3D Point Cloud Video Segmentation Based on Interaction Analysis

Xiao Lin, Josep R.Casas and Montse Pardàs

Image Processing Group,
Technical University of Catalonia (UPC)
james.lin.xiao@tsc.upc.edu

Abstract. Given the widespread availability of point cloud data from consumer depth sensors, 3D segmentation becomes a promising building block for high level applications such as scene understanding and interaction analysis. It benefits from the richer information contained in actual world 3D data compared to apparent (projected) data in 2D images. This also implies that the classical color segmentation challenges have recently shifted to RGBD data, whereas new emerging challenges are added as the depth information is usually noisy, sparse and unorganized. In this paper, we present a novel segmentation approach for 3D point cloud video based on low level features and oriented to the analysis of object interactions. A hierarchical representation of the input point cloud is proposed to efficiently segment point clouds at the finer level, and to temporally establish the correspondence between segments while dynamically managing the object split and merge at the coarser level. Experiments illustrate promising results for our approach and its potential application in object interaction analysis. ...

Keywords: object segmentation, 3D point clouds, dynamic split and merge management, object interactions

1 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. Traditionally, segmentation is defined as a process of grouping homogeneous pixels into multiple segments on a single image, which is also known as low level segmentation. The obtained segments are somehow more homogeneous and more perceptually meaningful than raw pixels. Based on that, the concept of semantic segmentation/labeling is proposed. It is devoted to segment an image into regions which *ideally* correspond to meaningful objects in the scene. To achieve this goal, high level knowledge is usually incorporated into the segmentation process, such as object models [2] exploited in constrained scenes, accurate object annotations required in the initialization [12, 16] and large databases containing fully annotated data in, for instance, label transfer approaches such as [9]. These approaches yield outstanding segmentation results; however, most computer vision applications involve large amounts

of data with different types of scenes containing several objects, which difficult the adaptation to generic scenes of those methods based on manual initialization or predefined/learned object models.

To generalize the methodology from constrained situations, larger attention has been drawn on investigating the spatial relation between segments and their temporal correspondences when temporal video (stream) data is available. A bunch of methods focusing on the spatio-temporal relation between segments are proposed [4, 15, 3, 5, 6, 1]. These methods mainly focus on tackling two problems: a higher level representation, which abstracts the raw data from scratch, and a method to establish the spatio-temporal correspondences. Several methods employ a generic model to represent the objects in the scene. Husain et al. [6] maintains a quadratic surface model to generally represent the object segments in the scene. The model is then updated along the sequence to obtain the final segmentation result. But it is difficult to handle objects with large displacement in successive frames. Similarly, a Gaussian Mixture Model (GMM) is used in [8] to represent the objects, while the model is incrementally updated for new frames in the sequence. However, it establishes the correspondence between the object model and the point cloud in the new frame by using the Iterative Closest Point (ICP) technique, which may lead to the accumulation of registration errors in the object model due to the deformation of the objects. More generally in scene representation, Richtsfeld et al. [13] propose to represent the 3D point cloud with a graph of surface patches detected in the scene, such as planes and non-uniform rational B-splines (NURBs). A SVM based learning process is then employed to decide the relation between surface patches for a sub-sequent graph cut segmentation. Grundmann et al. [3] use a graph-based model to hierarchically construct a consistent video segmentation from over-segmented frames. Similarly, Hickson et al. [5] extend the method to RGBD stream data. But the over-segmentation for each frame in these two approaches is still calculated independently, without the temporal coherence constraint, which may lead to a temporal inconsistency problem due to changes of corresponding over-segments in different frames. Abramov et al. [1] perform label transfer in the pixel level between frames by using optical flow. Then, they minimize the label distribution energy in the Potts model to generate labels for objects in the scene. This establishes the temporal correspondences in the pixel level, which makes the approach highly rely on the performance of optical flow estimation.

Motivated by the problems mentioned above, we propose a segmentation algorithm based on the definition of objects as "compact point clouds" in the 3D-space plus time domain. However, point clouds corresponding to an object can break into different compact sub-clouds due to occlusions, or can merge with compact point clouds corresponding to other objects, producing a single compact point cloud, when they become spatially close (object interaction). Our system aims to produce a robust spatio-temporal segmentation of the point clouds by analyzing their connectivity to define the objects according to the evidence observed up to a given temporal point. Our primary contributions are:

- We propose a novel tree structure representation for the point cloud of the scene which allows us to temporally update the similarities between nodes in the tree
- We propose to approach the temporal correspondences establishment task by a labelling assignment problem regarding the tree structure.
- A dynamic management of object splits and merges is exploited in our approach for generating a better segmentation result based on all low-level features available
- An over-segmentation method based on the compactness of the connection between neighboring super voxels in the graph is proposed.

The rest of the paper is organized as follow. In section 2 we explain how the 3D point cloud segmentation problem is modeled. Sections 3 and 4, present the framework of the proposed segmentation approach and show experimental results, respectively. Finally, section 5 discusses the results and yields conclusions.

2 Problem modeling and definition

In this section, we explain how the 3D point cloud segmentation task is modeled by the proposed tree structure.

2.1 Tree structure representation of the point cloud

Given a stream of RGBD data, our goal is to segment the foreground point cloud in each frame into meaningful sub-clouds and associate these sub-clouds in consecutive frames to maintain the trajectories for them without explicit object models or accurate initialization. More precisely, we represent the input point cloud as a graph G (shown in Fig.1(b)) with a super-voxel approach [10]. Nodes in the graph are homogeneous sub-cloud patches and edges define the spatial connections among patches. In this manner, the connectivity of a point cloud is interpreted as the connectivity in the corresponding graph representation. The set of connected nodes in the graph corresponds to the compact parts of the input point cloud, which we call blobs (shown in Fig.1(c) and marked in different colors). Then a tree structure with 4 levels varying from coarse to fine is exploited to represent the input point cloud at different scales of object-connectivity. Fig.1(a) shows the constructed tree structure for the point cloud data in the second row. The root of the tree represents the scene. The second level of the tree is the object level, in which each node stands for an object proposal. The next level, named component level, is employed to handle potential splits and merges of point clouds representing these objects. An object is represented by more than one component if it splits in different blobs in the graph. Components from different objects can be part of the same blob, because of the interactions between objects. Splits and merges of components are managed by maintaining the similarities among object components along time, which provides a temporally coherent way to obtain object proposals based on point cloud

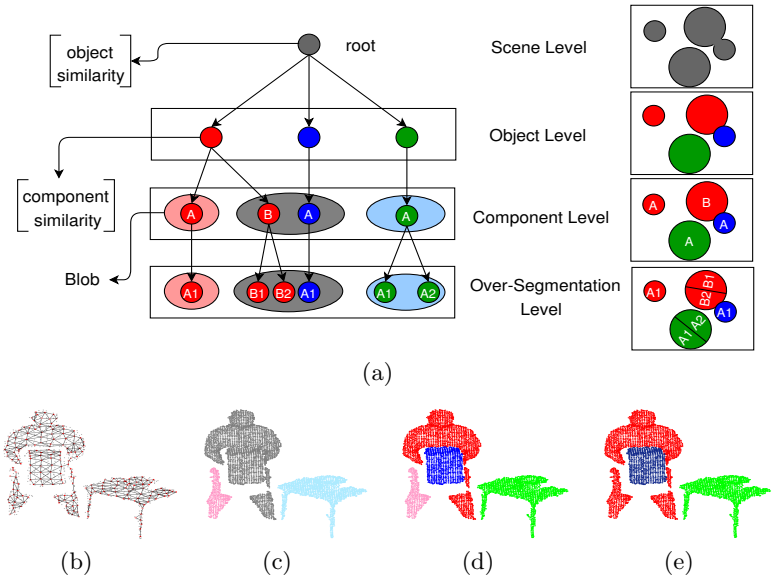


Fig. 1. An example of tree structure representation. (a) A tree structure representation of the input point cloud data. (b) The graph built on the input point cloud. (c) Blobs in the input point cloud. (d) Components for each object. (e) Objects segmentation obtained from the tree structure.

connectivity. The final level of the tree is the over-segmentation level. We over-segment components using normalised cut in their graphs in order to correctly establish correspondences between trees along time and update its structure, that is, the temporal coherent assignment of labels to the segmented objects. Note that three kinds of labels are used in Fig.1(a) to differentiate the nodes in the tree while showing their relationships, which are object label (color), component label (alphabet) and segment label (number). We aligned the color used in Fig.1(a) with the real point cloud data plots (Fig.1(c) and Fig.1(e)). In Fig.1(e), we present the object segmentation result obtained in this tree structure, while Fig.1(d) shows the components of each object from the point cloud view. Fig.1(c) presents the blobs in the input point cloud which is related to the ellipses marked with the same color in Fig.1(a).

2.2 Tree structure creation

Taking the point cloud in frame t as input data, we abstract it with super voxels, using the method proposed in [10]. The graph representation simplifies the input data by grouping homogeneous points on the point cloud into super voxels while preserving the boundary information. Then, a graph G is constructed regarding the spatial connectivity between super voxels. We group the point cloud into blobs by detecting the connected components in the graph. The tree in the first

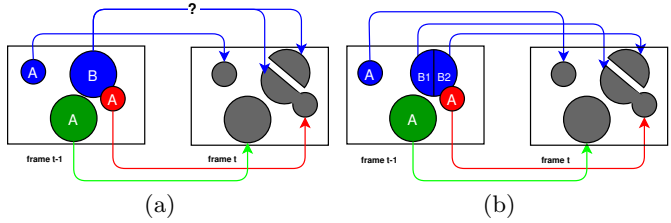


Fig. 2. An example of temporal inconsistency problem. (a) The problem when establishing the correspondence between components in the previous frame and blobs in the current frame. (b) Using the segments instead of components solves this problem.

frame is created by simply taking the detected blobs as the objects, as no prior information about the objects is provided. Accordingly, we create one component for each object and over-segment each component into segments. Apart from the first frame, the tree is built in a bottom-up way, starting at the component level. First, a correspondence is made between the connected components of the graph (blobs) in the current frame and the segments at the over-segmentation level of the tree structure in the previous frame. This over-segmentation level is employed to avoid temporal inconsistency problem. Fig.2(a) shows an example of it, where 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 and blobs. The problem may be tackled by over-segmenting the component B of the blue object into segments $B1$ and $B2$ (shown in Fig.2(b)) and associating the segments in frame $t-1$ with blobs in frame t. Establishing the correspondence between the blob labels and the segments is a problem of assigning M_b blob labels to M_s segments. This is a nonlinear integer programming problem which is solved using a Genetic Algorithm to minimize an energy function which is composed of three terms: one for representing the appearance changes E_a, one for the displacements E_d and the other one E_o for the penalty when objects move out of the scene.

A further segmentation is needed when segments that correspond to different objects in the previous frame are assigned to the same blob. A restricted graph cut method is employed to segment the graph of the blob by minimizing a segmentation energy function, in which we consider the degree that a graph cut fits the current data while being coherent with the minimum cut in the previous frame. Once the current segmentation is done, the components and objects in the current tree are created initially from it regarding the previous tree structure.

To dynamically manage object splits and merges along time, we maintain similarities between nodes at the component and object level respectively and update the tree based on it. The component similarities are measured among components which belong to the same object while the object similarities are measured among objects. These similarities are computed considering spatial distance and apprearance difference, which reveal the likelihood of object splits and merges.

We accumulate them along time by averaging the current similarity and the previous accumulated similarity regarding the established correspondences. Then object splits and merges are confirmed by thresholding the accumulated similarities. Finally, an over-segmentation is performed at the component level to generate segments for correctly establishing the correspondence to the next frame. Specifically, a normalized cut is performed in the graph representing the component iteratively until the cut cost is larger than a threshold.

3 Graph based dynamic 3D point cloud segmentation

In this section, we present the frame work of the proposed approach including data acquisition and initialization, temporal correspondences establishment and segmentation, the proposed dynamic management of object splits and merges mechanism and the over-segmentation method.

3.1 Data acquisition and initialization

We can transform the per-pixel distances provided in an RGBD image into a 3D point cloud $C_I \subseteq R^3$ using camera parameters. We focus on the interest area of the foreground cloud $C_{fg} \subseteq R^3$ in 3D space. Taking the foreground point cloud at frame t as input data, a graph representation is constructed $f(C_{fg}) \rightarrow G(v, e)$ via a graph building method f, where v is the set of vertices or nodes and e the edges of the graph. The super voxel method introduced in [10] is employed as the graph building method f in our approach. It aggregates points on the point cloud into homogeneous sub-cloud patches (super voxels) with respect to the point proximity and appearance similarity while preserving the boundary information. Then a graph G is built for the super voxels regarding their adjacency. The connectivity of C_{fg} is interpreted as the connectivity on G. Our system is initialized by building the tree structure for the first frame. Nodes in the tree are denoted as N^i_{level}, where we specify which level the node belongs to and its node number in this level. Each node is described by its related point cloud and graph $N^i_{level} \sim \left(C^i_{level}, G^i_{level}\right)$, where $C^i_{level} \subseteq C_{fg}$ and $G^i_{level} \subseteq G$. A node for the tree root is created as $N^1_{sc} \sim (C_{fg}, G)$. As mentioned in Section 2.2, we base the construction of the tree for the first frame only on the connectivity of the input point cloud. Thus, we extract blobs from C_{fg} by detecting connected components on graph G. Each blob is treated as one object proposal while accordingly we create one object node N^i_o in the tree. For each object node, one component node N^i_c is created. Components are over-segmented into M_s segments via an over segmentation method $OSeg(N^i_c) \rightarrow \left\{N^1_s \cdots N^{M_s}_s\right\}$.

3.2 Correspondences establishment and segmentation

Apart from the first frame, we create the current tree structure with respect to the tree in the previous frame Tr'. Similarly, a graph G is obtained for the current point cloud and blobs are detected on G. Then, the tree building process

is started by establishing the correspondences between the current data and Tr'. Associating a set of labels to another set of labels is treated as an assignment task, in which we optimize different assignment proposals with respect to an energy function. Since the problem scale increases exponentially with the number of labels, it is critical to limit the number of the labels. Therefore, we propose to assign the blob labels in the current frame to the segments in the previous tree Tr'. The blobs represent the few compact sub-clouds of the input point cloud. Assigning blob labels to segments reduces the problem scale while it respects the spatial disconnection between the compact sub-clouds, which may coincide with the object boundaries. The problem now becomes a task of assigning M_b blob labels in the current frame to ${M_s}'$ segments in the previous frame. To cope with the situation when objects go out of the scene, we employ a virtual blob B_{out} which stands for the space out of the interest area and does not represent any point cloud. Then, the assignment energy function is defined as:

$$E_{ass}(A) = E_a + E_d + E_o \tag{1}$$

Where A is an assignment proposal which maps a segment label in the previous frame ${l_s}'$ to a blob label l_b in the current frame, $A\left({l_s}'\right) \rightarrow l_b$. E_a stands for the summation of the energy of the appearance difference between the set of the point clouds $C_s^{j'}$ related to the segment $N_s^{j'}$ and the point cloud C_b^i related the blob B_i, where the segment label $l_s^{j'}$ is associated to the blob label l_b^i regarding the assignment A. The appearance difference is measured by comparing the number of points $NoP\left(\cdot\right)$ on the point cloud.

$$E_a(A) = \sum_{i=1}^{M_b} \left| NoP\left(C_b^i\right) - \sum_{A\left(l_s^{j'}\right)==l_b^i} NoP\left(C_s^{j'}\right) \right| \tag{2}$$

E_d is the summation of the energy of the displacement which is calculated by measuring the Hausdorff distance $dist_h\left(\cdot\right)$ between the point cloud $C_s^{j'}$ and the point cloud C_b^i, where $A\left(l_s^{j'}\right) == l_b^i$.

$$E_d(A) = \sum_{i=1}^{M_b} \sum_{A\left(l_s^{j'}\right)==l_b^i} dist_h\left(C_s^{j'}, C_b^i\right) \tag{3}$$

E_o is the summation of the distance between the point cloud of the segment $N_s^{j'}$, which is associated with blob B_{out}, to the closest boundary bd_i. The boundaries are predefined planes which are also used in the data acquisition step in Sec.3.1. The distance is calculated as the Euclidean distance ($dist_e\left(\cdot\right)$) from the centroid of the point cloud of the segment to the closest boundary plane.

$$E_o(A) = \sum_{A\left(l_s^{j'}\right)==l_b^{out}} \min_{bd_i \in Boundary} \left(dist_e\left(C_s^{j'}, bd_i\right)\right) \tag{4}$$

Optimizing this energy function is a nonlinear integer programming problem. We employed the Genetic Algorithm to solve it.
After the best assignment A^* is obtained, the blobs in the current frame are associated with segments in the previous frame. A further segmentation is needed when segments that correspond to different objects in the previous frame (according to the privous tree $T_r{}'$) are assigned to the same blob. Given a blob $B_i \sim \left(C_b^i, G_b^i\right)$ and the M unique object labels related to the associated segment labels $\left\{l_s^j \mid A^*\left(l_s^j\right) == l_b^i\right\}$, our goal is to segment the graph G_b^i into M parts. To this end, we employ the restricted graph cut approach proposed in [17] to seek for the minimal cut on graph G_b^i, which further segments this blob considering both spatial/feature homogeneity and the temporal consistency. The minimal cut is obtained by minimizing a segmentation energy function with the way introduced in [7]. The energy function for graph cut is usually defined as the summation of the data energy and the smoothness energy ($E(L) = E_{data} + E_{smooth}$), where L stands for the label proposal for the graph. The data energy is an unary energy term representing the degree that the label proposal fits the current data. The smoothness energy is a pair-wise energy which manages the label smoothness between nodes in the graph. The method proposed in [17] introduces a novel smoothness energy term considering the label smoothness regarding not only the current data but also the minimal cut obtained in the previous frame.
The further segmentation performed on the blob with segment labels related to multiple object labels yields the object partitions for this blob while establishing their correspondences to the objects in the previous tree. As mentioned in Section 2.1, an object is represented by more than one component if it splits in different blobs, in the current or previous frames. Thus, for each object partition in the blobs, we create one component for the related object. Accordingly, the correspondences between the current components and the components in the previous tree are made. Note that there is no correspondence established for the newly generated components in the current tree. In this step, the first three levels of the current tree structure are initially built regarding the established correspondences to the previous tree at each of the three levels.

3.3 Dynamic management of merge and split

Given the current tree obtained in the last step, an object proposal is implicit at the object level. This object proposal for the current input data is temporally coherent with the object proposal in the previous frame. However, it may not be a correct object proposal, since no accurate initialization is guaranteed at the beginning of this process in our approach. That is to say, this object proposal needs to be further analyzed, 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 trees along time. More specifically, we compute the similarities among the components corresponding to the same object, which forms a similarity matrix for each object in the tree. The object similarities are computed among objects in the tree producing an object similarity matrix. In

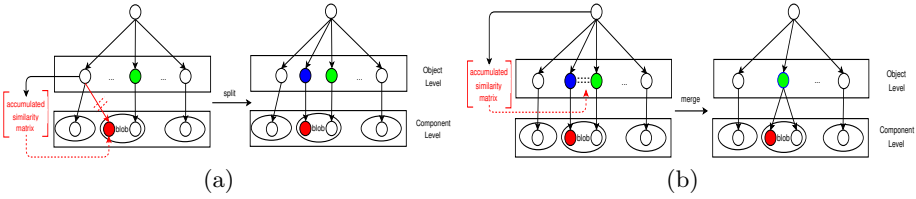


Fig. 3. Example of dynamic management of split (a) and merge (b)

our approach, the similarity between node N^i and N^j is defined as the Euclidean distance between their related point clouds in Eq.(5).

$$Sim\left(N^i, N^j\right) = \begin{cases} 0 & dist_e\left(C^i, C^j\right) > \psi \\ 1 - \frac{dist_e\left(C^i, C^j\right)}{\psi} & otherwise \end{cases} \tag{5}$$

Note that here the Euclidean distance between two point clouds is calculated as the distance between the closest point pair from them. ψ is a normalizing factor. Afterwards, the similarities are accumulated along time by averaging them with the corresponding accumulated similarity in the previous frame, in order to dynamically manage the merges and splits of objects in the scene on the fly. The accumulated similarity reveals the likelihood of object splits and merges regarding the evidence observed up to the current frame. Thus, the decisions for split and merge are made by thresholding the accumulated similarity regarding two thresholds, Th_{split} and Th_{merge}. Fig.3 shows an example how the object merge and split are dynamically managed. Specifically, a split for an object node is confirmed when a set of its child component nodes have all the accumulated similarities smaller than Th_{split} with respect to the rest of the child component nodes. Then a new object node is created as the parent node of the split component nodes. In Fig.3(a), the red component node splits from its parent object node and a new object node marked in blue is created as its new parent. A merge between object nodes is confirmed when they are physically connected while the accumulated similarities between them are larger than Th_{merge}. In Fig.3(b), a merge is confirmed between the blue object node and the green object node which is physically connected with each other. The nodes are merged to the one with the larger number of the points on the related point cloud (the green node). Their child component nodes are all connected to the green object node in the tree. Then the blue object node is removed from the tree.

3.4 Over segmentation

The first three levels in the current tree structure are built and updated in the last two steps. In this section, we introduce an over segmentation process in order to build the forth level of the tree. In the over-segmentation level, we generate segments for each component in the current tree. This is treated as the preparation for establishing the correspondence between segments in the

current tree to the blobs detected in the next frame. The more segments are generated in the current frame, the better they will respect the topology of the input point cloud in the next frame, which avoids the temporal inconsistency problem. However, the number of the segments will affect the problem scale of the assignment task in the correspondence establishment and segmentation process in the next frame. Therefore, instead of using a technique such as super voxel [10] which finely over-segments the point cloud, we propose a relatively coarser over-segmentation method to tackle this problem. Pratically, in the related graph $G_c^i(v, e)$ of a component node N_c^i, we define the touching points for connected nodes in G_c^i. The touching points TP_j^i from the node v^i to the node v^j is computed as the number of the points in v^i which have the closest Euclidean distance lower than a threshold Th_{tp} to v^j. The touching points TP_i^j from v^j to v^i is defined in the same manner. The number of touching points reveals the compactness CC between connected nodes in the graph, which is defined as:

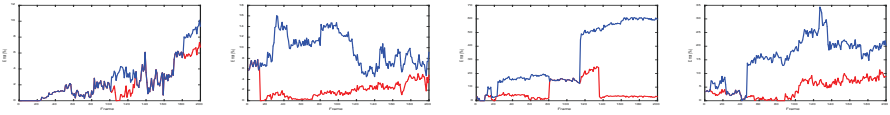
$$CC\left(v^i, v^j\right) = \frac{NoP\left(TP_j^i\right)}{NoP\left(v^i\right)} + \frac{NoP\left(TP_i^j\right)}{NoP\left(v^j\right)} \tag{6}$$

where $NoP(\cdot)$ stands for the number of the points in a graph node. We believe that any split of the point cloud will gradually lead the decrease in the number of touching points at the splitting position on the graph. Then the edges in the graph are weighted by CC and a normalized cut approach [14] is performed on the graph iteratively, which yields one segment node in each iteration until the cut cost is larger than a threshold Th_{mc}. In this manner, the component is iteratively over-segmented into segments at the positions which are less compact in the graph, which may coincide with the splits in the next frame.

4 Experiments

4.1 Segmentation result evaluation

To evaluate our approach from the 3D point cloud segmentation perspective, we select 4 sequences with 3D point cloud ground truth labeling in the human manipulation data set [11]. Each of them contains 200 frames. These 4 sequences



(a) (b) (c) (d)

Fig. 4. (a)-(d) present the segmentation results for sequence 1-4, shown as percentage of error points (vertical axis) per frame (horizontal). Red: GDS, Blue: GS.

vary from single attachment to multi-attachments, low motion to higher motion, double attached objects to multiple attached objects. The task of this experiment is to segment objects from the scene. The evaluation metrics is 3D segmentation accuracy (3D ACCU) proposed in [18] which computes the fraction of a ground truth segment that is correctly classified in our approach. Since the super voxel based graph representation method organises the input point cloud with voxels in 3D producing a down sampled point cloud, while the ground truth is labeled in the original cloud, we find K nearest neighbors for a point on the down sampled point cloud from the ground truth labeling and use majority voted label among K nearest neighbors as the ground truth labeling for this point.

Fig.4 shows the segmentation results of these 4 sequences. In each sub-figure, we plot the percentage of segmentation error against the frame number. We compare the segmentation performance of our graph based dynamic segmentation approach (GDS) with the graph based segmentation method (GS) proposed in [17] which provides a temporally coherent segmentation for RGBD stream data. The red lines in Fig.4 stand for the result of GDS while the blue lines stand for the result of GS. As mentioned in Section 3.2, GS is also integrated in our approach producing temporally coherent segmentation for blobs with more than one unique object labels. Thus, the comparison between them shows the importance of introducing the dynamic management of object split and merge mechanism. Particularly, GDS outperforms GS in all the 4 sequences shown in Fig.4, which proves that the dynamic management mechanism contributes in the low level to the better segmentation of actual objects in the scene. GDS achieves an overall foreground 3D point cloud segmentation 3.92% mean segmentation error. In Fig.4(c), we observe a dramatic increase (from 2.5% to 15%) in the segmentation error for GDS (red line). This is caused by the error in dynamic management of object merge and split. Fig.5(d)-5(f) show 3 key frames in this process, where objects are marked in different colors. The left arm of the human body in blue is confirmed to split from the torso in frame 81 due to the persistent self-occlusion. The self-occlusion breaks the human body into two compact point clouds (the left arm and the rest), which gradually decreases the accumulated similarity till the split is confirmed in frame 81. However, these two point clouds reattach to each other in frame 103, which continuously increases the accumulated similarity between them, till they are confirmed to merge in frame 136. Fig.5(j)-5(l) in the second row present another example, in which our system is incorrectly initialized in frame 1 (part of the torso marked in red is treated as one object because of the spatial disconnection caused by occlusion). They reattach to each other in frame 8 and get merged in frame 14 due to the dynamic management mechanism. These two examples illustrate the robustness of our system regarding the errors in previous frames while showing that our approach does not rely on an accurate initialization.

For comparison, we employ 3 more sequences proposed in [6] and perform our approach against the Adaptive Surface Models based 3D Segmentation method (ASMS) in [6]. ASMS maintains a quadratic surface model to generally represent the object segments in the scene. The model is updated along the sequence by

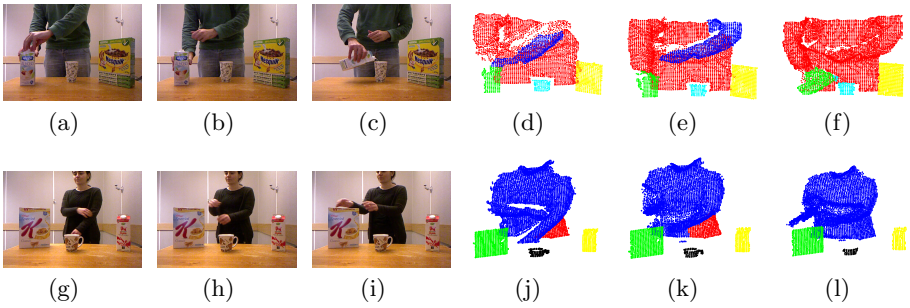


Fig. 5. Examples of dynamic management of merge and split. (a)-(c) present the color images of frame 81,103 and 136 in sequence 4, (d)-(f) show the segmentation results in these frames. (g)-(i) present the color images of frame 1,8 and 14 in sequence 2, (j)-(l) show the segmentation results in these frames.

finding and growing the overlapping area to obtain the final segmentation result. To adapt our approach to the scenes in these 3 sequences, we remove the background point cloud by using a plane fitting technique to extract foreground point clouds as input data. Fig. 6 shows a quantitative comparison between GDS 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 the human hand enters and leaves the scene, displacing the objects rapidly. The comparison results show that the proposed GDS approach outperforms ASMS in all the 3 sequences. Specifically, Fig.6(c) shows an example of the drawback in ASMS. The spikes in the blue curve are caused by rapid object movement, which leaves little or no overlap of corresponding segments for ASMS. However, our method has the robustness to rapid movements, since the correspondences establishment problem is treated as the optimization of an assignment energy.

Apart from the sequences used in the quantitative evaluation experiments, we employ 5 more sequences without ground truth labeling, recorded by ourselves with scenes displaying interactions in human manipulation scenes. Fig.7 shows

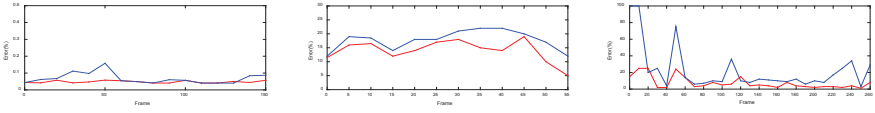


Fig. 6. Quantitative result of GDS (in red) and the ASMS (in blue) for the 3 sequences provided in [6]. From left to right, sequence 1-3.

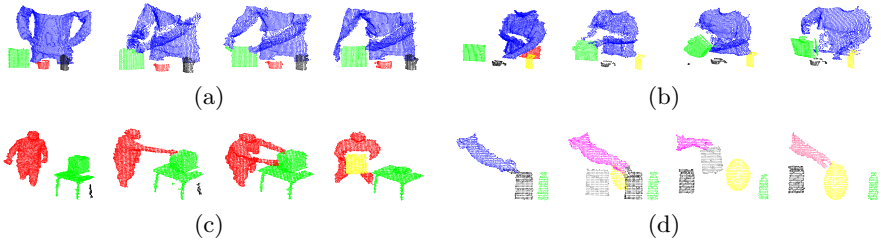


Fig. 7. Qualitative results of the proposed method: (a)-(b) from human manipulation dataset in [11], (c) from data recorded by ourselves and (d) from data in [6].

some qualitative results of our approach from 4 sequences. For each sequence, we uniformly sample 4 frames and show the segmentation result from our approach. More visual results are available on https://imatge.upc.edu/web/node/1806.

4.2 Interaction detection

Our approach is also capable to obtain the interactions between objects, which is implicit in the tree structure. As mentioned in Section 1, an interaction between objects is defined as a state when they become spatially close. In our method, an interaction is detected when a blob is related to more than one unique object label. An example of an object interaction is shown in Fig.8(a), where an interaction between a human body and a box is detected and marked as the black line connecting them. Based on this definition, we manually label the object interaction occurring in the 4 sequences used in the first experiment and calculate the times of interactions between objects in the scene detected by the proposed method. Fig.8(c) shows the interaction detection result in each sequence, where the top of each bar shows the number of interactions in the ground truth, the red line stands for the true positive detections in our approach and the blue line stands for the false positive detections. Our approach detected 742 truth positive interactions over 980 labeled interactions in the ground truth. We notice that the number of false positive detections in sequence 3 is relatively high. This is

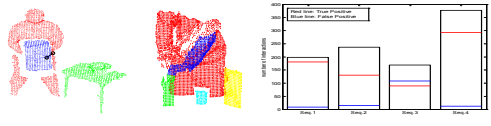


Fig. 8. (a) An example of object interaction. (b) An example of false positive detections (c) Interaction detection evaluation for 4 sequences from [11]

mainly caused by the errors in dynamic management of object split and merge process. Fig.8(b) shows the segmented point cloud in a frame of sequence 3, where an interaction is detected between the arm in blue and the juice box in green. However, the arm and the torso are not correctly merged as one object in this frame, which makes the detected interaction a false detection.

4.3 Computational cost analysis

In our approach, there are two main parts where the computational power is spent: the optimization for the multi-labels assignment for temporal correspondences establishment and the graph cut technique used in either the further segmentation or the over-segmentation process. The main problem of approaching the temporal correspondences association by a multi-labels assignment problem is the computation complexity. The problem scale increases exponentially when the number of labels grows. However, the number of the labels is controlled in our approach by finding a suitable over-segmentation level so that we can achieve the assignment task in a small problem scale while not leading to the temporal inconsistency problem. In the experiments, generally 20 segments and 5 blobs are involved in the assignment task in each frame. 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.

5 Conclusion

In this paper, we have introduced a graph based dynamic 3D point cloud segmentation method, which works at low level with a tree structure representation for segmenting generic objects in RGBD steam data. We have evaluated the performance of the proposed approach with a human manipulation data set and also compared it with the method proposed in [6]. Our approach achieves an overall 3.92% 3D point cloud segmentation error while outperforming in the comparison experiment. Our contribution can be summarized in 3 points:

- firstly, we proposed a novel over-segmentation method based on the compactness of the connection between neighboring super voxels on the graph
- then a novel tree structure representation for the scene is proposed, which allows to temporally update the similarities between nodes in the tree
- the temporal correspondences establishment task is approached by a labelling assignment problem that takes into account the appearance and displacement of the components. Based on the tree structure, a dynamic management of object splits and merges mechanism is proposed

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.

Acknowledgement: This work has been developed in the framework of the project TEC2013-43935-R, financed by the Spanish Ministerio de Economa y Competitividad and the European Regional Development Fund (ERDF).

References

1. Abramov, A., Pauwels, K., Papon, J., Wörgötter, F., Dellen, B.: Depth-supported real-time video segmentation with the kinect. In: Applications of Computer Vision (WACV), 2012 IEEE Workshop on. pp. 457–464. IEEE (2012)
2. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. Pattern Analysis and Machine Intelligence, IEEE Transactions on 32(9), 1627–1645 (2010)
3. Grundmann, M., Kwatra, V., Han, M., Essa, I.: Efficient hierarchical graph-based video segmentation. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. pp. 2141–2148. IEEE (2010)
4. He, X., Zemel, R.S., Carreira-Perpiñán, M.Á.: Multiscale conditional random fields for image labeling. In: In CVPR 2004. vol. 2, pp. II–695. IEEE (2004)
5. Hickson, S., Birchfield, S., Essa, I., Christensen, H.: Efficient hierarchical graph-based segmentation of rgbd videos. In: CVPR2014. IEEE Computer Society (2014)
6. Husain, F., Dellen, B., Torras, C.: Consistent depth video segmentation using adaptive surface models. Cybernetics, IEEE Transactions on 45(2), 266–278 (2015)
7. Kolmogorov, V., Zabin, R.: What energy functions can be minimized via graph cuts? Pattern Analysis and Machine Intelligence, IEEE Transactions on 26(2), 147–159 (2004)
8. Koo, S., Lee, D., Kwon, D.S.: 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)
9. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing via label transfer. Pattern Analysis and Machine Intelligence 33(12), 2368–2382 (2011)
10. Papon, J., Abramov, A., Schoeler, M., Worgotter, F.: Voxel cloud connectivity segmentation-supervoxels for point clouds. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on. pp. 2027–2034. IEEE (2013)
11. Pieropan, A., Salvi, G., Pauwels, K., Kjellstrom, H.: Audio-visual classification and detection of human manipulation actions. In: Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on. pp. 3045–3052. IEEE (2014)
12. Ren, X., Malik, J.: Tracking as repeated figure/ground segmentation. In: Computer Vision and Pattern Recognition, 2007. pp. 1–8. IEEE (2007)
13. Richtsfeld, A., Mörwald, T., Prankl, J., Zillich, M., Vincze, M.: Segmentation of unknown objects in indoor environments. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. pp. 4791–4796. IEEE (2012)
14. Shi, J., Malik, J.: Normalized cuts and image segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on 22(8), 888–905 (2000)
15. Shotton, J., Winn, J., Rother, C., Criminisi, A.: Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In: Computer Vision–ECCV 2006, pp. 1–15. Springer (2006)
16. Tsai, D., Flagg, M., Nakazawa, A., Rehg, J.M.: Motion coherent tracking using multi-label mrf optimization. IJCV 100(2), 190–202 (2012)
17. Xiao, L., Josep, C., Montse, P.: 3d point cloud segmentation oriented to the analysis of interactions. In: 24th European Signal Processing Conference (EUSIPCO),2016. p. (Accepted and to be published). IEEE (2016)
18. Xu, C., Corso, J.J.: Evaluation of super-voxel methods for early video processing. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 1202–1209. IEEE (2012)