Hybrid modelling of non-rigid scenes from RGBD cameras

Charles Malleson *Member, IEEE,* Jean-Yves Guillemaut *Member, IEEE,* and Adrian Hilton

***Abstract*—Recent advances in sensor technology have introduced low-cost RGB video plus depth sensors, such as the Kinect, which enable simultaneous acquisition of colour and depth images at video rates. This paper introduces a framework for representation of general dynamic scenes from video plus depth acquisition. A hybrid representation is proposed which combines the advantages of prior surfel graph surface segmentation and modelling work with the higher-resolution surface reconstruction capability of volumetric fusion techniques. The contributions are (1) extension of a prior piecewise surfel graph modelling approach for improved accuracy and completeness, (2) combination of this surfel graph modelling with TSDF surface fusion to generate dense geometry, and (3) proposal of means for validation of the reconstructed 4D scene model against the input data and efficient storage of any unmodelled regions via residual depth maps. The approach allows arbitrary dynamic scenes to be efficiently represented with temporally consistent structure and enhanced levels of detail and completeness where possible, but gracefully falls back to raw measurements where no structure can be inferred. The representation is shown to facilitate creative manipulation of real scene data which would previously require more complex capture setups or manual processing.**

***Index Terms*—Dynamic scene modelling, 4D reconstruction, RGBD, video plus depth.**

I. Introduction

SINCE the introduction of low-cost RGB video plus depth (RGBD) sensors in 2010, it has become cost-effective to capture a per-pixel depth map stream concurrently with a standard video. Such sensors have been applied in a wide range of applications by the computer vision community, including in static and dynamic scene modelling.

Raw captured depth maps provide scene geometry from a single point of view, but suffer from noise and missing measurements. They are also not able to capture surface regions occluded in the current frame, and the depth samples are not *temporally consistent*, in the sense that there is no known correspondence between surface points across frames. Temporal consistency is a useful property for a dynamic scene reconstruction, as it allows any edits to the content to be propagated over time [1], [2]. An important goal in content production from RGBD data is thus to process raw RGBD sequences to produce more complete, less noisy surface representations with temporal consistency ('4D sequences'), allowing content to be efficiently stored, and its shape, appearance or motion to be edited.

The authors are with The Centre for Vision, Speech and Signal Processing, University of Surrey, GU2 7XH, United Kingdom: e-mail: {charles.malleson, j.guillemaut, a.hilton} @surrey.ac.uk.

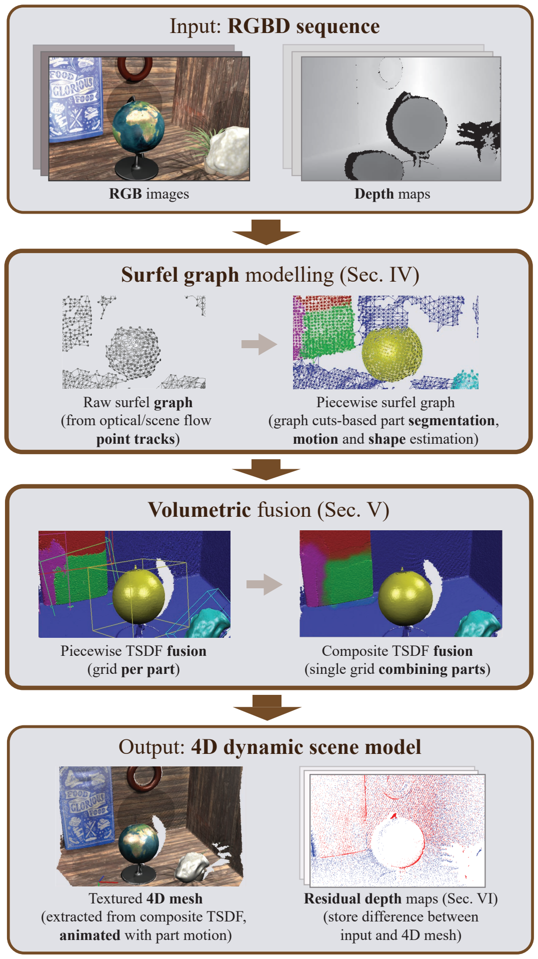

Input: RGBD sequence
RGB images
Depth maps
Surfel graph modelling (Sec. IV)
Raw surfel graph (from optical/scene flow point tracks)
Piecewise surfel graph (graph cuts-based part segmentation, motion and shape estimation)
Volumetric fusion (Sec. V)
Piecewise TSDF fusion (grid per part)
Composite TSDF fusion (single grid combining parts)
Output: 4D dynamic scene model
Textured 4D mesh (extracted from composite TSDF, animated with part motion)
Residual depth maps (Sec. VI) (store difference between input and 4D mesh)

Fig. 1. Overview of the hybrid reconstruction approach. RGBD data is processed first with surfel graph modelling, then volumetric fusion, yielding a dynamic 4D mesh sequence, together with a residual depth map to represent any unmodelled regions.

The key idea of this work is to combine surfel (surface element) graph modelling and volumetric fusion in a hybrid approach to reconstructing and representing non-rigid scenes captured from a single RGBD sensor (Fig. 1).

The surfel graph segmentation and model building approach proposed by Malleson *et al.* [3] is extended and integrated with volumetric surface fusion, allowing seamless, temporally consistent meshes to be produced. The representation can handle general scenes (multiple rigid and non-rigid objects, including the background) without requiring specific prior models or assumptions about scene content. The mesh representation is efficient because the reconstructed reference shape is stored for a single frame, while skinning weights and part motion trajectories are used to animate the mesh to produce output frames. The approach provides a structured, temporally consistent 4D scene representation, enabling subsequent manipulation and editing not possible with unstructured, noisy raw depth maps. At the same time, it offers reduced storage requirements, whilst allowing recovery of the full input depth maps to within a specified noise threshold. This is facilitated by residual depth maps, which allow the representation both of any unmodelled geometry and of any inconsistencies which may occur between the input and the modelled geometry.

The remainder of the paper is structured as follows. Section II puts the work in the context of related work on dynamic scene modelling and representation. Section III summarizes the proposed hybrid reconstruction pipeline, including the piecewise surfel graph modelling and volumetric fusion stages, which are discussed in Sections IV and, V, respectively. Model verification and residual depth maps are discussed in Section VI, and in Section VII, an extensive evaluation is presented on both real and synthetic data, including examples of edits facilitated by the representation. Finally, conclusions and future work are presented in Section VIII.

II. Related work

Multiple-view video has traditionally been used to capture full coverage of 3D scenes for reconstruction (*e.g.* [4], [5]). While high quality models can be obtained from them, adoption of multi-view 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.* [6]–[9]) 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. Our work considers modelling of non-rigid scenes from single-view RGBD cameras, which are now widely available (including on some mobile devices). RGBD sensors offer low cost and simplicity of use, and make dynamic 3D modelling considerably easier than with monocular video. Some of the problems of NRSfM, such as correspondence estimation and part/motion segmentation are in common with our application, however, the availability of a reliable estimate of per-pixel depth simplifies the problem. Although depth maps from commodity sensors tend to be noisy and incomplete [10], 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.

Depth maps are natively output by typical commodity RGBD sensors (*e.g.* Microsoft Kinect v1/v2) and cover only the surface seen from a specific camera view ('2.5D' geometry). Certain low-level processing tasks can be performed using the depth maps directly, such as bilateral filtering [11], motion-compensated RGB-guided upsampling [12], depth-guided matting [13], and 'deep' compositing[1]. Displacement maps can also be used to model certain classes of shape (*e.g.* the human head via a cylindrical parametrization [14]), but are generally not well suited to representation of arbitrary shapes. Tasks such as general dynamic scene editing require more complete 3D geometry with arbitrary topology, preferably with temporal consistency.

A core aspect of temporally consistent modelling is obtaining correspondences of surface points over time. Analogous to 2D *optical flow* between two RGB images (*e.g.* [15]), RGBD *scene flow* estimates a per-pixel 3D translation (*e.g.* [16]–[18]) or translation and rotation (*e.g.* [19], [20]) between two RGBD images. Frame-to-frame flow vectors can be propagated over time to form long-term feature tracks [15], which we use as an input to our modelling framework. We are agnostic as to the source of these point tracks, and show results both using the feature-assisted long-range optical flow of Sundaram *et al.* [15] and on the primal-dual RGBD scene flow of Jaimez *et al.* [17].

Surface meshes explicitly store oriented surfaces and are widely used in the manipulation of models in 3D graphics applications and media production. They are therefore a suitable target representation for 4D reconstruction. Although it is possible to fuse surface measurements using meshes directly [21], [22], it is relatively complex to maintain a manifold mesh connectivity; as a result, explicit mesh generation is usually only performed at the final stage of reconstruction, and 3D and 4D reconstruction approaches typically use other representations, including volumetric and point-based ones, often with a form of graph structure to aid non-rigid deformation.

Volumetric approaches (*e.g.* [23], [24]) store an implicit representation of the modelled surface, free space and unobserved space in a scene. In a truncated signed distance function (TSDF) approach, the scene is represented as a voxel (volume element) grid, in which each voxel within a truncation region on either side of the modelled surface stores an approximate signed distance to the surface and a weight according to measurement confidence. An explicit surface topology and mesh connectivity can be extracted through implicit surface triangulation using an approach such as marching cubes [25]. The spatial resolution and extent are usually fixed [26], though implementations exist for efficiently handling larger scenes using hierarchical data or partially allocated structures [27]–[30]. Fusion of non-rigid geometry is typically achieved either by using a piecewise-rigid segmentation [31] or a warping field defined over a single reference volume [32], [33].

Inherently temporally consistent reconstructions can be obtained by deforming a pre-generated template model of a foreground dynamic object to match RGBD input using volumetric representations offline [34] or in real-time [35]. In contrast, our method does not require prior scanning of a template shape and attempts to recover the whole scene without prior knowledge of shapes in the scene.

In DynamicFusion [32], 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.

[1]Nuke https://www.foundry.com/products/nuke

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 iterative closest point (ICP) [36] 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). Our method also employs a signed distance function representation for fusion of measurements, as well as a graph structure controlling surface deformation, but performs graph-based motion/deformation estimation prior to dense volumetric surface fusion (offline), optimizing the deformation globally over the entire sequence. Furthermore, DynamicFusion performs a continuous warping of a *single reference grid* to integrate incoming frames, whereas our method performs rigid transformations on each of a *set of discrete part grids* (with softly overlapping assignment for fusion into a composite grid representing the complete surface). Newcombe *et al.* [32] note that their method cannot handle fast motion. Our method, however, can handle relatively large/fast motion given appropriate point tracks (such as those from large displacement optical flow [15]). It is also not clear how well DynamicFusion would handle complete scenes including background and possibly multiple disconnected objects, since the examples they provide show only the foreground. The proposed method is applicable to complete scenes, including background/multiple objects. It is not clear what the storage requirements for the DynamicFusion warping field are, but it is likely to be greater than the compact per-part motion used in the proposed method. Our discrete part-wise representation may also aid manipulation, for example easily selecting a limb or the head of a character. It is also not clear to what extent 4D scene editing would be supported by DynamicFusion. Direct comparison with our method is not possible due to the unavailability of a common test dataset or a publicly available implementation of DynamicFusion.

Similar to DynamicFusion, Innmann *et al.* [37] 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.* [38] perform online reconstruction from multiple depth sensors for improved scene coverage. In our work, a single RGBD sensor is used to build a fused dynamic model of the complete scene observed over time. Slavcheva *et al.* [39] 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 a point-based surface representation, points on the surface (surface elements, or 'surfels') are stored, possibly with additional properties such as normal, radius and confidence. Free space and unobserved space are not explicitly modelled, and there is no explicit connectivity between points. The storage cost is relatively low compared to volumetric methods. Keller *et al.* [40] demonstrate a surfel-based alternative to online volumetric fusion, which uses a single unstructured set of surfels representing a static scene. This point set is added to or removed from incrementally. It is able to handle, to a limited degree, dynamics in the scene, by re-labelling as 'dynamic' any model points inconsistent with the incoming depth stream. However, it does not reconstruct the dynamic regions.

In [3], a method for reconstruction of dynamic scenes from single-view RGBD 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.* [41] 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.

In this work, we propose a hybrid method for fusion and representation of dynamic scenes from RGBD video which uses the complementary strengths of multiple representations at different stages of processing (see Fig. 1). Briefly, *depth maps* provide input 2.5D geometry and are used along with the corresponding RGB images to generate sparse point tracks (Section IV) for dense surface integration (Section V) and to store residual depth between the final output 4D model and raw input (Section VI). An 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 (Section IV). 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 (Section V). 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. The key contributions of this work are: **(1)** extensions to the surfel graph modelling approach of [3] for improved performance (Section IV), **(2)** combining surfel graph modelling with volumetric TSDF modelling via soft part-wise fusion and global fusion to produce efficient, editable 4D models of general dynamic scenes (Section V) and **(3)** the proposal of criteria for validating the resulting model against the raw input data (Section VI).

III. Overview

A. Problem statement

Given a dynamic scene captured as a sequence of frames $\mathcal{F} = \{t_0, ..., t_{max}\}$ containing colour images $C(t)$ and depth maps $D(t)$ from a single, optionally moving RGBD sensor[2], we aim to extract an efficient, temporally consistent '4D' model of the scene that is more complete and less noisy than the raw input. The output model consists of a mesh with part skinning weights and motion trajectories. A residual depth map sequence may also be output to represent any unmodelled regions of the input.

B. Overview of proposed approach

The proposed hybrid approach is summarized in Fig. 1. Firstly, a raw surfel graph is established from the input RGBD sequence. The raw surfel graph is a set of connected oriented points, each tracked over part or all of the sequence. This surfel graph is then fed to an iterative piecewise segmentation, shape and motion estimation procedure (Section IV). Next, the piecewise surfel graph model is used to configure moving part volumetric grids. Depth measurements are softly assigned to the parts by interpolation over the corresponding surfel labels and integrated into volumetric part models using TSDF fusion (Section V). The per-part local geometry at a reference frame is then composited into a combined volume from which a composite reference mesh is extracted. This mesh is assigned skinning weights and animated based on the part models, yielding an efficient, temporally consistent, editable mesh representation.

To complete the output representation, a residual depth map is computed, which retains any unmodelled regions in the input as well as serving as a consistency check on the model.

IV. Surfel graph piecewise modelling

This section describes the piecewise surfel graph modelling stage (second row in Fig. 1), which builds on the approach of Malleson *et al.* [3]. We briefly summarize surfel graph modelling in Section IV-A; the reader is referred to [3] for a detailed description. Next, we describe our proposed extensions for improved performance: an alternative regularization for the segmentation based on pairwise and label costs (Section IV-B), concatenation of part models for improved scene coverage (Section IV-C), removal of stretched edges to support changes in topology (Section IV-D), and a post-process to blend assignment weights between parts for smoother-deforming geometry (Section IV-E).

A. Overview

Surfel graph modelling begins with a raw surfel graph produced from the input RGBD sequence: long-range feature-assisted optical flow [15] is performed on the RGB sequence $\{C(t)\}$ to provide a set $\mathcal{P}$ of sparse 2D point tracks p over a set of frames $\mathcal{F}_p \subseteq \mathcal{F}$. These are lifted to 3D point tracks $\mathbf{p}_p(t)$ using the depth maps $\{D(t)\}$[3]. A surfel connectivity matrix, $\mathbf{E}$, is established to form a surfel graph based on locality and geodesic distance preservation over time [42]. Refer to the supplementary material for further detail on the generation of the raw surfel graph.

The raw surfel graph $\{\mathcal{P}, \mathbf{E}\}$, which is noisy and incomplete, is then simultaneously segmented into a set, $\mathcal{M}$, of piecewise-rigid parts, m. The segmentation is encoded in a $|\mathcal{P}| \times |\mathcal{F}|$ matrix $\mathbf{F}$ with elements $f_{p,m} \in [0,1]$ (supporting soft assignment of a point to multiple parts). Each part contains *intrinsic* modelled points, $\mathbf{r}_p^m$ (in the part's local coordinate frame) and a global motion sequence $\mathbf{T}_m(t)$ that represents the dynamic pose of the part. A set of *extrinsic* modelled point tracks $\mathbf{q}_p^m$, (in global coordinates) are defined as the part's intrinsic points transformed by the part pose. The full sequence of input surfels $\mathbf{p}_p(t)$ is thus compactly modelled by a fixed set of local points, $\mathbf{r}_p^m$, with part motions $\mathbf{T}_m(t)$. This modelling process yields a compact, piecewise-rigid representation of dynamic shapes, with increased surface completeness and decreased noise compared to the input.

The goal of the piecewise surfel graph modelling stage is: given the input surfel graph $\{\mathcal{P}, \mathbf{E}\}$, determine the point-to-model assignment matrix $\mathbf{F}$, motion trajectories $\{\mathbf{T}_m(t) \quad \forall m \in \mathcal{M}, \forall t \in \mathcal{F}\}$, and intrinsic shape points $\{\mathbf{r}_p^m\}$, so as to minimize a cost function $E(\mathbf{F})$.

The output piecewise surfel graph model is produced by iteratively alternating between graph-cuts optimization of the segmentation $\mathbf{F}$ and re-estimation of part shape $\{\mathbf{r}_p^m\}$, and part motion $\{\mathbf{T}_m(t)\}$ with the current segmentation, until convergence. The cost function includes data terms for modelling fidelity as well as regularization terms, as described below.

B. Segmentation cost function

The data term includes a $|\mathcal{P}| \times |\mathcal{M}|$ matrix $\mathbf{C}$ with elements $c'_{p,m}$:

$$c'_{p,m} = \frac{1}{|\mathcal{F}_{pm}|} \sum_{t \in \mathcal{F}_{pm}} \left\| \mathbf{q}_p^m(t) - \mathbf{p}_p(t) \right\|_2^2 + \beta i_{p,m} \tag{1}$$

where

$$i_{p,m} = \begin{cases} \frac{|\mathcal{F}_p|}{|\mathcal{F}_{pm}|} - 1 & \text{if } |\mathcal{F}_{pm}| > 0 \\ \infty & \text{otherwise.} \end{cases} \tag{2}$$

This is the mean Euclidean distance between input point tracks, $\mathbf{p}_p(t)$, and modelled point tracks, $\mathbf{q}_p^m(t)$, over the common frames $\mathcal{F}_{pm}$, with an incompleteness penalty, $i_{p,m}$, added to penalize point-to-model assignments where $\mathcal{F}_{pm} \neq \mathcal{F}_p$. An empirical weighting factor $\beta = 0.1$ was used in all experiments.

Regularization is used to encourage the labels of neighbouring surfels to share the same label (keeping boundaries short) and also to model the scene efficiently, with as few parts as possible. There are different variations on graph-cuts segmentation applicable to the application of surfel graph modelling. Malleson *et al.* [3] use a formulation from the non-rigid structure from motion literature [42] which enforces part overlap (assigns points to more than one part at part boundaries) and also encourages use of as few parts as possible via an MDL (minimum description length) term, which adds

[2]If the RGB and depth sensors are not co-located, as is the case with both versions of the Kinect sensor, the depth map, D, is first re-mapped to the RGB point of view as a pre-process.

[3]An alternative to this would be to generate 3D point tracks using RGBD scene flow. Lifting 2D optical flow-based tracks [15] using depth was found to produce more reliable output models than 3D tracks from RGBD scene flow [17] (see supplementary material for details).

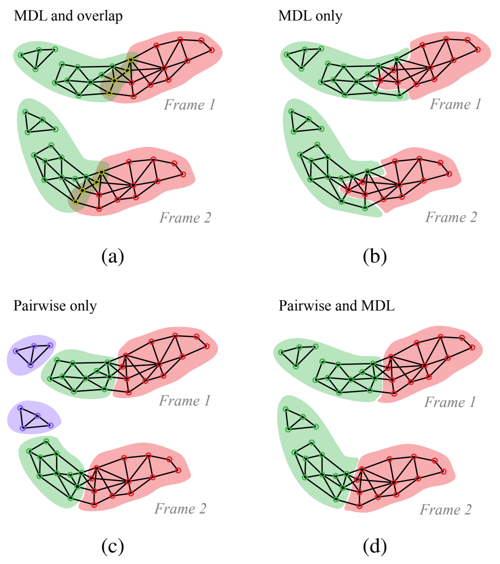


Fig. 2. Segmentation characteristics under different graph-cuts cost functions. a) Model-overlap and MDL b) MDL only (note the long boundary between the two parts) c) Pairwise only (note that the left hand region is segmented into two parts) d) Pairwise and MDL.

a fixed cost, MDL_m, for each label that has at least one point assigned to it. Delong *et al.* [43] propose a graph-cuts optimization procedure which includes both pair-wise and label cost (MDL) terms, and assigns each point to a single label. When applied to the surfel graph modelling application, these two formulations tend to have different behaviour in terms of the shape and number of segmented parts, as illustrated in Fig. 2. In our experiments, we found that pairwise plus MDL [43] produced the best segmentation boundaries (due to the pairwise smoothness term) while keeping the number of models low (due to the MDL term). Therefore, unless otherwise specified, the experiments in this work use the pairwise + MDL formulation [43]. The labelling cost function is as follows:

$$E(\mathbf{F}) = \overbrace{\sum_{p\in\mathcal{P}} c'_{p,f_p}}^{\text{data cost}} + \overbrace{\sum_{pq\in\mathcal{N}} V_{pq}(f_p, f_q)}^{\text{smoothness cost}} + \overbrace{\sum_{m\in\mathcal{M}'} \mathrm{MDL}_m}^{\text{label cost}} \quad (3)$$

where $\mathcal{M}'$ is the set of models with one or more points assigned to it and $\mathcal{N}$ are the edges in the surfel graph. In this, the Potts smoothing term V adds a cost for each edge connecting nodes with different labels:

$$V_{pq}(f_p, f_q) = \begin{cases} 0 & \text{if } f_p = f_q \\ \lambda_s & \text{otherwise} \end{cases} \quad (4)$$

where f_p and f_q are the labels for points p and q, respectively. The relative contributions of the smoothness and label costs compared to the data term are controlled by tunable parameters λ_s and MDL_m, respectively.

C. Part concatenation for improved coverage

After initial convergence of the piecewise surfel graph modelling stage, some approximately rigid surface regions may be modelled by several partially overlapping parts, each with incomplete temporal coverage. In order to improve the completeness of the reconstruction, we propose to perform a further set of iterations in which new parts, each generated by concatenating/merging two existing parts, are added to $\mathcal{M}$. A new part is added for each pair of existing parts. The set of points assigned to each new part model is the union of points assigned to the two source models. Any of these newly proposed models that perform as well as their source models (*e.g.* because they belong to the same approximately rigid surface region) will be selected instead of the two source models on the grounds of lower MDL cost. In all experiments, the concatenation and re-iteration of the modelling is performed three times for improved coverage. Examples of the improvement in coverage afforded by the part merging are provided in the supplementary material.

D. Stretched edge removal

Pairs of surfels belonging to separate surfaces that are in contact some of the time may be erroneously connected in the surfel graph. In order to correct the topology of the surfel graph, we propose to remove any edges which stretch significantly in the modelled surfel graph (in the experiments a stretch ratio threshold of 5:1 and an absolute stretch threshold of 15 cm were used). Removal of these spurious edges helps ensure that the topology of the final volumetric reconstruction is correct (separate objects are not joined together). Processing results with and without removal of stretched edges are shown in the supplementary material.

E. Segmentation blending

The graph-cuts segmentation produces a hard assignment of each point to a single part. This is not ideal for continuously deforming objects such as cloth. In order to reduce the effect of discontinuities at the boundaries between parts, we blend the point-to-part assignments. Smooth blending between rigid elements is common in the blend-skinning models widely used in computer graphics to approximate non-rigid surface deformation. This improves the perceptual quality when modelling scene content such as deforming cloth, by reducing the appearance of piecewise rigidity in the output while still using a small number of parts. As well as improving perceptual quality, it should also reduce the approximation error in most cases (*i.e.* thin-plate bending). Blending of the assignment weights would result in a smoother bending motion of the simplified representation of the non-rigid object shown in Fig. 2.

After convergence of the piecewise surfel graph segmentation, a *post hoc* blurring operation is applied to the weights using a Gaussian kernel in the surfel graph edge domain. More specifically, geodesic distances from each surfel point are computed using Dijkstra's algorithm, and used to weight contributions from each model. Geodesic distances are used to avoid blending the motion of surface regions which are not directly connected.

In Fig. 3, an example of the raw result is shown alongside the results after blurring with a standard deviation of 4 cm.

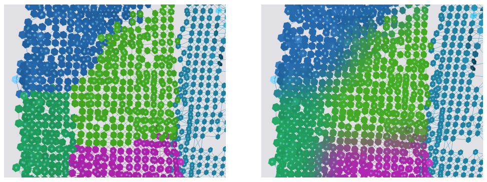


Fig. 3. Gaussian blurring of the piecewise segmentation assignment of the cloth in the *Globe* sequence. Left: raw segmentation result, right: after blurring with std. 4 cm. The soft transition between parts leads to more natural-looking output motion. Note that because geodesic distance in the surfel graph is used for the blurring, the wall to the right does not get mixed with the right-hand side of the cloth despite being close in Euclidean distance.

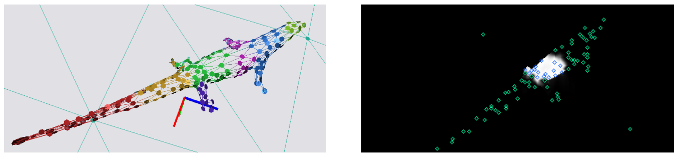


Fig. 4. Visualization of depth map pixel to part soft assignment map by scattered interpolation of surfel samples for the synthetic *Lizard* sequence. Left: Piecewise surfel graph model. Right: Samples assigned to the part are shown in blue (value 1), those not assigned are shown in green (value 0), and the interpolated values are shown in gray-scale. Note that IDW with truncated support offers a smooth and well-localised interpolation.

The amount of blur (standard deviation of the Gaussian kernel) affects the perceptual representation quality as well as the quantitative error. The evaluation section includes an empirical evaluation of blurring kernel size.

V. Volumetric fusion

Once the piecewise segmentation, sparse geometry and part motion sequences have been generated by the surfel-graph modelling stage, the next stage, volumetric TSDF fusion is performed (third row of Fig. 1). The geometry obtained from this volumetric fusion of the dense input depth maps is more detailed than that which would be obtained by surface fusion from only the sparse surfel graph geometry.

In the proposed approach, there are two stages of volumetric fusion. First, signed distance fusion of the depth measurements is performed in voxel grids for each part (intra-part fusion) yielding local geometry for each part. Second, the part voxel grids are composited in a global voxel grid to obtain the complete scene geometry (inter-part fusion). Assigning part 'skinning' to the mesh extracted from the global voxel grid, and animating the mesh using the part motion sequences, yields a dynamic temporally consistent 4D mesh suitable for editing or further analysis.

A. Per-part volumetric fusion

There are three components to the intra-part fusion process: initial configuration of each part voxel grid, soft assignment of depth map pixels to parts and fusion of the depth maps into the part voxel grids, using the soft part assignments.

1) Part voxel grid configuration: A TSDF voxel grid is generated for each part. The part grids are sized and initially posed so as to enclose all the modelled surfels for that part efficiently (refer to the supplementary material for details). The relative pose of the part grid for each frame is obtained simply by using the corresponding surfel graph part motion sequence. Note that the RGBD camera motion need not be explicitly modelled, but could be recovered, if desired, using the inverse of the pose of the background part (typically the largest part).

2) Soft depth to part assignment: As shown in Fig. 1, part voxel grids overlap with one another, both when the parts belong to the same non-rigid surface, and when separate objects are close to one another. In such a multiple voxel grid setup, depth measurements need to be selectively fused into only the corresponding part grid [31]. In this work we propose a dense soft assignment, allowing depth pixels to be integrated into multiple parts at the overlap between connected parts. The soft assignment technique allows the assignment of a given surface region to be shared between two or more volumetric parts in arbitrary non-rigid scenes. Specifically, a set of soft assignment maps $\mathcal{A} = \{A_m \ : m \in \mathcal{M}\}$ is produced, where $\mathcal{M}$ is the set of parts. Each assignment map is generated by using a sparse set of known assignment values, produced by projecting the part assignment values (between 0 and 1) from the 3D piecewise surfel graph into the depth map. Because the surfel graph sample points are irregularly distributed (rather than for instance forming a 2D grid), a scattered data interpolation method is required. The interpolation method needs to preserve the sample points and vary smoothly between sample data points of varying density without over/undershooting. It is also important to prevent propagation of part assignment values across depth discontinuities that may separate surface regions. Radial Basis Functions (RBFs) [44], and inverse distance weighting (IDW) [45] are two interpolation techniques which can easily be made to respect depth edges by replacing the Euclidean distance with geodesic distances. Radius-limited IDW [45] was found to perform better than RBF, which suffers from over/undershoot due to the irregular sampling of the projected surfels. An example of a dense soft assignment map for one part is shown in Fig. 4 and implementation details are provided below.

Given the input depth map $D(t)$, modelled surfel positions $\mathbf{q}_p(t) \quad \forall p \in \mathcal{P}$, and segmentation matrix $\mathbf{F}$ (with elements denoted $f_{p,m}$), the per-part soft assignment map $A_m(t)$ is generated. For simplicity of notation, the dependence on time and part are henceforth omitted. To begin with, there is a set $\mathcal{S} = \{a_p(\mathbf{u}_p)\}$ of known 2D pixel coordinates $\mathbf{u}_p$ and assignment values a_p. The coordinates $\mathbf{u}_p$ are produced by projecting all modelled surfel graph points $\mathbf{q}_p$ into D. The assignment values a_p are obtained directly from the corresponding entry in the surfel graph segmentation matrix: $a_p = f_{p,m}$. To eliminate occluded or inaccurate samples, a check for consistency in depth and normal is performed between the depth map and the sample points. Note that the sample positions are the same for each part, but their assignment values vary per part. The assignment values a for all interstitial pixels $\mathbf{u}$ are estimated by interpolation using inverse distance weighting with geodesic distances:

$$a(\mathbf{u}) = \begin{cases} \sum_{\mathbf{u}_p \in \mathcal{S}} d_g(\mathbf{u}, \mathbf{u}_p)^n \sum_{\mathbf{u}_p \in \mathcal{S}} d_g(\mathbf{u}, \mathbf{u}_p)^{-n} a_p & \text{if } \mathbf{u} \notin \mathcal{S} \\ a_p : \mathbf{u}_p = \mathbf{u} & \text{otherwise.} \end{cases} \tag{5}$$

where $d_g(\cdot,\cdot)$ denotes the geodesic distance, which is used instead of Euclidean distance to prevent propagation of assignment values across object boundaries. An image-plane geodesic distance is used, based on a graph of pixel nodes, each connected to its neighbours that are within a depth threshold. An inverse distance weighting power value of $n = 4$ was used in all experiments.

To avoid over-emphasis of densely sampled regions and to increase efficiency, a grid sub-sampling is performed on the set of projected sample points. This results in at most one sample point per grid bin. By setting the grid bin width to the input point track sampling density s, input sample density is maintained, but more densely sampled regions in the modelled surfel set (resulting from temporal extrapolation) are thinned out. In practice, this leads to about half the samples being kept. Note that there may still be regions with a significantly lower sample density, for instance due to a lack of optical flow-based input tracks in smoothly textured regions.

The IDW radius limit r is chosen to be large enough that most pixels are supported by several samples, *i.e.* $r = 5s$, where s is the assumed maximum sample spacing. The truncated distance maps are efficiently computed by processing only a region of interest window surrounding each sample point. This leads to a speed-up of two orders of magnitude in both the geodesic distance image computation and inverse distance summing. The geodesic distances are efficiently computed by pre-generating a distance map image for each sample point $\mathbf{u}_p$, and reusing these images to look up distance values when populating the interpolated values in each assignment map A_m. We use the image-plane geodesic distance, with 4-neighbours (city block distance). Edges in the graph are omitted if the depth difference between their nodes is more than an empirical threshold of 25 mm. The geodesic distance maps for each sample are computed by breadth-first search.

3) Intra-part fusion: For each frame, after obtaining the depth-to-part assignment maps, the depth map is fused into each part voxel grid in parallel on the GPU. The standard TSDF update equations are used for measurement fusion into each part grid G_m:

$$s_k(\mathbf{u}) = \frac{w_{k-1}(\mathbf{u}) s_{k-1}(\mathbf{u}) + w_k^m(\mathbf{u}) s_k^m(\mathbf{u})}{w_k(\mathbf{u})} \tag{6}$$

and

$$w_k(\mathbf{u}) = w_{k-1}(\mathbf{u}) + w_k^m(\mathbf{u}) \tag{7}$$

where s_k^m and w_k^m are the input TSDF and weight values for the current frame. The weighting, w_k^m, is set to the part assignment value $a(\mathbf{u})$ for the current part. This continuous weighting from the soft part assignment map results in a smooth falloff between connected parts, as illustrated in Fig 5. This smooth fall-off is used in blending the parts into a composite grid to obtain the global model, as described below.

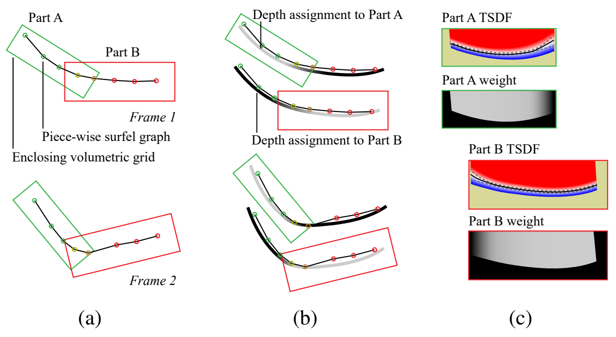


Fig. 5. Illustration of intra-part volumetric fusion (2D slices). (a) Two frames of piecewise surfel graph model showing part volumetric bounding boxes. (b) Dense soft assignment maps for each part at each frame. (c) Volumetric fusion showing TSDF and weight, along with implicit surface from each frame (dashed) and combined (solid).

B. Composite volumetric modelling

Once the piecewise-rigid geometry for each part has been generated, the part geometry is fused into a single composite volumetric model representing the full dynamic scene (inter-part fusion). This model is extracted to a mesh, which is skinned (with linear blend skinning), and finally topologically cleaned (filtered) to remove invalid faces. This final animated mesh is a temporally coherent 4D representation of the dynamic scene.

1) Inter-part fusion: A composite voxel grid $G_c = \{S_c, W_c\}$ is generated, sized and posed so as to enclose the grid volumes of the subset of part models, $\mathcal{M}_{cr}$ ("composite reference") which are valid (have poses) in an arbitrarily chosen reference frame t_{cr} (the middle frame of the sequence was used in the experiments). See the supplementary material for details.

The volumetric compositing is restricted to surface band regions only, *i.e.* specifically excluding free space regions. This is because incorporating free space regions at this stage could cause sections of surface to go missing, or become biased (shifted). Fig. 6 illustrates the TSDF grid compositing procedure. The compositing is performed part by part using tri-linear interpolation lookup into the part grids (in parallel on the GPU). The procedure is formalised in Algorithm 1, where $\phi_m(\mathbf{u})$ denotes the transform from G_c voxel coordinates to G_m voxel coordinates (at the frame t_{cr}) and ϵ is a tolerance which ensures that free-space regions are not included in the composite grid (ϵ = 1 mm in the experiments).

Algorithm 1 Volumetric compositing of part grids into the composite grid (μ is the TSDF truncation distance)

```
for $\forall m \in \mathcal{M}_{cr}$ do
    for $\forall \mathbf{u} \in G_c$ do
        if $s_m(\phi_m(\mathbf{u})) < \mu - \epsilon$ then
            $s_c(\mathbf{u}) \leftarrow [s_c(\mathbf{u}) \cdot w_c(\mathbf{u}) + s_m(\phi_m(\mathbf{u})) \cdot w_m(\phi_m(\mathbf{u}))] / [w_c(\mathbf{u}) + w_m(\phi_m(\mathbf{u}))]$
            $w_c(\mathbf{u}) \leftarrow w_c(\mathbf{u}) + w_m(\phi_m(\mathbf{u}))$
        end if
    end for
end for
```

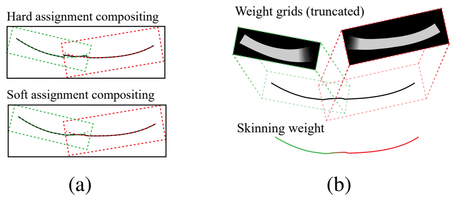


Fig. 6. Inter-part fusion (2D slices). (a) Fusion of two part grids without (top) and with (bottom) soft assignment weighting for intra-part fusion. Note the relatively smooth transition between parts with the soft assignment. (b) Composite mesh skinning by lookup into part SDF and weight grids.

A composite mesh with vertices $\mathcal{V}$ is then extracted using marching cubes [23]. This 'reference' mesh shape is valid for a single time instant t_{cr}. This reference mesh is animated to produce a 4D model over the sequence, as described in the following section. Note that the reference mesh includes dynamic as well as static regions in a unified representation, *e.g.* the background geometry is also included in the mesh.

2) Linear blend skinning for composite mesh: In order to animate the extracted mesh, a mapping between each vertex and one or more part models needs to be established. The $|\mathcal{V}| \times |\mathcal{M}|$ skinning weight matrix $\mathbf{W}$ encodes the weights $w_{i,m}$ of assignment of each mesh vertex $\mathbf{v}_i$ to each part model m. While it would be possible to refer back to the surfel graph model to estimate the assignment weights, it is simpler to refer to the intermediate part grids. The skinning weights are determined by looking up the weight in the voxel grids for each part (at t_{cr}), but setting the weight only if the point lies in the surface band, not in free space:

$$w_{i,m} = \begin{cases} w_m(\phi_m(\mathbf{v}_i)) & \text{if } w_m(\phi_m(\mathbf{v}_i)) > 0 \text{ and} \\ & \quad s_m(\phi_m(\mathbf{v}_i)) < \mu - \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where ϕ_m transforms from metric vertex coordinates to part m voxel coordinates. This is further illustrated in Fig. 6(b).

The vertex transform matrix $\mathbf{T}_m^V(t)$ of the mesh vertices under each model m at time t is determined as follows

$$\mathbf{T}_m^V(t) = \mathbf{T}_m(t)\mathbf{T}_m^{-1}(t_{cr})\mathbf{T}_m^G \tag{9}$$

where $\mathbf{T}_m^G$ is the composite grid pose (which is defined at t_{cr}). The transformed vertices $\mathbf{v}_i'$ are determined using a simple linear blend as follows:

$$\mathbf{v}_i' = \frac{\sum\limits_{m \in \mathcal{M}} w_{i,m}\mathbf{T}_m^V(t)\mathbf{v}_i}{\sum\limits_{m \in \mathcal{M}} w_{i,m}} \tag{10}$$

3) Mesh filtering: The composite mesh may have topology problems: if two disconnected parts are close to each other in the composite reference frame, the composite mesh will have the parts joined together. The skinning weights will be a blend between the two parts in this case. Such incorrect mesh topology can cause serious artifacts in the output especially if the two parts are not close together for the whole sequence. To mitigate this, a post-processing filter is applied to the skinned mesh. Any vertex which has non-zero skinning weights for at least one pair of non-connected surfel graph parts is removed.

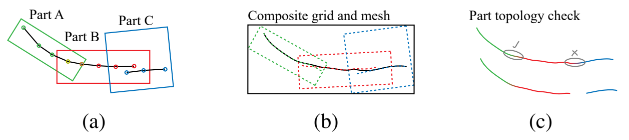


Fig. 7. Composite mesh topology check. (a) Piecewise surfel graph and bounding boxes with three parts. Note that part A and B are connected, but B and C are not. (b) Composite grid showing incorrectly fused part B and C in the extracted surface. (c) Top: topology check on skinned mesh - any valid blended regions need to be connected in the surfel graph. Bottom: invalid vertices removed.

Refer to Fig. 7 for an illustration of the mesh filtering approach.

VI. Verification of Dynamic Model and Residual Modelling

The dynamic mesh representation (referred to here as a 4D representation) offers enhanced manipulability by virtue of its part structure and temporally consistent vertex structure. By contrast, no such structure exists in the 2.5D input depth maps.

In practice, the proposed geometric modelling will not produce a fully complete 4D surface representation. It is possible that some input pixels are not represented in the model, for instance if there were no valid point tracks in their vicinity. It is also possible that some regions of the model produced can be inconsistent with the input, either because of modelling error or measurement outliers.

As a sanity check, one can render a depth map of the output model (for any given frame) from the point of view of the depth sensor and compare it with the input depth map. Such a rendering will reveal pixels for which no input or model exists (for which only 2D information is present), and input pixels that are not modelled (for which only 2.5D information is present). Below we propose a criteria by which to establish whether the input is consistent with the output, and then propose to output residual depth maps alongside the dynamic 4D mesh model as a fall-back for completeness of representation in the event of the 4D modelling procedure failing to model portions of the scene.

A. Input output consistency check

Certain assumptions are made about the depth capture process. These are described below and illustrated in Fig. 8. Firstly, most measurements d_i are reliable and fall within some noise threshold n of the true depth d_{gt}, except near depth edges:

$$|d_i - d_{gt}| < n(d_{gt}) \tag{11}$$

where n may be a function of distance (for instance based on the noise standard deviation, which for the Kinect v1 increases quadratically with depth [10]). For brevity, the dependence of n on depth is omitted from the notation.

Low-confidence regions (*e.g.* due to poor surface reflectance properties) are usually reported as missing ($d_i = 0$), but occasionally lead to gross outliers, which can be either behind or in front of the true surface. Secondly, on both sides of depth edges, there is a band of pixels with some width b whose depths are uncertain and could belong to either the

TABLE I
CATEGORIES FOR CHARACTERIZATION OF CONSISTENCY OF RENDERED MODEL DEPTH d_m WITH INPUT DEPTH d_i, SUBJECT TO NOISE THRESHOLD n AND DEPTH EDGE BAND CONDITION $e(d_i)$.

Category		Input	Model	Condition	Modelling status
🟦	1	-	-	-	2D
🟧	2	✓	-	-	2.5D
🟨	3	-	✓	-	4D (consistency unknown)
🟩	4	✓	✓	$\|d_i - d_m\| < n$	4D (consistent)
🟥	5	✓	✓	$d_i - d_m < -n, \neg e(d_i)$	4D (consistency unknown)
🟦	6	✓	✓	$\|d_i - d_m\| > n, e(d_i)$	4D (inconsistent in edge band)
🟩	7	✓	✓	$d_i - d_m > n, \neg e(d_i)$	4D (inconsistent)

local foreground or background. Finally, very fine structures smaller than a few pixels in width cannot be reliably captured, and pixels associated with these are reported as missing or fall on the surrounding background. These assumptions hold reasonably well for practical active depth sensors such as the Kinect.

Based on these assumptions, the modelling performance may be characterised in terms of the pixel-wise agreement between the input depth map and the depth map synthesised from the model. There are seven categories, as listed in Table I and illustrated in Fig. 8. Categories 1-3 cover cases where depth for the input, the model or both are missing. For an input pixel depth d_i and a modelled pixel depth d_m to be considered consistent, they must fall within a noise threshold n of each other (Cat. 4). The presence of a measurement in front of the model (Cat. 5) does not necessarily indicate that the model is in error, merely possibly incomplete. A special case is made for pixels lying within the depth edge band (indicated by the condition $e(d_i)$, which are inherently prone to being inconsistent (Cat. 6). The depth threshold was set to 25 mm and edge band to 4 pixels for all experiments. Away from depth edges, the model and input are known to be inconsistent if the input depth is significantly greater than the modelled depth (Cat. 7). This may either be due to an outlier depth measurement, or to error in the modelled surface.

An example input depth frame, and modelled rendering are shown in Fig. 12, along with the input depth edge bands and the colour coded category of each pixel as described in Table I. Depending on the capture conditions, scene content and processing settings, different proportions of the depth maps may fall into each category. These proportions are an indication of both capture quality (completeness of measured depth maps) and modelling fidelity (completeness and consistency of the model w.r.t. the input frames). In the evaluation section, these are reported for various test scenes.

B. Residual depth maps

One of the purposes of the 4D modelling procedure is to produce a less noisy and more complete model of the input. However, as discussed in the previous section, some of the input may not be represented in the 4D model, or the model may be inconsistent with the input. In order to validate the model w.r.t. the input and in order not to discard any unmodelled input depth pixels, the concept of a 'residual depth map' is introduced. A basic residual depth map simply stores the difference d_r between the input and modelled pixels:

$$d_r = d_i - d_m. \tag{12}$$

If the residual depth is added to the modelled depth map to form an 'output depth' d_o, the exact input is recovered (including all input noise). It would be more useful to store the 'noise-floored' difference $\hat{d}_r$, which zeros the differences when the depths are consistent (category 4) but stores the exact difference otherwise:

$$\hat{d}_r = \begin{cases} 0 & \text{if } (|d_i - d_m| < n) \\ d_r & \text{otherwise} \end{cases} \tag{13}$$

If the noise-floored residual depth map is added to the modelled depth map, the exact input is recovered wherever the model is missing or inconsistent with the input, thus yielding a depth map which uses the reduced noise modelled depths where possible, but retains the completeness of the input. Examples of the exact and noise-floored residual depth maps are shown in Figs. 12(f) and 12(g), respectively.

An output depth $d_o = d_m + \hat{d}_r$ has an error of up to n w.r.t. the input:

$$|d_o - d_i| < n. \tag{14}$$

For all pixels not associated with any very thin structures, depth edges, or gross measurement outliers, it follows from Eq. 11 that the bound on the error of d_o w.r.t. the true depth is $2n$:

$$|d_o - d_{gt}| < 2n. \tag{15}$$

It is, therefore possible for the model + noise-floored residual depth representation to add at most $2n$ error to any given pixel, however the average error level of the modelled depth may be lower than to the (noisy) input depth (see Section VII-A).

Because the noise-floored residual depth map contains a significant proportion of zero[4] entries, it can be losslessly compressed to a smaller file size than the input depth maps, thus improving storage efficiency compared to raw depth maps or exact residual depth maps.

VII. EVALUATION

We evaluate our proposed hybrid approach in terms of objective and subjective fidelity of the output model (shape and motion), completeness of the model representation w.r.t. the input data, as well as computational resources, and final storage cost. Unless otherwise specified, all the sequences are processed with $\text{MDL}_m = 1 \times 10^{-1}$ and $\lambda_s = 1 \times 10^{-1}$, and no blurring of assignment weights is performed ($\sigma = 0$).

A set of test sequences covering a range of scene content and capture devices (real and virtual) were used in the evaluation. This dataset has been made available for future research into dynamic RGBD scene modelling[5]. Refer to the supplementary video for visualizations of the full input sequences and processing results.

[4]In the internal processing as well as the PNG format used for storage, unsigned 16-bit values are used for depth maps, therefore a large constant value is added to the difference maps to ensure that no negative values occur.

[5]Available: http://cvssp.org/projects/4d/dynamic_rgbd_modelling. To the best of our knowledge, there are no publicly available datasets for dynamic RGBD modelling which include real captures as well as synthetic data with ground truth.

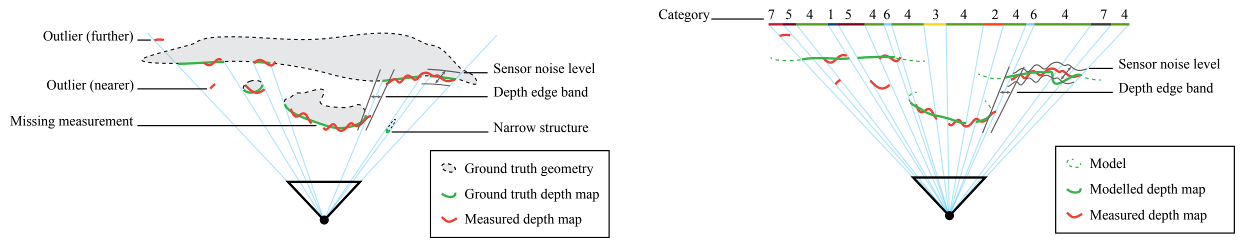


Fig. 8. Left: Consistency between ground truth and measured depth. Right: Consistency between measured depth and modelled depth (category numbers are as described in Table I).

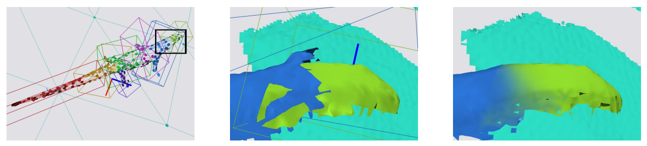


Fig. 9. Volumetric processing for the *Lizard* sequence. Left: Piecewise surfel graph showing volumetric part grid initialization. Centre: Detail of part meshes, where overlapping regions do not align exactly due to non-rigidity. Right: Detail of the composite mesh showing a seamless merging of part surface regions with a soft transition between parts. Colours indicate part membership.

A. Synthetic sequences

Two synthetic sequences, *Lizard* (250 frames) and *Globe* (300 frames) were generated using Blender [46] and clean ground truth depth used for reference. For input to our method, we simulated Kinect v1 capture artifacts (*e.g.* jagged edges, holes and missing fine structures) using BlenSor (Blender Sensor Simulation plugin for Blender) [47], along with depth-dependent random noise and quantization according to [10]. A sample simulated depth map is shown in the top row of Fig. 1.

Fig. 9 illustrates our pipeline on the *Lizard* sequence, which contains a single articulated surface and background. The piecewise surfel graph result is used to configure the overlapping part volumes (left), and using the soft depth-to-part assignment, depth measurements are fused to reconstruct surfaces for each part (centre). The separate part volumes are finally combined in the global volume (right). This composite mesh, along with the part motion trajectories and skinning weights, efficiently represents the dynamic surface of the scene.

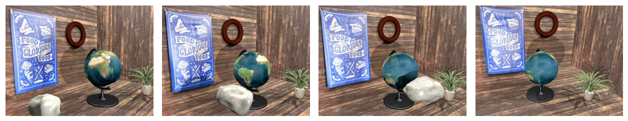
The *Globe* sequence contains multiple elements: a background which is moving w.r.t. the camera, a rotating globe, a non-rigidly deforming cloth, and a flying rock. The globe is a good test of the approach's ability to extrapolate occluded regions, and also to assess drift of the reconstruction on surface regions which reappear having been occluded (after a full revolution of the globe). Results from our full pipeline are shown in Fig. 10. The segmentation into parts has worked well, with the entire rigid background being assigned to a single part, and the deforming cloth being segmented into patches. The rock and globe are each segmented into a single part. Despite the globe undergoing 1.5 revolutions over the sequence, the full surface is reconstructed without noticeable drift. Since the cloth comprises only three patches, fine scale deformation of the cloth is not represented, and thus there is significant residual depth in the cloth, particularly in the third frame shown. Fig. 11 shows processing results over a range of part segmentation blurring kernel widths σ. All σ values, from 0 to 8 cm, produce similar values of the residual depth (w.r.t. the noisy input depth) and residual RMS error (w.r.t. the ground truth depth). Qualitatively, however, the deformation of the cloth is smoother when blending is enabled. Note that the RMS error in the noisy input depth for this scene (ignoring missing measurements) is 5.0 mm, and the RMS error in the reconstruction is 3.5 mm, *i.e.* the modelled depth is less noisy than the raw depth. However, a drawback is that the fine ripple deformations of the moving cloth are not represented in the composite mesh model.

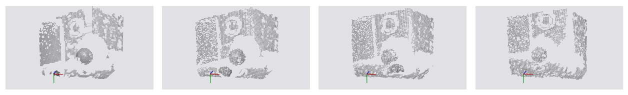
B. Real sequences

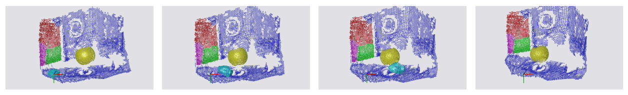
The approach was tested on several sequences recorded using the Kinect v1 (structured light) and Kinect v2 (time-of-flight) sensors. The Kinect v1 sequences include *Dog* (610 frames), *Cat* (251 frames), *Paris* (251 frames), *Rabbit and Deer* (311 frames), and *Turning* (200 frames). The Kinect v2 sequences include *Shirt* (100 frames), *Sitting* (200 frames), and *Entrance* (230 frames). We further test out approach on three sequences from the University of Tsinghua Dynamic RGBD dataset [34]: *Puppet* (300 frames), *Pillow1* (370 frames), and *Pillow2* (419 frames).

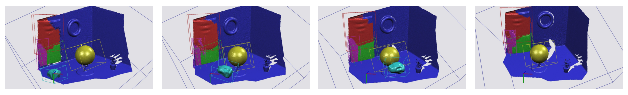
1) Kinect v1: As discussed in Section VI, it is possible to assess the completeness and consistency of the final representation w.r.t. the input data. Fig. 12 shows the modelling properties and residual depths for the *Dog* sequence. The input depth (Fig. 12(b)) and its edge bands (Fig. 12(c)) are used along with a rendering of the composite mesh model (Fig. 12(d)) to classify the modelling status of each pixel in the frame (Fig. 12(e)) according to the categories listed in Table I. The proportion of points in each category over the whole sequence is shown in Fig. 13, which also shows the RMS residual[6] between the input depth and the modelled depth over time (for consistent - *i.e.* category 4 - regions). This residual gives an indication of how closely the output model matches the input. Fig. 12(f) shows the exact residual

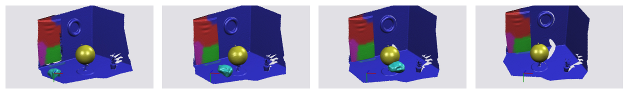
[6]In the case of noisy depth (real sequences), the residual is the result of both measurement error and modelling error (no ground truth depth is available). In the case of noise-free depth (clean, synthetic sequences), the residual is equivalent to modelling error.

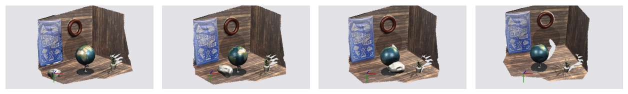

(a) Input RGB

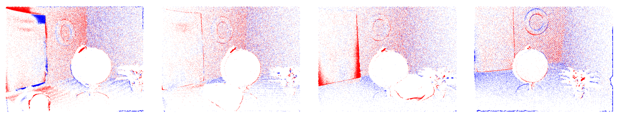

(b) Raw surfel graph


(c) Piecewise surfel graph reconstruction


(d) Piecewise volumetric reconstruction


(e) Composite volumetric reconstruction


(f) Composite volumetric reconstruction (with texture)


(g) Residual depth, scale: -20 20mm

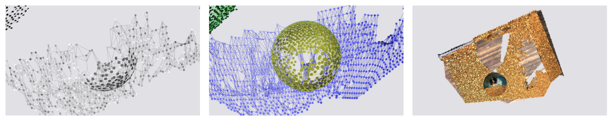
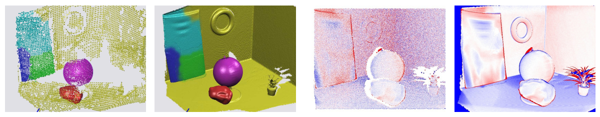
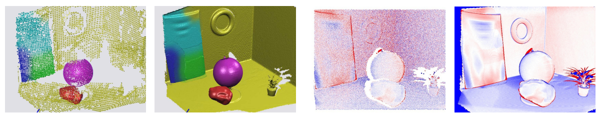

(h) Top view, from left to right: surfel graph, piecewise surfel graph, and output mesh showing input depth for the current frame overlaid. Note that the globe has been completed by the modelling process.

Fig. 10. Hybrid processing of *Globe* sequence using image-based point tracks and simulated Kinect noise. A high quality segmentation and reconstruction is produced by the proposed method and the globe has been faithfully reconstructed as a single object for the whole sequence without noticeable drift despite undergoing 1.5 revolutions.

depth between the output composite mesh model rendering and the input depth map. Note the high-frequency random noise. Fig. 12(g) shows the residual after zeroing of the low amplitude values within an assumed noise threshold of 10 mm, which is two standard deviations of Kinect v1 noise at typical scene distances [10][7]. Note that this omits the small residuals, but stores regions not captured by the modelling process, in particular the distant background (no model) and the tennis ball (which - because it disappears from view - is not present in the composite model for the whole sequence). Finally, for

[7]For simplicity, a fixed threshold is used, but in principle, a variable threshold could be used to account for sensor noise that varies with scene depth.


(a) $\sigma = 0$ mm


(b) $\sigma = 4$ cm

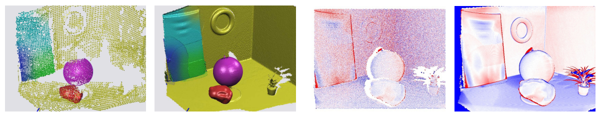
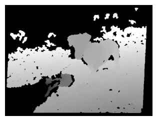
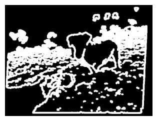
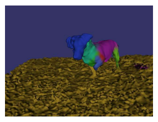

(c) $\sigma = 8$ cm

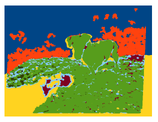
Fig. 11. *Globe* sequence processing with a range of segmentation blurring weights. From left to right: surfel graph model showing blended assignments, composite volumetric mesh, exact residual depth w.r.t. (noisy) input, exact residual depth w.r.t. ground truth depth (*i.e.* reconstruction error) - scale: -20 20mm. Across blurring radii, there is no significant difference in output/input RMS residual or output/ground truth RMS error (at 5.0 mm and 3.5 mm, respectively), but visually the deformation of the cloth appears more natural.

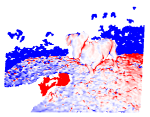
(a) Input RGB

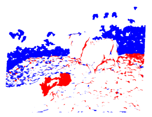

(b) Input depth


(c) Depth edge bands


(d) Composite mesh model render


(e) Modelling categories


(f) Exact depth residual


(g) Noise-floored depth residual

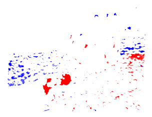

(h) Residual ignoring edge bands

Fig. 12. Consistency between input and model for a frame of the *Dog* sequence. Note that the hand, tennis ball and distant background in the input are not captured by the composite mesh model in this frame, and therefore the residual depth maps represent this geometry on a per-frame basis. Scale: -20 20mm.

completeness, a third variant of the residual depth is shown in Fig. 12(h), in which the depth edge bands have been omitted as well. Unless otherwise specified, the residual depth maps presented in the subsequent figures and numerical results are the noise-floored versions. Scene edits on the *Dog* sequence are presented in Section VII-D. Refer to the supplementary material for results on the other Kinect v1 sequences (*Cat*, *Rabbit and Deer*, and *Turning*).

2) Kinect v2: Results on the *Puppet* sequence are shown in Fig. 16. Our approach recovers the the non-rigidly deforming puppet and subject as well as the background. For comparison,

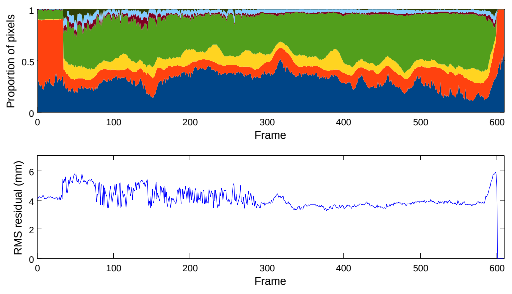


Fig. 13. Modelling properties for *Dog* sequence. Top: Proportion of image pixels in each modelling category (1-7) over time. Refer to Table I for interpretation of categories. Bottom: RMS error in consistent (category 4) regions. The lack of modelled regions at the start/end of the sequence is due to the hand-held RGBD sensor not being aimed directly at the scene.

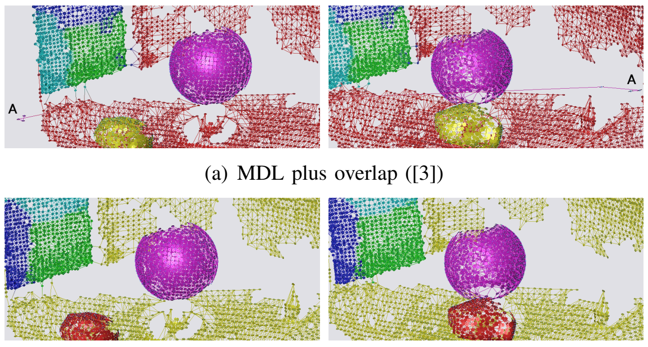
the results of Guo *et al.* [34] are shown, in which only the puppet and the subject's arm are reconstructed (with the aid of a pre-scanned template). The results on the other Kinect v2 sequences (*Shirt*, *Sitting*, *Entrance*, *Pillow1* and *Pillow2*) are presented in the supplementary material.

C. Segmentation regularization

In Fig. 14, additional results are presented to motivate use of the MDL plus pairwise smoothness regularization (Equation 3), used throughout this work in place of the MDL plus overlap regularization used in [3]. The sequence was processed using each method, choosing regularization values which produce the same number of final part models (*i.e.* 6 parts, as per the result in Fig. 10): $\mathrm{MDL}_m = 1 \times 10^{-1}$ and $\lambda_s = 1 \times 10^{-1}$ for our approach (Equation 3), and $\mathrm{MDL}_m = 1$ for [3]. Note that because of the pairwise term, the proposed approach leads to a more compact boundary between the patches in the waving cloth. While the modelling error for the visible point tracks is similar, at 3.1 mm for both, some points on the floor have been misassigned to the rotating globe when using the approach of [3]. Finally, we note that the processing time for the piecewise surfel graph modelling iterations in our approach is approximately half that of [3], owing to the simpler graph-cuts formulation used. The effect of the weightings of the MDL and smoothness terms on the number of part models is discussed in the supplementary material.

D. Scene editing using the representation

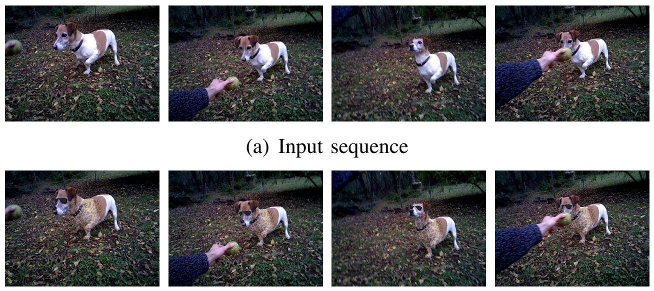
The reference composite mesh is saved with per-vertex colours in the PLY format. It can therefore be imported into standard 3D graphics editing software (*e.g.* Blender [46]), which is suited to creative manipulation of meshes. After the edits are performed, the mesh can be saved back to PLY and imported back into our software, where it is re-animated, rendered from the point of view of the input camera position, and finally composited with the input RGB frame to produce an edited video sequence. Fig. 15 shows the original *Dog* sequence, along with an edited version in which shape and appearance (texture) have been edited in Blender, rendered and composited back onto the image sequences. Note that the edits need only be performed on the reference mesh and are automatically propagated over the dynamic sequence.



(a) MDL plus overlap ([3])

(b) MDL plus pairwise (ours, Equation 3)

Fig. 14. Comparison of graph cuts segmentation performance on the *Globe* sequence under the two regularization schemes (two frames shown). In (a), note the irregular boundaries between patches on the cloth and the incorrectly assigned surfels, such as the cluster from the floor marked 'A' which has been assigned to the rotating globe part.



(a) Input sequence

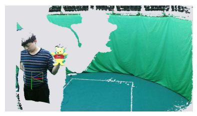
(b) Scene edits composited into sequence

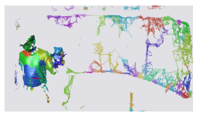
Fig. 15. Shape and texture editing of the *Dog* sequence - edits include a leopard spot pattern, a protruding rhino horn, goggles, and studs on the collar. The edits are performed on a single reference frame and propagated over the sequence as a result of the temporally consistent 4D mesh representation.

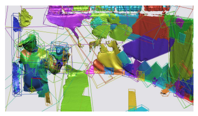
E. Computation and storage considerations

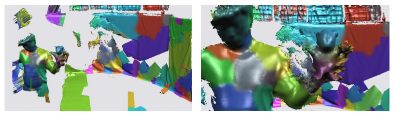
1) Processing time: In our implementation, which is mainly un-optimized single-threaded CPU code, the stages of processing take of the order of minutes to hours for typical sequences. Processing time varies depending on factors such as image resolution, number of surfels, voxel grid dimensions and number of frames (see the supplementary material for further detail).

2) Storage cost: The surfel graph and piecewise surfel graph (segmentation, intrinsic shape and motion) models are stored in binary files as discussed in [3]. Images (input depth maps and residual depth maps) are stored as 16-bit PNG images (with compression enabled). Meshes are stored in PLY format (with binary encoding), and the skinning matrix is stored as a sparse matrix in a binary file. We report the total file sizes after applying standard lossless (ZIP) file compression. Fig. 17 shows, for various test sequences, the storage costs of input depth map sequence, the raw surfel graph, the intermediate piecewise surfel graph, the composite mesh (with skinning), and residual depth maps. Note that in all cases, the final output representation, which has a temporally consistent structure, is smaller than the raw input depth maps. Using the output representation (third columns in the chart), the input


(a) Input RGBD


(b) Piecewise surfel graph


(c) Piecewise and volumetric reconstruction


(d) Composite volumetric reconstruction with closeup of puppet

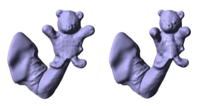

(e) Results of Guo *et al.* [34] (using a template shape)

Fig. 16. Hybrid processing of *Puppet* sequence comparing the result to Guo *et al.* [34]. Although our result suffers from some reconstruction artifacts, it reconstructs the entire dynamic scene including background without a template.

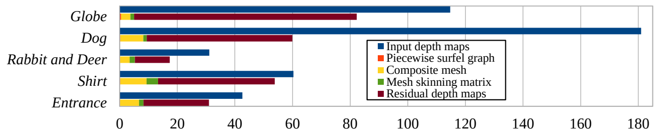


Fig. 17. Storage cost of dynamic sequences using the hybrid approach (with lossless ZIP compression). Note that the piecewise surfel graph files are relatively small (*e.g.* 200KB), so they are barely visible on this plot.

depth can be reproduced to within the noise threshold specified during processing, while the 4D model typically represents approximately 80% of the observed scene area (refer to the plots in the preceding section). Further processing such as edits which require temporal consistency can be performed using the output 4D mesh, and in applications which do not require retention of the residual depth maps, an order of magnitude compression is achieved w.r.t. the raw input depth.

VIII. Conclusions and Future Work

In this paper, a hybrid surfel graph + volumetric fusion approach was proposed for modelling and representing general non-rigid scenes from noisy RGBD sequences. The surfel graph modelling approach of [3] was extended to improve its performance and used as an intermediate sparse 4D representation to drive piecewise volumetric and composite volumetric fusion for dense surface reconstruction. This approach outputs a detailed, temporally consistent 4D mesh, along with residual depth maps, enabling efficient representation of 4D data where available, and gracefully falling back to unstructured 2.5D raw depth for any unmodelled regions.

The method was demonstrated on complex dynamic scene data from various sources including Kinect v1, Kinect v2 and synthetic data, and shown to produce reasonable results. Examples of shape and appearance editing were also presented, suggesting the method's potential for use in 3D and video content production applications. Future work could investigate motion editing using the approach, *e.g.* by learning a reduced-dimensionality motion parametrisation for a sequence.

The method has some limitations in terms of dynamic surface reconstruction quality. In particular, while it is effective at removing high-frequency noise from the input, the piecewise fusion loses some fine-scale non-rigid deformations, *e.g.* of cloth. Future work could investigate the use of a hierarchical/multi-scale version of the piecewise surfel graph modelling approach, along with long-term re-identification of partial point tracks, to ensure good large-scale consistency and persistence of the reconstructed surface, while capturing fine-scale, subtle deformations. Related to this, the method could potentially be improved by performing a data-driven blending of part assignment weights as part of the modelling optimization itself, rather than using a fixed radius blur heuristic, which may not optimally represent the deformation throughout a dynamic scene. It would be interesting to investigate possible use of machine learning approaches (*e.g.* deep learning) for this. Another avenue of future work would be to extend the approach to multiple RGBD views for improved scene coverage and robustness. This would involve obtaining point correspondences across views when generating the initial surfel graph representation.

Acknowledgment

This work was funded by the EU FP7 project *SCENE* and EPSRC *Audio-Visual Media Platform Grant* EP/PO22529/1.

References

[1] C. Budd, P. Huang, and A. Hilton, "Hierarchical Shape Matching for Temporally Consistent 3D Video," in *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, May 2011, pp. 172–179.
[2] M. Tejera and A. Hilton, "Learning part-based models for animation from surface motion capture," *International Conference on 3D Vision (3DV)*, 2013.
[3] C. Malleson, M. Klaudiny, J.-Y. Guillemaut, and A. Hilton, "Structured representation of non-rigid surfaces from single view 3D point tracks," in *International Conference on 3D Vision (3DV)*, 2014, pp. 625 – 632.
[4] J. Starck and A. Hilton, "Surface capture for performance-based animation." *Computer Graphics and Applications*, vol. 27, no. 3, pp. 21–31, 2007.
[5] C. Cagniart, E. Boyer, and S. Ilic, "Free-form mesh tracking: a patch-based approach," in *Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1339–1346.
[6] C. Russell, R. Yu, and L. Agapito, "Video Pop-up: Monocular 3D Reconstruction of Dynamic Scenes," *European Conference on Computer Vision (ECCV)*, pp. 583–598, 2014.
[7] R. Ranftl, V. Vineet, Q. Chen, and V. Koltun, "Dense Monocular Depth Estimation in Complex Dynamic Scenes," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4058–4066.
[8] S. Kumar, Y. Dai, and H. Li, "Monocular Dense 3D Reconstruction of a Complex Dynamic Scene from Two Perspective Frames," in *International Conference on Computer Vision (ICCV)*, 2017, pp. 4659–4667.
[9] P. Ji, H. Li, Y. Dai, and I. Reid, "'Maximizing Rigidity' Revisited: A Convex Programming Approach for Generic 3D Shape Reconstruction from Multiple Perspective Views," in *International Conference on Computer Vision (ICCV)*, vol. 2017-Octob, 2017, pp. 929–937.
[10] K. Khoshelham and S. O. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.
[11] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *International Conference on Computer Vision (ICCV)*, 1998, pp. 839–846.
[12] C. Richardt and C. Stoll, "Coherent spatiotemporal filtering, upsampling and rendering of RGBZ videos," *Computer Graphics Forum*, vol. 31, no. 2, 2012.
[13] O. Wang, J. Finger, Y. Qingxiong, J. Davis, and Y. Ruigang, "Automatic natural video matting with depth," *Pacific Conference on Computer Graphics and Applications*, pp. 469–472, 2007.

[14] M. Hernandez, J. Choi, and G. Medioni, "Near laser-scan quality 3-D face reconstruction from a low-quality depth stream," *Image and Vision Computing*, vol. 36, pp. 61–69, 2015.
[15] N. Sundaram, T. Brox, and K. Keutzer, "Dense point trajectories by GPU-accelerated large displacement optical flow," in *European Conference on Computer Vision (ECCV)*, 2010.
[16] D. Ferstl, G. Riegler, M. Rüether, and H. Bischof, "CP-Census: A Novel Model for Dense Variational Scene Flow from RGB-D Data," in *British Machine Vision Conference (BMVC)*, 2014, pp. 18.1–18.11.
[17] M. Jaimez, M. Souiai, J. Gonzalez-Jimenez, and D. Cremers, "A primal-dual framework for real-time dense RGB-D scene flow," in *International Conference on Robotics and Automation (ICRA)*, vol. 2015-June, no. June, 2015, pp. 98–104.
[18] P. Wang, W. Li, Z. Gao, Y. Zhang, C. Tang, and P. Ogunbona, "Scene Flow to Action Map: A New Representation for RGB-D based Action Recognition with Convolutional Neural Networks," *arXiv*, 2017.
[19] M. Hornáček, A. Fitzgibbon, and C. Rother, "SphereFlow: 6 DoF scene flow from RGB-D pairs," in *Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3526–3533.
[20] Y. Wang, J. Zhang, Z. Liu, Q. Wu, P. Chou, Z. Zhang, and Y. Jia, "Completed Dense Scene Flow in RGB-D Space," in *Asian Conference on Computer Vision (ACCV) Workshops*, vol. 9009, 2015, pp. 191–205.
[21] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *ACM SIGGRAPH*, 1994, pp. 311–318.
[22] E. Piazza, A. Romanoni, and M. Matteucci, "Real-time cpu-based large-scale three-dimensional mesh reconstruction," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1584–1591, July 2018.
[23] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Conference on Computer Graphics and Interactive Techniques*, 1996, pp. 303–312.
[24] A. Hilton, A. Stoddart, J. Illingworth, and T. Windeatt, "Implicit Surface-Based Geometric Fusion," *Computer Vision and Image Understanding*, vol. 69, no. 3, pp. 273–291, Mar. 1998.
[25] W. Lorensen and H. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *ACM SIGGRAPH*, vol. 21, no. 4. ACM, 1987, pp. 163–169.
[26] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2011, pp. 127–136.
[27] H. Roth and V. Marsette, "Moving Volume KinectFusion," *British Machine Vision Conference (BMVC)*, pp. 112.1—-112.11, 2012.
[28] T. Whelan, M. Kaess, and M. Fallon, "Kintinuous: Spatially extended kinectfusion," *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.
[29] J. Chen, D. Bautembach, and S. Izadi, "Scalable real-time volumetric surface reconstruction," *ACM Transactions on Graphics (TOG)*, 2013.
[30] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D Reconstruction at Scale Using Voxel Hashing," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 169:1—-169:11, 2013.
[31] C. Malleson, M. Klaudiny, A. Hilton, and J.-Y. Guillemaut, "Single-view RGBD-based Reconstruction of Dynamic Human Geometry," in *International Conference on Computer Vision (ICCV) Workshops*, 2013, pp. 307 – 314.
[32] R. Newcombe, D. Fox, and S. Seitz, "DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.
[33] K. Guo, F. Xu, T. Yu, X. Liu, Q. Dai, and Y. Liu, "Real-time geometry, albedo and motion reconstruction using a single rgbd camera," *ACM Transactions on Graphics (TOG)*, 2017.
[34] K. Guo, F. Xu, Y. Wang, Y. Liu, and Q. Dai, "Robust Non-rigid Motion Tracking and Surface Reconstruction Using L0 Regularization," in *International Conference on Computer Vision (ICCV)*, vol. 1, no. d. IEEE, dec 2015, pp. 3083–3091.
[35] M. Zollhöfer, M. Nießner, S. Izadi, and C. Rhemann, "Real-time Non-rigid Reconstruction using an RGB-D Camera," in *ACM SIGGRAPH*, 2014.
[36] P. Besl and N. McKay, "A method for registration of 3-D shapes," *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 14, no. 2, pp. 239–256, 1992.
[37] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger, "VolumeDeform: Real-Time Volumetric Non-rigid Reconstruction," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 362–379.
[38] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. O. Escolano, C. Rhemann, D. Kim, J. Taylor, P. Kohli, V. Tankovich, and S. Izadi, "Fusion4d: Real-time performance capture of challenging scenes," *ACM Transaction on Graphics (TOG)*, 2016.
[39] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic, "KillingFusion: Non-rigid 3D Reconstruction without Correspondences," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5474–5483.
[40] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion," *International Conference on 3D Vision (3DV)*, 2013.
[41] A. Tevs, A. Berner, M. Wand, I. Ihrke, M. Bokeloh, J. Kerber, and H.-p. Seidel, "Animation Cartography - Intrinsic Reconstruction of Shape and Motion," *ACM Transactions on Graphics (TOG)*, 2011.
[42] C. Russell, J. Fayad, and L. Agapito, "Energy based multiple model fitting for non-rigid structure from motion," in *Computer Vision and Pattern Recognition (CVPR)*, Jun. 2011, pp. 3009–3016.
[43] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov, "Fast Approximate Energy Minimization with Label Cost," *International Journal of Computer Vision (IJCV)*, vol. 96, no. June, pp. 1–27, 2012.
[44] D. Lazzaro and L. Montefusco, "Radial basis functions for the multivariate interpolation of large scattered data sets," *Journal of Computational and Applied Mathermatics*, vol. 140, pp. 521–536, 2002.
[45] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," *ACM National Conference*, pp. 517–524, 1968.
[46] Blender Foundation, "Blender," http://www.blender.org, accessed: July 2015.
[47] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree, "BlenSor: Blender sensor simulation toolbox," *Lecture Notes in Computer Science*, vol. 6939 LNCS, no. PART 2, pp. 199–208, 2011.

Charles Malleson is a post-doctoral researcher in 3D computer vision at the Centre for Vision, Speech and Signal Processing at the University of Surrey. He obtained his BEng (Hons) degree in Electronic Engineering from the University of Pretoria and MSc in Multimedia Technology and Systems followed by a PhD in computer vision from the University of Surrey. His research interests include dynamic scene modelling, performance capture, virtual and augmented reality.

Jean-Yves Guillemaut is a lecturer in 3D computer vision at the Centre for Vision, Speech and Signal Processing, University of Surrey. His main areas of expertise include 3D reconstruction, multi-modal registration, camera calibration, free-viewpoint video and stereoscopic content production. His current research focuses on developing novel video-based modelling techniques for the reconstruction of outdoor scenes and scenes with complex surface reflectance properties.

Adrian Hilton is professor of computer vision and graphics and director of the Centre for Vision, Speech and Signal Processing at the University of Surrey. He leads the Visual Media Research (V-Lab) in CVSSP which is conducting research in video analysis, computer vision and graphics for next generation communication and entertainment applications. A major theme of his research is to bridge-the-gap between real and computer generated imagery.