Temporally Coherent 3D Point Cloud Video Segmentation in Generic Scenes

Xiao Lin, Josep R. Casas, and Montse Pardàs

***Abstract*—Video segmentation is an important building block for high level applications such as scene understanding and interaction analysis. While outstanding results are achieved in this field by state-of-the-art learning and model based methods, they are restricted to certain types of scenes or require a large amount of annotated training data to achieve object segmentation in generic scenes. On the other hand, RGBD data, widely available with the introduction of consumer depth sensors, provides actual world 3D geometry compared to 2D images. The explicit geometry in RGBD data greatly helps in computer vision tasks, but the lack of annotations in this type of data may also hinder the extension of learning based methods to RGBD.**

In this paper, we present a novel generic segmentation approach for 3D point cloud video (stream data) thoroughly exploiting the explicit geometry in RGBD. Our proposal is only based on low level features, such as connectivity and compactness. We exploit temporal coherence by representing the rough estimation of objects in a single frame with a hierarchical structure, and propagating this hierarchy along time. The hierarchical structure provides an efficient way to establish temporal correspondences at different scales of object-connectivity, and to temporally manage the splits and merges of objects. This allows updating the segmentation according to the evidence observed in the history. The proposed method is evaluated on several challenging datasets, with promising results for the presented approach.

***Index Terms*—Video segmentation, RGBD data, point clouds, 3D connectivity, hierarchical segmentation.**

I. INTRODUCTION

Segmentation is an essential task in computer vision. It usually serves as the foundation for solving higher level problems such as object recognition, interaction analysis and scene understanding. Image segmentation is traditionally defined as partitioning the image into a set of segments showing some sort of pixel homogeneity. The segments obtained in low level segmentation are expected to be perceptually more meaningful than raw pixels, as in [35], [8]. The low level segments also produce a simplification of the image, which can be exploited by higher level segmentation and classification approaches in order to produce even more meaningful regions, *ideally* corresponding to semantic objects in the scene. To achieve this goal, high level knowledge or supervision is usually incorporated into the process, using object models [7] or large databases containing fully annotated data as, for instance, in label transfer [25] and convolutional networks [26] approaches.

Video object segmentation extends image segmentation to video frames while considering the temporal coherence of object segments. As in the case of image segmentation, most video segmentation methods are based on prior knowledge. For instance, strong priors based on accurate object annotation for the first frame are required in [32], [37] for initialization. The advantage of introducing prior knowledge is obvious, as it provides clear targets/models for the system to perform robust segmentation in the following video frames. However, most computer vision applications involve large amounts of data with different types of scenes containing several objects, which make these methods difficult to adapt to generic applications. In this situation, some methods attempt to leverage the supervision in more generic ways, such as [13]. They train a classifier to determine whether an image region is an "object-like" region or not, making the approach more generic to different types of objects and scenes. But these approaches strongly rely on the support of a large amount of annotated data for training, which may not be available in some cases.

On the other hand, the widespread availability of RGBD data from consumer depth sensors provides the possibility to work with explicit 3D geometry (i.e. point clouds instead of images). This richer information from actual 3D data in the real world can be exploited to improve segmentation. However, the lack of annotations for RGBD datasets compromises the application of data-eager supervised video segmentation methods for RGBD data. In this case, larger attention has been drawn on unsupervised methods, which exploit only generic features to cope with the semantic segmentation task in generic RGBD scenes. Unsupervised approaches generate meaningful segments by the analysis of generic features in space and time, such as connectivity, geometry and motion. Due to their genericity, unsupervised segmentation could also serve as a building block for more powerful supervised approaches in the future.

A. Related Work

Unsupervised methods for video segmentation [15], [17], [1], [18], [21] mainly focus on tackling two problems: a generic representation, which abstracts the raw data from scratch in order to extract more representative features, and a method to establish temporal correspondences, which makes the segmentation along a video sequence more stable with respect to occlusions and object interactions.

Representation: Generic object level representations are proposed to incrementally learn an object model along a sequence. For instance, Husain et al [17] maintains a quadratic surface model to represent the object segments in the scene. Each region is updated along the sequence using a split-and-merge procedure which includes an adaptive mechanism for the creation and removal of segments. Similarly, a Gaussian Mixture Model (GMM) is used in [21] to represent objects.

But these object level representations still rely on a good initial configuration for the model and, since they are incrementally updated in each frame, they are usually very sensitive to segmentation errors which may easily propagate along time.

More recently, several methods represent the raw data with a pool of object-like regions [22], [27], [41]. These object proposals are extracted from each frame based on generic spatial features. Then, temporal relations and motion are used to extract the primary object segment in the video sequence by means of optimization techniques. In [41], for instance, the selection is formulated as the longest path problem for a Directed Acyclic Graph. The work in [10] uses this approach for co-segmentation, introducing in this case RGBD images. In this line, and working with RGBD videos, the work in [9] proposes to select the significant objects from a pool of object proposals through graph optimization, introducing objectness, motion and RGBD video saliency to evaluate the importance of each object proposal. These approaches are generally computationally expensive, mainly due to the cost of the proposal generation process, and can not handle a varying number of objects in the sequence.

Hierarchical representations are employed in some methods, such as [2], [40], [28], [15], to represent the raw data from coarse to fine. A hierarchical representation usually starts from the segments at a relatively fine level, such as super-pixels or over-segmented regions recovered from a contour probability map [2] for RGB data, or super-voxels [30], [6] in the case of RGBD data. Then, the segments are gradually grouped into coarser level regions following strategies like Binary Partition Tree (BPT) [38], Multiscale Combinatorial Grouping [3] or Shape-Space Filtering [40]. One advantage of hierarchical representations in video object segmentation is that temporal correspondences can be established at different scales, which benefits the temporal coherence for the video object segmentation task considered afterwards. On the other hand, exploiting temporal coherence also helps to better construct the hierarchical representation at each frame. For instance, in [28], long term trajectories are leveraged to help building BPTs.

Establishing temporal correspondences: The video segmentation task establishes temporal correspondences based on the representation discussed above. For hierarchical representations [15], [12], temporal connectivity is a simple criterion to build the temporal correspondences between segments in successive frames. However, this might not be a suitable strategy at the finest level of the hierarchy. Scenes with fast moving objects may not be temporally consistent and fail to show temporally connected segments at this finest level when over-segmentation is independently performed for each frame, hampering the process of building temporal correspondences. The temporal correspondence can also be built based on optical flow [12], but the global optimum in the correspondence building task is still difficult to achieve considering the large scale of the problem. Instead, a local optimum is usually accepted as a solution for the correspondence problem, which makes the established correspondences less reliable in the upcoming analysis.

Methods based on object proposals [41], [10], [9] fully connect the object proposals in consecutive frames and convert the temporal correspondence building task into an optimization problem on the graph. However, they highly rely on the object proposal generator, and usually require a large (and potentially redundant) set of proposals to keep the possibility of having the proper object proposals and their temporal correspondences across all frames.

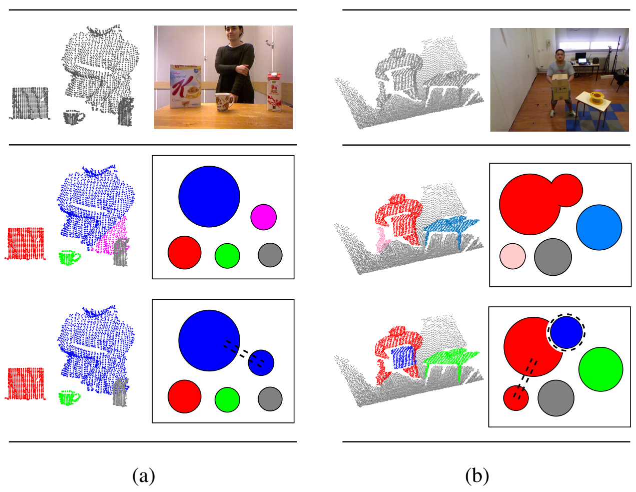
Fig. 1. First row: input point clouds and color images. Second row: segmentation errors (false split in purple on left, false merge in red on the right) in challenging scenes with occlusions/self-occlusions or object interactions, and the corresponding sketch maps shown besides. Third row: Our segmentation result by analyzing generic features in spatio-temporal domain to handle the challenges without introducing neither strong prior knowledge nor initialization, together with the corresponding sketch maps.

B. Our Proposal

Motivated by the discussion above, we propose a method that works with 3D point clouds obtained from RGBD stream data. It fully exploits the 3D geometry and temporal information in order to extract video objects and analyze their interaction in an unsupervised way. The proposed segmentation approach is *generic*, as it defines *objects* as "*compact point clouds*" in the 3D-space plus time domain. It allows point clouds corresponding to an object to break into different compact sub-clouds due to occlusions, or to merge with point clouds corresponding to other objects when they become spatially close (this is what we call *object interaction*). Fig. 1(a) shows an example where the *object* "human" breaks into two compact clouds (blue and purple) due to self-occlusion, while Fig. 1(b) shows an example where the objects "human" and "box" become spatially close and merge in a compact point cloud (in red).

We propose a *hierarchical representation* of the raw point cloud data to cope with these situations considering 3D spatial connectivity, and exploit the temporal information by building the *temporal correspondences* between the hierarchical structures in successive frames. But, in contrast to [15], [12], we do not construct the hierarchy from a relatively fine level. We rather start at a much higher level formed by blobs, segments, components and objects, as explained later in Section III, so that the task of building temporal correspondences can be solved globally as an optimization problem thanks to the reduced problem scale. Building temporal correspondences at a higher level does not affect object segmentation, since only object correspondences are concerned rather than object details in the segmentation task. Then, based on the established temporal correspondences, objects in a given frame are defined according to the evidence observed up to this frame.

A preliminary conference version of this work appeared in [24], presenting video object segmentation on RGBD stream data captured by a static camera in a predefined Space of Interest (SoI), where background objects were removed by manually setting the SoI. In this paper we avoid manual selection of the SoI by exploiting a point cloud transformation which defines point cloud connectivity for the whole scene rather than just for objects in the SoI as in [24]. This also allows us to detect and remove supporting planes more efficiently in the scene, as a previous step before object segmentation. In addition, we improve the segmentation result by introducing a fully connected CRF [20], in order to cope with more generic and challenging scenes involving complex backgrounds and camera motion, and extend our approach to a new application for the selection of "significan" objects. Comparison experiments are made between the proposed approach and two RGBD based generic object segmentation approaches [17], [9] on several benchmark datasets.

The rest of the paper is organized as follows: in section II we define point cloud connectivity and the detection of compact point clouds in a single frame. Section III presents the framework for the temporally coherent segmentation approach based on compact point clouds. Results are shown in section IV, and section V discusses the results and yields conclusions.

II. Single Frame Compact Point Cloud Detection

Individual pixel depths in RBG-D images captured by a consumer depth sensor can be transformed into a 3D point cloud from the camera intrinsic parameters. Our approach aims to segment objects in RGBD video sequences by detecting "compact point clouds" in 3D space plus time. We aim to fully exploit geometry in terms of 3D spatial connectivity and temporal correspondences. Let us first introduce the definition of spatial connectivity in a 3D point cloud.

A. *Spatial Connectivity in Point clouds*

Unlike the well organized space of image coordinates, a point cloud is a set of scattered 3D points. Spatial connectivity among those 3D points can simply be defined by a distance threshold. However, this would provide an excess of connectivity information which is not necessary for object segmentation and would significantly enlarge the data structure. In addition, point clouds obtained from a single RGBD camera have a significant drawback: point density, i.e. details available about scene geometry, falls rapidly with increasing distance from the camera. This prevents the definition of spatial connectivity using a single distance threshold. On the other hand, noise produced in the depth capturing process leads to depth estimation errors, which also affects the construction of spatial connectivity in point clouds. Point cloud filtering methods (see [14] for details) are usually applied to reduce noise in the point cloud.

In our approach, we follow the method introduced in [6] and [30] to robustly construct spatial connectivity for point clouds. We first compensate the decreasing point density and quantization with increasing depth by using the coordinate transformation in [6]. Next, we build a super-voxel representation [30] on the transformed point cloud. When building super-voxels of a point cloud, a grid voxel filtering step is first performed to organize the 3D space into voxel grids. In this manner, noise in a point cloud is somehow reduced by representing the points in a voxel with the voxel center. The spatial connectivity is defined at the super-voxel level, rather than on the raw point cloud. This also allows to start the segmentation from a higher level.

In practice, given a point cloud C, we transform C with the reversible transformation $T(C) \to C' : x' = x/z, y' = y/z, z' = log\,(z)$. The division of the x and y coordinates by z compensates for the perspective transformation [6], equalizing the point density in the $x-y-$plane across the depth range. Transforming the z coordinate helps to deal with the effects of depth quantization by compressing points as depth increases. The transformed point cloud C' is organized using a voxel grid, and voxels are grouped into super-voxels considering 1) their distance in 3D space, 2) their color similarity, and 3) local 3D shape similarity. The color of each voxel is represented by the mean color of the points in that voxel grid. To measure the shape similarity, the local 3D shape of each voxel is represented by Fast Point Feature Histograms (FPFH) introduced in [34]. We finally represent the transformed point cloud C' as a super-voxel graph $G\,(v, e)$, in which nodes $v_i \in v$ are super-voxels (homogeneous sub-cloud patches), and edges $e_{i,j}\,(v_i, v_j) \in e$ define the adjacency of patches. In this manner, a point cloud is simplified as a graph of super-voxels where important boundary information is kept.

The spatial connectivity in a point cloud is therefore dealt with as the connectivity among the super-voxels in the graph. Fig. 2 shows an example of the super-voxels generated from a point cloud. The spatial connectivity built on super-voxels represents the geometry of the scene finely enough. We also evaluated different graph building methods in our previous work [23], concluding that the method in [29] outperforms other methods such as [36] for an object segmentation task.

Based on the defined spatial connectivity, we call **compact point cloud** a point cloud represented by a set of super-voxels, when the graph built with this set of super-voxels is a connected graph.

B. *Compact Point Cloud Detection*

Due to occlusions and object interactions, and unless prior knowledge is introduced, detected compact point clouds in a single frame will not usually correspond to objects, as shown in Fig. 1. Instead, we attempt to obtain, for each single frame, those connected components that are not part of the background in the whole graph of the scene. The obtained connected components are our first approximation of the objects in the scene, which will then be refined exploiting

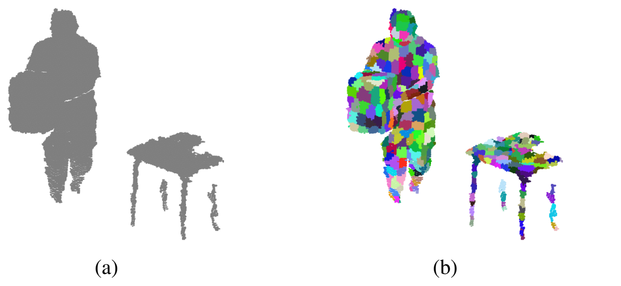

Fig. 2. An example of the super-voxels generated in our approach from a point cloud. (a) The original point cloud. (b) The super-voxels (each super-voxel is labeled with a random color).

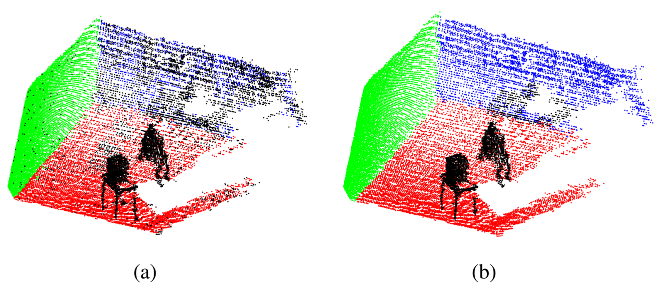

Fig. 3. An example of plane detection results. The points in red, green and blue belong to detected planes. The points in black are the points in the "not on any plane" class. (a) Plane detection result from [16]. (b) Plane detection result from our approach.

temporal information: splits, merges and object interactions. In practice, we build a super-voxel graph to represent the point cloud in each frame as explained in Section II-A. Then, we apply a plane detection technique to eliminate large plane shaped regions that may correspond to background in the point cloud. Finally, we extract connected components in the remainder of the graph by analyzing spatial connectivity.

1) Detection of Background Planes: In most scenes, foreground objects are captured together with (spatially connected) background such as floor, ceiling or walls. The background point cloud serving as supporting planes for foreground objects usually connects isolated foreground objects, thus preventing segmentation via connectivity analysis. Therefore, we remove large plane shaped regions before analyzing the connectivity in the point cloud by applying a 3D plane detection technique on the point cloud.

A plane detection method was proposed in [16], which categorizes the points on the cloud into $n+1$ classes corresponding to **plane 1...n** and "**not on any plane**" in two steps. First, a local surface normal vector for each point on the cloud is estimated by finding two vectors which are tangential to the local surface within the neighborhood of this point on the image plane. From these two tangential vectors, the normal is computed using the cross product. Then, these points are clustered in a voxelized normal space in order to obtain clusters of points with similar local surface normal orientation, while discarding clusters of small size. Secondly, each of the obtained clusters is split into plane clusters, so that each of the new clusters resembles a single plane. Thus, for each point in a cluster, the distance between the origin to the plane crossing this point with the averaged and normalized normal of this cluster is computed. Then a similar clustering step is applied in the distance space to classify points into different plane clusters. The obtained n plane clusters represent the detected n classes of planes, and the discarded clusters form the **not on any plane** class.

However, simply applying the plane detection method in [16] may fail when the point cloud is noisy (see Fig. 3(a)). So, based on the plane detection result, we build another layer modeled as a Conditional Random Field (CRF) [1] on the super-voxel graph to provide extra robustness to the noise in the point cloud considering its spatial connectivity. In this layer, we propose to label the nodes (super-voxels) in the graph with the $n+1$ labels using a unary data energy, defined on the basis of the detected planes, and a pairwise smoothness energy, defined on the basis of the graph structure. The energy in the CRF model is then optimized via graph cut [4] using alpha expansion [5] to obtain the best labelling for the graph.

In practice, given the $n+1$ classes for points on a cloud obtained from the plane detection result and its super-voxel graph G, the energy function, $E^p(\cdot)$, is formulated as the summation of the unary data energy μ^p and the pairwise smoothness energy ρ^p. In Eqs.1,2 and 3, $v_i \in v$ and $e_{i,j}(v_i, v_j) \in e$ stand for a node and an edge on the graph $G(v,e)$ respectively, l^p stands for a labeling that assigns each node $v_i \in v$ a label l_{v_i} in the label set $L^p = \{1,...,n+1\}$:

$$E^p(l^p) = \underbrace{\sum_{v_i \in v} \mu^p_{v_i}(l_{v_i})}_{unary\ energy} + \underbrace{\sum_{(v_i,v_j)\in e} \rho^p_{v_i,v_j}(l_{v_i}, l_{v_j})}_{pairwise\ energy} \quad (1)$$

The unary data energy measures the cost of assigning l_{v_i} to node v_i given the observed data. In our case, it depends on the percentage of points in point cloud C_{v_i} which are clustered to class l_{v_i} in the plane detection process, denoted as $C_{l_{v_i}}$. $NoP(C)$ computes the number of points in a point cloud C.

$$\mu^p_{v_i}(l_{v_i}) = 1 - \frac{NoP(C_{l_{v_i}})}{NoP(C_{v_i})} \quad (2)$$

The pairwise smoothness energy specifies the cost of assigning different labels to v_i and v_j connected by $e_{i,j}$. It is defined as the cosine similarity between the normals of these two nodes ($\overrightarrow{N_{v_i}}$ and $\overrightarrow{N_{v_j}}$), since edges connecting nodes with high normal difference usually coincide with boundaries. Note that the normal of a node (super-voxel) is computed by averaging the normals estimated with [16] at points belonging to this super-voxel.

$$\rho^p_{v_i,v_j}(l_{v_i}, l_{v_j}) = \begin{cases} max\left(\frac{\overrightarrow{N_{v_i}}\cdot\overrightarrow{N_{v_j}}}{\left\|\overrightarrow{N_{v_i}}\right\|\cdot\left\|\overrightarrow{N_{v_j}}\right\|}, \frac{\overrightarrow{N_{v_i}}\cdot-\overrightarrow{N_{v_j}}}{\left\|\overrightarrow{N_{v_i}}\right\|\cdot\left\|\overrightarrow{N_{v_j}}\right\|}\right) & l_{v_i} \neq l_{v_j} \\ 0 & otherwise \end{cases} \quad (3)$$

The energy function in Eq. 1 is optimized via a graph cut method [4] to generate the best labelling. The nodes (super-voxels) labeled as 1...n form n plane classes respectively.

[1]Traditional CRF models involve a unary energy μ and a pairwise energy ρ, which respectively represent the degree that one node belongs to a label and the strength of an edge connecting two nodes.

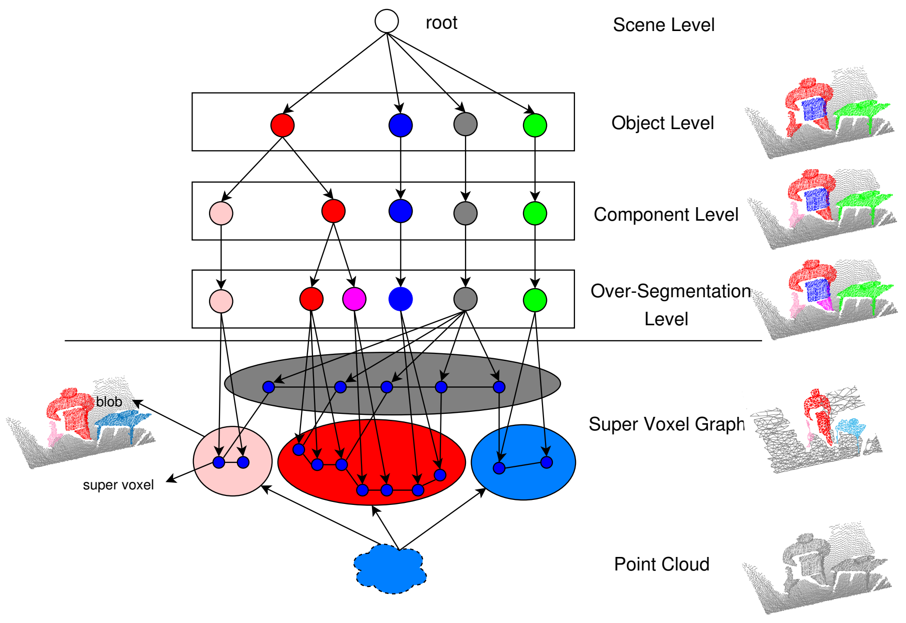

Fig. 4. The hierarchical structure built for a point cloud. Different nodes at the same level in the hierarchy are labeled with different colors. The point cloud beside it is labeled with the same color of its related node.

Fig. 3 compares an example of planes detected by the method in [16] and in our approach, showing higher robustness to noise in the point cloud.

C. Extracting Connected Components

After removing the nodes labeled as $1..n$ in the graph, we build a new graph with the **"not on any plane"** nodes. We extract m connected components on the new graph by analyzing the spatial connectivity. The lower part of Fig. 4 presents a simplified example of our single frame compact point cloud detection process, in which the blue circles and the edges between them stand for the graph representation built on the input point cloud. The ellipses marked in different colors show the detected compact point clouds in one frame. We denote those $n+m$ compact point clouds with the name "**blob**" in the rest of this paper. Note how the m non-plane like blobs (in red, pink and blue) are spatially connected to the floor blob (in grey). The elimination of the floor allows us to obtain non-plane like blobs as connected components in the graph.

III. Temporally Coherent 3D Segmentation

In this section, we explain how the 3D point cloud video segmentation task is tackled by modelling the detected blobs in each frame with the proposed hierarchical structure and propagating this structure along time. The single frame compact point cloud detection method described in Section II exploits only the spatial connectivity in one frame to extract blobs, which ideally correspond to objects. In real cases, the point cloud corresponding to an object can split in different blobs due to occlusions/self-occlusions, or can merge in a single blob with the point cloud corresponding to other objects due to what we call object interactions. In a single frame analysis, it is difficult to produce proper object segmentation without introducing prior knowledge. To tackle the problem of object splits/merges while keeping our approach generic, we propose to introduce temporal coherence when a stream of RGBD data is available. More specifically, we segment the detected blobs in each frame into meaningful sub-clouds and represent these sub-clouds in a hierarchical fashion. Then, we associate temporally these sub-clouds in the hierarchies in order to maintain the trajectories for them, and to analyze the correlation along time, always without explicit object models or accurate initialization to keep the genericity of our approach, and in order to make the best possible decision with the accumulated information at a given time.

A. Hierarchical Representation

Before introducing the proposed hierarchical structure, let us first define the terms and concepts that we use. Given a super-voxel graph $G(v,e)$ at time t and the object segmentation on it $\bigcup_{i=1}^{M_o} G_{o_i} = G$, $G_{o_k} \cap G_{o_q} = \emptyset$ for $k \neq q \in [1, M_o]$, **blobs** are the connected components on the super-voxel graph G, where $\bigcup_{i=1}^{M_b} G_{b_i} = G$, $G_{b_k} \cap G_{b_q} = \emptyset$ for $k \neq q \in [1, M_b]$. For each object o_i, $i \in [1, M_o]$, we define its **components** c_j^i, $j \in [1, M_c]$ as the connected components on G_{o_i}, where $\bigcup_{j=1}^{M_c} G_{c_j^i} = G_{o_i}$, $G_{c_k^i} \cap G_{c_q^i} = \emptyset$ for $k \neq q \in [1, M_c]$. For each component c_j^i, we over-segment it into **segments**, where $\bigcup_{u=1}^{M_s} G_{s_u^{i,j}} = G_{c_j^i}$, $G_{s_k^{i,j}} \cap G_{s_q^{i,j}} = \emptyset$ for $k \neq q \in [1, M_s]$.

We build the hierarchical structure as a tree, in which 4 levels varying from coarse to fine represent the object segmentation at different scales of object-connectivity. The upper part of Fig. 4 shows the hierarchical structure for a point cloud. Note that colors are used to differentiate nodes

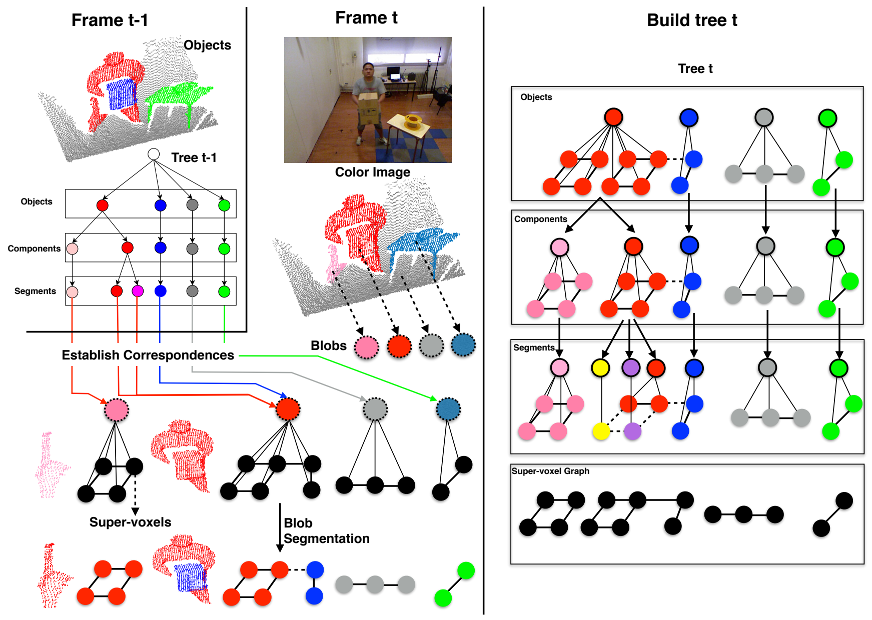

Fig. 5. An example of hierarchical structure creation. Upper-left: Object segmentation at $t-1$ and its hierarchical structure; Middle column: color image at t, detected blobs in the point cloud at t, illustrations about the establishment of correspondences and blob segmentation process; Right: hierarchical structure building process at t.

at the same level in the hierarchy and the point clouds plotted beside it are marked in the same color of their related nodes. The root of the tree represents the **scene**. The second level of the tree is the **object level**, in which each node stands for one object in the object segmentation. The merges between objects are handled at this level by maintaining the similarities among objects along time. The next level, named **component level**, is employed to handle potential splits of point clouds representing these objects. Thus, an object is represented by more than one component when it splits in different blobs. Components from different objects can be part of the same blob, because of the interactions between objects. Splits of an object are managed by maintaining the similarities among the components of this object along time. Managing object splits and merges in this manner provides a way to update the object segmentation according to the evidence observed up to time t. The final level of the tree is the **over-segmentation level**. We over-segment components into segments using normalized cut in their graphs in order to correctly establish correspondences between hierarchies along time and update its structure, that is, to obtain temporally coherent object labelling. However the amount of segments generated at this level (finest level in our hierarchy) is much less than the finest level employed in methods like [15], [12].

B. Hierarchical Structure Creation

In Section III-A, we have represented the detected blobs in one frame hierarchically, given the object segmentation of this frame. In this section, we explain how we obtain the object segmentation taking into account the objects segmented in the previous frame to create the hierarchical representation for the current frame. We model the segmentation task as a label assignment problem, in which we build temporal correspondences to label the super-voxels in the current frame considering the object labelling in the previous frame. But instead of following the method in [15], [12], [1] to build the temporal correspondences at a very fine level (super-voxels in our case), we propose to first relate the object labels in the previous hierarchy to the blobs detected in the current frame by minimizing an assignment energy defined on the difference of point cloud size and displacement. Then we split the blobs associated to more than one object label by modeling the problem as a multi-label segmentation task with a fully connected CRF [20]. There are several advantages for this method:

- Object labels in the previous hierarchy are associated to blobs in the current frame via a small number of segments at the finest level, which strongly reduces the scale of the problem of building temporal correspondences
- The task of building temporal correspondences can be solved globally as an optimization problem due to the reduced problem scale
- The temporal consistency problem can be easily addressed by generating sufficient segments at the over-segmentation level in the previous hierarchy

The object segmentation in the first frame is obtained by simply taking the detected blobs (i.e. connected components on the super-voxel graph) as the object segmentation, because no prior information whatsoever about the objects is provided. Accordingly, we create one component for each object and over-segment each component into segments.

1) Establishing Temporal Correspondences: Apart from the first frame, we obtain the object segmentation in a temporal

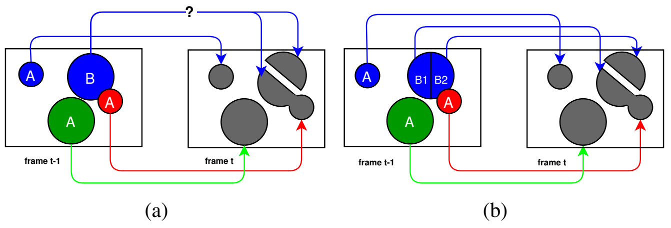

Fig. 6. An example of temporal inconsistency problem. (a) The problem when establishing the correspondences between components in the previous frame and blobs in the current frame. (b) Using the segments instead of components solves this problem.

coherent way by considering both the blobs detected in the current frame and the object segmentation in the previous frame. Fig. 5 shows an example of how we create the hierarchy for frame t. The object segmentation at $t-1$ and its hierarchical representation are shown at the upper-left corner of Fig. 5. For the data in frame t, we show the color image and detected blobs (labeled in different colors) in the middle. Then the point clouds of the blobs are represented as circles with a dash border, each of which consists of several super-voxels (e.g. the red blob consists of 6 super-voxels, as shown below). We also show the corresponding point cloud beside the blobs.

Our goal is to label the super-voxels in different blobs at frame t with the object labels contained in the hierarchy at $t-1$, and to obtain the object segmentation at t from this labelling. In our approach, a correspondence is made between the blobs in the current frame and the segments at the over-segmentation level of the hierarchy at $t-1$. This is a first step to warranty the temporal continuity of the segmentation in the video objects. Fig. 5 shows the label assignment process, in which object labels in hierarchy $t-1$ are associated to blob labels at t. The color of the lines, linking segments in hierarchy $t-1$ and blobs at t, correspond to the object nodes in hierarchy $t-1$, which indicates that object labels are associated to blobs via segments. The over-segmentation level in the tree is employed to tackle temporal consistency problems. Fig. 6(a) shows an example of this, where objects are marked in different colors and their components are denoted with letters. The component B of the blue object in frame $t-1$ splits into two blobs in frame t. In this case, no correct association is found between components at $t-1$ and blobs at t. The problem may be tackled by over-segmenting the component B of the blue object into segments $B1$ and $B2$ (shown in Fig. 6(b)) and associating the segments in frame $t-1$ with blobs in frame t.

Given the M_b blobs b_i detected in frame t, M_s segments s_i in frame $t-1$ and their corresponding point clouds (C_{b_i} and C_{s_i}), establishing the correspondence between the blobs and the segments is a problem of assigning M_b blob labels to M_s segments. This can be interpreted as a process of allocating balls (segments) into baskets (blobs), which finds the best allocation (temporal correspondences) between segments and blobs. In order to cope with possible segments moving out of the scene, we create a virtual empty blob b_{out}. This allows assigning b_{out} to any segment in the correspondence building process, which implicitly represents the segments moving out of the scene. The labels assignment task is a nonlinear integer programming problem. We solve it using a Genetic Algorithm [39] to minimize an energy function $E^{as}(\cdot)$, which is composed of three terms representing the appearance changes E^a, the displacements E^d and the penalty when objects move out of the scene E^o.

$$E^{as}\left(l^{as}\right) = E^a + E^d + E^o \tag{4}$$

where l^{as} is an assignment proposal which assigns each segment s_i a label l_{s_i} in the label set L^{as}. E^a stands for the overall appearance difference between each of the M_b blobs b_i and its corresponding segments $\left\{s_j \mid l_{s_j} = b_i\right\}$ under l^{as}. In practice, the appearance difference is defined as a size measure, by computing the difference on the number of points between them. $NoP\left(C\right)$ counts the number of points in the point cloud C.

$$E^a\left(l^{as}\right) = \sum_{i=1}^{M_b} \left| NoP\left(C_{b_i}\right) - \sum_{\left\{s_j \mid l_{s_j}=b_i\right\}} NoP\left(C_{s_j}\right) \right| \tag{5}$$

E^d represents the overall displacement for moving each of the M_s segments s_j at $t-1$ to the location of its corresponding blob b_i at t under l^{as}. Specifically, we employ the Hausdorff distance $d_h\left(\cdot\right)$ to compute the displacement between point clouds.

$$E^d(l^{as}) = \sum_{i=1}^{M_b} \sum_{\left\{s_j \mid l_{s_j}=b_i\right\}} d_h\left(C_{s_j}, C_{b_i}\right) \tag{6}$$

E^o stands for a penalty when segment s_j moves out of the scene, that is, when b_{out} is assigned to it. In this case, we calculate the Euclidean distance $d_e\left(\cdot\right)$ between the centroid of C_{s_j} to the closest boundary among the predefined $M_{\mathbb{P}}$ boundaries $\mathbb{P}_i$, which are set with respect to the field of view of the camera.

$$E^o(l^{as}) = \sum_{\left\{s_j \mid l_{s_j}=b_{out}\right\}} \min_{i=1\ldots M_{\mathbb{P}}} \left(d_e\left(C_{s_j}, \mathbb{P}_i\right)\right) \tag{7}$$

2) Blob Segmentation: In order to obtain the object segmentation, we still need to segment the blobs when segments corresponding to different objects in the previous frame are related to the same blob. For example, in Fig. 5, we perform segmentation in the red blob, where three segments that correspond to two different objects are related to it. Segmenting the red blob produces two partitions which respectively correspond to the red object and blue object in tree $t-1$. In this manner, each partition is related to only one object in the previous frame. These partitions and the blobs related with only one object label form the object segmentation for the current frame.

In our approach, we formulate the blob segmentation problem as a node labelling task on its related graph $G_b\left(v, e\right)$, in which nodes are super-voxels and edges show the adjacency of super-voxels. We label each of the super-voxels v_i in the graph with object labels o_i related to this blob. This is usually achieved by employing Conditional Random Field (CRF) models [33], [19]. Traditional CRF models involve a unary energy μ and a pairwise energy ρ, which respectively represent the degree that one node belongs to a label and the strength of an edge connecting two nodes. Minimizing the CRF energy produces the optimal labelling on the graph, that is also the segmentation of the blob. However, the pairwise energy is only computed for neighboring nodes on the graph in traditional CRF models, which makes the boundaries between different labels favor the "thinner" part of the graph with less edges. To overcome this limitation, we employ a fully connected CRF [20] model in our system. In the fully connected CRF model, the pairwise energy is established on any pair of nodes on the graph, which makes the "shape" of the graph less critical to the optimal labelling of the graph.

In practice, the energy function in the fully connected CRF model is modeled as:

$$E^s\left(l^s\right) = \underbrace{\sum_{v_i \in v} \mu^s_{v_i}\left(l_{v_i}\right)}_{unary\ energy} + \underbrace{\sum_{(v_i,v_j)\in e} \rho^s_{v_i,v_j}\left(l_{v_i}, l_{v_j}\right)}_{pairwise\ energy} \quad (8)$$

where l^s stands for the labelling which assigns each super-voxel v_i a label l_{v_i} in the label set L^s. We follow the unary energy defined in [23], where the unary energy of labelling node v_i with object label o_j, $\mu_{v^s_i}\left(l_i = o_j\right)$ is proportional to the mean distance between node v_i in the current frame and the k-nearest nodes labeled by o_j in the previous frame. For the pairwise energy, we extend the one defined in [20] for nodes representing pixels on 2D images to an energy which is suitable for nodes representing 3D point clouds. Specifically, we adopt an appearance and an smoothness term balanced with weights ω_1 and ω_2. The appearance energy term is defined as the 3D Euclidean distance $d_e\left(C_{v_i}, C_{v_j}\right)$ between the centroids of two point clouds and the color distance $d_{rgb}\left(C_{v_i}, C_{v_j}\right)$ as the difference between the mean color of each point cloud in the Gaussian kernel $\exp\left(-\frac{d(\cdot)}{2\sigma^2}\right)$. The smoothness energy term is defined as the 3D Euclidean distance between the centroid of two point clouds in the Gaussian kernel.

$$\rho^s_{v_i,vj}\left(l_i, l_j\right) = \begin{cases} \omega_1 \exp\left(-\frac{d_e\left(C_{v_i}, C_{v_j}\right)}{2\sigma^2_\alpha} - \frac{d_{rgb}\left(C_{v_i}, C_{v_j}\right)}{2\sigma^2_\beta}\right) & \\ \quad + \omega_2 \exp\left(-\frac{d_e\left(C_{v_i}, C_{v_j}\right)}{2\sigma^2_\gamma}\right) & l_i \neq l_j \\ 0 & otherwise \end{cases} \quad (9)$$

As shown in Eq. 9, ω_1 and ω_2 are used to balance the appearance energy and smoothness energy. σ_α, σ_β and σ_γ control the scale of the Gaussian kernel. The energy function in Eq. 8 is minimized using an efficient message passing implementation based on the mean fields approximation and high dimensional filtering [20]. The optimum represents the best labelling on graph G_b, which also corresponds to the segmentation of the blob. Each partition in this segmentation is related to an object in the previous frame.

The object segmentation for the current frame is then formed by the partitions and blobs related with only one object label. Then, the hierarchy in the current frame is built based on the current object segmentation, starting from the object level to the component level following the criteria explained in Section III-A (see the right part in Fig. 5), while the correspondences between the current hierarchy and previous hierarchy are established in the object and component level accordingly.

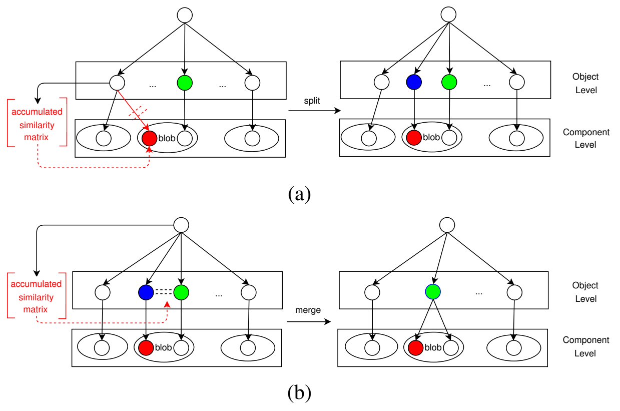

Fig. 7. Example of how we update the object segmentation in the current frame by dynamically managing object splits (a) and merges (b)

3) Dynamic Management of Merges and Splits: In the current hierarchy obtained in Section III-B2, we have the segmented objects for the current input data at the object level of the tree, which are temporally coherent with the segmented objects in the previous frame. However, they may not represent the proper object segmentation, since no accurate initialization is guaranteed at the beginning of this process in our approach. That is to say, the segmented objects need to be further analyzed along the time, in order to cope with the errors in the previous information. In this case, we exploit the established correspondences and analyze the behaviors of related nodes in hierarchies along time. More precisely, we maintain similarities between nodes at the component and object level respectively and update the object segmentation in the current hierarchy based on it. The component similarities are measured among components belonging to the same object, while the object similarities are measured among all objects. These similarities are computed by considering the distances between the point clouds of components C_c or objects C_o, which reveal the likelihood of object splits and merges. In our approach, the similarity between point cloud C and C^* is inversely proportional to the shortest distance between the two point clouds. The shortest distance $d_s\left(C, C^*\right)$ is measured based on the corresponding graphs (G and G^*) built on C and C^*, in which we search for the shortest distance between nodes v_i in G and nodes v_j in G^*. ψ is a normalizing factor which normalizes distances between two point cloud smaller than ψ while forcing the similarity between point clouds equal to zero when the distance is larger than ψ.

$$Sim\left(C, C^*\right) = \begin{cases} 0 & d_s\left(C, C^*\right) > \psi \\ 1 - \frac{d_s\left(C, C^*\right)}{\psi} & otherwise \end{cases} \quad (10)$$

$$d_s\left(C, C^*\right) = \min_{v_i \in G, v_j \in G^*} d_e\left(v_i, v_j\right) \quad (11)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2018.2811541, IEEE Transactions on Image Processing

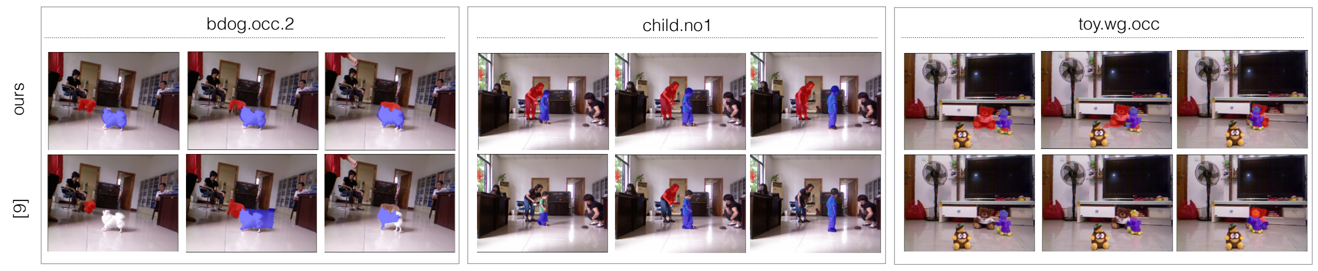

Fig. 8. Qualitative results in RGBD video foreground segmentation dataset

We accumulate the similarities along time by averaging the current similarity and the previous accumulated similarity using the correspondences built at component and object level between trees. The accumulated similarity reveals the likelihood of object splits and merges regarding the evidences observed up to the current frame. Then object splits and merges are confirmed by thresholding the accumulated similarities regarding two thresholds, Th_s and Th_m. More precisely, a split for an object is confirmed when one or several components of it have the accumulated similarities to the rest of its components smaller than Th_s. Then a new object node is created in the tree for the split components. Similarly, a merge is confirmed among objects when they are spatially connected and the similarities among those objects are larger than Th_m. Fig. 7 shows an example of object merge and split. In Fig. 7(a), the red component splits from its parent object and a new object marked in blue is created. In Fig. 7(b), the blue object is merged with the green object, since they are physically connected and the accumulated similarity between them is larger than Th_m. The components of these two objects are all connected to the one with larger size (the green object) while the one with smaller size (the blue object) is removed from the hierarchy.

Name	nFrames	[9]	ours
basketball2.2	40	42.60	55.13
bdog.occ2	20	59.24	78.98
br.occ.0	34	50.40	78.15
child.no1	47	54.34	84.26
dog.no.1	20	48.81	66.39
studentcenter2.1	23	20.34	58.81
toy.car.no	35	38.51	62.19
toy.green.occ	31	64.66	78.83
toy.wg.occ	56	86.45	72.07
tracking4	41	56.22	89.67
walking.no.occ	23	61.04	69.19
zcup.move.1	36	64.07	79.96
average	34	53.90	72.80

TABLE I
IOU SCORES FOR 12 SEQUENCES IN RGBD VIDEO FOREGROUND SEGMENTATION DATASET REPORTED IN [9] AND FOR OUR METHOD

C. Over Segmentation

The over-segmentation level in our hierarchical structure is employed to tackle the temporal consistency problem. In our approach, we over-segment components into segments to ensure that correct correspondences can be made between segments in the current hierarchy and the blobs detected in the next frame. Since the number of segments strongly affects the complexity of the task of building temporal correspondences, it is inappropriate to employ over-segmentation methods like super-voxels [29], which generate a large number of segments. In this case, we propose a normalized cut based over-segmentation method working with the super-voxel graph of a component. In this graph, each node represents a super-voxel, and each edge is weighted by a measure of compactness between the two super-voxels it connects. We assume that any split will gradually reduce the compactness of the connections in the graph. So we perform a normalized cut iteratively in this graph to generate sub-graphs which are less compactly connected, that is, anticipating possible splits in the next frame.

For a pair of connected super-voxels v_i and v_j in graph G_c, we first define the touching points $TP_{i,j}$ between them as the points in one point cloud with the closest Euclidean distance to the points in the other point cloud, smaller than a threshold Th_t. Then, the connection compactness $CC\left(v_i, v_j\right)$ between v_i and v_j is defined as the percentage of touching points between them.

$$CC\left(v_i, v_j\right) = \frac{NoP\left(TP_{i,j}\right)}{NoP\left(C_{v_i}\right) + NoP\left(C_{v_j}\right)} \tag{12}$$

A normalized min cut [35] is performed on the graph iteratively, thus creating one segment node in each iteration until the cut cost is larger than a threshold Th_c.

IV. Experiments

We evaluate the proposed method on two datasets for RGBD video segmentation: the RGBD video foreground segmentation dataset [9] and the Human Manipulation dataset [31]. Apart from that, we also test our approach on 3 sequences provided in [17] for comparison, and some other sequences without ground truth labelling for additional qualitative results. Note that all the RGBD videos used in our experiments have the same resolution (640 by 480 for both color images and depth maps), and the voxel grid used for building the super-voxel graph is $1cm^3$.

A. Comparison Experiments on RGBD Video Foreground Segmentation Dataset

The RGBD video foreground segmentation dataset [9] contains 12 RGBD sequences captured in 7 different types of

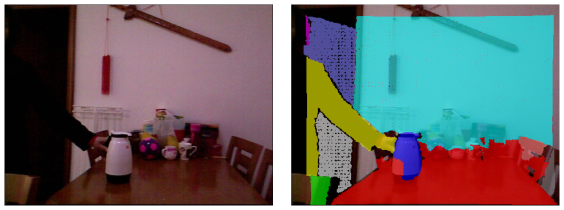

Fig. 9. An example of the segmentation result in our method: (a) a color image (b) related segmentation mask

scenes with multiple objects. The first two columns of Table I specify the name of the 12 sequences and the corresponding number of frames. Challenges in these sequences are scenes with occlusions, interactions between objects, fast moving objects and camera movement. The ground truth labeled at pixel level for multiple objects is given for one out of every 5 frames. The authors also provide the results of their method in this database, which is based on the selection through graph optimization within a pool of object proposals using RGBD data [9]. We compare our method with the results provided in this dataset by employing the average Intersection Over Union (IOU) [11] to measure the segmentation performance:

$$IOU = \frac{1}{M_o}\sum_{j=1}^{M_o}\max_i \frac{GT_j \cap R_i}{GT_j \cup R_i} \tag{13}$$

For each frame, M_o stands for the number of objects labeled in the ground truth, GT_j is the ground truth for object j and R_i represents the object proposals in the frame. Table I compares the IOU scores of the segmentation results of our approach with those obtained in [9], while Fig. 8 shows some qualitative results from both methods. In [9], the authors present the comparison between a number of other methods and their method, which achieves the best results in the RGBD video foreground segmentation dataset. Our approach achieves better average IOU score: 72.80% compared to 53.90% in [9] over the 12 sequences, as well as better average IOU in almost all sequences except “toy.wg.occ”. Note that, in Fig. 8, we only show the object proposals obtained with our approach which are related to the objects labeled in the ground truth to be able to compare with the method in [9]. In fact, our approach segments the whole point cloud of the scene and obtains all foreground objects and their supporting planes, which are labeled accordingly as shown in Fig. 9.

Application: Selection of Significant Objects. In this experiment, we aim to select the significant objects from the object proposals obtained by our method along a sequence. This is usually achieved by evaluating the importance of object proposals along time based on some object attributes. In [9], Fu et al. propose to select significant objects from a pool of object proposals through graph optimization, in which objectness, motion, RGBD video saliency are involved to evaluate the importance of each object proposal. In our case, we employ the Frobenius norm of optical flow gradients [41], which is the same motion term used in [9], to represent the importance of an object proposal. Then, the evaluation is simply performed by averaging the importance of an object proposal in each frame and selecting objects with higher importance along the sequence, since the temporal correspondences are made for all object proposals when they are generated in our approach. We compare our approach with [9] on the RGBD video foreground segmentation dataset. The qualittative comparison in Fig. 10 shows better results than [9]. Our approach generates less but more accurate object proposals in each frame, which allows the system to establish object proposal correspondences online and simplifies the significant object selection problem.

Name	[17]	ours
seq1	99.3	99.5
seq2	82.1	86.0
seq3	77.4	91.8
average	84.8	92.8

TABLE II
SEGMENTATION ACCURACY OF OUR APPROACH AND THE ASMS FOR THE 3 SEQUENCES PROVIDED IN [17].

B. Comparison Experiments for Sequences in [17]

For comparison, we employ 3 more sequences proposed in [17] and perform our approach against the Adaptive Surface Models based 3D Segmentation method (ASMS) in [17]. Table II shows a quantitative comparison between our approach and ASMS in these 3 sequences. Sequence 1 contains a scenario of a human hand rolling a green ball forward and then backward with the fingers. Sequence 2 involves a robot arm grasping a paper roll and moving it to a new position. Sequence 3 describes a scenario in which a human hand enters and leaves the scene, displacing the objects rapidly. We evaluate the segmentation result by global accuracy. Global accuracy counts the percentage of pixels which are correctly labeled with respect to the ground truth labelling. The comparison results show that the proposed approach outperforms ASMS in all 3 sequences. It also illustrates one of the drawbacks in ASMS. In sequence 3, rapid object movement leaves little or no overlap of corresponding segments for ASMS to build temporal correspondences between objects and update the object models. However, our method shows a higher robustness to cope with rapid movements, since the temporal correspondences are built by finding the global optimum of an assignment energy.

C. Ablation Experiments on Human Manipulation Dataset

To evaluate the improvements on performance when introducing Dynamic Management of Merges and Splits (DMMS), we employ 5 RGBD sequences in the Human Manipulation dataset with the 3D point cloud ground truth labelling in consecutive frames provided in [23]. Each of the sequences contains 201 frames of human manipulation actions which involve object interactions and self-occlusions/occlusions. The super-voxel based graph representation organizes the input point cloud with voxels in 3D producing a voxelized point cloud, while the ground truth is labeled in the original cloud. Therefore, we extend our segmentation result on the original cloud by simply finding k-nearest neighbors for each point on the original point cloud from our segmentation result.

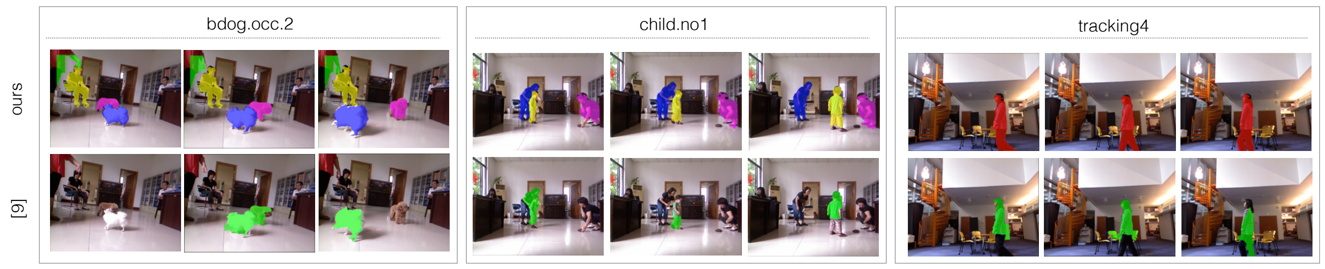

Fig. 10. Qualitative results of significant objects selection in RGBD video foreground segmentation dataset

	without DMMS	with DMMS
Seq.1	93.95	96.87
Seq.2	85.87	94.17
Seq.3	91.16	96.63
Seq.4	82.28	92.55
Seq.5	88.39	90.46
average	88.33	94.14

TABLE III
IOU SCORES FOR 5 SEQUENCES PRODUCED BY OUR METHOD WITHOUT DMMS AND WITH DMMS.

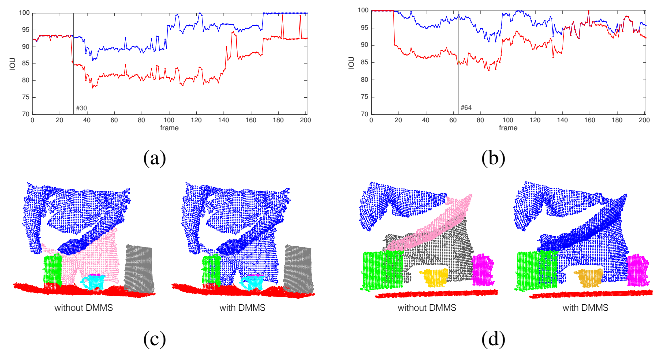

Fig. 11. (a)-(b) present the IOU (vertical axis) per frame (horizontal) results for Seq 2-3. Red: our approach without DMMS, Blue: with DMMS.(c)-(d) present point cloud plots in frame 30 of Seq 2 and in frame 64 of Seq 3, object proposals are marked in different colors.

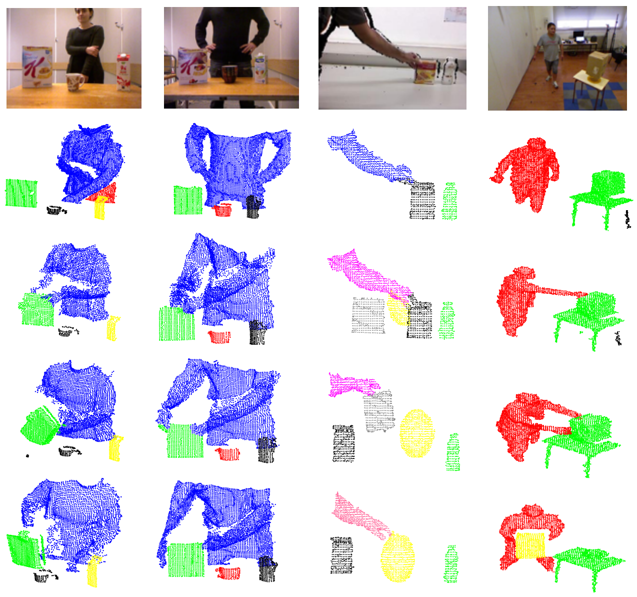

Fig. 12. Qualitative results of the proposed method. Column 1-2: from human manipulation dataset in [31], Column 3: from data in [17] and Column 4: from data recorded by ourselves.

This allows using the majority voted label among k-nearest neighbors as the label for this point. Similarly, we employ the average intersection over union to measure and compare the performance of our approach with and without introducing DMMS in each frame.

Table III shows the segmentation performance of our approach in the 5 sequences with or without employing DMMS. In the case of "without DMMS", we do not maintain the similarities between corresponding nodes in the tree structures to update the object proposals along time. Table III exploiting DMMS provides improvements on segmentation performance in all 5 sequences (around 6% improvement in average IOU scores), which proves that DMMS contributes in the low level to the better segmentation of actual objects in the scene. In Fig. 11(a)-11(b), we present the IOU score per frame in 2 of the 5 sequences. The point cloud view in Fig. 11(c)-11(d) show that the torso of the human body splits into two parts (marked in blue and pink in the left point cloud in Fig. 11(c)) due to the self-occlusion, which leads to an improper segmentation in frame 30 of Seq 2. However, this is handled by DMMS which analyzes the correlations between those two parts in the history and maintains the similarity between them to produce a proper segmentation in the point cloud (shown as the right point cloud in Fig. 11(c)). Fig. 11(d) presents a similar situation in frame 64 of Seq 3, which also shows the importance of introducing DMMS. Fig. 12 presents more qualitative results produced by our approach. We manually remove some segments in the background for the clarity of the illustration. Each row in Fig. 12 shows the segmentation results in 4 frames of a sequence, which are uniformly sampled along the sequence. More visual results showing the dynamic behavior of the presented method are available on https://imatge.upc.edu/web/node/1910.

D. Computational Cost

There are three main parts where the computational power is spent in our approach: the optimization for the multi-label assignment for the establishment of temporal correspondences,

	time	problem scale
Assignment Optimization	2.3s	$\sim$ 15 segments, $\sim$ 8 blob labels
Over-Segmentation	0.252s	$\sim$ 200 super-voxels $\sim$ 1500 edges
Blob Segmentation	0.021s	$\sim$ 250 super-voxels $\sim$ 1800 edges

TABLE IV
RUN-TIME PERFORMANCE OF THE PROPOSED APPROACH.

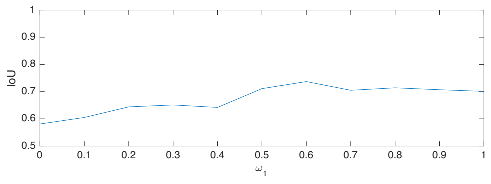

Fig. 13. IoU scores for 20 validation images under different settings of ω_1

the fully connected CRF method used in the blob segmentation and the graph cut technique in the over-segmentation process. The main problem of approaching the temporal correspondences association by a multi-label assignment problem is the computation complexity. The problem scale increases exponentially with the number of labels. However, the number of labels is well controlled in our approach by finding a suitable over-segmentation level so that we can achieve the assignment task in a small scale while not leading to the temporal inconsistency problem. In the experiments, generally 20 segments are involved in the assignment task in each frame. In the fully connected CRF method [20], the energy function is optimized using an efficient message passing method based on the mean fields approximation and high dimensional filtering, which makes the complexity of the approximated inference process sublinear in the number of the edges in the model. The graph cut technique used in our approach has the reported computation complexity $O\left(v^2 \cdot sqrt\left(e\right)\right)$ where v stands for the number of vertices and e the number of edges on the graph. Table IV shows the run-time performance of the three main parts in our approach.

E. Implementation Details

In this section we analyze the parameters used in the implementation of the proposed system. In the first place, the balance factors for the appearance and smoothness energies, ω_1 and ω_2 in Eq. 9, are learned from a small set of validation images. In practice, we select 20 examples, where blob segmentation is needed. For each example, we provide the ground truth object segmentation in the previous frame and build the previous hierarchy based on the ground truth object segmentation. Then we follow the proposed method to segment the current frame. The weights in Eq. 9 are then learned by searching for the best segmentation performance over these 20 training examples under different configuration of the weights. We set $\omega_1 + \omega_2 = 1$ and search for the best configuration of ω_1 from 0 to 1 with step length 0.1. Fig. 13 shows the best segmentation result is obtained when ω_1 is set to 0.6, and ω_2 is set to 0.4. One advantage of dealing with actual 3D data is that the point cloud maintains the real size of objects in the scene, which provides a more clear physical meaning for the related parameters. In our experiments, we fix σ_α and σ_γ to 0.3 meter with respect to their physical significance. σ_β is set to 13 following [20]. Th_s and Th_m are the two parameters used for confirming the split and merge of an object by thresholding the component and object similarity. Thus, Th_s and Th_m are set to $1/3$ and $2/3$, which splits the similarity interval $[0, 1]$ into 3 zones (similar, neutral and not similar). Th_t and Th_c are the parameters used in over-segmentation. Th_t is a 3D distance threshold specifying the touching points between two point clouds. Th_c represents a graph cut cost threshold when performing normalized cut on the connection compactness graph. In our experiment, 0.07 meter is set for Th_t and 0.03 is set for Th_c.

V. CONCLUSION

In this paper, we have introduced a generic and temporally coherent 3D point cloud segmentation method for segmenting objects from generic scenes in RGBD videos. We exploit temporal coherence by representing the generic point cloud segmentation in a single frame with a tree structure, and propagate it along time. Based on the tree structure representation, we generate temporally coherent object proposals at different scales of object-connectivity and establish reliable temporal correspondences between them. The behaviors of the temporally related nodes in the tree structures built along time are further analyzed to produce improved object proposals.

We evaluate the performance of the proposed approach with the RGBD video foreground segmentation dataset and the Human Manipulation data set, and compare it with state of the art. Our approach generates a better segmentation result based on all low-level features available. This guarantees it to be generic, as no explicit or learnt model of the objects or the scene are introduced in the proposed method.

REFERENCES

[1] A. Abramov, K. Pauwels, J. Papon, F. Wörgötter, and B. Dellen. Depth-supported real-time video segmentation with the kinect. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 457–464. IEEE, 2012.
[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.
[3] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 328–335, 2014.
[4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.
[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.
[6] S. Christoph Stein, M. Schoeler, J. Papon, and F. Worgotter. Object partitioning using local convexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311, 2014.
[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.
[8] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.
[9] H. Fu, D. Xu, and S. Lin. Object-based multiple foreground segmentation in rgbd video. *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, 2017.

[10] H. Fu, D. Xu, S. Lin, and J. Liu. Object-based rgbd image co-segmentation with mutex constraint. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4428–4436, 2015.

[11] H. Fu, D. Xu, B. Zhang, S. Lin, and R. K. Ward. Object-based multiple foreground video co-segmentation via multi-state selection graph. *IEEE Transactions on Image Processing*, 24(11):3415–3424, 2015.

[12] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2141–2148. IEEE, 2010.

[13] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[14] X.-F. Han, J. S. Jin, M.-J. Wang, W. Jiang, L. Gao, and L. Xiao. A review of algorithms for filtering the 3d point cloud. *Signal Processing: Image Communication*, 2017.

[15] S. Hickson, S. Birchfield, I. Essa, and H. Christensen. Efficient hierarchical graph-based segmentation of rgbd videos. In *CVPR2014*. IEEE Computer Society, 2014.

[16] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke. Real-time plane segmentation using rgb-d cameras. In *Robot Soccer World Cup*, pages 306–317. Springer, 2011.

[17] F. Husain, B. Dellen, and C. Torras. Consistent depth video segmentation using adaptive surface models. *Cybernetics, IEEE Transactions on*, 45(2):266–278, 2015.

[18] M. Keuper, B. Andres, and T. Brox. Motion trajectory segmentation via minimum cost multicuts. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3271–3279, 2015.

[19] P. Kohli, P. H. Torr, et al. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

[20] V. Koltun. Efficient inference in fully connected crfs with gaussian edge potential. *Adv. Neural Inf. Process. Syst*, 2(3):4, 2011.

[21] S. Koo, D. Lee, and D.-S. Kwon. Incremental object learning and robust tracking of multiple objects from rgb-d point set data. *Journal of Visual Communication and Image Representation*, 25(1):108–121, 2014.

[22] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *2011 International Conference on Computer Vision*, pages 1995–2002. IEEE, 2011.

[23] X. Lin, J. Casas, and M. Pardàs. 3d point cloud segmentation oriented to the analysis of interactions. In *The 24th European Signal Processing Conference (EUSIPCO 2016)*. Eurasip, 2016.

[24] X. Lin, J. R. Casas, and M. Pardàs. 3d point cloud video segmentation based on interaction analysis. In *Computer Vision–ECCV 2016 Workshops*, pages 821–835. Springer, 2016.

[25] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *Pattern Analysis and Machine Intelligence*, 33(12):2368–2382, 2011.

[26] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[27] T. Ma and L. J. Latecki. Maximum weight cliques with mutex constraints for video object segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 670–677. IEEE, 2012.

[28] G. Palou and P. Salembier. Hierarchical video representation with trajectory binary partition tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2099–2106, 2013.

[29] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2027–2034. IEEE, 2013.

[30] J. Papon, T. Kulvicius, E. E. Aksoy, and F. Wörgötter. Point cloud video object segmentation using a persistent supervoxel world-model. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3712–3718. IEEE, 2013.

[31] A. Pieropan, G. Salvi, K. Pauwels, and H. Kjellstrom. Audio-visual classification and detection of human manipulation actions. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3045–3052. IEEE, 2014.

[32] X. Ren and J. Malik. Tracking as repeated figure/ground segmentation. In *Computer Vision and Pattern Recognition, 2007.*, pages 1–8. IEEE, 2007.

[33] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.

[34] R. B. Rusu, A. Holzbach, N. Blodow, and M. Beetz. Fast geometric point labeling using conditional random fields. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 7–12. IEEE, 2009.

[35] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[36] X. Suau, J. Ruiz-Hidalgo, and J. R. Casas. Detecting end-effectors on 2.5 d data using geometric deformable models: Application to human pose estimation. *Computer Vision and Image Understanding*, 117(3):281–288, 2013.

[37] D. Tsai, M. Flagg, A. Nakazawa, and J. M. Rehg. Motion coherent tracking using multi-label mrf optimization. *IJCV*, 100(2):190–202, 2012.

[38] V. Vilaplana, F. Marques, and P. Salembier. Binary partition trees for object detection. *IEEE Transactions on Image Processing*, 17(11):2201–2216, 2008.

[39] D. Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[40] Y. Xu, E. Carlinet, T. Géraud, and L. Najman. Hierarchical segmentation using tree-based shape spaces. *IEEE transactions on pattern analysis and machine intelligence*, 39(3):457–469, 2017.

[41] D. Zhang, O. Javed, and M. Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 628–635, 2013.