Joint segmentation of color and depth data based on splitting and merging driven by surface fitting

Publisher:
Elsevier

Published version:
DOI: 10.1016/j.imavis.2017.12.004

Joint Segmentation of Color and Depth Data Based on Splitting and Merging Driven by Surface Fitting

Giampaolo Pagnutti and Pietro Zanuttigh

Department of Information Engineering, University of Padova
Via Gradenigo 6B, 35131 Padova, Italy

Abstract

This paper proposes a segmentation scheme based on the joint usage of color and depth data together with a 3D surface estimation scheme. Firstly a set of multi-dimensional vectors is built from color, geometry and surface orientation information. Normalized cuts spectral clustering is then applied in order to recursively segment the scene in two parts thus obtaining an over-segmentation. This procedure is followed by a recursive merging stage where close segments belonging to the same object are joined together. At each step of both procedures a NURBS model is fitted on the computed segments and the accuracy of the fitting is used as a measure of the plausibility that a segment represents a single surface or object. By comparing the accuracy to the one at the previous step it is possible to determine if each splitting or merging operation leads to a better scene representation and consequently whether to perform it or not. Experimental results show how the proposed method provides an accurate and reliable segmentation.

Keywords: Segmentation, Depth, Spectral Clustering, Kinect, NURBS

1. INTRODUCTION

Scene segmentation by way of images is a long-term research topic that, despite a huge amount of research, remains very challenging. There is a large number of works addressing image segmentation [1], but even the best performing ones are not able to provide a reliable solution in all conditions since it is very difficult to properly understand the underlying scene structure from a single image. The recent introduction of matricial Time-of-Flight range cameras and of structured-light consumer depth cameras (e.g., the two versions of Microsoft Kinect) has made geometry acquisition available to the mass market. Depth data is a very valuable aid for segmentation since it conveys information about the 3D structure of the scene, making the recognition of the various structures in it much easier. This allows to re-formulate the segmentation problem as the search for effective ways of partitioning a set of samples featuring color and geometry information. It resembles what happens inside the human brain where the disparity between the images seen by the two eyes is one of the clues used to separate the different objects together with prior knowledge and other features extracted from the color data acquired by the human visual system.

Following this rationale, this paper introduces a novel segmentation scheme capable to jointly exploit color and depth information. The segmentation is performed within a region splitting and merging framework. In the first stage, following the idea we introduced in a recent conference work [2], the segments are recursively split in two parts using a joint color and depth segmentation scheme based on spectral clustering [3]. In particular for the subdivision step we employ a modified version of the approach proposed in [4], exploiting orientation information in addition to geometry and color data. After the subdivision a Non-Uniform Rational B-Spline (NURBS) surface is fitted on each of the two resulting segments and the fitting accuracies before and after the splitting are compared. The motivation behind this approach is that segments representing single objects or surfaces are accurately fitted, while segments encompassing multiple surfaces will lead to a poor fitting accuracy. This procedure is followed by a bottom up recursive merging strategy that combines segments belonging to the same object [5]. The algorithm detects the neighboring segments with consistent depth, orientation and color values along the common edges, and attempts to merge them. Again, the surface fitting accuracies before and after the merging are used to evaluate which merge operations are properly joining two parts of the same object. Summarizing, this work extends our two recent conference works, i.e., the region splitting scheme of [2] and the region merging scheme of [5], and combines them into a general segmentation framework that gives better results than each of the two separate methods. In particular by combining the splitting and merging stages it overcomes some limitations of the previous works, e.g., the merging scheme was not able to recover from errors in the initial over-segmentation.

The paper is organized as follows. After presenting the related works in Section 2, Section 3 describes the general workflow of the segmentation algorithm. The joint color and depth segmentation scheme used for the various splitting operations is recalled in Section 4, while Section 5 presents the employed surface fitting algorithm. Then the two steps of the method, i.e, the recursive splitting and merging algorithms, are described in Section 6 and in Section 7 respectively. The results are presented in Section 8 while Section 9 draws the conclusions.

2. RELATED WORKS

As already pointed out image segmentation is a long-term research field and a huge number of techniques based on different insights have been proposed. Approaches based on graph theory and on clustering algorithms have been particularly successful [6, 3, 7]. Anyway, despite a huge amount of research none of the existing methods is able to obtain completely satisfactory results, since segmentation is an ill-posed problem and it is very difficult to properly estimate the scene structure from color data alone.

Even if the usage of depth data for segmentation purposes is a recent research field, several approaches addressing scene segmentation by way of color and geometry information have been proposed in the last few years. A simple solution is to perform two independent segmentations from the color and the depth data, and then join the two results as in [8].

Clustering techniques can easily be adapted to joint depth and color segmentation and different approaches based on this idea have been proposed. In [9] Mean Shift clustering [6] is applied to 6D vectors containing both the color and spatial component for each sample, while superparamagnetic clustering is used in [10]. A joint color, spatial and directional clustering method coupled with a planar region merging scheme is used in [11] and in the refined version of the same approach proposed in [12]. The method of [13] exploits instead a multi-layer clustering strategy.

A method based on graph cuts has been applied for joint color and depth segmentation in [14]. The approach of [15] exploits Normalized Cuts segmentation [3] together with saliency maps. In [4] we proposed a segmentation scheme based on normalized cuts that is capable to automatically balance the relevance of color and depth.

Region splitting and region growing methods have also been used. The work of [16] starts from an initial superpixel segmentation and joins the segments exploiting a saliency metric. Superpixels and region merging are used also in [17] that uses a graph-based approach for the merging stage. Superpixels are combined together also in [18] where regions corresponding to planar surfaces are computed using an approach based on Rao-Blackwellized Monte Carlo Markov Chain. The approach has been extended for the segmentation of multiple depth maps in [19]. A bottom-up approach is used also by [20], that builds an adjacency graph of surface patches. We exploited surface fitting in [2], where we fitted NURBS surfaces over the segments. The fitting accuracy is then used to evaluate the consistency of the segmented regions in order to further split segments not encompassing a single surface in an iterative approach. The work in [5] applies the NURBS fitting scheme within a region merging procedure, starting from an initial over-segmentation and joining adjacent segments on the basis of the fitting accuracy. These two works constitute the starting point for the approach of this paper.

Hierarchical segmentation based on the output of contour extraction has been used in [21], that also deals with object detection from the segmented data. Another combined approach for segmentation and object recognition has been presented in [22], where an initial over-segmentation based on the watershed algorithm is followed by a hierarchical scheme. Combined segmentation and labeling is also addressed in [23] that exploits a MRF superpixel segmentation associated with a tree-structured approach. Finally dynamic programming has been used in [24] to extract the planar surfaces in indoor scenes.

Some approaches deal with the close but less general problem of separating the foreground from the background [25, 26, 27, 28, 29]. In [29] two likelihood functions, based on color and depth data, are combined together for this task. The work of [25] uses two distinct Gaussian Mixture Models for the foreground in the depth and color spaces and combine them in a Bayesian Framework. Mixture of Gaussians are used in [26] as well. Both this approach and [27] consider also temporal constraints in depth and color videos. Finally some works try also to jointly solve the segmentation and stereo disparity estimation problems, e.g., [30, 31, 32].

3. GENERAL OVERVIEW

The proposed segmentation procedure is divided into 3 main steps as depicted in Fig. 1. The initial pre-processing step produces the input data for the clustering algorithm, then a recursive splitting scheme (*top-down phase*) divides the scene into smaller and smaller segments and finally a region merging algorithm (*bottom-up phase*) combines together close segments belonging to the same object.

In the first phase the color image and the depth map are converted to a unified representation. Color data, depth data and surface orientation information are represented by a set of 9D vectors containing the three sources of information.

The data is then recursively split into two segments using both color and depth information. At each step the segment being processed is divided into two sub-segments using the approach based on spectral clustering described in Section 4. Then a NURBS surface is fitted over each of the two sub-segments (Section 5). The fitting accuracy is compared to the one obtained at the previous step for the same segment. If the split operation has provided a better

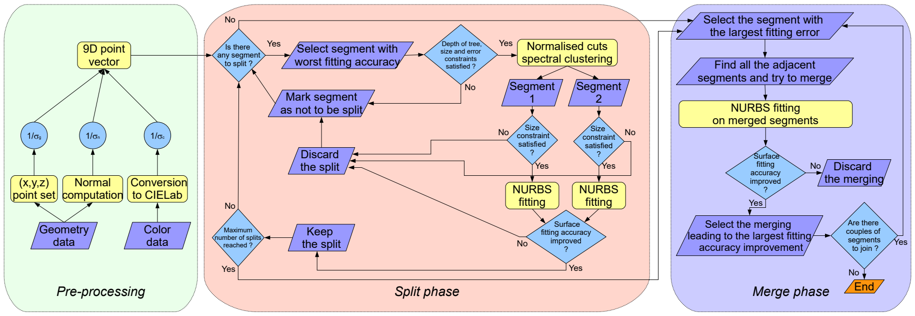

9D point vector
$1/\sigma_g$
$1/\sigma_n$
$1/\sigma_c$
(x,y,z) point set
Normal computation
Conversion to CIELab
Geometry data
Color data
Pre-processing
No
Is there any segment to split ?
Yes
Select segment with worst fitting accuracy
Depth of tree, size and error constraints satisfied ?
Yes
Normalised cuts spectral clustering
No
Mark segment as not to be split
Segment 1
Segment 2
No
Size constraint satisfied ?
Size constraint satisfied ?
No
Discard the split
Yes
Yes
NURBS fitting
NURBS fitting
No
Maximum number of splits reached ?
Keep the split
Surface fitting accuracy improved ?
No
Yes
Yes
Split phase
Select the segment with the largest fitting error
Find all the adjacent segments and try to merge
NURBS fitting on merged segments
Surface fitting accuracy improved ?
No
Discard the merging
Yes
Select the merging leading to the largest fitting accuracy improvement
Are there couples of segments to join ?
Yes
No
End
Merge phase

Figure 1: Overview of the 3 main stages of the proposed approach (construction of the data vectors, region splitting phase and region merging phase).

fitting then the process is iterated by recursively dividing the two sub-segments, otherwise it is stopped. The process is iterated until it is not possible to obtain any improvement by further subdividing any of the produced segments, as described in detail in Section 6.

At this point the algorithm starts to analyze adjacent clusters corresponding to surfaces with a common contour and similar orientation and color properties. For each couple of candidates the fitting accuracy on the union of the two segments is calculated and compared against the fitting accuracy on the original segments. If it is improved, the two segments are joined and replaced by the new resulting segment. The procedure continues until there are no more possible merging operations. This part is detailed in Section 7.

4. JOINT COLOR AND DEPTH SEGMENTATION

Following an approach similar to [5, 4, 33], before entering the proposed iterative clustering algorithm, a nine-dimensional representation of the scene samples is built from the geometry and color data. Firstly the depth and color cameras are jointly calibrated. Various methods are possible for this task [34], we employed the method of [35] for the Kinect sensor and the one of [36] for ToF sensors. After calibration it is possible to compute the coordinates (x, y, z) of each depth sample in 3D space and to reproject the depth samples on color data, thus associating to each sample also a vector containing the (R, G, B) color components. The surface orientation information, i.e, the surface normals (n_x, n_y, n_z) at each location, is also computed from the geometric data using the approach of [37]. Notice that the normals information, not considered in [4] and [2], is very useful to separate surfaces with similar colors and close spatial positions but with different orientations (e.g., the walls of a room).

In this way a 9D vector is available for each sample p_k. Since it contains heterogeneous data, the joint color, geometry and orientation information needs to be normalized to a consistent representation. Firstly, in order to give a perceptual significance to the distance between colors that will be used in the clustering algorithm, color values are converted to the CIELab perceptually uniform space, i.e., the color of each sample p_k is represented by the vector

$$\mathbf{p_k^c} = [L(p_k), a(p_k), b(p_k)]^T, \quad k = 1, \ldots, N \tag{1}$$

Geometry is simply represented by the 3D coordinates, i.e., as:

$$\mathbf{p_k^g} = [x(p_k), y(p_k), z(p_k)]^T, \quad k = 1, \ldots, N \tag{2}$$

Finally orientation information is given by the 3 components of the normal vector at each location, i.e., as:

$$\mathbf{p_k^n} = [n_x(p_k), n_y(p_k), n_z(p_k)]^T, \quad k = 1, \ldots, N \tag{3}$$

The scene segmentation algorithm should be insensitive to the relative scaling of the point-cloud geometry and bring color, geometry and normals into consistent representations. Therefore, the color data are normalized by the average standard deviation σ_c of the L, a and b components, obtaining the vectors $[\bar{L}(p_k), \bar{a}(p_k), \bar{b}(p_k)]$. Following the same rationale, geometry components are normalized by the average standard deviation σ_g of the point coordinates obtaining the vectors $[\bar{x}(p_k), \bar{y}(p_k), \bar{z}(p_k)]$. Similarly, the normal vectors $[\bar{n}_x(p_k), \bar{n}_y(p_k), \bar{n}_z(p_k)]$ are obtained by normalizing the 3 components of the orientation by their average standard deviation σ_n. From the above normalized information vectors, each point is finally represented as:

$$\begin{aligned}\mathbf{p}_k^f = [&\bar{L}(p_k), \bar{a}(p_k), \bar{b}(p_k), \bar{x}(p_k), \bar{y}(p_k), \bar{z}(p_k), \\ &\bar{n}_x(p_k), \bar{n}_y(p_k), \bar{n}_z(p_k)]^T, \quad k = 1, \ldots, N\end{aligned} \tag{4}$$

Notice that in previous approaches [2, 4] there was a parameter balancing the contribution of color and geometry, but the advanced merging algorithm used in the last step of this work allows us to avoid the usage of such a parameter, whose proper setting was critical.

The computed 9D vectors are the input of the clustering algorithm that will be used in the splitting step. Among the various clustering techniques, normalized cuts spectral clustering [3] is an effective approach for this task. Since it is computationally very expensive, several methods have been proposed for its efficient approximation. For this paper we used the method based on the integral eigenvalue problem proposed in [38]. In this approach the set of points is first randomly subsampled, then the subset is partitioned and finally the solution is propagated to the whole set by a technique called Nyström method. After each clustering step, in order to avoid very small regions due to noise, a final refinement stage removing regions smaller than a pre-defined threshold T_p and assigning them to the closest segment is applied (in the experimental results we used $T_p = 800$). More details about the clustering procedure can be found in [4], however notice that in [4] the algorithm was used to directly segment the scene, while here it is recursively applied to divide each region in two parts as detailed in Section 6.

5. SURFACE FITTING ON THE SEGMENTED DATA

NURBS (Non-Uniform Rational B-Splines) are piecewise rational polynomial functions expressed in terms of proper bases, see [39] for a thorough introduction. They allow representation of freeform parametric curves and surfaces in a concise way, by means of control points. Notice that by including this model in the proposed method we are able to handle complex geometries, unlike many approaches in the literature, e.g., [24] and [19], that are limited to planar surfaces.

A parametric NURBS surface is defined as

$$\mathbf{S}(u,v) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \tag{5}$$

where $\mathbf{P}_{i,j}$ are the control points, $w_{i,j}$ are the corresponding weights, $N_{i,p}$ are the univariate B-spline basis functions, and p, q are the degrees in the u, v parametric directions respectively.

In our method, we initially set the degrees in the u and v directions equal to 3. We set the weights all equal to one, thus our fitted surfaces are non-rational (i.e., splines). Since the points to fit are a subset of the rectangular grid given by the sensor pixel arrangement, we set the corresponding (u_k, v_l) surface parameter values as lying on the image plane of the camera. The number of surface control points gives the degrees of freedom in our model. In order to set it adaptively depending on the number of input samples, we consider the horizontal and vertical extents of the segment to fit. Let us denote with W the horizontal image size and with H the vertical one (e.g., $W = 640$ and $H = 480$ for the Kinect data used in the results). We set $N_u = 15$ and $N_v = \frac{H}{W}N_u$ as the maximum number of control points in the u and v parametric directions, to be used in case of a segment covering the whole image, while for smaller ones we determine the number proportionally to the segment extents as follows. Let u_0, u_1 and v_0, v_1 be the minimum and maximum pixel values on the sensor grid corresponding to the segment in the horizontal and vertical directions. We set the number of control points in the u, v parametric directions respectively as

$$n = \max\left\{3, \left[N_u \frac{u_1 - u_0}{W}\right]\right\}, \quad m = \max\left\{3, \left[N_v \frac{v_1 - v_0}{H}\right]\right\} \tag{6}$$

Notice that since the minimum number of control points for a cubic spline is 4, for smaller segments we lower the surface degree to quadratic in order to allow 3 control points as actual minimum. These parameters turn out to be a reasonable choice, since they provide enough degrees of freedom to represent the shape of any common object, while the adaptive scheme at the same time prevents the fitting to always be more accurate for smaller segments, independently on how the segmentation algorithm was successful in detecting the objects in the scene.

Once determined the (u_k, v_l) parameter values corresponding to the points to fit, the surface degrees and the number of control points in the u, v parametric directions, we consequently obtain the NURBS knots (needed for the definition of the $N_{i,p}$ basis functions) as in [39]. Finally, by considering Eq. (5) evaluated at (u_k, v_l) and equated to the points to fit, we obtain an over-determined system of linear equations. We solve it in the least-squares sense by means of robust Cholesky decomposition of the normal equations matrix thus obtaining the surface control points. This approach is much faster than the SVD decomposition used in [2]. Notice also that a solution of the least-squares problem always exists. An example of fitted surface with the corresponding fitting error is shown in Fig. 2.

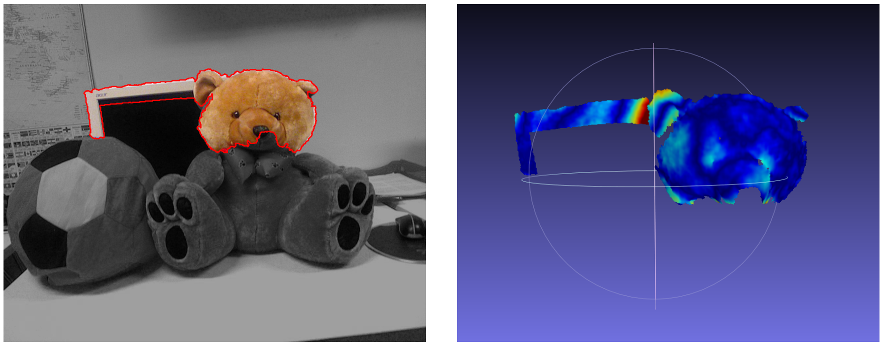


Figure 2: A 3D NURBS surface fitted over two clusters originated by segmentation of the scene in Fig. 7, third row. The red areas correspond to larger fit error. Notice how the large fit error between the teddy head and the monitor portion reveals that the two segments do not actually belong to the same object. (*Best viewed in color)*

6. ITERATIVE REGION SPLITTING PROCEDURE

After presenting the two main building blocks, we can now introduce the recursive region splitting procedure. A 9D point cloud corresponding to the scene is built according to Eq. (4) and used as initial input. Then, a binary segmentation using the method of Section 4 is applied, and a NURBS surface is fitted on each of the two resulting segments, from which the corresponding fitting error values are calculated. We use the Mean Squared Error (MSE) between the depth samples in the segment and the corresponding points on the approximating NURBS surface as the error metric. Notice that other fitting accuracy measures besides MSE can be considered. In [40] a review of different metrics either based on the evaluation of the fitting error or on geometric properties (curvature descriptors in particular) of the fitted surfaces is presented. According to the evaluation performed there, the MSE is the one providing the best performances for the proposed application.

At each next step of the procedure, the segment with the greatest fitting error is examined, and it is further split provided it is not too small and the recursion has not yet produced enough segments. More in detail, consider an intermediate step during which segment S_i is the one being processed since its fitting error e_i is the largest one. The following conditions are checked in order to consider S_i for a further split operation:

1. The size of S_i must be greater than $2T_p$, otherwise the split would produce at least one segment smaller than T_p. This is consistent with the choice of not allowing segments smaller than T_p made in Section 4.
2. The fitting error e_i must be greater than a threshold $T_{mse} = 0.0005$. The idea is that if this is not the case, the segment is already representing very accurately an object in the scene and there is no point in further dividing it.
3. The number of recursive splits starting from the initial scene and leading to S_i must be smaller than T_d (we used $T_d = 10$ for the results). In other words the depth of the recursion tree must be smaller than T_d on the branch containing S_i (an example of the tree structure is shown in Fig. 3).
4. A maximum number of total splits (i.e., segments) T_s must not have been reached yet. This corresponds to setting a maximum number of possi-

ble segments (e.g., $T_s = 50$ segments). Notice that this is only an upper bound, differently from many segmentation algorithms on which the actual number of segments is pre-determined. Notice also that the stopping conditions can be modified in order to obtain exactly a pre-determined number of segments if required for some applications.

If all the above conditions hold, S_i is split into sub-segments S_{i0} and S_{i1}, the corresponding fitting errors e_{i0} and e_{i1} are computed and the procedure is iterated by processing the next segment with the greatest fitting error (the list includes also S_{i0} and S_{i1} at this point). If condition 4 is violated the procedure is stopped, while if any of conditions 1, 2, 3 does not hold, S_i is kept as part of the final segmentation and the algorithm continues on another branch of the tree. This happens also if the subdivision of S_i fails to actually create two sub-segments, since one between S_{i0} and S_{i1} is smaller than T_p and the post-processing step described on Section 4 merges it again with the other one (this is a rare case actually).

Notice that this splitting procedure could be used directly for the final segmentation, without the merging step, as we proposed in [2] and [40]. Following the approach used in these works, the fact that the error is improved or not with respect of the original segment can be used as criterion to accept or reject the split operation by checking an additional condition. The weighted average of the fitting errors on the sub-segments is compared to the one of the original segment, that is

$$\frac{e_{i0}|S_{i0}| + e_{i1}|S_{i1}|}{e_i|S_i|} < 1 \tag{7}$$

where the weights are the number of points belonging to the segments. If the condition is not satisfied, the split is discarded and the algorithm moves to the next segment to process. This check is used to avoid an over-segmentation of the scene during the split stage. However in the method proposed in this work this is not critical, since the segments will be recursively recombined based on the improving fitting accuracy in the merging phase later. For this reason the check of Eq. (7) can be dropped.

In summary, at each step of the splitting phase the previously introduced conditions are checked for the segment with the greatest MSE, i.e.:

$$(|S_i| \geq 2T_p) \wedge (|e_i| \geq T_{mse}) \wedge (depth(S_i) < T_d) \wedge (count(i) < T_s) \qquad (8)$$

where $depth(S_i)$ is the depth of S_i in the recursive tree structure, and $count(i)$ is the number of splits made until the current iteration. If the conditions are satisfied the segment is split, otherwise the next available one with the greatest MSE is processed. The procedure is applied recursively to all the segments and the generated sub-segments, until either the maximum number of splits is reached, or there are no more available segments satisfying all the conditions. This is summarized in Algorithm 1 and a visual scheme is shown in the central block of Fig. 1. The result is a tree structure like the one of Fig. 3, where the leaves are the elements of the final over-segmentation that will be used as input for the following merging procedure. An example of the progressive subdivision in smaller and smaller segments produced by this approach is shown in Fig. 4.

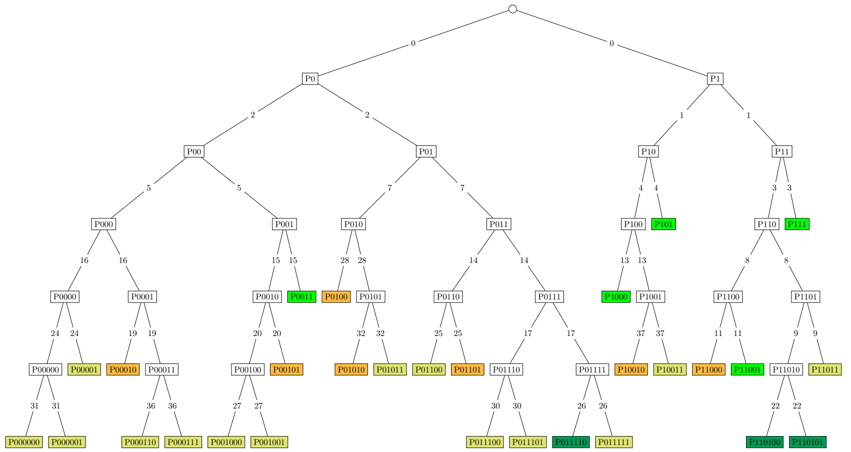
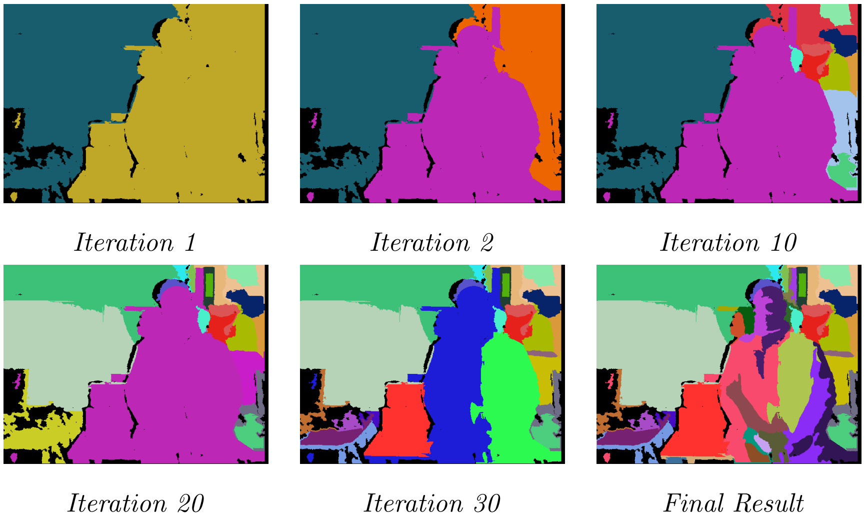

0 0
P0 P1
2 2 1 1
P00 P01 P10 P11
5 5 7 7 4 4 3 3
P000 P001 P010 P011 P100 P101 P110 P111
16 16 15 15 28 28 14 14 13 13 8 8
P0000 P0001 P0010 P0011 P0100 P0101 P0110 P0111 P1000 P1001 P1100 P1101
24 24 19 19 20 20 32 32 25 25 17 17 37 37 11 11 9 9
P00000 P00001 P00010 P00011 P00100 P00101 P01010 P01011 P01100 P01101 P01110 P01111 P10010 P10011 P11000 P11001 P11010 P11011
31 31 36 36 27 27 30 30 26 26 22 22
P000000 P000001 P000110 P000111 P001000 P001001 P011100 P011101 P011110 P011111 P110100 P110101

Figure 3: Tree structure originated by the splitting procedure on a sample scene. The colored nodes correspond to the final segments. *Orange segments*: the segmentation did not create two sub-segments, since one of them would be smaller than T_p. *Light green*: not split since smaller than $2T_p$. *Bright green*: not split since fitting error smaller than T_{mse}. *Green*: stopped since the maximum tree depth T_d was reached. (*Best viewed in color)*

```
while there are still segments available to split do
  Select segment $S_i$ with the largest MSE
  if $S_i$ satisfies all conditions of Eq. (8) then
    Split $S_i$ in two parts $S_{i0}$ and $S_{i1}$ (Sec. 4)
    if $(|S_{i0}| < T_p) \vee (|S_{i1}| < T_p)$ then
      Remove $S_i$ from list of segments available to split
      continue
    end if
    Fit a NURBS surface on $S_{i0}$ and $S_{i1}$ (Sec. 5)
    Compute fitting errors $e_{i0}$ and $e_{i1}$
    (optional) Check if fitting errors satisfy Eq. (7)
    Add $S_{i0}$ and $S_{i1}$ to list of segments available to split
  else
    Remove $S_i$ from list of segments available to split
    continue
  end if
  if maximum number of splits reached then
    return
  end if
end while
```

Algorithm 1: Split algorithm



Iteration 1 *Iteration 2* *Iteration 10*

Iteration 20 *Iteration 30* *Final Result*

Figure 4: Execution of the splitting procedure on the scene of Fig. 7, sixth row. The images show the segmentation after 1, 2, 10, 20 and 30 iterations and the final resulting over-segmentation (after 50 iterations).

7. ITERATIVE REGION MERGING PROCEDURE

With the recursive splitting algorithm alone it is difficult to control the trade-off between an over-segmentation of the scene and the recognition of all the objects and structures in it. In particular in some splitting steps the binary segmentation can mistakenly divide a single object due to misleading clues. For these reasons a final merging stage is used to recombine multiple segments corresponding to the same scene object. The workflow of this merging procedure is depicted in the third part of Fig. 1 and summarized in Algorithm 2. The goal is to analyze the neighboring segments in order to join the adjacent ones belonging to the same region.

Notice that during the splitting phase a NURBS surface has been fitted on each final segment S_i and the corresponding fitting error e_i has been computed. Then, the algorithm starts by sorting all the segments in decreasing order based on the fitting error, thus producing an ordered list L_S where the segments with larger fitting errors come first. It also analyzes all the segments and builds an adjacency matrix, storing for each couple of segments whether they are *adjacent* or not. Two segments are considered as *adjacent* if they satisfy the following conditions:

1. They must be connected on the depth map lattice (using 4-connectivity) and the length l_{cc} of the shared boundary C_C (depicted in red in the example of Fig. 5) must be bigger than a threshold T_l. For the 640 by 480 pixels images used in the results a reasonable value is $T_l = 15$.
2. The depth values on the shared boundary must be similar. In order to check this we compute the difference ΔZ_k between the depth values on the two sides of the edge at each contour location C_k (see Fig. 5). This step is exemplified in Fig. 5, where the orange arrows show which differences are considered. The number of points l_{cc}^d of the shared boundary with a depth difference smaller than a threshold $T_z = 0.15$ is computed, and this condition must be satisfied by the majority of the boundary pixels, i.e., length l_{cc}^d must be more than half of the total length. This can be expressed as:
$$\frac{|p_k : (p_k \in C_C) \wedge (\Delta Z_k \leq T_z)|}{|p_k : p_k \in C_C|} = \frac{l_{cc}^d}{l_{cc}} > 0.5 \tag{9}$$
3. The color values must also be similar along the shared contour. The approach is the same as before except that the color difference in the CIELab is used in place of the depth value. More in detail, the color difference ΔC_k between samples on the two sides of the shared boundary is calculated and the number of points l_{cc}^c with a color difference smaller than a threshold $T_c = 5$ is computed. Similarly to the previous case, l_{cc}^c must account for the majority of the boundary pixels , i.e.,
$$\frac{|p_k : (p_k \in C_C) \wedge (\Delta C_i \leq T_c)|}{|p_k : p_k \in C_C|} = \frac{l_{cc}^c}{l_{cc}} > 0.5 \tag{10}$$
4. Finally the same approach is used also for orientation information. In this case the angle between the two normal vectors $\Delta\theta_k$ is calculated for each couple of samples on the two sides of the shared boundary and the number of points l_{cc}^n for which $\Delta\theta_k \leq T_\theta = 2°$ is computed. As before, l_{cc}^n must

be more than half of the total length , i.e.,

$$\frac{|p_K : (p_k \in C_C) \wedge (\Delta\theta_i \leq T_\theta)|}{|p_k : p_k \in C_C|} = \frac{l^n_{cc}}{l_{cc}} > 0.5 \tag{11}$$

If all the above conditions hold the two segments are marked as adjacent. Notice that the checks are performed in the presented order, and in case any of them is not satisfied the following ones are skipped. This allows us to avoid unnecessary computations, since we exclude most couple of segments before computing all the depth, color and normal differences along the contour.

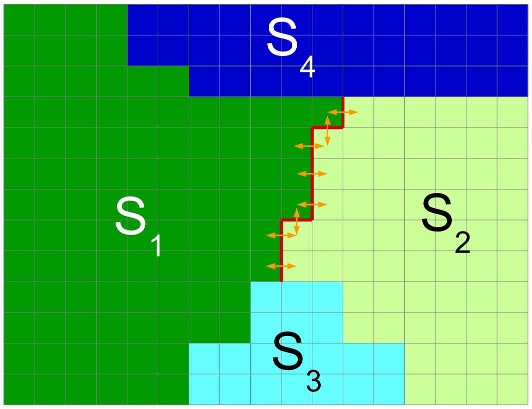


Figure 5: Example of boundary region with the common contour between two sample segments S_1 and S_2 and the differences used for Equations (9), (10) and (11).

The procedure then selects the segment with the largest fitting error and tries to join it with all the ones that are adjacent (according to the previously introduced criteria). In detail, let S_i be the segment with the greatest fitting error e_i. The algorithm considers each adjacent segment S_j (with fitting error e_j), and tries to join S_i and S_j obtaining the segment $S_{i\cup j}$. Then, a NURBS surface is fitted the merged segment $S_{i\cup j}$, the corresponding fitting error $e_{i\cup j}$ is computed and it is compared against the weighted average of the fitting errors on S_i and S_j:

$$\frac{e_i|S_i| + e_j|S_j|}{e_{i\cup j}(|S_i| + |S_j|)} > 1 \tag{12}$$

```
Compute $L_S$ (list of segments) and sort it according to $e_i$
For each segment $S_i$ compute set $\mathcal{A}_i$ of adjacent segments
$i = 1$ (select as $S_i$ the first segment in $L_S$)
**while** $i < length(L_S)$ **do**
  **for** all segments $S_j$ adjacent to $S_i$ **do**
    compute fitting error on merged segment $S_{i \cup j}$
    check if condition of Eq. (12) is satisfied
  **end for**
  **if** at least one merge operation satisfies Eq. (12) **then**
    Select merge operation leading to best fitting accuracy improvement
    (the corresponding segment is $S_{j^*}$)
    Remove $S_i$ and $S_{j^*}$ from $L_S$
    Add $S_{i \cup j^*}$ to $L_S$
    Compute $\mathcal{A}_{i \cup j^*}$
    $i = 1$ ( $S_i$ is the first segment in $L_S$)
  **else**
    $i = i + 1$ ($S_i$ is the next segment in $L_S$)
  **end if**
**end while**
```

Algorithm 2: Merge algorithm

If the fitting accuracy improves, i.e., the condition of Eq. (12) is satisfied, the merging operation between S_i and S_j is candidate to be accepted, otherwise it is discarded. The procedure is repeated for all the segments adjacent to S_i and, among the ones for which the merging operation would improve the fitting error (if any), the segment S_{j^*} that provides the maximum improvement according to Eq. (12) is selected. The two segments S_i and S_{j^*} are then merged, and the list L_S is updated by removing them and inserting their union $S_{i \cup j^*}$ in the position corresponding to its fitting error $e_{i \cup j^*}$. The adjacency information is also updated by considering $S_{i \cup j^*}$ adjacent to all the segments that were adjacent to either S_i or S_{j^*}. In case instead there are no merging operations with S_i that would improve the fitting error, the algorithm selects the next segment in L_S as new segment S_i and the procedure is iterated.

The algorithm then continues by processing the next segment with the greatest fitting error and iterates until no more segments can be considered for a merge operation. The procedure is summarized in Algorithm 2 and its progress on a sample scene is visualized in Fig. 6, where a graph representing the various merge operations and the resulting segmentations at several iterations are shown. Some videos showing the sequence of merging steps on a few sample scenes are available in the additional material.

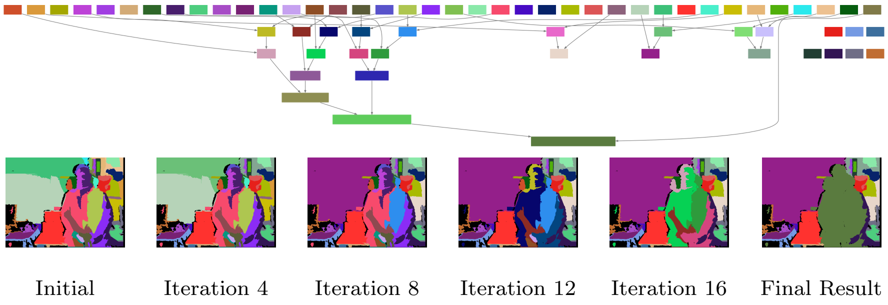


Figure 6: Example of the merging procedure on the scene of Fig. 7, sixth row. The images show the initial over-segmentation, the merging output after 4,8,12 and 16 iterations and the final result (iteration 21). The graph shows the merge operations between the various segments. The colors in the images correspond to those of the graph nodes. (*Best viewed in color)*

8. EXPERIMENTAL RESULTS

The proposed method has been experimentally evaluated by firstly analyzing the performances of the approach and of the various sub-components (roughly corresponding to our previous works) on a small sample dataset and then by comparing it with state-of-the-art approaches on the large and challenging NYU Depth Dataset V2 (NYUv2).

The first and simpler dataset has been acquired at the LTTM laboratory [2] and is available at `http://lttm.dei.unipd.it/downloads/segmentation`. It contains 6 different images with the corresponding depth maps acquired with PrimeSense devices (i.e., Kinect v1 and Xtion sensors). Ground truth information obtained by manual segmentation is also available. Even though it is a small dataset, it contains a large variety of different structures and objects with different shapes, colors and properties thus representing a good starting point to evaluate the various components of the proposed approach. It also represents a completely different environment from the NYUv2 dataset, thus ensuring that the approach has not been over-fitted on that dataset. The scenes have been segmented with the approaches of [4], [40], [5] and with the proposed method. This is an interesting test to understand the relevance of the various components since the first method directly segments the scene into the desired number of regions with an approach based on spectral clustering and can be considered a simplified version of the initial segmentation scheme of Sec. 4. The second one exploits a region splitting scheme similar to the split phase of the proposed method, while the third one corresponds to the merging phase.

The visual results are shown in Fig. 7, and Table 1 lists the numerical values from the comparison between the results and the ground truth. Starting from visual results, from Fig. 7 it is clear how the proposed method obtains better performances than the compared approaches. The approach of [4] tends to over-segment the considered scenes and has some difficulties in properly capturing all the objects. In particular the background is completely wrong in some scenes (e.g., 5 and 6), since the geometry dependent term combined with the bias towards segments of similar size of the normalized cuts algorithm [3] forces the large regions to be divided into multiple pieces. Notice how in the proposed approach the merging scheme solves this problem by recombining together segments belonging to the same surface. The approach of [40] produces less segments but is not able to recognize all the objects (e.g., in scenes 4 and 5). Notice in particular that the splitting approach alone can hardly recognize all the scene objects and at the same time avoid over-segmentation, while the proposed method can perform a larger number of splitting steps without affecting the final result since over-segmentation issues will be fixed in the merging phase. Since the approach in [40] does not exploit orientation information, to fully evaluate the performances of the splitting algorithm we implemented also a modified version that takes normals into account. This better corresponds to the splitting stage in our method. It is possible to notice that orientation information allows to better capture some surfaces, e.g., the background in scene 2 and the table in scene 5. The region merging scheme [5] is the one that obtains results closer to the proposed approach. Both [5] and the proposed method avoid the creation of small segments caused by noise or by complex surfaces, and at the same time they properly extract most of the structures in the scene. Moreover, the use of orientation information makes the various walls and the surfaces with different orientation properly recognized (e.g., the table in row 3). However, notice how the background (especially in scenes 1 and 2) is properly captured only by the proposed approach, while [5] fails to correctly recognize the various background surfaces. In general in the proposed approach the objects are well recognized, and there are a very few segments extending over separate objects at different depths.

The visual evaluation is confirmed by the numerical results, as shown by Table 1. In order to compare the results with ground truth data we used two different metrics, the Variation of Information (VoI) and the Rand Index (RI). A description of these error metrics can be found in [41], in particular notice that lower values correspond to better results for the VoI metric while higher values are better for the RI metric. The table lists the average values of the two metrics on the six considered scenes. It shows that according to both metrics the proposed approach outperforms all the compared ones. The VoI metric value is better by a large gap with respect to [4] and [40]. The merging approach has closer performances, as can be noticed from the visual results, but the proposed scheme is able to outperform it with an average VoI score of 1.66 against 1.74. The behavior is similar for the RI metric, where the proposed approach is the best one with a score of 0.91. Notice that approaches exploiting orientation information give in general better performances.

Approach	*VoI*	*RI*
(Clustering) [4]	2.71	0.81
(Split) [40]	2.01	0.83
(Split+normals) [40]	1.92	0.88
(Merge) [5]	1.74	0.88
Proposed Method	**1.66**	**0.91**

Table 1: Comparison of the performances of the proposed method with [4, 40, 5]. The table shows the average value of the VoI and RI metrics on the six scenes of the LTTM dataset.

The main evaluation has been carried on the NYU Depth Dataset V2 [22] dataset that is much larger and thus more interesting. It has been widely used for the evaluation of joint color and depth segmentation algorithms and allows to compare the proposed method with the state-of-the-art methods for this task. The dataset has been acquired with a Kinect and contains 1449 depth and color frames from a variety of indoor scenes. Ground truth data is also available but, since on the original ground truth data has sometimes missing values in proximity of edges, for the numerical evaluation we used the updated versions of the ground truth labels provided by the authors of [42]. Table 2 shows the comparison between our method, our previous approaches roughly corresponding to the splitting and merging sub-components, and some state-of-the-art schemes from the literature (for some competing approaches we collected the results from [11] and [13]). The compared state-of-the art

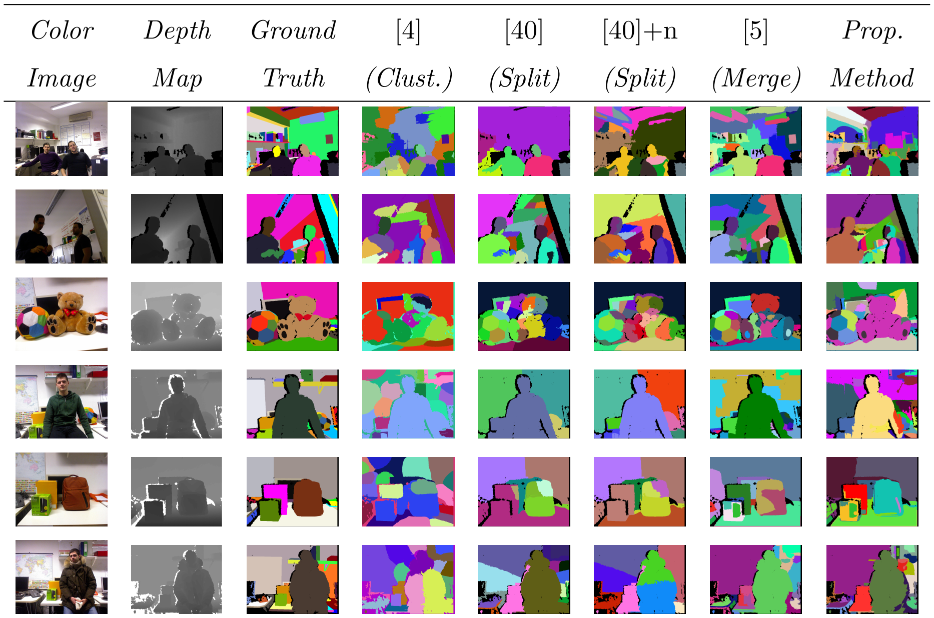


Figure 7: Segmentation of some sample scenes with the proposed method and with the approaches of [4], [40] (with and without normals) and [5]. The black regions for the proposed approach correspond to samples without a valid depth value from the Kinect that have not been considered for the segmentation.

approaches are the clustering and region merging method of [11], including the recent improved version [12], the MRF scene labeling scheme of [23], a modified version of [7] that accounts also for geometry information, the dynamic programming scheme of [24], the bottom-up approach of [20] (for this approach we used the implementation in the PCL open source library) and the multi-layer clustering strategy of [13]. Notice that we used the same parameter values for all the scenes.

The average values obtained by our method are 2.17 as to the VoI metric and 0.89 as to RI. The VoI metric results show that our approach outperforms all the compared ones and the improvement over most of them is relevant. The two approaches that get performances closer to the proposed one are [12] and our previous work based on the merging scheme [5], with VoI values of 2.20 and 2.23 respectively compared to 2.17 (recall that for VoI smaller is better). If the RI metric is considered, the proposed method outperforms most compared approaches and obtains results very similar to those of the state-of-the-art methods of [23] and [12], with a small difference of just 0.01 and 0.02 respectively. Notice also that our approach does not make any assumption about the presence of planar surfaces differently from [11], [12] and [24], so it better generalizes to scenes with non-planar surfaces (in the NYUv2 dataset all the scenes are indoor settings with many planar surfaces like walls and furniture, while outdoor settings have a larger variability). In addition the method of [23] exploits a learning stage, while our approach does not assume any previous knowledge on the data.

A visual comparison on 7 different scenes from this dataset is shown in Fig. 8. Notice that the scenes have been selected by the authors of [11]. Even if this dataset is more challenging, the proposed approach provides a reliable segmentation on all the considered scenes as shown in the last column of Fig. 8. The obtained segmentations are clearly better than the approaches of [7], [24], [4], [13] and [20] (columns 6-7-8-9-10). The comparison with the two best performing approaches, i.e., [11] and [23] (columns 4 and 5), is more challenging and there is not a clear winner. The various objects are properly extracted by our

Approach	*VoI*	*RI*
Felzenszwalb et al. [7]	2.32	0.81
Ren et al. [23]	2.35	0.90
Taylor et al. [24]	3.15	0.85
Stein et al. [20]	2.68	0.87
Khan et al. [13]	2.42	0.87
Hasnat et al. (JCSA) [11]	2.72	0.87
Hasnat et al. (JCSA-RM) [11]	2.29	0.90
Hasnat et al. (JCSD-RM) [12]	2.20	**0.91**
(Clustering) [4]	3.09	0.84
(Split) [40]	2.62	0.75
(Split+normals) [40]	2.52	0.76
(Merge) [5]	2.23	0.88
Proposed Method	**2.17**	0.89

Table 2: Performances of the proposed method, of some state-of-the-art approaches (first block) and of approaches corresponding to the sub-components of the method (second block). The table shows the average values of the VoI and RI metrics on the NYUv2 dataset.

approach and the background region is correctly handled on most scenes. It also does not produce noisy small segments in proximity of edges, an issue happening with other approaches on some scenes. Some small errors are present, e.g., in the corridor and bed scenes (rows 2 and 3). In particular the blanket of the bed scene (row 3) is quite critical since color data is very noisy and the normals on the rough surface are very unstable.

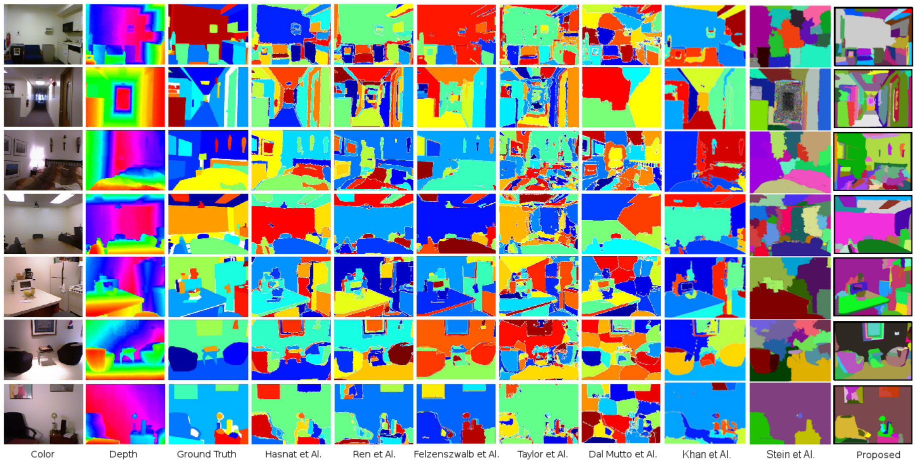


Figure 8: Segmentation of some sample scenes from the NYUv2 dataset: (column 1) color data; (column 2) depth data; (column 3) ground truth; (column 4) [11]; (column 5) [23]; (column 6) [7]; (column 7) [24]; (column 8) [4]; (column 9) [13]; (column 10) [20]; (column 11) proposed method. The results for some of the competing methods have been collected from [11] and [13].

Some critical cases for our approach are shown in Fig. 9. Notice how very complex geometries (columns 1 and 2) are challenging to segment. Furthermore thin structures, far objects or transparent surfaces are not properly captured by the Kinect and this affects the final segmentation (columns 1, 2 and 5). Finally the color can saturate on very bright (column 4) or very dark (column 3) scenes, thus making the separation of some objects difficult (notice that even geometry does not help in these cases).

The total computation time is comparable to the one of a direct segmentation with normalized cuts since in most iterations both the spectral clustering and

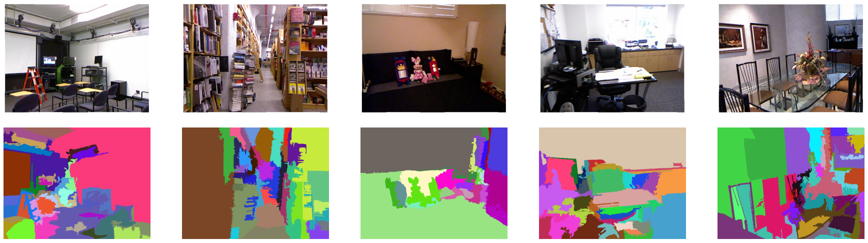


Figure 9: Examples of critical scenes. The figure shows the color data and the segmentation computed with the proposed approach for scenes 7, 86, 283, 617 and 1439 of the NYUv2 dataset.

the surface fitting algorithms are applied to small subsets of the scene (the surface fitting actually adds just a small overhead to the segmentation time).

9. CONCLUSIONS

In this paper we proposed a region splitting and merging scheme for the joint segmentation of color and depth information. Spectral clustering is used inside a tree-structured algorithm to recursively divide the scene into a set of segments, that are then recombined by a recursive merging procedure into larger regions corresponding to the actual scene objects. The proposed approach exploits a NURBS surface fitting scheme to determine if each splitting or merging operation has led to a more accurate representation of the 3D surfaces, and consequently whether it should be accepted or discarded. Experimental results demonstrate its ability to avoid over-segmentation and at the same time properly segment the various structures in the scene, outperforming several state-of-the-art approaches. Future research will be devoted to exploit the proposed approach for semantic segmentation schemes using also the surface fitting information among the classification features. Finally GPU and parallel implementations will be considered to reduce the computation time.

References

[1] R. Szeliski, Computer vision: algorithms and applications, Springer Science & Business Media, 2010.

[2] G. Pagnutti, P. Zanuttigh, Scene segmentation from depth and color data driven by surface fitting, in: Proceedings of IEEE International Conference on Image Processing (ICIP), 2014, pp. 4407–4411.

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

[4] C. Dal Mutto, P. Zanuttigh, G. Cortelazzo, Fusion of geometry and color information for scene segmentation, IEEE Journal of Selected Topics in Signal Processing 6 (5) (2012) 505–521.

[5] G. Pagnutti, P. Zanuttigh, Joint color and depth segmentation based on region merging and surface fitting, in: Proceedings of International Joint Conference on Computer Vision Theory and Applications (VISAPP), 2016.

[6] D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (5) (2002) 603–619.

[7] P. Felzenszwalb, D. Huttenlocher, Efficient graph-based image segmentation, International Journal of Computer Vision 59 (2) (2004) 167–181.

[8] F. Calderero, F. Marques, Hierarchical fusion of color and depth information at partition level by cooperative region merging, in: Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2009, pp. 973–976.

[9] A. Bleiweiss, M. Werman, Fusing time-of-flight depth and color for real-time segmentation and tracking, in: Proceedings of German Association for Pattern Recognition (DAGM) Conference, 2009.

[10] M. Wallenberg, M. Felsberg, P.-E. Forssén, B. Dellen, Channel coding for joint colour and depth segmentation, in: Proceedings of German Association for Pattern Recognition (DAGM) Conference, 2011, pp. 306–315.

[11] M. A. Hasnat, O. Alata, A. Trmeau, Unsupervised RGB-D image segmentation using joint clustering and region merging, in: Proceedings of British Machine Vision Conference (BMVC), 2014.

[12] M. Hasnat, O. Alata, A. Tremeau, Joint color-spatial-directional clustering and region merging (jcsd-rm) for unsupervised RGB-D image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence.

[13] M. R. Khan, A. B. M. M. Rahman, G. M. A. Rahaman, M. A. Hasnat, Unsupervised RGB-D image segmentation by multi-layer clustering, in: International Conference on Informatics, Electronics and Vision (ICIEV), 2016, pp. 719–724.

[14] M. Dahan, N. Chen, A. Shamir, D. Cohen-Or, Combining color and depth for enhanced image segmentation and retargeting, The Visual Computer 28 (12) (2012) 1181–1193.

[15] J.-E. Lee, R.-H. Park, Segmentation with saliency map using colour and depth images, IET Image Processing 9 (1) (2015) 62–70.

[16] Z. Deng, L. J. Latecki, Unsupervised segmentation of RGB-D images, in: Proceedings of Asian Conference on Computer Vision (ACCV), Springer, 2014, pp. 423–435.

[17] J. Yang, Z. Gan, K. Li, C. Hou, Graph-based segmentation for rgb-d data using 3-d geometry enhanced superpixels, IEEE Transactions on Cybernetics 45 (5) (2015) 927–940.

[18] C. Erdogan, M. Paluri, F. Dellaert, Planar segmentation of RGBD images using fast linear fitting and markov chain monte carlo, in: Proceedings of Conference on Computer and Robot Vision (CRV), 2012.

[19] N. Srinivasan, F. Dellaert, A rao-blackwellized mcmc algorithm for recovering piecewise planar 3d model from multiple view RGBD images, in: Proceedings of IEEE International Conference on Image Processing (ICIP), 2014.

[20] S. Christoph Stein, M. Schoeler, J. Papon, F. Worgotter, Object partitioning using local convexity, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 304–311.

[21] S. Gupta, P. Arbeláez, R. Girshick, J. Malik, Indoor scene understanding with RGB-D images: Bottom-up segmentation, object detection and semantic segmentation, International Journal of Computer Vision (2014) 1–17.

[22] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from RGBD images, in: Proceedings of European Conference on Computer Vision (ECCV), 2012.

[23] X. Ren, L. Bo, D. Fox, RGB-D scene labeling: Features and algorithms, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[24] C. J. Taylor, A. Cowley, Parsing indoor scenes using RGB-D imagery, in: Robotics: Science and Systems, Vol. 8, 2013, pp. 401–408.

[25] J. Gallego, M. Pardàs, Region based foreground segmentation combining color and depth sensors via logarithmic opinion pool decision, Journal of Visual Communication and Image Representation 25 (1) (2014) 184–194.

[26] M. Harville, G. Gordon, J. Woodfill, Foreground segmentation using adaptive mixture models in color and depth, in: Proceedings of IEEE Workshop on Detection and Recognition of Events in Video, 2001.

[27] J. Leens, S. Pirard, O. Barnich, M. Van Droogenbroeck, J. Wagner, Combining color, depth, and motion for video segmentation, in: Computer Vision Systems, 2009.

[28] L.-H. Juang, M.-N. Wu, F.-M. Tsou, A dynamic portrait segmentation by merging colors and depth information, International Journal of Control, Automation and Systems 13 (5) (2015) 1286–1293.

[29] L. Wang, C. Zhang, R. Yang, C. Zhang, TofCut: Towards robust real-time foreground extraction using a time-of-flight camera, in: International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT), 2010.

[30] L. Ladicky, P. Sturgess, C. Russell, S. Sengupta, Y. Bastanlar, W. Clocksin, P. Torr, Joint optimisation for object class segmentation and dense stereo reconstruction, in: Proceedings of British Machine Vision Conference (BMVC), 2010.

[31] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, C. Rother, Bi-layer segmentation of binocular stereo video, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2, 2005, p. 1186 vol. 2.

[32] M. Bleyer, C. Rother, P. Kohli, D. Scharstein, S. Sinha, Object stereo-joint stereo matching and object segmentation, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.

[33] C. Dal Mutto, P. Zanuttigh, G. Cortelazzo, Scene segmentation assisted by stereo vision, in: Proceedings of Joint 3DIM/3DPVT Conference (3DIMPVT), Hangzhou, China, 2011.

[34] C. Dal Mutto, P. Zanuttigh, G. M. Cortelazzo, Time-of-Flight Cameras and Microsoft Kinect, SpringerBriefs in Electrical and Computer Engineering, Springer, 2012.

[35] D. Herrera C, J. Kannala, J. Heikkila, Joint depth and color camera calibration with distortion correction, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (10) (2012) 2058–2064.

[36] C. Dal Mutto, P. Zanuttigh, G. Cortelazzo, A probabilistic approach to tof and stereo data fusion, in: International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT), Paris, France, 2010.

[37] S. Holzer, R. Rusu, M. Dixon, S. Gedikli, N. Navab, Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images, in: Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012, pp. 2684–2689.

[38] C. Fowlkes, S. Belongie, F. Chung, J. Malik, Spectral grouping using the nyström method, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2) (2004) 214–225.

[39] L. Piegl, W. Tiller, The NURBS Book (2Nd Ed.), Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[40] G. Pagnutti, P. Zanuttigh, Scene segmentation based on nurbs surface fitting metrics, in: Proceedings of Smart Tools and Apps in computer Graphics (STAG), 2015.

[41] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (5) (2011) 898–916.

[42] S. Gupta, P. Arbelaez, J. Malik, Perceptual organization and recognition of indoor scenes from RGB-D images, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.