# Scene Segmentation Driven by Deep Learning and Surface Fitting

Ludovico Minto, Giampaolo Pagnutti, and Pietro Zanuttigh(✉)

Department of Information Engineering, University of Padova, Padova, Italy
{mintolud,pagnutti,zanuttigh}@dei.unipd.it

**Abstract.** This paper proposes a joint color and depth segmentation scheme exploiting together geometrical clues and a learning stage. The approach starts from an initial over-segmentation based on spectral clustering. The input data is also fed to a Convolutional Neural Network (CNN) thus producing a per-pixel descriptor vector for each scene sample. An iterative merging procedure is then used to recombine the segments into the regions corresponding to the various objects and surfaces. The proposed algorithm starts by considering all the adjacent segments and computing a similarity metric according to the CNN features. The couples of segments with higher similarity are considered for merging. Finally the algorithm uses a NURBS surface fitting scheme on the segments in order to understand if the selected couples correspond to a single surface. The comparison with state-of-the-art methods shows how the proposed method provides an accurate and reliable scene segmentation.

**Keywords:** Segmentation · Depth · Color · Kinect · NURBS · Deep Learning · CNN

## 1 Introduction

The introduction of depth cameras in the consumer market has opened the way to novel algorithms able to exploit depth in order to tackle classical challenging computer vision problems. Among them segmentation has always been a critical issue despite a huge amount of research devoted to this problem since it is an ill-posed problem and the information content in color images is often not sufficient to completely solve the task. The 3D representation of the acquired scene contained in depth data is very useful for this task and recently various approaches combining it with color information have been proposed.

Among the various segmentation techniques, one of the best performing solutions is normalised cuts spectral clustering [25]. This approach can be easily extended to the joint segmentation of image and depth data by feeding to the clustering scheme multi-dimensional vectors containing both kinds of information [6]. In this way, a relatively reliable segmentation can be obtained but, since the approach has a bias towards segments of similar sizes, it is often difficult to properly segment all the objects and at the same time avoid an over-segmentation of the scene.

© Springer International Publishing Switzerland 2016
G. Hua and H. Jégou (Eds.): ECCV 2016 Workshops, Part III, LNCS 9915, pp. 118–132, 2016.
DOI: 10.1007/978-3-319-49409-8_12

The idea exploited in this work is to start from an over-segmentation performed with spectral clustering and then exploit an iterative region merging scheme in order to obtain the final segmentation. The key problem of deciding which segments must be merged is solved with the combined use of two clues. The first is a similarity index between segments computed by comparing the descriptors produced by a Convolutional Neural Network. The second has been derived from [22] and is based on the accuracy obtained by fitting a Non-Uniform Rational B-Spline (NURBS) model on the union of the two segments and comparing the accuracy of the fitting with the one obtained on each of the two merged regions alone. If the accuracy remains similar the segments are probably part of the same surface and the merging is accepted, otherwise it is discarded.

The paper is organized in the following way: a review of related works is presented in Sect. 2. Section 3 introduces the general architecture of the proposed algorithm. The over-segmentation step is described in Sect. 4 while Sect. 5 presents the similarity analysis based on deep learning. Then the employed region merging algorithm is presented in Sect. 7. The last two sections contain the results (Sect. 8) and the conclusions (Sect. 9).

## 2 Related Works

The idea of using also the information from an associated depth representation to improve segmentation algorithm performances has been exploited in various recent scene segmentation schemes, a review of this family of approaches is contained in [32]. Clustering techniques can easily be extended to joint depth and color segmentation by modifying the feature vectors as in [2,5,30]. A segmentation scheme based on spectral clustering able to automatically balance the relevance of color and depth clues has been proposed in [6].

Region splitting and growing approaches have also been considered. In [8] superpixels produced by an over-segmentation of the scene are combined together in regions corresponding to the planar surfaces using an approach based on Rao-Blackwellized Monte Carlo Markov Chain. The approach has been extended to the segmentation of multiple depth maps in [27]. The top down approach (region splitting) has been used in [21] where the segmentation is progressively refined in an iterative scheme by recursively splitting the segments that do not represent a single surface in the 3D space. Hierarchical segmentation based on the output of contour extraction has been used in [13], that also deals with object detection from the segmented data. Another combined approach for segmentation and object recognition has been presented in [26], that exploits an initial over-segmentation exploiting the watershed algorithm followed by a hierarchical scheme. A joint clustering method on the color, 3D position and normal information followed by a statistical planar region merging scheme has been presented in [17] and in the refined version of the approach of [16]. Finally dynamic programming has been used in [28] to extract the planar surfaces in indoor scenes.

Machine learning techniques have been used specially for the task of semantic segmentation where a label is also associated to each segment. In [24] a Markov

Random Fields superpixel segmentation is combined with a tree-structured approach for scene labeling. Conditional Random Fields have been employed in [7] together with mutex constraints based on the geometric structure of the scene. Deep learning techniques and in particular Convolutional Neural Networks (CNN) have also been used for this task [4,19,20]. For example a multiscale CNN has been used in [4] while the method of [20] is based on Fully Convolutional Networks. Another approach based on deep learning is [14], that exploits a CNN applied on features extracted from the geometry description.

## 3 General Overview

The proposed algorithm can be divided into three main steps as depicted in Fig. 1. The color image and the depth map are firstly converted into a set of 9D vectors containing the 3D position, the orientation information and the color coordinates in the CIELab color space of each sample. Then we perform an oversegmentation of the scene based on the joint usage of the three sources of information inside a spectral clustering framework derived from [6] (see Sect. 4). In parallel, the color and orientation data are also fed to a CNN classifier that computes a vector of descriptors for each pixel. The descriptors are then aggregated inside each segment in order to produce a single descriptor for each segmented region. The final step is an iterative region merging procedure. This stage starts by analyzing the segmentation and computing an adjacency map for the segments, where two segments are considered adjacent if they touch each other and their properties are similar on the common contour. The couples of adjacent

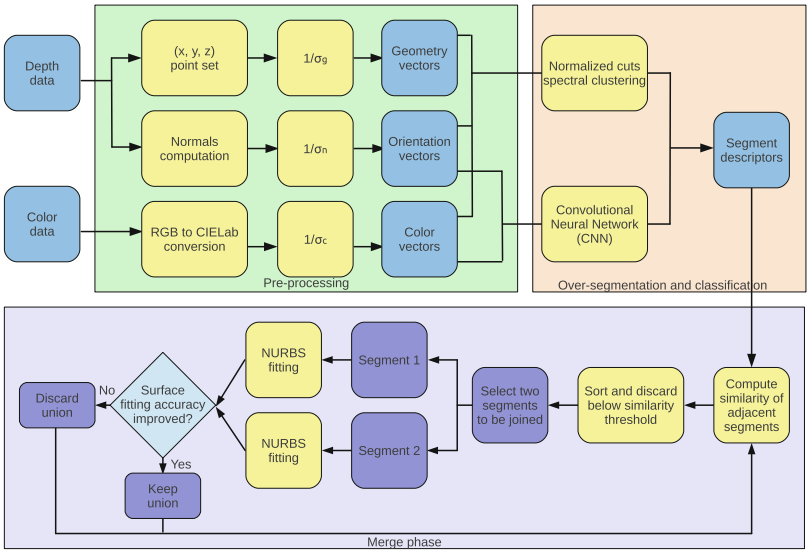


**Fig. 1.** Overview of the proposed approach

segments are sorted on the basis of the similarity of their descriptors computed by the CNN classifier and the couples with a similarity below a threshold are discarded. Starting from the most similar couple a parametric NURBS surface is fitted over each of the two segments. The same model is fitted also over the merged region obtained by fusing the two segments. Similarly to [22], the surface fitting error is computed and compared with the weighted average of the fitting error on the two merged pieces. If the error on the merged region is smaller (a hint that the two regions are part of the same surface) the merging operation is accepted, if it increases (i.e., they probably belong to two different surfaces), the merging is discarded. The procedure is repeated iteratively in a tree structure until no more merging operations are possible.

## 4 Over-Segmentation of Color and Depth Data

The proposed method takes as input a color image with its corresponding depth map. For each pixel with a valid depth value $p_i$, $i = 1, \ldots, N$ (where $N$ is the number of valid pixels) it builds a 9-dimensional vector $\mathbf{p}_i^{9D}$ containing the color, spatial and orientation information. More in detail, the first three dimensions are the $L(p_i), a(p_i), b(p_i)$ components containing the color view information converted to the CIELab perceptually uniform space. The 3D coordinates $x(p_i), y(p_i), z(p_i)$ and the surface normals $n_x(p_i), n_y(p_i), n_z(p_i)$ associated to each sample are then computed exploiting the calibration information. The over-segmentation is performed by clustering the multi-dimensional vectors containing the color, the position in the 3D space and the orientation information associated to the samples [6,22].

The clustering algorithm must be insensitive to the scaling of the point-cloud geometry and needs geometry, color and orientation to be into consistent representations. For these reasons, the geometry components are normalized by the average $\sigma_g$ of the standard deviations of the point coordinates, obtaining the vectors $[\bar{x}(p_i), \bar{y}(p_i), \bar{z}(p_i)]$. Following the same rationale, the normal vectors $[\bar{n}_x(p_i), \bar{n}_y(p_i), \bar{n}_z(p_i)]$ are computed by normalizing the three components of the orientation by the average $\sigma_n$ of their standard deviations. Color information vectors $[\bar{L}(p_i), \bar{a}(p_i), \bar{b}(p_i)]$ are also obtained by normalizing color data with the average $\sigma_c$ of the standard deviations of the $L$, $a$ and $b$ components. Finally, a 9D representation is built from the above normalized vectors, such that each point of the scene is represented as

$$\mathbf{p}_i^{9D} = [\bar{L}(p_i), \bar{a}(p_i), \bar{b}(p_i), \bar{x}(p_i), \bar{y}(p_i), \bar{z}(p_i), \bar{n}_x(p_i), \bar{n}_y(p_i), \bar{n}_z(p_i)], \quad i = 1, \ldots, N. \tag{1}$$

Normalized cuts spectral clustering [25] optimized with the Nyström method [11] is then applied to the 9D vectors in order to segment the acquired scene. Notice that the parameters of the clustering algorithm are set in order to produce a large number of segments that will later be merged in order to obtain the final solution by the method of Sect. 7.

## 5 Classification with Deep Learning

In order to be able to decide which segments from the over-segmentation need to be recombined we employed a machine learning stage based on a Convolutional Neural Network (CNN). The goal of this step is to obtain a fundamental clue to drive the merging operation described in Sect. 7 together with the NURBS surface fitting approach derived from [22]. The idea is to exploit the output of a CNN trained for a semantic segmentation task in order to produce a pixel-wise higher-level description of the input images, and use this information to compute a similarity measure between adjacent segments. The merging strategy described in Sect. 7 exploits the proposed similarity measure both in the selection of which adjacent segment pairs are going to be merged as well as in determining the order of the selection of the candidate pairs for the merging operations.

Both color and normal information is used as input to the CNN. Normalized color and normal components computed in Sect. 4 are first combined into 6D vectors representing each point of the scene as

$$\mathbf{p}_i^{cn} = [\bar{L}(p_i), \bar{a}(p_i), \bar{b}(p_i), \bar{n}_x(p_i), \bar{n}_y(p_i), \bar{n}_z(p_i)], \quad i = 1, \ldots, N. \tag{2}$$

Then, a six channel input image is produced for each scene in the dataset by arranging the vectors over the image pixels lattice.

A multi-scale architecture [9] is used to achieve a greater expressiveness without increasing the number of network parameters. Each input image is fed to the network at three different scales, both to account for the varying size at which similar objects may appear in the scene, and to take advantage of increasingly larger contexts. An overview of its structure is shown in Fig. 2.

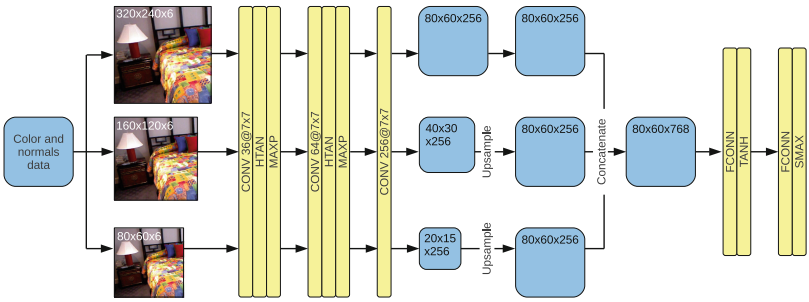


**Fig. 2.** Architecture of the Convolutional Neural Network

Similarly to [3,9], the network can be divided in two parts. In the first part a local representation of the input is extracted by applying a sequence of convolutional layers sharing their weights across the three scales. Specifically, the three input scales are feed-forwarded through three convolutional layers (denoted with CONV in Fig. 2). The first two convolutional layers are followed by a hyperbolic

tangent activation function (TANH) and a max-pooling (MAXP) layer, while the third one is applied as a simple bank of linear filters, producing the three outputs corresponding to the three scales. The outputs are then upsampled and concatenated to provide for each pixel a vector of feature descriptors. The second part of the network is composed by two fully-connected layers (FCONN), with hyperbolic tangent and soft-max (SMAX) activation functions respectively.

Input images are fed to the network at three different scales, namely 320×240, 160×120 and 80×60, both to account for the varying size at which similar objects may appear in the scene, and to take advantage of increasingly larger contexts.

In our experiments the three convolutional layers have 36, 64 and 256 filters respectively, all filters being $7 \times 7$ pixels wide, while the fully-connected layers have 1024 and 15 units respectively. The filters in the first convolutional layer are divided into 6 groups and each group is connected to one of the 6 input channels separately. In order to ease the convergence of the first layer filter weights, local contrast normalization is applied to each channel independently.

The network is trained in order to assign one out of the 15 labels to each pixel in the input image. Specifically, we clustered the 894 categories from the ground truth provided by [12] into 15 classes similarly to [3,18,31]. As in [9] we split the training process into two separate steps. The filter weights of the three convolutional layers are first trained separately by applying a simple linear classifier to the output of the first part of the network, with soft-max activation and multi-class cross-entropy loss function. Next, the weights and biases of the last two fully-connected layers are trained while keeping the convolutional weights as calculated in the previous step fixed. Again, the multi-class cross-entropy loss function is minimized.

Rather than the final predicted labels, the output of the soft-max activation function in the last fully-convolutional layer is considered in order to compute the descriptors used for the similarity measure between any two segments. The output of the soft-max, a 3D array of size $80 \times 60 \times 15$, is linearly interpolated to the size of the input image so that a descriptor vector $\mathbf{c}_i = [c_i^1, \ldots, c_i^{15}]$ is associated to each pixel $p_i$. As each descriptor vector has non-negative elements summing up to one, it can be seen as a discrete probability distribution function (PDF) associated to the pixel. The PDF $\mathbf{s}_i = [s_i^1, \ldots, s_i^k]$ associated to a segment $S_i$ can be computed simply as the average of the PDFs of the pixels belonging to the segment, i.e.,

$$\mathbf{s}_i = \frac{\sum_{j \in S_i} \mathbf{c}_j}{|S_i|}. \tag{3}$$

Given two segments $S_i$ and $S_j$, their similarity can be estimated by comparing their descriptors $\mathbf{s}_i$ and $\mathbf{s}_j$, which are actually two PDFs. An effective approach in order to compare two PDFs is to use the Bhattacharrya coefficient

$$b_{i,j} = \sum_{t=1,\ldots,k} \sqrt{s_i^t s_j^t}. \tag{4}$$

An example of the output of this approach is shown in Fig. 3. The color of the boundary between each couple of segments in the figure is proportional to the

a) b)

**Fig. 3.** Computation of $b_{i,j}$ on a sample scene (i.e., the one in the sixth row of Fig. 5): (a) $b_{i,j}$ values on the initial over-segmentation; (b) $b_{i,j}$ values on the final result after all the merging steps. The boundary of the segments have been colored proportionally to the similarity between the two touching segments (black corresponds to low $b_{i,j}$ values and white to large ones)

corresponding $b_{i,j}$ value. Notice how in Fig. 3a the boundaries between different objects correspond to low values of the coefficient, while boundaries between parts of the same object that need to be merged correspond to high $b_{i,j}$ values. In Fig. 3b it is possible to notice how the remaining boundaries at the end of the procedure of Sect. 7 typically correspond to low similarity values.

## 6 Surface Fitting on the Segmented Data

In order to evaluate if each segment corresponds to a single scene object we approximate it with a Non-Uniform Rational B-Spline (NURBS) surface [23]. By using this approach we are able to provide an appropriate geometric model for quite complex shapes, unlike competing approaches [27,28] that are limited to planar surfaces.

A parametric NURBS surface is defined as

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}}, \tag{5}$$

where $\mathbf{P}_{i,j}$ are the control points, $w_{i,j}$ the corresponding weights, $N_{i,p}$ the univariate B-spline basis functions, and $p, q$ the degrees in the $u, v$ parametric directions respectively. We set the degrees in the $u$ and $v$ directions equal to 3 and the weights all equal to one, thus our fitted surfaces are non-rational (i.e., splines). The number of surface control points are the degrees of freedom in our model and we adaptively set it depending on the number of input samples in the considered segment. This is necessary to prevent the fitting accuracy to be biased in favor of smaller segments [22]. Finally, by using Eq. (5) evaluated at the depth lattice points and equated to the points to fit (see [22] for details), we obtain an over-determined system of linear equations and we solve it in the least-squares sense thus obtaining the surface control points.

## 7 Region Merging Procedure

The final step of the proposed approach is the merging phase that recombines the large number of segments produced by the over-segmentation into a smaller number of segments representing the various structures in the scene. The procedure is summarized in the bottom part of Fig. 1 and in Algorithm 1.

**Algorithm 1.** Merge algorithm

```
Compute L_S (list of the segments)
For each segment S_i compute the set A_i of the adjacent segments.
Create list of couples of adjacent segments A_{i,j}
For each couple of adjacent segments i and j compute their similarity b_{i,j} according
to Equation (4)
Sort the list of adjacent couples A_{i,j} according to b_{i,j}
Discard the couples with a score b_{i,j} < T_sim
for all the couples in A_{i,j} do
  Compute the fitting error on the merged segment S_{i∪j}
  Check if the threshold of Equation (9) is satisfied
  if Equation (9) is satisfied then
    Remove all the couples involving S_i and S_j from A_{i,j}
    Compute the adjacent segments S_k to S_{i∪j}
    Compute A_{i∪j,k} for all the adjacent segments
    Insert the new segments in A_{i,j} and sort
  end if
  Move to next entry in A_{i,j}
end for
```

Firstly the segments are analyzed in order to detect the couples of close segments that are candidate to be joined. For this task an approach similar to [22] is used in order to build 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 lattice defined by the depth map.
2. The depth values on the shared boundary must be consistent. In order to perform this check, for each point $P_i$ in the shared boundary $C_C$ we compute the difference $\Delta Z_i$ between the depth values on the two sides of the edge. The difference must be smaller than a threshold $T_d$ for at least half of the points in the shared boundary, i.e.:

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta Z_i \leq T_d)|}{|P_i : P_i \in C_C|} > 0.5 \tag{6}$$

3. The color values must also be similar on both sides of the common contour. The approach is the same used for depth data except that the color difference

in the CIELab space $\Delta C_i$ is used instead of the depth value. More in detail, being $T_c$ the color threshold:

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta C_i \leq T_c)|}{|P_i : P_i \in C_C|} > 0.5 \tag{7}$$

4. Finally the same approach is used also for normal data. In this case the angle between the two normal vectors $\Delta\theta_i$ is compared to a threshold $T_\theta$:

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta\theta_i \leq T_\theta)|}{|P_i : P_i \in C_C|} > 0.5 \tag{8}$$

If all the conditions are satisfied the two segments are marked as adjacent (for the results we used $T_d = 0.2$ m, $T_c = 8$ and $T_\theta = 4°$). For more details on this step see [22]. Notice that the evaluation of the color, depth and orientation consistency on the couples of close segments can be skipped in order to simplify and speed-up the approach, but it allows to slightly improve the performances of the proposed method.

At this point the algorithm analyzes the couples of adjacent segments and computes the similarity between the two segments in each couple as described in Sect. 5.

The next step consists in sorting the couples of adjacent segments based on the $b_{i,j}$ values, that is, according to how the two segments were estimated to be similar during the machine learning stage. Furthermore the couples of segments with a similarity value $b_{i,j}$ below a threshold $T_{sim}$ are discarded and they will not be considered for the merging operations (for the results we used $T_{sim} = 0.75$). The rationale behind this is to avoid merging segments with different properties since they probably belong to distinct objects and parts of the scene.

The algorithm then selects the couple with the highest similarity score. Let us denote with $S_{i^*}$ and $S_{j^*}$ the two segments in the couple and with $S_{i^* \cup j^*}$ the segment obtained by merging the two segments. A NURBS surface is fitted on each of the two regions $i^*$ and $j^*$ (see Sect. 6). The fitting error, i.e., the Mean Squared Error (MSE) between the actual surface and the fitted surface, is computed for both segments thus obtaining the values $e_{i^*}$ and $e_{j^*}$. The fitting error $e_{i^* \cup j^*}$ on segment $S_{i^* \cup j^*}$ is also computed and compared to the weighted average of the errors on $S_i^*$ and $S_j^*$:

$$e_{i^*}|S_{i^*}| + e_{j^*}|S_{j^*}| > e_{i^* \cup j^*}(|S_{i^*}| + |S_{j^*}|) \tag{9}$$

If the fitting accuracy is improved, i.e., the condition of Eq. (9) is satisfied, the two segments are merged together, otherwise the merging operation is discarded. If the two segments $S_i^*$ and $S_j^*$ are merged, all the couples involving them are removed from the list $L_S$. The adjacency information is then updated by considering the union $S_{i^* \cup j^*}$ as adjacent to all the segments that were previously adjacent to any of the two segments. The descriptor $\mathbf{s}_{i^* \cup j^*}$ associated to $S_{i^* \cup j^*}$ is computed using Eq. (3) and the similarity score is computed for all the newly created couples involving the segment $S_{i^* \cup j^*}$ created by the merging operation.

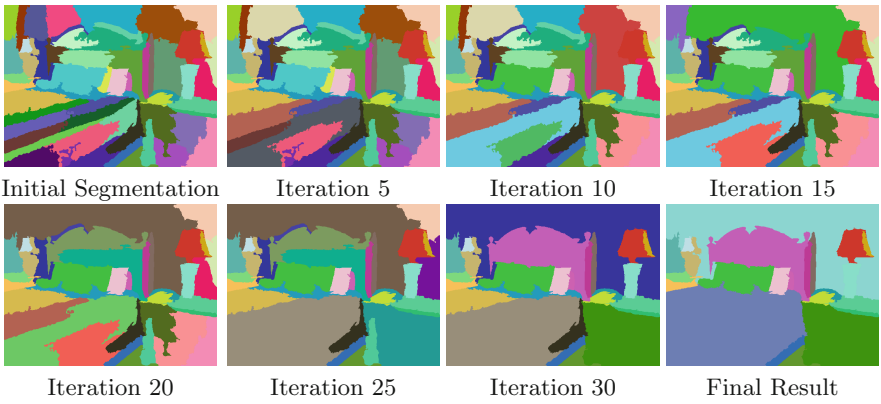


**Fig. 4.** Example of the merging procedure on the scene of Fig. 5, row 6. The images show the initial over-segmentation, the merging output after 5, 10, 15, 20, 25, 30 iterations and the final result (iteration 34). (Best viewed in the color version online)

Finally the new couples are inserted in the list $L_S$ at the positions corresponding to their similarity score (provided their similarity is bigger than $T_{sim}$, otherwise they are discarded). The algorithm then selects the next couple in the sorted list and the procedure is repeated until no more segments can be considered for the merging operation. The procedure is summarized in Algorithm 1 and its progress on a sample scene is visualized in Fig. 4. The sequence of merging steps on some scenes is also shown in the videos available at http://lttm.dei.unipd.it/paper_data/deepnurbs.

## 8 Experimental Results

In order to evaluate the performances of the proposed method we tested it on the NYU-Depth V2 dataset (NYUDv2) [26]. This dataset has been acquired with the Kinect and contains 1449 depth and color frames from a variety of indoor scenes. The updated versions of the ground truth labels provided by the authors of [12] has been used.

The dataset has been split in two parts using the subdivision of [12]. In order to get the results on the full dataset we performed two independent tests. In the first test the CNN has been trained using the first part with the corresponding ground truth labels. The trained network has then been used to compute the descriptors for the scenes in the second part. In the second test we swapped the train and test sets and performed the same procedure. The semantic classification is not the main target of this work, however notice that the proposed deep learning architecture, if used for classification purposes, is able to obtain an average mean pixel accuracy on the dataset of 52.8 % that is similar to the results reported in [3]. To mitigate the effect of overfitting the dataset has been expanded by randomly rotating each sample by an angle between $-6$ and $6°$.

Moreover, quadratic regularization with coefficient 0.001 has been used. The network weights have been updated using stochastic gradient descent, with initial learning rate equal to 0.01 and constant decay by a factor 0.5 every 15 epochs.

**Table 1.** Average values of the VoI and RI metrics on the 1449 scenes of the NYUDv2 dataset for the proposed approach and for some state-of-the-art approaches from the literature

| Approach | VoI | RI |
|---|---|---|
| Hasnat et al. [17] | 2.29 | 0.90 |
| Hasnat et al. [16] | 2.20 | **0.91** |
| Ren et al. [24] | 2.35 | 0.90 |
| Felzenszwalb and Huttenlocher [10] | 2.32 | 0.81 |
| Taylor and Cowley [28] | 3.15 | 0.85 |
| Dal Mutto et al. [6] | 3.09 | 0.84 |
| Pagnutti and Zanuttigh [22] | 2.23 | 0.88 |
| **Proposed method** | **1.93** | **0.91** |

Table 1 shows the comparison between our approach and some state-of-the-art approaches on this dataset (for the other approaches we collected the results from [17,22]). The compared approaches are the clustering and region merging method of [17], the MRF scene labeling scheme of [24], that exploits Kernel Descriptors and SVM for machine learning, a modified version of [10] that accounts also for geometry information, the dynamic programming scheme of [28], the clustering-based approach of [6] and the region merging scheme of [22]. Notice that the latter is also based on over-segmentation and NURBS fitting but it does not have a machine learning stage and it uses a different merging procedure. The comparison between the two approaches can be a hint of the improvement provided by the use of the CNN descriptors.

The results have been evaluated by comparing the results with ground truth data using two different metrics, i.e., the Variation of Information (VoI) and the Rand Index (RI). For a detailed description of these metrics see [1], notice that for the VoI metric a lower value is better while a higher one is better for RI. The average VoI score of our method is 1.93. According to this metric our approach is the best among the considered ones with a significant gap with respect of all the competing approaches. If the RI metric is employed the average score is 0.91. This value is better than the one of the schemes of [6,10,17,22,24,28] and is exactly the same of the best competing approach, i.e., [16]. Furthermore our approach does not assume the presence of planar surfaces thanks to the NURBS surface fitting scheme, while some competing ones (e.g., [16,17,28]) strongly rely on this clue that gives very good results on the NYUDv2 dataset where most of the surfaces are planar, but reduces the capability of the approaches to generalize to different kind of scenes with non-planar surfaces.

Some visual results for the proposed approach are shown in Fig. 5 while some videos showing the merging steps leading to the presented results are available


**Fig. 5.** Segmentation of some sample scenes from the NYUDv2 dataset. The figure shows the color images, the initial over-segmentation and the final result for scenes 72, 330, 450, 846, 1105, 1110 and 1313 (Color figure online)

at http://lttm.dei.unipd.it/paper_data/deepnurbs. By looking at the images it is possible to see how the approach is able to efficiently deal with different challenging situations and to various scene types. The initial over-segmentation typically divides the background and the large objects in several pieces but

they are properly recombined by the proposed approach, this is due to the CNN descriptors that allow to recognize which segments belong to the same structure. Notice how the contours of the objects are well defined and there are not noisy small segments in proximity of edges as in other approaches. The approach is also able to correctly segment most of the objects in the scene even if a few inaccuracies are present only on very small objects.

The proposed method has been implemented using the Theano deep learning library [29]. On a standard desktop PC (an i7-4790 with 16 GB of Ram) the segmentation of an image with the corresponding depth map takes less than two minutes. More in detail, the initial over-segmentation takes most of the time (i.e., 87 s), but notice that it can be easily replaced with other super-pixel segmentation techniques. The CNN classification takes only about 3.7 s on the CPU. Finally the merging procedure of Sect. 7 requires around 20 s. Notice that the current implementation has not been optimized, in particular we used the GPU only for the training of the CNN but not for the classification and segmentation tasks.

## 9 Conclusions and Future Work

In this paper we proposed a novel joint color and depth segmentation scheme. An iterative merging procedure starting from an initial over-segmentation is employed. The key idea consists in controlling the merging operation by using together geometrical clues and an estimation of the segments similarity computed with Convolutional Neural Networks. The adopted surface fitting comparison makes it possible to avoid merging segments belonging to different surfaces, and the proposed similarity metric based on the comparison of the descriptors computed by the CNN proves to be reliable. As shown by experimental results, our method achieves state-of-the-art performances on the challenging NYUDv2 dataset.

Further research will explore the performances of the proposed approach in the semantic segmentation task. The exploitation of surface fitting information into the CNN classifier will also be considered. Finally different deep learning architectures including Fully Convolutional Networks [20] and hypercolumns [15] will be tested into the proposed framework.

**Acknowledgments.** We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for the training of the CNN.

## References

1. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(5), 898–916 (2011)
2. Bleiweiss, A., Werman, M.: Fusing time-of-flight depth and color for real-time segmentation and tracking. In: Kolb, A., Koch, R. (eds.) Dyn3D 2009. LNCS, vol. 5742, pp. 58–69. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03778-8_5

3. Couprie, C., Farabet, C., Najman, L., LeCun, Y.: Indoor semantic segmentation using depth information. In: International Conference on Learning Representations (2013)
4. Couprie, C., Farabet, C., Najman, L., Lecun, Y.: Convolutional nets and watershed cuts for real-time semantic labeling of RGBD videos. J. Mach. Learn. Res. **15**(1), 3489–3511 (2014)
5. Dal Mutto, C., Zanuttigh, P., Cortelazzo, G.: Scene segmentation assisted by stereo vision. In: Proceedings of 3DIMPVT 2011, Hangzhou, China, May 2011
6. Dal Mutto, D., Zanuttigh, P., Cortelazzo, G.: Fusion of geometry and color information for scene segmentation. IEEE J. Sel. Top. Sig. Process. **6**(5), 505–521 (2012)
7. Deng, Z., Todorovic, S., Jan Latecki, L.: Semantic segmentation of RGBD images with mutex constraints. In: Proceedings of International Conference on Computer Vision (ICCV), pp. 1733–1741 (2015)
8. Erdogan, C., Paluri, M., Dellaert, F.: Planar segmentation of RGBD images using fast linear fitting and Markov chain monte carlo. In: Proceedings of the CRV (2012)
9. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1915–1929 (2013)
10. Felzenszwalb, P., Huttenlocher, D.: Efficient graph-based image segmentation. Int. J. Comput. Vis. **59**(2), 167–181 (2004)
11. Fowlkes, C., Belongie, S., Chung, F., Malik, J.: Spectral grouping using the nyström method. IEEE Trans. Pattern Anal. Mach. Intell. **26**(2), 214–225 (2004)
12. Gupta, S., Arbelaez, P., Malik, J.: Perceptual organization and recognition of indoor scenes from RGB-D images. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2013)
13. Gupta, S., Arbeláez, P., Girshick, R., Malik, J.: Indoor scene understanding with RGB-D images: bottom-up segmentation, object detection and semantic segmentation. Int. J. Comput. Vis. **112**, 1–17 (2014)
14. Gupta, S., Girshick, R., Arbeláez, P., Malik, J.: Learning rich features from RGB-D images for object detection and segmentation. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8695, pp. 345–360. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10584-0_23
15. Hariharan, B., Arbeláez, P., Girshick, R., Malik, J.: Hypercolumns for object segmentation and fine-grained localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 447–456 (2015)
16. Hasnat, M.A., Alata, O., Trémeau, A.: Joint color-spatial-directional clustering and region merging (JCSD-RM) for unsupervised RGB-D image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **38**(11) (2016)
17. Hasnat, M.A., Alata, O., Trémeau, A.: Unsupervised RGB-D image segmentation using joint clustering and region merging. In: Proceedings of British Machine Vision Conference (BMVC) (2014)
18. Hickson, S., Essa, I., Christensen, H.: Semantic instance labeling leveraging hierarchical segmentation. In: 2015 IEEE Winter Conference on Applications of Computer Vision, pp. 1068–1075. IEEE (2015)
19. Höft, N., Schulz, H., Behnke, S.: Fast semantic segmentation of RGB-D scenes with GPU-accelerated deep neural networks. In: Lutz, C., Thielscher, M. (eds.) KI 2014. LNCS (LNAI), vol. 8736, pp. 80–85. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11206-0_9
20. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3431–3440 (2015)

21. Pagnutti, G., Zanuttigh, P.: Scene segmentation from depth and color data driven by surface fitting. In: Proceedings of IEEE International Conference on Image Processing (ICIP), pp. 4407–4411. IEEE (2014)
22. Pagnutti, G., Zanuttigh, P.: Joint color and depth segmentation based on region merging and surface fitting. In: Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP) (2016)
23. Piegl, L., Tiller, W.: The NURBS Book, 2nd edn. Springer, New York (1997)
24. Ren, X., Bo, L., Fox, D.: RGB-(D) scene labeling: features and algorithms. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
25. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)
26. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7576, pp. 746–760. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33715-4_54
27. Srinivasan, N., Dellaert, F.: 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)
28. Taylor, C.J., Cowley, A.: Parsing indoor scenes using RGB-D imagery. In: Robotics: Science and Systems. vol. 8, pp. 401–408 (2013)
29. Theano Development Team: Theano: a Python framework for fast computation of mathematical expressions. arXiv e-prints abs/1605.02688, May 2016. https://arxiv.org/abs/1605.02688
30. Wallenberg, M., Felsberg, M., Forssén, P.-E., Dellen, B.: Channel coding for joint colour and depth segmentation. In: Mester, R., Felsberg, M. (eds.) DAGM 2011. LNCS, vol. 6835, pp. 306–315. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23123-0_31
31. Wang, A., Lu, J., Wang, G., Cai, J., Cham, T.-J.: Multi-modal unsupervised feature learning for RGB-D scene labeling. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 453–467. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10602-1_30
32. Zanuttigh, P., Marin, G., Dal Mutto, C., Dominio, F., Minto, L., Cortelazzo, G.M.: Time-of-Flight and Structured Light Depth Cameras. Springer, Heidelberg (2016)