

# Chapter 5

## 3D Reconstruction from RGB-D Data



Charles Malleson, Jean-Yves Guillemaut and Adrian Hilton

**Abstract** A key task in computer vision is that of generating virtual 3D models of real-world scenes by reconstructing the shape, appearance and, in the case of dynamic scenes, motion of the scene from visual sensors. Recently, low-cost video plus depth (RGB-D) sensors have become widely available and have been applied to 3D reconstruction of both static and dynamic scenes. RGB-D sensors contain an active depth sensor, which provides a stream of depth maps alongside standard colour video. The low cost and ease of use of RGB-D devices as well as their video rate capture of images along with depth make them well suited to 3D reconstruction. Use of active depth capture overcomes some of the limitations of passive monocular or multiple-view video-based approaches since reliable, metrically accurate estimates of the scene depth at each pixel can be obtained from a single view, even in scenes that lack distinctive texture. There are two key components to 3D reconstruction from RGB-D data: (1) spatial alignment of the surface over time and, (2) fusion of noisy, partial surface measurements into a more complete, consistent 3D model. In the case of static scenes, the sensor is typically moved around the scene and its pose is estimated over time. For dynamic scenes, there may be multiple rigid, articulated, or non-rigidly deforming surfaces to be tracked over time. The fusion component consists of integration of the aligned surface measurements, typically using an intermediate representation, such as the volumetric truncated signed distance field (TSDF). In this chapter, we discuss key recent approaches to 3D reconstruction from depth or RGB-D input, with an emphasis on real-time reconstruction of static scenes.

---

C. Malleson (✉) · J.-Y. Guillemaut · A. Hilton  
Centre for Vision, Speech and Signal Processing & University of Surrey, Guildford  
GU2 7XH, UK

e-mail: [charles.malleson@surrey.ac.uk](mailto:charles.malleson@surrey.ac.uk)

J.-Y. Guillemaut

e-mail: [j.guillemaut@surrey.ac.uk](mailto:j.guillemaut@surrey.ac.uk)

A. Hilton

e-mail: [a.hilton@surrey.ac.uk](mailto:a.hilton@surrey.ac.uk)

© Springer Nature Switzerland AG 2019  
P. L. Rosin et al. (eds.), *RGB-D Image Analysis and Processing*,  
Advances in Computer Vision and Pattern Recognition,  
[https://doi.org/10.1007/978-3-030-28603-3\\_5](https://doi.org/10.1007/978-3-030-28603-3_5)

## 5.1 Introduction

The ability to model the real world in 3D is useful in various application areas from archaeology and cultural heritage preservation to digital media production and interactive entertainment, robotics and healthcare. A key task in computer vision is that of automatically generating virtual 3D models of real world scenes by reconstructing the shape, appearance and, in the case of dynamic scenes, motion of surfaces within the scene from images, video and other sensor input.

Traditionally, 3D reconstruction has been performed by photogrammetry from standard RGB cameras or using costly, specialized laser scanning equipment. Recently, low-cost video plus depth (RGB-D) sensors have become widely available and have been applied to 3D reconstruction of both static and dynamic scenes. RGB-D sensors contain an active depth sensor, which provides a stream of depth maps alongside standard colour video. Typical depth sensors are based on infrared structured light or time-of-flight principles (see Chap. 1 for an in depth overview of commodity depth capture devices). The low cost and ease of use of RGB-D devices as well as their video rate capture of images along with depth make them well suited to 3D reconstruction. Use of active depth capture overcomes some of the limitations of passive monocular or multiple-view video-based approaches since reliable, metrically accurate estimates of the scene depth at each pixel can be obtained from a single view, even in scenes that lack distinctive texture.

There are two key components to 3D reconstruction from RGB-D data:

- Spatial registration (alignment) of the surface over time
- Fusion of noisy, partial surface measurements into a more complete, consistent 3D model.

In the case of static scenes, the sensor is typically moved around the scene to obtain more complete coverage and the registration process amounts to estimating the sensor pose (ego motion) over time. For dynamic scenes, in addition to sensor motion, there may be multiple rigid, articulated, or non-rigidly deforming surfaces present, which need to be tracked over time in order to obtain a consistent surface model. The fusion component in 3D reconstruction consists of integration of the aligned surface measurements, typically using an intermediate representation, such as the volumetric truncated signed distance field (TSDF) before extracting an output mesh model.

In this chapter, we provide an overview of several approaches to static and dynamic 3D reconstruction from depth or RGB-D input, some of which operate online, often in real-time, and others which require offline or batch processing. A broad overview of recent static reconstruction approaches is presented in Sect. 5.2, followed in Sect. 5.3 by a more detailed description and evaluation of two real-time static scene reconstruction approaches, volumetric-based KinectFusion [44] and point-based surface fusion [33]. A brief overview of recent dynamic scene reconstruction approaches is presented in Sect. 5.4 and concluding remarks are provided in Sect. 5.5.

## 5.2 Overview of Rigid Reconstruction Approaches

Standard cameras produce images containing colour or intensity information. These images are inherently 2D and for general scenes, estimated correspondences between multiple disparate images are required in order to infer metrically accurate 3D geometry if camera poses are known (multiview stereo) or 3D geometry up to a scale factor if not (structure from motion [11, 45]). It is possible to estimate an approximate depth directly from a single monocular image, for instance using deep learning-based approaches, e.g. [18], however due to the ill-posed nature of the problem, such inferred depth is typically limited in terms of metric accuracy. Active depth sensors such as structured light or time-of-flight (ToF) cameras on the other hand, natively output either images of metrically accurate depth values, i.e. 2.5D depth maps which can be re-projected into 3D using the intrinsic camera calibration parameters; or in the case of some laser scanners may directly output a 3D ‘point cloud’, with or without a 2D image structure. Core approaches to registration and integration of surface measurements in are discussed below in Sects. 5.2.1 and 5.2.2, respectively.

### 5.2.1 Surface Registration

Much research has been done on tracking (finding the 6DOF pose of the sensor) and mapping (measurement integration) using multiple depth maps. If there is a large relative motion between the point clouds to be registered, a *coarse* registration needs to be performed in order to get them into approximate alignment and avoid local minima when subsequently performing *fine* registration (see [60] for a detailed review). Coarse registration is often performed using sparse feature matching, whereas accurate fine registration is usually performed using the full data set [44].

The Iterative Closest Point (ICP) algorithm introduced by Besl and McKay [2] forms the basis of most registration algorithms. In ICP data alignment is formulated as an iterative optimization of a 3D rigid body transform so as to minimize a cost function representing the distance between points on a ‘source’ (data) and their corresponding closest points on a ‘target’ (model) surface in alternation with updating the closest point correspondences. The translation is found directly from the centroids, and the rotation is found by constructing a cross-covariance matrix. In practice, because the ICP optimization converges monotonically to a local minimum, one either needs to try several initial rotations, or use a feature-based initial coarse alignment algorithm to increase the chance of finding the global minimum. The more complex the shape the more initial states are required (highly symmetrical shapes are most problematic and may result in the solution being under-constrained). Besl and McKay’s method cannot directly handle non-overlapping data unless it is modified for robustness, for instance by using a maximum correspondence distance [60]. If, however the sensor is not moved significantly between frames (as is usually the case when using a handheld video rate sensor such as the Kinect [44]), the pose from the previous frame can be used as initialization, without performing a coarse registration step.

There exist many variants of the ICP algorithm which offer improved registration performance and or computational efficiency (see [55] for a detailed review). For instance, the point-to-plane method proposed by Chen and Medioni [9] minimizes the distance from the source point to the plane that is defined by the target point and its normal. This makes intuitive sense since the finite sample spacing means that samples in one image will generally not coincide with corresponding samples in the other. This has been shown to improve convergence and is preferred when surface normal estimates are available, as is the case when depth maps are used as input [44, 54]. A normal orientation test can be used to increase robustness by preventing matching of surfaces of opposite orientation (as could occur with thin objects). Luminance or colour image data has also been used in the ICP framework to help constrain the registration in cases where shape alone is ambiguous (for instance spheres). In [73], luminance information from a colour camera used in parallel with a depth sensor is used to establish point-to-point correspondences via a form of optical flow at each iteration.

Obtaining closest point associations for ICP is computationally expensive. When the points to be aligned come in a structured form (as with the 2D grid structure of depth images), significant speedups can be introduced by using the projective data association algorithm proposed by Blais and Levine [3]: using the intrinsic camera calibration information, transformed 3D points from the target image are projected into the source image to get the pixel index correspondences. Fitzgibbon [15] extends the ICP algorithm to perform robust registration using a Huber kernel and Levenberg–Marquardt (LM) non-linear optimization. This optimization approach yields a wider basin of convergence than standard ICP. The ‘generalized ICP’ proposed by Segal et al. [61] introduces a probabilistic framework and uses planar surface structure in both the data and the model (a plane-to-plane metric).

One way of increasing the speed of convergence of ICP is by performing early iterations on a subset of the available points for instance using a coarse-to-fine (multi-resolution) sampling of the depth map [44]. Other useful ways of subsampling include random subsampling and sampling based on colour information. Rusinkiewicz and Levoy [55] propose normal-space subsampling which bins points based on normals and samples uniformly across buckets, thus promoting correct registration of scenes containing no large distinctive features. ICP registration can also be extended to handle articulated point clouds [7, 13, 50] in which case pose parameters are iteratively determined for each bone in a skeletal model.

### 5.2.2 *Surface Fusion*

As stated in [23], the goal of 3D surface reconstruction is to estimate a manifold surface (with the correct topology) that accurately approximates an unknown object surface from a set of measured 3D sample points. When additional information (such as measurement uncertainty) is available it can aid reconstruction. There are two classes of technique for reconstructing 3D models from 2.5D images [23]. The first

class uses prior models with an explicit parametric representation and fits the range data to them. The disadvantage of such methods is that they can only represent models of known object classes, for which the topological genus and modes of shape variation are known upfront (e.g. using a radial displacement map on a cylinder to model a human head [22]). The second class of techniques, which generate triangulated mesh representations is more generally applicable because it can represent arbitrary geometry and topology (which are often not known up front). The focus of this discussion will be on non-parametric approaches, in particular the widely used signed distance function.

### 5.2.2.1 Signed Distance Functions

To facilitate the generation of a 3D surface model by the fusion of aligned 2.5D depth maps it is common to use an intermediate non-parametric representation of volumetric occupancy. A representation widely used in computer vision graphics is the Signed Distance Function (SDF) introduced by Curless and Levoy [10]. The SDF is simply a field whose value at any given point contains the (signed) Euclidean distance between that point and the surface. Thus the SDF is zero at the surface interface, positive outside it (observed free space), and negative inside it (unobserved space). In practice, the SDF is represented in a discrete voxel grid defining the reconstructed volume and is truncated at a certain distance from the surface i.e. values more than a certain distance in front of a surface measurement receive a maximum value, and values more than a certain distance,  $\mu$ , behind it receive no measurement (null). This truncation helps prevent surfaces from interfering with each other. Along with each Truncated Signed Distance Function (TSDF) value a weight is maintained which reflects the confidence in the TSDF value. These weights may depend on the confidence of a measurement (if available) or heuristics (for instance penalizing vertices whose estimated normal is close to perpendicular to the viewing direction or which are close to depth discontinuities [68]). A simple weighted running average update rule for the SDF and weight voxel grid can be used to incrementally incorporate measurements into the model, which adds any previously unobserved regions to the model, while averaging out noise in regions previously observed.

Obtaining the surface interface from the TSDF is simply a matter of extracting the zero crossings (an iso-surface at level zero). This is an advantage over probabilistic occupancy grids where one needs to seek the modes of the probability distribution in the grid [44]. If only a single view is required, one can perform a direct raycast [49] which is independent of scene complexity since areas outside the viewing frustum need not be visited. If, however, a complete polygonal mesh model is required a triangulation algorithm such as marching cubes is more suitable.

Originally proposed by Lorensen and Cline [36], the marching cubes algorithm is widely used for extracting triangle meshes from constant density surfaces (iso-surfaces) in volumetric datasets. The algorithm scans through a voxel grid and processes one  $2 \times 2 \times 2$  cell at a time using lookup tables to determine the triangle

topology within the cell and interpolation between vertices to find iso-surface intersections. This is efficient, but results in non-uniform triangle shape and size.

The ‘marching triangles’ algorithm proposed by Hilton et al. [24] performs the same task as marching cubes, but uses Delaunay triangulation and places vertices according to local surface geometry thus producing triangles with more uniform shape and size.

### 5.2.2.2 Other Surface Fusion Approaches

The point-based implicit surface reconstruction of Hoppe et al. [25] works with unorganized points and generates simplicial surfaces (i.e. triangle meshes) of arbitrary topology. It uses a signed distance function computed with the aid of normals estimated from  $k$ -nearest neighbour PCA with a graph optimization to get consistent orientations. When the source points come from (inherently organized) depth maps, normals may be estimated more efficiently using the image structure.

Turk and Levoy [68] create polygon meshes from multiple (ICP-registered) range images and then ‘zipper’ them together, that is they remove redundant triangles and connect (‘clip’) the meshes together. The mesh growing technique of Rutishauser et al. [57] merges depth maps incrementally with particular emphasis on the (anisotropic Gaussian) error model of their sensor and uses an explicit boundary representation to prevent filling surfaces in the model where no measurements have been made. Soucy and Laurendeau [63] estimate an integrated surface model piecewise from the canonical subset of the Venn diagram of the set of range views (here, a canonical subset contains a group of points exclusively visible in a particular combination of range views). This membership information is used in an averaging process, taking particular care at the intersections of subsets. The ball pivoting algorithm of Bernardini et al. [1] triangulates point clouds efficiently by beginning with a seed triangle and rotating a sphere around an edge until another point is reached, at which point another triangle is formed.

Hilton et al. [23] introduce a mesh-based geometric fusion algorithm based on a continuous implicit surface which (unlike previous algorithms employing discrete representations) can better reconstruct regions of complex geometry (holes, crease edges and thin objects). The algorithm also uses geometric constraints and statistical tests based on measurement uncertainty to guide reconstruction of complex geometry. While outperforming other integration strategies [10, 25, 57, 63, 68] in terms of complexity and the minimum feature size, minimum crease angle and minimum surface separation (thickness), this approach is relatively computationally expensive.

Radial Basis Functions (RBFs) have been used for interpolation of a surface from point samples. Globally supported RBFs are good at filling in missing data, but are computationally inefficient. Conversely, locally supported RBFs are less good at filling in missing data, but are more computationally efficient. Ohtake et al. [48] therefore propose to use compactly supported RBFs with a coarse-to-fine sampling of the points. The coarse levels fill in missing data and serve as carriers for the finer levels which add detail.

Kazhdan et al. [32] used oriented points to define an indicator function with value 1 inside the model and 0 outside it and cast the optimization as a Poisson problem. The resulting reconstructions are inherently watertight. Like Radial Basis Function (RBF) approaches, the method creates smooth surfaces. This can result in spurious protrusions from regions where no samples exist. The method is best suited to scenarios where the capture process has produced full surface coverage, such as single objects captured via a moving laser scanner or segmented performers in a multiview reconstruction setup, but is less well suited to partial coverage of larger scenes.

### 5.3 Real-Time Rigid Reconstruction

The first reported system that uses a low-cost depth sensor to perform real-time, online and metrically consistent 3D reconstruction of small to medium-sized scenes on a commodity PC is the ‘KinectFusion’ system of Newcombe et al. [29, 44], which is described in detail in the following subsections. Since KinectFusion was introduced, several variations on the theme of static scene reconstruction from depth maps have been proposed. Some of these have addressed handling of larger scenes within the limited GPU memory budget, for instance the moving volume approach of Roth and Vona [53], the ‘Kintinuous’ system of Whelan et al. [74], the hierarchical data structure of Chen et al. [8], and the spatial voxel hashing approach of Niessner et al. [46].

Keller et al. [33] propose a point-based alternative to (volumetric) KinectFusion. The unstructured ‘surfel’ (surface element) representation is more memory efficient than volumetric structures and manipulation (e.g. insertion/removal) of individual entities is easier than with structured representations such as meshes. The memory footprint of the surfel representation is significantly lower than for volumetric fusion, but mesh extraction is less straightforward. The dense planar SLAM system of [58] is based on the surfel fusion system of [33], but additionally detects planar regions, which can be stored in a compressed form and used for semantic understanding of the scene.

In SLAM++ [59], 3D models of known objects (such as chairs) are used in an object-level SLAM system which recognizes and tracks repeated instances of these objects in a cluttered indoor scene. The main benefits over standard approaches that use primitive-level tracking and mapping are increased representational efficiency and the native semantic structure of the output scene.

By allowing offline (post) processing, other recent works are able to produce higher quality models than currently possible with real-time methods. Zhou et al. [77] perform fusion of small fragments of a scene, which are each locally accurate, and then combine them via an elastic registration scheme to produce a complete surface with higher detail and reduced low-frequency distortion when compared to using a single grid for the whole scene. The method is off-line and requires hours to days of GPU processing time. Fuhrmann and Goesele’s floating scale reconstruction approach [16] uses compactly supported basis functions for integration into an octree

voxel grid structure and is formulated to take into account the fact that surface measurements represent finite sample areas rather than individual points, thus avoiding potential blurring of fine details by coarser samples that were captured from further away. In [78], Zollhofer et al. refine a model obtained by TSDF fusion of depth maps by using an offline shape-from-shading stage to enhance the level of reconstructed detail compared to depth-only fusion approaches.

The KinectFusion approach of Newcombe et al. [44], described in this section demonstrates the ability of low-cost depth sensors to quickly and cost-effectively produce compelling 3D models of small to medium-sized static indoor scenes by employing GPU acceleration of ICP sensor pose estimation and TSDF volumetric measurement fusion. Online real-time reconstruction of static scenes is achieved using a sequence of depth maps from a handheld Kinect sensor. The core components of the KinectFusion pipeline are model building by integration of captured depth maps into a volumetric TSDF representation (Sect. 5.3.3) and ICP registration of input depth maps to this model (Sect. 5.3.2). Outside the core registration and fusion loop, a textured mesh may be extracted using marching cubes and textured using projective texturing.

The parallelizable parts of the reconstruction pipeline may be implemented on the GPU by using, for instance, NVidia’s CUDA toolkit [47]. Such GPU parallelization involves uploading input data from CPU memory to GPU memory; splitting it into parts, each of which is concurrently processed by a *kernel* function running in parallel *threads* across hundreds or thousands of GPU processing cores; and finally downloading the result back into CPU memory. How the work is split up depends on the application and performance considerations. For example a kernel may perform an operation on one or a small block of pixels or voxels.

Preliminaries of KinectFusion are presented below in Sect. 5.3.1, followed by details on ICP registration (Sect. 5.3.2), TSDF fusion (Sect. 5.3.3) and textured mesh extraction (Sect. 5.3.4). Finally, in Sect. 5.3.5, a related approach proposed by Keller et al. [33] is discussed, in which point-based fusion is used in place of volumetric TSDF fusion.

### 5.3.1 Preliminaries

The input from an RGB-D sensor consists of a video rate stream of RGB colour images,  $C_i(t)$  and depth images,  $D_i(t)$ , containing Cartesian depth  $d(\mathbf{u})$  for each pixel  $\mathbf{u} = (col, row)$ .

A standard pinhole camera model is used to characterise the RGB-D sensor. A fixed  $3 \times 3$  intrinsic camera matrix  $\mathbf{K}$  and a  $4 \times 4$  camera pose matrix  $\mathbf{T}$  (which varies over time) can be used to map between world and pixel coordinates. In practice, the RGB and depth cameras are usually not generated through the same lens aperture, and are thus offset from one another. For simplicity of processing, the depth map is often re-rendered from the RGB camera point of view in order to obtain direct



pixel correspondence between them (this may be done on board the device or as a post-process). The camera matrix  $\mathbf{K}$  is defined as

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (5.1)$$

where  $f_x$  and  $f_y$  and  $c_x$  and  $c_y$  are the  $x$  and  $y$  focal lengths and principal point coordinates, respectively. A rigid body transformation matrix  $\mathbf{T}$  contains a  $3 \times 3$  orthonormal rotation matrix  $\mathbf{R}$  (which has three degrees of freedom) and a 3D translation vector  $\mathbf{t}$ :

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \quad (5.2)$$

The intrinsic parameters of the camera model can be estimated using, for instance, a checkerboard-based approach of Zhang et al. [75].

The following describes how the camera intrinsics and pose can be used to transform between 2D/2.5D image space, camera-local 3D space and global 3D space. Given the intrinsic camera calibration matrix,  $\mathbf{K}$ , an image-space depth measurement  $d(\mathbf{u})$  can be converted to a camera-local 3D point,  $\mathbf{p}_{cam}(\mathbf{u})$ :

$$\mathbf{p}_{cam}(\mathbf{u}) = d(\mathbf{u}) \cdot \mathbf{K}^{-1} \dot{\mathbf{u}}, \quad (5.3)$$

where a dot on a vector denotes its homogeneous form,  $\dot{\mathbf{u}} = [\mathbf{u}^T \ 1]^T$ . This camera-local 3D point can be transformed to a global 3D point,  $\mathbf{p}_{gbl}$ , using the camera pose  $\mathbf{T}$ :

$$\mathbf{p}_{gbl}(\mathbf{u}) = \rho(\mathbf{T} \dot{\mathbf{p}}_{cam}), \quad (5.4)$$

where  $\rho$  is the de-homogenization operator,  $\rho([\mathbf{a}^T \ w]^T) = \mathbf{a}/w$ . Similarly, any global 3D point,  $\mathbf{p}_{gbl}$  can be transformed into camera space:

$$\mathbf{p}_{cam} = \rho(\mathbf{T}^{-1} \dot{\mathbf{p}}_{gbl}), \quad (5.5)$$

and a camera-space 3D point can be projected into the depth map using the intrinsics as follows:

$$[x, y, z]^T = \mathbf{K} \cdot \mathbf{p}_{cam} \quad (5.6)$$

$$\mathbf{u} = [x/z, y/z]^T \quad (5.7)$$

$$d(\mathbf{u}) = z. \quad (5.8)$$

We now describe how the camera pose,  $\mathbf{T}$  is estimated online for each incoming frame, thus allowing input depth frames to be registered for consistent surface integration.

### 5.3.2 Registration Using ICP

The camera pose estimation of KinectFusion is based on the ICP algorithm with fast projective data association [3] and the point-to-plane error metric [9] (see Sect. 5.2.1). The registration is done using the current depth map  $D_i(t)$  as the source and a depth map synthesized from the current volumetric TSDF model of the scene as the target. This synthetic depth map is generated by ray-casting [49] the TSDF voxel grid from a prediction of the sensor pose for the current current frame. By assuming small frame-to-frame camera motion the pose of the target frame can be used as the pose from which to ray cast and also as the initial pose of the source in the ICP algorithm. Because any error in registration of the previous frame will have a relatively small effect on the model, the frame-to-model registration approach yields increased accuracy and significantly reduces the accumulation of drift that occurs in the raw frame-to-frame case, without requiring off-line optimization for loop closure.

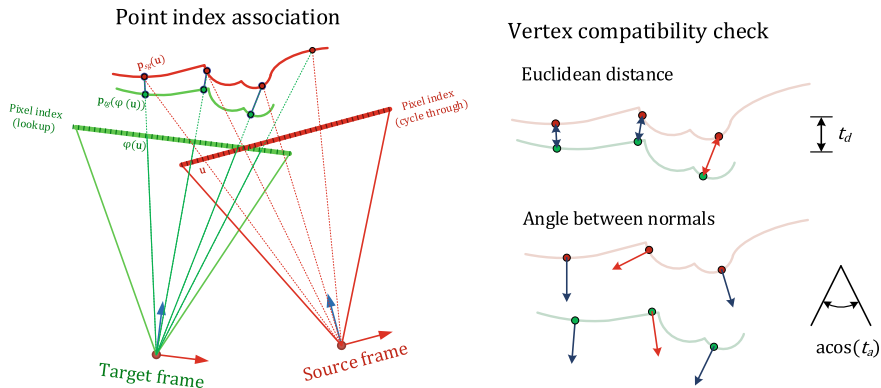
The data association and error minimization stages of the ICP require normals  $\mathbf{n}(\mathbf{u})$  for each pixel  $\mathbf{u}$ . Because the depth map is organized in a grid, adjacency is known and a given normal can be efficiently estimated using the point and its neighbours in the depth image (without the expensive neighbour finding computations required for general unorganized point clouds [25]). Because of their inherent noise however, the raw depth maps produce unacceptably poor normal maps, therefore a GPU-parallelized version of the bilateral filter [67] is applied to the depth map before using it in the registration algorithm, smoothing out noise while preserving depth discontinuities.

As with the normal estimation, the usually expensive data association component of ICP can be sped up significantly by employing the image structure of the depth images. Given global poses for both the source and target frames, each pixel index  $\mathbf{u}$  in the source image is un-projected to form a 3D source point which is projected onto the target image plane to look up the target pixel index  $\Omega(\mathbf{u})$ . The target pixel at this index is then un-projected to get the target 3D point. This data association approach assumes that there is a small frame-to-frame transform between source and target. To remove false correspondences, any point pair separated by a (Euclidean) distance of more than  $t_d$  or whose normals have a dot product of less than  $t_a$  are rejected (see Fig. 5.1). If there is not a valid correspondence between source and target at  $\mathbf{u}$  then  $\Omega(\mathbf{u}) = \text{null}$ . The association is implemented in parallel on the GPU with one thread per source pixel.

Let  $\mathbf{p}_{ss}(\mathbf{u})$  be the 3D point produced by the pixel with index  $\mathbf{u}$  in the source depth map (in its local coordinate system). Let  $\mathbf{p}_{tg}(\phi(\mathbf{u}))$  be the global 3D point produced by the target depth map pixel associated with pixel  $\mathbf{u}$  in the source image.

For any estimated global source frame pose  $\mathbf{T}_{sg}$  the total point-to-plane error  $E$  is then given by

$$E(\mathbf{T}_{sg}) = \sum_{\Omega(\mathbf{u}) \neq \text{null}} \left| \left[ \rho(\mathbf{T}_{sg} \mathbf{p}_{ss}(\mathbf{u})) - \mathbf{p}_{tg}(\Omega(\mathbf{u})) \right]^T \mathbf{n}_{tg}(\Omega(\mathbf{u})) \right|. \quad (5.9)$$



**Fig. 5.1** Illustration of the fast projective data association technique. Left: each source image pixel is un-projected (producing a point in global 3D space) and then projected onto the target image plane to find its associated pixel index (2D coordinates) in the target image. The target image pixel at this index is then looked up and un-projected to produce the target 3D point. Right: the source and associated target points are checked for compatibility, rejecting inconsistent matches. Figure from [37]

The rigid body transform that minimizes  $E$  can be formulated by linearizing the rotation matrix (making use of a small angle assumption for incremental transforms) and writing the transform as a 6D parameter vector  $\mathbf{x} = [\alpha, \beta, \gamma, t_x, t_y, t_z]^T$ , where  $\alpha$ ,  $\beta$  and  $\gamma$  are the rotation angles (in radians) about the  $x$ ,  $y$  and  $z$ -axes, respectively and  $t_x$ ,  $t_y$  and  $t_z$  are the translation components.

As shown in [44], differentiating the linearised objective function and setting it to zero yields a  $6 \times 6$  symmetric linear system

$$\sum_{\phi(\mathbf{u}) \neq \text{null}} \mathbf{a}^T \mathbf{a} \mathbf{x} = \sum \mathbf{a}^T \mathbf{b} \quad (5.10)$$

where

$$\mathbf{a}^T = \left[ [\mathbf{p}_{sg}]_{\times} \mid \mathbf{I}_{3 \times 3} \right]^T \mathbf{n}_{tg}, \quad (5.11)$$

$$[\mathbf{p}]_{\times} := \begin{bmatrix} 0 & -p_z & p_y \\ p_z & 0 & -p_x \\ -p_y & p_x & 0 \end{bmatrix} \quad (5.12)$$

and

$$\mathbf{b} = \mathbf{n}_{tg}^T [\mathbf{p}_{tg} - \mathbf{p}_{sg}]. \quad (5.13)$$

The summands of the normal system are computed in parallel on the GPU, summed using a parallel reduction, and finally solved on the CPU using a Cholesky decomposition followed by forward/backward substitution. At each iteration the

solved incremental transform vector  $\mathbf{x}$  is converted to a  $4 \times 4$  rigid body transform matrix and composed onto the current pose estimate for use in the next iteration. To speed-up convergence of the ICP registration, a coarse-to-fine approach may be used, in which decimated versions of the depth maps are used for early iterations and finally using all points for a more precise registration.

### 5.3.3 Fusion Using TSDFs

Surface integration is performed using TSDF fusion [10], integrating incoming depth maps into the model in an online manner, integrating out noise and increasing scene coverage as more frames are added. The voxel grid  $G = \{S, W\}$  consists of grids  $S$  and  $W$  which contain, for each voxel  $\mathbf{v} = (x, y, z)$ , the truncated signed distance function (TSDF) values  $s(\mathbf{v})$  and weights  $w(\mathbf{v})$ , respectively. The voxel grid dimensions and leaf size as well as its location in global coordinates need to be chosen appropriately. The main constraint on the dimensions of the voxel grid is the limited size of the GPU memory. The leaf size (resolution) is implicitly calculated in terms of the physical volume and memory available. For example, a  $2 \text{ m}^3$  cubic volume with  $512^3$  voxels would have leaves of side 4.0 mm and require approximately 1 GB of GPU memory using 32-bit floating point values. In the absence of constraints on memory, the voxel leaf size should be chosen on the order of the effective size of the input depth pixels in order to reconstruct all available detail.

For efficiency of implementation the projective signed distance function is used (Fig. 5.2). This allows each voxel site  $\mathbf{v}$  to be visited in parallel and the distance along the ray used as an estimate of the TSDF  $s(\mathbf{v})$ . The model is updated incrementally as measurements from frame  $t$  are added using a weighted running average:

$$s_t(\mathbf{v}) = \frac{w_{t-1}(\mathbf{v})s_{t-1}(\mathbf{v}) + w_t^m(\mathbf{v})s_t^m(\mathbf{v})}{w_t(\mathbf{v})} \quad (5.14)$$

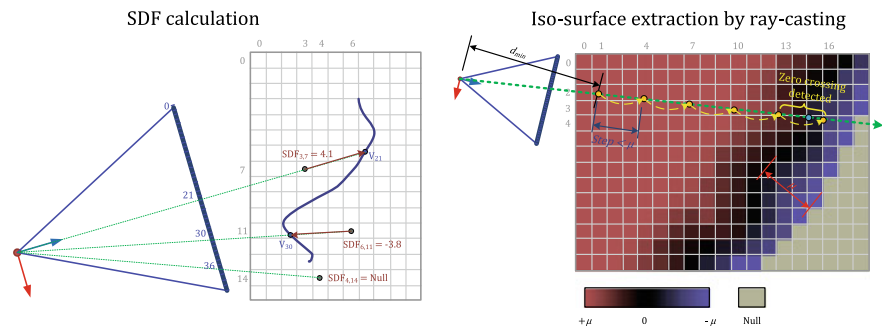
and

$$w_t(\mathbf{v}) = w_{t-1}(\mathbf{v}) + w_t^m(\mathbf{v}) \quad (5.15)$$

where  $s_t^m$  and  $w_t^m$  are the input TSDF and weight values for the current frame.

The truncation distance,  $\mu$ , affects the minimum thickness of objects that can be reconstructed using the TSDF representation (surfaces thinner than  $\mu$  can interfere with each other). This distance also affects the speed of reconstruction: larger values of  $\mu$  allow bigger jumps when ray-casting (see below). If accuracy of fine detail is important, then  $\mu$  should be made as small as possible whilst remaining larger than a few voxel leaf sides. It may also be made depth-dependent to account for the uncertainty in the depth measurement, which in the case of the Kinect v1 sensor increases quadratically with depth [34].

Using the pose from the last frame and the depth intrinsics, a synthetic depth image is generated by doing a per-pixel ray cast into the signed distance voxel grid  $S$ . Rays



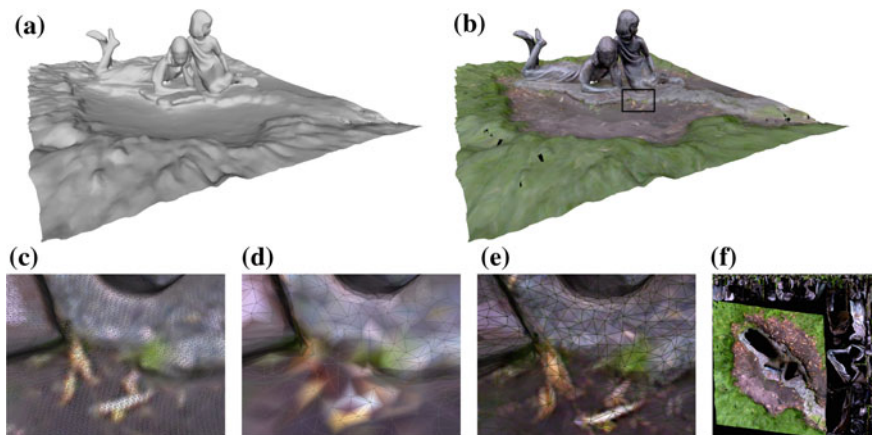
**Fig. 5.2** Illustration of voxel grid model generation and extraction. Left: for a given frame, signed distance values are obtained for each voxel by projecting it onto the image plane and computing the distance between it and this un-projected pixel. Note that the truncation is not shown. Right: to extract a depth map, each pixel is un-projected along its ray starting at the minimum depth  $d_{min}$  and evaluating the tri-linearly interpolated voxel before skipping to the next. When a zero crossing is detected, ray-casting stops and the crossing is located more precisely. Figure from [37]

are marched from the minimum depth sensing range and in steps of  $0.8 \mu$  (slightly less than the minimum truncation distance) until the sign changes from positive to negative indicating a zero crossing (refer to Fig. 5.2). This skipping provides a speed-up whilst ensuring that a zero-crossing is not missed. When a zero crossing is detected, its location is found more precisely by tri-linear interpolation of the SDF before and after the sign change. If a negative-to-positive transition is found marching stops. Each of the pixels is ray cast by a single thread in parallel on the GPU.

### 5.3.4 Model Extraction and Texturing

The marching cubes algorithm [36] may be used to triangulate the TSDF model and generate an output mesh model of the scene. Each vertex of the resulting mesh is then projected into the input RGB images (using the previously estimated camera poses) and the corresponding pixel values looked up. A depth map is produced via an OpenGL rendering of the mesh from the point of view of the RGB camera. This depth map is used to check for occlusions of a given vertex with respect to the corresponding pixel in the RGB image and also to check the proximity of the test pixel to a depth discontinuity.

A simple weighting scheme is used to determine colours for each vertex by incremental update of a weight and colour frame by frame (analogous to Eq. 5.14). The weighting scheme weights contributions from different frame according to the proximity of vertex to depth discontinuities in the current frame as well as the angle between the vertex normal and the camera ray, down weighting contributions for pixels near depth edges and pixel corresponding to obliquely viewed surfaces. The aim of this weighting scheme is to reduce foreground/background texture contami-

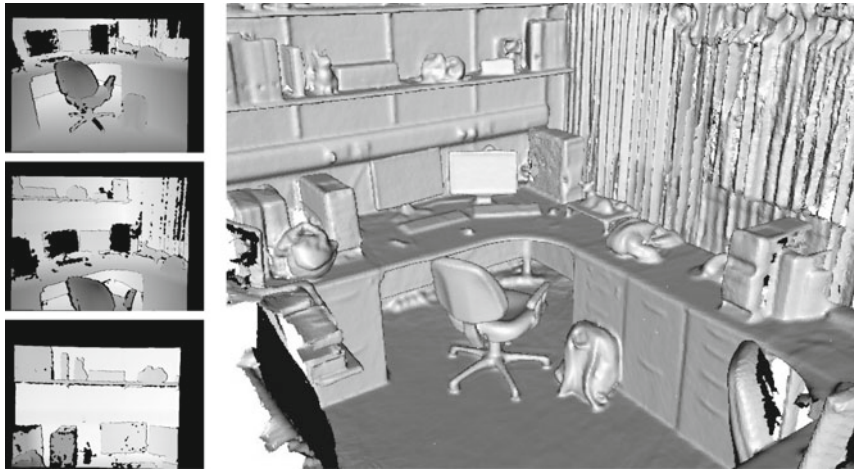


**Fig. 5.3** Reconstruction of a statue using the Asus Xtion Pro Live RGB-D sensor. **a** Decimated mesh (35k vertices). **b** Decimated mesh with texture map applied. **c** Raw mesh (per-vertex colour). **d** Decimated mesh (per-vertex colour). **e** Decimated mesh (texture map colour). **f** Texture map. Figure from [37], using their reimplementation of KinectFusion

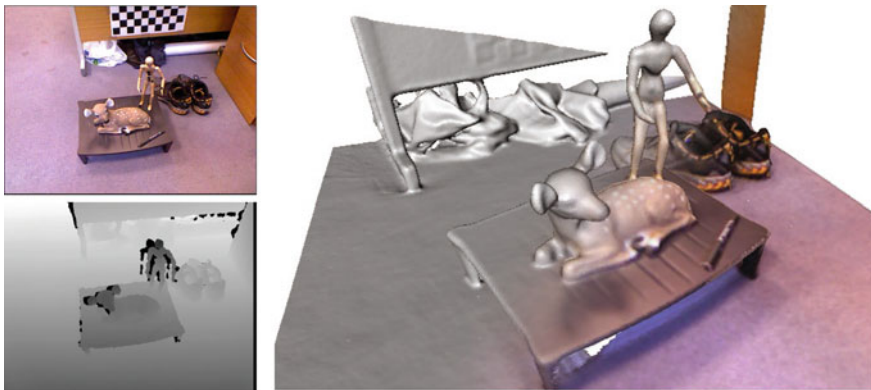
nation and ghosting caused by any inaccuracies in the model or error in the intrinsic calibration and estimated pose of the camera. To avoid excessive ghosting of texture, a vertex colour is no longer updated once its colour weighting exceeds a threshold.

The mesh output by marching cubes is inherently densely sampled, with no triangles bigger than the voxel diagonal, even for flat, low curvature surface regions. This can lead to unnecessarily large mesh files which are inefficient to store, render and manipulate. Some regions with fine features or high curvature do benefit from having small triangles, however flatter surfaces can be decimated without losing any significant detail. The scan of a statue shown in Fig. 5.3 was produced using voxels of side 4.1 mm yielding a raw output mesh with 830k vertices. Using quadric edge collapse decimation [17], the mesh can be reduced to 35k vertices, with little loss of geometric detail, but resulting in loss of detail in the per-vertex colour due to lower resolution sampling. Using [69], a dense texture map may be generated, allowing the decimated mesh to represent the detailed appearance of the captured images while representing the shape more efficiently. Two further examples of scenes reconstructed with KinectFusion are shown in Fig. 5.4.

Due to the nature of typical sensors' depth map generation process (see Chap. 1), the size of reconstructed features is inherently limited. The depth maps produced typically also suffer from significant levels of quantization and random noise causing several mm of error in depth at typical indoor scene distances [34]. However, the fusion of hundreds of frames from slightly different viewpoints integrates this noise away to produce surface relief with submillimetre resolution, even when using TSDF voxels larger than 1 mm [39].



(a) Office scene. Three input depth frames (left) and meshed output model (right).



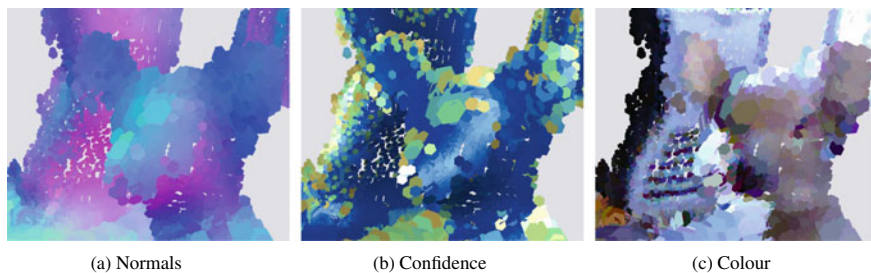
(b) Deer scene. Input RGB and depth frame (left) and reconstruction with/without texture (right).


**Fig. 5.4** Sample input and volumetric reconstructions for two indoor scenes using the KinectFusion approach [44]. Figure from [37], using their reimplement of KinectFusion

### 5.3.5 Online Surfel-Based Fusion

Keller et al. [33] propose an alternative approach to GPU-accelerated online rigid scene reconstruction similar in many respects to KinectFusion, but using a surface element (surfel) representation rather than a volumetric grid. Keller's method also contains a simple mechanism for detecting and removing dynamic outliers from the fused representation.

The representation used for fusion and the output model is a flat (unstructured) array of surface elements (surfels). Each surfel primitive consists of a position  $\mathbf{p}$ , normal  $\mathbf{n}$ , radius  $r$ , confidence  $c$ , timestamp  $t$  (last frame observed), and colour. A



**Fig. 5.5** Close up of surfel representation for 3D scene reconstruction from RGB-D video input. The representation consists of an unstructured set of points with additional properties including normal, radius, confidence, and colour. The points are rendered using hexagons. The confidence value is visualized using the following colour scale: 0  30. Figure from [37], using their reimplementation of the approach of Keller et al. [33]

closeup of the surfel primitives is shown in Fig. 5.5 which illustrates these properties. The density of modelled surfels corresponds directly to the input depth sample density. Correspondences between incoming depth measurements and the surfel IDs are established by projection into a super-sampled lookup image for the current depth frame. The fusion technique is similar to TSDF fusion, in that it is based on a running weighted average of observations. As with KinectFusion, ICP registration is performed using a depth map synthesized from the current model. In this case, the depth map is synthesized by splat rendering the surfels using a graphics shader which outputs an oriented hexagon for each surfel (as opposed to the ray-casting used in the volumetric approach).

Upon commencement of reconstruction, new surface points are assigned zero confidence. As the corresponding surface point is re-observed in incoming frames, its position and normal are averaged with the incoming measurements and the confidence value increased. Surfels with a confidence below a threshold value  $c_{stable}$  are referred to as ‘unstable’. The unstable points are excluded from the registration stage, so as to increase robustness to outliers and any dynamic elements in the scene.

An additional feature of the method of Keller et al. [33] is a simple online labelling of dynamic elements in the scene. This is performed by detecting model surfels which are outliers with respect to the current input frame, performing a region growing operation to expand the detected dynamic region, and demoting all these surfels to unstable status (with a weight just below the stability threshold  $c_{stable}$ ). The demotion to unstable status aims to prevent dynamic objects from contaminating the (static scene) model by immediately removing them from the registration point set, but allows the points to be reintroduced, if the object stops moving (and the points attain stable status once again).

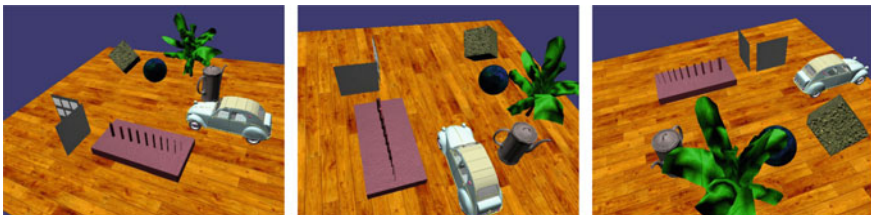


### 5.3.6 Evaluation

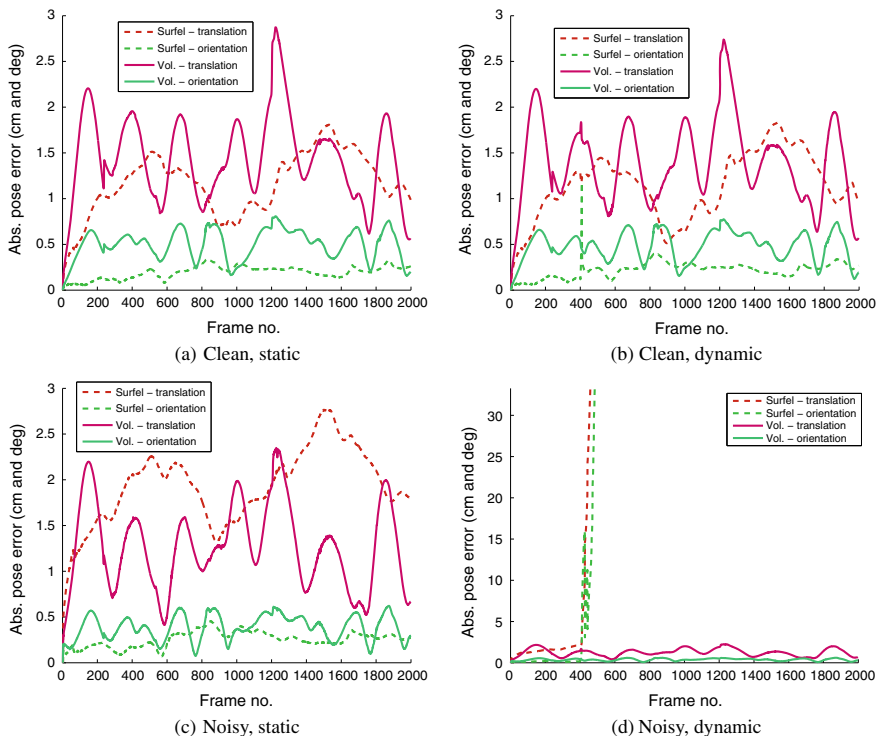
A 2000 frame synthetic dataset, *Tabletop*, was created by Malleson [37] to compare the performance of the volumetric [42] and surfel fusion [33] in terms of reconstructed geometry fidelity, camera pose estimation accuracy and functioning in the presence of dynamic content in the scene. The test scene, a tabletop 1.8 m across with various objects on it, is shown in Fig. 5.6. It contains thin and narrow structures as well as a toy car that moves across the table. The virtual RGB-D camera moves smoothly, around the scene as well as up and down.

The registration performance of volumetric and surfel-based fusion were evaluated for four variants of the *Tabletop* scene. The variants are the combinations of clean depth/simulated Kinect noisy depth (with noise simulated using [34]), and static geometry only/with the moving car. The absolute error in position and orientation with respect to ground truth is plotted in Fig. 5.7. Both the volumetric and surfel-based approaches produce good tracking on the clean data, with or without the moving car present. On the moving car variant the surfel-based registration fails when the car starts moving (at frame 400). A summary of the registration and reconstruction performance for each variant is presented in Table 5.1.

To directly compare the quality of the surface reconstruction between the volumetric and the surfel-based methods, tests were performed in which the two systems were fed the ground truth camera pose trajectories, with the version of the scene not containing the moving car. This factors out any differences resulting from error in ICP registration and handling of dynamic geometry. To make the reconstructed geometry of similar density between the methods, the voxel size for the volumetric tests was chosen such as to produce roughly the same number of vertices as surfels are produced by the surfel-based tests, roughly 1.5M in each case. Figure 5.8 visualizes the Hausdorff distance between the ground truth geometry and the reconstructions with each method. The RMS error for the surfel fusion method is 1.1 mm, compared to 3.5 mm for the volumetric method. Qualitatively, the surfel fusion approach does not suffer from ‘lipping’ artefacts in reconstructions from TSDF fusion—the square edges of the cube are more faithfully reproduced. The TSDF fusion process also cannot handle very thin objects viewed from both sides because the opposing



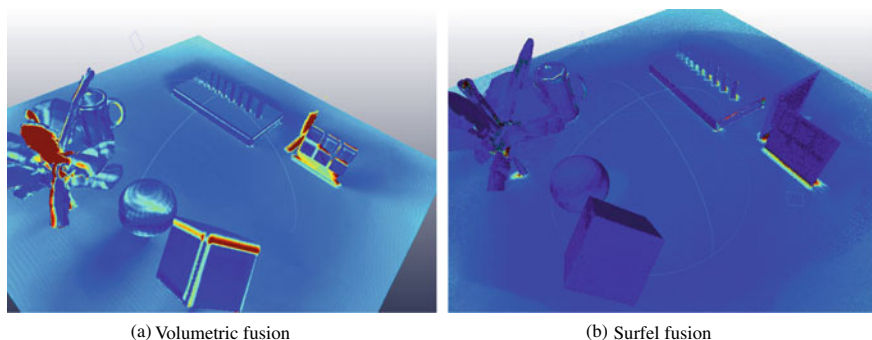
**Fig. 5.6** Three frames from the RGB-D rendering of the 2000 frame *Tabletop* test scene for assessing the performance of surfel-based fusion and volumetric fusion. Note the narrow and thin objects as well as the moving car (which is static for the first 400 frames of the sequence). Figure from [37]




**Fig. 5.7** Comparison of volumetric and surfel-based ICP registration performance on the four variants of the *Tabletop* sequence. **a** Clean depth, no moving objects. **b** Clean depth, with moving car. **c** Noisy depth, no moving objects. **d** Noisy depth, with moving car (note that the car moves from frame 400 onwards, which at which point the ICP loses track in this variant). Figure from [37]

**Table 5.1** Registration and geometric accuracy for the *Tabletop* scene, using volumetric (Vol.) and surfel-based (Surf.) reconstruction approaches. The variants are as follows: fixed geometry only (F) with moving car (M); ground truth (GT)/ICP (ICP) camera registration; and clean (C)/noisy (N) depth maps

Variant			RMS position error (mm)		RMS orientation error (deg)		RMS recon error (mm)		Num model elements ( $\times 10^6$ )	
			Vol.	Surf.	Vol.	Surf.	Vol.	Surf.	Vol.	Surf.
F	GT	C	–	–	–	–	3.47	1.08	1.14	1.55
F	GT	N	–	–	–	–	3.73	1.04	1.07	1.58
F	ICP	C	9.54	12.17	0.27	0.21	4.09	3.12	1.20	1.59
F	ICP	N	8.05	19.77	0.20	0.27	4.18	3.28	1.20	1.67
M	GT	C	–	–	–	–	3.56	1.08	1.08	1.65
M	GT	N	–	–	–	–	3.78	1.04	1.11	1.67
M	ICP	C	8.87	11.99	0.23	0.23	4.19	3.76	1.23	1.68
M	ICP	N	7.69	2139.50	0.19	138.83	4.21	(failed)	–	–

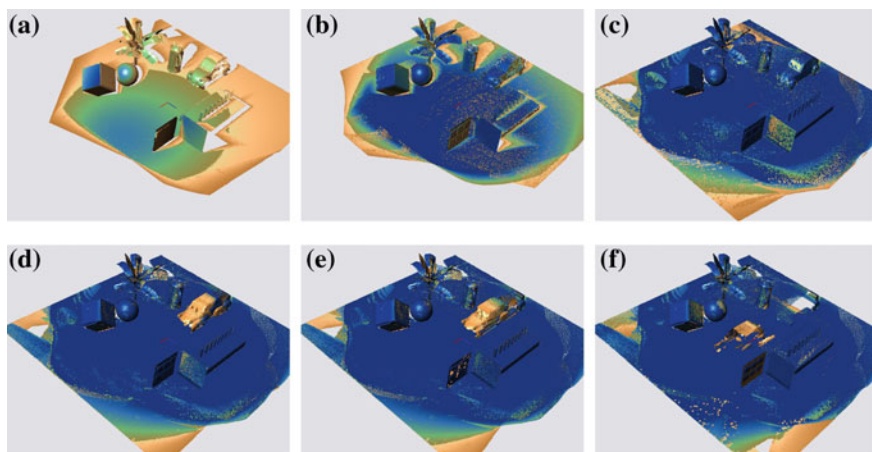



**Fig. 5.8** Hausdorff distance (0  5mm) between ground truth and reconstructed geometry for the two fusion approaches (using ground truth camera pose, no scene motion and clean depth maps). Note that the volumetric approach has lost the narrow and thin structures and that it exhibits lipping artefacts on corners of the cube. Figure from [37]

surfaces can ‘cancel each other out’ leading to artefacts. The surfel fusion method can handle thin surfaces without such artefacts, even for the zero-thickness sheet shown in the upper right of the scene in Fig. 5.8. The surfel method is also able to resolve the second smallest cylinder, which is not resolved by the volumetric method, since it is the same diameter as the size of a voxel (2.2 mm). (Neither of the methods can resolve the smallest cylinder which is a single pixel thick in some input depth frames, and not visible at all in others.)

The method proposed by Keller et al. [33] for segmenting out dynamic regions of the model is based on detecting inconsistencies between the incoming depth and the surfel model. The labelling is based on the value of the confidence field of the surfels, which begins at zero and increases as observations are added. This confidence field is analogous to the weight in the signed distance fusion in volumetric reconstruction. The progression of fusion is shown in Fig. 5.9, which shows surfel confidence via a colour coding. Surfels with confidence below a threshold are labelled as ‘unstable’. Unstable points are excluded from the ICP registration. A ‘dynamics map’ is seeded with all registration outliers and a region growing approach based on position and normal similarity between neighbouring points is applied. Modelled points in the model marked in the dynamics map are demoted to unstable status. The region growing method used is fairly simplistic and does not work robustly in all scenarios. For example, as new model points are added at the edge of the frame (e.g. a floor or tabletop) as the camera pans, they will initially be unstable, and thus have no ICP correspondence, the dynamics depth map points in this region could then be expanded by the region growing to cover a large static area (e.g. the rest of the surface of the desk). In the test example of the *Tabletop* scene, the segmentation approach is not able to prevent the model from being corrupted when the car begins to move.

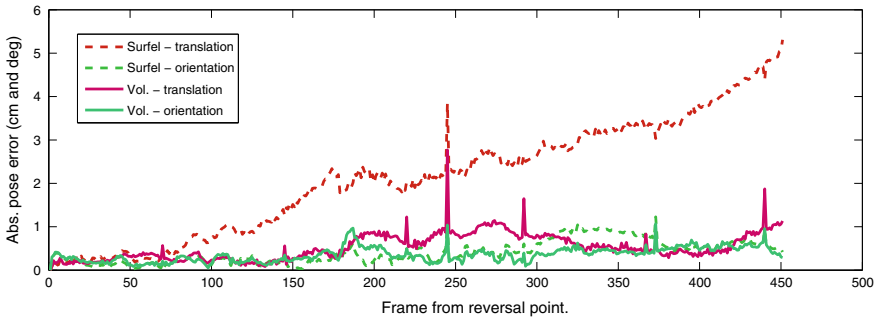
To evaluate drift on real-world data, a 900 frame time-mirrored sequence (450 frames played forward and then in reverse) was generated from the *Office* capture [37]. The difference between the estimated pose for the first and last frame of this



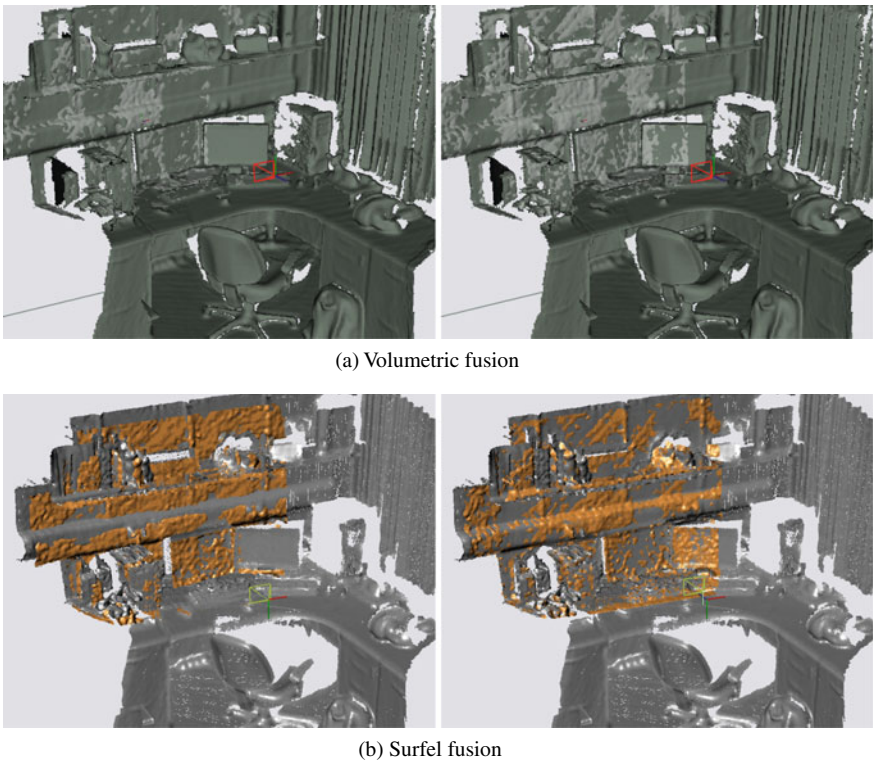
**Fig. 5.9** Splat rendering showing progression of surfel-based fusion of the *Tabletop* sequence. The confidence value is visualized using the following colour scale: 0  30, where the black line is the stability threshold. In **a–c** the car is stationary. In **d–f** the car is moving. Note the demotion of points on the car to unstable when it starts moving as well as the low confidence of new surfels on the moving car, each of which are not consistently observed for long enough to achieve stable status. Figure from [37]

sequence (which should be identical) gives an indication of the global pose estimation stability. The magnitude of the difference in estimated camera position and orientation at the start and end of the sequence were evaluated for both volumetric and surfel-based reconstruction methods and the results are shown in Fig. 5.10. Note that the surfel-based method proves less robust on this real data, with an accumulated drift of 5 cm compared to 1 cm for the volumetric method. The effect of this is demonstrated in Fig. 5.11 which shows the final reconstructed models and posed depth map for the first and last frames. The camera offset from the origin can be seen in the *last* frame, particularly for the surfel-based method. The gradual accumulation of drift in pose goes hand in hand with accumulated drift in the model. Therefore, the depth map in the *last* frame is consistent with the depth map, which means that the depth map and model are inconsistent with one another at the *first* frame. This mismatch is larger for the surfel-based method on account of the greater level of drift. The surfel-based reconstructed surface is also less complete than the volumetric surface, since some surface regions are only briefly observed in the input, meaning that they are treated as outliers by the fusion algorithm.

The flat array of surfels used in the surfel fusion approach has a memory footprint proportional to reconstructed surface area, whereas that of the fully allocated volumetric grids of KinectFusion is proportional to scene volume (regardless of occupied surface area). For a roughly equal sampling density, and typical scene content, the surfel representation is far more compact. For the example scene presented here, the 2 m<sup>2</sup> voxel grid containing 180M voxels requires 1.4 GB to store, compared to just



**Fig. 5.10** Difference in pose between corresponding frames in time-mirrored *Office* sequence as reconstructed using the volumetric and surfel-based approaches. Note the relatively large error for the surfel-based approach. Figure from [37]



**Fig. 5.11** First (left) and last (right) frames in the time-mirrored *Office* sequence using **a** volumetric and **b** surfel-based reconstruction. The input depth map is shown projected into the reconstructed model. For the surfel-based reconstruction, note the significant pose offset from identity in the right-hand frame and misalignment of depth and model in the left-hand frame. Figure from [37]

72 MB for the comparably detailed 1.5 M surfel array (assuming 4 byte data types are used throughout).

Note that the density of the surfels is directly set according to the local input sample density, and it is not necessary to define a limited spacial extent for the reconstruction up front as with a voxel grid.

One of the most prominent reconstruction artefacts manifested by the volumetric method is ‘lipping’ at sharp corners (which results from the projective approximation to the true signed distance function [32]). This is particularly noticeable at the edges of the cube in Fig. 5.8. The surfel-based approach does not suffer from this type of artefact, thus given clean data and the simulated condition of ground truth camera pose trajectories and clean depth, it produces cleaner geometry. However under real-world conditions, i.e. using noisy Kinect depth and ICP for camera pose estimation, registration and reconstruction were found to be more robust using the volumetric fusion representation. This may be due to specific implementation details (e.g. rounding behaviour), or perhaps qualitative differences in depth maps from ray-casting versus hexagonal splat rendering.

## 5.4 Dynamic Scene Reconstruction

In the case of static scene reconstruction, surface registration is equivalent to finding the 6-DoF camera pose for each frame and a simple fixed TSDF voxel grid is sufficient for measurement fusion (see Sect. 5.3). The core aspects of both static and dynamic scene reconstruction are surface registration and surface fusion. Both these aspects are, however, more challenging in the case of dynamic scenes, which may contain multiple rigid, articulated, and non-rigidly deforming surfaces that need to be tracked and consistently integrated into a surface model. This section provides a brief summary of recent techniques for registration and fusion for dynamic scenes.

Multiple-view video has traditionally been used to capture full coverage of dynamic 3D scenes for reconstruction (e.g. [6, 19, 64]). While high quality models can be obtained from them, adoption of multiview video reconstruction systems has been limited by the cost and complexity of operation of multi-camera setups. On the other hand, non-rigid structure from motion (NRSfM) approaches (e.g. [31, 35, 51, 56]) attempt to recover dynamic 3D shape and motion from a sequence of images from a single, monocular RGB camera, making them usable with standard video cameras and existing video footage. NRSfM is, however, a highly challenging, under-constrained problem, since absolute depth is not known beforehand. Although depth maps from commodity sensors tend to be noisy and incomplete, with a lower resolution than current video cameras, their depth estimates are more robust than those estimated from RGB images alone, particularly in low-textured or repetitively textured regions. The availability of a reliable estimate of per-pixel depth for each frame simplifies the reconstruction problem, however surface registration and temporally consistent fusion of dynamic scenes remains a challenging problem.

Depth maps are natively output by typical commodity RGB-D sensors (e.g. Microsoft Kinect v1/v2) and cover only the surface seen from a specific camera view. Certain low-level processing tasks can be performed using the depth maps directly, such as bilateral filtering [67], motion-compensated RGB-guided upsampling [52], depth-guided matting [70], and depth-aware video compositing. Tasks such as general dynamic scene editing can, however benefit from more complete 3D geometry preferably with ‘4D’ temporal consistency, i.e. 3D surfaces which have known correspondences over time, which allows edits to appearance, shape and motion to be automatically propagated over a sequence (see [4, 5, 27]). In applications where a template scan of a non-rigid object of interest is able to be obtained beforehand (e.g. using a static reconstruction approach without the object deforming), this template model may be dynamically deformed to match RGB-D input of the object in a scene by using volumetric representations, either offline (e.g. [20]) or in real-time (e.g. [79]).

A core challenge in temporally consistent modelling is obtaining correspondences of surface points over time. Analogous to 2D *optical flow* between two RGB images (e.g. [65]), RGB-D *scene flow* estimates a per-pixel 3D translation (e.g. [14, 30, 71]) or translation and rotation (e.g. [26, 72]) between two RGB-D images. Frame-to-frame flow vectors can be propagated over time to form long-term feature tracks [65], which may use as an input to RGB-D-based dynamic scene modelling [38].

Surface meshes explicitly store oriented surfaces and are widely used in the manipulation of models in 3D graphics applications and media production. However, as is the case with static scene reconstruction approaches, intermediate representations such as volumetric and point-based, are often used to facilitate surface fusion. Fusion of non-rigid geometry using signed distance functions may be achieved, for instance, using a piecewise-rigid segmentation [41] or a warping field defined over a single reference volume [21, 43].

In DynamicFusion [43], Newcombe et al. perform real-time online tracking and reconstruction of dynamic objects from depth sensors without a template. Their approach is to warp each input frame back to a canonical frame using a per-frame volumetric warping field, and then perform TSDF fusion in this frame. For efficiency, only sparse warping field samples are estimated, and dense values are inferred by interpolation. The TSDF fusion weights take into account the confidence in the warping field, which decreases with distance from the warping field samples. The warping field is estimated by optimizing an energy consisting of an ICP data term and a regularization term that encourages smooth variation of the warping function (where the transformation nodes are connected with edges in a hierarchical deformation graph).

Similar to DynamicFusion, Innmann et al. [28] propose VolumeDeform, which incorporates sparse image features from the RGB images as well as dense depth constraints, which help in correct registration of scenes with low geometric variation.

In their Fusion4D approach, Dou et al. [12] perform online reconstruction from multiple depth sensors for improved scene coverage. Slavcheva et al. [62] propose KillingFusion, which performs real-time, non-rigid reconstruction using TSDF fusion without computing explicit point correspondences, instead directly optimizing

a warping field between TSDFs. Because point correspondences are not computed, however, it does not support applications which require texture mapping (e.g. appearance editing).

In [40], a method for reconstruction of dynamic scenes from single-view RGB-D data based on a sparse set of temporally coherent surfels (tracked 3D points) which are explicitly connected using neighbourhood-based connectivity is proposed: simultaneous segmentation, shape and motion estimation of arbitrary scenes is performed without prior knowledge of the shape or non-rigid deformation of the scene. This surfel graph modelling is, however, limited in terms of the shape detail reproduced, and does not natively output a surface mesh. As a result, a subsequent dense surface reconstruction stage is required in order to obtain a detailed surface mesh. In their ‘animation cartography’ approach, Tevs et al. [66] employ surface ‘charts’ with shared, tracked landmarks in multiple graph structures. Probabilistic sparse matching is performed on the landmarks, and dense correspondence is then established for the remaining chart points by comparing landmark coordinates. They note that their system does not perform well on very noisy time-of-flight depth data and suggest using additional cues (e.g. colour) for such data.

A hybrid method for fusion and representation of dynamic scenes from RGB-D video has been proposed [38] which uses the complementary strengths of multiple representations at different stages of processing. *Depth maps* provide input 2.5D geometry and are used along with the corresponding RGB images to generate a graph of *sparse point tracks* for dense *volumetric* surface integration, while *residual depth maps* store differences between the final output 4D model and raw input. The intermediate *surfel graph* structure stores sparse, dynamic 3D geometry with neighbourhood-based connectivity, and is used for efficient segmentation and initial reconstruction of part shape and motion. The surfel graph representation drives a further intermediate TSDF *volumetric implicit surface* representation, which is used to integrate noisy input depth measurements into dense piecewise and global 3D geometry. The volumetric representation is finally extracted to an explicit, dense surface *mesh* suitable for dynamic scene rendering, as well as editing of shape, appearance and motion.

## 5.5 Conclusion

In this chapter, an overview of techniques for reconstruction from RGB-D input was presented and further detail provided on two approaches to real-time static scene reconstruction, namely KinectFusion [44] and surfel fusion [33]. Such volumetric and surfel-based reconstruction approaches are able to register and integrate hundreds or thousands of noisy depth maps in an online manner and produce metrically consistent models of static scenes with greater coverage and less noise than the individual input depth maps.

The frame-to-model ICP tracking approach proposed by Newcombe et al. [44] mitigates accumulation of error, which would be more severe with frame-to-frame



tracking and thus helps maintain the level of detail in the reconstructed models. Assuming adequately small voxels (of the order of the depth pixel size), the main limiting factor in reconstruction resolution is the image (domain) resolution rather than the noise and quantization of depth values (range), which can be integrated away over time as frames are added. (The case is similar for the surfel-based representation, where the model resolution corresponds directly to the input sample density, rather than depending on a separately specified voxel size.) Higher quality, larger scale reconstructions can be achieved using offline reconstruction approaches such as that of Zhou et al. [76], which employs global optimization of the sensor pose and scene geometry.

Static scene reconstruction from RGB-D input is a well-developed field and current approaches are able to produce high quality results in real-time. Temporally consistent reconstruction of general dynamic scenes from RGB-D is a challenging open problem, however the field is fast moving and recent approaches such as DynamicFusion [43] and KillingFusion [62] have made significant progress towards reconstruction of dynamic, non-rigidly deforming objects through use of deforming volumetric representations for surface integration and tracking.

## References

1. Bernardini F, Mittleman J (1999) The ball-pivoting algorithm for surface reconstruction. *Trans Vis Comput Graph (TVCG)*
2. Besl P, McKay N (1992) A method for registration of 3-D shapes. *Trans Pattern Anal Mach Intell (PAMI)* 14(2):239–256. <https://doi.org/10.1109/34.121791>
3. Blais G, Levine M (1995) Registering multiview range data to create 3D computer objects. *Trans Pattern Anal Mach Intell (PAMI)* 17(8):820–824. <https://doi.org/10.1109/34.400574>
4. Budd C, Huang P, Hilton A (2011) Hierarchical shape matching for temporally consistent 3D video. In: *3D imaging, modeling, processing, visualization and transmission (3DIMPVT)*, pp 172–179. <https://doi.org/10.1109/3DIMPVT.2011.29>
5. Budd C, Huang P, Klaudiny M, Hilton A (2013) Global non-rigid alignment of surface sequences. *Int J Comput Vis (IJCV)* 102(1–3):256–270. <https://doi.org/10.1007/s11263-012-0553-4>
6. Cagniard C, Boyer E, Ilic S (2010) Free-form mesh tracking: a patch-based approach. In: *Computer vision and pattern recognition (CVPR)*, pp 1339–1346
7. Chang W, Zwicker M (2011) Global registration of dynamic range scans for articulated model reconstruction. *ACM Trans Graph (TOG)* 30
8. Chen J, Bautembach D, Izadi S (2013) Scalable real-time volumetric surface reconstruction. *ACM Trans Graph (TOG)*
9. Chen Y, Medioni G (1991) Object modeling by registration of multiple range images. In: *International conference on robotics and automation*, vol 3, pp 2724–2729. <https://doi.org/10.1109/ROBOT.1991.132043>
10. Curless B, Levoy M (1996) A volumetric method for building complex models from range images. In: *Conference on computer graphics and interactive techniques*, pp 303–312
11. Davison AJ (2003) Real-time simultaneous localisation and mapping with a single camera. In: *International conference on computer vision (ICCV)*, vol 2, pp 1403–1410. <https://doi.org/10.1109/ICCV.2003.1238654>
12. Dou M, Khamis S, Degtyarev Y, Davidson P, Fanello SR, Kowdle A, Escolano SO, Rhemann C, Kim D, Taylor J, Kohli P, Tankovich V, Izadi S (2016) Fusion4d: real-time performance

- capture of challenging scenes. *ACM Trans Graph (TOG)*. <https://doi.org/10.1145/2897824.2925969>
13. Fechteler P, Eisert P (2011) Recovering articulated pose of 3D point clouds. In: European conference on visual media production (CVMP), p 2011
  14. Ferstl D, Riegler G, Rüether M, Bischof H (2014) CP-census: a novel model for dense variational scene flow from RGB-D data. In: British machine vision conference (BMVC), pp 18.1–18.11. <https://doi.org/10.5244/C.28.18>
  15. Fitzgibbon A (2003) Robust registration of 2D and 3D point sets. *Image Vis. Comput.* 21(13–14):1145–1153. <https://doi.org/10.1016/j.imavis.2003.09.004>
  16. Fuhrmann S, Goesele M (2014) Floating scale surface reconstruction. In: ASM SIGGRAPH
  17. Garland M, Heckbert P (1998) Simplifying surfaces with color and texture using quadric error metrics. In: Conference on visualization. <https://doi.org/10.1109/VISUAL.1998.745312>
  18. Godard C, Mac Aodha O, Brostow GJ (2017) Unsupervised monocular depth estimation with left-right consistency. In: CVPR
  19. Guillemaut JY, Hilton A (2010) Joint multi-layer segmentation and reconstruction for free-viewpoint video applications. *Int J Comput Vis (IJCV)* 93(1):73–100. <https://doi.org/10.1007/s11263-010-0413-z>
  20. Guo K, Xu F, Wang Y, Liu Y, Dai Q (2015) Robust non-rigid motion tracking and surface reconstruction using L0 regularization. In: International conference on computer vision (ICCV), vol 1. IEEE, pp 3083–3091. <https://doi.org/10.1109/ICCV.2015.353>
  21. Guo K, Xu F, Yu T, Liu X, Dai Q, Liu Y (2017) Real-time geometry, albedo and motion reconstruction using a single RGBD camera. *ACM Trans Graph (TOG)*
  22. Hernandez M, Choi J, Medioni G (2015) Near laser-scan quality 3-D face reconstruction from a low-quality depth stream. *Image Vis Comput* 36:61–69. <https://doi.org/10.1016/j.imavis.2014.12.004>
  23. Hilton A, Stoddart A, Illingworth J, Windeatt T (1998) Implicit surface-based geometric fusion. *Comput Vis Image Underst* 69(3):273–291. <https://doi.org/10.1006/cviu.1998.0664>
  24. Hilton A, Stoddart AJ, Illingworth J, Windeatt T (1996) Marching triangles: range image fusion for complex object modelling. In: International conference on image processing, vol 2, pp 381–384. <https://doi.org/10.1109/ICIP.1996.560840>
  25. Hoppe H, DeRose T, Duchamp T, McDonald JA, Stuetzle W (1992) Surface reconstruction from unorganized points. In: ACM SIGGRAPH, pp 71–78
  26. Hornáček M, Fitzgibbon A, Rother C (2014) SphereFlow: 6 DoF scene flow from RGB-D pairs. In: Computer Vision and Pattern Recognition (CVPR), pp 3526–3533 (2014). <https://doi.org/10.1109/CVPR.2014.451>
  27. Huang P, Budd C, Hilton A (2011) Global temporal registration of multiple non-rigid surface sequences. In: Computer vision and pattern recognition (CVPR), pp 3473–3480. <https://doi.org/10.1109/CVPR.2011.5995438>
  28. Innmann M, Zollhöfer M, Nießner M, Theobalt C, Stamminger M (2016) VolumeDeform: real-time volumetric non-rigid reconstruction. In: European conference on computer vision (ECCV), pp 362–379
  29. Izadi S, Kim D, Hilliges O, Molyneaux D, Newcombe R, Kohli P, Shotton J, Hodges S, Freeman D, Davison A (2011) Others: KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In: ACM symposium on user interface software and technology (UIST), pp. 559–568
  30. Jaimez M, Souiai M, Gonzalez-Jimenez J, Cremers D (2015) A primal-dual framework for real-time dense RGB-D scene flow. In: International conference on robotics and automation (ICRA), vol 2015-June, pp 98–104. <https://doi.org/10.1109/ICRA.2015.7138986>
  31. Ji P, Li H, Dai Y, Reid I (2017) ‘Maximizing Rigidity’ revisited: a convex programming approach for generic 3D shape reconstruction from multiple perspective views. In: International conference on computer vision (ICCV), pp 929–937. <https://doi.org/10.1109/ICCV.2017.106>
  32. Kazhdan M, Bolitho M, Hoppe H (2006) Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing, pp 61–70

33. Keller M, Lefloch D, Lambers M, Izadi S, Weyrich T, Kolb A (2013) Real-time 3D reconstruction in dynamic scenes using point-based fusion. In: International conference on 3D vision (3DV)
34. Khoshelham K, Elberink SO (2012) Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors* 12(2):1437–1454. <https://doi.org/10.3390/s120201437>
35. Kumar S, Dai Y, Li H (2017) Monocular dense 3D reconstruction of a complex dynamic scene from two perspective frames. In: International conference on computer vision (ICCV), pp 4659–4667. <https://doi.org/10.1109/ICCV.2017.498>
36. Lorensen W, Cline H (1987) Marching cubes: a high resolution 3D surface construction algorithm. In: ACM SIGGRAPH, vol 21. ACM, pp 163–169
37. Malleson C (2015) Dynamic scene modelling and representation from video and depth. PhD thesis, CVSSP, University of Surrey, Guildford, GU2 7XH, UK (2015)
38. Malleson C, Guillemaut J, Hilton A (2018) Hybrid modelling of non-rigid scenes from RGBD cameras. *IEEE Trans Circuits Syst Video Technol* 1–1. <https://doi.org/10.1109/TCSVT.2018.2863027>
39. Malleson C, Hilton A, Guillemaut JY (2012) Evaluation of kinect fusion for set modelling. In: European conference on visual media production (CVMP) (2012)
40. Malleson C, Klaudiny M, Guillemaut JY, Hilton A (2014) Structured representation of non-rigid surfaces from single view 3D point tracks. In: International conference on 3D vision (3DV), pp 625–632
41. Malleson C, Klaudiny M, Hilton A, Guillemaut JY (2013) Single-view RGBD-based reconstruction of dynamic human geometry. In: International conference on computer vision (ICCV) workshops, pp 307–314
42. Newcombe R, Davison A (2010) Live dense reconstruction with a single moving camera. In: Computer vision and pattern recognition (CVPR), pp 1498–1505. <https://doi.org/10.1109/CVPR.2010.5539794>
43. Newcombe R, Fox D, Seitz S (2015) DynamicFusion: reconstruction and tracking of non-rigid scenes in real-time. In: Computer vision and pattern recognition (CVPR)
44. Newcombe R, Izadi S, Hilliges O, Molyneaux D, Kim D, Davison A, Kohli P, Shotton J, Hodges S, Fitzgibbon A (2011) KinectFusion: real-time dense surface mapping and tracking. In: International symposium on mixed and augmented reality (ISMAR), pp 127–136
45. Newcombe R, Lovegrove S, Davison A (2011) DTAM: Dense tracking and mapping in real-time. In: International conference on computer vision (ICCV), pp 2320–2327 (2011)
46. Nießner M, Zollhöfer M, Izadi S, Stamminger M (2013) Real-time 3D reconstruction at scale using voxel hashing. *ACM Trans Graph* 32(6), 169:1—169:11 (2013). <https://doi.org/10.1145/2508363.2508374>
47. NVidia: CUDA. <http://www.nvidia.com/cuda> (2012). Accessed Feb 2012
48. Ohtake Y, Belyaev A, Seidel HP (2003) A multi-scale approach to 3D scattered data interpolation with compactly supported basis functions. In: Shape modeling international, SMI '03, pp 153–161
49. Parker S, Shirley P, Livnat Y, Hansen C, Sloan PP (1998) Interactive ray tracing for isosurface rendering. In: Conference on visualization, pp 233–238. <https://doi.org/10.1109/VISUAL.1998.745713>
50. Pellegrini S, Schindler K, Nardi D (2008) A generalisation of the ICP algorithm for articulated bodies. In: British machine vision conference (BMVC), Lm
51. Ranftl R, Vineet V, Chen Q, Koltun V (2016) Dense monocular depth estimation in complex dynamic scenes. In: Conference on computer vision and pattern recognition (CVPR), pp 4058–4066. <https://doi.org/10.1109/CVPR.2016.440>
52. Richardt C, Stoll C (2012) Coherent spatiotemporal filtering, upsampling and rendering of RGBZ videos. *Comput Grap Forum* 31(2)
53. Roth H, Marsette V (2012) Moving volume kinectfusion. In: British Machine Vision Conference (BMVC), pp 112.1—112.11. <https://doi.org/10.5244/C.26.112>
54. Rusinkiewicz S, Hall-Holt O, Levoy M (2002) Real-time 3D model acquisition. *ACM Trans Graph (TOG)* 21(3) (2002). <https://doi.org/10.1145/566654.566600>

55. Rusinkiewicz S, Levoy M (2001) Efficient variants of the ICP algorithm. In: 3-D digital imaging and modeling, pp 145–152. IEEE (2001)
56. Russell C, Yu R, Agapito L (2014) Video pop-up: monocular 3D reconstruction of dynamic scenes. In: European conference on computer vision (ECCV), pp 583–598 (2014)
57. Rutishauser M, Stricker M, Trobina M (1994) Merging range images of arbitrarily shaped objects. In: Computer vision and pattern recognition (CVPR), pp 573–580 (1994). <https://doi.org/10.1109/CVPR.1994.323797>
58. Salas-Moreno R, Glocker B, Kelly P, Davison A (2014) Dense planar SLAM. In: International symposium on mixed and augmented reality (ISMAR)
59. Salas-Moreno RF, Newcombe RA, Strasdat H, Kelly PHJ, Davison AJ (2013) SLAM++: simultaneous localisation and mapping at the level of objects. In: Computer Vision and Pattern Recognition (CVPR), pp 1352–1359 (2013). <https://doi.org/10.1109/CVPR.2013.178>
60. Salvi J, Matabosch C, Fofi D, Forest J (2007) A review of recent range image registration methods with accuracy evaluation. *Image Vis Comput* 25(5):578–596. <https://doi.org/10.1016/j.imavis.2006.05.012>
61. Segal A, Haehnel D, Thrun S (2009) Generalized-ICP. In: Robotics, science and systems (2009)
62. Slavcheva M, Baust M, Cremers D, Ilic S (2017) KillingFusion: non-rigid 3D reconstruction without correspondences. In: Conference on computer vision and pattern recognition (CVPR), pp 5474–5483. <https://doi.org/10.1109/CVPR.2017.581>
63. Soucy M, Laurendeau D (1995) A general surface approach to the integration of a set of range views. *Trans Pattern Anal Mach Intell (PAMI)* 17(4):344–358. <https://doi.org/10.1109/34.385982>
64. Starck J, Hilton A (2007) Surface capture for performance-based animation. *Comput Graph Appl* 27(3):21–31
65. Sundaram N, Brox T, Keutzer K (2010) Dense point trajectories by GPU-accelerated large displacement optical flow. In: European conference on computer vision (ECCV)
66. Tevs A, Berner A, Wand M, Ihrke I, Bokeloh M, Kerber J, Seidel HP (2011) Animation cartography—intrinsic reconstruction of shape and motion. *ACM Trans Graph (TOG)* (2011)
67. Tomasi C, Manduchi R (1998) Bilateral filtering for gray and color images. In: International conference on computer vision (ICCV), pp 839–846
68. Turk G, Levoy M (1994) Zipped polygon meshes from range images. In: ACM SIGGRAPH, pp 311–318. <https://doi.org/10.1145/192161.192241>
69. Volino M, Hilton A (2013) Layered view-dependent texture maps. In: European conference on visual media production (CVMP)
70. Wang O, Finger J, Qingxiong Y, Davis J, Ruigang Y (2007) Automatic natural video matting with depth. In: Pacific Conference on Computer Graphics and Applications, pp 469–472 (2007). <https://doi.org/10.1109/PG.2007.11>
71. Wang P, Li W, Gao Z, Zhang Y, Tang C, Ogunbona P (2017) Scene flow to action map: a new representation for RGB-D based action recognition with convolutional neural networks. arXiv. <https://doi.org/10.1109/CVPR.2017.52>
72. Wang Y, Zhang J, Liu Z, Wu Q, Chou P, Zhang Z, Jia Y (2015) Completed dense scene flow in RGB-D space. In: Asian conference on computer vision (ACCV) workshops, vol 9009, pp 191–205 (2015). <https://doi.org/10.1007/978-3-319-16631-5>
73. Weik S (1997) Registration of 3-D partial surface models using luminance and depth information. In: International conference on recent advances in 3-D digital imaging and modeling. <https://doi.org/10.1109/IM.1997.603853>
74. Whelan T, Kaess M, Fallon M (2012) Kintinuous: spatially extended kinectfusion. RSS workshop on RGB-D: advanced reasoning with depth cameras
75. Zhang Z (1999) Flexible camera calibration by viewing a plane from unknown orientations. In: International conference on computer vision (ICCV), vol 00, pp 0–7
76. Zhou Q, Koltun V (2013) Dense scene reconstruction with points of interest. *ACM Trans. Graph. (TOG)* (2013)
77. Zhou Q, Miller S, Koltun V (2013) Elastic fragments for dense scene reconstruction. In: International conference on computer vision (ICCV), Figure 2 (2013). <https://doi.org/10.1109/ICCV.2013.65>

78. Zollhöfer M, Dai A, Innmann M, Wu C, Stamminger M, Theobalt C, Nießner M (2015) Shading-based refinement on volumetric signed distance functions. *ACM Trans Graph (TOG)* (2015)
79. Zollhöfer M, Nießner M, Izadi S, Rhemann C (2014) Real-time non-rigid reconstruction using an RGB-D camera. In: *ACM SIGGRAPH* (2014)