Segmentation of Color and Depth Data Based on Surface Fitting

Giampaolo Pagnutti

Ph.D. School on Information Engineering
University of Padova

Advisor: Prof. Pietro Zanuttigh

January 30, 2017

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Sede Amministrativa: Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Scuola di Dottorato di Ricerca in: INGEGNERIA DELL'INFORMAZIONE
Indirizzo: SCIENZA E TECNOLOGIA DELL'INFORMAZIONE
Ciclo: XXVIII

Titolo della tesi: SEGMENTATION OF COLOR AND DEPTH DATA BASED ON SURFACE FITTING

Direttore della Scuola: Prof. Matteo Bertocco
Coordinatore d'indirizzo: Prof. Carlo Ferrari
Supervisore: Prof. Pietro Zanuttigh

Dottorando: Giampaolo Pagnutti

Abstract

This thesis presents novel iterative schemes for the segmentation of scenes acquired by RGB-D sensors. Both the problems of objects segmentation and of semantic segmentation (labeling) are considered.

The first building block of the proposed methods is the Normalized Cuts algorithm, based on graph theory and spectral clustering techniques, that provides a segmentation exploiting both geometry and color information. A limitation is the fact that the number of segments (equivalently, the number of objects in the scene) must either be decided in advance, or requires an arbitrary threshold on the normalized cut measure to be controlled. In addition, this method tends to provide segments of similar size, while in many real world scenes the dimensions of the objects and structures are widely variable. To overcome these drawbacks, we present iterative schemes based on the approximation with parametric NURBS surfaces (Non-Uniform Rational B-Splines). The key idea is to consider the results of the surface fitting as an estimation of how good the current segmentation is. This makes it possible to build region splitting and region merging procedures, in which the fitting results are compared at each step against the previous ones, and the iterations are moved forward based on whether they turn out to be improved or not, until an optimal final solution is reached. The rationale is that, if a segment properly corresponds to an actual object in the scene, the fitting result is expected to be good, while segments that need to be subdivided or merged with other ones are expected to give a larger error. A discussion of several possible metrics to evaluate the quality of the surface fitting is presented. In all the presented schemes, the employment of NURBS surfaces approximation is a novel contribution.

Subsequently, it is described how the proposed iterative schemes can be coupled with a Deep Learning classification step performed with CNNs (Convolutional Neural Networks), by introducing a measure of similarity between the elements of an initial over-segmentation. This information is used together with the surface fitting results to control the steps of a revised iterative region merging procedure. In addition, some information (fitting error, surface curvatures) resulting from the NURBS fitting on the initial over-segmentation is fed into the Convolutional Neural

Networks themselves. To the best of our knowledge, this is the first work where this kind of information is used within a Deep Learning framework. Finally, the objects segmentation resulting from the region merging procedure is exploited to effectively improve the initial classification.

An extensive evaluation of the proposed methods is performed, with quantitative comparison against several state-of-the-art approaches on a standard dataset. The experimental results show that the proposed schemes provide equivalent or better results with respect to the competing approaches on most of the considered scenes, both for the task of objects segmentation and for the task of semantic labeling. In particular, the optimal number of segments is automatically provided by the iterative procedures, while it must be arbitrarily set in advance on several other segmentation algorithms. Moreover, no assumption is done on the objects shape, while some competing methods are optimized for planar surfaces. This is provided by the usage of NURBS surfaces as geometric model, since they can represent both simple entities as planes, spheres, cylinders, and complex free-form shapes.

Sommario

In questa tesi vengono presentati schemi iterativi per la segmentazione di scene acquisite da sensori di colore e profondità. Sia il problema della segmentazione in diversi oggetti che il problema della classificazione semantica vengono affrontati.

Un primo componente dei metodi proposti è l'algoritmo Normalized Cuts, basato su teoria dei grafi e analisi spettrale, che fornisce una segmentazione basata sia sulle informazioni di colore che di geometria. Una limitazione di questo metodo è il fatto che il numero delle regioni (equivalentemente, il numero degli oggetti nella scena) deve essere deciso a priori, oppure richiede l'impostazione di una soglia arbitraria sulla metrica normalized cut per essere controllato. Inoltre, il metodo tende a restituire segmenti di dimensioni similari, mentre le scene reali spesso contengono oggetti e strutture di grandezza molto variabile. Per superare questi limiti, vengono proposti schemi iterativi basati sull'approssimazione mediante superfici parametriche NURBS (Non-Uniform Rational B-Splines). L'idea principale consiste nel considerare il risultato dell'approssimazione come una stima di quanto sia buona la segmentazione corrente. Questo rende possibile costruire procedure di tipo region splitting e region merging in cui i risultati dell'approssimazione sono confrontati ad ogni passo con i precedenti, e l'iterazione viene proseguita in base al fatto che essi risultino migliorati oppure no, fino ad ottenere un risultato ottimale. L'assunzione di fondo è che se un segmento corrisponde ad un oggetto della scena ci si aspetta che l'approssimazione mediante superfici risulti buona, mentre segmenti che devono essere ulteriormente suddivisi o uniti ad altri debbano corrispondere ad un errore maggiore. Per valutare la bontà dell'approssimazione vengono discusse diverse possibili metriche. In tutti gli schemi presentati, l'impiego dell'approssimazione mediante superfici NURBS è in particolare un contributo nuovo.

In seguito, viene descritto come per gli schemi iterativi proposti possano essere proficuamente utilizzate anche le informazioni di classificazione ottenute tramite l'impiego di reti neurali convoluzionali (CNN). Infatti, in base alla classificazione viene introdotta una nozione di similarità tra gli elementi di una sovrasegmentazione iniziale, e questa informazione viene utilizzata assieme al risultato dell'ap-

prossimazione mediante superfici ottenendo una variante della procedura iterativa di tipo region merging precedentemente sviluppata. Inoltre, alcuni dati risultanti dall'approssimazione (errore, curvature delle superfici) vengono forniti in ingresso alle stesse reti neurali convoluzionali; in base alla nostra conoscenza, questo è il primo lavoro in cui dati di questo tipo vengono utilizzati in un'architettura di tipo Deep Learning. Infine, la segmentazione in oggetti ottenuta dalla procedura iterativa viene sfruttata per raffinare ulteriormente la classificazione iniziale.

Viene presentata una estensiva valutazione dei metodi proposti, mediante confronto quantitativo con diversi metodi allo stato dell'arte su un dataset standard. I risultati sperimentali mostrano come gli schemi proposti ottengano risultati equivalenti o migliorati rispetto ai metodi concorrenti sulla maggior parte delle scene considerate, sia per il problema della segmentazione nei diversi oggetti che per il problema della classificazione semantica. In particolare, il numero ottimale di regioni risultanti viene automaticamente determinato dalle procedure iterative, mentre deve essere arbitrariamente deciso a priori in diversi algoritmi di segmentazione. Inoltre, non vengono poste assunzioni sulla forma degli oggetti nelle scene, a differenza di vari metodi concorrenti che sono ottimizzati per superfici planari. Questo è reso possibile dall'utilizzo delle superfici NURBS, che possono rappresentare indifferentemente sia elementi semplici come piani, sfere, cilindri che forme articolate e complesse.

Contents

Acknowledgements

I am deeply grateful to Pietro Zanuttigh for his friendly and helpful guidance as my PhD advisor.

I wish to thank all the researchers and PhD students in the LTTM group at the University of Padova for many inspiring conversations and for the fruitful collaboration. In particular, Ludovico Minto contributed to the research work of this thesis with the implementation of the Convolutional Neural Networks used for the semantic labeling.

I am grateful to my colleagues and managers at solidThinking Inc. for their encouragement to pursue my doctoral studies.

Finally, my deepest gratitude goes to my wife Ornella for her generous and unconditional support.

To Martina and Valeria, my brightest stars
and to Ornella, their mom

Chapter 1

Introduction

This chapter introduces the tasks addressed by this thesis, that is, the problems of objects segmentation and semantic labeling. They are described in Section 1.1, while Section 1.2 presents a survey of the related literature. An outline of the whole thesis and an overview of the proposed methods is provided by Section 1.3.

1.1 Problem description

One of the first tasks to understand a scene acquired by some sensor is to subdivide it into the separate objects and elements it consists of. This is called objects segmentation, or simply segmentation.

Scene segmentation on color images is a long-term research topic, which is far from being fully solved despite the huge amount of literature dedicated to it. Even the best performing methods are not able to provide a fully reliable solution in all conditions, since the problem is ill-posed and intrinsically difficult. For example, there can be separate touching objects with the same color, which are very hard to distinguish. Conversely, a single object can be made of portions with different colors, and it is not easily determined whether they should be separated or not.

The past recent years have seen a growing diffusion of low cost consumer depth cameras, as the Microsoft Kinect or the Asus Xtion. As illustrated in Figure 1.1, these sensors acquire a color image coupled with a depth map. By using the sensor calibration information, from this data it is possible to obtain a 3D point cloud, where each element has three spatial coordinates and an associated RGB color. The cloud is structured, since each point corresponds to a position on the sensor rectangular grid (or, equivalently, to a pixel on the depth map). Notice that while the color information is generally complete, there can be missing areas in the depth map caused by occlusions or bad light conditions deceiving the sensor.

Figure 1.1: Example of depth cameras (left), the acquired color image and depth map (center), the corresponding 3D colored point cloud (right).

Depth data is a very valuable aid for segmentation, since it conveys information about the 3D structure of the scene. Notice that the availability of spatial positions, coupled with the fact that the points are arranged on a grid, makes it possible to reconstruct the orientation of the object surfaces. Thanks to this geometric information, the segmentation problem can be restated as the attempt to partition a set of samples comprising both visual and spatial clues. The process becomes similar to what is done by the human brain, when the disparity between the images perceived by the two eyes is combined with the features observed from the color data, and they are used together to separate the different objects based on prior knowledge. Following this rationale, the segmentation methods presented in this thesis all exploit both color and depth information.

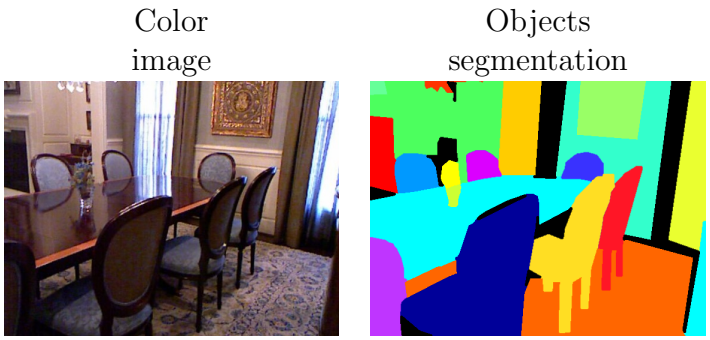

Figure 1.2: A sample scene with a manually created ground truth for the objects segmentation problem. All the object instances (e.g., the chairs) are given a different color.

A closely related problem is semantic segmentation, in which it is attempted to

assign each point of the scene to some predefined class (labeling). Notice that this automatically provides a subdivision of the scene, by considering as segments the regions that are assigned the same label. The difference with respect to objects segmentation is that separate object instances belonging to the same class are not distinguished. For example, in Figure 1.2 (showing an objects segmentation) the different chairs, walls and windows are considered as separate items, while in Figure 1.3 (semantic segmentation) they are labeled the same way since belonging to the same class. In this thesis, we propose schemes that address both the objects segmentation and the semantic segmentation tasks.

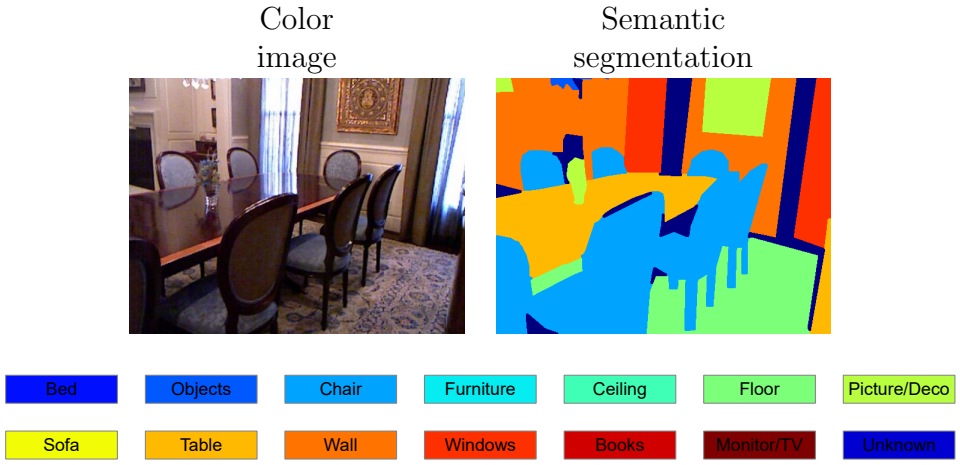

Figure 1.3: The same scene of Figure 1.2 with the ground truth for the semantic segmentation task. Each pixel is assigned to one of the predefined listed classes.

1.2 Related works

A large number of works address image segmentation. A survey of the best known ones is in [51]. As already pointed out, this is a long-term research field for which many techniques based on different insights have been proposed. Among them, the approaches based on graph theory and clustering algorithms have been particularly successful [6, 48, 18]. Anyway, despite the huge amount of research none of the existing methods can provide fully satisfactory results in all the possible conditions, since it is very difficult to properly estimate the scene structure from color data alone.

Even if the usage of depth data for segmentation purposes is a quite recent research field, several approaches jointly exploiting color and geometry information have already been proposed. A recent review is presented in [59]. A simple solution

is to perform two independent segmentations on the color data and on the depth data, and then join the two results as in [5].

In general, several clustering techniques originally applied to images can be directly adapted to joint depth and color segmentation, and different approaches based on this idea have been proposed. In [3] Mean Shift clustering [6] is applied to 6D vectors containing both the color and the spatial components for each sample, while superparamagnetic clustering is used in [54]. A joint color, spatial and directional clustering method associated to a planar region merging scheme is used in [28] and in the refined version of the same approach proposed in [27].

A method based on graph cuts is applied in [9] for joint color and depth segmentation. The approach of [37] exploits Normalized Cuts segmentation [48] together with saliency maps. In [11] a segmentation scheme also based on Normalized Cuts, that is able to automatically balance the relevance of color and depth, is proposed. This approach, extended to include also orientation information as described in Section 2.2 , is used at the intermediate steps of our iterative schemes.

Region splitting and region growing methods have also been proposed. The approach of [12] starts from an initial superpixel segmentation and then joins the segments based on a saliency metric. Superpixels and region merging are used also in [58] where graph-based criteria are used for the merging stage. Superpixels are combined together also in [16] , that computes regions corresponding to planar surfaces with an approach based on Rao-Blackwellized Monte Carlo Markov Chain. This idea is extended for the segmentation of multiple depth maps in [50].

In [46] a graph cuts method is applied to a mixture of planes and NURBS surfaces, with the relations between the surface patches learned by a support vector machine from user annotated training data. This work bears some indirect resemblance to our schemes for the fact that NURBS surface fitting is employed, even though in our case the approximation results are directly used to control the iterative procedures and this is a fully different approach. A planar model is used also in [52], where dynamic programming is exploited to extract the flat surfaces inside indoor scenes.

Some approaches deal with the close but less general problem of separating the foreground from the background [20, 26, 38, 33, 57]. In [57] two likelihood functions, based on color and depth data, are used together for this purpose. The work of [20] uses two distinct Gaussian Mixture Models in the depth and color spaces to represent the foreground and combine them in a Bayesian framework. Mixture of Gaussians are used in [26] as well. Both this approach and [38] consider also temporal constraints in depth and color videos. Finally some works as [36, 35, 4] try to solve the segmentation and the stereo disparity estimation problems

together.

Several works address the problem of classifying the segments after separating them, thus dealing with the tasks of objects segmentation and labeling together. Among these, [24] performs a hierarchical segmentation based on the output of contour extraction. Another combined approach is presented in [49], where an initial over-segmentation based on the watershed algorithm is followed by a hierarchical scheme. Combined segmentation and labeling are also addressed in [45] that exploits a MRF superpixel segmentation associated with a tree-structured approach. The semantic segmentation problem is typically addressed by using machine learning approaches. Ren et al. [45] exploit an over-segmentation with Markov Random Fields followed by a tree-structured algorithm. The works of [13] and [39] use Conditional Random Fields (CRF) instead. The approach of [13] combines CRF with mutex constraints based on geometry data, while the method of [39] combines 2D segmentation, 3D geometry data and contextual information. The work of [2] is 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 with notable results [31, 7, 47]. A pioneer work among these is [7], that uses a multiscale Convolutional Neural Network (CNN). The method of [47] achieves a very high accuracy exploiting Fully Convolutional Networks. The approach of [25] is based on a Convolutional Neural Network that acts on geometric features. Wang et al. [56] use two different CNNs, one for color and one for depth, and then a feature transformation network that separates the information shared by the two clues from the one specific to each of them. The work of [15] provides a semantic labeling together with depth and normal estimations using a multiscale CNN. Finally, the method of [55] uses a deep learning approach to extract superpixel features that are afterwards classified by support vector machines.

1.3 Outline of the proposed methods

This thesis is organized as follows. Our implementation of the Normalized Cuts algorithm exploiting both geometry and color information is presented in Chapter 2. It extends the approach of [11] by including also the normals information, and is one of the two main building blocks of our iterative segmentation schemes. The second one is the approximation with NURBS surfaces, that is detailed in Chapter 3.

Then, Chapter 4 describes a first set of methods we propose. They are unsuper-

vised, since they rely on surface fitting alone to control the recursive procedures, without requiring any preliminary training stage. The first one is the region splitting method, exposed in Section 4.1. In this scheme, an initial segmentation into two clusters is performed with the Normalized Cuts algorithm of Chapter 2. Then, corresponding NURBS surfaces are determined by means of least squares approximation, and the mean squared error (MSE) between the positions measured by the sensor and the corresponding points on the approximating surfaces are determined. Subsequently, the segments with greater MSE get further subdivided into sub-segments, that get fitted by NURBS surfaces as well obtaining corresponding MSE values. The fact that MSE (properly weighted based on segments size) becomes better or worse is used as criterion to accept or reject the subdivision. The rationale is that, if a segment properly corresponds to an actual object in the scene (and thus it is not to be further split), the fitting error is expected to be small, while segments that need to be further subdivided are expected to give a larger error. By iterating the procedure until no more subdivisions that improve the MSE are possible, a binary tree is obtained, and the final nodes are the elements of the segmentation proposed as result. We proposed this method initially in [43], and then in [42] where a discussion of several possible fitting error metrics other than the MSE is presented. These metrics are described in Section 3.2.

In addition to the region splitting scheme, a region merging iterative method is proposed in Section 4.2. It applies an initial over-segmentation, and then determines the adjacent segments based on compatibility criteria that depend on the 3D positions, on the normal directions and on the color information along the boundaries of the segments. For each couple of adjacent segments, the union is considered and approximated by a parametric NURBS surface (similarly as in the region splitting method). The corresponding fitting error is compared to the one calculated on the two separate segments, and the union operation is accepted if the error is improving. As in the previous region splitting method, the assumption is that if a segment actually represents a single object of the scene, and then must not be merged with the adjacent ones, the approximation error is expected to be already optimal, while it is expected to improve for segments corresponding only to portions of the objects. This scheme was presented in [41].

An additional method is proposed in Section 4.3 by combining the previous ones, that is, a region splitting iterative scheme is first applied, followed by a subsequent region merging procedure, both leveraging the NURBS surfaces approximation and the analysis of the corresponding fitting error to control the iterations. An extensive evaluation of the proposed methods is presented in Section 4.4, where a quantitative comparison against several state-of-the-art approaches on a standard

dataset is performed.

Chapter 5 describes how the proposed iterative schemes can be coupled with a deep learning classification step performed with CNNs (Convolutional Neural Networks). We test two CNN architectures, both detailed in Section 5.1. After a learning stage on a training set, the implemented CNNs provide the probability values for each scene pixel to belong to each considered class. This gives a discrete probability density function (PDF) for every segment of a preliminary over-segmentation, by averaging over the segment pixels. Then, by estimating how two PDFs are similar to each other, one can determine a similarity score for each couple of adjacent segments. This gives the opportunity to develop a variant of the previous region merging scheme, where the similarity scores are used together with the surface fitting errors to control the iterative procedure. This method is described in Section 5.2 and a first implementation was presented in [40]. In the extended version proposed here, some information (fitting error, surface curvatures) resulting from the NURBS fitting on the initial over-segmentation is used as input for the CNN, in addition to the color and geometry data. Moreover, after obtaining the final segmentation from the merging procedure each segment is classified by checking the most probable label on each of its points (as computed by the CNN) and choosing the most common one. Section 5.3 presents the experimental results obtained by this scheme and the comparison against state-of-the-art methods, both for the task of objects segmentation and for semantic segmentation. Finally, Chapter 6 draws the conclusions.

Chapter 2

Joint segmentation of color and depth data

Our iterative segmentation procedures apply a variant of the Normalized Cuts method [48], revised in order to use geometric information in addition to color data. We use this algorithm both at the intermediate steps of our region splitting methods, and to provide the initial over-segmentation in our region merging methods. Section 2.1 briefly introduces the Normalized Cuts method, while Section 2.2 provides the details of our implementation.

2.1 Spectral clustering

In the Normalized Cuts method, introduced in [48] for images, the segmentation problem is restated as the partitioning of a weighted graph containing a node for each image pixel. The graph is fully connected, that is, all the couples of nodes are linked by an edge. The corresponding weight is some measure of similarity between the two pixels, typically depending both on their color difference and on their spatial proximity.

Given two groups of nodes A and B partitioning the graph, their cut is defined as the sum of weights of the edges that must be deleted to make A and B disconnected, that is

$$cut(A, B) = \sum_{i \in A, j \in B} w_{ij} \tag{2.1}$$

where w_{ij} is the similarity between nodes i and j. A possible segmentation method could consist in minimizing the cut, since this is equivalent to remove the links corresponding to low similarity. However, this does not provide an optimal solution, since the minimum cuts usually involve small sets of isolated graph nodes. A better

criterion can be obtained by minimizing the normalized cut, defined as

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{2.2}$$

where $assoc(A,V) = \sum_{i \in A, j \in V} w_{ij}$ is the sum of similarities between the nodes in A and all the nodes in the graph. $Ncut(A,B)$ is a measure of the disassociation between the groups, normalized to keep their total edge connections (and then their size) into account. Similarly, a measure of the total normalized association within the two groups A and B can be defined as

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}\,. \tag{2.3}$$

Since $assoc(A,V) = assoc(A,A) + cut(A, V \setminus A)$, it is

$$Ncut(A,B) = 2 - Nassoc(A,B)\,, \tag{2.4}$$

then minimizing the disassociation between the groups is equivalent to maximizing the association within each group.

Since the problem of computing the optimal $Ncut$ is NP-complete, Shi and Malik [48] show that an approximate solution can be obtained by considering the generalized eigenvalue problem

$$(D - W)y = \lambda D y \tag{2.5}$$

where D is a diagonal matrix storing the total connections exiting from each node, $D_{ii} = \sum_j w_{ij}$, and W is the matrix of the weights w_{ij}. Eq. 2.5 is equivalent to the regular eigenvalue problem

$$\left(I - D^{-1/2} W D^{-1/2}\right) z = \lambda z \tag{2.6}$$

and from the eigenvector corresponding to the second smallest eigenvalue it is possible to obtain the partition of the graph into two groups. To achieve the final segmentation, the two groups are then recursively repartitioned (2-way Ncut) until a threshold on the $Ncut$ value is reached. Alternatively, the number of final segments can be decided in advance and the final segmentation is obtained by considering all the top eigenvectors simultaneously (K-way Ncut).

The Normalized Cuts method is an effective approach for segmentation and is a generic framework, since it can be used with different similarity formulations. A drawback is that the weights matrix W grows as the square of the number of ele-

ments to be clustered, and this is a problem both in terms of required memory and of computation time needed for the eigenvalues calculation. An efficient approximation is proposed in [19], that exploits a technique for the numerical solution of integral eigenfunction problems known as the Nyström method. In this approach, a subset of the input points is randomly chosen and partitioned with highly reduced computational burden, then the solution is propagated to the complete points set (see [19] for further details). Notice that this method requires the number of final segments to be set in advance. This is not a limitation for our iterative procedures, since in our region splitting method we perform a binary segmentation at each step (see Section 4.1), while in the region merging schemes we apply the Normalized Cuts only for the initial over-segmentation (see Section 4.3). On this one, the number of initial segments can be quite arbitrary, since it is just an upper bound for the final number that is determined by the iterative scheme itself.

2.2 Geometry and color segmentation

In our implementation, we extend the idea of clustering multidimensional vectors containing both the color and the 3D position of the samples presented in [11], by considering also the normal direction of the surfaces. This orientation information used together with the spatial position makes it possible to better subdivide the different geometrical elements.

We build a nine-dimensional representation of the scene samples $\boldsymbol{p_k}$, $k = 1, \ldots, n_p$ by combining geometry and color data. For this, first of all the depth and color cameras are jointly calibrated (the approaches of [29] or [10] can be used for this). After calibration it is possible to obtain the 3D coordinates $x(p_k), y(p_k), z(p_k)$ of each sample in the 3D space, together with the corresponding color information given by the three $R(p_k), G(p_k), B(p_k)$ components. The 3D points represent a surface thanks to the implicit connectivity information given by the regular grid of the depth map, then we can compute the directions perpendicular to this surface with the border and depth dependent smoothing scheme of [32]. This results in a further normal vector $n_x(p_k), n_y(p_k), n_z(p_k)$ associated to each sample.

The 9D vectors obtained this way contain different types of information, then they can not be directly fed to the clustering algorithm of previous section. In order to obtain a suitable representation, color values are first converted to the CIELab perceptually uniform space, in order to give a perceptual significance to the distance calculated on the color components. The color of each sample p_k is

then represented by the vector

$$\boldsymbol{p_k^c} = [L(p_k), a(p_k), b(p_k)]\,, \quad k = 1, \ldots, n_p\,. \tag{2.7}$$

The geometric positions are simply represented by the 3D coordinates $x(p_k)$, $y(p_k)$, and $z(p_k)$, that is by the vector

$$\boldsymbol{p_k^g} = [x(p_k), y(p_k), z(p_k)]\,, \quad k = 1, \ldots, n_p\,. \tag{2.8}$$

Finally, orientation information is given by the 3 components of the normal vectors:

$$\boldsymbol{p_k^n} = [n_x(p_k), n_y(p_k), n_z(p_k)]\,, \quad k = 1, \ldots, n_p\,. \tag{2.9}$$

The segmentation algorithm must be insensitive to the scaling of the point-cloud geometry. Moreover, geometry and color distances must be evaluated on consistent representations. Therefore, the color data are normalized by the average standard deviation σ_c of the L, a and b components, obtaining the vectors $[\bar{L}(p_k), \bar{a}(p_k), \bar{b}(p_k)]$. Following the same rationale, the geometry components are normalized by the average standard deviation σ_g of the point coordinates, providing the vectors $[\bar{x}(p_k), \bar{y}(p_k), \bar{z}(p_k)]$. Similarly, the normal vectors $[\bar{n}_x(p_k), \bar{n}_y(p_k), \bar{n}_z(p_k)]$ are obtained by normalizing the three orientation components by their average standard deviation σ_n. Considering the above normalized information vectors together, each point is finally represented as

$$\begin{aligned}\boldsymbol{p_k^f} = [&\bar{L}(p_k), \bar{a}(p_k), \bar{b}(p_k), \lambda_1\bar{x}(p_k), \lambda_1\bar{y}(p_k), \lambda_1\bar{z}(p_k),\\ &\lambda_2\bar{n}_x(p_k), \lambda_2\bar{n}_y(p_k), \lambda_2\bar{n}_z(p_k)], \quad k = 1, \ldots, n_p\end{aligned} \tag{2.10}$$

where the λ_1 and λ_2 parameters control the relative contribution of the three types of information. High values of them increase the relevance of the spatial position and surface orientation, while low values increase the relevance of color information. They can be automatically tuned by extending the approach used in [11] to balance between color and geometry data, at the price of an increased computational complexity. As detailed in Chapter 4, in the preliminary versions of our iterative schemes we set λ_1 and λ_2 heuristically. In the final and more advanced ones instead we equally weight the color, position and orientation information, then we are able to avoid using these parameters whose proper setting was critical.

Once obtained the above 9D vectors, the Normalized Cuts clustering method with Nyström acceleration described on previous Section 2.1 can be applied to partition them. In Sections 4.1 and 4.3 we describe in detail at which steps of

our iterative segmentation methods we perform this operation. After the clustering algorithm, to avoid tiny regions caused by noise we apply a refinement stage removing the ones smaller than a predefined threshold T_p (we reassign them to the neighboring region that shares the longest common boundary). In case the clustering is applied to obtain the initial over-segmentation for a region merging procedure, we also split the disjoint segments into their connected components, to let the merging scheme reassign them to the proper region if needed.

Figure 2.1 shows the result of Normalized Cuts clustering on a sample scene. The algorithm is applied in order to partition the scene into 50 segments. The refinement stages with the small segments removed and the disjoint regions split into connected components are also shown. Notice that the normals information is necessary to separate surfaces with similar colors and close spatial positions but with different orientations, as the wall and the ceiling on the top left corner.

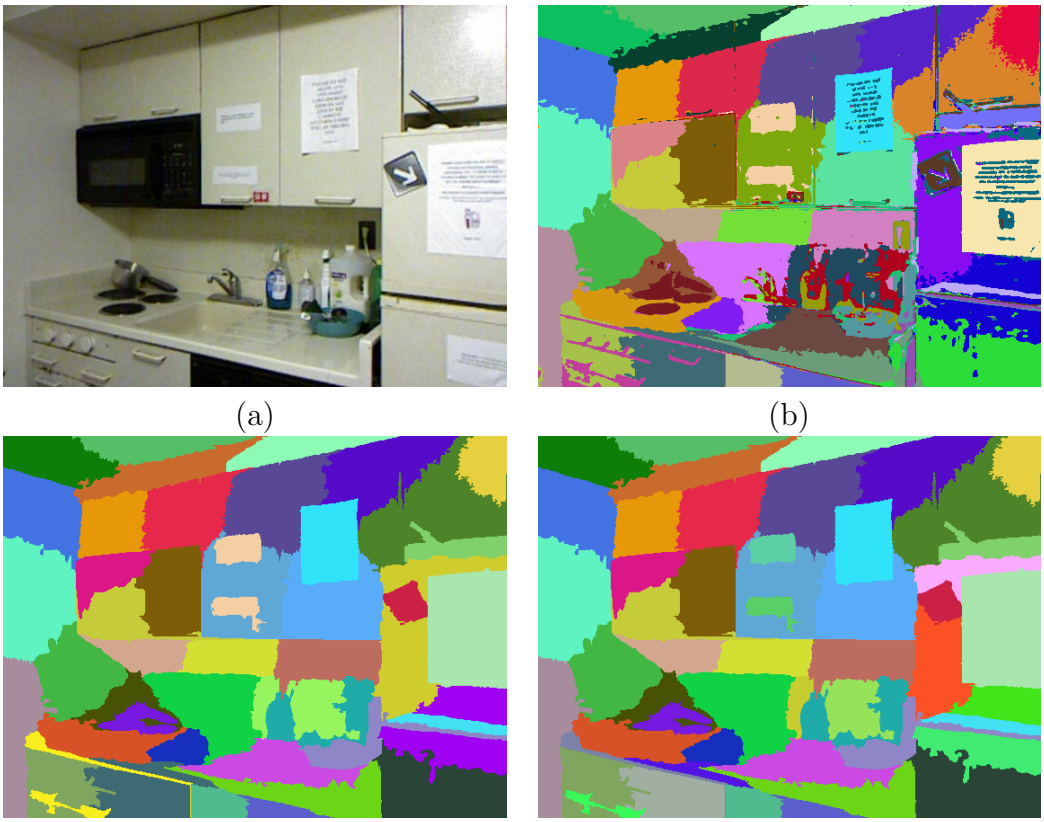

Figure 2.1: Segmentation of a sample scene. (a) Color Image. (b) Clustering performed on 9D vectors including spatial and color information (50 segments). (c) Result after reassignment of regions smaller than $T_p = 600$ pixels to nearest cluster (46 segments). (d) Final result after splitting disjoint segments into their connected components (52 segments).

Chapter 3

Surface fitting on segmented data

A key idea in the presented work is to apply a surface approximation scheme to the regions of a segmentation, to evaluate its accuracy and improve it with recursive splitting or merging procedures. Section 3.1 of this chapter briefly introduces the NURBS surfaces, that are the model we adopt for the approximation, and provides the details about our fitting implementation. Section 3.2 presents several possible metrics that can be used to evaluate the fitting results, while Section 3.3 discusses the numerical stability and the performances of the fitting algorithm.

3.1 Surface fitting

NURBS (Non-Uniform Rational B-Splines) are piecewise rational polynomial functions expressed in terms of proper bases. A complete overview of this topic can be found in [44], to which we refer for all the basic definitions and properties. They are frequently used in software applications in which 3D shapes need to be handled, thanks to their capability to represent both primitive objects as planes, spheres, cylinders without approximation, and free-form organic surfaces. The representation is given by parametric curves and surfaces expressed in a concise way by means of control points, whose coordinates are the coefficients of the linear combinations in terms of the bases functions. Namely, a parametric NURBS surface is defined as

$$\boldsymbol{S}(u,v) = \frac{\sum_{i=1}^{n}\sum_{i=1}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}\boldsymbol{P}_{i,j}}{\sum_{i=1}^{n}\sum_{i=1}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \tag{3.1}$$

where the $\boldsymbol{P}_{i,j}$ are the control points, the $w_{i,j}$ are the corresponding weights, the $N_{i,p}$ are the univariate normalized B-spline basis functions, and p, q are the degrees in the u, v parametric directions respectively. In our implementation we set the weights all equal to one, thus our fitted surfaces are non-rational (i.e., spline) and

Eq. 3.1 reduces to

$$\boldsymbol{S}(u,v) = \sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(u)N_{j,q}(v)\boldsymbol{P}_{i,j}\,. \tag{3.2}$$

In order to obtain a linear system from Eq. 3.2 we need to choose the (u_k, v_k) values in the 2D parametric domain corresponding to the 3D points P_k to approximate (that is, the ones that belong to the segment to fit). For this, we consider that each point is related to an element of the rectangular grid given by the sensor pixel arrangement, then we set the corresponding surface parameter value as the 2D pixel location on the image plane of the camera.

As to the degrees in the u and v directions, we initially set them equal to 3. Since the number of surface control points gives the degrees of freedom in our model, we set it adaptively depending on the number of input samples. In order to do this, we consider the horizontal and vertical extents of the segment to fit. Let W be the horizontal image size and H the vertical size (for example $W = 640$ and $H = 480$ in case of Kinect data). First, we set the values N_u and $N_v = \frac{H}{W}N_u$, that is, the maximum number of control points in the u and v parametric directions, to be used in case of a segment covering the whole image. For smaller ones we determine the number proportionally to the segment extents as follows. Let u_0, u_1 and v_0, v_1 be the minimum and maximum pixel values on the sensor grid corresponding to the segment in the horizontal and vertical directions. We set the number of control points in the u, v parametric directions respectively as

$$n = \max\left\{3, \left[N_u\,\frac{u_1 - u_0}{W}\right]\right\}, \quad m = \max\left\{3, \left[N_v\,\frac{v_1 - v_0}{H}\right]\right\}. \tag{3.3}$$

Notice that since the minimum number of control points for a cubic spline is 4, for smaller segments we lower the surface degree to quadratic in order to allow 3 control points as actual minimum. In our methods we set $N_u = 15$ or $N_u = 20$. This choice of parameters provides enough degrees of freedom to represent the shape of any common object, and the adaptive scheme at the same time prevents the fitting to always be more accurate for smaller segments, independently on how the segmentation algorithm was successful in detecting the objects in the scene. Figure 3.1 shows a sample segmentation, and for each segment the numbers n, m of control points determined accordingly to its extents (in this example it is $N_u = 20$).

Once determined the (u_k, v_k) parameter values corresponding to the points to fit, the surface degrees and the number of control points in the u, v parametric directions, we use the methods of [44] to obtain the NURBS knots, needed for the

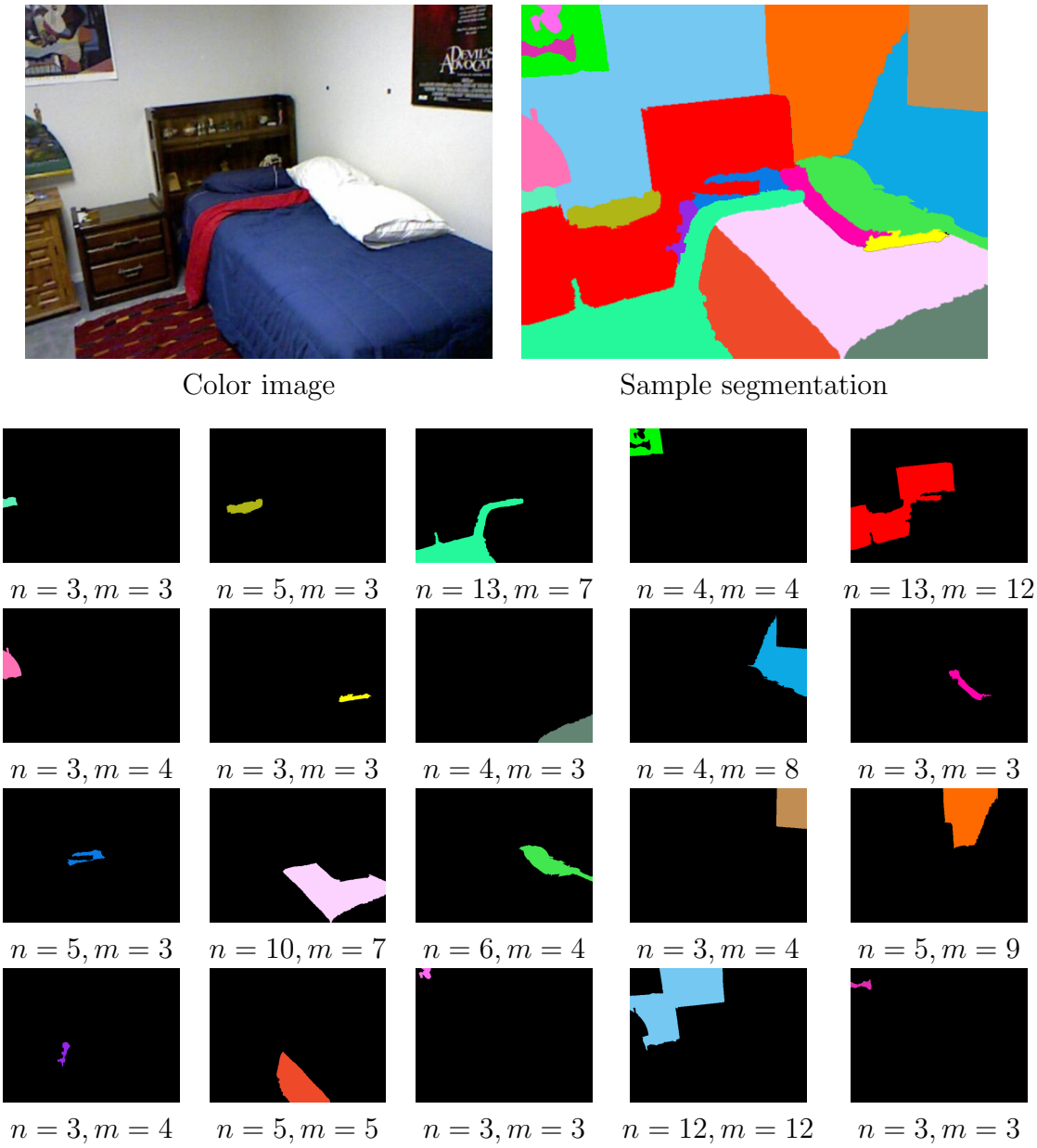

Figure 3.1: A sample segmentation with the binary masks for each region, and the corresponding numbers of control points (in the u and v parametric directions) used for the NURBS surface approximation.

definition of the $N_{i,p}$ basis functions. Finally, by considering Eq. 3.2 evaluated at (u_k, v_k) and equated to the points to fit, we obtain the linear system

$$NX = P \tag{3.4}$$

where:

- N is a n_p by nm matrix, where n_p is the number of points to fit and nm is the total number of control points. The k-th row contains the basis functions evaluated at the parameter values corresponding to the k-th point to fit, that is, the $N_{i,p}(u_k)N_{j,q}(v_k)$ terms listed with a single index ranging from 1 to nm.
- X is a nm by 3 matrix with the x, y, z unknown coordinates of the control points as columns.
- P is a n_p by 3 matrix with the x, y, z coordinates of the points to fit as columns.

It is always $n_p > nm$, then the linear system is over-determined. We solve it in the least squares sense, thus obtaining the surface control points.

Figure 3.2 shows a NURBS surface fitted over one region of a sample segmentation. The grid of control points is shown (3×4 in this case). The portion of the surface that actually corresponds to the segment points is highlighted in magenta. Notice that since we use bivariate tensor product NURBS surfaces [44], the parametric domain is always rectangular, while the segment shape on the sensor grid is usually irregular. This makes the surface larger then the segment itself, anyway, the fitting accuracy is evaluated only on the overlapping area. The surface portion extending outside the segment points is not relevant for our purposes then.

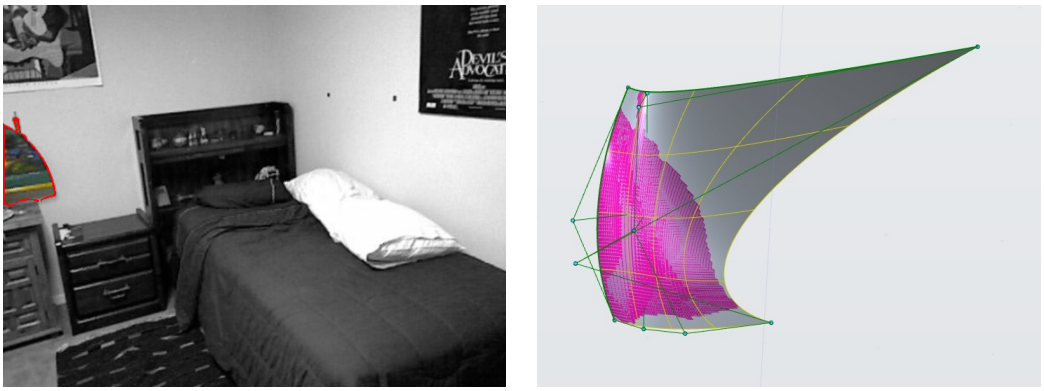

Figure 3.2: NURBS surface (right) fitted over the highlighted segment in the sample scene (left).

3.2 Fitting metrics

The goal of the surface approximation scheme of previous section is to provide a tool in our methods to evaluate how the elements of a candidate segmentation actually give a good representation for the objects in the scene. Our assumption is that there is a relationship between the fitting accuracy and the quality of the segmentation. In particular, we expect that if a segment contains multiple objects, the different depth values along their borders or the sharp edges will affect the fitting. This is visible in Figure 3.3, that shows a surface fitted over two segments. The surface is colored based on the pointwise fit error, that is, for each 3D sample P_k in a segment, we consider the corresponding 3D position $\boldsymbol{S}(u_k, v_k)$ on the fitted surface, and we calculate the fit error as the Euclidean distance $|P_k - \boldsymbol{S}(u_k, v_k)|$ between the two locations. In the figure color map, low and large fit errors correspond to dark blue and red respectively. Notice how the large fit error (red area) between the teddy head and the monitor portion clearly reveals that the two segments do not actually belong to the same object.

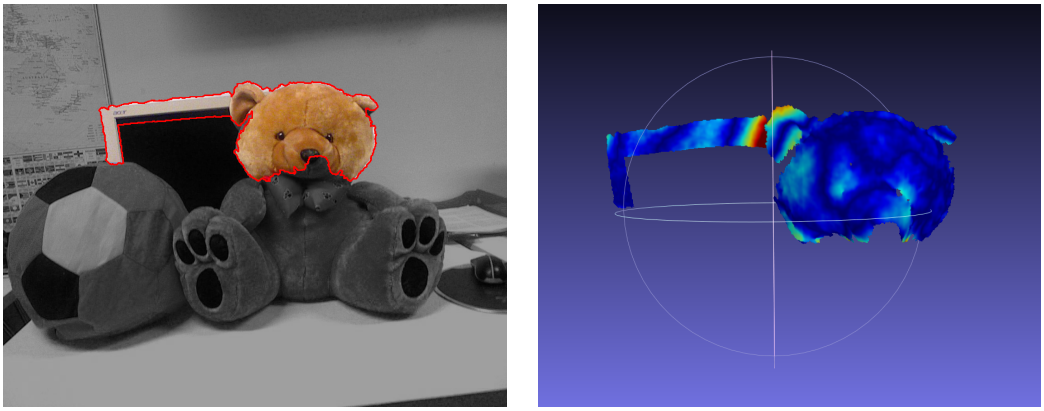

Figure 3.3: A 3D NURBS surface fitted over two regions of a sample segmentation. The red areas correspond to larger displacement between the points and the fitted surface.

The same can be seen on Figure 3.4, that shows the colored pointwise fit error map for all the regions in the sample segmentation of previous section. Notice how the segments S_2 and S_4, that span different objects, contain large red areas showing large fit errors. The corresponding fitted NURBS surfaces are shown with a shaded visualization on Figure 3.5, where it is visible how their shape is affected by the underlying structure of the scene objects.

Since we want to obtain iterative region splitting or region merging schemes based on these ideas, we need to associate some measure of the fitting accuracy to the whole segments. Namely, we want to define some measure e_i for segment S_i,

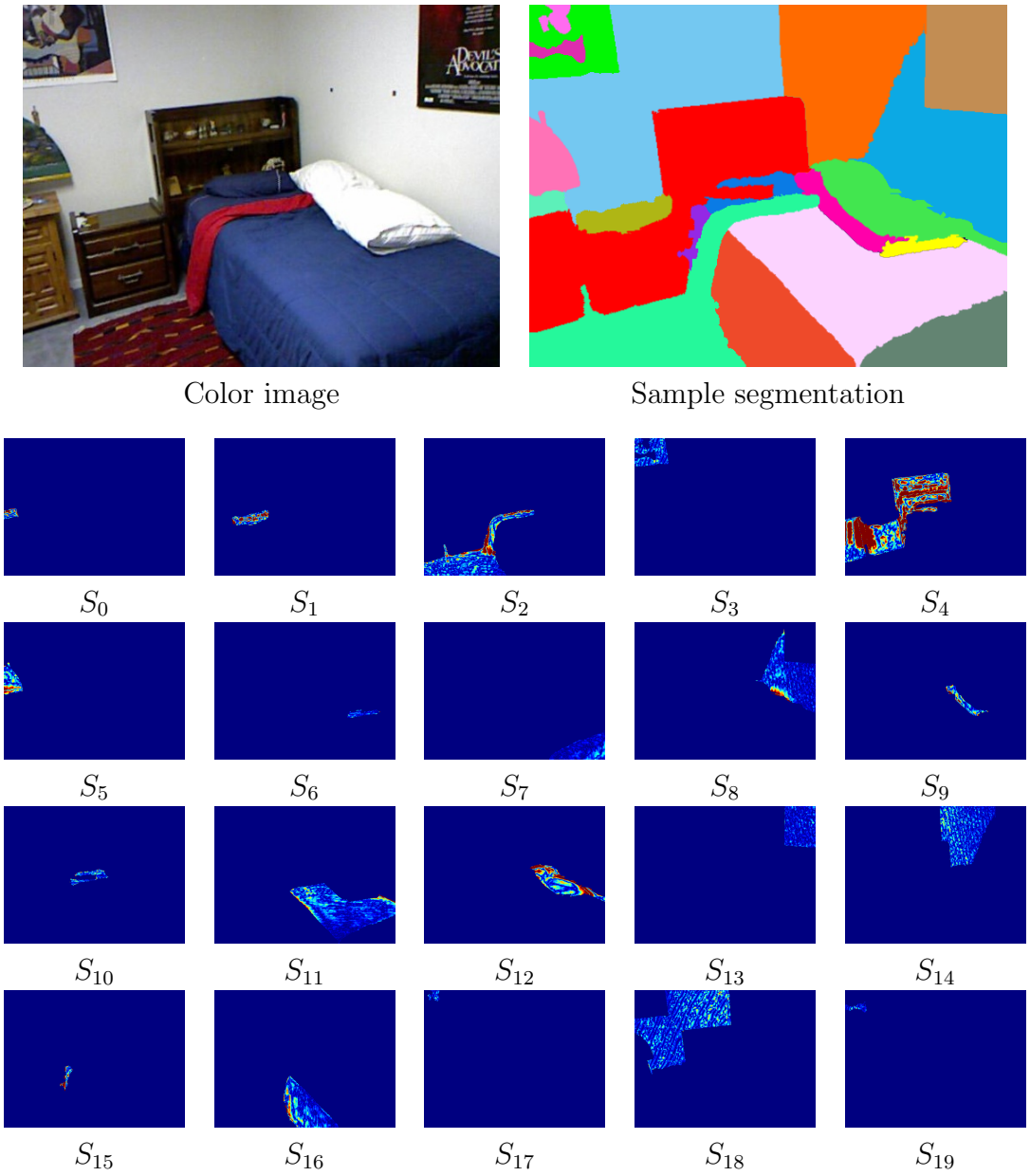

Figure 3.4: Pointwise fit error maps for all the regions in the sample segmentation of previous section. Dark blue and red correspond respectively to low and large fit error.

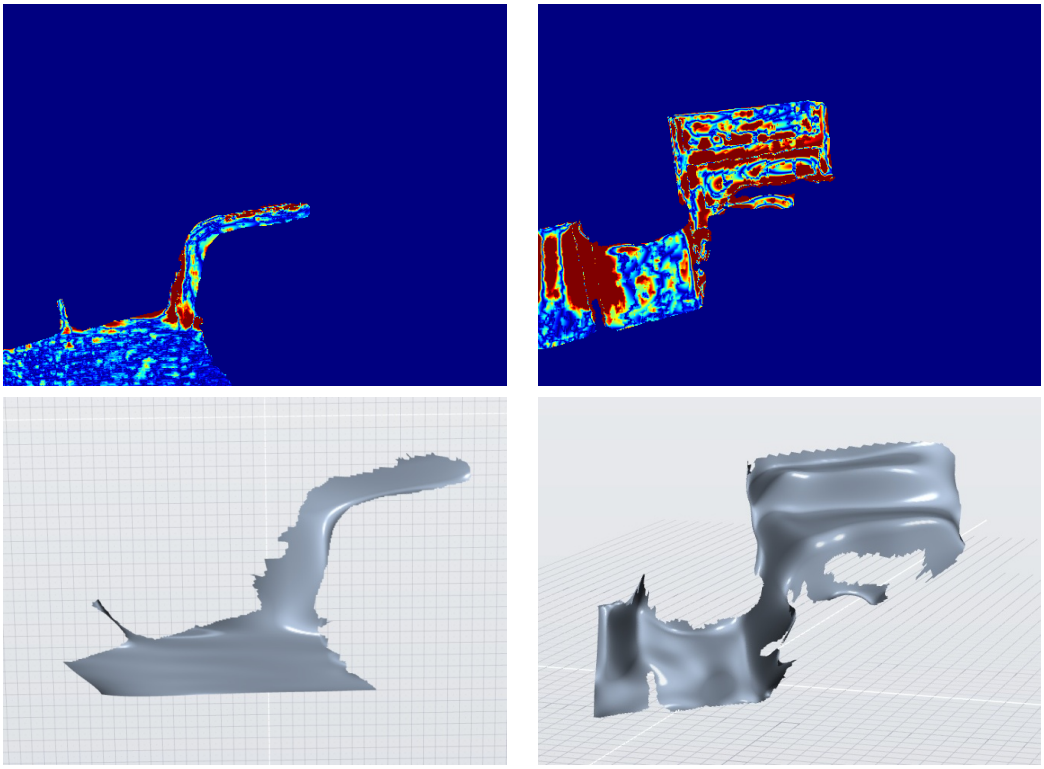

Figure 3.5: Detailed view of the fit error maps for segments S_2 and S_4 from previous figure (top), and the corresponding fitted NURBS surfaces (bottom).

dependent on the fitting results, to be used in a condition like

$$\frac{e_{i0}|S_{i0}| + e_{i1}|S_{i1}|}{e_i|S_i|} < T_e \tag{3.5}$$

where S_{i0} and S_{i1} are sub-segments candidate to replace S_i in a region splitting scheme, or conversely S_i is their union in a region merging scheme. The weights are the cardinalities of the two segments, while $T_e \leq 1$ is a threshold that accounts for noise in the data. In the discussion about the results on Section 4.4 we present some tests with the three different values 0.8, 0.9 and 1. Notice that with the latter value the criterion makes the segmentation with S_i replaced by S_{i0} and S_{i1} accepted if there is any improvement in the accuracy, independently on how large the improvement is.

In a region splitting method, the subdivision of S_i into S_{i0} and S_{i1} is accepted if Eq. 3.5 is satisfied, and discarded if not. Conversely, in a region merging method the fusion of S_{i0} and S_{i1} into S_i is accepted if Eq. 3.5 does not hold. In both cases, the rationale is that the split or merge operation is accepted if improving the fitting accuracy, and discarded otherwise.

To define a possible metric e_i, we propose two different approaches. In the first family the fitting accuracy is considered, then e_i is defined in terms of the pointwise fitting error. We define four metrics following this rationale, that is, the Mean Square Error (MSE), the Mean Absolute Error (MAE), the Variance of the Error (VE) and the Number of points with a Large Error (NLE). The second idea consists in analyzing the curvature of the fitted surfaces, since we expect edges or jumps in the depth values to result in large curvature values or variations. Based on this idea we define other five metrics, that is, the Variance of the MaX Curvature absolute value (VMXC), the Variance of the MEan Curvature (VMEC), the Variance of the Gaussian Curvature (VGC), the Mean of the MaX Curvature absolute value (MMXC) and the Number of points with a Large Curvature (NLC). The definition of these metrics is given in the following sections, while a comparison between the results obtained with each of them is presented in Section 4.4.

Mean Square Error (MSE)

This metric is the Mean Square Error (MSE) between the 3D positions P_k in segment S_i and the points obtained on its NURBS approximation by evaluation at the corresponding parameter values (u_k, v_k), that is:

$$e_i^{MSE} = \frac{\sum_{P_k \in S_i} |P_k - \boldsymbol{S}(u_k, v_k)|^2}{|S_i|}\,. \tag{3.6}$$

The MSE is directly related to the fitting accuracy. By employing it, we exploit the already mentioned idea that properly segmented regions should be accurately fitted and then provide a low MSE value, since containing a single object surface, while segments enclosing multiple surfaces at different depths can not be accurately fitted and will give higher MSE values. With this metric, the criterion of Eq. 3.5 to perform a subdivision of segment S_i into S_{i0} and S_{i1} in a region splitting procedure, or conversely to reject the union operation in a region merging scheme, becomes

$$\frac{e_{i0}^{MSE}|S_{i0}| + e_{i1}^{MSE}|S_{i1}|}{e_i^{MSE}|S_i|} < T_e \tag{3.7}$$

where T_e is the threshold already introduced.

Mean Absolute Error (MAE)

We obtain another metric directly related to the fitting accuracy as the Mean Absolute Error (MAE) between the 3D positions in segment S_i and the corresponding points evaluated on the NURBS fitted surface. We define it similarly to the MSE, except for the fact that the absolute values are used instead of their squares, that is:

$$e_i^{MAE} = \frac{\sum_{P_k \in S_i} |P_k - \boldsymbol{S}(u_k, v_k)|}{|S_i|} \,. \tag{3.8}$$

While the MSE gives more importance to large errors due to the square operation, this metric assigns a more uniform weight to the fitting errors. The criterion to evaluate the segmentation becomes

$$\frac{e_{i0}^{MAE}|S_{i0}| + e_{i1}^{MAE}|S_{i1}|}{e_i^{MAE}|S_i|} < T_e \tag{3.9}$$

where T_e is as above.

Variance of the Error (VE)

This metric measures the Variance of the Error (VE) between the 3D positions in segment S_i and the corresponding points evaluated on the NURBS approximation, that is:

$$e_i^{VE} = \frac{\sum_{P_k \in S_i} \left(|P_k - \boldsymbol{S}(u_k, v_k)| - e_i^{MAE}\right)^2}{|S_i|} \,. \tag{3.10}$$

Instead of taking the absolute error value into account, this metric considers its deviation from the mean. The rationale is that the discontinuities could produce large errors in restricted areas, and yet there can be large areas with small errors

at the same time. The criterion for the splitting and merging schemes involves the ratio between the variances before and after the segment subdivision, as for the other metrics:

$$\frac{e_{i0}^{VE}|S_{i0}| + e_{i1}^{VE}|S_{i1}|}{e_i^{VE}|S_i|} < T_e \tag{3.11}$$

where T_e is as above.

Number of points with a Large Error (NLE)

Following the same rationale, that is, considering the possible presence of localized regions with large fitting errors, we define another metric based on the number of points with an associate absolute error greater than a threshold T_{le}. The idea here is that the samples corresponding to an edge or close to the jumps between two objects in the same segment should be detected since having an error larger than the threshold, while the error should stay within it for the other ones. A drawback is that properly setting the threshold T_{le} is critical to obtain optimal results. A proper value should depend on the amount of noise on the depth camera data. For the comparison of these metrics discussed on Section 4.4 we set the value to 0.1 (our data are in meters).

As to the criterion to evaluate the segmentation, an improving subdivision of a segment should make the sum of the number of samples $e_{i0}^{NLE} + e_{i1}^{NLE}$ with a large error on the two parts smaller than the number of points e_i^{NLE} with a large error on the original segment, that is:

$$\frac{e_{i0}^{NLE} + e_{i1}^{NLE}}{e_i^{NLE}} < T_e \tag{3.12}$$

where $T_e \leq 1$ is the threshold previously defined (notice that in this case two thresholds are required).

Variance of the MaX Curvature absolute value (VMXC)

For this metric, we consider the two principal curvatures κ_1 and κ_2 of the NURBS approximating surface at the locations corresponding to the points in S_i (i.e., the maximum and minimum local curvature values, see [14]). We expect that for segments containing multiple objects, the fitted surface would show high oscillations caused by the depth jumps, corresponding to large curvature values. Therefore, we take the maximum of the absolute values of the principal curvatures, $\kappa^{max} = \max(|\kappa_1|, |\kappa_2|)$, and we consider as error metric its variance over the points

in the segment S_i

$$e_i^{VMXC} = \frac{\sum_{P_k \in S_i} \left(\kappa_k^{max} - e_i^{MMXC}\right)^2}{|S_i|} \tag{3.13}$$

where

$$e_i^{MMXC} = \frac{\sum_{P_k \in S_i} \kappa_k^{max}}{|S_i|} \tag{3.14}$$

is the mean of κ^{max} over S_i (notice that we denote by κ_k^{max} the value of κ^{max} for the NURBS fitting surface at the location corresponding to P_k). The criterion to accept the segmentation of S_i into segments S_{i0} and S_{i1} is then

$$\frac{e_{i0}^{VMXC}|S_{i0}| + e_{i1}^{VMXC}|S_{i1}|}{e_i^{VMXC}|S_i|} < T_e \tag{3.15}$$

where T_e is as for the previous metrics.

Variance of the MEan Curvature (VMEC)

Similarly as above, we consider as error metric the variance e_i^{VMEC} of the mean curvature, $H = \frac{1}{2}(\kappa_1 + \kappa_2)$. That is, considering $e_i^{MMEC} = \frac{\sum_{P_k \in S_i} H_k}{|S_i|}$ which is the mean of H over S_i, it is

$$e_i^{VMEC} = \frac{\sum_{P_k \in S_i} \left(H_k - e_i^{MMEC}\right)^2}{|S_i|} \tag{3.16}$$

where H_k is the mean curvature of the NURBS fitting surface at the location corresponding to P_k. This is a variation of metric VMXC, based on the idea that the maximum curvature absolute value could be very high because of just one of the two principal curvatures, and then a metric based on the mean curvature could be more adequate for some shapes. Moreover, by taking the absolute values it is possible that information about large variations between positive and negative curvature values gets lost, while it is taken into account by using the mean curvature instead. The criterion for the segmentation is then

$$\frac{e_{i0}^{VMEC}|S_{i0}| + e_{i1}^{VMEC}|S_{i1}|}{e_i^{VMEC}|S_i|} < T_e \tag{3.17}$$

where T_e is as above.

Variance of the Gaussian Curvature (VGC)

We obtain another variation of metric VMXC by considering the variance of the Gaussian curvature $K = \kappa_1\kappa_2$. That is, the metric is

$$e_i^{VGC} = \frac{\sum_{P_k \in S_i} \left(K_k - e_i^{MGC}\right)^2}{|S_i|} \tag{3.18}$$

where $e_i^{MGC} = \frac{\sum_{P_k \in S_i} K_k}{|S_i|}$ is the mean of K over S_i (and K_k is the Gaussian curvature of the NURBS approximation at the location corresponding to P_k). Notice that the previous considerations about using the mean curvature instead of the maximum curvature absolute value hold for the Gaussian curvature too. The corresponding criterion for the segmentation is

$$\frac{e_{i0}^{VGC}|S_{i0}| + e_{i1}^{VGC}|S_{i1}|}{e_i^{VGC}|S_i|} < T_e \tag{3.19}$$

where T_e is as above.

Mean of the MaX Curvature absolute value (MMXC)

In addition to the curvature variance as an indicator of surface oscillations, we consider large values of the curvatures themselves as an index of poor segmentation, since they are expected to correspond to sharp edges, or gaps between separate objects. Following this rationale, we consider e_i^{MMXC} defined in Eq. 3.14, that is, the mean of κ^{max} maximum of the absolute values of the principal curvatures over the points in a segment S_i, as a further error metric. The corresponding criterion for the segmentation is

$$\frac{e_{i0}^{MMXC}|S_{i0}| + e_{i1}^{MMXC}|S_{i1}|}{e_i^{MMXC}|S_i|} < T_e \tag{3.20}$$

where T_e is as above.

Number of points with a Large Curvature (NLC)

To investigate the presence of local regions with large curvature values, we also take into account the number of points for which the maximum of the absolute values of the principal curvatures κ^{max} is greater than a threshold T_κ. This is roughly equivalent to counting the points that correspond to edges or gaps between the objects. Clearly it is not straightforward how to set the threshold T_κ, since an optimal value would depend both on the noise in the data and on the shape of the

objects. For our results we set it to 10, corresponding to a radius of curvature of 0.1. Then, the criterion we obtain for the segmentation is

$$\frac{e_{i0}^{NLC} + e_{i1}^{NLC}}{e_i^{NLC}} < T_e \tag{3.21}$$

where e_i^{NLC} is the number of points in the segment S_i for which κ^{max} is greater than T_κ, and T_e is as for the previous metrics (two thresholds are required then).

3.3 Numerical stability and performances

In this section we provide some details about how we solve the linear least squares problem given by Eq. 3.4. First of all, care must be taken since the data acquired by sensors like the Kinect are noisy, and this can make the linear system ill conditioned. Also, the fact that the NURBS surfaces are defined on a rectangular parametric range while the segments to approximate may have irregular shapes can make matrix N rank deficient. Then, to check the behavior on real data we have tested several algorithms.

A widely used method for solving the least squares problem is to consider the normal equations

$$N^T N X = N^T P \tag{3.22}$$

for which a solution always exists (it is also unique when N is full rank since $N^T N$ is symmetric positive definite). Stability problems can arise when $N^T N$ is ill conditioned. A more stable approach, suitable also for the case where $N^T N$ is singular, is to calculate the SVD decomposition of N, even if computationally more expensive (see [21] for a thorough discussion). An even higher stability can be obtained with truncated SVD or Tikhonov regularization [21]. Clearly, the choice of the proper method to use is highly dependent on the problem to solve and on the data. Considering this, we have tested the following methods (in our implementation we use Eigen [22], an optimized C++ linear algebra library):

- (SVD) Singular values decomposition of matrix N.
- (TSVD) Truncated singular values decomposition.
- (SVDTR) Singular values decomposition with Tikhonov regularization.
- (LDLT) Robust Choleski decomposition of matrix $N^T N$. This method exploits the fact that $N^T N$ is positive definite (semidefinite if N is rank deficient).

- (LU-FP) LU decomposition of matrix N^TN with full pivoting. This method provides one of the non-unique solutions also if N is rank deficient and thus N^TN is singular.

- (LU-PP) LU decomposition of matrix N^TN with partial pivoting. This method assumes N^TN to be invertible, than it can not be applied if N is rank deficient.

Table 3.1 lists the timing results of the tested methods on the sample segmentation of Figure 3.1. For each of the 20 regions the number n_p of points to fit and the number nm of NURBS surface control points are shown (recall that they are respectively the number of rows and columns of matrix N, see Section 3.1). The table also reports the condition number $\kappa(N)$ of matrix N, defined as the ratio between its maximum and minimum singular values ($+\infty$ means that N is not full rank and N^TN is singular). Table 3.2 shows the MSE values calculated between the segments and the NURBS surfaces obtained with the various methods.

First, we notice that there is a huge speed difference between the SVD-based methods and the ones based on solution of the normal equations, as expected. Among the latter ones, LU-PP is not suitable since it does not guarantee a valid result when N^TN is singular (this happens for segments S_2, S_4 and S_{18}). More interestingly, the speed penalty of the SVD-based methods is not balanced by any improvement on the results accuracy, since the differences on the resulting MSE values are negligible, as evident in Table 3.2. In particular, the LDLT method provides the same values of SVD decomposition. Moreover, by comparing SVD against TSVD, we observe that there are noticeable differences regarding the MSE values only for segments S_2 and S_{18}, two cases where N^TN is singular. These differences are however very small (0.16% and 0.1% respectively), then it turns out that there is no gain in using regularization techniques for our purposes. This may seem unexpected considering the noise on data coming from the employed RGB-D sensors, but it can be justified by the high numerical stability of the NURBS B-spline bases [44].

We have obtained similar outcomes on all the sample segmentations used to test the various methods. Based on this results, in all our segmentation schemes we use the LDLT method.

	n_p	nm	$\kappa(N)$	SVD	TSVD	SVDTR	LDLT	LU-FP	LU-PP
S_0	1123	9	17,0	< 5	< 5	< 5	< 5	< 5	< 5
S_1	3141	15	164,1	< 5	< 5	15	< 5	< 5	< 5
S_2	20740	91	$+\infty$	797	796	812	46	36	31
S_3	6514	16	45,4	15	< 5	< 5	< 5	< 5	< 5
S_4	37311	156	$+\infty$	3842	3843	3851	156	165	156
S_5	3729	12	79,7	< 5	< 5	< 5	< 5	< 5	< 5
S_6	1293	9	25,0	< 5	< 5	< 5	< 5	< 5	< 5
S_7	9183	12	139,7	< 5	15	15	< 5	< 5	< 5
S_8	21297	32	5872,6	94	109	93	< 5	< 5	< 5
S_9	2667	9	123,2	< 5	15	< 5	< 5	< 5	< 5
S_{10}	2115	15	108,9	< 5	< 5	< 5	< 5	< 5	< 5
S_{11}	28583	30	1722,2	119	125	125	15	< 5	15
S_{12}	9193	24	1619,8	20	15	31	< 5	< 5	< 5
S_{13}	11987	12	19,8	11	< 5	< 5	< 5	< 5	< 5
S_{14}	22302	45	8956,5	218	203	218	15	25	< 5
S_{15}	1033	12	106,8	< 5	< 5	< 5	< 5	< 5	< 5
S_{16}	14065	9	9134,7	33	31	46	< 5	15	< 5
S_{17}	802	9	16,5	< 5	< 5	< 5	< 5	< 5	< 5
S_{18}	39879	144	$+\infty$	4354	4187	4218	140	156	156
S_{19}	1043	9	29,0	< 5	< 5	< 5	< 5	< 5	< 5
Total				9514	9339	9424	372	409	358

Table 3.1: Comparison of times (milliseconds) taken by the different solution methods for each segment S_i in the sample segmentation of Figure 3.1. The table also lists the number of rows n_p and columns nm of matrix N, and its condition number $\kappa(N)$.

	$\kappa(N)$	SVD	TSVD	SVDTR	LDLT	LU-FP	LU-PP
S_0	17.0	0.001332	0.001332	0.001332	0.001332	0.001332	0.001332
S_1	164.1	0.001697	0.001697	0.001726	0.001697	0.001697	0.001697
S_2	$+\infty$	0.000806	0.000807	0.001081	0.000806	0.000806	0.000806
S_3	45.4	0.000104	0.000104	0.000104	0.000104	0.000104	0.000104
S_4	$+\infty$	0.005537	0.005538	0.005944	0.005537	0.005537	NaN
S_5	79.7	0.000536	0.000536	0.000539	0.000536	0.000536	0.000536
S_6	25.0	0.000081	0.000081	0.000081	0.000081	0.000081	0.000081
S_7	139.7	0.000037	0.000037	0.000038	0.000037	0.000037	0.000037
S_8	5872.6	0.000434	0.000434	0.000438	0.000434	0.000434	0.000434
S_9	123.2	0.000635	0.000635	0.000647	0.000635	0.000635	0.000635
S_{10}	108.9	0.000219	0.000219	0.000250	0.000219	0.000219	0.000219
S_{11}	1722.2	0.000146	0.000146	0.000146	0.000146	0.000146	0.000146
S_{12}	1619.8	0.001608	0.001608	0.002001	0.001608	0.001608	0.001608
S_{13}	19.8	0.000037	0.000037	0.000037	0.000037	0.000037	0.000037
S_{14}	8956.5	0.000098	0.000098	0.000100	0.000098	0.000098	0.000098
S_{15}	106.8	0.002624	0.002624	0.002704	0.002624	0.002624	0.002624
S_{16}	9134.7	0.000278	0.000278	0.000290	0.000278	0.000278	0.000278
S_{17}	16.5	0.000089	0.000089	0.000089	0.000089	0.000089	0.000089
S_{18}	$+\infty$	0.000113	0.000113	0.000118	0.000113	0.000113	NaN
S_{19}	29.0	0.000072	0.000072	0.000072	0.000072	0.000072	0.000072
Avg.		0.000824	0.000824	0.000887	0.000824	0.000824	NaN

Table 3.2: Comparison of the MSE values (square meters) obtained with the different solution methods for the same segmentation of previous table. The condition number $\kappa(N)$ of matrix N is also shown (notice that N is rank deficient for segments S_2, S_4 and S_{18}). In the NaN cases the LU-PP method failed to provide a solution and early exited due to a division by zero.

Chapter 4

Segmentation schemes based on surface fitting

After introducing the main building blocks in the previous chapters, we present a first family of iterative segmentation schemes. They rely on the surface fitting results to control the iterations, and do not require any knowledge or learning stage on the input data. The first one is the region splitting scheme, detailed in Section 4.1. In Section 4.2 the region merging scheme is presented, while Section 4.3 describes how a third method can be obtained by combining the previous ones. Section 4.4 discusses the experimental results.

4.1 Region splitting

A general overview of our region splitting method is shown in Figure 4.1. First, the nine-dimensional point cloud representation of Eq. 2.10 is built and used as initial input. Then, a binary segmentation using the method of Section 2.2 is applied, and a NURBS surface is fitted on each of the two resulting segments as described in Section 3.1, in order to determine the corresponding fitting error values. As seen in Section 3.2, different possible fitting error metrics can be used, either based on the evaluation of the fitting error or on the curvature of the fitted surfaces. In [42] we perform a detailed comparison, whose results are presented also in Section 4.4 of this work. According to that evaluation, the error metric given by the Mean Square Error (MSE) between the depth samples in the segment and the corresponding points on the approximating NURBS surface is the most adequate in general. We will refer to the MSE as fitting error in the following then, unless otherwise specified.

At each next step of the procedure, the segment with the greatest fitting error is

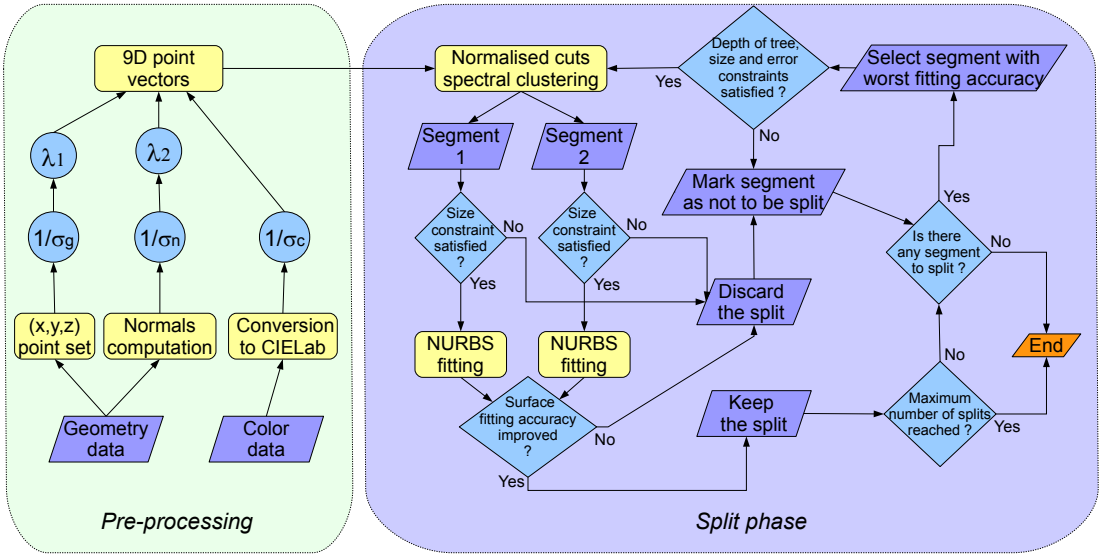

Figure 4.1: Overview of the region splitting approach.

examined, and it is further split provided it is not too small, or the recursion has not yet produced enough segments. If the split operation improves the fitting accuracy, it is accepted, and the segment is replaced by the two sub-segments. Otherwise the operation is discarded, that is, the segment is kept unchanged and marked as not to be considered for splitting anymore. The procedure then continues by processing the next segment with greatest fitting error.

More in detail, consider an intermediate step during which segment S_i is the one being processed, its fitting error e_i being the largest one. The following conditions are checked in order to consider S_i for a split operation:

1. The size of S_i must be greater than $2T_p$, otherwise the split would produce at least one segment smaller than T_p. This is consistent with the choice of not allowing segments smaller than T_p made in Section 2.2 (recall that T_p is the minimum segment area enforced by the segmentation final refinement step). For the 640 by 480 pixels images used in the results a reasonable value is $T_p = 800$.

2. The fitting error e_i must be greater than a threshold $T_{ferr} = 0.0005$. The idea is that if this is not the case, the segment is already representing very accurately an object in the scene and there is no point in further dividing it. We consider this condition only when using fitting metrics associated to the fitting error (that is, not to surface curvature).

3. The number of recursive splits starting from the initial scene and leading to S_i must be smaller than a threshold $T_d = 10$. In other words, the depth of

the recursion tree must be smaller than T_d on the branch containing S_i (an example of the tree structure is shown in Figure 4.2).

4. A maximum number of total splits T_s must not have been reached yet. This is equivalent to setting a maximum number of possible segments. Notice that this is only an upper bound, differently from many segmentation algorithms on which the actual number of segments is decided in advance. We set $T_s = 50$ for our results.

If condition 4 is violated the procedure is stopped, while if any of conditions 1, 2, 3 does not hold, S_i is kept as part of the final segmentation and the algorithm continues on another branch of the tree. This happens also if the subdivision of S_i fails to actually create two sub-segments, since one between S_{i0} and S_{i1} is smaller than T_p and the post-processing refinement step described on Section 2.2 merges it again with the other one (this is a rare case actually).

If all the above conditions hold, S_i is split into sub-segments S_{i0} and S_{i1}, and the corresponding fitting errors e_{i0} and e_{i1} are computed. The fact that the error is improved or not with respect of the original segment is used as criterion to accept or reject the split operation. To check this, the weighted average of the fitting errors on the sub-segments is compared to the one of the original segment, that is:

$$\frac{e_{i0}|S_{i0}| + e_{i1}|S_{i1}|}{e_i|S_i|} < T_e \tag{4.1}$$

where the weights are the number of points belonging to the segments and $T_e \leq 1$ is a suitable threshold. As to the latter, in [42] we test several values (0.8, 0.9 and 1, as anticipated on Section 3.2). The experimental results, reported also in Section 4.4 of this work, show that the simple choice $T_e = 1$ (that is, just ensuring that the fitting accuracy improves even by a small amount) is the best option.

If the constraint of Eq. 4.1 is satisfied, the splitting operation improves the fitting accuracy, and we consider this as an indication that it provides a better scene representation by recognizing the different surfaces (i.e., objects). Then, we replace segment S_i with the sub-segments S_{i0} and S_{i1}. If the condition is not satisfied, the split is discarded and the algorithm moves to the next segment to process, leaving S_i unchanged.

In summary, at each step of the splitting phase the previously introduced conditions are checked for the segment with the greatest fitting error, that is:

$$\begin{aligned}(|S_i| \geq 2T_p) \wedge (|e_i| \geq T_{ferr}) \wedge (depth(S_i) < T_d) \\ \wedge (count(i) < T_s)\end{aligned} \tag{4.2}$$

where $depth(S_i)$ is the depth of S_i in the recursive tree structure, and $count(i)$ is the number of split operations made until the current iteration. If the conditions are satisfied the segment is split, and the operation is accepted if improving the fitting accuracy, discarded otherwise. Then, the next available segment with greatest fitting error is processed. The procedure is applied recursively to all the segments and the generated sub-segments, until either the maximum number of splits is reached, or there are no more available segments satisfying all the conditions.

The whole process is summarized in Algorithm 1 and a visual scheme is shown in Figure 4.1. In the pre-processing step, the scheme includes the λ_1, λ_2 parameters used to control the relative contribution of color, positions and normals information (see Section 2.2). Actually, except in [42] where we propose the region splitting method without using the normals (equivalent to setting $\lambda_2 = 0$), for our results we set $\lambda_1 = \lambda_2 = 1$, that is, we equally weight the three types of information.

The result is a tree structure like the one of Figure 4.2, where the leaves are the elements of the final segmentation. An example of the progressive subdivision into gradually smaller segments produced by this approach is shown in Figure 4.3.

Algorithm 1 Split algorithm

```
while there are still segments available to split do
  Select segment S_i with the largest fitting error
  if S_i satisfies all conditions of Eq. 4.2 then
    Split S_i into S_i0 and S_i1 (Section 2.2)
    if (|S_i0| < T_p) ∨ (|S_i1| < T_p) then
      Remove S_i from the list of segments available to split
      continue
    end if
    Fit a NURBS surface on S_i0 and S_i1 (Section 3.1)
    Compute fitting errors e_i0 and e_i1
    if fitting errors satisfy Eq. 4.1 then
      Add S_i0 and S_i1 to the list of segments available to split
    else
      Remove S_i from the list of segments available to split
      continue
    end if
  else
    Remove S_i from the list of segments available to split
    continue
  end if
  if maximum number of splits reached then
    return
  end if
end while
```

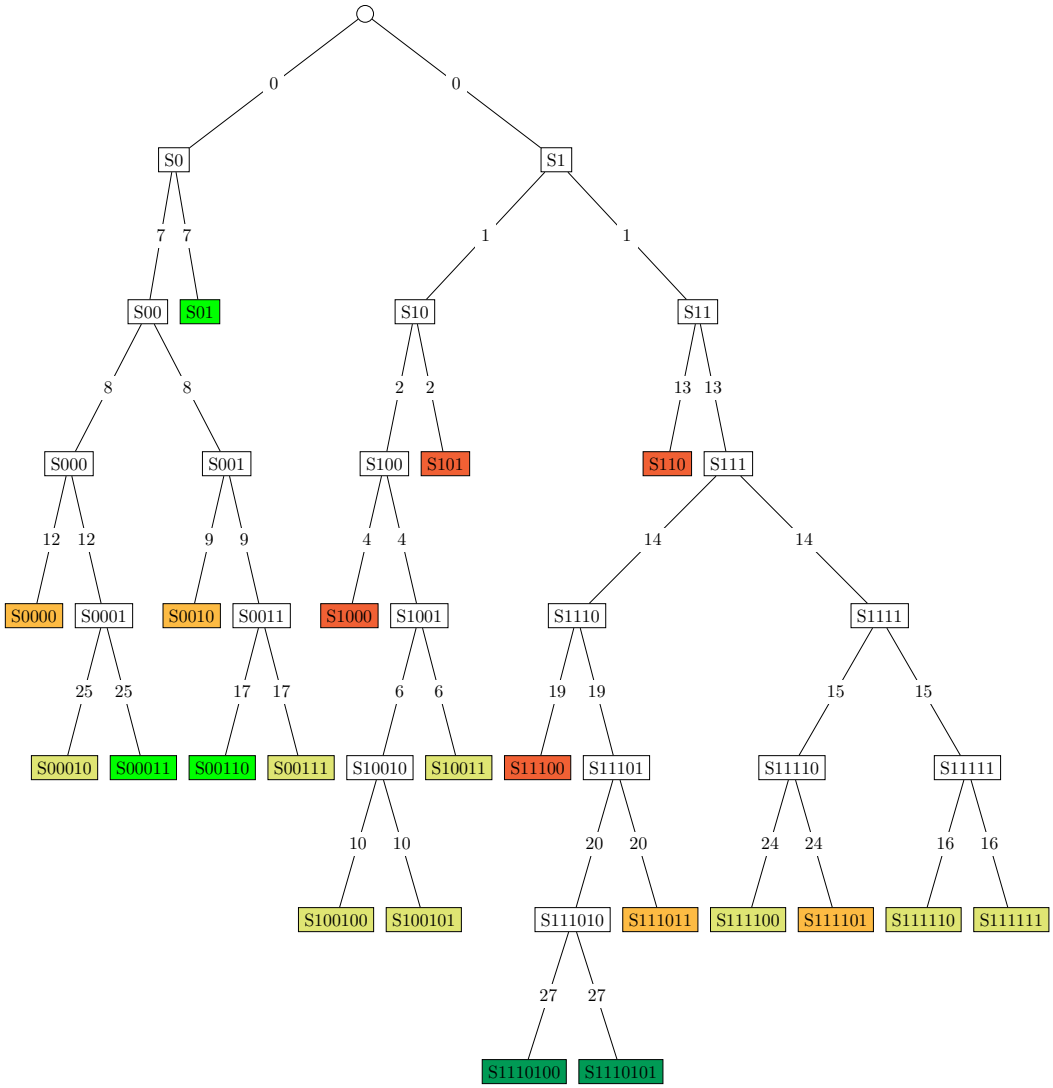

Figure 4.2: Tree structure originated by the region splitting algorithm on a sample scene. The progressive indexes of the split operations are shown on the edges. The colored nodes correspond to the final segments, and the meaning of the colors is as follows. Red ■: further segmentation was attempted but rejected since not satisfying the fitting accuracy improvement constraint. Orange ■: the segmentation was rejected since one of the resulting sub-segments would be smaller than T_p. Light green ■: not split since smaller than $2T_p$. Bright green ■: not split since fitting error smaller then T_{ferr}. Green ■: stopped since the maximum tree depth T_d was reached.

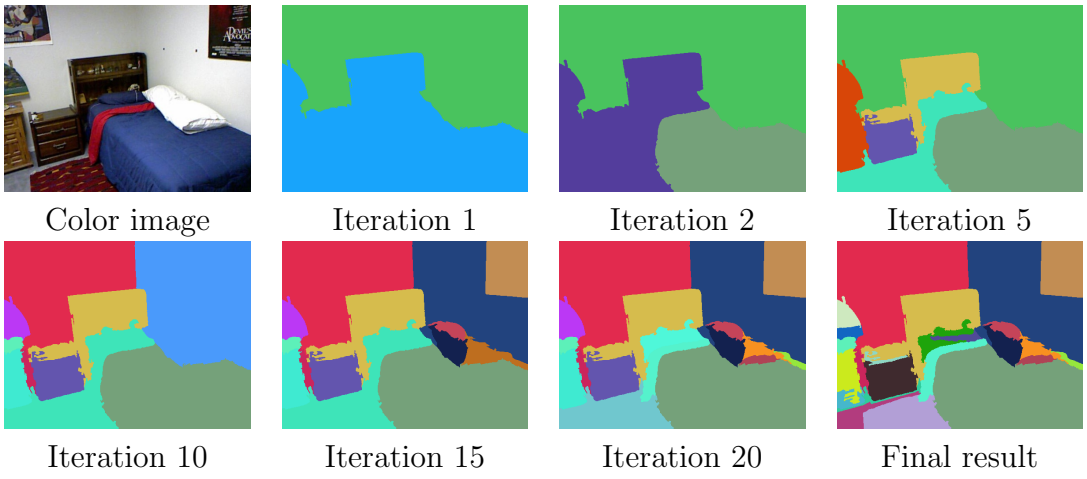

Figure 4.3: Execution of the splitting procedure on the sample scene of Figure 3.4. The images show the segmentation after 1, 2, 5, 10, 15, 20 iterations and the final result (iteration 30).

4.2 Region merging

The region splitting method of previous section is a top-down approach, that starts from the input scene and progressively refines it until a final segmentation is obtained. We develop our region merging method as a bottom-up process instead, that is, we start from an initial over-segmentation and we analyze the neighboring segments with the goal to join the adjacent ones belonging to the same surface or object in the scene. As before, the accuracy of the NURBS surface fitting is the criterion we use at the intermediate steps to decide whether to merge the neighboring segments or not.

The workflow of this merging procedure is displayed in Figure 4.4. The initial over-segmentation is obtained with the approach of Section 2.2, with the spectral clustering algorithm applied to the 9D vectors of the scene. In this preliminary step, the number of segments is chosen so that we can reasonably expect all the objects to be divided into one segment at least (we use 50 segments in our results). As to the λ_1, λ_2 parameters used to control the mutual relevance of color, positions and normals information, for our results we set $\lambda_1 = 1.5$ and $\lambda_2 = 0.5$ (that is, we weight the positions and normals data slightly more and slightly less than the color clue respectively).

Then, a NURBS surface is fitted on each segmented region S_i using the approach of Section 3.1 and the corresponding fitting errors e_i are computed. The algorithm starts by sorting all the segments in decreasing order based on the fitting error, thus producing an ordered list L_S where the segments with larger fitting errors come

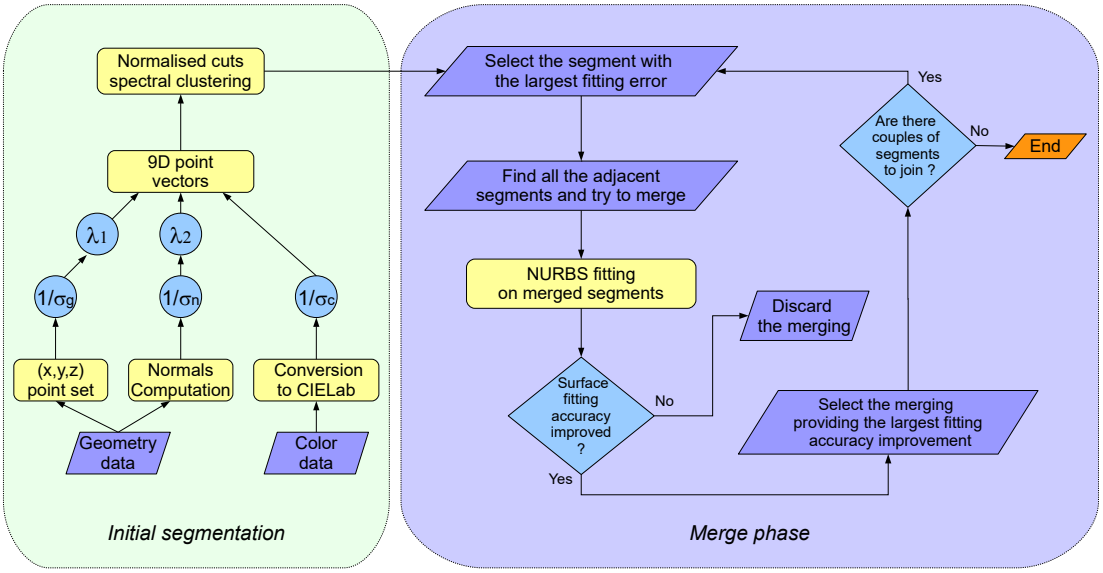

Figure 4.4: Overview of the region merging approach.

first. It also analyzes all the segments and builds an adjacency matrix, storing for each couple of segments whether they are *adjacent* or not. Two segments are considered as *adjacent* if they satisfy the following conditions:

1. They must be connected on the depth map lattice (using 4-connectivity) and the length l_{cc} of the shared boundary C_C (highlighted in red in the example of Figure 4.5) must be bigger than a threshold T_l. For the 640 by 480 pixels images used in the results a reasonable value is $T_l = 15$.

2. The depth values on the shared boundary must be similar. In order to check this, we compute the difference ΔZ_i between the depth values on the two sides of the edge for each point P_i in the common boundary C_C (in the example of Figure 4.5 the orange arrows show which differences are considered). The number of points l_{cc}^d in the shared boundary with a depth difference smaller than a threshold $T_z = 0.2\,m$ is then computed. The ratio between l_{cc}^d and the total length of the shared boundary must be greater than a threshold R (the same value is used also in the following Equations 4.4 and 4.5 and we set it to 0.6), that is:

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta Z_i \leq T_z)|}{|P_i : P_i \in C_C|} = \frac{l_{cc}^d}{l_{cc}} > R\,. \tag{4.3}$$

3. Also the color values must be similar on both sides of the common contour. We check this as for the depth values, except for the fact that we consider the difference ΔC_i between the values in the CIELab color space. Namely,

we compute the color difference ΔC_i between the samples on both side of the shared boundary, we determine the number of points l^c_{cc} for which this difference is smaller than a threshold $T_c = 6$, and we require the ratio between l^c_{cc} and the total length to be greater than R, that is:

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta C_i \leq T_c)|}{|P_i : P_i \in C_C|} = \frac{l^c_{cc}}{l_{cc}} > R\,. \tag{4.4}$$

4. The same condition is required also for normal information. That is, the angle between the two normal vectors $\Delta\theta_i$ is computed for each couple of samples on the two sides of the shared boundary. The number of points l^n_{cc} for which the angle between the normal vectors is smaller than a threshold $T_\theta = 5°$ is computed, and again the ratio between l^n_{cc} and the total length must be greater than R, that is:

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta\theta_i \leq T_\theta)|}{|P_i : P_i \in C_C|} = \frac{l^n_{cc}}{l_{cc}} > R\,. \tag{4.5}$$

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

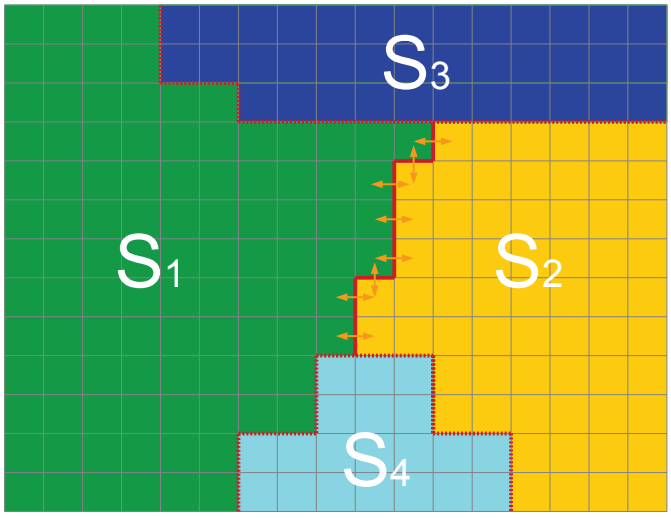

Figure 4.5: Example of boundary region with the common contour between two sample segments S_1 and S_2. The arrows show along which edges of the contour the differences used in Equations 4.3, 4.4, 4.5 are calculated.

The procedure then selects the segment with the largest fitting error and tries

Algorithm 2 Merge algorithm

```
Compute L_S (list of segments) and sort it according to e_i
For each segment S_i compute the set A_i of adjacent segments
i = 1 (select as S_i the first segment in L_S)
while i < length(L_S) do
  for all segments S_j adjacent to S_i do
    compute fitting error on merged segment S_{i∪j}
    check if the condition of Eq. 4.6 is satisfied
  end for
  if at least one merge operation satisfies Eq. 4.6 then
    Select the merge operation leading to best fitting accuracy improvement
    (the corresponding segment is S_j^*)
    Remove S_i and S_j^* from L_S
    Add S_{i∪j^*} to L_S
    Compute A_{i∪j^*}
    i = 1 ( S_i is the first segment in L_S)
  else
    i = i + 1 (S_i is the next segment in L_S)
  end if
end while
```

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

$$\frac{e_i|S_i| + e_j|S_j|}{e_{i\cup j}(|S_i| + |S_j|)} > 1\,. \tag{4.6}$$

Notice that this condition is the reverse of Eq. 4.1 used in the splitting method, since it requires that the weighted fitting accuracy is better on the union than on the two separate parts (the opposite is requested in the splitting scheme).

If the fitting accuracy improves, that is, the condition of Eq. 4.6 is satisfied, the merging operation between S_i and S_j is candidate to be accepted, otherwise it is discarded. The procedure is repeated for all the segments adjacent to S_i and, among the ones for which the merging operation would improve the fitting error (if any), the segment S_j^* that provides the maximum improvement according to Eq. 4.6 is selected. The two segments S_i and S_j^* are then merged, and the list L_S is updated by removing them and inserting their union $S_{i\cup j^*}$ in the position corresponding to its fitting error $e_{i\cup j^*}$. The adjacency information is also updated

by considering $S_{i \cup j^*}$ adjacent to all the segments that were adjacent to either S_i or S_j^*. In case instead there are no merging operations with S_i that would improve the fitting error, the algorithm just moves to the next segment in L_S.

After selecting the next segment with the greatest fitting error, the algorithm continues as above and it iterates until no more segments can be considered for a merge operation. The procedure is summarized in Algorithm 2 and its progress on a sample scene is displayed in Fig. 4.6, where a graph representing the various merge operations and the resulting segmentations at several iterations are shown.

Figure 4.6: Example of the merging procedure on the scene of Figure 4.9, sixth row. The images show the initial over-segmentation, the merging output after 8, 16, 24, 32 iterations and the final result (iteration 41). The graph shows the merge operations between the various segments. The colors in the images correspond to those of the graph nodes.

4.3 Combined region splitting and merging

The region splitting and region merging schemes of the previous sections are both capable to provide good segmentation results, as will be discussed in Section 4.4. However, with the recursive splitting algorithm alone it is sometimes

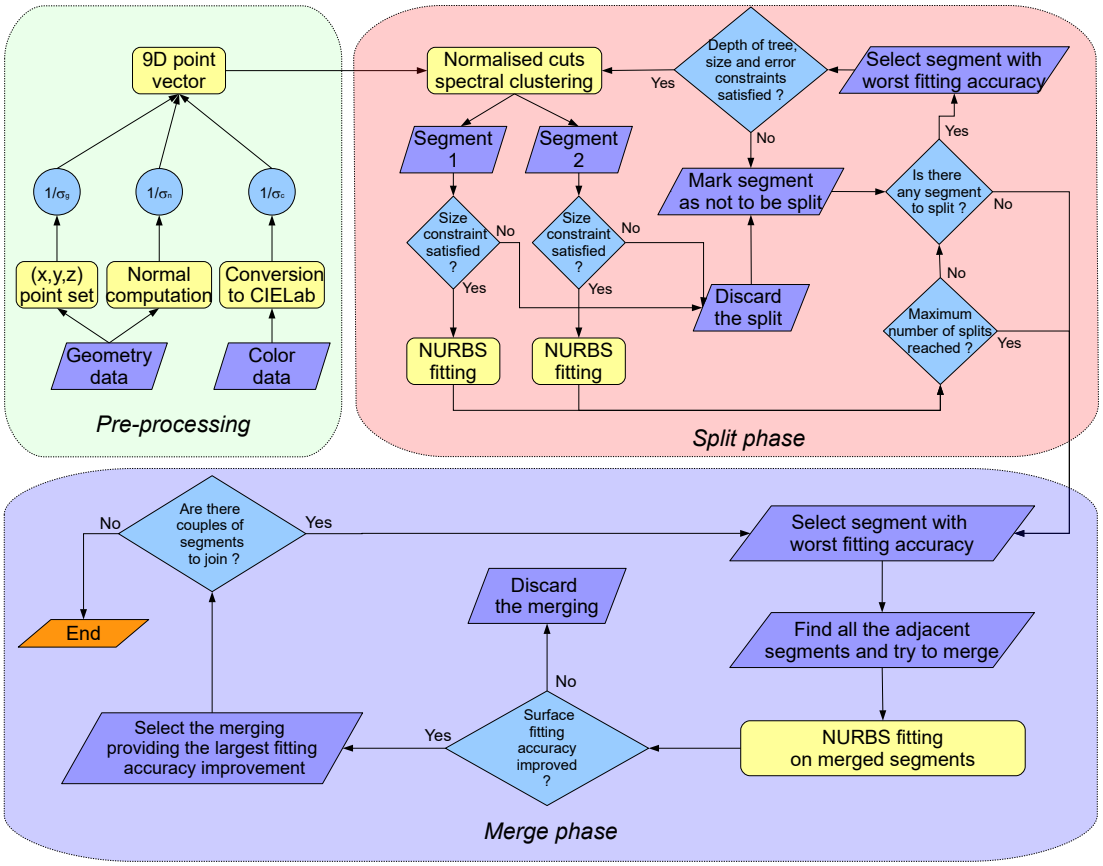

Figure 4.7: Overview of the combined region splitting and merging approach.

difficult to balance the trade-off between an over-segmentation of the scene and the recognition of all the objects and structures in it. In particular, in some splitting steps the binary segmentation can mistakenly divide a single object due to misleading clues. If this happens, a final merging stage recombining the multiple segments corresponding to the same scene object would improve the final result. On the other hand, the region merging scheme is performed on a generic over-segmentation, that does not take the surface fitting results into account. Using the surface fitting approach also in this preliminary step is more consistent, and can provide an improvement in the final segmentation.

Based on these premises, we propose an additional segmentation method by combining the region splitting and the region merging ones. That is, we apply a variation of the region splitting scheme to obtain the initial over-segmentation that is used as input for the region merging procedure. An overview of this combined approach is shown in Figure 4.7. We refer to Section 4.1 and to Section 4.2 for the details about the two single methods, and we describe in the following the modifications we make in order to use them in the combined procedure.

Regarding the region splitting phase, an important point is that we now aim to

obtain an over-segmentation rather than the final result. To achieve this, we drop the condition of Eq. 4.1, thus accepting a split operation even if not improving the surface fitting accuracy. Notice that the fitting results still remain relevant in the process, since we keep the fact that at each step the segment that gets processed is the one with the largest fitting error. With this modification, the iterative procedure keeps subdividing the obtained regions, until either they have all become too small to be further split, or one of thresholds T_d (maximum split tree depth) or T_s (maximum number of segments) have been reached. About the latter, notice that the $T_s = 50$ value we set is consistent with the choice of 50 regions in the initial over-segmentation provided by the normalized cuts algorithm for the region merging method alone.

As to the merging phase, in the combined approach we simplify the conditions required on the neighboring segments to be considered as adjacent, by using the value 0.5 (instead of 0.6) for the threshold R of Section 4.2. The Equations 4.3, 4.4 and 4.5 then become:

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta Z_i \leq T_z)|}{|P_i : P_i \in C_C|} > 0.5\,, \tag{4.7}$$

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta C_i \leq T_c)|}{|P_i : P_i \in C_C|} > 0.5\,, \tag{4.8}$$

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta \theta_i \leq T_\theta)|}{|P_i : P_i \in C_C|} > 0.5\,, \tag{4.9}$$

that is, the compatibility conditions regarding the depth, color and orientation values are requested along at least half (instead of 60%) of the common boundary. To balance this, in our experimental results we use the slightly tighter threshold values $T_z = 0.15\,m$, $T_c = 5$ and $T_\theta = 2°$ (instead of $T_z = 0.2\,m$, $T_c = 6$ and $T_\theta = 5°$ used for the region merging scheme alone). Regarding the other steps and parameters of the algorithm we do not make any other change.

As will be discussed in Section 4.4, this combined scheme obtains improved results with respect to each of the two single methods alone. The progress of both the splitting and the merging phase on a sample scene is shown in Figure 4.8. On that one, it is visible how the merging iterations are successful in recombining portions of regions that were mistakenly subdivided during the splitting phase (e.g., the background wall or the person).

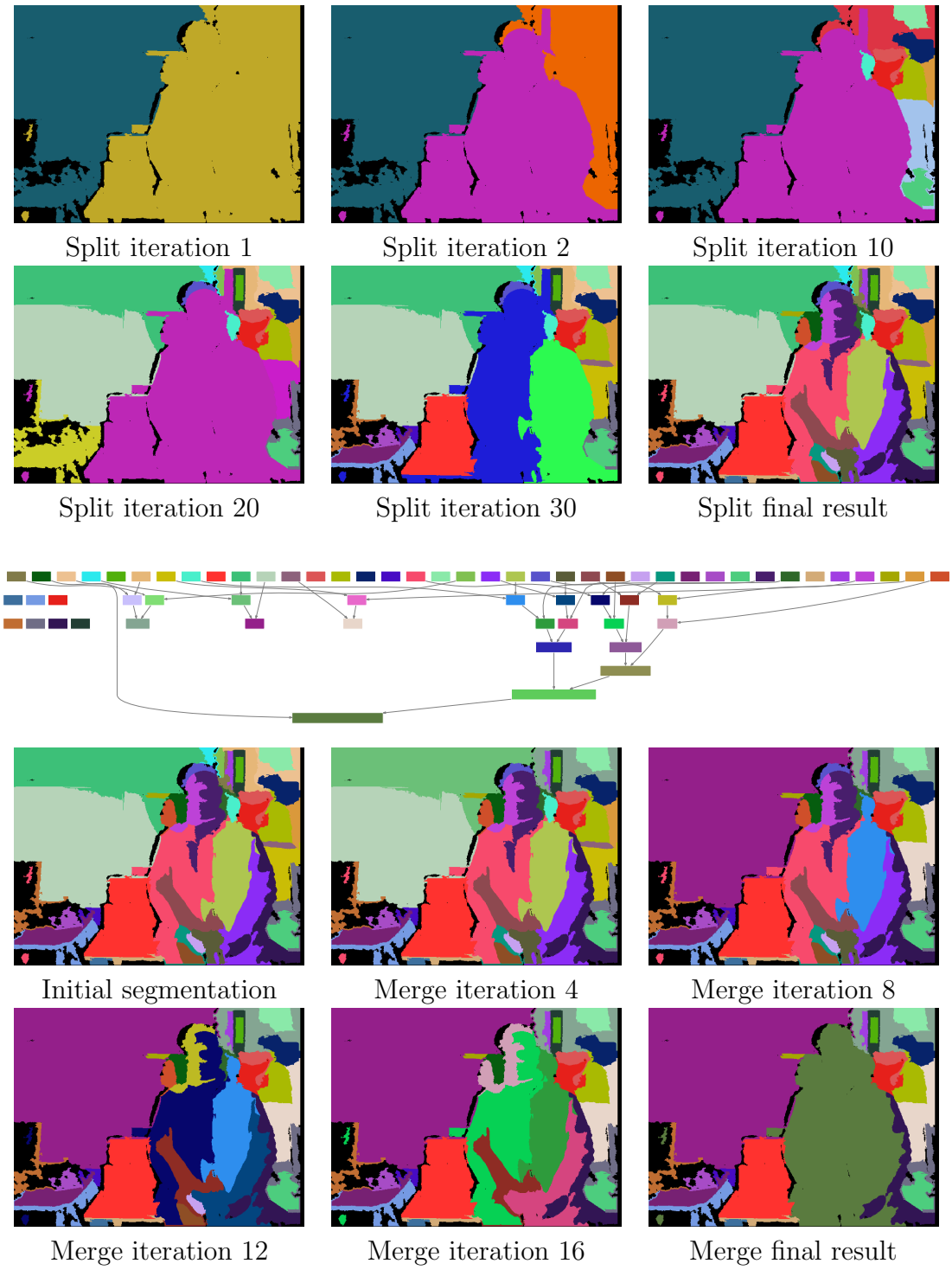

Figure 4.8: Combined splitting and merging procedure on the scene of Figure 4.9, sixth row. The first group of images show the segmentation after 1, 2, 10, 20, 30 iterations of the splitting phase and the final over-segmentation (iteration 50), which is used as input for the merging phase (first image in the second group). The other images in the second group show the merging output after 4, 8, 12, 16 iterations and the final result (iteration 21). The graph in the middle shows the merge operations between the various segments (in the second group of images the colors correspond to those of the graph nodes).

4.4 Experimental results

We perform an experimental evaluation of the proposed methods by first analyzing their results on a small sample dataset, then by comparing them against state-of-the-art approaches on the large and challenging NYU Depth Dataset V2 (NYUv2) [49].

Results on the LTTM dataset

The first and simpler dataset was acquired at the LTTM laboratory [43] and is available at `http://lttm.dei.unipd.it/downloads/segmentation`. It contains 6 different images with the corresponding depth maps acquired by PrimeSense devices (i.e., Kinect v1 and Xtion sensors). Ground truth information obtained by manual segmentation is also available. It is a small dataset but contains a large variety of different structures and objects with varying shapes, colors and properties, then it is a good starting point to evaluate the various proposed approaches. It also represents a completely different environment from the NYUv2 dataset, then it is useful to ensure that our methods have not been over-fitted on that dataset.

We present the segmentation results obtained with the approaches of [11], [42], [41] and with the proposed combined region splitting and merging method (referred to as Split+Merge in the following). This is an interesting test to compare our approaches and also understand the relevance of the various components, since the first cited work directly segments the scene into the desired number of regions with an approach based on spectral clustering, that can be considered as a simplified version of the segmentation scheme of Section 2.2. The second one exploits the region splitting scheme of Section 4.1, simplified to not consider the orientation information, while the third one leverages the region merging approach of Section 4.2.

The visual results are shown in Figure 4.9, and Table 4.1 lists the numerical values obtained by comparing the results and the ground truth. Starting from the visual results, on Figure 4.9 it is clear how the Split+Merge method obtains better performances than the compared approaches. The approach of [11] tends to over-segment the considered scenes and has some difficulties in properly capturing all the objects. In particular the background is completely wrong in some scenes (e.g., 5 and 6), since the geometry dependent term combined with the bias towards segments of similar size of the normalized cuts algorithm [48] forces the large regions to be divided into multiple pieces. Notice how in our approaches the merging scheme solves this problem by recombining together segments belonging

to the same surface. The approach of [42] produces less segments but is not able to recognize all the objects (e.g., in scenes 4 and 5). Notice in particular how this splitting approach alone can hardly recognize all the scene objects and at the same time avoid over-segmentation, while the Split+Merge method can perform a larger number of splitting steps without affecting the final result since over-segmentation issues get fixed in the merging phase. Since the approach in [42] does not exploit orientation information, to fully evaluate the performances of the splitting algorithm we show also the results of a modified implementation that takes normals into account. This version then exactly corresponds to the splitting scheme of Section 4.1. It can be noticed that the orientation information allows some surfaces to be better captured, e.g., the background in scene 2 and the table in scene 5. The region merging scheme [41] is the one that obtains results closer to the Split+Merge approach. Both the methods avoid the creation of small segments caused by noise or by complex surfaces, and at the same time they properly extract most of the structures in the scene. Moreover, the use of orientation information makes the various walls and the surfaces with different orientation properly recognized (e.g., the table in row 3). However, notice how the background (especially in scenes 1 and 2) is properly captured only by the Split+Merge approach, while [41] fails to correctly recognize the various background surfaces. In general in the Split+Merge approach the objects are well recognized, and there are very few segments extending over separate objects at different depths.

The visual evaluation is confirmed by the numerical results, as shown by Table 4.1. In order to compare the results with ground truth data we use two different metrics, the Variation of Information (VoI) and the Rand Index (RI). A description of these metrics can be found in [1], in particular notice that lower values correspond to better results for the VoI metric while higher values are better for the RI metric. The table lists the average values of the two metrics on the six considered scenes. It shows that according to both metrics the Split+Merge approach outperforms all the compared ones. The VoI metric value is better by a large gap with respect to [11] and [42]. The merging approach has closer performances, as already noticed also on the visual results, but the Split+Merge scheme is able to outperform it anyway with an average VoI score of 1.69 against 1.74. The behavior is similar for the RI metric, with which the Split+Merge approach is the best one with a score of 0.91. Notice that the approaches exploiting orientation information provide in general better performances.

Approach	*VoI*	*RI*
(Clustering) [11]	2.71	0.81
(Split) [42]	2.01	0.83
(Split+normals) [42]	1.92	0.88
(Merge) [41]	1.74	0.88
(Split+Merge)	**1.66**	**0.91**

Table 4.1: Comparison of the performances of the proposed Split+Merge method with [11, 42, 41]. The table shows the average value of the VoI and RI metrics on the six scenes of the LTTM dataset.

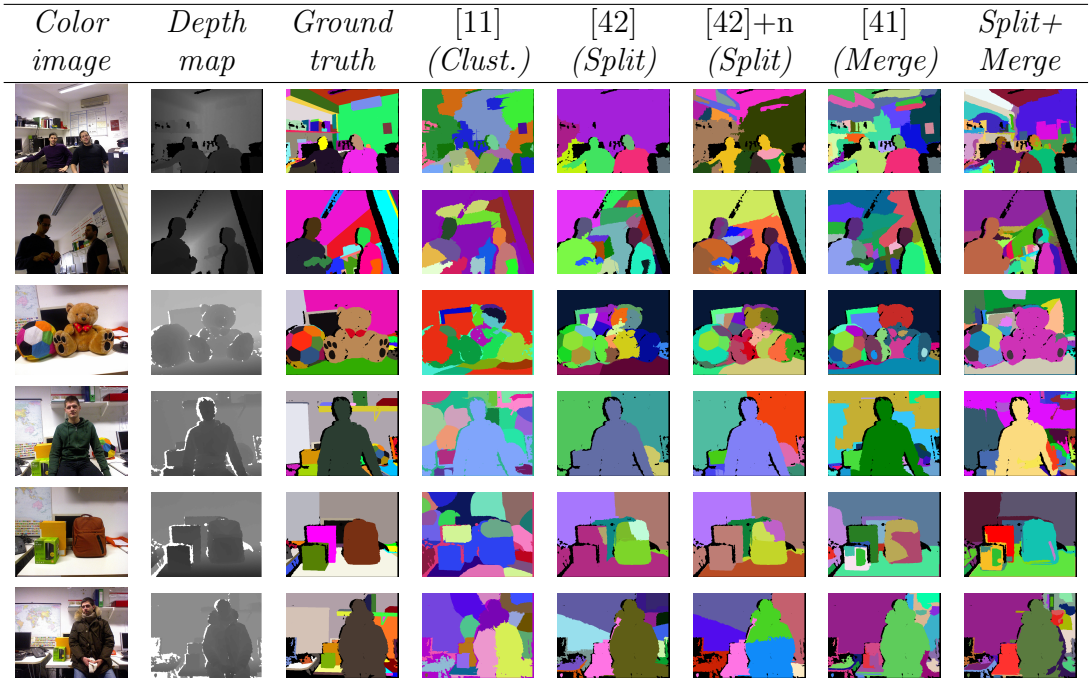

Figure 4.9: Segmentation of some sample scenes with the proposed Split+Merge method and with the approaches of [11], [42] and [41]. The black regions in the results correspond to samples without a valid depth value from the Kinect that are not considered for the segmentation.

Fitting metrics comparison

We report also some results about the various fitting metrics defined in Section 3.2. In [42] we apply the region splitting method (without using orientation information) to the six scenes of the LTTM dataset, and we compare the segmentation performances obtained with each of the available fitting metrics. We test also the values 0.8, 0.9 and 1 for the threshold T_e of Eq. 3.5 (recall that T_e determines the required amount of improvement on the fitting error of two subsegments to accept a subdivision operation in the splitting scheme).

Table 4.2 shows the accuracy of the obtained segmentations according to the RI metric. A first basic result is that on average error based metrics produce better results than curvature based approaches, even if this is not true for all the metrics and all the scenes. Notice in particular that none of the metrics is the best on all the considered scenes. The number of points with a large error (NLE) provides the best average results. The MSE and MAE metrics both provide good results (they behave similarly, as expected). Curvature based metrics provide on average slightly lower performances. The best one among them is the NLC (number of points with a large curvature) metric, that is the best on scene 4 and has average performances similar to the MSE and MAE. The other curvature metrics have lower scores according to this metric, but not too far from the error based ones. Finally notice that according to the RI measure the choice $T_e = 1$, equivalent to simply ensuring that the metric values improve even by a very small amount, is the best option for all metrics except VMEC.

Table 4.3 shows the results according to the VoI metric. MSE with $T = 1$ is the best option, even if NLE gets very close to the MSE score. MAE and VE also achieve good results. Again, curvature based approaches have low performances, with the exception of NLC, that is the only curvature based approach with close results to those of the error based approaches.

Given these results, we consider the MSE metric with $T_e = 1$ as the most adequate choice in general. All the results of our approaches listed in this section are obtained with these options.

Regarding this, notice that Table 4.1 reports VoI and RI scores of 2.01 and 0.83 respectively, instead of 1.90 and 0.85 as in Tables 4.2 and 4.3 (see the MSE, $T_e = 1$ case). This is because the values in [42] are calculated considering the black regions made of invalid points (i.e., not acquired by the sensor) as valid segments both on the segmentation and on the ground truth, thus overestimating the segmentation accuracy. Table 4.1 lists the correct values instead, obtained with the invalid points ignored.

Metric	T_e	Scene						Mean
		1	2	3	4	5	6	
MSE	**1**	0.70	0.90	0.89	0.84	0.87	0.91	0.85
	0.9	0.67	0.90	0.60	0.78	0.83	0.90	0.78
	0.8	0.67	0.84	0.60	0.75	0.84	0.86	0.76
MAE	**1**	0.67	0.90	**0.89**	0.84	0.88	0.92	0.85
	0.9	0.67	0.84	0.60	0.77	0.82	0.85	0.76
	0.8	0.67	0.79	0.60	0.77	0.82	0.81	0.75
VE	**1**	**0.90**	0.91	0.60	0.84	**0.89**	0.85	0.83
	0.9	0.70	**0.91**	0.60	0.75	0.83	0.90	0.78
	0.8	0.67	0.90	0.60	0.75	0.84	0.86	0.77
NLE	**1**	0.87	0.90	0.82	0.78	0.87	**0.92**	**0.86**
	0.9	0.67	0.89	0.60	0.78	0.84	0.91	0.78
	0.8	0.67	0.89	0.60	0.78	0.84	0.81	0.77
VMXC	**1**	0.79	0.79	0.86	0.86	0.72	0.86	0.81
	0.9	0.79	0.79	0.87	0.86	0.70	0.86	0.81
	0.8	0.79	0.79	0.84	0.86	0.70	0.85	0.81
VMEC	1	0.79	0.79	0.89	0.87	0.72	0.87	0.82
	0.9	0.79	0.79	0.88	0.86	0.74	0.87	0.82
	0.8	0.63	0.79	0.88	0.86	0.74	0.87	0.79
VGC	**1**	0.79	0.85	0.85	0.87	0.70	0.85	0.82
	0.9	0.79	0.85	0.85	0.87	0.70	0.85	0.82
	0.8	0.79	0.85	0.85	0.86	0.70	0.85	0.82
MMXC	**1**	0.79	0.79	0.89	0.87	0.72	0.81	0.81
	0.9	0.63	0.79	0.60	0.78	0.73	0.82	0.72
	0.8	0.63	0.79	0.60	0.78	0.70	0.81	0.72
NLC	**1**	0.86	0.79	0.87	**0.87**	0.88	0.81	0.85
	0.9	0.70	0.79	0.86	0.87	0.75	0.82	0.80
	0.8	0.70	0.79	0.60	0.78	0.70	0.70	0.71

Table 4.2: Results of the region splitting method [42] according to the RI metric (higher is better), with different fitting metrics and T_e values used. The T_e value giving the best result for each metric and the best results on each scene and on average are marked in bold.

Metric	T_e	Scene						Mean
		1	2	3	4	5	6	
MSE	**1**	2.42	1.79	1.99	1.73	1.68	1.77	**1.90**
	0.9	2.79	1.39	2.58	1.60	1.67	1.79	1.97
	0.8	2.79	1.33	2.58	1.80	1.53	1.98	2.00
MAE	1	2.79	1.66	**1.88**	1.72	2.69	1.68	2.07
	0.9	2.79	**1.30**	2.68	1.61	1.76	1.82	1.99
	0.8	2.79	1.72	2.58	1.61	1.58	2.02	2.05
VE	1	2.39	1.79	2.58	1.73	1.61	2.64	2.12
	0.9	2.42	1.40	2.58	1.80	1.65	1.78	1.94
	0.8	2.79	1.39	2.58	1.80	1.53	1.98	2.01
NLE	**1**	2.84	1.55	2.23	1.60	1.54	**1.67**	1.91
	0.9	2.79	1.51	2.58	1.60	**1.53**	1.75	1.96
	0.8	2.79	1.51	2.58	1.60	**1.53**	2.02	2.00
VMXC	1	2.73	1.72	2.60	**1.59**	2.49	2.49	2.27
	0.9	2.73	1.72	2.52	**1.59**	1.91	2.46	2.16
	0.8	2.73	1.72	2.58	**1.59**	1.91	2.03	2.09
VMEC	1	2.73	1.72	2.13	1.78	2.38	2.43	2.19
	0.9	2.73	1.72	2.18	1.71	2.12	2.43	2.15
	0.8	3.03	1.72	2.10	1.59	2.12	2.14	2.12
VGC	1	2.73	1.85	2.53	1.79	1.92	2.56	2.23
	0.9	2.73	1.85	2.53	1.79	1.92	2.57	2.23
	0.8	2.73	1.85	2.53	1.71	1.92	2.37	2.19
MMXC	**1**	2.73	1.72	2.06	1.78	2.38	2.26	2.15
	0.9	3.03	1.72	2.68	1.63	2.29	2.20	2.26
	0.8	3.03	1.72	2.68	1.63	1.92	1.98	2.16
NLC	**1**	**2.19**	1.67	2.20	1.75	2.04	2.35	2.03
	0.9	2.49	1.67	2.19	1.78	2.04	2.20	2.06
	0.8	2.49	1.67	2.68	1.63	1.91	2.37	2.13

Table 4.3: Results of the region splitting method [42] according to the VoI metric (lower is better), with different fitting metrics and T_e values used. The T_e value giving the best result for each metric and the best results on each scene and on average are marked in bold.

Results on the NYUv2 dataset

We perform the main experimental evaluation of our proposed approaches on the NYU Depth Dataset V2 [49], that is much larger and thus more interesting. Since it has been widely used for the evaluation of joint color and depth segmentation algorithms, it allows the comparison of our proposed schemes with the state-of-the-art methods for this task. The dataset was acquired with a Kinect and contains 1449 depth and color frames from a variety of indoor scenes. Ground truth data is also available. Since on the original ground truth data there are sometimes missing values in proximity of edges, for our numerical evaluation we use the updated versions of the ground truth labels provided by the authors of [23]. Table 4.4 shows the comparison between our Split+Merge method, our region splitting and region merging schemes, and some state-of-the-art approaches from the literature (for some competing methods we use the results collected by [28]). The compared state-of-the art approaches are the clustering and region merging method of [28], including the recent improved version [27], the MRF scene labeling scheme of [45], a modified version of [18] that accounts also for geometry information, the dynamic programming scheme of [52] and the multi-layer clustering strategy of [34]. The average values obtained by our Split+Merge method are 2.17 as to the VoI metric and 0.89 as to RI. The VoI metric results show that our approach outperforms all the compared ones and the improvement over most of them is relevant. The two approaches that get performances closer to the proposed one are [27] and our merging scheme [41], with VoI values of 2.20 and 2.23 respectively compared to 2.17 (recall that for VoI smaller is better). If the RI metric is considered, the proposed Split+Merge method outperforms most compared approaches and obtains results very similar to those of the state-of-the-art methods of [45] and [27], with a small difference of just 0.01 and 0.02 respectively. Notice also that our approach does not make any assumption about the presence of planar surfaces differently from [28], [27] and [52], so it is expected to better generalizes to scenes with non-planar surfaces (in the NYUv2 dataset all the scenes are indoor settings with many planar surfaces like walls and furniture, while outdoor settings have a larger variability). In addition the method of [45] exploits a learning stage, while our proposed approaches do not assume any previous knowledge on the data.

A visual comparison on 7 different scenes from this dataset is shown in Figure 4.10. Notice that the scenes have been selected by the authors of [28]. In the figure we show the results of our Split+Merge method only, since it is the best performing among our proposed ones. Even if this dataset is more challenging, the proposed Split+Merge approach provides a reliable segmentation on all the

considered scenes as shown in the last column of Figure 4.10. The obtained segmentations are clearly better than the approaches of [18], [11] and [52] (columns 6-7-8). The comparison with the two best performing approaches, that is, [28] and [45], is more challenging and there is not a clear winner. The various objects are properly extracted by our approach and the background region is correctly handled on most scenes. In addition, our method does not produce noisy small segments in proximity of edges, an issue happening with other approaches on some scenes. Some small errors are present, e.g., in the corridor and bed scenes (rows 2 and 3). In particular the blanket of the bed scene (row 3) is quite critical since color data is very noisy and the normals on the rough surface are very unstable.

The total computation time is comparable to that of a direct segmentation with normalized cuts, since on most iterations both the spectral clustering and the surface fitting algorithms are applied to small subsets of the scene (the surface fitting actually adds just a small overhead to the segmentation time).

Approach	*VoI*	*RI*
Felzenszwalb et al. [18]	2.32	0.81
Ren et al. [45]	2.35	0.90
Taylor et al. [52]	3.15	0.85
Hasnat et al. (JCSA) [28]	2.72	0.87
Hasnat et al. (JCSA-RM) [28]	2.29	0.90
Hasnat et al. (JCSD-RM) [27]	2.20	**0.91**
Khan et al. [34]	2.42	0.87
(Clustering) [11]	3.09	0.84
(Split) [42]	2.62	0.75
(Split+normals) [42]	2.52	0.76
(Merge) [41]	2.23	0.88
(Split+Merge)	**2.17**	0.89

Table 4.4: Performances of the proposed Split+Merge method, of some state-of-the-art approaches (first block) and of approaches corresponding to the sub-components of the Split+Merge method (second block). The table shows the average values of the VoI and RI metrics on the NYUv2 dataset.

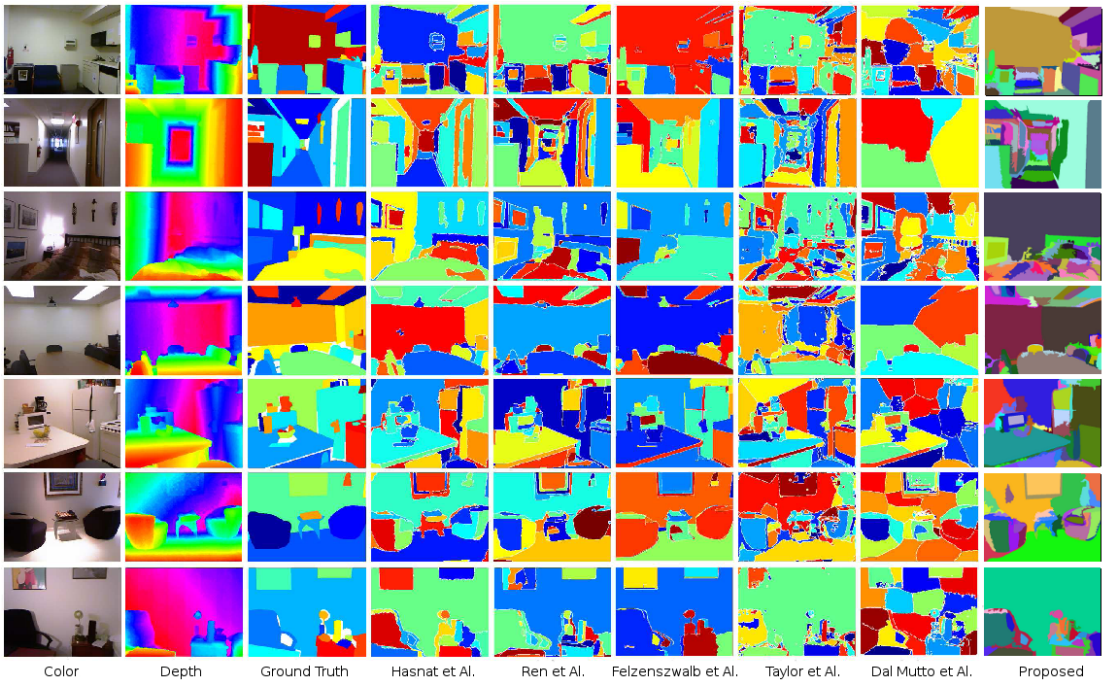

Figure 4.10: Segmentation of some sample scenes from the NYUv2 dataset: (column 1) color data; (column 2) depth data; (column 3) ground truth; (column 4) [28]; (column 5) [45]; (column 6) [18]; (column 7) [52]; (column 8) [11]; (column 9) proposed Split+Merge method. The results for the competing methods have been collected from [28].

Chapter 5

Segmentation schemes based on deep learning and surface fitting

In this chapter we show how a classification step based on Convolutional Neural Networks (CNNs) can be exploited to improve the results of one of the iterative segmentation methods of previous chapter, the region merging scheme. We also point out that the resulting algorithm is effective for the semantic labeling problem as well. Section 5.1 describes the architecture of the CNNs used for the preliminary classification step. In Section 5.2 we present a variant of the region merging algorithm of previous chapter, revised in order to take advantage of the classification returned by the CNN, and also to improve it. Finally, in Section 5.3 we discuss the experimental results both for the objects segmentation problem and for the semantic labeling task.

5.1 Classification with deep learning

In order to improve the region merging algorithm presented in Section 4.2, we introduce additional clues besides the NURBS surface fitting accuracy to control the merging operations. For this, we employ a machine learning stage that returns classification data for the input scene, which can be used not only to produce semantic labels but also to decide which regions of the initial 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. For this, a descriptor vector is associated to each pixel by considering the final layer of the network, a standard soft-max classifier. This information is used to compute a similarity score between couples of adjacent

segments and, at the same time, to provide the input image with semantic labels. The similarity score is the information that we use to control the iterations of the revised region merging scheme, as described in next Section 5.2 (the similarity data is employed both to decide whether two adjacent segments should be merged, and to determine in which order the couples of segments are selected to attempt a merge operation).

To test different possible approaches, we develop two separate structures of Convolutional Neural Networks. We refer to them as CNN1 and CNN2 to disambiguate in the following. In the first one, presented in [40], both color and normals information are used as input. First, the normalized color and normal components computed in Section 2.2 are combined into 6D vectors representing each point of the scene as

$$\boldsymbol{p}_k^{cn} = \left[\bar{L}(p_k), \bar{a}(p_k), \bar{b}(p_k), \bar{n}_x(p_k), \bar{n}_y(p_k), \bar{n}_z(p_k)\right], \quad k = 1, \ldots, n_p\,. \tag{5.1}$$

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

A multi-scale architecture [17] is then used to achieve a greater expressiveness without increasing the number of network parameters. Each image is passed as input to the network at the three different scales 320×240, 160×120 and 80×60, both to account for the varying size at which similar objects may appear in the scene, and to take advantage of increasingly larger contexts. The structure of the network is shown in Figure 5.1.

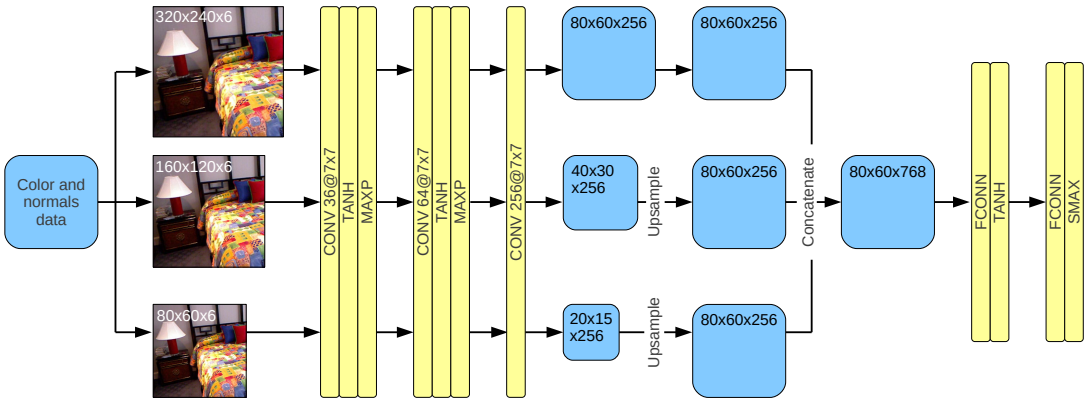

Figure 5.1: Layout of the proposed Convolutional Neural Network CNN1.

Similarly as in [17, 8], the network can be divided in two parts. In the first part a local representation of the input is extracted by applying a sequence of convolutional layers sharing their weights across the three scales. More in detail, the three input scales are feed-forwarded through three convolutional layers (denoted

with CONV in Figure 5.1). The first two convolutional layers are followed by a hyperbolic tangent activation function (TANH) and a max-pooling (MAXP) layer, while the third one is applied as a simple bank of linear filters, producing the three outputs corresponding to the three scales. The outputs are then upsampled and concatenated to provide a vector of feature descriptors for each pixel. The second part of the network is composed by two fully-connected layers (FCONN), with hyperbolic tangent and soft-max (SMAX) activation functions respectively.

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

The network is trained to produce a semantic segmentation of the input image by labeling each pixel in the scene with one among 14 different semantic labels. To this purpose, a multi-class cross-entropy loss function is minimized throughout the training process. For this task, we obtain a suitable ground truth from the one provided in [23] by clustering the original 894 categories of the NYUv2 dataset [49] into 14 classes, as proposed in [8].

As in [17] we split the training process into two separate steps. The filter weights of the three convolutional layers are first trained separately by applying a simple linear classifier to the output of the first part of the network, with soft-max activation and multi-class cross-entropy loss function. Next, the weights and biases of the last two fully-connected layers are trained while keeping fixed the convolutional weights as calculated in the previous step. Again, the multi-class cross-entropy loss function is minimized.

The output of the soft-max activation function in the last fully-convolutional provides the final predicted labels. In addition, as detailed later for both the CNNs considered in our approach, it provides the descriptors that we use to define the similarity measure between any two segments.

The second Convolutional Neural Network (CNN2) we propose is built from the approach of the first one and of [17, 8] with several modifications. It does not adopt any multi-scale input representation and it receives various clues as input, namely:

- Color data, represented by the 3 components in the RGB color space.
- Geometric information, given by three channels containing for each sample the horizontal disparity h_1, the height above the floor h_2, and the angle a be-

tween the normal vector and the vertical direction. With this representation, introduced in [25] and denoted as HHA, the classification results are usually better than those obtained by directly using the positions and orientation information.

- Surface fitting information, represented by three components containing the fitting error e and the two principal curvatures κ_1 and κ_2 of the NURBS approximating surfaces (by considering these data we assume that NURBS surface fitting has been performed on the input over-segmentation).

By combining these different types of information we obtain 9D vectors that represent the points of the scene as

$$\boldsymbol{p}_k^{cgf} = [R(p_k), G(p_k), B(p_k), h_1(p_k), h_2(p_k), a(p_k), e(p_k), \kappa_1(p_k), \kappa_2(p_k)] \ , \quad k = 1, \ldots, n_p \, . \tag{5.2}$$

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

A sequence of convolutional layers is then applied in order to extract a local representation of the input. The architecture of the employed network is shown in Figure 5.2.

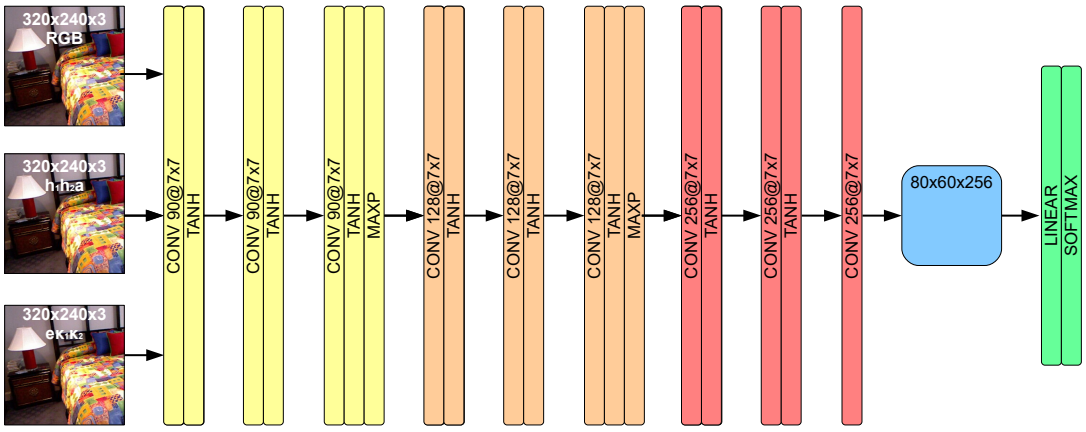

Figure 5.2: Layout of the proposed Convolutional Neural Network CNN2.

More in detail, each 9-channel input image is processed through nine convolutional layers, combined into three main blocks, each one containing three layers. Every block contains three convolutional layers (CONV), all followed by a hyperbolic tangent activation function (TANH). The first two blocks include also a final max-pooling (MAXP) layer, while the third convolutional layer of the last block

does not have any activation function. Finally, a pixel-wise soft-max classifier is applied on top of the last convolutional layer.

In order to decrease the computation time, the input images are fed to the CNN at the coarser resolution of 320×240 (the resolution is then reduced to 160×120 in the second main block and to 80×60 in the third one). The convolutional layers have 90 filters in the first block, 128 in the second block and 256 in the last one. As for the CNN1 network, all filters are 7×7 pixels size. The final soft-max classifier has a weight matrix of size 256×14 and no bias.

The first layer filters are arranged into 9 groups so that the filters in the i-th group are connected to the i-th input channel only. As in the CNN1 case, local contrast normalization is applied to each input channel independently, so that the filter weights in the first convolutional layer converge faster.

The network is trained to produce a semantic segmentation of the input image by assigning each pixel to one of the 14 different classes proposed in [8], the same ones used for CNN1. With both networks, in addition to obtaining the final predicted labels, we leverage the output of the soft-max classifier to compute the descriptors defining the similarity measure between two regions of the initial over-segmentation. The output of the soft-max is a 3D array of size $80 \times 60 \times 14$, and we linearly interpolate it to the size of the input image so that a descriptor vector $\boldsymbol{c}_k = [c_k^1, \ldots, c_k^{14}]$ is associated to each pixel p_k. Each descriptor vector can be considered as a discrete probability density function (PDF) associated to the corresponding pixel, since its elements are non-negative and sum up to 1. Then, a probability density function $\boldsymbol{s}_i = [s_i^1, \ldots, s_i^{14}]$ can be associated to each segment S_i by simply averaging the PDFs associated to the pixels it contains, that is:

$$\boldsymbol{s}_i = \frac{\sum_{j \in S_i} \boldsymbol{c}_j}{|S_i|} \,. \tag{5.3}$$

Given two segments S_i and S_j, their similarity can then be estimated by comparing their descriptors $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$, which are two PDFs. An effective approach to compare two PDFs consists in using the Bhattacharyya coefficient

$$b_{i,j} = \sum_{t=1,..,14} \sqrt{s_i^t s_j^t} \,, \tag{5.4}$$

then we consider this quantity as our similarity measure.

As an example, Figure 5.3 displays the proposed similarity score between neighboring segments on a sample scene. The color of each boundary between two segments is proportional to the corresponding $b_{i,j}$ value (darker corresponds to

lower similarity values). In particular, notice in Figure 5.3a that most neighboring regions corresponding to different objects have a low similarity value (dark gray boundaries), while segments belonging to the same object share higher values (light gray boundaries). In Figure 5.3b, the boundaries between segments resulting from the final steps of the merging procedure (see Section 5.2) typically correspond to low similarity scores (dark gray).

(a) (b)

Figure 5.3: 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 colors are proportional to the similarity between the neighboring segments (dark gray corresponds to low $b_{i,j}$ values, light gray to large ones).

5.2 Region merging

In this section we present our additional region merging segmentation method. The scheme originates from the approach already described in Section 4.2, revised to take advantage of the classification data obtained by the CNNs of previous section, and to consider the task of semantic labeling in addition to objects segmentation. We refer to Section 4.2 for the overall procedure then, and we highlight here the main differences in this revised version.

The workflow using the CNN2 network of Section 5.1 is displayed in Figure 5.4. In [40] we employ the simpler CNN1 network instead, as to the figure notice that the difference in this case is that the HHA descriptors and the NURBS surfaces do not need to be calculated, since they are not used as input.

After obtaining the initial over-segmentation and the classification from the CNN, the algorithm starts by selecting the couples of segments that are candidate to be joined. For this, it determines whether the neighboring segments can be considered as adjacent as in the previous version, that is, by checking compatibility conditions along their common boundary. The same constraints on depth, color

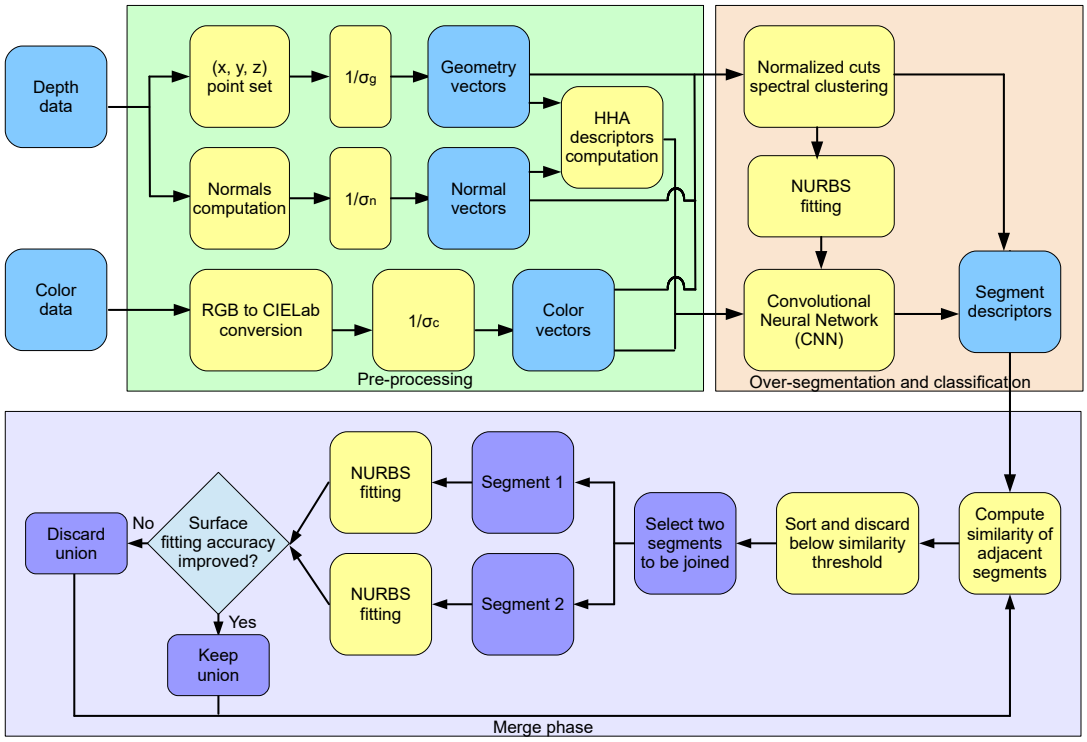

Figure 5.4: Overview of the revised region merging approach.

and orientation of Section 4.3 are used,

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta Z_i \leq T_z)|}{|P_i : P_i \in C_C|} > 0.5\,, \tag{5.5}$$

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta C_i \leq T_c)|}{|P_i : P_i \in C_C|} > 0.5\,, \tag{5.6}$$

$$\frac{|P_i : (P_i \in C_C) \wedge (\Delta \theta_i \leq T_\theta)|}{|P_i : P_i \in C_C|} > 0.5\,, \tag{5.7}$$

in addition to the requirement that the common boundary C_C itself must be longer than T_l. We set $T_l = 50$ instead of the previous $T_l = 15$ to allow more neighboring segments, in order to increase the effect of the similarity information returned by the CNN. As to the other adjacency thresholds, we use $T_z = 0.2\ m$, $T_c = 8$ and $T_\theta = 4°$ in [40] (based on the CNN1 network) and we slightly change $T_c = 10$ for the latest results based on the CNN2 network.

At this point, obtained the list L_A of couples of adjacent segments, the algorithm computes the similarity value $b_{i,j}$ for each couple based on the results of the machine learning stage, as described in Section 5.1. The list L_A is sorted according to the values $b_{i,j}$. This is a major difference with respect to the original region merging scheme, since in that one the processing order of the segments depends

on the surface fitting accuracy instead. Furthermore, the couples of segments with a similarity value $b_{i,j}$ below a threshold T_{sim} are removed from L_A and thus not considered for a merging operation (we use $T_{sim} = 0.75$ in [40] and $T_{sim} = 0.77$ for the results obtained with the CNN2 network). The rationale is that we do not want to merge segments with low similarity, that is, with different properties, since they probably belong to distinct objects or parts of the scene.

The algorithm then processes the first couple in L_A, that is, the one with the highest similarity score. Let S_{i^*} and S_{j^*} be the two segments and let $S_{i^* \cup j^*}$ be their union. As in the original region merging scheme, a NURBS surface approximation is calculated for the two segments and for their union, and the corresponding fitting error e_{i^*}, e_{j^*} and $e_{i^* \cup j^*}$ are determined. As previously, the merge operation is accepted provided it improves the fitting accuracy, that is:

$$\frac{e_{i^*}|S_{i^*}| + e_{j^*}|S_{j^*}|}{e_{i^* \cup j^*}(|S_{i^*}| + |S_{j^*}|)} > 1\,. \tag{5.8}$$

If the condition of Eq. 5.8 is not satisfied, the merging operation is discarded. Otherwise, the two segments S_i^* and S_j^* are merged. If this is the case, all the couples involving any of them are removed from the list L_A, and the adjacency information is updated by considering the union $S_{i^* \cup j^*}$ as adjacent to all the segments that were previously adjacent to any of the two single segments. The descriptor $\boldsymbol{s}_{i^* \cup j^*}$ associated to $S_{i^* \cup j^*}$ is then computed using Eq. 5.3 and the similarity scores are calculated for all the new couples containing segment $S_{i^* \cup j^*}$ created by the merge operation. These new couples are inserted in the list L_A at the positions corresponding to their similarity score, provided that the value is greater than T_{sim}. The algorithm then selects the next couple in L_A and the procedure is iterated until no more segments can be considered for a merging operation.

After obtaining the final segmentation, a semantic label is also associated to each segment by checking the descriptors of all its pixels (computed in Section 5.1), and assigning the most common class to the segment itself. Then, the classification data provided by the CNN is not only used as input in the region merging phase, but is also improved at the end of the process.

In summary, this revised region merging algorithm exploits both the classification information and the results of the NURBS surface fitting. The first is used to decide which couples of neighboring segments are processed first, and to discard those with a low similarity score. The second is used as a criterion to accept or discard each single merge operation.

The procedure is summarized in Algorithm 3, and some intermediate steps on a sample scene are shown in Figure 5.5.

Algorithm 3 Revised merge algorithm

Create list of couples of adjacent segments L_A
For each couple of adjacent segments compute similarity $b_{i,j}$ by Eq. 5.4
Sort L_A according to $b_{i,j}$
Discard the couples with $b_{i,j} < T_{sim}$
for each couple of adjacent segments $\{S_i, S_j\}$ in L_A **do**
  Compute fitting error on merged segment $S_{i \cup j}$
  if Eq. 5.8 satisfied **then**
    Remove all couples containing S_i or S_j from L_A
    Compute the adjacent segments S_k to $S_{i \cup j}$
    Insert the new couples of segments with $b_{k,i \cup j} \geq T_{sim}$ in L_A and sort
  end if
end for
Compute semantic labeling for all the segments

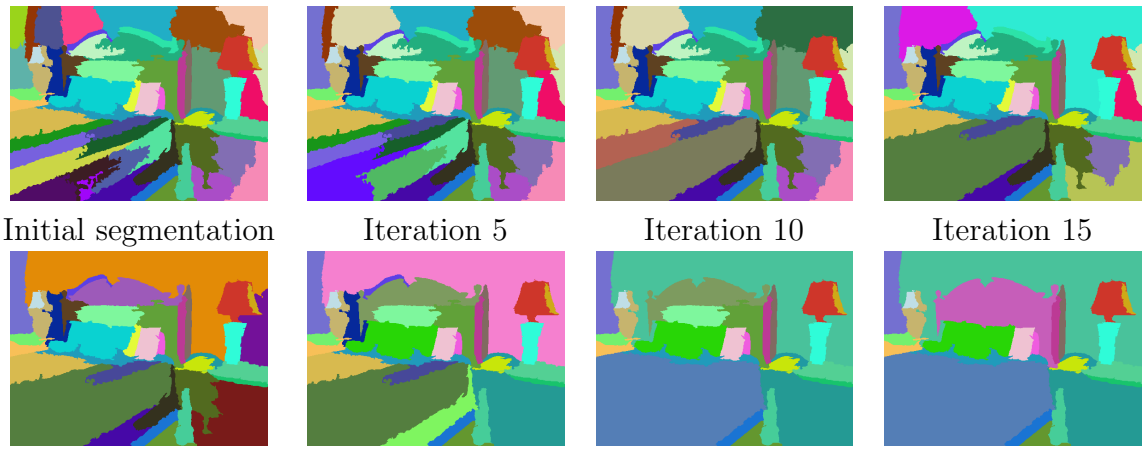

Figure 5.5: Example of the merging procedure on the scene of Fig. 5.6, row 6. The images show the initial over-segmentation, the merging output after 5, 10, 15, 20, 25, 30 iterations and the final result (iteration 32).

5.3 Experimental results

To test the approach proposed in Section 5.2 and compare it with state-of-the-art methods, we perform an experimental evaluation on the NYU Depth Dataset V2 (NYUDv2) [49], already used in Section 4.4 to test the methods of previous chapter. As previously mentioned, this dataset contains 1449 depth and color frames from a variety of indoor scenes acquired with a Kinect v1 sensor. We use the updated ground truth labels from [23] since the original ones have missing areas.

We consider the subdivision into a training set with 795 scenes and a test set with 654 scenes adopted by [23]. For the semantic labeling task we provide the results on the test set, since this is the approach used by all the competing approaches. Regarding the segmentation instead, most methods (including the ones we proposed in the previous chapter) are usually evaluated on the complete dataset. To obtain corresponding results with our method that requires a learning stage, we perform two independent tests. In the first one we use the standard training/test subdivision of the dataset, to train the CNN with the ground truth labels and then obtain the scene descriptors on the test set. In the second one we swap the train and test sets and perform the same procedure.

We consider both the Convolutional Neural Network architectures introduced in Section 5.1 for the results comparison. Then, in the following we refer to the region merging method of Section 5.2 as CNN1+Merge or CNN2+Merge to specify whether the CNN1 or the CNN2 network was used for the classification. Concerning the CNN optimization, for both CNN1 and CNN2 we use quadratic regularization with coefficient 0.001. We update the network weights using stochastic gradient descent with initial learning rate equal to 0.01, and constant decay by a factor 0.5 every 15 epochs for CNN1. For CNN2 we apply an adaptive decay policy reducing the learning rate by a factor of 0.7 after 10 epochs without improvement. In case of CNN1, to mitigate the possible over-fitting we expand the dataset by randomly rotating each sample by an angle between -6 and 6 degrees. We do not apply any dataset expansion for CNN2 instead.

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. We present the results for the two different tasks separately in the next two subsections.

Segmentation accuracy

Table 5.1 shows the comparison between our approach and some state-of-the-art methods on the NYUDv2 dataset (for the competing approaches we use the results collected by [28]). As in Section 4.4, the compared approaches are the clustering and region merging method of [28], the MRF scene labeling scheme of [45], that exploits Kernel Descriptors and SVM for machine learning, a modified version of [18] that accounts also for geometry information, the dynamic programming scheme of [52], the multi-layer strategy of [34] and the spectral clustering approach of [11]. As to the latter, recall that it can be considered as a simplified version of the segmentation scheme of Section 2.2. We include in the comparison also our region merging scheme [41] of Section 4.2 and our combined region splitting and merging scheme of Section 4.3 (denoted as Split+Merge). In particular, notice that the comparison with [41] gives an indication of the improvement obtained by using the CNN descriptors, since also that method is based on an initial over-segmentation and a merging iteration controlled by NURBS surface fitting (without any machine learning stage). Concerning our method of Section 5.2, we list the results both for the CNN1+Merge and the CNN2+Merge variants.

As in Section 4.4, in our evaluation we compare the results against ground truth data using the Variation of Information (VoI) and the Rand Index (RI) segmentation metrics. Recall that for the VoI metric a lower value is better while a higher one is better for RI. The average VoI score of our CNN2+Merge method is 1.92. According to this metric the approach is the best among the considered ones, with a significant gap with respect to all the competing approaches. Notice that the difference with CNN1+Merge (same segmentation scheme with a less accurate semantic classification) is very small. Much more significant differences concerning the semantic labeling will be highlighted in the next subsection.

If the RI metric is considered, the average score of the CNN2+Merge method is 0.91. This value is better than that of [18], [52], [11], [28], [45], [41] (our region merging approach) and of our Split+Merge scheme, while it is exactly the same of the best competing approach [27] and of the CNN1+Merge method [40]. As previously mentioned for our methods based on surface fitting alone, we point out that also our CNN-based approach does not assume the presence of planar surfaces, thanks to the NURBS surface approximation scheme, while some competing methods (e.g., [28], [27] and [52]) rely on this clue. They obtain good results on the NYUDv2 dataset where most of the surfaces are planar in this way, at the price of reduced generalization capabilities when considering scenes with arbitrary surfaces.

Approach	*VoI*	*RI*
Hasnat et al. (2014) [28]	2.29	0.90
Hasnat et al. (2016) [27]	2.20	**0.91**
Ren et al. [45]	2.35	0.90
Felzenszwalb et al. [18]	2.32	0.81
Taylor et al. [52]	3.15	0.85
Khan et al. [34]	2.42	0.87
(Clustering) [11]	3.09	0.84
(Merge) [41]	2.23	0.88
(Split+Merge)	2.17	0.89
(CNN1+Merge) [40]	1.93	**0.91**
(CNN2+Merge)	**1.92**	**0.91**

Table 5.1: Average values of the VoI and RI metrics on the 1449 scenes of the NYUDv2 dataset for our proposed approaches and for some state-of-the-art methods from the literature.

Some visual results for the proposed approach are displayed in Figure 5.6. 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 scheme thanks to the contribution of the CNN descriptors, that make it possible 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 no noisy small segments close to the edges as in other approaches. Only a few inaccuracies are present, on small objects in particular. The visual results confirm that our CNN1+Merge and CNN2+Merge approaches achieve almost identical segmentation results, even if some slight improvements given by the latter are noticeable (e.g., the background wall on scene 450 and the corner between the two background walls on scene 1110).

Classification accuracy

The proposed approach provides also a semantic label for each segment. In order to evaluate the labeling accuracy we compare it with some state-of-the-art approaches on the test set of the NYUDv2 dataset.

The compared approaches are the method of [7] that uses a multi-scale CNN, the method of [30] that uses a hierarchy of super pixels to train a random forest classifier, the method of [55] that leverages deep learning to extract superpixels features and the method of [56] that exploits two different CNNs.

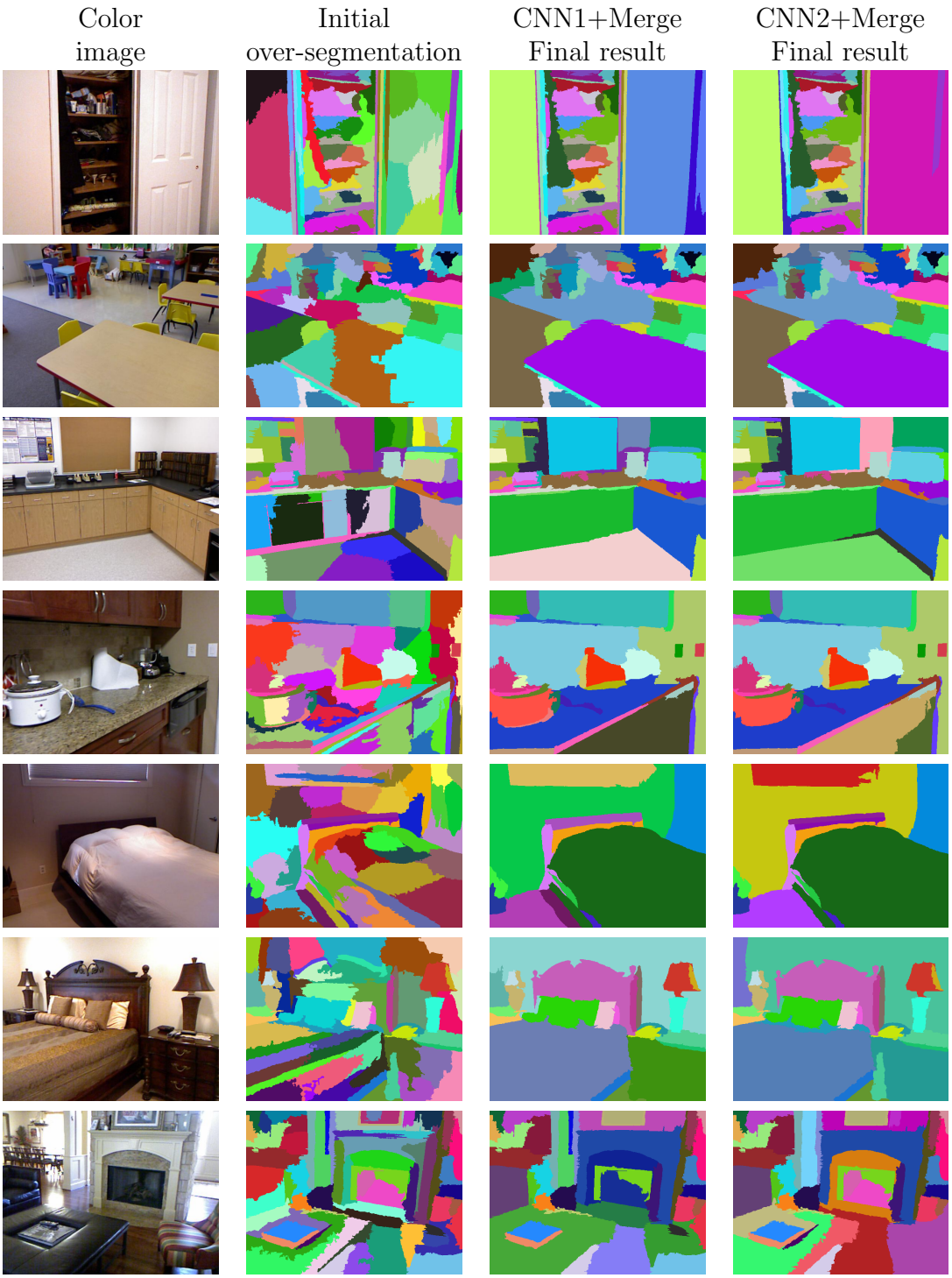

Figure 5.6: Segmentation of some sample scenes from the NYUDv2 dataset. The figure shows the color images, the initial over-segmentation and the final results for scenes 72, 330, 450, 846, 1105, 1110 and 1313.

Table 5.2 lists the results. We consider two different metrics: the per-pixel accuracy, that is, 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 CNN2 is able to obtain an average pixel accuracy of 64.4% on the test set. By leveraging the segmentation output of the merging iteration to assign a single label to each segment, as described in Section 5.2, 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 [30] and [56].

The results are confirmed also by the average class accuracy. The output accuracy of CNN2 is 51.7%, a remarkable result outperforming all compared approaches except [56]. By refining it with the segmentation the accuracy increases to 54.4%, outperforming all the compared approaches including [56]. Table 5.3 reports also the accuracy for each class. Notice how the value is very high for several classes, and quite low only for few of them (typically the 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 Figure 5.7. Notice how the classification is accurate even in challenging situations (e.g., the closed windows), and how the refinement provided by the segmentation largely improves the edges accuracy. Only few errors are present, e.g., beds exchanged with sofas that have a similar visual appearance.

In the proposed method, the CNN implementation is done using the Theano deep learning library [53]. Even if the current implementation has not been optimized, the segmentation of a single image with the associated depth map takes less than two minutes on average. Most of computation time is spent on the initial over-segmentation ($87s$), performed with the normalized cuts approach of Section 2.2. Notice that some simpler and faster superpixel segmentation scheme could be employed for this step.

Approach	*Pixel Accuracy*	*Class Accuracy*
Couprie et al [28]	52.4%	36.2%
Hickson et al [30]	53.0%	47.6%
A. Wang et al [55]	46.3%	42.2%
J. Wang et al [56]	54.8%	52.7%
Proposed (CNN1 output) [40]	59.6%	42.8%
Proposed (CNN2 output)	64.4%	51.7%
Proposed (CNN2 with segmentation)	67.2%	54.4%

Table 5.2: Average values of the pixel and class accuracies on the 654 scenes in the test set of the NYUDv2 dataset for our proposed approaches and for some state-of-the-art methods from the literature.

Class	*Accuracy (CNN2 output)*	*Accuracy (CNN2 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%

Table 5.3: Average accuracy for each of the 13 classes on the test set of the NYUDv2 dataset for our proposed CNN2+Merge approach (the *unknown* class has not been considered consistently with the evaluation of all the compared approaches).

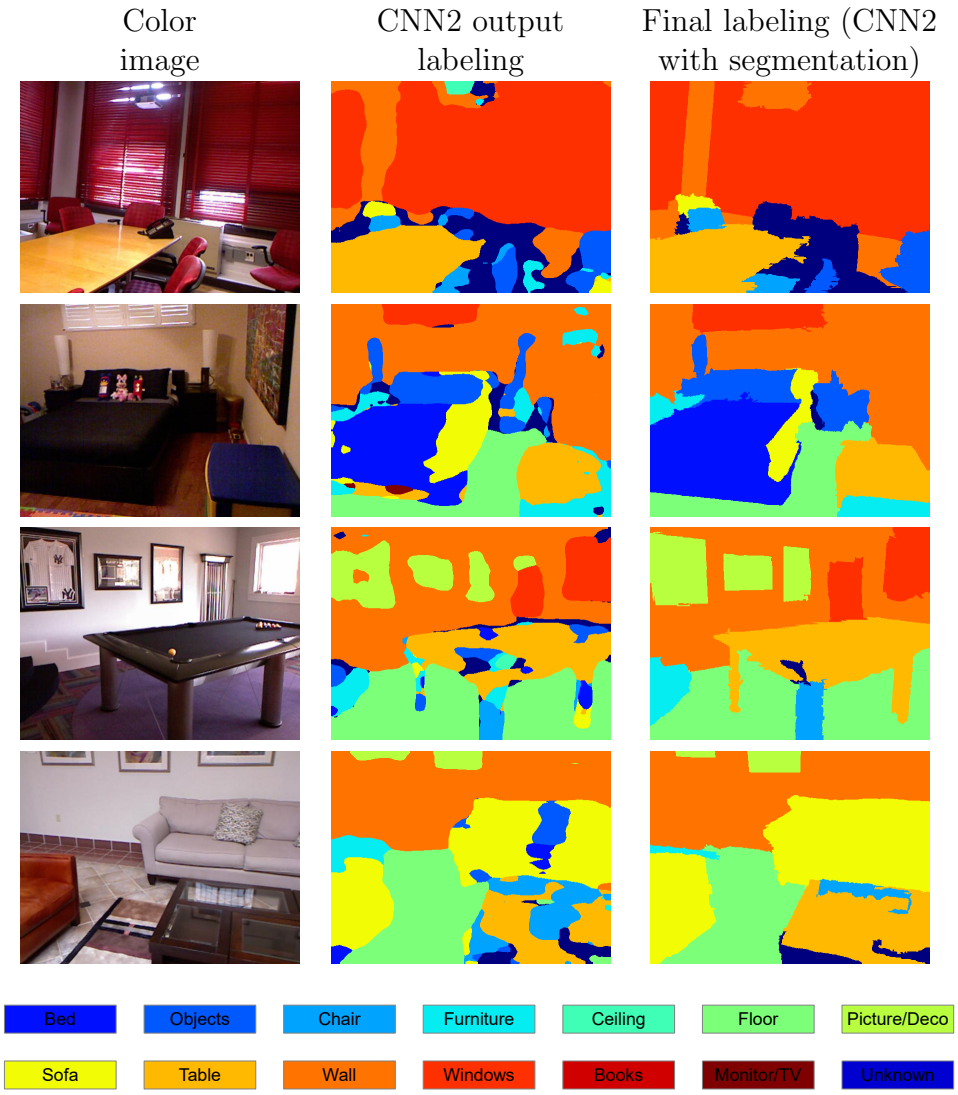

Figure 5.7: Semantic labeling of some sample scenes from the NYUDv2 dataset. The figure shows the color images, the labeling from the Convolutional Neural Network CNN2 and the refined labeling exploiting segmentation data for scenes 39, 280, 433 and 462.

Chapter 6

Conclusions

This thesis proposes novel region splitting and merging schemes for the joint segmentation of color and depth information. Spectral clustering is used inside tree-structured algorithms to progressively divide the scene into a set of segments, that are then recombined by a recursive merging procedure into larger regions corresponding to the actual scene objects. The proposed approaches exploit a NURBS surface fitting scheme to determine if each splitting or merging operation leads to a more accurate representation of the 3D surfaces, and consequently whether it should be accepted or discarded. The key idea consists in interpreting the surface approximation error as a measure of the plausibility that each segment actually corresponds to a single object or part of the scene. This gives a criterion to control the steps of the iterative splitting and merging procedures. Experimental results demonstrate that the proposed methods are capable to avoid over-segmentation and at the same time properly segment the various structures in the scene, outperforming several state-of-the-art approaches.

In addition, a joint segmentation and semantic labeling scheme exploiting deep learning and an iterative merging procedure is proposed. In this approach, the surface fitting information is used both to control the iterative merging steps and as an additional input channel to improve the performances of the adopted Convolutional Neural Network. The joint usage of geometric clues and of a segments similarity measure estimated from the CNN descriptors provides a way to properly select the merging operations to be performed. As shown by experimental results, this method achieves state-of-the-art performances both for the segmentation and for the semantic labeling task.

Bibliography

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. "Contour Detection and Hierarchical Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.5 (2011), pp. 898–916 (cit. on p. 45).

[2] D. Banica and C. Sminchisescu. "Second-order constrained parametric proposals and sequential search-based structured prediction for semantic segmentation in RGB-D images". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3517–3526 (cit. on p. 5).

[3] A. Bleiweiss and M. Werman. "Fusing time-of-flight depth and color for real-time segmentation and tracking". In: *Proc. of DAGM Workshop*. 2009, pp. 58–69 (cit. on p. 4).

[4] M. Bleyer, C. Rother, P. Kohli, D. Scharstein, and S. Sinha. "Object Stereo-Joint Stereo Matching and Object Segmentation". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011 (cit. on p. 4).

[5] F. Calderero and F. Marques. "Hierarchical fusion of color and depth information at partition level by cooperative region merging". In: *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing ICASSP*. 2009, pp. 973–976 (cit. on p. 4).

[6] D. Comaniciu and P. Meer. "Mean shift: a robust approach toward feature space analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.5 (2002), pp. 603–619 (cit. on pp. 3, 4).

[7] C. Couprie, C. Farabet, L. Najman, and Y. Lecun. "Convolutional nets and watershed cuts for real-time semantic Labeling of RGBD videos." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 3489–3511 (cit. on pp. 5, 64).

[8] C. Couprie, C. Farabet, L. Najman, and Y. LeCun. "Indoor semantic segmentation using depth information". In: *International Conference on Learning Representations*. 2013 (cit. on pp. 54, 55, 57).

[9] M. Dahan, N. Chen, A. Shamir, and D. Cohen-Or. "Combining color and depth for enhanced image segmentation and retargeting". In: *The Visual Computer* 28.12 (2012), pp. 1181–1193 (cit. on p. 4).

[10] C. Dal Mutto, P. Zanuttigh, and G. Cortelazzo. "A Probabilistic Approach to ToF and Stereo Data Fusion". In: *3DPVT*. Paris, France, 2010 (cit. on p. 11).

[11] C. Dal Mutto, P. Zanuttigh, and G. Cortelazzo. "Fusion of Geometry and Color Information for Scene Segmentation". In: *IEEE Journal of Selected Topics in Signal Processing* 6.5 (2012), pp. 505–521 (cit. on pp. 4, 5, 11, 12, 44–46, 51, 52, 63, 64).

[12] Z. Deng and L. J. Latecki. "Unsupervised Segmentation of RGB-D Images". In: *Proceedings of Asian Conference on Computer Vision (ACCV)*. Springer, 2014, pp. 423–435 (cit. on p. 4).

[13] Z. Deng, S. Todorovic, and L. Jan Latecki. "Semantic segmentation of RGBD images with mutex constraints". In: *Proceedings of International Conference on Computer Vision (ICCV)*. 2015, pp. 1733–1741 (cit. on p. 5).

[14] M. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976 (cit. on p. 24).

[15] D. Eigen and R. Fergus. "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2650–2658 (cit. on p. 5).

[16] C. Erdogan, M. Paluri, and F. Dellaert. "Planar Segmentation of RGBD Images Using Fast Linear Fitting and Markov Chain Monte Carlo". In: *Proc. of CRV*. 2012 (cit. on p. 4).

[17] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. "Learning hierarchical features for scene labeling". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1915–1929 (cit. on pp. 54, 55).

[18] P. Felzenszwalb and D. Huttenlocher. "Efficient Graph-Based Image Segmentation". In: *International Journal of Computer Vision* 59.2 (Sept. 2004), pp. 167–181 (cit. on pp. 3, 50–52, 63, 64).

[19] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. "Spectral grouping using the Nyström method". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.2 (2004), pp. 214–225 (cit. on p. 11).

[20] J. Gallego and M. Pardàs. "Region based foreground segmentation combining color and depth sensors via logarithmic opinion pool decision". In: *Journal of Visual Communication and Image Representation* 25.1 (2014), pp. 184–194 (cit. on p. 4).

[21] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.)* Baltimore, MD, USA: Johns Hopkins University Press, 1996 (cit. on p. 27).

[22] G. Guennebaud, B. Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010 (cit. on p. 27).

[23] S. Gupta, P. Arbelaez, and J. Malik. "Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013 (cit. on pp. 50, 55, 62).

[24] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik. "Indoor Scene Understanding with RGB-D Images: Bottom-up Segmentation, Object Detection and Semantic Segmentation". In: *International Journal of Computer Vision* (2014), pp. 1–17 (cit. on p. 5).

[25] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. "Learning rich features from RGB-D images for object detection and segmentation". In: *Proceedings of European Conference on Computer Vision (ECCV)*. 2014, pp. 345–360 (cit. on pp. 5, 56).

[26] M. Harville, G. Gordon, and J. Woodfill. "Foreground segmentation using adaptive mixture models in color and depth". In: *Proc. of IEEE Workshop on Detection and Recognition of Events in Video*. 2001 (cit. on p. 4).

[27] M. Hasnat, O. Alata, and A. Tremeau. "Joint Color-Spatial-Directional clustering and Region Merging (JCSD-RM) for unsupervised RGB-D image segmentation". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (2016) (cit. on pp. 4, 50, 51, 63, 64).

[28] M. A. Hasnat, O. Alata, and A. Tremeau. "Unsupervised RGB-D image segmentation using joint clustering and region merging". In: *Proceedings of BMVC*. 2014 (cit. on pp. 4, 50–52, 63, 64, 67).

[29] D. Herrera C, J. Kannala, and J. Heikkila. "Joint Depth and Color Camera Calibration with Distortion Correction". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.10 (Oct. 2012), pp. 2058–2064 (cit. on p. 11).

[30] S. Hickson, I. Essa, and H. Christensen. "Semantic Instance Labeling Leveraging Hierarchical Segmentation". In: *Winter Conference on Applications of Computer Vision*. 2015, pp. 1068–1075 (cit. on pp. 64, 66, 67).

[31] N. Höft, H. Schulz, and S. Behnke. "Fast semantic segmentation of RGB-D scenes with GPU-accelerated deep neural networks". In: *Joint German/Austrian Conference on Artificial Intelligence*. 2014, pp. 80–85 (cit. on p. 5).

[32] S. Holzer, R. Rusu, M. Dixon, S. Gedikli, and N. Navab. "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images". In: *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012, pp. 2684–2689 (cit. on p. 11).

[33] L.-H. Juang, M.-N. Wu, and F.-M. Tsou. "A dynamic portrait segmentation by merging colors and depth information". In: *International Journal of Control, Automation and Systems* 13.5 (2015), pp. 1286–1293 (cit. on p. 4).

[34] M. R. Khan, A. B. M. M. Rahman, G. M. A. Rahaman, and M. A. Hasnat. "Unsupervised RGB-D image segmentation by multi-layer clustering". In: *Proceedings of International Conference on Informatics, Electronics and Vision*. 2016, pp. 719–724 (cit. on pp. 50, 51, 63, 64).

[35] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. "Bi-layer segmentation of binocular stereo video". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. 2005, 1186 vol. 2 (cit. on p. 4).

[36] L. Ladicky, P. Sturgess, C. Russell, S. Sengupta, Y. Bastanlar, W. Clocksin, and P. Torr. "Joint Optimisation for Object Class Segmentation and Dense Stereo Reconstruction". In: *Proceedings of the British Machine Vision Conference*. 2010 (cit. on p. 4).

[37] J.-E. Lee and R.-H. Park. "Segmentation with saliency map using colour and depth images". In: *IET Image Processing* 9.1 (2015), pp. 62–70 (cit. on p. 4).

[38] J. Leens, S. Piérard, O. Barnich, M. Van Droogenbroeck, and J. Wagner. "Combining Color, Depth, and Motion for Video Segmentation". In: *Computer Vision Systems*. 2009 (cit. on p. 4).

[39] D. Lin, S. Fidler, and R. Urtasun. "Holistic scene understanding for 3d object detection with rgbd cameras". In: *Proceedings of International Conference on Computer Vision (ICCV)*. 2013, pp. 1417–1424 (cit. on p. 5).

[40] L. Minto, G. Pagnutti, and P. Zanuttigh. "Scene Segmentation Driven by Deep Learning and Surface Fitting". In: *Proceedings of ECCV Geometry meets deep learning workshop*. 2016 (cit. on pp. 7, 54, 58–60, 63, 64, 67).

[41] G. Pagnutti and P. Zanuttigh. "Joint Color and Depth Segmentation Based on Region Merging and Surface Fitting". In: *Proceedings of VISAPP*. 2016 (cit. on pp. 6, 44–46, 50, 51, 63, 64).

[42] G. Pagnutti and P. Zanuttigh. "Scene segmentation based on NURBS surface fitting metrics". In: *Proceedings of STAG*. 2015 (cit. on pp. 6, 31, 33, 34, 44–49, 51).

[43] G. Pagnutti and P. Zanuttigh. "Scene segmentation from depth and color data driven by surface fitting". In: *Proceedings of IEEE International Conference on Image Processing (ICIP)*. 2014, pp. 4407–4411 (cit. on pp. 6, 44).

[44] L. Piegl and W. Tiller. *The NURBS Book (2Nd Ed.)* New York, NY, USA: Springer-Verlag New York, Inc., 1997 (cit. on pp. 15, 16, 18, 28).

[45] X. Ren, L. Bo, and D. Fox. "RGB-D scene labeling: Features and algorithms". In: *Proceedings of CVPR*. 2012 (cit. on pp. 5, 50–52, 63, 64).

[46] A. Richtsfeld, T. Morwald, J. Prankl, M. Zillich, and M. Vincze. "Segmentation of unknown objects in indoor environments". In: *IROS*. 2012 (cit. on p. 4).

[47] E. Shelhamer, J. Long, and T. Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2016) (cit. on p. 5).

[48] J. Shi and J. Malik. "Normalized Cuts and Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 888–905 (cit. on pp. 3, 4, 9, 10, 44).

[49] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. "Indoor Segmentation and Support Inference from RGBD Images". In: *Proceedings of European Conference on Computer Vision (ECCV)*. 2012 (cit. on pp. 5, 44, 50, 55, 62).

[50] N. Srinivasan and F. Dellaert. "A Rao-Blackwellized MCMC Algorithm for Recovering Piecewise Planar 3D model from Multiple View RGBD Images". In: *Proceedings of IEEE International Conference on Image Processing (ICIP)*. 2014 (cit. on p. 4).

[51] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010 (cit. on p. 3).

[52] C. J. Taylor and A. Cowley. "Parsing indoor scenes using RGB-D imagery". In: *Robotics: Science and Systems*. Vol. 8. 2013, pp. 401–408 (cit. on pp. 4, 50–52, 63, 64).

[53] Theano Development Team. "Theano: A Python framework for fast computation of mathematical expressions". In: *arXiv e-prints* (2016) (cit. on p. 66).

[54] M. Wallenberg, M. Felsberg, P.-E. Forssén, and B. Dellen. "Channel Coding for Joint Colour and Depth Segmentation". In: *Proceedings of DAGM*. 2011, pp. 306–315 (cit. on p. 4).

[55] A. Wang, J. Lu, G. Wang, J. Cai, and T.-J. Cham. "Multi-modal unsupervised feature learning for rgb-d scene labeling". In: *Proceedings of European Conference on Computer Vision (ECCV)*. 2014, pp. 453–467 (cit. on pp. 5, 64, 67).

[56] J. Wang, Z. Wang, D. Tao, S. See, and G. Wang. "Learning Common and Specific Features for RGB-D Semantic Segmentation with Deconvolutional Networks". In: *Proceedings of European Conference on Computer Vision (ECCV)*. 2016, pp. 664–679 (cit. on pp. 5, 64, 66, 67).

[57] L. Wang, C. Zhang, R. Yang, and C. Zhang. "TofCut: Towards Robust Real-time Foreground Extraction Using a Time-of-Flight Camera". In: *3DPVT*. 2010 (cit. on p. 4).

[58] J. Yang, Z. Gan, K. Li, and C. Hou. "Graph-Based Segmentation for RGB-D Data Using 3-D Geometry Enhanced Superpixels". In: *IEEE Transactions on Cybernetics* 45.5 (2015), pp. 927–940 (cit. on p. 4).

[59] P. Zanuttigh, G. Marin, C. Dal Mutto, F. Dominio, L. Minto, and G. M. Cortelazzo. *Time-of-Flight and Structured Light Depth Cameras*. Springer, 2016 (cit. on p. 3).