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: This paper proposes an approach for segmentation and semantic labeling of RGBD data based on the joint usage of geometrical clues and deep learning techniques. An initial over-segmentation is performed using spectral clustering and a set of NURBS surfaces is then fitted on the extracted segments. The input data are then fed to a Convolutional Neural Network (CNN) together with surface fitting parameters. The network is made of nine convolutional stages followed by a softmax classifier and produces a per-pixel descriptor vector for each sample. An iterative merging procedure is then used to recombine the segments into the regions corresponding to the various objects and surfaces. The couples of adjacent segments with higher similarity according to the CNN features are considered for merging and the NURBS surface fitting accuracy is used in order to understand if the selected couples correspond to a single surface. By combining the obtained segmentation with the descriptors from the CNN a set of labeled segments is obtained. The comparison with state-of-the-art methods shows how the proposed method provides an accurate and reliable scene 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 one 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 one 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., normalised cuts spectral clustering [1], are an effective approach for segmentation that can be easily extended to the joint segmentation of image and depth data [2]. However, 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.

The problem can be solved by starting from an over-segmentation performed with spectral clustering and exploiting an iterative region merging scheme in order to obtain the final segmentation. This work follows this rationale and uses together two different clues in order to decide which segments must be merged. The first one is a similarity index between segments computed by comparing the descriptors produced by a Convolutional Neural Network (CNN). The second one is obtained, for a given pair 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].

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 (the conference paper dealt only with the segmentation problem).

More in detail:

- A deeper Convolutional Neural Network architecture has been employed.
- Surface curvatures and fitting error are also fed to the CNN (up to our knowledge this is the first time this kind of data is used in a deep learning framework).
- Orientation data have been replaced by HHA descriptors (disparity, height and orientation angle).
- The experimental evaluation now addresses also the semantic labeling.

2. Related Works

Segmentation of RGBD data has been the subject of many research works (a recent review is contained in [5]). Clustering techniques are commonly used for image segmentation and they can easily be extended to joint depth and color segmentation by modifying the feature vectors [6, 2]. The method of [7] performs multiple clusterings with K-means and combines them together.

Region splitting and growing approaches have also been considered. The approach of [8] starts from an over-segmentation of the scene and combine together the segments in regions corresponding to the planar surfaces using an approach based on Rao-Blackwellized Monte Carlo Markov Chain. Region splitting has been used in [9] 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. The work of [3] uses the same criteria in a bottom-up approach starting from an over-segmentation of the scene. Gupta et al. [10] use an hierarchical segmentation based on the output of a contour detector. Another combined approach for segmentation and object recognition is [11], where an initial over-segmentation is obtained exploiting the watershed algorithm followed by a hierarchical scheme. Hasnat et al. [12, 13] use a joint clustering method on the color, 3D position and normal information followed by a statistical planar region merging scheme. Finally dynamic programming has been used in [14] to extract the planar surfaces in indoor scenes.

Machine learning techniques have been used specially for the task of semantic segmentation, i.e., joint segmentation and labeling of the segments. Ren et al. [15] exploit a Markov Random Fields superpixel segmentation combined with a tree-structured approach. Conditional Random Fields (CRF) have also been exploited in several works [16, 17]. The approach of [16] combines the CRF with mutex constraints based on the geometric structure of the scene while the approach of [17] combines 2D segmentation, 3D geometry data and contextual information. The work of [18] is instead based on a proposal process that generates spatial layout hypotheses followed by a sequential inference algorithm.

Recently, deep learning techniques and in particular Convolutional Neural Networks (CNN) have been exploited for the semantic segmentation task [19, 20, 21]. One of the first solutions based on deep learning is [20], that uses a multiscale CNN. The method of [21] is able to achieve a very high accuracy by exploiting Fully Convolutional Networks. Another approach based on deep learning is [22], that exploits a CNN applied on features extracted from the geometry description.

Wang et al. [23] 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 [24] jointly solves the semantic labeling together with depth and normal estimation using a multi-scale CNN. Finally the method of [25] uses deep learning to extract superpixel features that are then classified with SVMs.

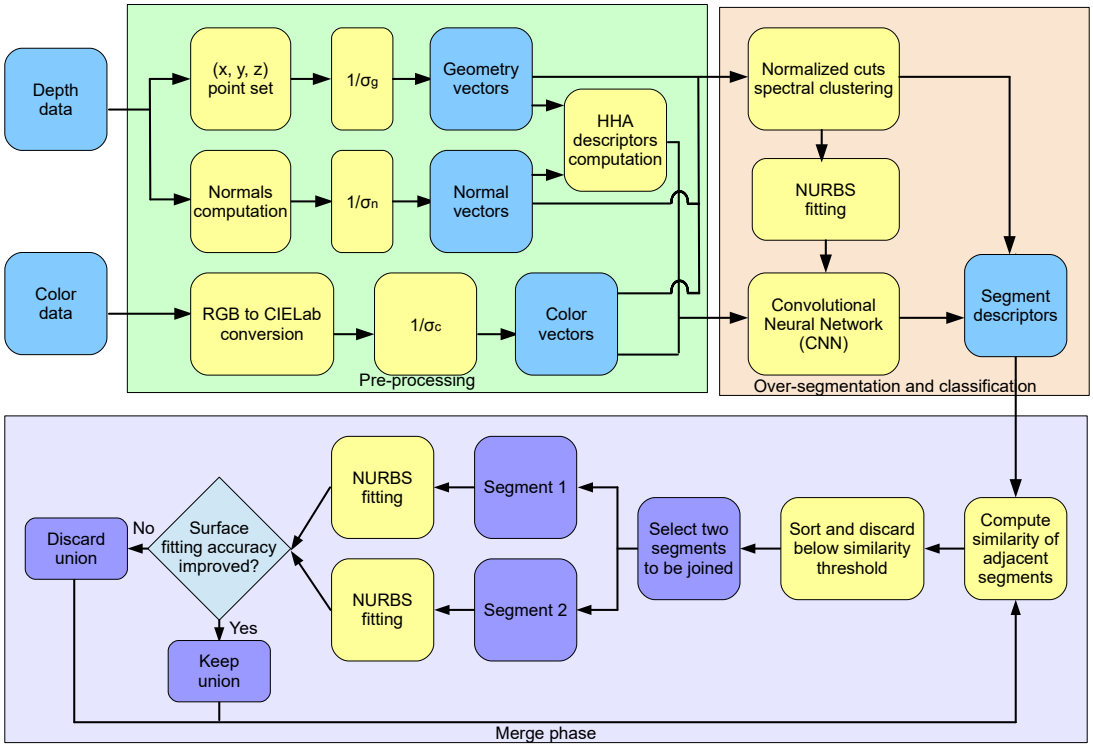

Fig. 1. *Overview of the proposed approach*

3. General Overview

The proposed algorithm is organized in three main steps as depicted in Fig. 1. Color and depth data are converted into a set of 9D vectors containing the 3D position, the orientation information and the color coordinates for each sample. Then the algorithm performs an over-segmentation of the scene based on the joint usage of the three sources of information using a spectral clustering framework derived from [2, 3] (see Section 4). After performing the over-segmentation, a parametric 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 CNN classifier (Section 6) is trained for a semantic labeling task and computes a vector of descriptors for each pixel representing the probabilities of the various classes at the pixel location. The descriptors are then aggregated inside each segment in order to produce a single descriptor for each segmented region. The third step is an iterative region merging procedure (Section 7). This stage starts by analyzing the segmentation and computing an adjacency map, where two segments are considered adjacent if they touch each other and their properties are similar on the common

contour. The couples of adjacent segments are then 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. The sorted list of couples is then processed in order of similarity. For each couple a parametric NURBS surface is fitted over each of the two segments and the same model is fitted also over the merged region obtained by fusing the two segments. The surface fitting error of the two segments and of the merged region are compared. 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 the merging is discarded. The procedure is repeated in a tree structure until no more merging operations are possible. 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 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 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 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 [2, 3].

The clustering algorithm needs geometry, color and orientation to be into consistent representations. For this reason, the geometry, orientation and color components are normalized by the average of the respective standard deviations, obtaining the normalized vectors $[\bar{x}(p_i), \bar{y}(p_i), \bar{z}(p_i)]$, $[\bar{n}_x(p_i), \bar{n}_y(p_i), \bar{n}_z(p_i)]$ and $[\bar{L}(p_i), \bar{a}(p_i), \bar{b}(p_i)]$. In this way a 9D representation is built from the above normalized vectors:

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

Normalized cuts spectral clustering [1] optimized with the Nyström method [26] 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 to obtain the final solution by the method of Section 7.

5. Surface Fitting on the Segmented Data

The following step is the approximation of each segment with a Non-Uniform Rational B-Spline (NURBS) surface [27]. This is used twice in the proposed method: firstly, in order to produce an additional set of input clues 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).

A parametric NURBS surface is defined as

$$\mathbf{S}(u, v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j} \mathbf{P}_{i,j}}{\sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j}}, \tag{2}$$

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 pixels in the considered segment. This is necessary to prevent the fitting accuracy to be biased in favor of smaller segments [3]. Finally, by using Equation (2) evaluated at the depth lattice points and equated to the points to fit (see [3] for details), we obtain an over-determined system of linear equations and solve it in the least-squares sense thus obtaining the surface control points. Notice that, by using this approach, we are able to provide an appropriate geometric model also for complex shapes, unlike many competing approaches [14, 8] that rely on the assumption that most surfaces in the scene are planar.

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). The second one, related to the geometric shape of the fitting surface, is given by the two principal curvatures (i.e., the maximum and minimum local curvature values, see [28]) at each pixel location. This is used only 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 (i.e., to drive the merging operation described in Section 7). For this task we exploit a Convolutional Neural Network (CNN) trained for the semantic segmentation task that produces a pixel-wise higher-level description of the input images.

The idea is to use the output of a CNN to provide each pixel with a descriptor vector and to use this information to compute a similarity measure between adjacent segments, besides using it also for the semantic labeling. Notice that the merging strategy described in Section 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.

The CNN takes in input various clues:

- 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 [22] 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.

These representations are combined into 9D vectors representing each point of the scene as

$$\mathbf{p}_i^{cn} = [R(p_i), G(p_i), B(p_i), h_1(p_i), h_2(p_i), a(p_i), f(p_i), c_1(p_i), c_2(p_i)]. \quad (3)$$

Then, a 9-channels input image is produced for each scene in the dataset by arranging the vectors over the image pixels lattice.

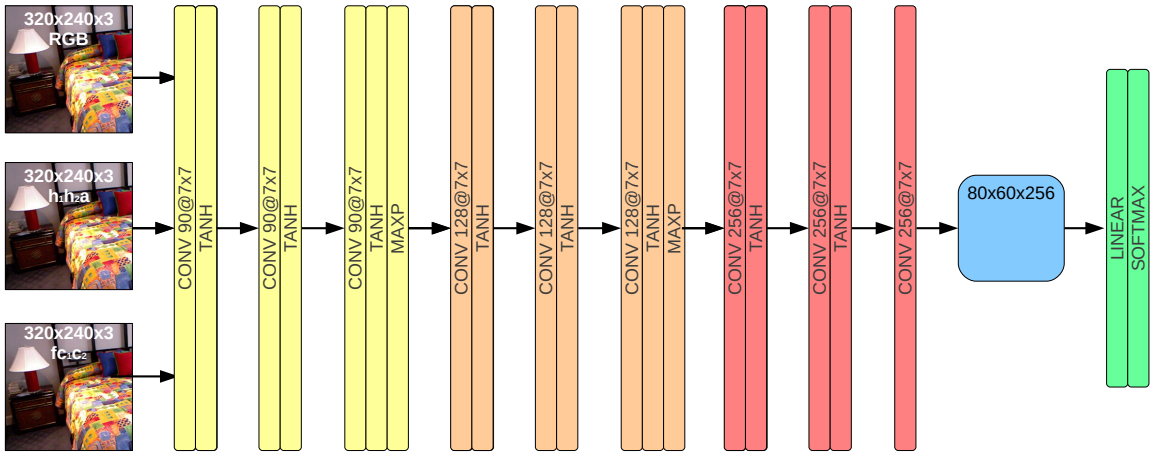

Fig. 2. *Architecture of the Convolutional Neural Network*

An overview of the employed network structure is shown in Fig. 2. The network has been constructed starting from the approach of [29, 30, 4] yet no multi-scale input representation has been used. A local representation of the input is extracted by applying a sequence of convolutional layers. Specifically, the 9D input vectors are feed-forwarded through nine convolutional layers arranged in three blocks each with three layers (see Fig. 2). Every block contains 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 used on top of the last convolutional layer.

Input images are fed to the network at the resolution of 320×240, while the convolutional layers have 90 filters in the first block, 128 in the second and 256 in the last one. All filters are 7×7 pixels wide, while the softmax classifier has a weight matrix of size 256×14 and no bias. The filters in the first convolutional layer are divided into 9 groups and each group is connected to one of the 9 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 14 labels to each pixel in the input image. Specifically, we clustered the 894 categories from the ground truth provided by [31] into 14 classes as in [30]. To this aim, a multi-class cross-entropy loss function is minimized throughout the training process.

Besides the final predicted labels, the output of the softmax classifier is also exploited in order to compute the descriptors used for the similarity measure between any two segments. The output of the softmax, a 3D array of size $80 \times 60 \times 14$, is linearly interpolated to the size of the input image so that a descriptor vector $\mathbf{c}_i = [c_i^1, \ldots, c_i^{14}]$ is associated to each 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 $\mathbf{s}_i = [s_i^1, \ldots, s_i^{14}]$ 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.,

$$\mathbf{s}_i = \frac{\sum_{j \in S_i} \mathbf{c}_j}{|S_i|}. \tag{4}$$

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

$$b_{i,j} = \sum_{t=1,..,14} \sqrt{s_i^t s_j^t}. \tag{5}$$

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 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 Section 7 typically correspond to low similarity values.

a) b)

Fig. 3. *Computation of $b_{i,j}$ on a sample scene: 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)*

7. Region Merging Procedure

The next step 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 as summarized in the bottom part of Fig. 1 and in Algorithm 1.

The algorithm starts by selecting the couples of close segments that are candidate to be joined. It builds an adjacency matrix storing for each couple of segments whether they are *adjacent* (i.e., candidate to be joined) or not. In order to mark two segments as *adjacent* they must satisfy a set of conditions (see [3] for more details):

1. They must be connected on the lattice defined by the depth map.
2. The depth values on the shared boundary must be consistent. More in detail, for each point P_i in the shared boundary C_C we compute the difference ΔZ_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. We used the same approach of depth data applied to the color difference in the CIELab space ΔC_i with threshold T_c:

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

4. Finally the same approach is used also for orientation data by comparing the angle between the two normal vectors $\Delta\theta_i$ to a threshold T_θ:

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

If all the conditions are satisfied the two segments are marked as adjacent (for the experimental results we used $T_d = 0.2\ m$, $T_c = 10$ and $T_\theta = 4°$).

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 Section 6.

The couples of adjacent segments are sorted according to the similarity between the two segments estimated during the machine learning stage, i.e., on the basis of the $b_{i,j}$ values. 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 (in the results we used $T_{sim} = 0.77$). 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 $S_{i^* \cup j^*}$ their union. A NURBS surface is fitted on each of the two regions i^* and j^* (see Section 5). The fitting error, i.e., the Mean Squared Error (MSE) between the actual and the fitted surface, is then 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 Equation (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 Equation (4) and the similarity score is computed for all the newly created couples involving the segment $S_{i^* \cup j^*}$ created by the merging operation. 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}). The algorithm then selects the next couple in the sorted list and the procedure is repeated until no more segments can be considered for merging.

After getting the final segmentation a semantic label is also associated to 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.

The procedure is summarized in Algorithm 1 and some examples of its progress are shown in Fig. 4 and in the videos in the additional material.

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 (5)
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 $S_{i \cup j}$
  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 $S_{i \cup j}$
    Compute $A_{i \cup j,k}$ 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
Compute the semantic labeling for all the segments

8. Experimental Results

We tested the proposed approach on the NYU-Depth V2 dataset (NYUDv2) [11]. The NYUDv2 dataset contains 1449 depth and color frames from a variety of indoor scenes acquired with a first generation Kinect. We used the updated ground truth labels from [31] since the original ones have missing areas. The dataset has been divided in two parts using the standard train/test subdivision with 795 and 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 we performed two independent tests: in the first test we used the standard train/test subdivision as before. In the second test we swapped the train and test sets and performed the same procedure.

Notice that no expansion of the dataset has been used in the training. Concerning the CNN

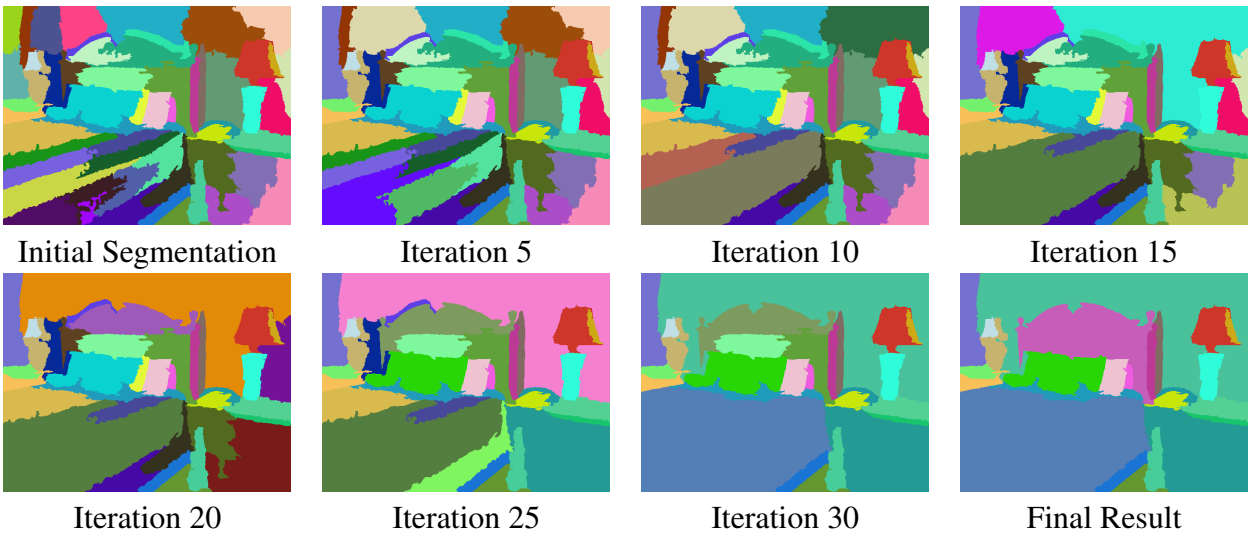

Fig. 4. *Example of the merging procedure on the scene of Fig. 5, row 6. The images show the initial over-segmentation, the merging output after 5, 10, 15, 20, 25, 30 iterations and the final result (iteration 32). (Best viewed in color)*

optimization, we used quadratic regularization with coefficient 0.001 and the network weights have been updated using stochastic gradient descent, with initial learning rate equal to 0.01 and an adaptive decay policy reducing it of a factor 0.7 after 10 epochs without improvement.

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

Table 1 shows the comparison between our approach and some state-of-the-art methods on the NYUDv2 dataset (for the other approaches we collected the results from [12] and [3]). The compared approaches are the clustering and region merging method of [12], the MRF scene labeling scheme of [15], a modified version of [32] that accounts also for geometry information, the dynamic programming scheme of [14] and the multi-layer clustering strategy of [7]. We also compared with our previous works, i.e., the clustering-based approach of [2], the region merging scheme of [3] and the method of [4] that represent the starting point for this work. The comparison with [3] can be a hint of the improvement provided by the use of the CNN descriptor since it is also based on over-segmentation and NURBS fitting but it does not have a machine learning stage.

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) [33], 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 1.92. According to this metric our approach is the best among the considered ones with a significant gap with respect of all the competing approaches and a small improvement with respect to our previous conference work [4] (that exploits the same segmentation scheme with a less accurate semantic classification). If the RI metric is employed the average score is 0.91. This value is better than the one of [32], [14], [2], [3], [12] and [15], while it is exactly the same of the best competing approaches, i.e., [13] and [4]. Furthermore our approach does not assume the presence of planar surfaces thanks to the NURBS surface fitting scheme, while some competing

ones (e.g., [12], [13] and [14]) rely on this clue obtaining good results on the NYUDv2 dataset where most of the surfaces are planar, but reducing their generalization capabilities on scenes with non-planar surfaces.

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 methods from the literature

Approach	*VoI*	*RI*
Hasnat et al. (2014) [12]	2.29	0.90
Hasnat et al. (2016) [13]	2.20	**0.91**
Ren et al. [15]	2.35	0.90
Felzenszwalb et al. [32]	2.32	0.81
Taylor et al. [14]	3.15	0.85
Khan et al. [7]	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**

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 in the additional material. The images show how the approach is able to efficiently deal with challenging scenes of different types. The initial over-segmentation divides the background and the larger structures in several pieces but they are properly recombined by the proposed approach thanks to the CNN descriptors that allow to recognize which segments belong to the same structure. At the same time most of the objects in the scene are correctly recognized and kept separated. Furthermore the contours of the objects are well defined and there are not noisy small segments in proximity of edges as in other approaches. However a few inaccuracies are present 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 state-of-the-art approaches on the test set of the NYUDv2 dataset. The compared approaches are the method of [20] that uses a multi-scale CNN, the method of [34] that uses a hierarchy of super pixels to train a random forest classifier, the method of [25] that uses deep learning to extract superpixels features and the method of [23] exploiting two different CNNs.

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 is able to obtain an average pixel accuracy of 64.4% on the testset. 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 an impressive result outperforming all the compared approaches including the very recent state-of-the-art methods of [34] and [23].

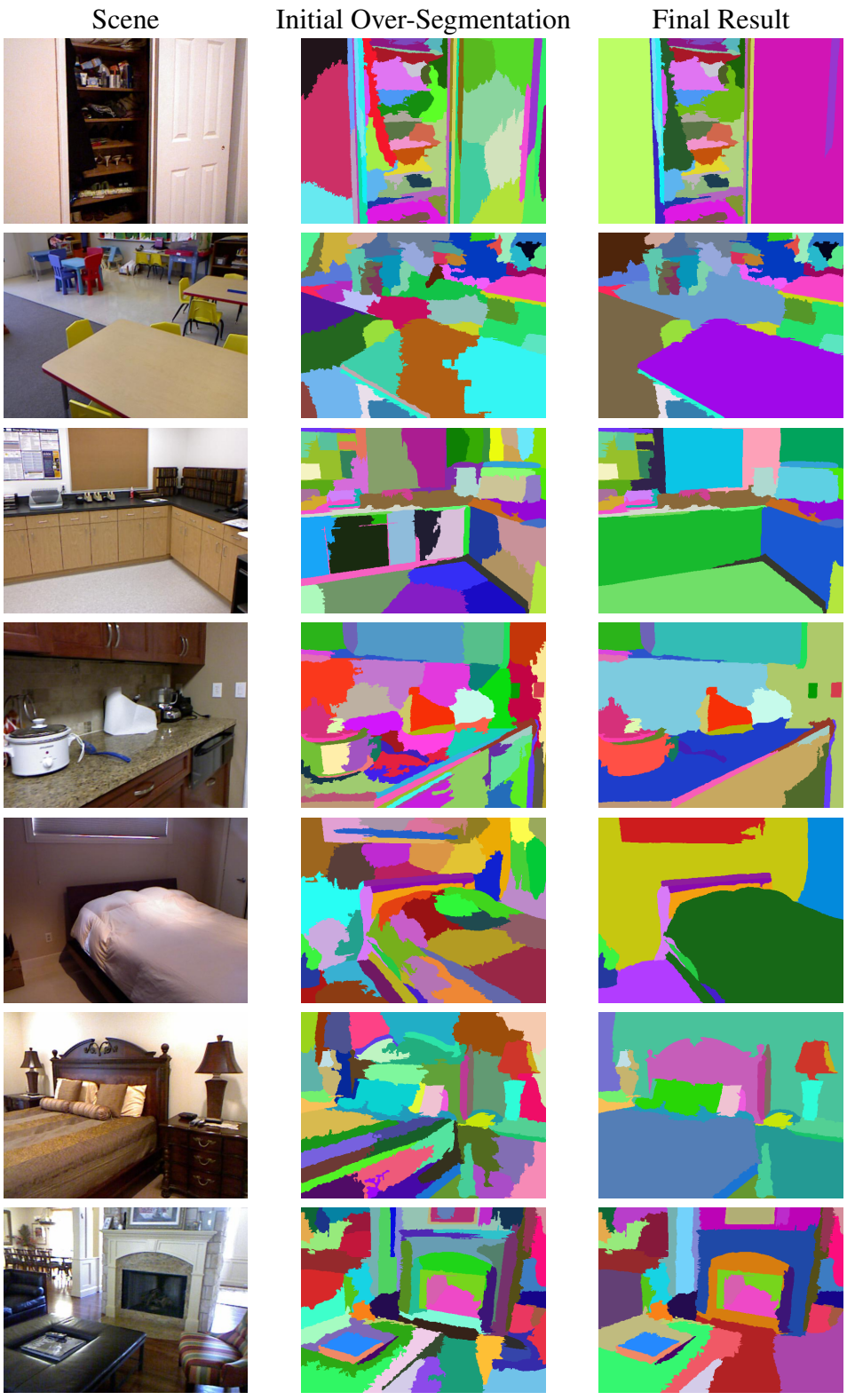

Fig. 5. *Segmentation of some sample scenes from the NYUDv2 dataset. The figure shows the color images, the initial over-segmentation and the final result for scenes 72, 330, 450, 846, 1105, 1110 and 1313*

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 [23]. By refining it with the segmentation the accuracy increases to 54.4%, outperforming all the compared approaches including [23]. Table 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 (typically uncommon ones for which a limited amount of training data is available).

A visual evaluation of the results on some sample scenes is shown in Fig. 6, 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.

The current implementation has not been optimized, however the segmentation of an image with the corresponding depth map takes less than two 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 values of the pixel and class accuracies on the 654 scenes in the test set of the NYUDv2 dataset for the proposed approach and for some state-of-the-art approaches from the literature

Approach	*Pixel Accuracy*	*Class Accuracy*
Couprie et al [12]	52.4%	36.2%
Hickson et al [34]	**53.0%**	47.6%
A. Wang et al [25]	**46.3%**	42.2%
J. Wang et al [23]	**54.8%**	52.7%
Proposed method (CNN output)	64.4%	51.7%
Proposed method (with segmentation)	67.2%	54.4%

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)*
Bed	58.0%	64.1%
Objects	43.2%	41.8%
Chair	35.4%	38.4%
Furniture	64.7%	70.2%
Ceiling	62.8%	64.2%
Floor	92.2%	93.7%
Picture / wall deco	30.5%	26.8%
Sofa	55.8%	66.5%
Table	42.0%	46.0%
Wall	83.7%	86.3%
Window	53.9%	55.8%
Books	23.8%	24.0%
Monitor / TV	26.2%	29.1%
Average	51.7%	54.4%

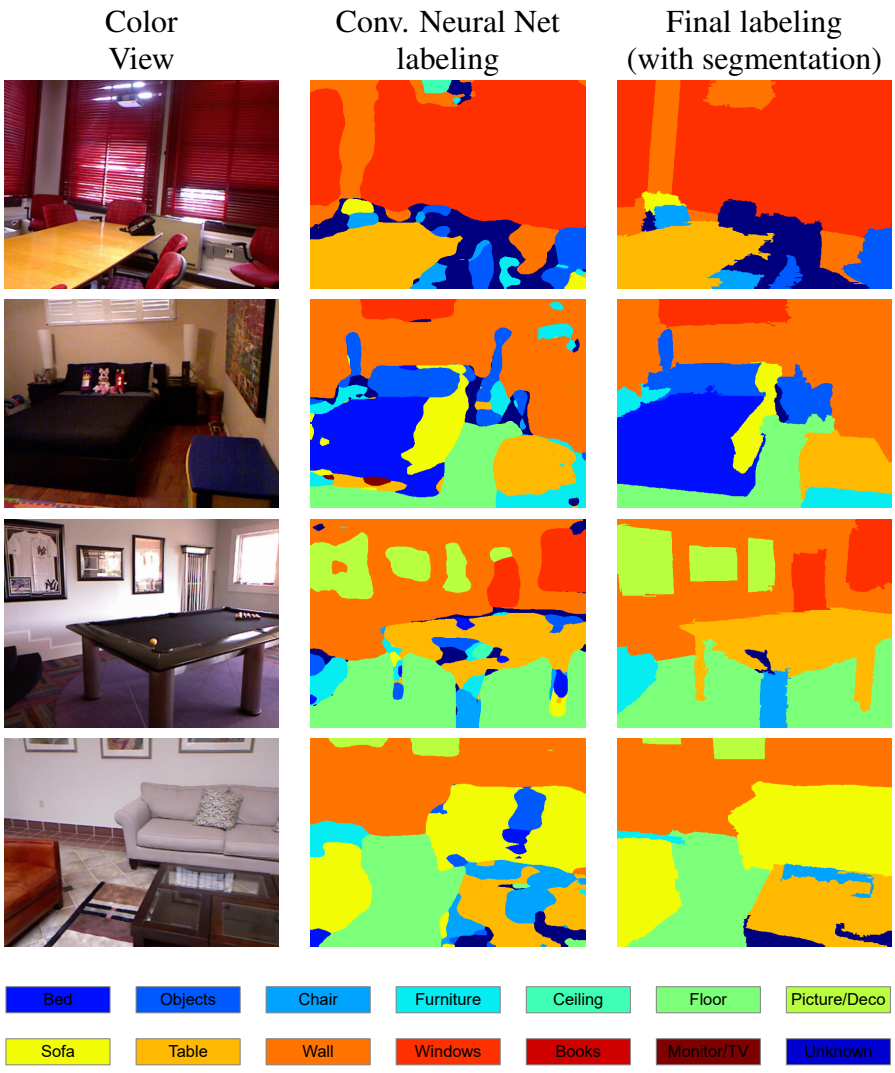

Fig. 6. *Semantic labeling of some sample scenes from the NYUDv2 dataset. The figure shows the color images, the labeling from the Convolutional Neural Network and the refined labeling exploiting segmentation data for scenes 39, 280, 433 and 462*

9. Conclusions and Future Work

In this paper we proposed a joint RGBD segmentation and semantic labeling scheme exploiting deep learning and an iterative merging procedure. Surface fitting information has been used both to control the merging and as an additional input channel to improve the performances of the Convolutional Neural Network. The joint usage of geometrical clues and an estimation of the segments similarity from the CNN descriptors allowed to properly select the merging operations to be performed. As shown by experimental results, our method achieves state-of-the-art performances both for the segmentation and for the semantic labeling task.

Further research will explore the usage of different deep learning architectures, e.g., Fully Convolutional Networks [21]. The direct usage of deep learning techniques for the selection of the merging operations to be performed will also be considered.

10. Acknowledgment

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

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

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

[7] 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).

[8] 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).

[9] 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).

[10] 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).

[11] 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).

[12] 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).

[13] 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).

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

[15] 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).

[16] 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).

[17] 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).

[18] 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).

[19] 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).

[20] 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).

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

[22] 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).

[23] 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).

[24] 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).

[25] 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).

[26] 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).

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

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

[29] 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).

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

[31] 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).

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

[33] 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).

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