by Kinect at time k. $R_k(\mathbf{u})$ is actually a depth image where $\mathbf{u} = (u, v)$ denotes the image coordinates. In general, the depth data $R_k(\mathbf{u})$ are noisy. In KinectFusion, the data are processed by a bilateral filter [37] to obtain a clean depth data $D_k(\mathbf{u})$ as follows:

$$D_k(\mathbf{u}) = \frac{1}{W_p} \sum_{\mathbf{q} \in U} N_{\sigma_s}(\|\mathbf{u} - \mathbf{q}\|_2) \cdot N_{\sigma_r}(\|R_k(\mathbf{u}) - R_k(\mathbf{q})\|_2) R_k(\mathbf{q}) \quad (1)$$

where U represents the set of all image coordinates, $N_\sigma(t) = \exp(-t^2\sigma^{-2})$, $\|\mathbf{x}\|_2$ indicates the L2-norm of $\mathbf{x}$, and W_p is a normalization term. A bilateral filter is similar to a Gaussian filter, but the data with large deviations are weighted by another Gaussian function to remove the influence of the outliers.

With $D_k(\mathbf{u})$, the 3D interpretation of each pixel in $D_k(\mathbf{u})$ can be obtained by

$$\mathbf{V}_k(\mathbf{u}) = D_k(\mathbf{u})\mathbf{K}^{-1}\dot{\mathbf{u}} \quad (2)$$

where $\dot{\mathbf{u}}$ is the homogeneous coordinate representation of $\mathbf{u}$ and $\mathbf{K}$ is the camera calibration matrix of Kinect. The idea of

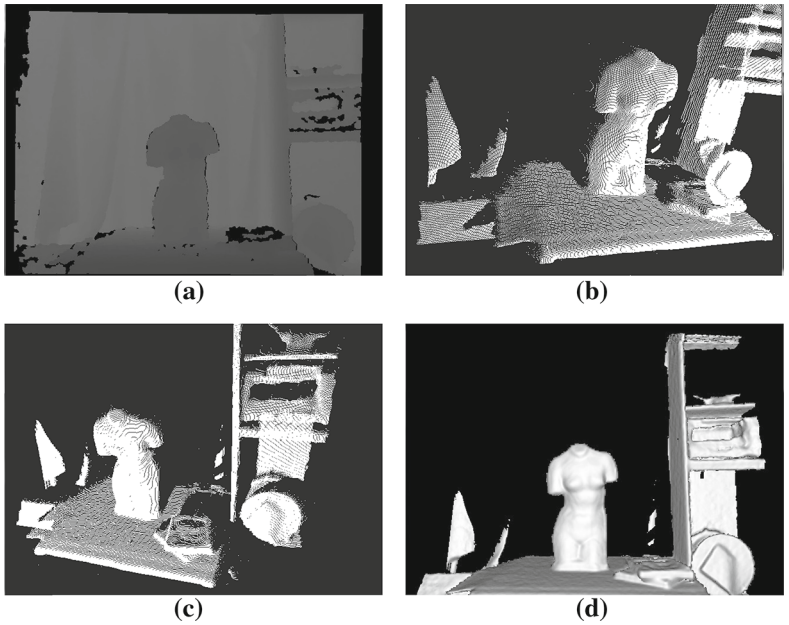


Fig. 6 An example of point cloud generated by Kinect. **a** The input depth image. **b**, **c** The point cloud generated by Kinect from different viewpoints. **d** The surface generated by KinectFusion

(2) is that we can employ the inverse matrix of $\mathbf{K}$ to transform a point from image coordinates to camera coordinates and the point becomes a 3D point in the camera coordinate system after multiplying the depth of this point. Because the depth data are actually images, we can immediately find the normal associated with each 3D point by the formula

$$N_k(\mathbf{u}) = Nor\left[(\mathbf{V}_k(u+1,v) - \mathbf{V}_k(u,v)) \times (\mathbf{V}_k(u,v+1) - \mathbf{V}_k(u,v))\right] \tag{3}$$

where $Nor(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|_2$, which represents vector normalization. The 3D points and the normals are used in the subsequent step for estimating the camera pose. Figure 6 shows an example of computing point cloud from a depth image. The generated point cloud will be used in the subsequent module for camera pose estimation.

3.1.2 Estimating the camera pose

Camera pose estimation is a critical step in KinectFusion as we need this information to update the TSDF values from the current depth data. In KinectFusion, camera tracking is achieved by exploiting the camera pose of the previous frame. First, a set of 3D points is estimated from the TSDF using the camera pose of the previous frame. The process for generating 3D points from the TSDF is called surface prediction. KinectFusion assumes that the camera is moved slowly; thus, the camera poses for two adjacent frames will not differ too much. Hence, by utilizing an ICP algorithm on the predicted 3D points and the measured 3D points, the camera pose of the current frame can be estimated.

3.1.3 Maintaining the TSDF

When the camera pose of the current frame is estimated, the TSDF values can be updated using the current depth data. For each cell $\mathbf{p}$ in space, the TSDF value is computed by the following equations:

$$F_{R_k}(\mathbf{p}) = \Psi(\lambda^{-1}\|\mathbf{t}_{g,k} - \mathbf{p}\|_2 - R_k(\mathbf{x})) \tag{4}$$

$$\lambda = \|\mathbf{K}^{-1}\dot{\mathbf{x}}\|_2 \tag{5}$$

$$\mathbf{x} = \left\lfloor \pi(\mathbf{K}\mathbf{T}_{g,k}^{-1}\mathbf{p}) \right\rfloor \tag{6}$$

$$\Psi(\eta) = \begin{cases} \min(1, \eta/\mu)\mathrm{sgn}(\eta), & \text{if } \eta \geq -\mu \\ \text{null}, & \text{otherwise} \end{cases} \tag{7}$$

where $\mathbf{T}_{g,k}$ represents a transformation between the camera coordinate system of Kinect at time k to the world coordinate system, i.e., if $\mathbf{p}_c$ is a point in the camera coordinate system, it can be transformed to the world coordinate system by $\mathbf{p}_w = \mathbf{T}_{g,k}\mathbf{p}_c$. $\mathbf{T}_{g,k}$ is a 4×4 matrix with the following form:

$$\mathbf{T}_{g,k} = \begin{bmatrix} \mathbf{R}_{g,k} & \mathbf{t}_{g,k} \\ \mathbf{0} & 1 \end{bmatrix} \tag{8}$$

where $\mathbf{R}_{g,k}$ and $\mathbf{t}_{g,k}$ are the estimated camera rotation and translation in world coordinate system, i.e., the camera pose of Kinect at time k.

In (6), π represents the process of image projection and the notation $\lfloor\ \rfloor$ indicates the selection of the nearest integer point. Equation (6) indicates that for any point $\mathbf{p}$ in space, we first transform it from the world coordinate system to the camera coordinate system of the current frame by using the estimated camera pose and then project it onto the image

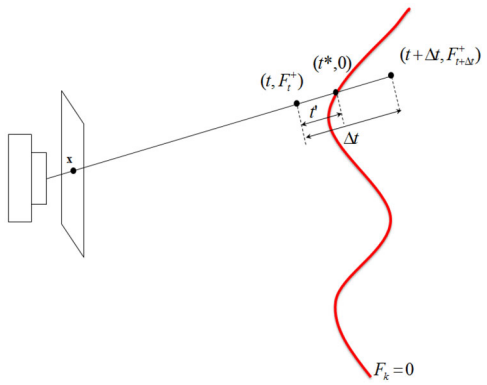


Fig. 7 Predicting the parameter $t = t^*$ of a point on a surface (*red line*)

plane to obtain $\mathbf{x}$. Hence, $\mathbf{x}$ is actually the imaged point of $\mathbf{p}$ by the Kinect at time k. Substituting $\mathbf{x}$ into the depth image R_k, we can obtain the depth value at the direction to $\mathbf{p}$. Finally, in (4), $\|\mathbf{t}_{g,k} - \mathbf{p}\|_2$ indicates the distance of $\mathbf{p}$ to the center of the camera and hence $\lambda^{-1}\|\mathbf{t}_{g,k} - \mathbf{p}\|_2$ is just the depth distance of $\mathbf{p}$ to the camera. By subtracting $\lambda^{-1}\|\mathbf{t}_{g,k} - \mathbf{p}\|_2$ and $R_k(\mathbf{x})$, and putting the result into a truncated function $\Psi(\eta)$ in (7), the TSDF value at a point $\mathbf{p}$ for the depth data $R_k(\mathbf{u})$ is obtained.

$F_{R_k}(\mathbf{p})$ in (4) is just the TSDF values estimated from the depth data at time k. The data are then used to fuse with the previous TSDF to obtain a new TSDF at time k. The update is achieved by linearly combining the previously accumulating TSDF with the current TSDF, as follows:

$$F_k(\mathbf{p}) = \frac{W_{k-1}(\mathbf{p})F_{k-1}(\mathbf{p}) + W_{R_k}(\mathbf{p})F_{R_k}(\mathbf{p})}{W_{k-1}(\mathbf{p}) + W_{R_k}(\mathbf{p})} \tag{9}$$

$$W_k(\mathbf{p}) = W_{k-1}(\mathbf{p}) + W_{R_k}(\mathbf{p}) \tag{10}$$

where $W_{R_k}(\mathbf{p}) \propto \cos\theta / R_k(\mathbf{x})$ and θ is the angle between the surface normal and the ray from the camera center to $\mathbf{p}$. This weight indicates that if the surface faces the camera, the depth value is more reliable.

3.1.4 Predicting the surface

The surface of the scene can be estimated from the TSDF if the TSDF is correctly maintained and updated. The predicted surface can be used in camera pose estimation for new frames and in subsequent object segmentation and extraction. The procedure for estimating the surface from the TSDF is illustrated in Fig. 7. The basic idea for surface prediction is ray casting. Given the camera pose of a frame, for each pixel in the image, a ray is emitted from the center of the camera passing through the pixel and then the surface is determined by finding the zero level set of the TSDF. Specifically, we can step along the ray until a zero crossing is detected. As shown in Fig. 7, the parameter t^* for $F_k = 0$ is determined by the following equation:

$$\frac{t'}{\Delta t} = \frac{0 - F_t^+}{F_{t+\Delta t}^+ - F_t^+} \Rightarrow t^* = t + t' = t - \frac{\Delta t F_t^+}{F_{t+\Delta t}^+ - F_t^+} \tag{11}$$

where F_t^+ and $F_{t+\Delta t}^+$ denote the TSDF values at t and $t + \Delta t$. These values can be estimated by linear interpolation of the TSDF at the grid points. After estimating t^*, we then use the equation $t^*\mathbf{T}_{g,k}\mathbf{K}^{-1}\dot{\mathbf{x}}$ to obtain a 3D point on the surface. In this way, an accurate 3D point cloud for the surface in the scene can be reconstructed. An example of predicted surface from TSDF values is illustrated in Fig. 4d.

3.2 Object extraction

After obtaining a point cloud of the scene, we then extract the points on the object whose 3D model we want to create. Our point segmentation algorithm is based on Lazy Snapping, which is an interactive segmentation algorithm. Some foreground and background seeds are first selected, and then each 3D point is treated as a node of a graph. Subsequently, the edges of the graph are defined by linking each 3D point with its neighboring points. Finally, the graph is cut to separate the foreground and background nodes. The procedures for selecting foreground/background seeds, establishing neighboring relations between points, and segmenting foreground/background points are described in the following subsections.

3.2.1 Selecting foreground and background seeds

Lazy Snapping is primarily designed for image segmentation; thus, the selection of the foreground and background seeds in the original Lazy Snapping is straightforward. A user can select the seeds by drawing them on the image directly. However, in 3D space, the process of selecting foreground and background seeds is more difficult. In our system, we achieve this by selecting a pair of 3D points in space. Then, the points inside the cuboid defined by this pair of points are selected as our foreground or background seeds. Although this approach is not as straightforward as the image case, we can still obtain satisfactory results if an appropriate viewpoint is chosen and multiple pairs of points are manually selected. Figure 4e shows an example of our foreground/background seed selection.

3.2.2 Establishing neighboring relations

Our system provides two approaches for defining the nodes in a graph. For the cell-based approach, the center point of

each cell is defined as a node of the graph and the neighboring nodes and the edges between them can be directly obtained from the structure of the TSDF cells, just as in the 2D image case. For the point-based approach, the task is somewhat more complex as the reconstructed 3D points may be scattered in the 3D space. Searching the nearest points using a brute force approach is very time-consuming, as we typically have tens of thousands of points; thus, we need a more efficient method to find neighboring nodes. In our system, we employ a Kd-tree to improve the searching efficiency for finding the nearest points. We use the 6 nearest points to define the edges of a node, and the weight for each edge is defined in the next subsection.

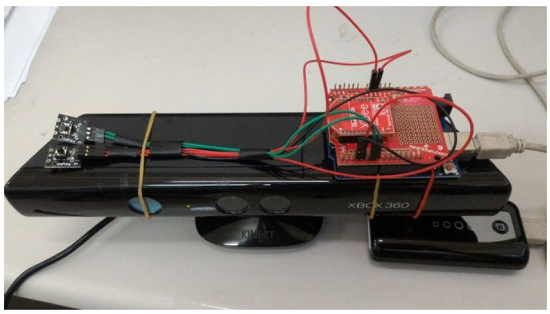


Fig. 8 Our Kinect with the Arduino and BlueTooth XBee module

3.2.3 Segmenting the foreground and background

Our 3D point segmentation is equivalent to minimizing the following cost function:

$$E = \sum_{i \in V} E_1(x_i) + \lambda \sum_{(i,j) \in \Lambda} E_2(x_i, x_j) \tag{12}$$

where V and Λ represent the set of nodes and edges in the graph, respectively. The symbol λ is a parameter controlling the relative significance of the two summation terms, and x_i denotes the segmentation result of the ith node. $x_i = 1$

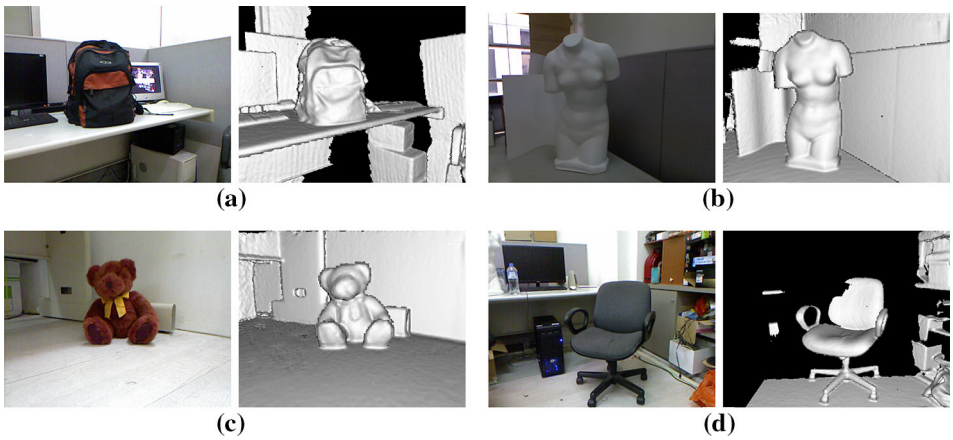


Fig. 9 Examples of reconstructed surfaces using KinectFusion

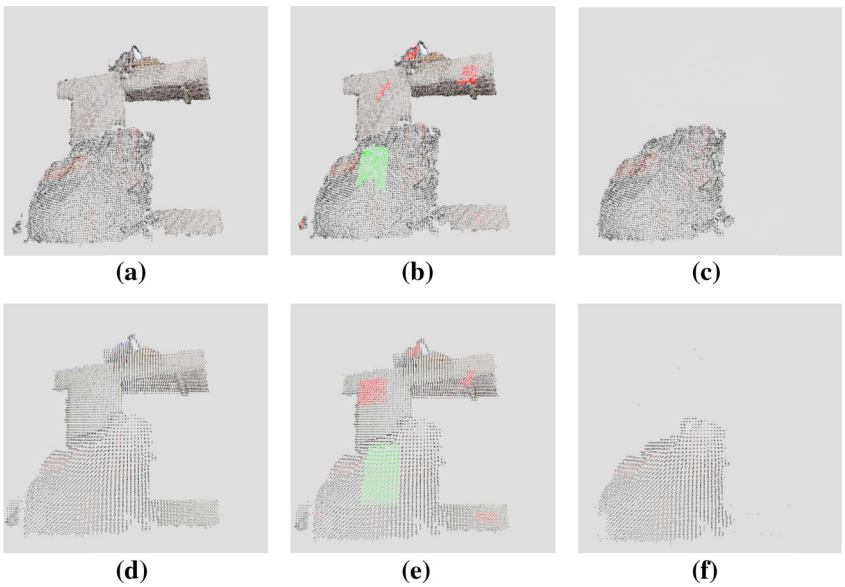


Fig. 10 Example reconstruction of a backpack. The *upper row* **a**–**c**) shows the results of the point-based approach and the *bottom row* **d**–**f** displays the results of the cell-based approach. **a** and **d** are the reconstructed point clouds; **b** and **e** show the selected foreground (*green*)/background (*red*) seeds; **c** and **f** are the final segmentation results

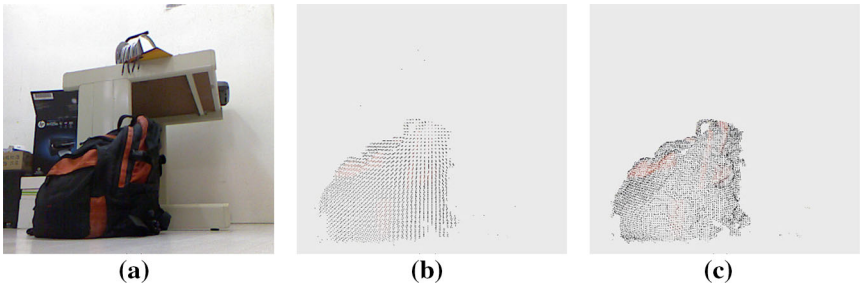


Fig. 11 **a** One of the backpack images. **b** Reconstructed points for the backpack using the cell-based approach. **c** Reconstructed points using the ray-casting algorithm from the results of the cell-based segmentation

Table 2 Comparison of the point- and cell-based approaches for the reconstruction of a backpack as an example

	Number of points	
	Point based	Cell based
Total points	20909	12155
Foreground seeds	714	700
Background seeds	798	803
	Processing time (s)	
	Point based	Cell based
Search neighboring points	0.078	0.015
Segment foreground/background	0.156	0.125

indicates that the point is a foreground point, while $x_i = 0$ indicates that the point is a background point.

The function $E_1(x_i)$ is the cost for the ith point to be classified as a foreground or background point. Its value is given by the color difference between the node and the foreground/background seeds [21]. The function $E_2(x_i, x_j)$ measures the cost of adjacent nodes, i.e., the weight of an edge. Its value is defined by the color difference between adjacent nodes. After the construction of the graph and setting of the weights of the edges, the graph is cut by a min-cut/max-flow algorithm [4] to obtain an optimal segmentation.

3.2.4 Filtering the Segmentation Results

Sometimes, after segmentation some isolated clusters of points may be generated. These points can be easily filtered out by checking their distance to the majority of the foreground points. If a cluster of points is far from the majority of the foreground points, these points are filtered out. After the filtering process, the points on the foreground object are well extracted and can be used to serve as the seeds of the next segmentation.

4 Experimental results

To confirm the feasibility of the proposed system, we have tried to reconstruct 3D models of several objects using our system. Our system is run on a desktop computer with an Intel Core i7-2600 CPU, 4GB memory, and a Nvidia GTX 560 graphics card. The graphics card is necessary since Kinect-

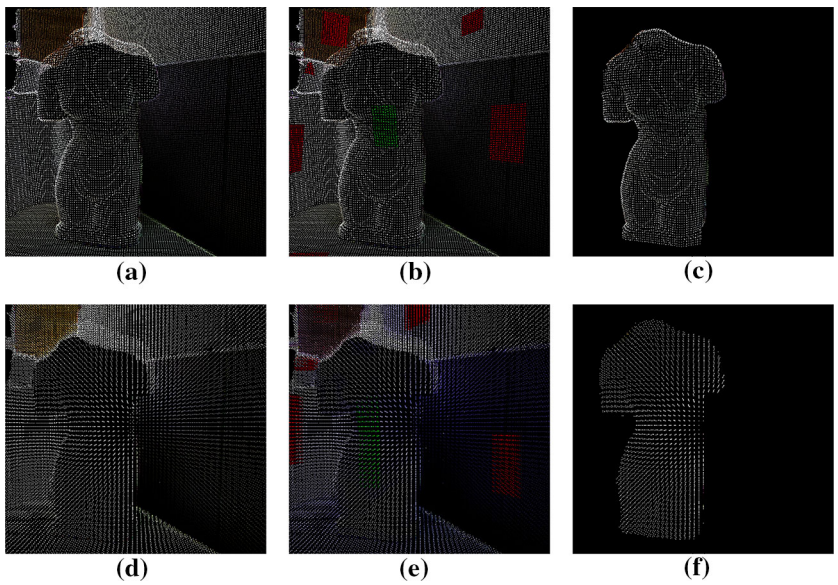


Fig. 12 Example reconstruction of a statue. The upper row **a**–**c** shows the results of the point-based approach, and the bottom row **d**–**f** displays the results of the cell-based approach. **a** and **d** are the reconstructed point clouds; **b** and **e** show the selected foreground (*green*)/background (*red*) seeds; **c** and **f** are the final segmentation results

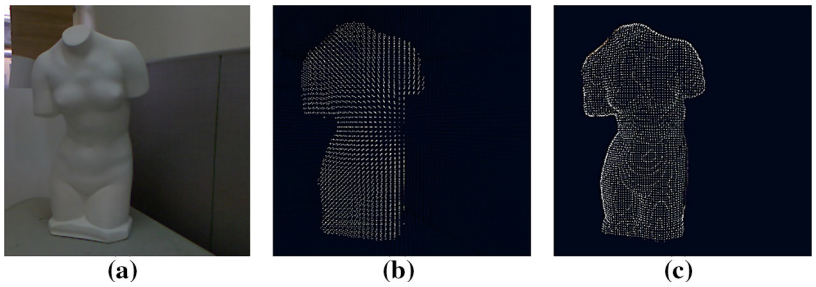


Fig. 13 **a** One of the statue images. **b** Reconstructed points for the statue using the cell-based approach. **c** Reconstructed points using the ray-casting algorithm from the results of the cell-based segmentation

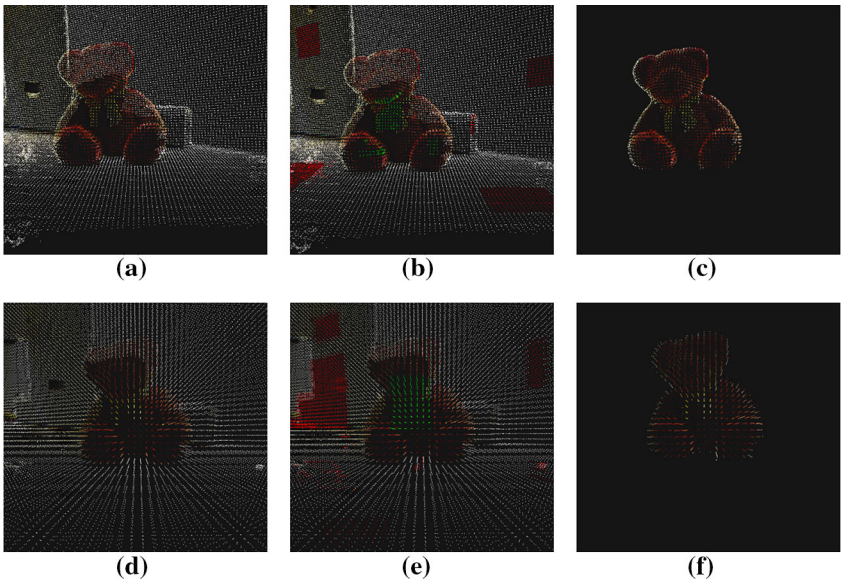


Fig. 14 Example reconstruction of a bear. The *upper row* **a**–**c** shows the results of the point-based approach, and the *bottom row* **d**–**f** displays the results of the cell-based approach. **a** and **d** are the reconstructed point clouds; **b** and **e** show the selected foreground (green)/background (red) seeds; **c** and **f** are the final segmentation results

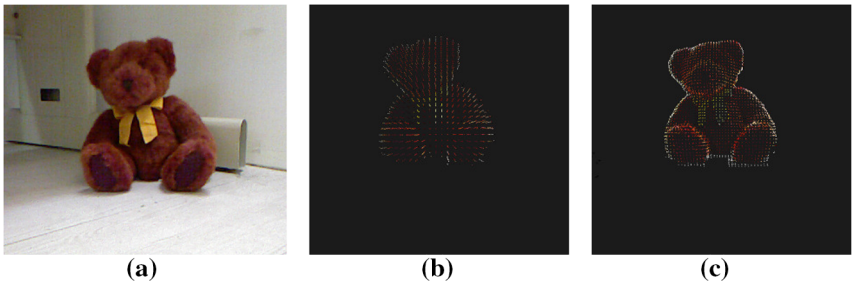
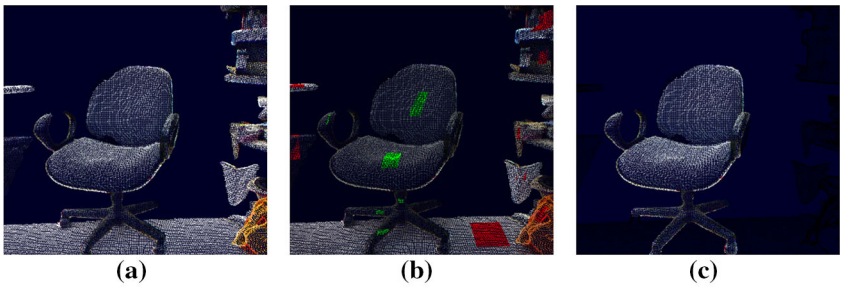


Fig. 15 **a** One of the bear images. **b** Reconstructed points for the bear using the cell-based approach. **c** Reconstructed points using the ray-casting algorithm from the results of the cell-based segmentation

Fusion is run in the GPU to achieve real-time performance. To facilitate the operation of selecting the foreground and background seeds, we have bound an Arduino module with a BlueTooth XBee on the Kinect (see Fig. 8) to replace the function of a mouse. With this Arduino module, users do not need to go back to the desktop, but instead directly move the Kinect in order to control the movement of the window center, which serves as the cursor, and then press the button on the Arduino to achieve the seed selection.

In our system, we use $512 \times 512 \times 512$ TSDF cells for scene reconstruction. Figure 9 shows the reconstructed surfaces for several scenes using KinectFusion. Obviously, the number of cells results in a trade-off between the resolution of the reconstructed surface and the required memory. The occupied space is $1.5 \times 1.5 \times 1.5$ m, about 75 cm in front of the Kinect. This indicates that our system has a size limitation for reconstructed scenes. The object to be reconstructed cannot be too large and should be placed about 1 m in front of the Kinect sensor. This point is also illustrated in Fig. 9, where the black background indicates that the object is outside the volume of the TSDF cells.

Figure 10 shows an example of application of our system to extract the 3D point cloud of a backpack. Here, we show both the results of the point-based and cell-based approaches for comparison. From this figure, we can observe that the point-based approach can produce denser points than the cell-based approach. As discussed in the previous sections, the cell-based approach extracts the object based on the TSDF cells; thus, each point in Fig. 10d–f represents one cell. Hence, the number of reconstructed points



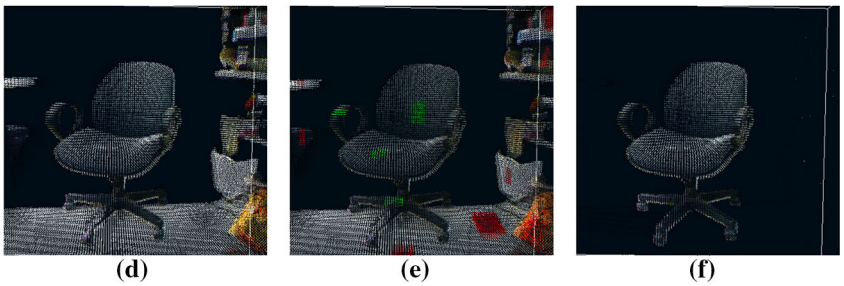


Fig. 16 Example reconstruction of a chair. The *upper row* **a**–**c** shows the results of the point-based approach, and the *bottom row* **d**–**f** displays the results of the cell-based approach. **a** and **d** are the reconstructed point clouds; **b** and **e** show the selected foreground (*green*)/background (*red*) seeds; **c** and **f** are the final segmentation results

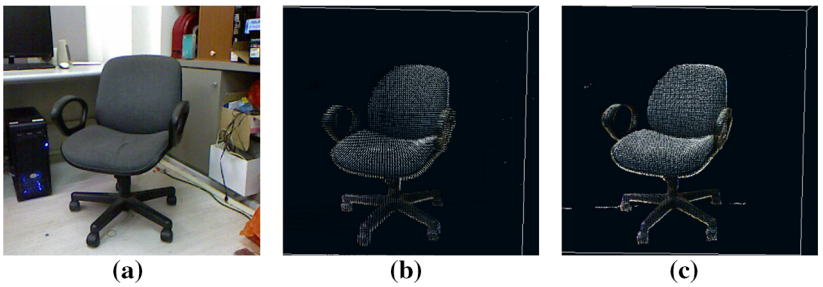


Fig. 17 **a** One of the chair images. **b** Reconstructed points for the chair using the cell-based approach. **c** Reconstructed points using the ray-casting algorithm from the results of the cell-based segmentation

depends on the number of created cells. On the other hand, the point-based approach generates a point cloud using ray casting and then combines points frame by frame using the ICP algorithm. Therefore, it can typically produce more points, since several rays may intersect the same TSDF cell and thus produce more points for a cell. Nevertheless, the cell-based approach can also produce dense points if a post-ray-casting process is applied to the identified cells. This point is illustrated in Fig. 11, which shows a source image, the extracted cell points for the backpack, and a dense point reconstruction after applying the ray-casting algorithm on the segmented cells. Comparing Figs. 10c and 11c, we can see that after this post-processing the segmentation result of the cell-based approach is comparable to that of the point-based approach.

Although the point-based approach can recover more points during the segmentation process, it requires more computational time in searching for adjacent nodes. Table 2 shows the number of reconstructed points, the number of selected foreground/background seeds, and the required time for searching for neighboring points and the segmentation process. In this example, the selected numbers of seeds for the point- and cell-based approaches are very close, thus reducing their effects on the comparison of timing. From this table,

Table 3 Comparison of the point- and cell-based approaches for the reconstruction of a statue as an example

	Number of points	
	Point based	Cell based
Total points	53737	42850
Foreground seeds	319	301
Background seeds	3170	3070
	Processing time (s)	
	Point based	Cell based
Search neighboring points	0.226	0.03
Segment foreground/background	0.955	0.337

the advantages of the cell-based approach are illustrated (i.e., it has more efficient neighboring node identification and faster foreground extraction). The major reason for the high efficiency of the cell-based approach in identifying neighboring nodes comes from the regular arrangement of the TSDF cells. This indicates that the cell-based approach does not need any complex searching process, such as the Kd-tree used in the point-based approach. For the process of segmenting the foreground and background, since the cell-based

Table 4 Comparison of the point- and cell-based approaches for the reconstruction of a bear as an example

	Number of points	
	Point based	Cell based
Total points	61862	28730
Foreground seeds	263	340
Background seeds	3004	2995
	Processing time (s)	
	Point based	Cell based
Search neighboring points	0.218	0.032
Segment foreground/background	0.405	0.125

Table 5 Comparison of the point- and cell-based approaches for the reconstruction of a chair as an example

	Number of points	
	Point based	Cell based
Total points	97994	74915
Foreground seeds	1112	832
Background seeds	2890	2749
	Processing time (s)	
	Point based	Cell based
Search neighboring points	0.387	0.029
Segment foreground/background	0.797	0.587

approach typically recovers fewer points than the point-based approach, it is definitely faster than the point-based approach.

To further evaluate the effectiveness of the proposed system, more experiments were conducted. The results are shown in Figs. 12, 13, 14, 15, 16, 17 and Tables 3, 4, 5. These examples include a statue, a Teddy bear, and a chair with a quite complex background. From these examples, we can draw similar conclusions as from the backpack case. Our system can create the 3D structure of the scene and successfully extract the 3D point cloud of the target object. The selection of the foreground and background seeds is also simple. We only need to roughly select several small regions in the background and foreground areas. The system can take over the remaining task to complete the object extraction.

Another feature of the proposed system is progressive reconstruction and segmentation. The result of the initial segmentation can serve as the seeds for subsequent segmentation. Thus, when we move the Kinect around the object, the segmentation process will be repeatedly activated to recover a more complete point cloud. Figure 18 shows the intermediate segmentation results for the statue example. As we can see, the point cloud of the statue is gradually reconstructed, including the points on the front and back surfaces of the statue. Figure 19 shows another example of progressive reconstruction where a more complete point cloud can be further observed.

After recovering points on the objects, we can then generate a polygonal mesh model for the objects. Figure 20 shows two examples of polygonal mesh creation using the point clouds generated by our system. Here, we employed the greedy triangulation algorithm from Point Cloud Library (PCL) to achieve the task. Because of the simplicity of this algorithm, there are some small holes on the recovered polygonal mesh model. Nevertheless, the model is still quite complete and we think that with a more sophisticated mesh

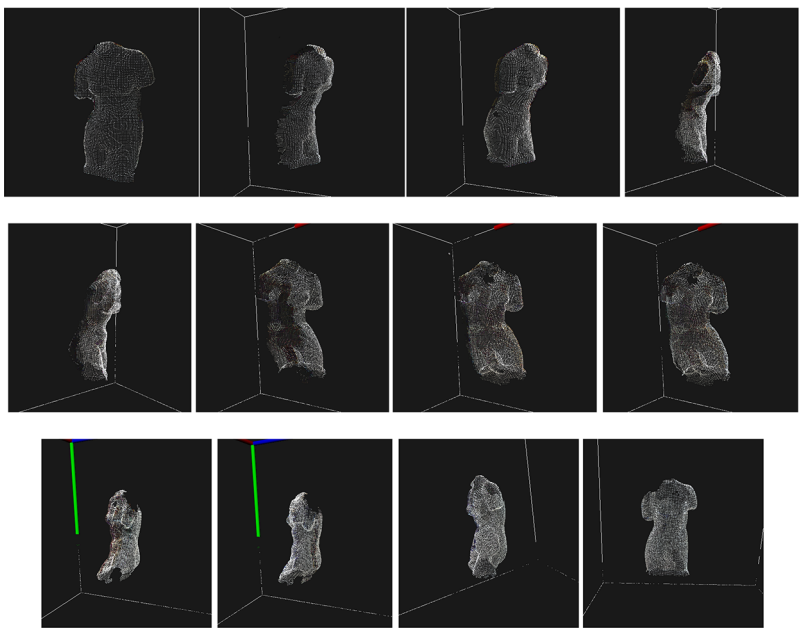


Fig. 18 Progressive reconstruction of a statue example

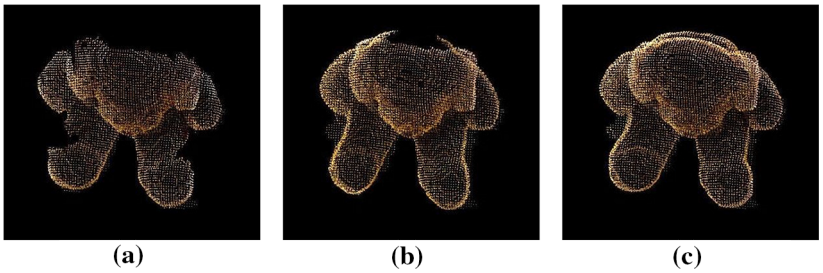


Fig. 19 Another example of a progressive reconstruction. **a** Initial reconstruction of a bear. **b**, **c** More complete point cloud for the object can be progressively reconstructed when we move the sensor around the object

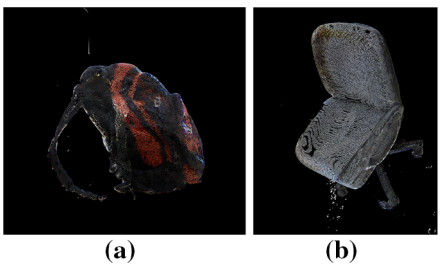


Fig. 20 Polygonal mesh models for a backpack and a chair

creation algorithm, the small holes on the model can be filled to generate a more accurate model.

5 Conclusion

In this paper, we presented an interactive 3D model reconstruction system based on a commercial RGB-D sensor. This system is designed based on a well-performing scene surface recovery algorithm and an interactive segmentation method to achieve real-time object reconstruction. By collecting depth information and averaging them in TSDF cells, the system can effectively eliminate the influence of noisy depth data and hence produce satisfactory surface reconstruction. By exploiting the power of a graph-cut segmentation algorithm, our system does not require users to carefully select the foreground and background seeds, and thus maximally reduces the required user effort.

To maintain the real-time characteristics, the segmentation process is implemented in another thread and the 3D structure of the target object can be progressively reconstructed when a user moves the sensor around the object. In addition, in this paper we also presented two approaches for segmentation graph creation, i.e., the point- and cell-based approaches. Both have advantages and some experiments were conducted to compare their performance. These experiments also confirmed the feasibility and effectiveness of our system for object 3D reconstruction. The 3D models of several objects were reconstructed with acceptable accuracy.

Currently, our system has the drawback of size limitation, i.e., it cannot be applied on the objects with size greater than 1.5m^3 because of the setup of the TSDF volume. Thus, our future work is to relieve the size limitation by exploiting a more efficient data structure to store the sparse TSDF cells. However, this implies that we require an efficient method to find the adjacent TSDF cells. That is, we need to incorporate the mechanism of finding adjacent points in point-based approach with the cell-based approach to achieve sparse TSDF cell processing. We hope that by this approach, we can achieve large object 3D reconstruction. In addition, the sensor we used is the first-generation Kinect sensor. We believe that the reconstructed result may be further improved with the next-generation Kinect sensor.

Acknowledgements This work was supported by the Ministry of Science and Technology, Taiwan, under Grant Nos. NSC 102-2221-E-155-075 and MOST 105-2218-E-155-010.

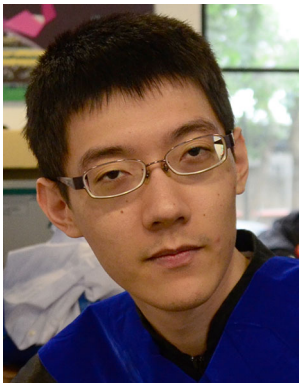
References

1. Albitar, I., Graebling, P., Doignon, C.: Robust structured light coding for 3D reconstruction. In: 11th IEEE International Conference on Computer Vision, pp. 1–6 (2007)
2. Anwer, A., Ali, S.S.A., Mériaudeau, F.: Underwater online 3D mapping and scene reconstruction using low cost Kinect RGB-D sensor. In: 6th International Conference on Intelligent and Advanced Systems (ICIAS) (2016)
3. Bay, H., Ess, A., Tuytelaars, T., van Gool, L.: Speeded-up robust features (SURF). Comput. Vision Image Underst. **110**(3), 346–359 (2008)
4. Boykov, Y., Kolomogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans. Pattern Anal. Mach. Intell. **26**, 1124–1137 (2004)
5. Bradley, D., Boubekeur, T., Heidrich, W.: Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In: International Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008)
6. Chen, C., Zou, W., Wang, J.: 3D surface reconstruction based on Kinect. In: 2013 IEEE Third International Conference on Information Science and Technology (ICIST), pp. 986–990 (2013)
7. Chen, K., Lai, Y.K., Wu, Y.X., Martin, R., Hu, S.M.: Automatic semantic modeling of indoor scenes from low-quality RGB-D data using contextual information. ACM Trans. Graph. **33**(6), 208:1–208:12 (2014)
8. Clark, J.J.: Active photometric stereo. In: International Conference on Computer Vision and Pattern Recognition, pp. 29–34 (1992)
9. Cochran, S.D., Medioni, G.: 3D surface description from binocular stereo. IEEE Trans. Pattern Anal. Mach. Intell. **14**(10), 981–994 (1992)
10. Dibra, E., Jain, H., Öztireli, A.C., Ziegler, R., Gross, M.H.: HS-Nets: Estimating human body shape from silhouettes with con-

volutional neural networks. In: Fourth International Conference on 3D Vision, 3DV 2016, Stanford, CA, USA, October 25-28, 2016, pp. 108–117 (2016). http://dx.doi.org/10.1109/3DV.2016.19
11. Du, H., Henry, P., Ren, X., Cheng, M., Goldman, D.B., Seitz, S.M., Fox, D.: Interactive 3D modeling of indoor environments with a consumer depth camera. In: 13th ACM International Conference on Ubiquitous Computing, pp. 75–84. Beijing, China (2011)
12. Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., Burgard, W.: An evaluation of the RGB-D SLAM system. In: IEEE International Conference on Robotics and Automation (2012)
13. Engelhard, N., Endres, F., Hess, J., Sturm, J., Burgard, W.: Real-time 3D visual slam with a hand-held RGB-D camera. In: The RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum (2011)
14. Geiger, A., Ziegler, J., Stiller, C.: Stereoscan: Dense 3D reconstruction in real-time. In: IEEE Intelligent Vehicles Symposium, pp. 963–968 (2011)
15. Geng, J.: Structured-light 3D surface imaging: a tutorial. Adv. Opt. Photonics **3**, 128–160 (2011)
16. Han, J., Shao, L., Xu, D., Shotton, J.: Enhanced computer vision with microsoft kinect sensor: a review. IEEE Trans. Cybern. **43**(5), 1318–1334 (2013)
17. Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.: RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In: 12th International Symposium on Experimental Robotics (ISER) (2010)
18. Higo, T., Matsushita, Y., Joshi, N., Ikeuchi, K.: A hand-held photometric stereo camera for 3D modeling. In: 12th IEEE International Conference on Computer Vision, pp. 1234–1241 (2009)
19. Horaud, R., Hansard, M., Evangelidis, G., Ménier, C.: An overview of depth cameras and range scanners based on time-of-flight technologies. Mach. Vis. Appl. **27**, 1005–1020 (2016)
20. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A.: KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In: 24th Annual ACM Symposium on User Interface Software and Technology, pp. 559–568 (2011)
21. Li, Y., Sun, J., Tang, C.K., Shum, H.Y.: Lazy snapping. ACM Trans. Graph. **23**(3), 303–308 (2004)
22. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
23. Meilland, M., Comport, A., Rives, P.: Real-time dense visual tracking under large lighting variations. In: British Machine Vision Conference, pp. 1–11 (2011)
24. Meilland, M., Comport, A., Rives, P.: Dense RGB-D mapping for real-time localisation and navigation. In: Workshop Navigation Positioning Mapping (2012)
25. Meister, S., Izadi, S., Kohli, P., Haemmerle, M., Rother, C., Kondermann, D.: When can we use KinectFusion for ground truth acquisition? In: Workshop Color-Depth Camera Fusion Robot (2012)
26. Morana, M.: 3D scene reconstruction using Kinect. In: Gaglio, S., Lo Re, G. (eds.) Advances onto the Internet of Things: How Ontologies Make the Internet of Things Meaningful, pp. 179–190. Springer, Cham (2014)
27. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking. In: IEEE International Symposium on Mixed and Augmented Reality, pp. 127–136. Basel, Switzerland (2011)
28. Pan, R., Taubin, G.: Automatic segmentation of point clouds from multi-view reconstruction using graph-cut. Vis. Comput. **32**(5), 601–609 (2016)
29. Pollefeys, M., Gool, L.V., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., Koch, R.: Visual modeling with a hand-held camera. Int. J. Comput. Vis. **59**(3), 207–232 (2004)
30. Ribo, M., Brandner, M.: State of the art on vision-based structured light systems for 3D measurements. In: International Workshop on Robotic Sensors: Robotic and Sensor Environments 2005, pp. 2–6 (2005)
31. Roth, H., Vona, M.: Moving volume KinectFusion. In: British Machine Vision Conference, pp. 1–11 (2012)
32. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: International Conference on Computer Vision, pp. 2564–2571 (2011)
33. Sahillioğlu, Y., Yemez, Y.: Coarse-to-fine surface reconstruction from silhouettes and range data using mesh deformation. Comput. Vis. Image Underst. **114**, 334–348 (2010)
34. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3D. ACM Trans. Graph. (SIGGRAPH 2006) **25**(3), 835–846 (2006)
35. Song, T., Lyu, Z., Ding, X., Wan, Y.: 3D surface reconstruction based on kinect sensor. Int. J. Comput. Theory Eng. **5**(3), 567–573 (2013)
36. Song, X., Zhong, F., Wang, Y., Qin, X.: Estimation of Kinect depth confidence through self-training. Vis. Comput. **30**(6), 855–865 (2014)
37. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: International Conference on Computer Vision (1998)
38. Valentin, J., Vineet, V., Cheng, M.M., Kim, D., Shotton, J., Kohli, P., NieBner, M., Criminisi, A., Izadi, S., Torr, P.: SemanticPaint: interactive 3D labeling and learning at your fingertips. ACM Trans. Graph. **34**(5), 154 (2015)
39. van den Hengel, A., Dick, A., Thormählen, T., Ward, B., Torr, P.H.S.: VideoTrace: rapid interactive scene modeling from video. ACM Trans. Graph. **26**(3), 86 (2007). doi:10.1145/1276377.1276485
40. Whelan, T., Kaess, M., Fallon, M., Johannsson, H., Leonard, J., McDonald, J.: Kintinuous: Spatially extended KinectFusion. Tech. rep., Computer Science and Artificial Intelligence Laboratory (2012). MIT-CSAIL-TR-2012-020
41. Wolberg, G., Zokai, S.: PhotoSketch: a photocentric urban 3D modeling system. The Visual Computer (2017). doi:10.1007/s00371-017-1365-x
42. Zhang, Z.: Iterative point matching for registration of free-form curves and surfaces. Int. J. Comput. Vis. **13**(2), 119–152 (1994)
43. Zheng, J.Y.: Acquiring 3-D models from sequences of contours. IEEE Trans. Pattern Anal. Mach. Intell. **16**(2), 163–178 (1994)

Chin-Hung Teng received his Ph.D. degree in Electrical Engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2005. He is currently an assistant professor at the Department of Information Communication, Yuan Ze University. His research interests include color image processing, computer vision, computer graphics, image-based modeling, and augmented reality.

Kai-Yuan Chuo received his M.S. degree in Information Communication from Yuan Ze University, Taiwan, in 2015. His research interests include computer vision and 3D reconstruction.



Chen-Yuan Hsieh received his M.S. degree in Information Communication from Yuan Ze University, Taiwan, in 2013. His research interests include Kinect application and 3D reconstruction