Segmentation and Semantic Labeling of RGBD Data with Convolutional Neural Networks and Surface Fitting

Giampaolo Pagnutti [1,*], Ludovico Minto[1,*], Pietro Zanuttigh[1,*,**]

[1]Department of Information Engineering, University of Padova, Italy
[*]All authors equally contributed to the paper
[**]zanuttigh@dei.unipd.it

Abstract: We present an approach for segmentation and semantic labeling of RGBD data exploiting together geometrical cues and deep learning techniques. An initial over-segmentation is performed using spectral clustering and a set of NURBS surfaces is fitted on the extracted segments. Then a Convolutional Neural Network (CNN) receives in input color and geometry data together with surface fitting parameters. The network is made of nine convolutional stages followed by a softmax classifier and produces a vector of descriptors for each sample. In the next step an iterative merging algorithm recombines the output of the over-segmentation into larger regions matching the various elements of the scene. The couples of adjacent segments with higher similarity according to the CNN features are candidate to be merged and the surface fitting accuracy is used to detect which couples of segments belong to the same surface. Finally, a set of labeled segments is obtained by combining the segmentation output with the descriptors from the CNN. Experimental results show how the proposed approach outperforms state-of-the-art methods and provides an accurate segmentation and labeling.

1. Introduction

Recent achievements in the computer vision field allowed to obtain a relevant improvement in algorithms dealing with the semantic segmentation task. In particular we focus on two key advancements. The first is the development of more powerful machine learning algorithms, specially deep learning techniques, that allowed to better understand the semantic content of the images. The second is the introduction of consumer depth sensors that allowed to easily acquire the 3D geometry of the scene, a very useful source of information overcoming several limitations and ambiguities of color information.

Clustering techniques, e.g., normalized cuts spectral clustering [1], are an effective approach for segmentation well-suited for the extension to the joint segmentation of color and geometry information [2]. However, the normalized cuts algorithm has a bias towards producing regions of similar sizes and for this reason it is challenging to properly separate all the objects avoiding at the same time to over-segment the scene.

The problem can be solved by exploiting an over-segmentation performed with normalized cuts followed by an iterative region merging approach scheme. This work follows this rationale and uses together two different cues in order to decide which segments must be merged. The first is a segment similarity measure obtained from the descriptors computed by a Convolutional Neural Network (CNN). The other is obtained, for a given couple of segments, by fitting a Non-Uniform

Rational B-Spline (NURBS) on each segment taken separately and on their union. The fitting accuracies are then compared and the two segments are merged whenever their union results in an increased fitting accuracy [3]. Notice how this idea allows to detect if the two segments are part of the same scene surface (and thus are candidate to be merged) and to properly handle also non-planar object and surfaces.

The approach was firstly proposed in [4], this extended journal version exploits a more advanced classification algorithm and presents a combined segmentation and semantic labeling approach. In particular, with respect to the conference work a deeper Convolutional Neural Network architecture has been employed and surface curvatures and fitting error have also been used inside the CNN (up to our knowledge, this is the first time this kind of data is used in a deep learning framework). Furthermore orientation data have been replaced by HHA descriptors [5] (disparity, height and orientation angle) and the experimental evaluation now addresses also the semantic labeling task.

2. Related Work

Segmentation of RGBD data has been the subject of many research works (a recent review is contained in [6]). Clustering techniques are commonly used for image segmentation and they have been exploited for the combined segmentation of color and geometry by using multi-channel feature vectors [7, 2]. The method of [8] performs multiple clusterings with K-means and combines them together.

Region splitting and growing methods are another commonly used approach. The approach of [9] starts from an over-segmentation and combines segments corresponding to the same planar region by exploiting a method based on Monte Carlo Markov Chain and Rao Blackwellization. The method of [10] exploits region splitting and iteratively refines the segmentation by recursively splitting regions that do not correspond to a single surface. The work of [3] uses the same criteria in a bottom-up approach starting from an over-segmentation of the scene. Gupta et al. [11] use a hierarchical segmentation starting from edge detection information. The method of [12] starts with an over-segmentation computed with watersheds and then exploits a hierarchical approach. Hasnat et al. [13, 14] start from a joint clustering method on the color, geometry and orientation data and then apply a region merging algorithm searching for planar regions. Finally, [15] uses dynamic programming to extract planar surfaces.

A closely related problem is semantic segmentation, i.e., joint segmentation and labeling of the segments. This problem is typically solved by using machine learning approaches. Ren et al. [16] exploit an over-segmentation with Markov Random Fields followed by a tree-structured algorithm. The works of [17] and [18] instead use Conditional Random Fields (CRF). The method of [19] also uses a CRF model that captures planar surfaces and dominant lines. Another work based on CRFs is [20], that combines them with decision forests. The approach of [17] combines CRF with mutex constraints based on geometry data while the approach of [18] combines 2D segmentation, 3D geometry data and contextual information. The work of [21] is instead based on a proposal process that generates spatial layout hypotheses followed by a sequential inference algorithm.

Recently, deep learning algorithms have been exploited for the semantic segmentation task [22, 23, 24]. One of the first solutions exploiting deep learning is [23], that uses a multiscale Convolutional Neural Network. The method of [24] is able to achieve a very high accuracy by exploiting Fully Convolutional Networks. The approach of [5] uses a CNN working on geometric features. Wang et al. [25] use two different CNNs, one for color and one for depth, and a feature

transformation network able to separate the information common to the two modalities from the one specific of each modality. The work of [26] jointly solves the semantic labeling together with depth and normal estimation using a multiscale CNN. Finally the method of [27] uses deep learning to extract superpixel features that are then classified with SVMs.

The combination of segmentation and semantic labeling has been considered in [28], that employs multiple segmentations to generate the regions to be used for object detection and recognition. Multiple segmentations are used also by [29] that deals with the problem of object segmentation using a sequence of constrained parametric min-cut problems.

Even if several different approaches for this task have been proposed, the proposed method has some original features not present in previous works, in particular the usage of surface fitting cues and the strict coupling between the semantic labeling and the segmentation, with the idea of exploiting the deep learning descriptors to control the iterative merging together with fitting cues. This allows it to obtain very accurate results even if the deep learning framework is simpler than some of the related works. Furthermore, while many related works strongly rely on the planar surface assumption the proposed model properly accounts for arbitrarily shaped regions.

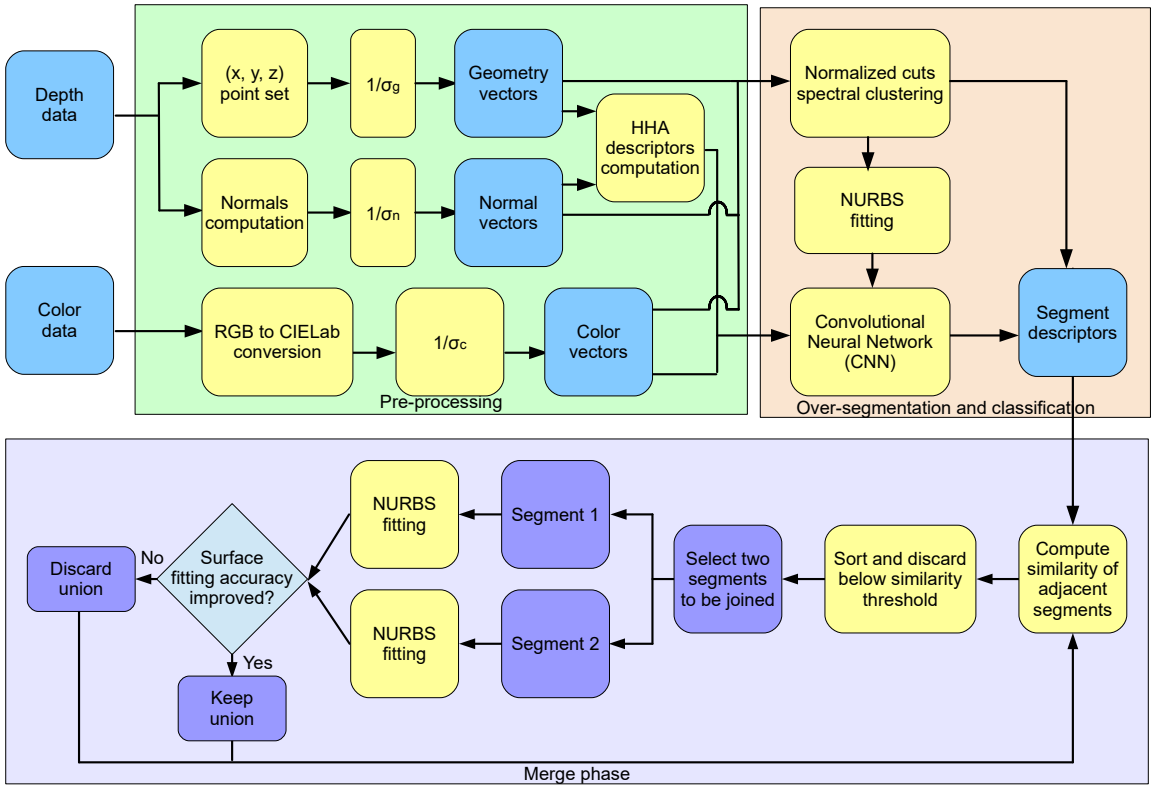

Fig. 1. *General architecture of the proposed method*

3. Architecture of the proposed method

The proposed method is organized in 3 main blocks as shown in Figure 1. Color and depth data are used to compute a set of nine-dimensional vectors containing the 3D location, the surface normal information and the color representation for each sample. Then the algorithm computes

an over-segmentation of the scene using the 9D vectors exploiting a spectral clustering algorithm derived from [2, 3] (Section 4). After performing the over-segmentation, a NURBS surface is fitted over each segment using the approach detailed in Section 5. The fitting error and the curvature information for the fitted surfaces are also extracted. This information is fed to a Convolutional Neural Network together with color and HHA descriptors. The network (Section 6) is trained for the semantic labeling task and computes a descriptor vector for each sample representing the probabilities of the various classes at the pixel location. The descriptors are aggregated inside each segment in order to obtain a unique descriptor for the segment. The third step is an iterative region merging algorithm (Section 7). It firstly analyzes the segments and computes an adjacency map. In the map two segments are marked as adjacent if they are connected and have compatible color and geometry data on their shared boundary. The similarity between CNN descriptors is then used to sort the couples of adjacent segments. Couples with a low similarity score according to the CNN descriptors are discarded and the remaining ones are processed in order of similarity. After selecting a couple, a NURBS surface is fitted over the merged region obtained by joining the two segments and the accuracy of the fitting is compared with the ones of the two segments. If the fitting error decreases after the merging (a hint that the two segments belong to the same surface) the merging is performed, otherwise the operation is discarded. The algorithm proceeds iteratively until there are no more segments to merge. Finally the probability vectors from the CNN are used to assign a label to each of the final segments in order to get also the semantic information. The obtained results are presented in Section 8.

4. Over-segmentation of RGBD Data

To segment the acquired scene, a multi-dimensional vector enclosing the color and spatial information is built for every pixel p_i of the input image with valid depth information. The first three components $L(p_i), a(p_i), b(p_i)$ contain the color information in the perceptually uniform CIELab space. Then, the 3D position $x(p_i), y(p_i), z(p_i)$ and the surface normal $n_x(p_i), n_y(p_i), n_z(p_i)$ are considered (the 3D coordinates are calculated based on the sensor calibration information while for normal computation we used the approach of [30]). To achieve a consistent representation for these different types of information, the color, geometry and orientation average standard deviations σ_c, σ_g and σ_n are computed from the input image and depth map. Then, each of the 3 set of components is normalized by the corresponding standard deviation, providing normalized vectors $[\bar{L}(p_i), \bar{a}(p_i), \bar{b}(p_i)]$, $[\bar{x}(p_i), \bar{y}(p_i), \bar{z}(p_i)]$ and $[\bar{n}_x(p_i), \bar{n}_y(p_i), \bar{n}_z(p_i)]$ and finally the nine-dimensional representation:

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

These 9D vectors representing the acquired scene are then segmented [2, 3] by means of Normalized Cuts spectral clustering [1] with Nyström acceleration [31]. The algorithm parameters are set in order to create a large number of segments (over-segmentation), since the final result will be produced by the merging procedure of Section 7.

5. NURBS Fitting on the Segmented Regions

The following step is the approximation of each segment with a Non-Uniform Rational B-Spline (NURBS) surface [32]. This is used twice in the proposed method: firstly, in order to produce

an additional set of input cues for the CNN classifier (Section 6), secondly, in order to evaluate if segments produced by the merging operations correspond to a single scene object (Section 7).

By means of NURBS we can approximate each segment (including non-planar regions) with a continuous parametric surface $\mathbf{S}(u, v)$, computed by solving an over-determined system of linear equations in the least-squares sense. We refer the reader to [3] for the details on how we select the NURBS parameters (e.g., the degrees) and how we setup the linear system. We underline that in our formulation the number of surface control points, corresponding to the degrees of freedom of the model, is proportional to the number of pixels in the segment to approximate. This prevents the fitting accuracy to be better on smaller segments, favoring over-segmentation in the final result [3]. Moreover, the usage of NURBS surfaces provides a geometric model suitable for arbitrary shapes, differently from several competing schemes [15, 9] that are more appropriate for scenes where most surfaces are planar.

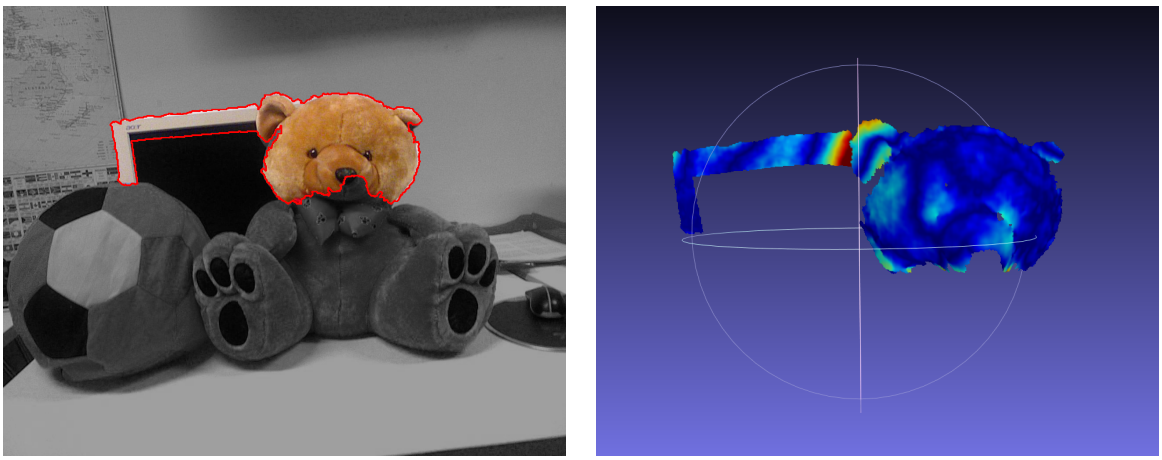

Fig. 2. *Fitting error for a NURBS surface approximating a segment containing 2 different objects. The red areas correspond to larger fit error. Notice how the large fit error between the teddy head and the monitor reveals that the two segments do not actually belong to the same object.*

After fitting the NURBS surfaces, two additional clues can be associated to each sample. The first one is the fitting error, i.e., the distance between each 3D position acquired by the sensor and the corresponding location on the fitted surface. This will be used both as an input for the CNN classifier and to recognize if a segment contains a single object (for segments, the Mean Squared Error will be considered). Notice how a large fitting error is a hint of the fact that a segment covers multiple objects, as exemplified in Fig. 2. The second one is given by the two principal curvatures (i.e., the maximum and minimum local curvature values, see [33]) at each pixel location. These quantities are intrinsically related to the geometric shape of the fitted surface, then we use them as an additional input channel for the CNN classifier.

6. Classification with Deep Learning

In this step we employ a machine learning stage in order to produce classification data for the input scene, that is used not only to produce the semantic labels but also to decide which regions of the over-segmentation should belong to the same segment in the final segmentation.

The idea is to exploit the output of a Convolutional Neural Network (CNN) trained for semantic

image segmentation in order to compute a pixel-wise high-level description of the input scene. Specifically, a descriptor vector is associated to each pixel by considering the final layer of the network, a standard softmax classifier. This information is then used to compute a similarity score between couples of adjacent segments and, at the same time, to provide the input image with semantic labels. In particular, the proposed similarity score is exploited to drive the merging procedure detailed in Section 7, using it both to decide whether any two adjacent segments should be merged together as well as to determine the order in which candidate couples of segments are selected for the merging operations.

The CNN takes in input various cues:

- Color data, represented by the 3 components in the RGB color space.
- The geometry information. We represented it with three channels containing, for each sample, the horizontal disparity h_1, the height above the floor h_2, and the angle of the normal with the vertical direction a. This representation, typically abbreviated with HHA, has been introduced by [5] and provided better performances than the direct usage of geometry and orientation information.
- Surface fitting information, represented with a 3D vector containing the fitting error f and the two principal curvatures c_1 and c_2.

For each point of the scene, a 9D vector is used to store this information as

$$\mathbf{p}_i^{cn} = [R(p_i), G(p_i), B(p_i), h_1(p_i), h_2(p_i), a(p_i), f(p_i), c_1(p_i), c_2(p_i)]. \quad (2)$$

Finally, the vectors are arranged over the image pixel lattice to produce, for each scene in the dataset, a 9-channel input representation.

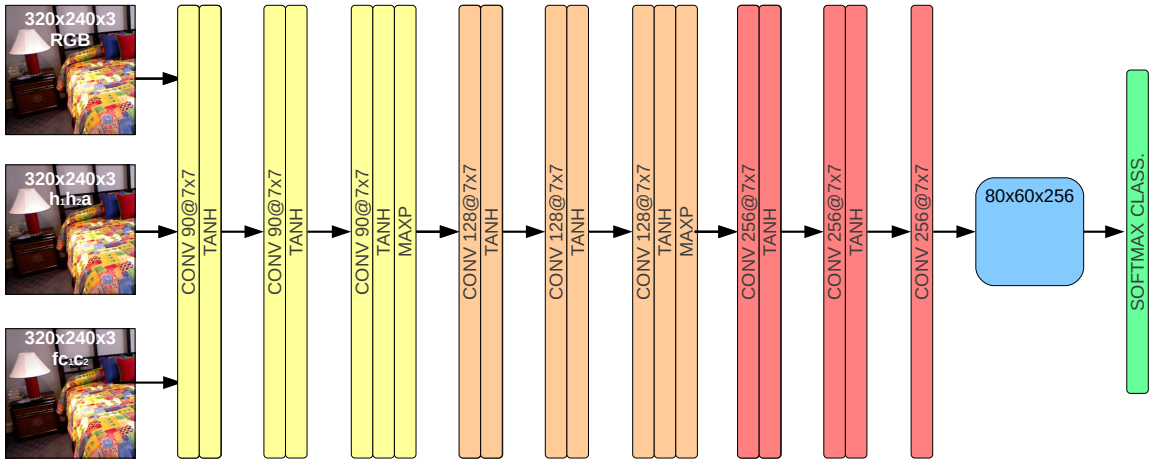

Fig. 3. *Layout of the proposed Convolutional Neural Network*

An overview of the employed network structure is shown in Fig. 3. The network has been constructed starting from the architecture employed in [34, 35, 4] by extending each of the original layers into a group of 3 layers. In order to avoid a too large increase in computation time no multi-scale input representation has been used. A sequence of convolutional layers is applied in order to extract a local representation of the input.

More in detail, each 9-channel input image passes through nine convolutional layers, arranged into three main blocks each one containing three layers (see Fig. 3). Each block corresponds to one of the layers in our previous approach [4] and works with a constant resolution and number of filters. Moving from one block to the next instead the resolution is reduced of a factor of 2 and the number of filter increases. Every block is made up to three convolutional layers (CONV) each followed by a hyperbolic tangent activation function (TANH). The first two blocks have also a final max-pooling (MAXP) layer, while the last convolutional layer of the last block does not have any activation function. Finally, a pixel-wise softmax classifier is applied on top of the last convolutional layer.

In order to reduce the computation time, input data are fed to the CNN at the reduced resolution of 320×240. The convolutional layers have 90 filters in the first block, 128 in the second block and 256 in the last one (notice that the layers in the last blocks work at a lower resolution, thus an higher number of filters can be used without affecting too much the computation time). All filters have a size of 7×7 pixels, while the final softmax classifier has a weight matrix of size 256×14 and no bias.

The first layer filters are arranged into 9 groups so that filters in the i-th group are connected to the i-th input channel only. Also, local contrast normalization is applied to each input channel independently, allowing filter weights in the first convolutional layer to converge faster.

The network is trained to produce a semantic segmentation of the input image by labeling each pixel in the scene with one out of 14 different semantic labels. To this aim, a multi-class cross-entropy loss function is minimized throughout the training process.

7. Region Merging Procedure

The next step is the merging procedure, that starts from the initial over-segmentation and iteratively joins couples of segments to finally obtain the objects of the scene. The process is visualized in the bottom half of Fig. 1 and summarized in Algorithm 1.

The procedure first identifies the couples of segments that are suitable to be merged. To this aim, it creates an adjacency matrix, storing for each couple of segments whether they are *adjacent* (that is, candidate for merging) or not. Two segments are considered as *adjacent* if the following conditions hold (see [3] for additional details):

1. They must be neighboring on the grid given by the depth map.
2. The depth information must be compatible along the shared boundary. Precisely, the difference ΔZ_i between the depth values on the two sides of the edge is computed for each point P_i in the common boundary C_C. This difference must be smaller than a threshold T_d along at least half of the boundary, i.e.,:

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta Z_i \leq T_d)|}{|P_i : P_i \in C_C|} > 0.5 \tag{3}$$

3. Also the color information must be consistent. A condition similar to the one used for the depth data is required for the color difference in the CIELab space ΔC_i with a threshold T_c:

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta C_i \leq T_c)|}{|P_i : P_i \in C_C|} > 0.5 \tag{4}$$

4. The same is required for the orientation information, the angle between the two normal vectors $\Delta\theta_i$ being compared to a threshold T_θ:

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta\theta_i \leq T_\theta)|}{|P_i : P_i \in C_C|} > 0.5 \tag{5}$$

If the above conditions are fulfilled, the two segments are considered as adjacent. If necessary in order to reduce computation time, this procedure can be dropped and replaced by the assumption that all the connected segments are adjacent with a limited impact on the algorithm performances.

Subsequently, for each couple of adjacent segments the similarity $b_{i,j}$ is computed from the information inferred during the machine learning stage. The exploited idea is that, apart from predicting the semantic labels, the output of the softmax classifier can also be used to produce a descriptor vector associated to each segment and, in the end, to compute a similarity score for any couple of segments. For each pixel p_i, a descriptor vector $\mathbf{c}_i = [c_i^1, \ldots, c_i^{14}]$ is extracted from the output of the softmax classifier. Notice that a linear interpolation is applied in order to resize the output from its actual size (i.e., $80 \times 60 \times 14$ pixels) to the size of the input image. Each descriptor vector can be considered as a discrete probability distribution (PDF) associated to the corresponding pixel, since its elements are non-negative values summing up to 1.

A probability density function $\mathbf{s}_i = [s_i^1, \ldots, s_i^{14}]$ can be associated to each segment S_i as well, by simply computing the average of the PDFs associated to the pixels belonging to the segment, i.e.,

$$\mathbf{s}_i = \frac{\sum_{j \in S_i} \mathbf{c}_j}{|S_i|}. \tag{6}$$

Given two segments S_i and S_j, an effective approach in order to estimate their similarity is to compute the Bhattacharrya coefficient between their PDFs $\mathbf{s}_i$ and $\mathbf{s}_j$ respectively, i.e.,

$$b_{i,j} = \sum_{t=1,..,14} \sqrt{s_i^t s_j^t}. \tag{7}$$

As an example, Fig. 4 depicts the proposed similarity score between touching segments on a sample image. The lower is the similarity $b_{i,j}$ between segments, the darker is the color of the boundary between them. As can be seen in Fig. 4a, different objects usually have a low similarity value while segments belonging to the same object tipically share higher $b_{i,j}$ values (lighter boundaries). Notice how in Fig. 4b the boundaries between segments at the very end of the merging stage (see Section 7) are more likely to correspond to low similarity scores.

The couples are placed in a priority queue Q_A sorted based on their similarity values $b_{i,j}$, and the ones with similarity $b_{i,j}$ smaller than a threshold T_{sim} are discarded so that they will not be considered for the merging operations (we used $T_{sim} = 0.77$ in the results). The aim is to prevent segments with low similarity to be merged, since it is reasonable to expect that they correspond to different objects or portions of the scene.

The algorithm then processes the couple with the highest similarity score. Let S_{i^*} and S_{j^*} be the two segments and let $S_{i^* \cup j^*}$ be their union. A NURBS surface is fitted on each of the two regions i^* and j^* (see Section 5) and the fitting error, i.e., the Mean Squared Error (MSE) between the actual and the fitted surface, is computed for both segments providing the values e_{i^*} and e_{j^*}.

a) b)

Fig. 4. *Similarity values $b_{i,j}$ computed on a sample scene: a) $b_{i,j}$ values between segments in the initial over-segmentation; b) $b_{i,j}$ values between segments at the end of the merging procedure. The color of the boundary between any two touching segments is proportional to their similarity score (white corresponds to high $b_{i,j}$ values and black to low ones)*

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^*:

$$e_{i^*}|S_{i^*}| + e_{j^*}|S_{j^*}| > e_{i^* \cup j^*}(|S_{i^*}| + |S_{j^*}|) \tag{8}$$

If Equation (8) is satisfied, i.e., the fitting error is reduced, then the two segments are joined, otherwise the union of the two segments is discarded. If the joining of S_i^* and S_j^* is performed, all the couples including any of the two segments are deleted from Q_A. The priority queue is then updated by considering $S_{i^* \cup j^*}$ adjacent to all segments that were previously adjacent to S_{i^*} or to S_{j^*} and by adding the corresponding couples if their similarity score is greater than or equal to T_{sim}. In order to compute the similarity of the new couples, the descriptor vector $\mathbf{s}_{i^* \cup j^*}$ associated to $S_{i^* \cup j^*}$ is first calculated with Equation (6), then Equation (7) is used.

The next couple to be processed is then removed from the front of the queue and the algorithm iterates until no more couples are present in the queue.

After getting the final segmentation, a semantic label is also computed for each segment by checking the descriptors of all the pixels in the segment (computed in Section 6) and assigning the most common class to the segment. Notice that average and max pooling have also been considered for this task, but they did not lead to better performances than this simple strategy.

Algorithm 1 summarizes the whole merging procedure, while some examples of intermediate steps are shown in Fig. 5 and in the videos available as additional material.

8. Experimental Results

We tested the proposed approach on the NYU Depth Dataset V2 (NYUD2) [12]. The NYUD2 dataset is made of 1449 indoor scenes acquired with a first generation Kinect. For each scene a color view and a depth map are provided. We used the updated ground truth labels from [36] since the original ones have missing areas. The original 894 categories have been clustered into 14 classes as proposed in [35] since this grouping is used for the evaluation of competing approaches. The dataset has been divided in two subsets using the standard train/test separation with 795 and

Algorithm 1 Merge algorithm

$Q_A \leftarrow$ Priority queue, sort by similarity value
for each couple of adjacent segments $A_{i,j} = \{S_i, S_j\}$ **do**
  if $b_{i,j} \geq T_{sim}$ **then**
    $Q_A.push(A_{i,j})$
  end if
end for
while $Q_A \neq \emptyset$ **do**
  $A_{i^*,j^*} \leftarrow Q_A.pop()$
  Compute fitting error on merged segment $S_{i^* \cup j^*}$
  if Equation (8) is satisfied **then**
    Remove all $A_{i,j}$ such that $i = i^* \vee j = j^*$ from Q_A
    for each S_k adjacent to $S_{i^* \cup j^*}$ **do**
      if $b_{i^* \cup j^*,k} \geq T_{sim}$ **then**
        $Q_A.push(A_{i^* \cup j^*,k})$
      end if
    end for
  end if
end while

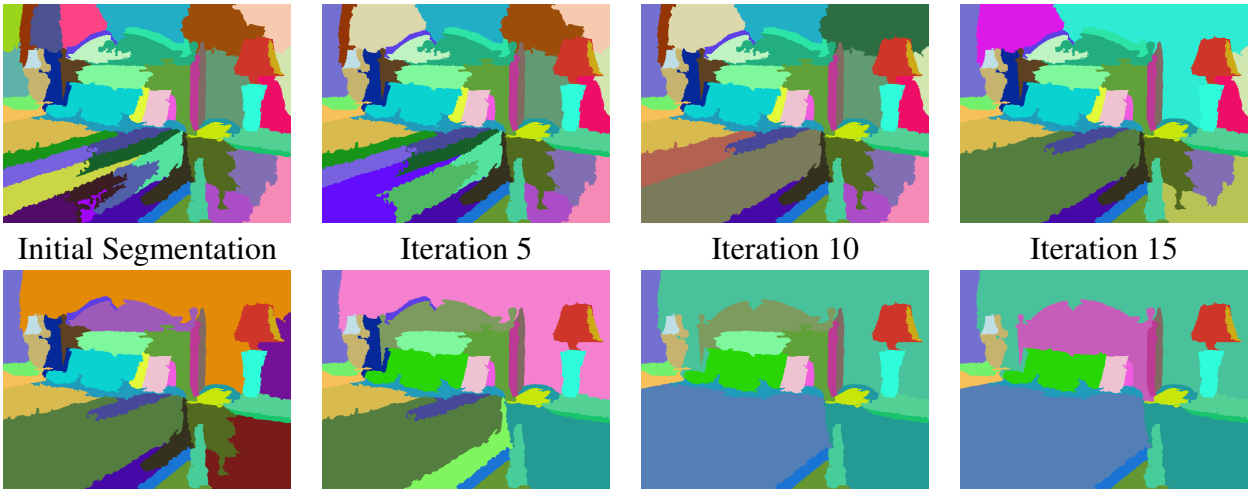

Fig. 5. *Some steps of the merging procedure on the scene of Fig. 6, row 6. The initial over-segmentation, the output after 5, 10, 15, 20, 25, 30 iterations and the final result (iteration 32) are shown.*

654 scenes respectively. For the semantic labeling we provide the results on the test set since this is the approach used by all the competing approaches. For segmentation instead most approaches are evaluated on the complete dataset. To get the results in this case two independent evaluations have been done: in the first experiment we used the standard train/test subdivision as before while in the second the train and test sets have been swapped.

Notice that no expansion of the dataset has been used in the training. Concerning the parameters we used $\sigma = 3$ for the normalized cuts algorithm while for the adjacency computation we used $T_d = 0.2\ m$, $T_c = 10$ and $T_\theta = 4°$ as threshold values. In the CNN optimization, we use quadratic regularization with coefficient 0.001 and stochastic gradient descent for the weights optimization. The learning rate has initial value 0.01 and is updated with an adaptive decay policy reducing it of a factor 0.7 after 10 epochs without improvement. The training of the CNN network on the NYUD2 dataset required around 36 hours on a workstation with an Intel i7-970 CPU at 3.2 Ghz and an NVIDIA Tesla K40 GPU.

The proposed method produces a segmentation with semantic labels and it can be exploited both as a segmentation algorithm and as a semantic classification one. This section is split in two parts evaluating the proposed algorithm for the two considered tasks.

8.1. Evaluation of the segmentation accuracy

The comparison of our approach with state-of-the-art methods on the NYUD2 dataset is presented in Table 1 (the results of some competing approaches have been collected from [13] and [3]). The compared approaches are the works of [13] and [14] based on clustering and region merging, the MRF scene labeling approach of [16], a variation of [37] that exploits also geometry data, the method of [15] exploiting dynamic programming and the multi-layer clustering strategy of [8]. We also compared with our previous works, i.e., the approach of [2] based on normalized cuts, the region merging approach of [3] and the method of [4] that represents the starting point for this work. The method of [3] exploits an iterative merging scheme driven by surface fitting but does not exploit any machine learning clue, so it can be used as a reference to evaluate the impact on the performances due to NURBS surface fitting (i.e., roughly corresponding to the difference between [2] and [3]) and due to the CNN descriptors (i.e., the further improvement from [3] to the proposed work).

The results have been compared with ground truth data using 2 different metrics (see [38] for details). The first is the Variation of Information (VoI) and the second the Rand Index (RI). The mean VoI score of our approach is 1.92. This is the best value among all the considered approaches and the gap is significant. The only method getting close to the proposed one is [4] (i.e., the previous version of this work based on the same segmentation algorithm with a simpler semantic classification stage). The mean score according to the RI metric is 0.91. This value is better than most compared approaches, i.e., [37], [15], [2], [3], [13] and [16], and is the same of the two best competitors, i.e., [14] and [4]. Another advantage is that our method does not rely on the assumption of planar surfaces (NURBS can handle complex shapes), while many competitors (e.g., [13], [14] and [15]) exploit this assumption thus getting accurate results on the NYUD2 dataset (that has many planar surfaces), but reducing their generalization capabilities on scenes with non-planar surfaces.

Some visual examples of segmentations performed with the proposed approach are displayed in Fig. 6. The sequence of merging steps for a couple of sample scenes is shown in the videos available at `http://lttm.dei.unipd.it/paper_data/iet_semantic` . The images show that the algorithm is able to properly segment different challenging scenes. The background and the

Table 1 Average VoI and RI values on the NYUD2 dataset (1449 scenes). The Table shows the data for some state-of-the-art methods from the literature, for our previous works and for the proposed method. Note that lower values are better for VoI while higher ones are better for RI.

Method	*VoI*	*RI*
Hasnat et al. (2014) [13]	2.29	0.90
Hasnat et al. (2016) [14]	2.20	**0.91**
Ren et al. [16]	2.35	0.90
Felzenszwalb et al. [37]	2.32	0.81
Taylor et al. [15]	3.15	0.85
Khan et al. [8]	2.42	0.87
Dal Mutto et al. [2]	3.09	0.84
Pagnutti et al. [3]	2.23	0.88
Minto et al. [4]	1.93	**0.91**
Proposed method	**1.92**	**0.91**

larger surfaces are divided in several segments in the initial over-segmentation but they are properly merged by the iterative algorithm since the CNN descriptors are a very useful clue in order to detect if segments are part of the same object or region. On the other side the algorithm is able to correctly recognize and keep separate most of the scene objects. It is also possible to observe that the objects' edges are precise and there are no small segments close to edges as in some competing methods. However there are a few minor mistakes specially on small objects.

8.2. Evaluation of the classification accuracy

The proposed approach provides also a semantic label for each segment. In order to evaluate the accuracy of this labeling we compared it with some competing approaches on the NYUD2 test set. The compared state-of-the-art approaches are the methods of [23] that uses a multi-scale CNN, of [39] that uses a hierarchy of super-pixels to train a random forest classifier, of [27] that uses deep learning to extract super-pixels features, of [25] exploiting two different CNNs, of [20] using Random Forest and CRFs and finally [26] using a multi-scale deep learning architecture.

Table 2 reports the results: two different metrics have been considered, the per-pixel accuracy, counting the percentage of correctly classified pixels and the average class accuracy, obtained by computing the percentage of correctly classified pixels for each class independently and averaging the values. Notice that the second number is smaller since classes with a low number of samples are typically harder to recognize.

The proposed deep learning architecture achieves a mean pixel accuracy of 64.4% on the test set. By taking the segmentation output of Subsection 8.1 and assigning a single label to each segment as described in Section 7 it is possible to refine the labeling and increase the accuracy to 67.2%. This is a very good result outperforming all the compared approaches except [26]. Notice that [26] achieves very high performances by exploiting a much more complex deep learning architecture. In any case our method is the one that gets closer to it, while even the very recent methods of [39] and [25] have lower performances than ours.

The results are also confirmed by the average class accuracy. The CNN output accuracy is of 51.7%, a remarkable result outperforming all compared approaches except [25] and [26]. By refining it with the segmentation the accuracy increases to 54.4%, outperforming also [25]. Table

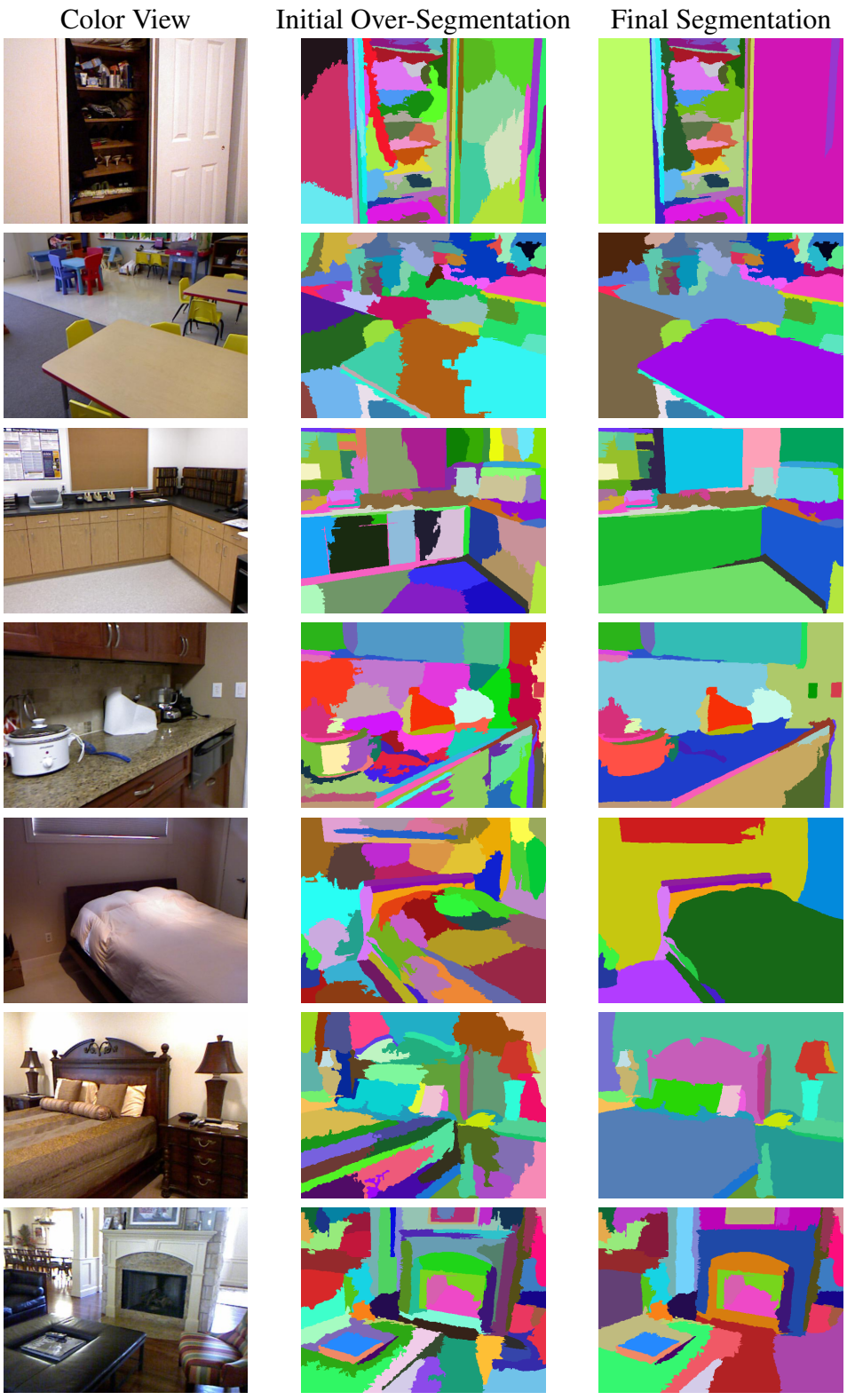

Fig. 6. *Initial over-segmentation and output of the proposed approach for some example scenes. The displayed scenes are the number 72, 330, 450, 846, 1105, 1110 and 1313 of the NYUD2 dataset.*

3 reports also the accuracy for each class, notice how it is very high on several classes and quite low only for a few classes. In particular the accuracy is lower on classes without a well defined geometric structure, like the *Objects* class, that includes many different things inside or the *Picture/wall deco* class, that is associated to a quite flat and not very descriptive geometry. Another issue is that the dataset is not balanced and there are uncommon classes for which a limited amount of training data is available, e.g., the *TV/monitor* class, that accounts for just 1% of the samples and is the worst for our approach. On the other side the *Floor* and *Wall* classes have a very regular structure are detected with an high accuracy. Table 2 also report the improvement obtained by refining the output with the segmentation, notice how it improves for all classes except a small loss on the *Picture/wall deco* and *Object* classes. This is due to the fact that the segmentation improves the boundary accuracy and removes isolated detections typically due to noise (but they can seldom correspond to small objects).

A visual evaluation of the results on some sample scenes is shown in Fig. 7, notice how the classification is accurate even in challenging situations (e.g., closed windows) and how the refinement with segmentation largely improves the edges accuracy. However a few errors are present, e.g., beds exchanged with sofas that have a similar visual appearance.

We also tried different approaches than simply selecting the most common label for the segmentation-based refinement, however the average pooling scheme lead to the same pixel accuracy of 67.2% with just small differences on the accuracy of the single classes, while the max-pooling scheme lead to less satisfactory performances with a 65.9% accuracy.

The implementation of the approach has not been optimized, currently the processing of a scene requires on average less than 2 minutes. Furthermore most computation time is spent on the initial over-segmentation ($87s$) that could be replaced with a simpler superpixel segmentation scheme.

Table 2 Average pixel and class accuracies on the test set of the NYUD2 dataset (654 scenes) for some state-of-the-art methods from the literature and for the proposed method.

Approach	*Pixel Accuracy*	*Class Accuracy*
Couprie et al. [23]	52.4%	36.2%
Hickson et al. [39]	53.0%	47.6%
A. Wang et al. [27]	46.3%	42.2%
J. Wang et al. [25]	54.8%	52.7%
A. Hermans et al. [20]	54.2%	48.0%
D. Eigen et al. [26]	75.4%	66.9%
Proposed method (CNN output)	64.4%	51.7%
Proposed method (with segmentation)	**67.2%**	**54.4%**

9. Conclusions

This work introduced a combined RGBD segmentation and semantic labeling algorithm exploiting deep learning and an iterative merging procedure. Surface fitting information has been used both to control the merging and as an additional clue improving the performances of the Convolutional Neural Network. The joint usage of geometric information and of an estimate of the similarity between segments from the CNN descriptors allowed to properly select the merging operations to be performed. The experimental evaluation showed that our approach obtains state-of-the-art

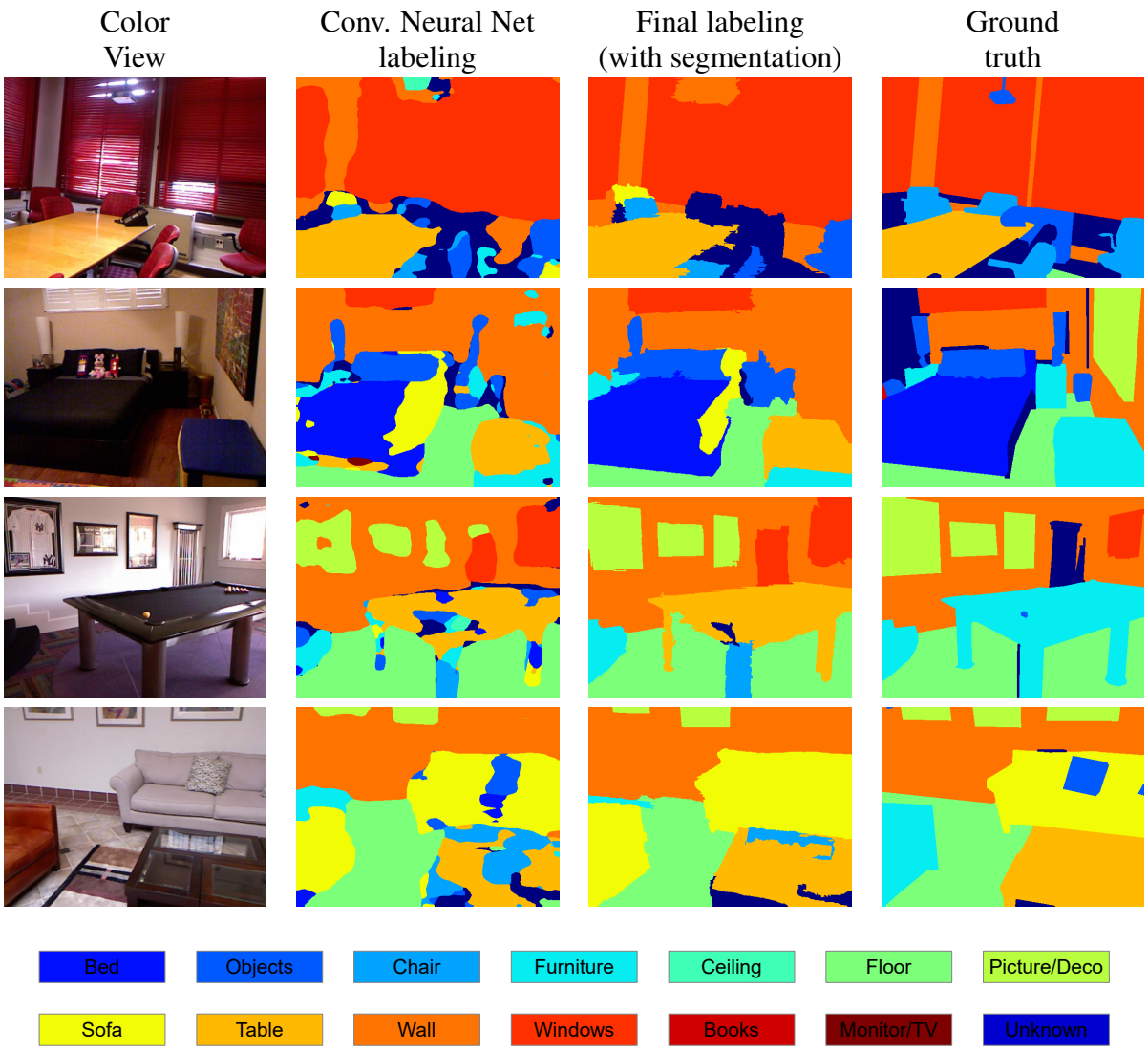

Fig. 7. *Semantic labeling of some sample scenes from the NYUDv2 dataset. The figure shows the color images, the labeling from the Convolutional Neural Network, the refined labeling exploiting segmentation data and the ground truth for scenes 39, 280, 433 and 462*

Table 3 Average accuracy for each of the 13 classes on the test set of the NYUDv2 dataset for the proposed approach (the *unknown* class has not been considered consistently with the evaluation of all the compared approaches)

Class	*Accuracy (CNN)*	*Accuracy (with segmentation)*	*Accuracy Improvement*
Bed	58.0%	64.1%	6,1%
Objects	43.2%	41.8%	-1,4%
Chair	35.4%	38.4%	3,0%
Furniture	64.7%	70.2%	5,5%
Ceiling	62.8%	64.2%	1,4%
Floor	92.2%	93.7%	1,5%
Picture / wall deco	30.5%	26.8%	-3,7%
Sofa	55.8%	66.5%	10,7%
Table	42.0%	46.0%	4,0%
Wall	83.7%	86.3%	2,6%
Window	53.9%	55.8%	1,9%
Books	23.8%	24.0%	0,2%
Monitor / TV	26.2%	29.1%	2,9%
Average	51.7%	54.4%	2,7%

performances both in the segmentation and in the semantic labeling task.

Further research will explore the usage of other deep learning structures, e.g., Fully Convolutional Networks [24]. The direct usage of deep learning techniques for the selection of the merging operations to be performed will also be considered. The initial over-segmentation step is another step that can be improved with faster approaches or multiple segmentations combined together. Finally more advanced strategies for the refinement of the labeling with segmentation information will be considered including the use of deep learning architectures for this task [40].

10. Acknowledgment

We gratefully acknowledge NVIDIA for the Tesla K40 exploited for the training of the Convolutional Neural Network.

11. References

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

[2] 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)**, pp. 505–521, (2012).

[3] G. Pagnutti, P. Zanuttigh. “Joint color and depth segmentation based on region merging and surface fitting”, *Proceedings of International Conference on Computer Vision Theory and Applications*, (2016).

[4] L. Minto, G. Pagnutti, P. Zanuttigh. "Scene segmentation driven by deep learning and surface fitting", *Proceedings of ECCV Geometry meets deep learning workshop*, (2016).

[5] S. Gupta, R. Girshick, P. Arbeláez, J. Malik. "Learning rich features from RGB-D images for object detection and segmentation", *Proceedings of European Conference on Computer Vision (ECCV)*, pp. 345–360, (2014).

[6] P. Zanuttigh, G. Marin, C. Dal Mutto, F. Dominio, L. Minto, G. M. Cortelazzo. *Time-of-Flight and Structured Light Depth Cameras*, (Springer, 2016).

[7] C. Dal Mutto, P. Zanuttigh, G. Cortelazzo. "Scene segmentation assisted by stereo vision", *Proceedings of 3DIMPVT 2011*, (Hangzhou, China2011).

[8] M. R. Khan, A. B. M. M. Rahman, G. M. A. Rahaman, M. A. Hasnat. "Unsupervised rgb-d image segmentation by multi-layer clustering", *Proceedings of International Conference on Informatics, Electronics and Vision*, pp. 719–724, (2016).

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

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

[11] 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*, **112(2)**, pp. 133–149, (2015).

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

[13] M. A. Hasnat, O. Alata, A. Trémeau. "Unsupervised RGB-D image segmentation using joint clustering and region merging", *Proceedings of British Machine Vision Conference (BMVC)*, (2014).

[14] M. A. Hasnat, O. Alata, A. Trémeau. "Joint color-spatial-directional clustering and region merging (JCSD-RM) for unsupervised RGB-D image segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2016).

[15] C. J. Taylor, A. Cowley. "Parsing indoor scenes using RGB-D imagery", *Robotics: Science and Systems*, volume 8, pp. 401–408, (2013).

[16] X. Ren, L. Bo, D. Fox. "Rgb-(d) scene labeling: Features and algorithms", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2012).

[17] Z. Deng, S. Todorovic, L. Jan Latecki. "Semantic segmentation of RGBD images with mutex constraints", *Proceedings of International Conference on Computer Vision (ICCV)*, pp. 1733–1741, (2015).

[18] D. Lin, S. Fidler, R. Urtasun. "Holistic scene understanding for 3d object detection with rgbd cameras", *Proceedings of International Conference on Computer Vision (ICCV)*, pp. 1417–1424, (2013).

[19] S. H. Khan, M. Bennamoun, F. Sohel, R. Togneri. "Geometry driven semantic labeling of indoor scenes", *Proceedings of European Conference on Computer Vision (ECCV)*, pp. 679–694, (Springer, 2014).

[20] A. Hermans, G. Floros, B. Leibe. "Dense 3d semantic mapping of indoor scenes from rgb-d images", *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 2631–2638, (IEEE, 2014).

[21] D. Banica, C. Sminchisescu. "Second-order constrained parametric proposals and sequential search-based structured prediction for semantic segmentation in rgb-d images", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3517–3526, (2015).

[22] N. Höft, H. Schulz, S. Behnke. "Fast semantic segmentation of RGB-D scenes with gpu-accelerated deep neural networks", *Joint German/Austrian Conference on Artificial Intelligence*, pp. 80–85, (2014).

[23] C. Couprie, C. Farabet, L. Najman, Y. Lecun. "Convolutional nets and watershed cuts for real-time semantic labeling of RGBD videos.", *Journal of Machine Learning Research*, **15(1)**, pp. 3489–3511, (2014).

[24] E. Shelhamer, J. Long, T. Darrell. "Fully convolutional networks for semantic segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2016).

[25] J. Wang, Z. Wang, D. Tao, S. See, G. Wang. "Learning common and specific features for rgb-d semantic segmentation with deconvolutional networks", *Proceedings of European Conference on Computer Vision (ECCV)*, pp. 664–679, (2016).

[26] D. Eigen, R. Fergus. "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture", *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2650–2658, (2015).

[27] A. Wang, J. Lu, G. Wang, J. Cai, T.-J. Cham. "Multi-modal unsupervised feature learning for rgb-d scene labeling", *Proceedings of European Conference on Computer Vision (ECCV)*, pp. 453–467, (2014).

[28] J. R. Uijlings, K. E. Van De Sande, T. Gevers, A. W. Smeulders. "Selective search for object recognition", *International Journal of Computer Vision*, **104(2)**, pp. 154–171, (2013).

[29] J. Carreira, C. Sminchisescu. "Cpmc: Automatic object segmentation using constrained parametric min-cuts", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **34(7)**, pp. 1312–1328, (2012).

[30] 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", *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2684–2689, (2012).

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

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

[33] M. do Carmo. *Differential Geometry of Curves and Surfaces*, (Prentice-Hall, 1976).

[34] C. Farabet, C. Couprie, L. Najman, Y. LeCun. "Learning hierarchical features for scene labeling", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35(8)**, pp. 1915–1929, (2013).

[35] C. Couprie, C. Farabet, L. Najman, Y. LeCun. "Indoor semantic segmentation using depth information", *International Conference on Learning Representations*, (2013).

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

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

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

[39] S. Hickson, I. Essa, H. Christensen. "Semantic instance labeling leveraging hierarchical segmentation", *Winter Conference on Applications of Computer Vision*, pp. 1068–1075, (2015).

[40] H. Caesar, J. Uijlings, V. Ferrari. "Region-based semantic segmentation with end-to-end training", *Proceedings of European Conference on Computer Vision (ECCV)*, pp. 381–397, (Springer, 2016).