Die approbierte Originalversion dieser Dissertation ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

DISSERTATION

Learning and Modeling Scene Context for Semantic Segmentation of 3D Point Clouds

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
E376
Institut für Automatisierungs- und Regelungstechnik

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

Dipl.-Ing. Daniel Wolf
geb. am 02.07.1987
Matr. Nr.: 1229647

Wien, im Mai 2017

Daniel Wolf

Abstract

With autonomous robotic systems advancing and finally also making their way onto our roads and into our homes in the foreseeable future, it is vital that these systems are equipped with capabilities to recognize and interpret their environment and react to it with intelligent informed decisions. Autonomous cars need to quickly distinguish between drivable road, cars, sidewalks, buildings and people. Or mobile service robots, regardless if they are used in an industrial setting or at home, have to recognize and understand their immediate surroundings to fully exploit their potential.

A fundamental underlying problem to achieve this level of holistic visual scene understanding is semantic segmentation, which describes the decomposition of a scene into its semantically meaningful parts. From a computer vision perspective, the major challenge of semantic segmentation is to resolve the frequent ambiguities that are observed in an image of a scene, which potentially consists of hundreds of different objects that often also occlude each other.

However, an essential property of most man-made scenes is that they are repeatedly arranged in a similar fashion, such as rooms serving a particular purpose. Being able to identify and consider this scene context can guide a semantic segmentation system to resolve challenging scenes and improve the performance.

Focusing on this capability, this thesis introduces two novel concepts to automatically learn and model contextual information from 3D point clouds of a scene and exploit it to improve semantic segmentation. Developed for computational efficiency, both methods exhibit fast processing times, which is a crucial factor to consider for online applications on a robot.

Our first approach is based on a random forest classifier to obtain a local semantic prediction, which is then refined using a densely connected conditional random field. Label compatibility parameters are learned and incorporated in the pairwise terms of the model, emphasizing frequently appearing pairwise combinations of objects, depending on their geometric arrangement.

The second method enhances the classifier by introducing a novel set of so-called 3D entangled features. This feature set directly enables a random forest to explicitly model and incorporate contextual and geometric relations between different objects, such that a separate refinement step is not required.

We compare both methods to each other and the current state of the art in a detailed evaluation on several indoor datasets. The results clearly indicate that taking context into account is crucial for semantic segmentation, boosting the performance in each case. In an in-depth analysis we further examine the individual contributions of our new entangled feature set and provide a comprehensive evaluation of the computational efficiency of our methods, proving their suitability for the deployment on a mobile robotic system.

This manuscript is best to read in color.

Acknowledgement

Throughout the development of this thesis I have been privileged to meet and work with many inspiring people and without whom this work would not have been possible.

First and foremost, I would like to thank my supervisor Prof. Markus Vincze and my mentor Dr. Johann Prankl for their continuous support. They have always granted me the freedom to pursue my own ideas, but also provided valuable guidance and new impulses when needed. Furthermore, I would like to thank Prof. Georg Schitter and Prof. Emanuele Menegatti for their time as second examiner and reviewer.

A big thank you goes to all of my exceptional colleagues at Vision4Robotics, who make the group the stimulating and very enjoyable working environment it is. Thanks to all of you for the many inspiring discussions over a cup of desperately needed coffee, during relaxed lunches on the rooftop or countless intense rounds of table soccer. Especially, I would like to thank my friends Astrid Weiss, Markus Bajones, Georg Halmetschlager-Funek and David Fischinger, who made this time a truly memorable experience, inside and outside of the office.

I am also indebted to the general public, who has largely funded this work through several EU and nationally funded projects.

Not least, I would like to express my sincere gratitude to my parents Doris and Michael and my brother Thomas, who enabled and always encouraged me to follow my own path. Finally, an especially warm thank you goes to my wonderful partner and friend Simone for her unlimited support and patience throughout this challenging and exciting time.

Contents

Nomenclature **IV**

1 Introduction **1**
1.1 Problem Statement 2
1.2 Proposed Framework 3
1.3 Contributions 7
1.3.1 Learning Context with a Dense Conditional Random Field - Chapter 3 7
1.3.2 3D Entangled Forests - Chapter 4 8
1.4 Related Work 8
1.4.1 Scene Understanding on 2D Images 9
1.4.2 Scene Understanding on RGB-D Data 10
1.4.3 Semantic Segmentation using Deep Learning 11
1.5 List of Publications 12
1.6 Outline 13

2 Random Forests for Classification **15**
2.1 The Decision Tree Model 16
2.2 Random Forest Training 16
2.2.1 Split Functions 18
2.2.2 Objective Function for Split Evaluation 19
2.2.3 Injecting Randomness for Generalization 19
2.2.4 Coping with imbalanced Data Sets 20
2.3 Random Forest Inference 21
2.3.1 Dynamic Pruning at Inference Time 21

3 Context Learning with a Dense CRF **25**
3.1 Oversegmentation 25
3.2 Feature Extraction 27
3.3 Random Forest Classifier 28
3.3.1 Training 28
3.3.2 Inference 29
3.4 Dense Conditional Random Field 29
3.4.1 Inference 32

3.4.2 Parameter Learning 32

4 Context Learning with 3D Entangled Forests 35

4.1 Overview 36
4.2 Preprocessing and Segmentation 37
4.3 3D Entangled Forests for Classification 38
4.3.1 Unary Features 38
4.3.2 3D Entangled Features 40
4.3.3 Modeling Spatial Relations with Geometric Constraints 41
4.3.4 Capturing Contextual Relations with Binary Tests 43
4.3.5 Implementation Considerations 46
4.3.6 Training and Inference 46

5 Evaluation 49

5.1 Datasets 49
5.2 Experimental Setup 51
5.2.1 Parameter settings 52
5.2.2 Performance Metrics 53
5.2.3 Ground Truth Mapping 54
5.3 Overall Quantitative Evaluation 54
5.3.1 NYU Depth Dataset v1 56
5.3.2 NYU Depth Dataset v2 56
5.4 Qualitative Evaluation 61
5.4.1 CRF Pipeline 61
5.4.2 3D Entangled Forest 61
5.5 Parameter Analysis 64
5.5.1 CRF Pipeline 64
5.5.2 3D Entangled Forest 66
5.6 Runtime Analysis 70
5.6.1 Full Pipelines 70
5.6.2 Scalability of the 3DEF 72

6 Conclusion and Outlook 75

6.1 Summary 75
6.2 Outlook 76
6.2.1 From Single Frame Predictions to Semantic Maps 76
6.2.2 From Pointwise Predictions to Object Instances 77

List of Figures 81

List of Tables 83

Bibliography 85

Curriculum Vitae 93

Nomenclature

Random Forests

$\mathcal{F}$	... Random Forest
$\mathcal{T}_l$	... Random Tree
$\mathcal{C}$	... set of available class labels
c	... class label
t	... number of trees
d	... tree depth
$\mathcal{N}$	... set of tree nodes
$\boldsymbol{n}_k$	... tree node
$\boldsymbol{n}_{k,l}, \boldsymbol{n}_{k,r}$	... child nodes of node k
$\boldsymbol{n}_0$	... root node
$\mathcal{X}$	... training dataset
$\mathcal{X}_k$	... training data subset for node k
$\mathcal{X}_{k,l}, \mathcal{X}_{k,r}$	... training data subset of child nodes of node k
$\boldsymbol{x}$	... data point
x_j	... single feature value of a training data point
ρ	... split function
$\boldsymbol{\theta}$	... split function parameters
τ	... split function threshold
ξ	... objective function for splits
ξ_{min}	... minimum objective function value for split
d_{max}	... maximum tree depth
p_{min}	... minimum number of points to split
$p(\mathcal{C}\|\mathcal{X})$	... class label distribution in training set $\mathcal{X}$
H	... entropy
b	... bagging ratio
f	... number of sampled feature indices per node
s	... number of sampled thresholds per feature
r	... number of available features per node

Dense Conditional Random Field

$\mathcal{X}$ ... set of random variables
X_i ... random variable
$\boldsymbol{\rho}$ ... model parameters
$\boldsymbol{P}$ ... voxelized point cloud
$\boldsymbol{v}_i$ ... voxel
$\mathcal{C}$ ... set of class labels
c_i ... class label
$\boldsymbol{f}_i$ ... feature vector of voxel i
P ... Gibbs distribution
E ... Gibbs energy
$\mathcal{G}$ ... graph
$\mathcal{V}$ ... set of vertices
$\mathcal{E}$ ... set of edges
$\mathcal{Q}_{\mathcal{G}}$ ... set of cliques in $\mathcal{G}$
q ... clique
$\boldsymbol{c}$ ... class label assignment
ϕ_q ... clique potential
ϕ_i ... unary potential
ϕ_{ij} ... pairwise potential
k_m ... kernel function
μ_m ... label compatibility term for kernel function k_m
w_m ... weight for kernel k_m
θ ... kernel range

Chapter 1

Introduction

When we as humans move around in an unknown environment, such as a city we have never visited or a house we have never entered before, our visual perception system is so well-trained and tuned, that in most cases the immediate gist of the scene we are observing is instantly clear to us [18, 19]. In a split second, we can tell in which room of an apartment we are, even though it has a structure and contains furniture we have never seen before. We can distinguish between a table, its surrounding chairs, and enclosing walls and deduct that we are in a dining room.

What comes so effortless for us as humans is still a problem far from being solved in computer vision research, where the problem is generally referred to as (visual) *scene understanding* [20, 22, 30, 50, 51, 74, 86, 92]. Every room and every city is unique, there are thousands of different ways how chairs, cars or any other object can look, particularly when observed with an imperfect sensor with capabilities falling far short of the human visual perception system.

Nevertheless, with robotic systems advancing and finally also making their way onto our roads and into our homes in the foreseeable future, it is vital that these systems are equipped with capabilities to recognize and interpret their environment and react to it with intelligent informed decisions. For example, an autonomous car needs to be constantly aware of its ever-changing surroundings, distinguishing between drivable road, cars, sidewalks, buildings and people. Or imagine a service robot at home that needs to recognize different room types and configurations in order to adapt its behavior accordingly.

An important cornerstone towards achieving this goal is the ability of a computer vision system to decompose an observed scene into its semantically meaningful parts—also known as semantic segmentation or semantic labeling, which is the central topic of this thesis.

Given an input scene, represented by a 2D image or a 3D point cloud, semantic segmentation generally delivers a pixel- or pointwise labeling. An example result is shown in Figure 1.1. This output represents an important first stage of semantic scene understanding, which can then be further exploited. For example, semantic segmentation allows a robot to intelligently search for particular objects; a remote control is likely to be found near a sofa or a TV, cups and plates are on tables

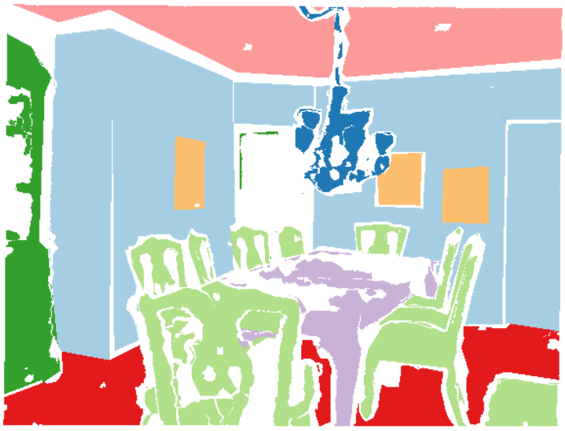


Figure 1.1: Semantic decomposition of a dining scene distinguishing different types of furniture (chairs, table, cabinet, lamp, pictures) and the structural components of the room (floor, wall, ceiling).

or in cabinets and so on. Or, to follow up on a previous example, if a robot can recognize a table that is surrounded by chairs, it could infer that it is most likely in a dining room. This conclusion could in turn trigger an adapted behavior model, enable room-specific features etc.

Semantic knowledge is also helpful during navigation and mapping. If a system can distinguish objects that likely remain static, such as cabinets or sofas, from possibly moving ones, such as chairs or objects on tables or the floor, localization can be focused on robust, static parts of the scene, providing a more stable pose estimation.

1.1 Problem Statement

Catalyzed by innovative sensor technologies such as the Microsoft Kinect[1] and other structured light sensors producing dense RGB-D data, the research field of semantic segmentation has received much attention in recent years [1, 9, 27, 28, 32, 33, 37, 38, 39, 53, 72, 81, 75, 91]. However, despite significant progress that has already been achieved, it remains a very difficult problem and is still far from being solved.

The major challenges include

- imperfect sensor systems delivering noisy, incomplete data with a limited field of view;
- the huge variety of different objects and object types to be observed in the open world, which can be hardly captured in training datasets;
- constantly occurring occlusions between different objects, resulting in partial and possibly degenerate views.

---

[1] Microsoft Corporation Kinect for Windows and Xbox 360

Some scenes exemplifying these challenges are shown in Figure 1.2.

These issues all contribute to the overall problem that different elements composing a scene often seem very ambiguous, particularly when being captured in a single 2D image or 3D point cloud. For example, if the backrest is occluded, a chair can easily look like a small table. Tables in turn are often so occupied with clutter that only their legs, but not the table plane itself is visible. Consequently, a semantic segmentation approach that treats different scene elements independently from each other and only relies on local cues to identify them is likely to fail for complex scenes.

On the other hand, however, most of the environments in which autonomous mobile robots operate (e.g. apartments, streets, warehouses) also exhibit a helpful common property: They are *man-made*, which in most cases implies that they have been designed to serve a particular purpose. That is, objects and structures are frequently arranged in a very distinctive manner; in apartments, rooms are enclosed by walls, tables are surrounded by chairs, nightstands are next to beds; outdoors, cars drive on streets, which are framed by sidewalks, populated by pedestrians.

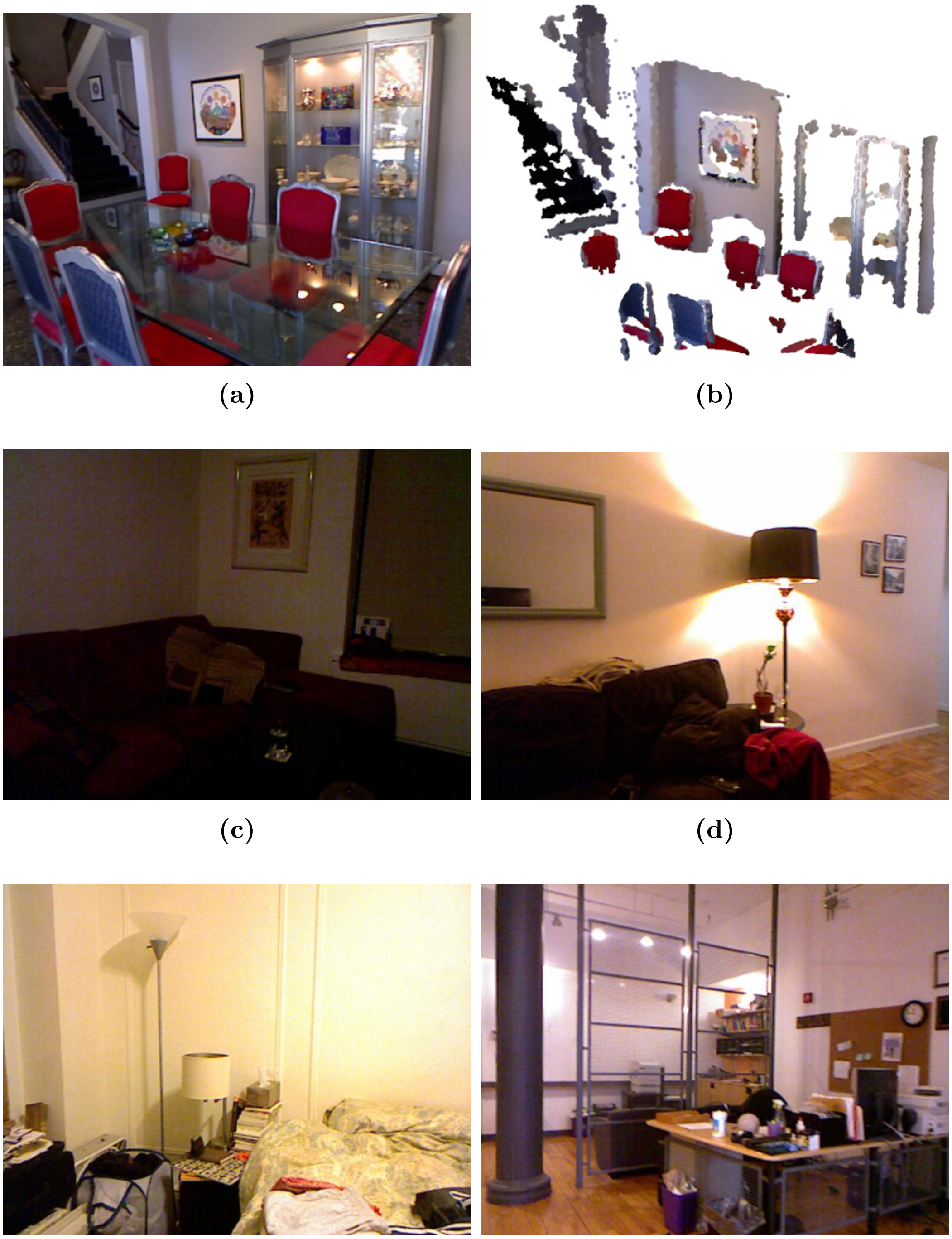
We claim that the exploitation of these common patterns and relations is the key to further improve semantic segmentation of complex scenes, where objects are often only partially visible and ambiguous. For example, consider the dining scene shown in Figure 1.1. The table and the chairs in the front are mostly visible, providing sufficient evidence for a classifier. In contrast, the chairs behind the table are heavily occluded, such that only a part of their backrest is visible. Processing these small clusters of points in isolation, it would be very difficult to recognize them as parts of chairs. However, if the table and some chairs in front of it can be identified and it is further known that tables are often surrounded by chairs, it can be inferred that the ambiguous point segments behind the table are most likely chairs as well.

Inspired by this concept, in this thesis we present two different data-driven methods for semantic segmentation, which are both capable of inferring, learning and modeling such informative contextual relations. We show that by exploiting these more global properties of a scene, which in the remainder will be simply referred to as *context*, we significantly improve the performance of common semantic segmentation architectures. Furthermore, with future robotic applications in mind, our methods are also designed for computational efficiency. They achieve comparably high frames rates of more than one frame per second without requiring a high-performance and energy-demanding GPU.

In the following sections, we present the basic architecture behind both of our methods, followed by their key contributions and a comprehensive analysis of related work.

1.2 Proposed Framework

Both of our proposed approaches operate on 3D point clouds. In this work, we focus on indoor scenarios, where such data can easily be acquired using RGB-D sensors such



(a) (b)

(c) (d)

(e) (f)

Figure 1.2: Exemplary scenes for the different challenges faced in semantic segmentation. (a) to (e) all show problems mainly caused by the sensor: (b) is the corresponding point cloud of the RGB image shown in (a) and highlights the problem that shiny surfaces, such as the table plane or the cabinet doors, are observed in the RGB image, but do not produce a useful response in the depth channel of structured light cameras. (c) and (d) illustrate problematic scenes due to bad illumination and (e) exemplifies the challenge of understanding a scene with a limited field of view. The last example (f) shows the difficulty caused by occlusions. Objects occupy almost the whole table plane, while the table in turn occludes a large portion of the chair behind it.

as a Microsoft Kinect. However, in general our methods are directly applicable to outdoor scenes as well, as long as registered depth and color information are provided.

Projected from the 2D image plane to 3D coordinates, a single point cloud recorded with a Microsoft Kinect has a resolution of 640 × 480 points, resulting in 307,200 points per frame in total. The base architecture of both of our pipelines is very similar, a principle overview is shown in Figure 1.3.

In a first step, the input point cloud is oversegmented into homogeneous point segments, considering the 3D locations, surface normals and the color of the points. This segmentation stage is based on Voxel Cloud Connectivity Segmentation (VCCS) from Papon et al. [66].

Efficient but descriptive feature vectors, capturing geometric properties as well as color information of a segment, are then computed for all acquired segments and represent the atomic units for the following classification step. Focusing on computational efficiency, we thereby purposefully avoid calculating complex features such as HOG [12], spin images [36] or textons [49, 79] and rely on simple color features, spectral properties and height and angle with respect to the ground plane. The latter is directly available since in a robotics application, we assume that we know the camera pose with respect to the ground plane.

In the second pipeline, the initial segmentation is further processed by iteratively merging similar adjacent segments to obtain larger, mostly planar patches. Consequently, compared to the segmentation approach of the first pipeline, the resulting segments exhibit more variation in size, which means that they contain more descriptive information. This allows us to compute additional expressive bounding box and height features in the second pipeline.

The general idea to classify homogeneous segments instead of single points stems from the fact that a feature vector calculated for a segment consisting of points with similar appearance is much more meaningful and stable than a feature vector computed for a local neighborhood of a point bounded by k-nearest-neighbors or a fixed radius. Consequently, the result is less prone to classification noise and eventually, because of the reduced size of input data, classification is also faster.

For the classification stage of the first pipeline, we train a random forest (RF) classifier, which outputs conditional probabilities for each considered semantic label. RFs are very versatile and efficient multi-label classifiers, which makes them very suitable for the task at hand. A detailed description of their theoretical background is given in chapter 2.

Even though an RF classifier predicts labels for larger segments instead of single points, it generally processes the different feature vectors independently from each other. Therefore, its output suffers from classification noise and, more importantly, the capability of incorporating more complex global information about a scene is limited.

We argue that lifting this limitation and being able to learn and model more global, contextual information is crucial in order to elevate the performance of semantic segmentation methods, particularly for complex scenes, e.g. where objects are seen from difficult viewpoints and where substantial mutual occlusion occurs.

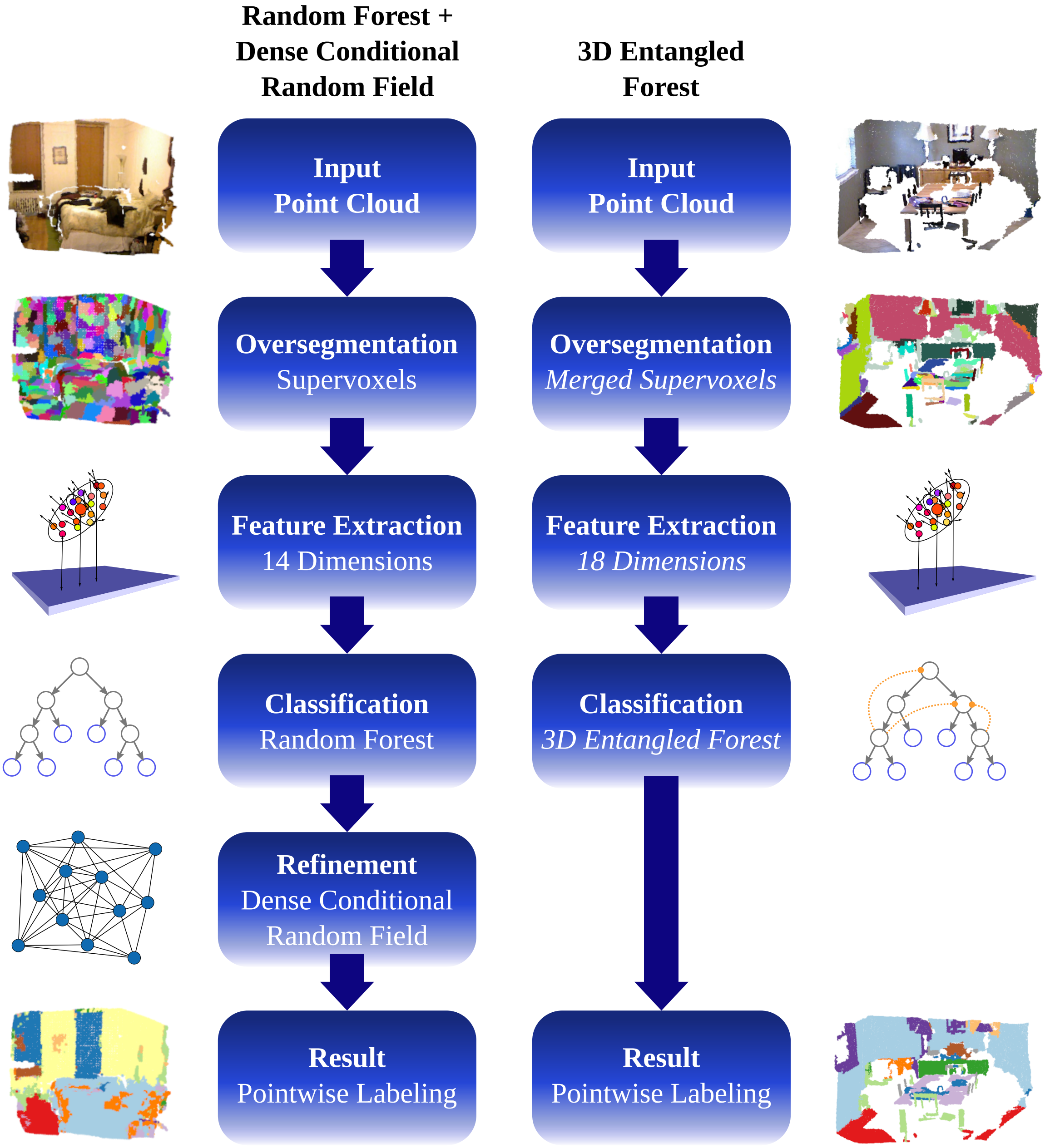


Figure 1.3: Overview of both proposed semantic segmentation pipelines. In the left half, our first approach is shown, in which a random forest classifier labels small homogeneous segments, followed by a refinement stage with a dense conditional random field with learned parameters. The right half shows the second pipeline, in which we replace the combination of random forest and conditional random field by a single classification stage consisting of a 3D Entangled Forest. Operating on larger, mostly planar point segments, this new classifier is able to learn, model and exploit contextual information in one step and efficiently delivers smooth and accurate results.

As the key contribution of this thesis we therefore introduce two different approaches to extend the common framework architecture consisting of segmentation, feature extraction, and classification, with this capability. In the next section, we present the contributions of our work in more detail.

1.3 Contributions

1.3.1 Learning Context with a Dense Conditional Random Field - Chapter 3

The first method adds an additional refinement step after classification, consisting of a densely connected conditional random field (CRF). CRFs are graphical models that are generally used as a final smoothing stage after classification. The basic idea is to optimize a cost function, which penalizes similar adjacent points being assigned different labels. In many applications, the parameters of the cost function are manually tuned, which is, considering the number of different labels and the large variation between different scenes, far from ideal for our application.

Therefore, in this thesis we apply a more expressive CRF model and (1) use a more complex cost function that specifically models all pairwise label combinations and (2) learn all individual weights from training data. This allows our framework to learn frequently occurring label combinations and emphasize or penalize different label configurations, depending on the appearance and geometric properties of the respective points. Furthermore, because all weights are learned from training data, the CRF is well adapted to the task at hand without the need for manual tuning.

By using a densely connected CRF (where the term “dense” refers to the fact that each point of the graphical model is connected to every other point in the scene) different parts of a scene can influence local results over long ranges, thus enabling global scene context to be incorporated in the final result.

To solve the complex graphical model, we apply an efficient method called mean field approximation, which allows for very fast inference, such that the entire pipeline is able to process a single Kinect frame in half a second. These fast processing times make our method very feasible to act as a solid base for several follow-up applications in the scope of mobile robotics.

We thoroughly evaluate and compare the approach on several challenging datasets and label configurations, proving the benefits of learning the CRF parameters for a more complex model compared to the standard smoothing approach. Furthermore, we analyze the learned weights to gain insight in the structure of the model and the contextual information that was gleaned from the datasets. Finally, the influence of different parameters during the learning procedure of the RF is examined, underlining the robustness of the classifier.

1.3.2 3D Entangled Forests - Chapter 4

For our second pipeline we replace the two separate steps of local classification and global refinement by a more compact approach, unifying both stages. To this end, we introduce a new classifier called *3D Entangled Forests* (3DEF). A 3DEF is not only very efficient at test time, but also able to learn complex, informative contextual and spatial relations between classes.

Consequently, in contrast to our first pipeline, this approach does not rely on a complex graphical model to capture those connections. Instead, we extend the capabilities of a standard RF by introducing a new powerful concept of 3D entangled features, which (1) capture frequently appearing combinations of classes and (2) explicitly learn their three-dimensional geometric configuration in the scene. These relations are not only learned between different classes, but also within the same class, which inherently leads to a smooth classification result without any further postprocessing being required. Thus, the additional refinement step from our first pipeline can be spared, resulting in a very compact overall framework that only requires a single inference step per input point cloud. As a consequence, the available limited amount of training data can be utilized more efficiently, since only one stage has to be trained and training data does not need to be shared between multiple learning stages.

The general underlying architecture of a 3DEF is still an RF, which yields a very efficient inference procedure that is also easily parallelizable. Consequently, while further improving the results of our first approach, our second method still offers similarly fast processing times in only a single inference step and is very suitable for real-time robotic applications.

We show the effectiveness of the new classifier with a detailed evaluation, outperforming several comparable approaches for semantic segmentation. Besides an overall quantitative evaluation, we also present a detailed analysis of the new 3D entangled features, highlighting their expressiveness and their contribution to the overall performance. Finally, we present illustrative examples that prove their capability to learn frequently observed meaningful contextual relations and geometric arrangements from training data.

1.4 Related Work

The complex problem of visual scene understanding has been tackled on various different levels and from different angles. However, a common scheme that can be found in most approaches is the utilization of what is generally referred to as *scene context* [1, 31, 71, 79, 80, 99].

We divide this literature analysis into three sections. Beginning with an overview of earlier scene understanding methods mainly based on 2D images, we then shift the focus to more recent work similar to our methods, operating on RGB-D data. Finally, we also include a review of novel approaches, particularly for semantic segmentation, based on deep learning architectures such as convolutional neural networks (CNN).

Since all of these methods require a powerful GPU to achieve competitive runtimes, from a robotics perspective, their applications are still limited. Nevertheless, their results set the new benchmark in many fields of computer vision.

1.4.1 Scene Understanding on 2D Images

Before the rise of cheap RGB-D sensors, such as the Microsoft Kinect, and geometric information of a scene was not directly accessible, scene understanding methods mainly operated on 2D images. In most approaches, the motivation to exploit contextual information was to improve the performance of object detection or recognition. In order to gain meaningful geometric information of a scene, many methods also focused on the accurate estimation of the scene or room layout.

In [87], scene context is used to improve a multi-class object recognition system. A complex hierarchical, generative model is introduced, incorporating the parts composing the target objects, the relative spatial relationships between the objects and the scene surrounding them. The parameters of the model are learned using a Gibbs sampling algorithm [23].

Exploited in a very narrow sense, in [99] contextual information boosts the accuracy of a license plate recognition framework. In a probabilistic model of a composition machine, different parts of the scene and specifically of license plates are modeled, incorporating different feature responses.

In [34], context refers to a global understanding of the scene composition with regard to the perspective, the surface orientations and the horizon in the scene. Using this information, person detections in street scenes are filtered, leaving only candidates that physically make sense. In a follow-up work [35], the authors add even more input channels and show that these sets of global properties complement each other, jointly solving for surface orientations, occlusion boundaries, objects, camera viewpoint and relative depth.

The work of [31] again focuses on improving object detections by incorporating context from informative regions extracted from the scene. A Bayesian network is formulated for each image, linking specific classes of an object detector such as “bike” or “car” to classes describing environments such as “roads”, “bushes” etc. The parameters of the model are learned using the Expectation-Maximization (EM) algorithm [13].

For the particular task of semantic segmentation, an approach for context modeling related to our first pipeline is used in [71]. The authors use a densely connected CRF to incorporate semantic information between segments obtained from a previous segmentation stage. However, their pairwise model is defined only by a co-occurrence count, which does not consider any spatial information.

A more complex CRF model is applied in the work of [80]. Their energy function consists of four major terms, including spatial and appearance information captured by a TextonBoost classifier [79] and Gaussian Mixture Models (GMMs). The CRF parameters are learned using a piecewise learning approach. Regarding computation time, for an image of size 320×240, the overall runtime stated is $5 - 6$ seconds.

A region based approach for scene understanding is presented in [26]. Similar to the work of [80], their method is based on the optimization of a unified energy function, which scores the entire description of the input scene. This includes pixel-to-region association weights, semantic label proposals for regions, geometric relations, appearances and the location of the horizon. The individual components are computed either by separate classifiers such as a boosting method or a logistic classifier or by simple potential functions. Since exact inference of this huge function is intractable, an approximative greedy method is presented to obtain a result. Nevertheless, because of the model complexity, computation on a single image can still take— depending on the scene configuration—between 30 seconds and 10 minutes.

Generally it should be noted that none of the mentioned methods have been developed focusing on computational efficiency. Consequently, in a robotics scope, where computational resources are limited and runtime is a very crucial factor, these methods are not feasible.

1.4.2 Scene Understanding on RGB-D Data

For mobile robots operating in indoor environments, RGB-D sensors such as the Microsoft Kinect have become a popular choice to equip the visual perception system. Especially since the publication of large scale datasets of RGB-D data by [1, 81, 82], many different approaches tackling the problem of semantic segmentation have been presented.

Anand et al. [1] learn a large graphical model capturing local appearance cues as well as complex pairwise co-occurrence and geometric relationships, achieving strong results on their own dataset. On the dataset of [81], the benchmark is still represented by [72], a system combining segmentation trees, kernel descriptors and a Markov Random Field (MRF). Gupta et al. [27] use a hierarchical segmentation approach using contour detection, a comprehensive set of complex features and amodal completion to classify 3D surfaces. In [28], they extend their framework by adding more descriptive features calculated by a CNN.

Classifying a point cloud on a coarse voxel grid, [39] models the segmentation problem with a CRF, enforcing consistency through higher order potentials initialized by several input channels such as plane and object detectors and kernel descriptors from [72]. A similar approach is presented by [38], where planar regions are used to form cliques, defining higher order potentials in a CRF. The principle of stacked predictors is used to build Spatial Inference Machines in [75], where the contributions of several RF classifiers are weighted in a factor graph, modeling contextual dependencies.

All of the approaches described above have in common that their models account for various forms of semantic context. However, again none of them are focused on computational performance, which leaves them aside for the online application on a mobile robot.

In contrast, the first deep learning methods of [9, 17, 33] achieve fast inference times, but the labeling performance of their learned CNNs clearly fall short compared to other work. Coming with the drawback of requiring a high-performance GPU for fast computation, recent developments have shown that deep learning methods are also very powerful for semantic segmentation, and we will present promising approaches in the next section.

To our knowledge, the only comparable approaches to our methods—with regard to performance and runtime—are from Hermans et al. [32], Müller et al. [63] and Stückler et al. [84, 85]. With the carefully designed framework of [32], based on RFs and CRFs, scenes can be incrementally labeled online. However, their applied CRFs only enforce spatial and temporal consistency, are manually defined and do not incorporate additional contextual information.

In [84], the authors use a GPU implementation of an RF classifier for object segmentation, which they extend in [85] to a real-time-capable system for 3D semantic mapping. However, they only present results for the simplest task including only four different structural labels, and it is unclear how the system scales on larger label sets.

The GPU RF implementation of [84] is also used in [63]. Similar to our method, in their work, a CRF with learned potentials is applied to refine the coarse predictions of the RF, however, it is unclear if the CRF is densely connected or only establishes pairwise terms between adjacent segments.

Our first pipeline, described in chapter 3, is based on a similar architecture to [32] and [63], with an RF classifier providing the initialization for a dense CRF to refine the result taking global contextual information into account. However, contrasting both methods, we use a different set of features calculated in 3D space and more complex pairwise terms in the CRF, such that our approach is able to model more expressive pairwise relations between different labels. Exploiting a very efficient inference procedure for the dense CRF, our method shows competitive runtimes of two frames per second, even though it does not require a GPU.

Our new 3DEF classifier in turn does not require a separate refinement step with a CRF. The 3DEF encapsulates the principle of hierarchical prediction in a single RF, and therefore combines the efficiency of RFs with the capability of hierarchical frameworks to explicitly capture contextual and spatial label relations. The concept of entangled features for RFs has been introduced by Montillo et al. [61] to automatically segment 2D grayscale images produced by CT scans. In chapter 4, we describe our 3DEF approach, containing a new three-dimensional form of those features.

1.4.3 Semantic Segmentation using Deep Learning

After setting new impressive benchmarks for several computer vision tasks such as object detection and classification [25, 44, 83, 88], approaches using deep learning architectures, particularly CNNs, have recently also raised the bar in the field of semantic segmentation.

In [69], Pinheiro et al. present a recurrent convolutional neural network (RCNN) that is able to iteratively add contextual information from long range pixel label dependencies, operating directly on the pixel level. Because results from CNNs are not sufficiently localized for accurate segmentation, the work of [7] combines a CNN with a dense CRF with manually defined potentials and parameters for refinement.

In [8], the authors extend their work and improve the segmentation accuracy of their CNN.

Zheng et al. [101] introduce an interesting approach that seamlessly integrates the advantages of a dense CRF into a CNN. The authors formulate the CRF model from Krähenbühl and Koltun [42] using Gaussian kernels and mean-field approximation for inference as an RCNN. Consequently, the whole network, including a traditional CNN and the dense CRF, can be trained end-to-end applying the back-propagation algorithm.

The work of [53] introduces fully convolutional networks (FCNs), which are able to produce accurate pixel-wise semantic segmentation results of the same size as the input layer. The authors cast the successful object classifier from [44] into FCNs, which are then applied to feature maps generated by different network layers. Their architecture also allows for end-to-end training of the whole pipeline.

Some of the limitations of [53], such as a fixed-size receptive field, which can lead to fragmented results for very large or very small objects, are tackled by the work of [64]. In contrast to combining feature maps from different layers in a final prediction layer, in [64] a deep deconvolution network is learned to perform the non-linear upsampling.

A similar approach is presented in [2]. In this work, the authors particularly focus on memory efficiency and the accuracy trade-offs that come with different upsampling techniques in the network.

In [14], the localization problem is tackled by a sequence of CNNs operating on different scales. While achieving good results on several different training sets, the resulting system is very complex and difficult to train.

1.5 List of Publications

Parts of the content presented in this thesis have been previously published in the following articles:

> *Daniel Wolf, Markus Bajones, Johann Prankl and Markus Vincze.* **Find my mug: Efficient object search with a mobile robot using semantic segmentation**. In Workshop of the Austrian Association for Pattern Recognition (ÖAGM), 2014. [95]

> *Markus Bajones, Daniel Wolf, Johann Prankl and Markus Vincze.* **Where to look first? Behavior control for fetch-and-carry missions of service robots**. In Austrian Robotics Workshop (ARW), 2014. [3]

> *Daniel Wolf, Johann Prankl and Markus Vincze.* **Fast Semantic Segmentation of 3D Point Clouds using a Dense CRF with Learned Parameters**. IEEE International Conference on Robotics and Automation (ICRA), 2015. [96]

Daniel Wolf, Johann Prankl and Markus Vincze. **Enhancing Semantic Segmentation for Robotics: The Power of 3-D Entangled Forests**. Robotics and Automation Letters, IEEE, vol.1, no.1, 2016. [97]

1.6 Outline

The remainder of this thesis is organized as follows: The next chapter is dedicated to an introduction to RFs for classification, since this powerful machine learning tool serves as the backbone for both of our presented pipelines. We will present the general idea and architecture behind the approach, followed by a discussion of its properties and the training and inference procedures.

In chapter 3, we describe our first approach for contextual learning in semantic segmentation using an RF classifier and a dense CRF. Chapter 4 then presents our second method, which is based on our new 3DEF classifier. We introduce the concept of entanglement and discuss its advantages compared to the standard RF architecture.

Both of our methods are then thoroughly evaluated on different datasets and label configurations in chapter 5. We compare both pipelines to each other and to several other recent methods for semantic segmentation. Moreover, we particularly analyze the capability of our methods to learn and model contextual information of scenes and how this improves the overall performance.

Finally, we present the conclusions achieved in this thesis and provide possible directions for further developments in the field of semantic scene understanding in chapter 6.

Chapter 2

Random Forests for Classification

Random forests [6] (also referred to as *randomized decision forests*) are a very powerful and flexible machine learning tool and have been the method of choice for many different computer vision problems, ranging from keypoint tracking [48], detection [11, 21], regression [60], pose estimation [16] to (semantic) segmentation [24, 61, 65, 68, 77]. However, they have got most attention in the field of classification [4, 46, 47, 55, 59, 62, 100], particularly since RFs have been used in the popular body part recognition system of the Microsoft Kinect [76, 78].

The general concept behind RFs for classification is to solve the complex classification decision by breaking it up into a series of simpler tests. Each test output gets the classifier one step closer to a reliable decision about the data at hand. During training, the most suitable tests to separate differently labeled training data from another are learned. At test time, a previously unseen data point then runs through the learned tests and is finally assigned the class label distribution of the training data subset that produced the same test results.

This architecture leads to the following desirable properties of RFs for classification tasks:

- they offer a probabilistic output;
- they are inherently capable of handling multiple labels;
- they can be dynamically pruned at inference time to find a suitable trade-off between accuracy and runtime;
- they easily handle different scales of different feature dimensions;
- their simple tests are very efficient to compute and both training and inference procedures are perfectly suited for parallelization.

In this chapter, we will give a brief introduction to the theoretical background behind RFs for classification. For a broader overview and in-depth analysis of different use-cases and configurations, we refer the reader to the comprehensive work of Criminisi et al. [10].

2.1 The Decision Tree Model

An RF $\mathcal{F}$ is composed by an ensemble of different random trees $\mathcal{T}_l$, which are in fact (generally binary) decision trees. The term “random” highlights a particular step of the learning method, where randomization is injected for generalization.

A decision tree is a hierarchical graph without loops. Thus, it consists of a set of nodes $\mathcal{N} = \{\boldsymbol{n}_k\}$ and directed edges, where each edge connects a parent node from one depth level d to a child node from the next depth level $d + 1$. During training and inference, a data point $\boldsymbol{x}$ from a training or test set $\mathcal{X}$ traverses a tree along the edges, starting from the first parent node or root node $\boldsymbol{n}_0$ on level 0.

A node $\boldsymbol{n}_k$ can be either a split node with two child nodes $\boldsymbol{n}_{k,l}$ and $\boldsymbol{n}_{k,r}$ or a leaf node without any child nodes. A split node $\boldsymbol{n}_k$ stores a split function $\rho_k(\boldsymbol{\theta}_k, \boldsymbol{x})$ with a parameter set $\boldsymbol{\theta}_k$ that is learned during training and evaluates on a given data point $\boldsymbol{x}$ to either true or false. If evaluating to true, the respective data point traverses to the left child node $\boldsymbol{n}_{k,l}$, otherwise it continues to the right child node $\boldsymbol{n}_{k,r}$. A leaf node $\boldsymbol{n}_k$ in turn stores a discrete label distribution $p(\mathcal{C}|\mathcal{X}_k)$, where $\mathcal{C} = \{c\}$ denotes the set of all class labels and $\mathcal{X}_k$ is the subset of training data which has reached the leaf node. This label distribution is assigned to every data point reaching the leaf node during inference.

An example of a decision tree is shown in Figure 2.1. In the next sections, we will describe the training and inference procedures in more detail.

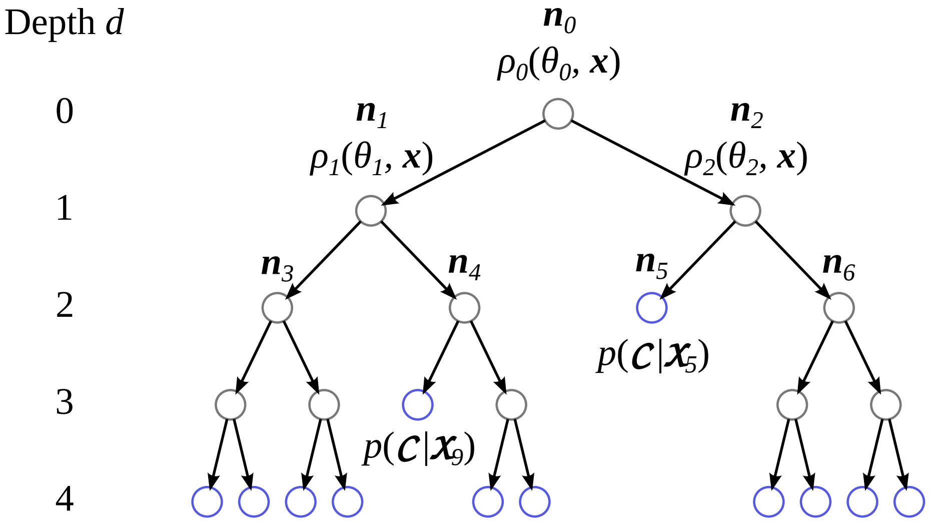

Depth d
0
1
2
3
4
$\boldsymbol{n}_0$
$\rho_0(\theta_0, \boldsymbol{x})$
$\boldsymbol{n}_1$
$\rho_1(\theta_1, \boldsymbol{x})$
$\boldsymbol{n}_2$
$\rho_2(\theta_2, \boldsymbol{x})$
$\boldsymbol{n}_3$
$\boldsymbol{n}_4$
$\boldsymbol{n}_5$
$\boldsymbol{n}_6$
$p(\mathcal{C}|\mathcal{X}_5)$
$p(\mathcal{C}|\mathcal{X}_9)$

Figure 2.1: An example of a trained decision tree with maximal depth $d = 4$. Split nodes are shown in gray, leaf nodes are shown in blue color.

2.2 Random Forest Training

The training procedure for an RF $\mathcal{F} = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_t\}$ is split into the training procedures of the individual trees composing the forest. In general, the number of trees a forest contains, denoted by t, is predefined (a discussion about the influence of t on performance is given in section 2.3.1). Since each tree can be trained independently from the other trees, the procedure is very easy to parallelize.

In the beginning, a tree is initialized with a new root node $\boldsymbol{n}_0$, and all available training data $\mathcal{X}$ for this tree is assigned to that node, such that $\mathcal{X}_0 = \mathcal{X}$. In the next step, the goal is to find a split function $\rho_0(\boldsymbol{\theta}_0, \boldsymbol{x})$, which divides $\mathcal{X}_0$ into two parts $\mathcal{X}_{0,l}$ and $\mathcal{X}_{0,r}$, such that $\mathcal{X}_0 = \mathcal{X}_{0,l} \cup \mathcal{X}_{0,r}$, in a way that separates data points with different class labels "best" from another.

To that end, a defined number of s split function candidates is randomly drawn from a pool of available functions, and the best candidate is found by maximizing an objective function ξ that evaluates the quality of a split:

$$\rho_0(\boldsymbol{\theta}_0, \boldsymbol{x}) = \underset{\rho(\boldsymbol{\theta}, \boldsymbol{x})}{\arg\max}\ \xi\left(\mathcal{X}_0, \rho\left(\boldsymbol{\theta}, \boldsymbol{x}\right)\right), \quad \boldsymbol{x} \in \mathcal{X}_0 \tag{2.1}$$

In practice, this optimization is implemented as a simple search, thus by evaluating ξ on all split function candidates for the training data points $\boldsymbol{x} \in \mathcal{X}_0$ assigned to the node. The best split function is then stored in the split node and two child nodes $\boldsymbol{n}_1 = \boldsymbol{n}_{0,l}$ and $\boldsymbol{n}_2 = \boldsymbol{n}_{0,r}$ are created on the next depth level. Additionally, the training data subsets $\mathcal{X}_1 = \mathcal{X}_{0,l}$ and $\mathcal{X}_2 = \mathcal{X}_{0,r}$, resulting from the learned split, are assigned to the corresponding child nodes. The idea of the split procedure is visualized in Figure 2.2, more details on split functions and the objective function are given in sections 2.2.1 and 2.2.2.

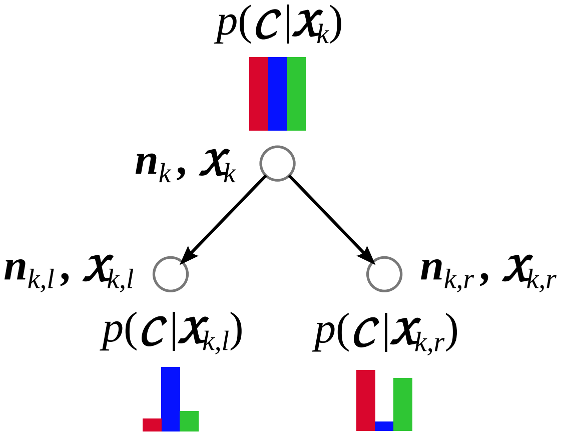

$p(\mathcal{C}|\mathcal{X}_k)$
$\boldsymbol{n}_k, \mathcal{X}_k$
$\boldsymbol{n}_{k,l}, \mathcal{X}_{k,l}$
$\boldsymbol{n}_{k,r}, \mathcal{X}_{k,r}$
$p(\mathcal{C}|\mathcal{X}_{k,l})$
$p(\mathcal{C}|\mathcal{X}_{k,r})$

Figure 2.2: A split function dividing a dataset containing three different labels red, blue and green into two disjunct subsets. The color bars visualize the individual label distributions at each node. In this example, the majority of data points with the blue label is separated from the data points labeled red and green.

This node expansion procedure is repeated for all newly generated child nodes, either in a breadth- or depth-first fashion (the former, however, is better suited for parallelization). A new child node $\boldsymbol{n}_i$ is converted to a leaf node, if at least one of the following conditions are fulfilled:

1. The class label distribution of the training data subset $\mathcal{X}_k$ is pure, i.e. it only contains data points from the same class label.

2. The node is at a defined maximal depth-level d_{max}:

$$d(\boldsymbol{n}_k) = d_{max} \tag{2.2}$$

3. The value of the objective function for the best achievable split is below a defined threshold:

$$\max_{\rho(\boldsymbol{\theta},\boldsymbol{x})} \xi\left(\mathcal{X}_k, \rho_k\left(\boldsymbol{\theta}_k, \boldsymbol{x}\right)\right) < \xi_{min} \tag{2.3}$$

4. The size of the training data subset $\mathcal{X}_k$ assigned to the node is below a defined threshold p_{min}:

$$|\mathcal{X}_k| < p_{min} \tag{2.4}$$

It should be noted that conditions 2 to 4 are all optional and can be implemented independently from each other.

If a node $\boldsymbol{n}_k$ is converted to a leaf node, it stores the discrete distribution of the class labels $\mathcal{C}$ in the training data subset $\mathcal{X}_k$ that has reached the node, denoted as $p(\mathcal{C}, \mathcal{X}_k)$. The training procedure for a tree stops if all child nodes on the last level have been converted to leaf nodes.

2.2.1 Split Functions

In the general decision tree model, a split function $\rho(\boldsymbol{\theta}, \boldsymbol{x})$ is a binary test function with a set of parameters $\boldsymbol{\theta}$, which accepts a data point (respectively feature vector) $\boldsymbol{x}$ as input and evaluates to either true or false as output. While a data point can theoretically have infinite dimensions, a split function is generally limited to a very few (≤ 2), such that parameter sampling can be done efficiently. A thorough analysis on different split function models is presented in [10].

Focusing on computational efficiency, in our work we apply the simplest split function model: axis-aligned hyperplanes. This causes the parameter set $\boldsymbol{\theta}$ to consist only of two parameters: (1) a "selector" index j defining the feature vector dimension to split on and (2) a threshold τ for this dimension. More formally, the split functions used in our RF implementation are defined as follows:

$$\rho(\boldsymbol{\theta}, \boldsymbol{x}) = \rho(j, \tau, \boldsymbol{x}) = \begin{cases} \text{true} & x_j > \tau, \\ \text{false} & \text{otherwise} \end{cases} \tag{2.5}$$

During training, first j is randomly drawn from a uniform distribution over all available feature vector dimensions. Then, the range of the feature values along this dimension $[x_{j,min}, x_{j,max}]$ is determined and τ is sampled from a uniform distribution over this range. More details about the random sampling of split functions are given in section 2.2.3.

2.2.2 Objective Function for Split Evaluation

Determining the "best" split of a given training dataset is a crucial step during the training procedure. For a classification task, a good split of a dataset $\mathcal{X}$ means that data points with different ground truth labels are well separated into the two subsets $\mathcal{X}_l$ and $\mathcal{X}_r$ (with $\mathcal{X} = \mathcal{X}_l \cup \mathcal{X}_r$). In other words, the class label distributions in $\mathcal{X}_l$ and $\mathcal{X}_r$ are more pure and contain less uncertainty than the distribution of $\mathcal{X}$.

Measuring the uncertainty of a distribution, the *entropy* H is the commonly used metric to determine the quality of a split during random forest training. It is defined as follows:

$$H(\mathcal{X}) = \sum_{c \in \mathcal{C}} -p(c|\mathcal{X}) \log \left(p \left(c|\mathcal{X} \right) \right), \tag{2.6}$$

where $p(c|\mathcal{X})$ denotes the distribution of class labels $c \in \mathcal{C}$ in a training (sub)set $\mathcal{X}$.

After a "good" split, the cumulated entropy of the distributions of the training subsets of the child nodes will be significantly lower than the entropy of the overall training set of the parent node. This difference is also denoted *information gain* and the objective function measuring the quality of a split:

$$\xi = H(\mathcal{X}) - \sum_{k \in [l,r]} \frac{|\mathcal{X}_k|}{|\mathcal{X}|} H(\mathcal{X}_k) \tag{2.7}$$

The entropies of the training sets of the child nodes are weighted by their cardinality to prevent unbalanced splits with very few points in one subset.

2.2.3 Injecting Randomness for Generalization

Similar to other classifiers, also RFs suffer from overfitting. A common way to avoid this effect and improve generalization is to decorrelate the individual trees from another by injecting randomness at certain stages of the training procedure:

Bagging

As a first step, so-called bagging can be applied when assigning training data to a newly initialized tree. Bagging was first introduced in [5, 6] and means that each tree is only trained on a smaller, randomly sampled subset (a bag) of all available training data. Consequently, training is not only faster, but individual trees also get decorrelated from each other. The bagging ratio b defines the fraction of training data to be used for each tree.

One apparent downside of the bagging technique is that not all training data is utilized in a tree, which, besides the fact that labeled data is usually expensive and should not be "wasted", can particularly cause problems if the dataset is imbalanced and some classes are highly underrepresented. One way to counter this issue is a recalculation of all class label distributions in the leaf nodes after training is done, using all training data. That is, after training, the learned tree structure and the split functions remain fixed, but the whole training dataset is parsed through each tree and the leaf node distributions are updated, depending on which data points reach which leaf node. A hybrid method is also possible, where the new distribution is a weighted average between the distribution learned during training and the one acquired using all data points.

Randomized Split Function Sampling

Of course, the number of sampled split function candidates per node highly influences the correlation of different trees. The fewer split function candidates are sampled, the less similar individual trees will be. However, if the number of sampled candidates is too low, it is likely that no good split is found at all and node expansion is stopped too early.

We divide the number of sampled split function candidates into two parameters:

1. The number of sampled feature selectors j, denoted by f.

2. The number of sampled thresholds per feature τ, denoted by h.

An additional layer of randomness can be introduced by limiting the available pool of features at each node to a randomly sampled subset of feature dimensions. The ratio r defines the fraction of available feature dimensions at each node.

2.2.4 Coping with imbalanced Data Sets

Especially in the scope of semantic segmentation, labeled datasets are often highly imbalanced. Some classes are only present in a few scene types, while others will be part of every scene, e.g. bed or TV compared to floor or wall. Additionally, the sizes of the objects corresponding to the various classes can be vastly different. A wall or a bookshelf occupies a large portion of the scene and consists of up to tens of thousands of 3D points, while smaller objects lying on a table will only be represented by a few hundred points.

Such imbalances can be problematic for the evaluation to find the optimal split as well as for the resulting label distributions in the leaf nodes, in which underrepresented class labels might never end up being the most likely label.

There are two common methods to counter this problem:

1. If bagging is applied, uniformly sample training data points from each class label while selecting bags.

2. Reweigh the contribution of each individual class label to the class label distribution in the leaf nodes by the inverse frequency of the respective label in the training dataset. The same weighting can also be applied in the evaluation of the objective function to find the best split node.

2.3 Random Forest Inference

After the training procedure has finished, the tree structure, as well as the split functions in the split nodes and the class label distributions in the leaf nodes are fixed. At inference time, a data point $\boldsymbol{x}$ is then parsed through every tree of the forest, starting at the root node. The path through the trees follows the evaluations of the respective split functions in the split nodes until a leaf node is reached in every tree.

The final class label prediction $p(\mathcal{C}|\boldsymbol{x})$ for $\boldsymbol{x}$ is then obtained combining the individual class label distributions $p(\mathcal{C}|\mathcal{X}_k)$ stored in the reached leaf nodes. A common method to calculate the final output is simple averaging of all distributions, which is also the method of choice in this thesis:

$$p(\mathcal{C}|\boldsymbol{x}) = \frac{1}{t} \sum_{\boldsymbol{n}_k \in \mathcal{L}} p(\mathcal{C}|\mathcal{X}_k), \tag{2.8}$$

where t is the number of trees in the forest and $\mathcal{L}$ is the set of all leaf nodes $\boldsymbol{x}$ has reached. The inference procedure is visualized in Figure 2.3.

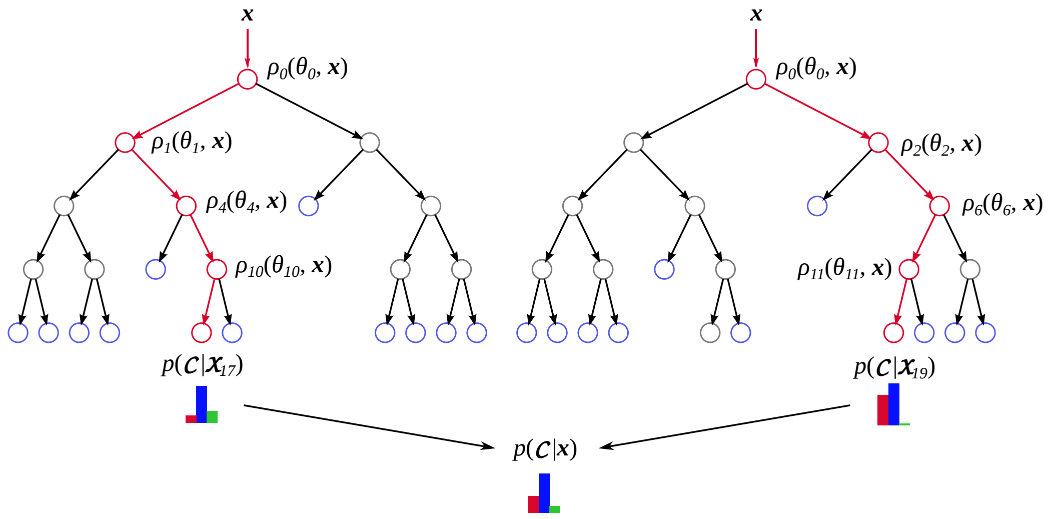


Figure 2.3: Visualization of the inference procedure for an RF with 2 trees and three available labels red, blue and green. A data point $\boldsymbol{x}$ is passed through each tree and follows a path defined by the evaluated split functions (shown in red). After reaching a leaf node in each tree, the classification result is obtained by averaging over the label distributions of the reached leaf nodes. In this example, $\boldsymbol{x}$ will be classified as blue.

2.3.1 Dynamic Pruning at Inference Time

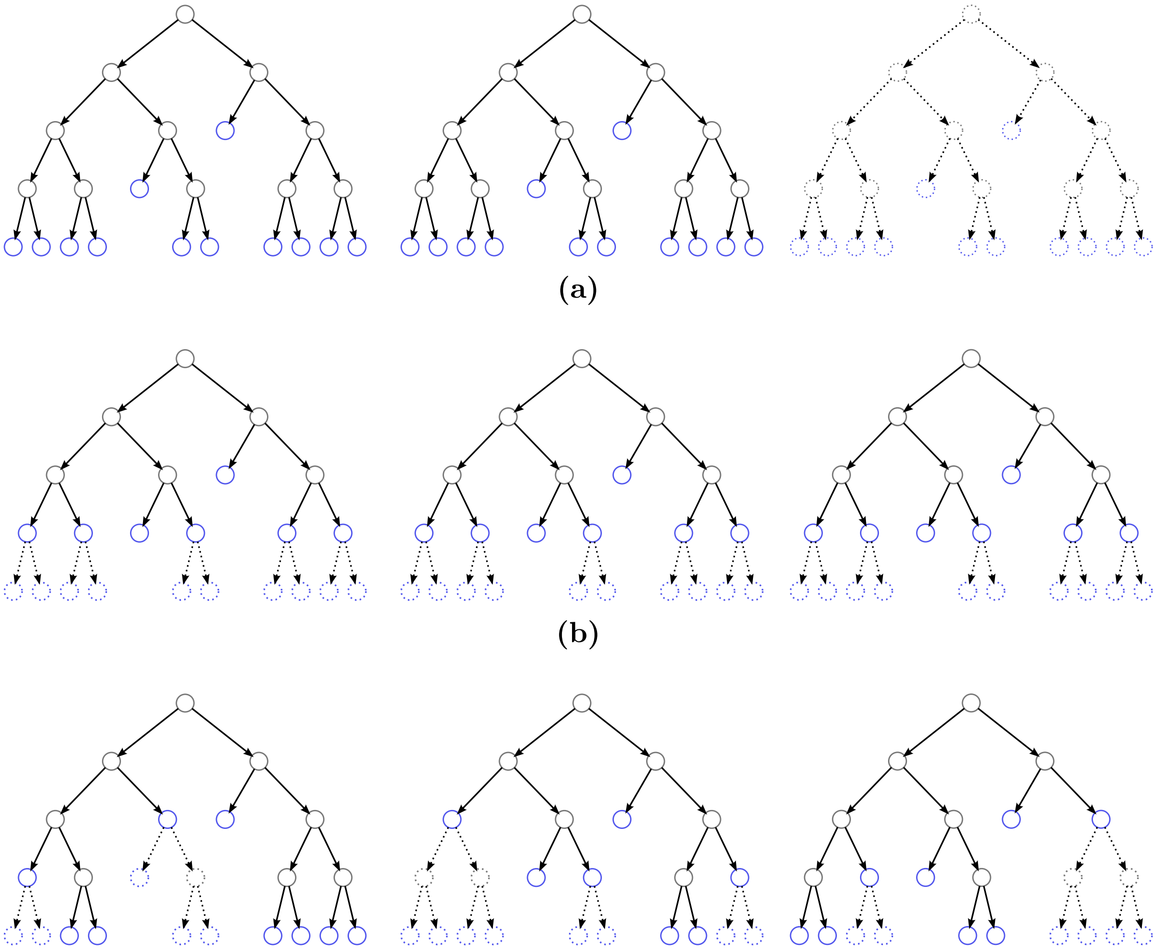
One key feature of RFs is their flexibility at inference time. Even though after training the classifier structure remains fixed, its modular nature allows for different configurations, such that an RF can basically be pruned at inference time to find the best trade-off between classification accuracy and runtime for the task at hand.

First, the number of trees considered during inference can be varied between 1 and t. Since classification performance might saturate after a certain number of added trees and runtime scales linearly with the number of trees, this parameter can easily be optimized for best performance.

It is further useful to store the class label distributions $p(\mathcal{C}|\mathcal{X}_k)$ for all split nodes as well. This enables the option to treat a split node as a leaf node and therefore dynamically prune the forest to a maximum used tree depth between 1 and the maximum learned tree depth d_{max} at inference time. This not only affects runtime exponentially, but is also a method to avoid the negative influence of possible overfitting effects at deeper nodes.

Additionally, the achieved information gain of a node, achieved during training, can be stored in each split node as well. Setting a minimal threshold at inference time enables an alternative method to control the used "depth" of the inference procedure.

All three pruning methods are visualized in Figure 2.4.



(a)

(b)

(c)

Figure 2.4: Different methods to dynamically prune an RF at inference time. Dotted lines indicate nodes not considered during inference due to pruning. (a) shows pruning using a smaller number of trees, (b) pruning by a maximum depth level and (c) shows pruning by minimal achieved information gain. The latter seems to be the most reasonable method, however, the minimum information gain is not an intuitive parameter to define. Furthermore, this pruning method can result in very imbalanced trees, such that inference times vary a lot for different data points, depending on their path through the trees.

Chapter 3

Context Learning with a Dense CRF

In this chapter, we describe our first semantic segmentation pipeline, based on an RF classifier and a densely connected CRF incorporating contextual information.

An overview of the general architecture of the approach is given in Figure 3.1. In a first step, the raw RGB-D input point cloud is transformed into a voxel representation with a defined minimum voxel size. For the created voxelized point cloud we then compute an oversegmentation, such that the scene is clustered into small segments of voxels with similar appearance. In the next step, a feature vector, which captures shape and appearance properties, is extracted for each segment and then processed by a pretrained RF classifier. For each segment and label, the classifier outputs conditional label probabilities, which are used in the final step to initialize the unary potentials of a dense CRF.

In the CRF model, we define pairwise smoothness potentials to locally reduce the noise of the classifier stage. Furthermore, as the key element of the model, we add long-range pairwise potentials incorporating the likelihood of the co-occurrence of different classes in the scene. Since these potentials also depend on the geometric properties and the appearance of the corresponding 3D points, this enables the CRF to learn and model scene context, which is essential to resolve ambiguous classification results. The pairwise potentials consist of label compatibility terms, which are entirely learned from training data.

The overall labeling result is finally obtained by a minimization of the CRF energy, which can be efficiently computed using mean field approximation. Our whole pipeline has been carefully designed for computational efficiency, achieving runtimes of approximately two Kinect frames per second on a regular laptop.

In the following sections, we describe the separate stages of our method in more detail.

3.1 Oversegmentation

To be able to compose a robust and meaningful set of features for the RF classifier, we do not calculate a feature vector for every single 3D point of the input point cloud,

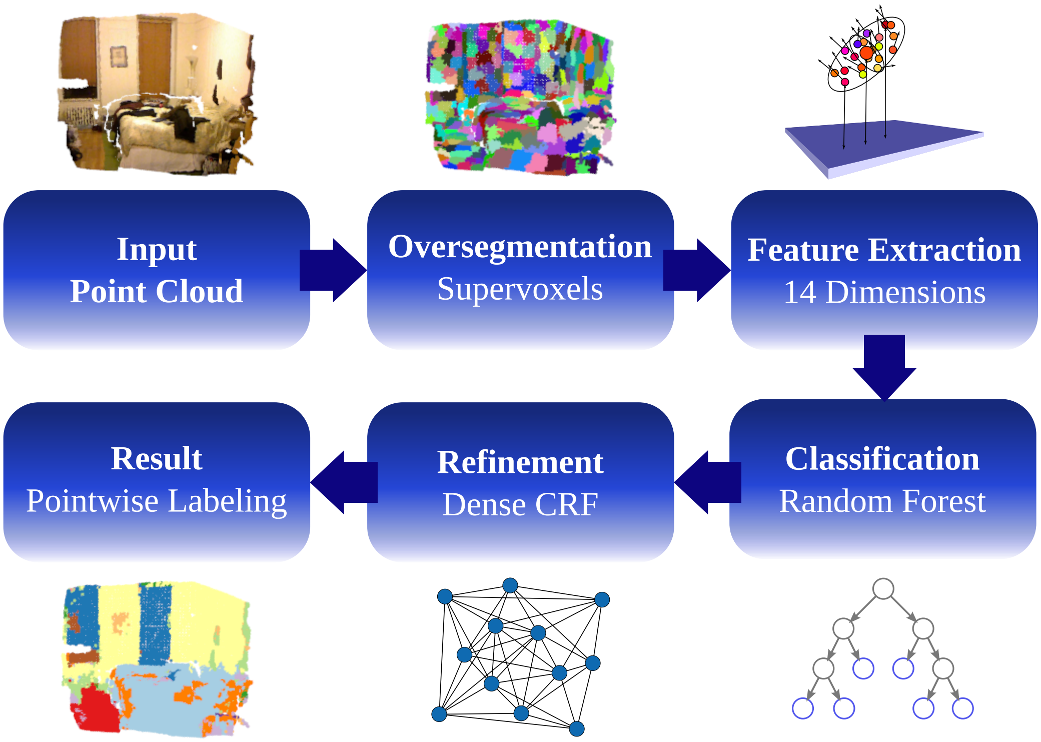


Figure 3.1: Overview of our semantic segmentation pipeline based on an RF classifier and a dense CRF for contextual refinement.

but for whole segments consisting of several points which likely belong to the same class. Assuming that the points of such a segment have a similar appearance, we first calculate an oversegmentation of the input point cloud, which takes color and surface orientation into account to group adjacent points together.

To this end, we apply the Voxel Cloud Connectivity Segmentation (VCCS) method presented by Papon et al. [66], which is publicly available in the Point Cloud Library (PCL)[1] [73].

VCCS applies a region growing k-means [52, 54] variant directly in 3D space. Since seeding takes place in 3D instead of 2D compared to other segmentation methods, the resulting segments (also referred to as “supervoxels”) are evenly distributed across the scene.

The input point cloud is transformed into an underlying octree [57] structure, which is used to enforce spatial connectivity of the points of a supervoxel in 3D space. Consequently, supervoxels adhere well to object boundaries.

Since VCCS works on a voxel representation of the input point cloud, its accuracy is specified by the voxel size r_v. Additionally, the maximum cluster size r_c and three weighting parameters w_c, w_n and w_g, controlling the influence of color, the surface normals and geometric regularity of the clusters, have to be defined. An example of a VCCS oversegmentation result can be seen in Figure 3.2.

---

[1] `http://www.pointclouds.org`

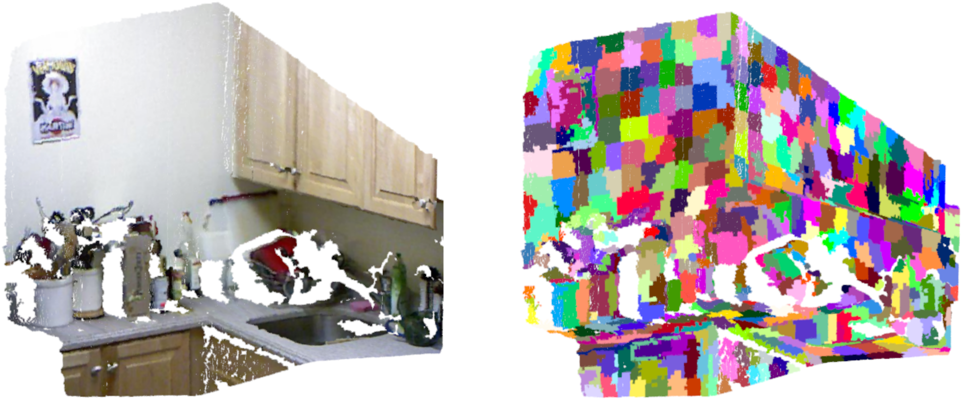


Figure 3.2: Example result of the VCCS method [66] used by our pipeline, applied on a kitchen scene. The voxel size r_v is set to 1 cm and the maximum cluster size r_c to 10 cm. The respective weights for color, surface normals and geometric regularity are 0.6, 0.1 and 1.0.

3.2 Feature Extraction

For each of the segments generated by the oversegmentation, we calculate a feature vector $\boldsymbol{x}$ that takes color information as well as geometric properties of the segment into account. To this end, we carefully selected 14 different features, which on one hand capture the distinct properties of the different classes, but on the other hand are also very efficient to compute. As a consequence, we refrain from including more complex features, such as HOG [12], spin images [36] or textons [49, 79].

Two very descriptive features of a segment are the enclosed angle between its surface normal and the ground plane, as well as its height above the ground plane. Both features can easily be computed, because in a robotics setup, we assume that the camera pose with respect to the ground plane is known.

For the angular feature, we both compute the average normal angle as well as its standard deviation, computed over all 3D points of the segment. This is done using directional statistics:

$$Y = \frac{\sum_{i=1}^{n} \sin(\alpha_i)}{n}, \qquad X = \frac{\sum_{i=1}^{n} \cos(\alpha_i)}{n}, \qquad r = \sqrt{X^2 + Y^2}, \tag{3.1}$$

$$\theta = \arctan\left(\frac{Y}{X}\right), \qquad \sigma = \sqrt{-2\,\log(r)}, \tag{3.2}$$

where α_i is the angle between a point's surface normal and the upright vector of the scene, n is the number of points of the segment and θ and σ are the resulting mean angle and its standard deviation, respectively.

The height feature is threefold and consists of the heights of the lowest and highest point of the segment as well as the segment centroid, all of which computed with respect to the ground plane. We further calculate six color features in the CIELAB color space, namely mean and standard deviation for each of the three channels, computed over all points of a segment. We prefer CIELAB over RGB since it separates the luminance from the color values.

Inspired by [58], we complete our feature set with three spectral features *compactness* c, *planarity* p and *linearity* l, which are computed from the eigenvalues λ_0, λ_1, λ_2 of the scatter matrix of the segment:

$$c = \lambda_0, \qquad p = \lambda_1 - \lambda_0, \qquad l = \lambda_2 - \lambda_1, \qquad \lambda_0 \leq \lambda_1 \leq \lambda_2. \tag{3.3}$$

A summary of all computed features is given in Table 3.1.

Table 3.1: List of all unary features calculated for a 3D segment and their dimensionality. $\lambda_0 \leq \lambda_1 \leq \lambda_2$ are the eigenvalues of the scatter matrix of the segment.

Feature	Dimensions
Compactness (λ_0)	1
Planarity ($\lambda_1 - \lambda_0$)	1
Linearity ($\lambda_2 - \lambda_1$)	1
Angle with ground plane (mean and standard deviation)	2
Height (top, centroid, and bottom point)	3
Color in CIELAB space (mean and standard deviation)	6
Total number of features	14

3.3 Random Forest Classifier

To calculate label predictions $p\left(c|\boldsymbol{x}\right)$ for each class label $c \in \mathcal{C} = \{c_1, \ldots, c_M\}$ and scene segment based on its feature vector $\boldsymbol{x}$, we use a standard RF classifier as described in detail in chapter 2. RFs have the advantage that they can cope with different types of features without the need for any further preprocessing (e.g. normalization) of the feature vector. Furthermore, the intuitive training and inference procedures can be highly parallelized and the obtained output for the input vectors are probabilistic label distributions that in turn directly define the unary potentials of the CRF model described in section 3.4.

3.3.1 Training

Since available training data for semantic segmentation is very limited, we augment the used datasets to train the RFs, which means that we artificially extend the set of available data. Because our used oversegmentation method (refer to section 3.1) is based on an octree representation of the point cloud, all points are assigned to a bin (= a voxel) on a three-dimensional grid, depending on their position relative to the camera. Consequently, if the input point cloud is slightly rotated, some points will be assigned to a different bin, which in turn leads to a slightly different segmentation result. We exploit this property to augment our training set by adding 10 additional examples per training point cloud by mirroring and applying small random rotations about each axis before applying segmentation.

We adapt the default training procedure for RFs used for classification, presented in chapter 2, to our application. To increase generalization of the classifier, we apply the bagging technique and because the available datasets have a few dominant and many underrepresented classes (e.g. *wall* resp. *object*), we calculate individual class weights corresponding to the inverse frequency of the class labels in each bag. These weights are taken into account when the information gain is evaluated at each split node. Furthermore, we also recalculate the final label distributions in the leaf nodes of the trees according to the class weights.

As termination conditions for the training procedure we implemented all of the options described in chapter 2. Thus the node expansion ends if:

1. the label distribution in a node is pure,
2. the best achievable information gain is below ξ_{min},
3. the minimum number of data points to split is below p_{min} or
4. the tree depth has reached d_{max}.

The exact values selected for these thresholds are given in the experimental setup section of chapter 5.

3.3.2 Inference

To classify a feature vector $\boldsymbol{x}$ with a learned RF, it traverses through all trees in the forest until a leaf node is reached in each tree. The paths through the trees are determined by the respective learned split functions, on which $\boldsymbol{x}$ is evaluated. The final class predictions $p\left(c|\boldsymbol{x}\right)$ are then obtained by averaging the label distributions stored in the reached leaf nodes during training. More details on the RF inference procedure are given in chapter 2.

In Figure 3.3, we show an example of the intermediate labeling output of the RF classifier, where we simply assigned the most likely label to each input segment of the scene. It can be seen that the computed class label probabilities are only a coarse prediction on the point segment level and very ambiguous, since the classifier processes each segment individually based on its locally computed feature vector.

In the next section we describe the core feature of our pipeline, a densely connected CRF to smooth and refine the result on the finer voxel level, exploiting learned contextual class relations to resolve the ambiguous predictions of the classification stage.

3.4 Dense Conditional Random Field

A CRF is defined over a set $\mathcal{X} = \{X_1, \ldots, X_N\}$ of random variables. The variables are conditioned on model parameters $\boldsymbol{\rho}$ and, in our particular case, the voxelized

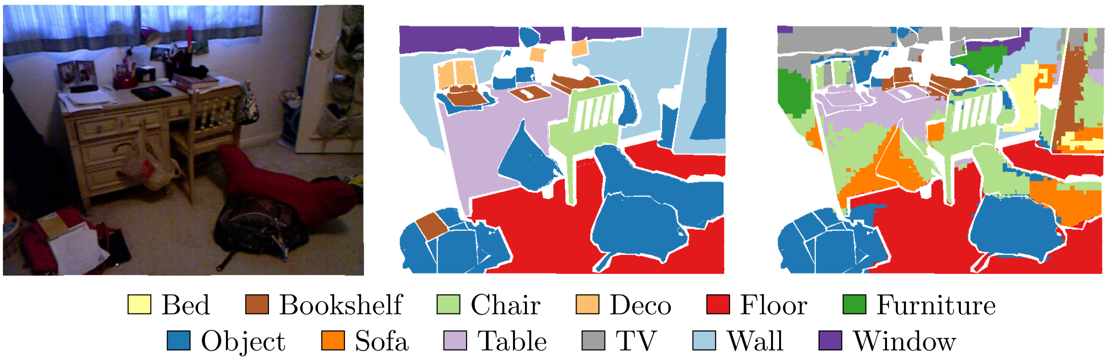


Figure 3.3: Intermediate result from the RF classifier. Left: input point cloud; middle: Ground truth; right: RF result.

input point cloud $\boldsymbol{P}$. Thus, each variable X_i corresponds to a voxel $\boldsymbol{v}_i \in \boldsymbol{P}$ and can take a value from the label set $\mathcal{C} = \{c_1, \dots, c_k\}$. Furthermore, it is associated with a feature vector $\boldsymbol{f}_i$, determined by $\boldsymbol{P}$. Note that this is a different feature vector than the one defined and used for classification in sections 3.2 and 3.3!

The CRF is then defined by a Gibbs distribution on a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, which consists of a set of vertices $\mathcal{V}$ connected by a set of edges $\mathcal{E}$:

$$P(\mathcal{X}|\boldsymbol{P}, \boldsymbol{\rho}) = \frac{1}{Z(\boldsymbol{P}, \boldsymbol{\rho})} \exp\left(-\sum_{q \in \mathcal{Q}_{\mathcal{G}}} \phi_q\left(\mathcal{X}_q|\boldsymbol{P}, \boldsymbol{\rho}\right)\right), \tag{3.4}$$

where the each vertex of $\mathcal{G}$ represents a voxel $\boldsymbol{v}_i$, $\mathcal{Q}_{\mathcal{G}}$ is a set of cliques in $\mathcal{G}$ and each clique $q \in \mathcal{Q}_{\mathcal{G}}$ has a corresponding potential $\phi_q\left(\mathcal{X}_q|\boldsymbol{P}, \boldsymbol{\rho}\right)$ [45].

A complete label assignment $\boldsymbol{c} \in \mathcal{C}^N$ then has a corresponding Gibbs energy, defined by

$$E\left(\boldsymbol{c}|\boldsymbol{P}, \boldsymbol{\rho}\right) = \sum_{q \in \mathcal{Q}_{\mathcal{G}}} \phi_q\left(\mathcal{X}_q|\boldsymbol{P}, \boldsymbol{\rho}\right). \tag{3.5}$$

In a fully (or "densely") connected CRF, each vertex of the graph $\mathcal{G}$ is connected to all other vertices, such that the set of cliques $\mathcal{Q}_{\mathcal{G}}$ is defined as the set of all unary and pairwise cliques. Consequently, the Gibbs energy of a dense CRF can be reformulated as

$$E\left(\boldsymbol{c}|\boldsymbol{P}, \boldsymbol{\rho}\right) = \sum_{i} \phi_i\left(c_i|\boldsymbol{P}, \boldsymbol{\rho}\right) + \sum_{i<j} \phi_{ij}\left(c_i, c_j|\boldsymbol{P}, \boldsymbol{\rho}\right), \tag{3.6}$$

where $\phi_i\left(c_i|\boldsymbol{P}, \boldsymbol{\rho}\right)$ is the potential of a unary and $\phi_{ij}\left(c_i|\boldsymbol{P}, \boldsymbol{\rho}\right)$ is the potential of a pairwise clique and $1 \leq i, j \leq N$.

The optimal labeling $\boldsymbol{c}^*$ can then be obtained by calculating the maximum a posteriori solution of $P(\mathcal{X}|\boldsymbol{P}, \boldsymbol{\rho})$, which corresponds to finding the labeling that minimizes the Gibbs energy:

$$\boldsymbol{c}^* = \underset{\boldsymbol{c} \in \mathcal{C}^N}{\arg\min}\, E\left(\boldsymbol{c}|\boldsymbol{P}, \boldsymbol{\rho}\right) \tag{3.7}$$

The unary potential of our energy model is individually obtained for each voxel from the RF classifier. To this end, we transform the label distribution calculated by the RF to a cost by using the negative log-likelihood:

$$\phi_i\left(c_i|\boldsymbol{P},\boldsymbol{\rho}\right) = -\log\left(p\left(c_i|\boldsymbol{x_i}\right)\right), \tag{3.8}$$

where $\boldsymbol{x}_i$ is the feature vector of the scene segment to which voxel $\boldsymbol{v}_i$ belongs. The pairwise potential is modeled as a linear combination of m kernel functions:

$$\phi_{ij}\left(c_i, c_j|\boldsymbol{P},\boldsymbol{\rho}\right) = \sum_m \mu_m\left(c_i, c_j|\boldsymbol{\rho}\right) k_m\left(\boldsymbol{f}_i, \boldsymbol{f}_j\right), \tag{3.9}$$

where μ_m is a label compatibility function that models contextual relations between classes in the sense that it defines weighting factors, depending on how likely two classes occur near each other.

For reasons explained later in this section, we limit the choice of kernel functions k_m to Gaussian kernels:

$$k_m\left(\boldsymbol{f}_i, \boldsymbol{f}_j\right) = w_m \exp\left(-\frac{1}{2}\left(\boldsymbol{f}_i - \boldsymbol{f}_j\right)^T \Lambda_m \left(\boldsymbol{f}_i - \boldsymbol{f}_j\right)\right), \tag{3.10}$$

where w_m are linear combination weights and Λ_m is a symmetric, positive-definite precision matrix, defining the shape of the kernel.

For our application on 3D voxels, we define two kinds of kernel functions, similar to the work of Hermans et al. [32]. The first one is a smoothness kernel that is only active in the local neighborhood of each voxel. It reduces the classification noise by favoring the assignment of the same label to two close voxels with a similar surface orientation:

$$k_1 = w_1 \exp\left(-\frac{|\boldsymbol{p}_i - \boldsymbol{p}_j|}{2\theta_{p,s}^2} - \frac{|\boldsymbol{n}_i - \boldsymbol{n}_j|}{2\theta_n^2}\right), \tag{3.11}$$

where $\boldsymbol{p}$ are the 3D voxel positions and $\boldsymbol{n}$ are the respective surface normals. $\theta_{p,s}$ controls the spatial influence range of the kernel, whereas θ_n defines the degree of similarity of the normals. The second kernel function is an appearance kernel that also allows information flow across larger distances between voxels of similar color:

$$k_2 = w_2 \exp\left(-\frac{|\boldsymbol{p}_i - \boldsymbol{p}_j|}{2\theta_{p,l}^2} - \frac{|\boldsymbol{l}_i - \boldsymbol{l}_j|}{2\theta_l^2}\right), \tag{3.12}$$

where $\theta_{p,l} \gg \theta_{p,s}$ and $\boldsymbol{l}$ are the color vectors of the corresponding voxels, transformed to the CIELAB color space.

In contrast to [32], where a manually defined Potts model is used for both kernels, we define separate label compatibility functions μ_m for each of them. For the smoothness kernel k_1 we use a simple Potts model as well:

$$\mu_1(c_i, c_j|\boldsymbol{\rho}) = \begin{cases} 0, & \text{if } c_i = c_j \\ 1, & \text{otherwise} \end{cases} \tag{3.13}$$

For the appearance kernel k_2, however, we use a more expressive model, since it is supposed to capture contextual relations between different classes across larger distances. In our approach, μ_2 is a full, symmetric $M \times M$ matrix, where all pairwise class relations are defined individually.

Of course, manually trying to find an optimal set of parameters for the compatibility matrix and the respective linear combination weights is not intuitive and for a large number of different labels, as in our case, intractable. Instead, we iteratively learn them automatically from training data. More details about the parameter learning for our dense CRF model are given in section 3.4.2.

3.4.1 Inference

Because we only use Gaussian kernels to define the pairwise potentials, we can apply a highly efficient inference method presented by Krähenbühl and Koltun [42], based on a mean field approximation of the CRF distribution. In each step of the iterative optimization, a single variable is updated by aggregating information from all other variables. The mean field update of the whole densely connected CRF is performed using Gaussian filtering in feature space. As a consequence, the approximate inference method is sublinear in the number of pairwise potentials and enables all pairs of variables to be connected by pairwise potentials.

For our application this has two key advantages: First, it allows us to define the CRF on the finer voxel level instead of the segment level while maintaining fast inference times. Consequently, the CRF improves the labeling on a finer scale and across segment boundaries, such that it is able to correct eventual segmentation errors. Second, because of the dense connectivity, information can propagate across large distances in the scene. Therefore, the model is able to capture global contextual relations between different classes, helping to resolve ambiguities in the labeling result.

3.4.2 Parameter Learning

To be able to fully exploit the capabilities of the complex CRF model, its parameters have to be well defined. However, since the number of parameters grows exponentially with the number of labels and there are many dependencies among them, this is a difficult task. Recently, Krähenbühl and Koltun extended their CRF model with a learning framework for this problem, based on the minimization of differentiable loss functions over the mean field marginals [43]. In contrast to piecewise training methods, this approach is able to jointly estimate all parameters, such that dependencies between them can be captured.

Common options for the loss functions include log-likelihood, robust log-likelihood or the Hamming loss. In our case, however, the intersection over union objective is better suited since it is able to compensate for imbalanced training data. The general intersection over union score is defined as

$$L = \sum_c \frac{i_c(\boldsymbol{c})}{u_c(\boldsymbol{c})} = \sum_c \frac{tp_c}{n_c + fp_c}, \tag{3.14}$$

where tp_c is the number of true positives for class label c, fp_c is the number of false positives, and n_c is the number of data points with class label c in the ground truth.

Adapting the learning framework from [43] to our application, we are able to simultaneously estimate all linear combination weights as well as the label compatibility functions, such that the individual class relations can be learned from training data. In the evaluation results presented in chapter 5 it can be seen that modeling this contextual information significantly improves the overall performance of our framework compared to using a simple CRF model with potts potentials and manually defined parameters.

Chapter 4

Context Learning with 3D Entangled Forests

In the previous chapter, we have presented our first concept to exploit scene context by learning the pairwise terms of a densely connected CRF from training data. This enables the pipeline, in the remainder of this thesis simply referred to as *CRF pipeline*, to capture and model frequently occurring combinations of different labels, such that the final refinement step is able to compensate potential classification errors from the RF. While our experiments show that the dense CRF indeed improves the overall classification performance, however, the approach also comes with two drawbacks:

1. The pairwise terms of the CRF need to be constrained to Gaussian kernels in order to be able to apply the computationally efficient mean field approximation for inference, which in turn limits the expressiveness of the model.
2. The split of the pipeline into a separate classification and refinement step requires to also split the available training data into two smaller sets. Since training data for semantic segmentation is very expensive to obtain and therefore limited, this further reduction potentially restrains the performance of the two individual steps.

Therefore, in this chapter we present an alternative method for context learning in semantic segmentation, overcoming both limitations of the CRF pipeline. We extend the capabilities of a standard RF classifier with a powerful concept called *Entanglement* and introduce *3D Entangled Forests (3DEF)*. A 3DEF is a new classifier, combining the advantages and efficiency of the RF architecture with the ability to incorporate more global information of the whole input scene. This enables the approach to learn not only individual relations between different class labels, but also to explicitly model their respective geometric configuration in the scene.

This capability of a 3DEF allows to spare an additional refinement step, such that the resulting pipeline, depicted in Figure 4.1, is very compact and efficient.

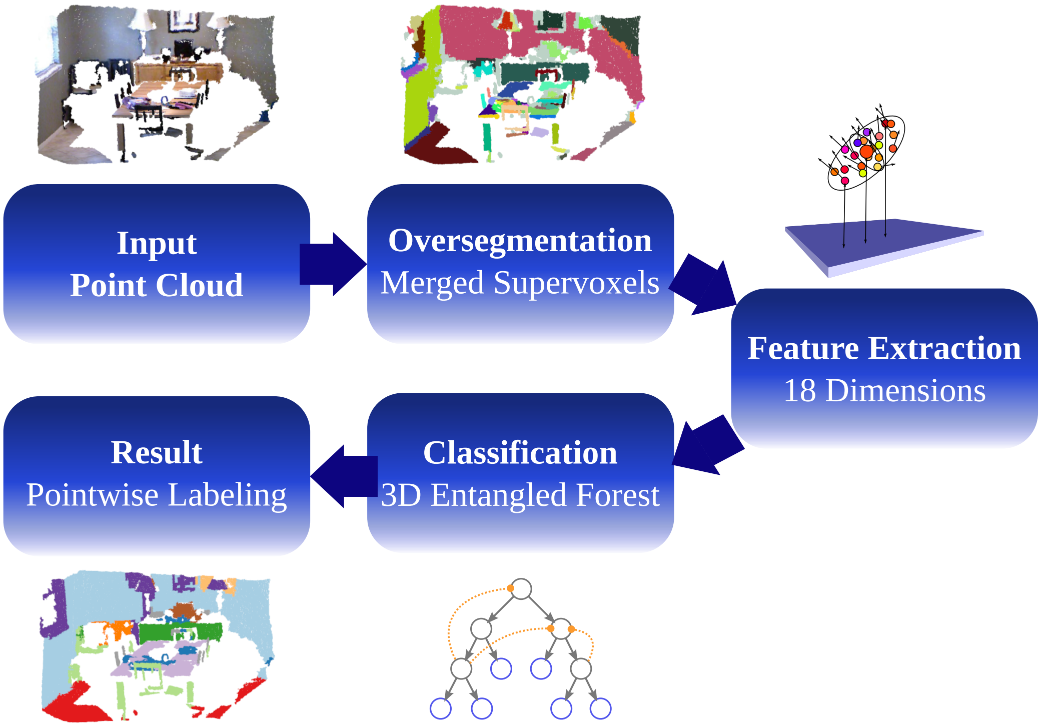


Figure 4.1: Overview of the second pipeline using a 3DEF for classification and context learning.

4.1 Overview

The 3DEF operates on pre-segmented 3D point clouds, which have been recorded with an RGB-D sensor such as the Microsoft Kinect. Similar to the CRF pipeline, in the first step we calculate an oversegmentation of the scene to obtain homogeneous segments of similar size, applying VCCS. In an additional step, we then subsequently merge neighboring segments exhibiting similarity with respect to appearance, structure and orientation into larger, mostly planar segments.

Compared to the segmentation result computed in chapter 3, these larger segments generally contain richer information about the corresponding object or structure. For example, a wall is much easier to recognize if represented by one single large segment (where the segment's size is already a strong indicator for the segment being a wall) than if it is fragmented into many small segments that have to be classified individually. Additionally, features calculated on larger segments are much more robust against noise in the input data.

Generally, we follow the architecture introduced for the CRF pipeline, such that for each of the resulting segments, a unary feature vector is calculated, containing local appearance and geometry features. Additionally, as a key element of the 3DEF, so-called *3D entangled features* are computed on-the-fly during training and classification.

The power of entangled features lies within the possibility that the path a data point (in our case, a segment) takes through a decision tree *is influenced by the* *current position of other data points in the tree.* For example, while classifying one of the segments in the scene, considering the current most likely label of an adjacent segment can help the classifier to better predict the current segment's label. It is intuitive that the mere presence of a segment likely labeled chair in the scene should increase the "sensitivity" of the classifier to other chair or table segments, since these object classes often co-occur. Besides this contextual relation, the information *where* the informative adjacent segment has been found is also important as it establishes a geometric relation between different classes. Our 3D entangled features are able to learn and exploit both of these relations, which is why they are very discriminative.

In the next sections, we first describe the segmentation step, followed by a detailed explanation of the concept and the implementation of the 3DEF and its training and inference procedures.

4.2 Preprocessing and Segmentation

In indoor RGB-D datasets for scene understanding, the distances of the scene contents from the camera are generally much larger compared to datasets with a different scope, e.g. tabletop scenes for object recognition. Consequently, the point clouds contain significant noise dominated by quantization errors of the camera. To improve the quality of the segmentation stage, we first reduce the effect of camera noise before segmenting the input point cloud, applying an approximated bilateral filter [67]. The filter smoothes the depth information while preserving edges, which is important for our following segmentation step. An example result of the bilateral filtering is pictured in Figure 4.2.

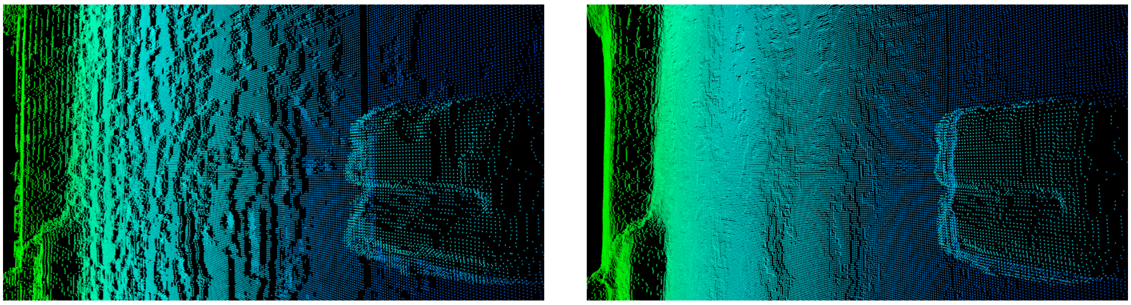


Figure 4.2: An example for the effect of bilateral filtering applied to the input point cloud. Left: input point cloud showing severe quantization noise. Right: point cloud after filtering. Note the well-preserved edges of the box.

After filtering, the filtered point cloud is oversegmented into homogeneous segments using VCCS [66], similar to the CRF pipeline. Both the approximate bilateral filter and VCCS are available in the Point Cloud Library (PCL) [73].

In the next step, we apply a region growing algorithm, recursively merging two adjacent segments c_i and c_j into one larger segment $s_{i,j}$ if the evaluation of a distance function $d(c_i, c_j)$ is below a threshold τ_{merge}. The distance function is a linear combination of the color, surface normal and point-to-plane distance between the segments:

$$d(c_i, c_j) = w_c d_c(c_i, c_j) + w_n d_n(c_i, c_j) + w_p d_p(c_i, c_j), \tag{4.1}$$

where d_c is the color distance in Lab CIE 94 color space, d_n the surface normal difference indicated by the dot product $(1 - n_i n_j^T)$, d_p is the max of the point-to-plane distance from c_i to c_j and vice versa and w_c, w_n and w_p are the respective weights, normalized to sum up to 1. The algorithm stops if there are no more adjacent segments to be merged and returns the final set of segments $\mathcal{S}$, which serves as the input to the 3DEF. An example result of our segmentation procedure is shown in Figure 4.3.

4.3 3D Entangled Forests for Classification

The standard RF classifier, as it is applied in the CRF pipeline described in chapter 3, generally processes data points independently from each other, and each data point corresponds to a feature vector which is calculated in a closely bounded local region. Therefore, classification results not only suffer from classification noise; the capability of incorporating more complex contextual information is also very limited.

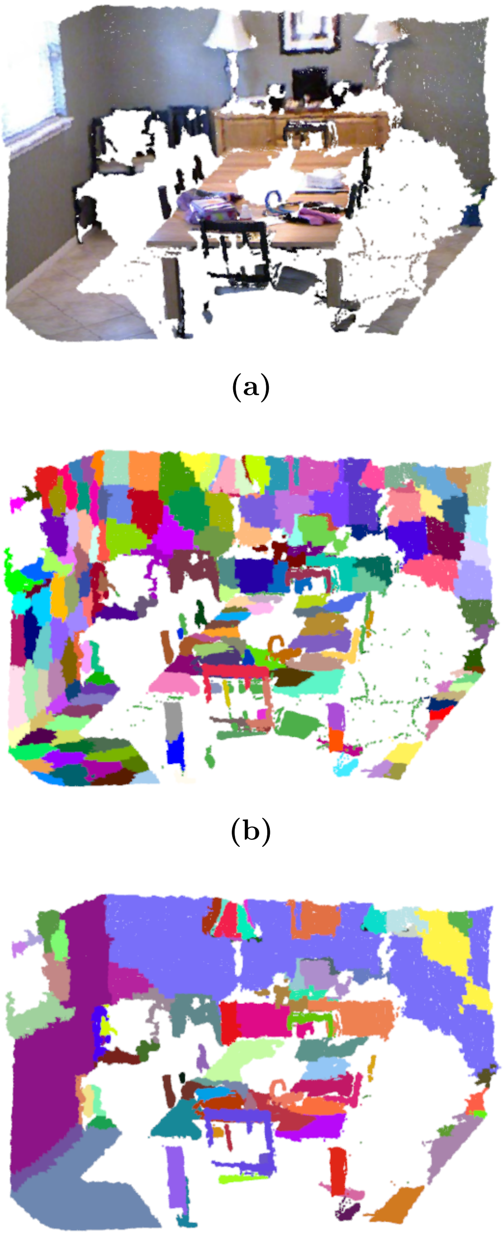
However, the architecture of the standard RF classifier can be modified to allow for a new type of features called *entangled features* [61], which are able to compensate for those deficits. In particular, they utilize the learned tree structure and intermediate class distributions, resulting from training the nodes closer to the root node, to provide contextual information which helps splitting training data when learning deeper levels of the tree.

So far, entangled features have only been applied in 2D image space, mainly to classify grayscale medical images, e.g. from a CT scan [40, 61]. In the next sections, we take the concept of entanglement to the next level and introduce *3D Entangled Forests*.

First, we describe our used feature sets: section 4.3.1 explains the unary feature vector, individually computed for each segment $s_i \in \mathcal{S}$, and in sections 4.3.2 to 4.3.4, we present our new set of 3D entangled features. The implementation details applied in our approach for training and inference are listed in sections 4.3.5 and 4.3.6.

4.3.1 Unary Features

As a prerequisite for the feature calculation, we first align the segmented point cloud with the ground plane. Then, for each segment $s_i \in \mathcal{S}$ obtained in the segmentation step, we calculate an 18-dimensional unary (i.e. locally calculated) feature vector u_i that is based on the feature set used in the CRF pipeline. However, we exploit that the input segments for our new approach are much larger and their geometric properties are more meaningful by replacing some of the features from the original set with more descriptive ones.



(a)

(b)

(c)

Figure 4.3: Example results of the single steps of our segmentation stage. (a) shows the input point cloud, (b) the intermediate result from VCCS with a maximum segment size of 30 cm, after applying the bilateral filter on the input point cloud. (c) is the final segmentation result after the merging step.

Instead of calculating the spectral features (compactness, linearity and planarity), we extract features from the oriented three-dimensional bounding box enclosing a segment. Besides its absolute height, depth and width, we also use the ratios between the bounding box dimensions to calculate the elongation in horizontal and vertical direction as well as the “thickness” of the segment. Furthermore, we compute the area occupied in the horizontal and vertical plane.

The remaining features include the mean and standard deviation for each color channel of the points in CIELAB color space, the mean angle of the surface normal with respect to the ground plane and its standard deviation and the height above the ground plane of the highest and lowest point of the segment. A complete overview of the full feature set is listed in Table 4.1.

During training of the 3DEF, the only parameter which needs to be sampled for a unary feature is a threshold ϕ. If a segment's value of the evaluated unary feature is larger than ϕ, the feature evaluates to `true`.

Table 4.1: List of all unary features calculated for a 3D segment and their dimensionality.

Feature	Dimensions
Color in CIELAB space (mean and standard deviation)	6
Angle with ground plane (mean and standard deviation)	2
Height above ground plane (min and max)	2
Bounding box dimensions and ratios	6
Vertical and horizontal plane area (length×height, length×depth)	2
Total number of features	18

4.3.2 3D Entangled Features

A crucial component of entangled features is a method which is able to learn where to find—*repeatably and across different scenes*—other data points, whose contextual information helps the most to predict the current data point.

As introduced in [61], for 2D images this problem breaks down to learning 2D pixel offsets relative to the current pixel to select a close-by area to evaluate the entangled feature on. Especially for medical images, this works well because they are always recorded from the same viewpoint and the objects pictured (e.g. organs) are generally arranged in roughly the same spatial configuration in the image plane.

For 3D point clouds, however, the problem gets disproportionately harder, since data lives in projective space. Simple offsets would not learn any meaningful relations, because an offset pointing to an informative area in one training point cloud could be completely meaningless in the next point cloud if the viewing angle on the scene is different.

Therefore, in the following section we present a novel scheme to select close-by segments, working directly in 3D space, that enables our features to reliably find segments providing useful contextual information. The basic idea is that each 3D entangled feature learns a set of specific geometric constraints, in general relative to the currently classified segment s_t, which reduces the set $\mathcal{S}$ of all available segments in the scene to a small set of *candidate segments* $\mathcal{S}_C^t \subset \mathcal{S}$. The feature then learns parameters of a binary test, applied to each of the candidate segments $s_i \in \mathcal{S}_C^t$, to extract contextual information from them. The feature evaluates to `true` if at least one candidate segment passes the test.

As an example, consider a feature trying to classify segments belonging to seats of chairs. The feature could learn that the existence of an adjacent, perpendicular segment, which is furthermore likely being labeled *chair* (the back rest of the chair), is a strong indicator that the seat segment should also be labeled *chair*. This feature would learn geometric constraints that limit $\mathcal{S}_C^t$ to segments which are (1) directly adjacent and (2) perpendicular to the current segment. The learned binary test of the feature would then check if $\mathcal{S}_C^t$ contains a segment s_i, for which the classifier currently predicts the label *chair*.

In the following sections, we describe the available geometric constraints and the different binary tests in more detail.

4.3.3 Modeling Spatial Relations with Geometric Constraints

The first constraint requires the point-to-plane distance $d^{t,i}$, measured along the surface normal of s_t, from the centroid of s_t to the closest point of s_i, to fall within the range $[d_{min}, d_{max}]$. The second restricts the enclosed angle $\alpha^{t,i}$ between the surface normals of s_t and s_i, but here we distinguish between two versions of the constraint: One measures the angle in the horizontal plane (parallel to the ground plane) and the other in the vertical plane. For indoor scenes, these oriented angles are very descriptive, because objects and structures are often aligned along the three dominant room axes.

As a final constraint, only segments with a Euclidean distance $d_e^{t,i}$ of up to $d_{e,max} = 1\,\text{m}$ from s_t are considered; a final, fixed constraint to limit the influence of distant segments.

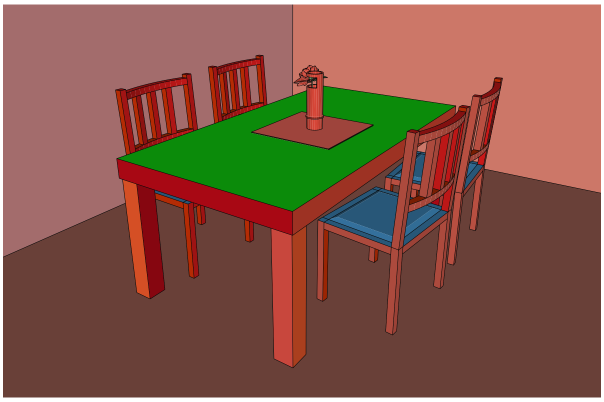
More formally, we can define the geometric constraints to obtain the set of candidate segments $\mathcal{S}_C^t$ for the target segment s_t as follows:

$$\mathcal{S}_C^t = \{s_i \in \mathcal{S} \,|\, i \neq t \wedge d^{t,i} \in [d_{min}, d_{max}] \wedge \alpha_v^{t,i} \in [\alpha_{v,min}, \alpha_{v,max}] \wedge d_e^{t,i} \leq d_{e,max}\}, \quad (4.2)$$

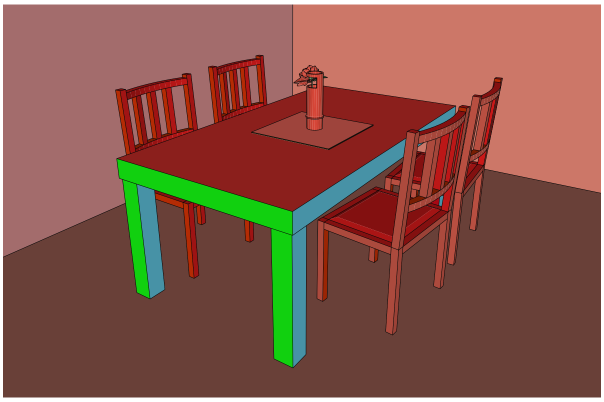
$$\mathcal{S}_C^t = \{s_i \in \mathcal{S} \,|\, i \neq t \wedge d^{t,i} \in [d_{min}, d_{max}] \wedge \alpha_h^{t,i} \in [\alpha_{h,min}, \alpha_{h,max}] \wedge d_e^{t,i} \leq d_{e,max}\}, \quad (4.3)$$

where $\alpha_v^{t,i}$ and $\alpha_h^{t,i}$ refer to the normal angle difference in the vertical and horizontal plane, respectively. Note that the parameters d_{min}, d_{max} and $\alpha_{v,min}$, $\alpha_{v,max}$ respectively $\alpha_{h,min}$, $\alpha_{h,max}$ are all individually learned for each feature at each node, only $d_{e,max}$ is a fixed constraint.

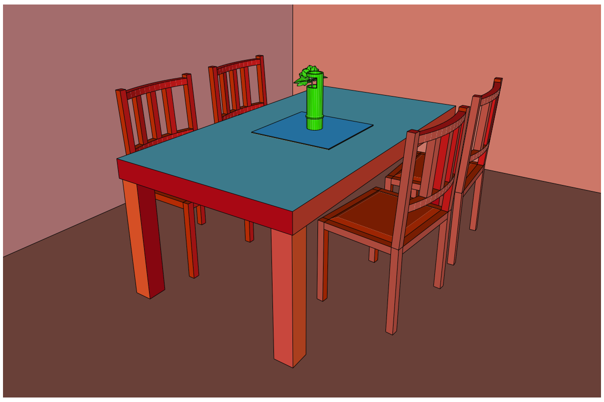
Some illustrative, simplified examples of important spatial relations, which can be captured with these constraints, are given in Figure 4.4.



(a) Table planes and chair seats



(b) Corners of tables



(c) Objects on tables

Figure 4.4: Idealized examples of learned geometric constraints, demonstrating the ability to capture frequent, informative spatial configurations between classes. The green segment is the respective target segment s_t, blue segments fulfill the constraints and compose the set of candidate segments $\mathcal{S}_C^t$ and red segments do not meet the constraints and are ignored by the respective feature. (a) is learned using the vertical constraint, requiring candidate segments to be approximately 30 cm below and parallel to the table plane. (b) shows the horizontal constraint applied, only selecting segments which are perpendicular and directly adjacent to the front segment of the table. (c) depicts the inverse application as used for the *Inverse TopN Segment Feature*: Candidate segments have to be parallel to the ground plane and need to be adjacent to the target segment.

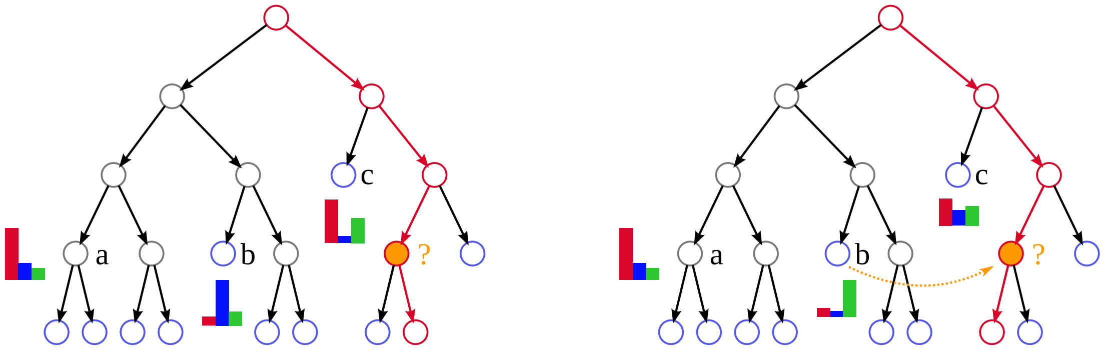


Figure 4.5: Illustration of the TopN and Inverse TopN Segment Feature during inference for a forest with the three labels red, blue and green. In this example, the feature is evaluated in the split node marked orange, and its learned geometric constraints yield three candidate segments, which have reached the nodes denoted $a - c$. Suppose the learned target label l is green and parameter N is 1. In the left example, the feature would evaluate to `false`, because for none of the candidate segments, the current most likely label is green. In the right example, the feature evaluates to `true`, because for candidate segment b, the current node has green as its most likely label.

4.3.4 Capturing Contextual Relations with Binary Tests

We introduce five different 3D entangled features and their corresponding binary tests:

1. **Existing Segment Feature**
   This feature evaluates to `true` if at least one of the candidate segments fulfills the geometric constraints ($|\mathcal{S}_C^t| > 0$). Therefore, it does not require to learn any additional parameters besides the geometric constraints.

2. **TopN Segment Feature**
   Based on the *Existing Segment Feature*, this feature additionally takes the intermediate label predictions for the candidate segments into account and learns two additional parameters: a candidate label l and a positive integer N. The feature analyzes the class label distributions of the current tree nodes which the candidate segments have reached so far during classification. If the label l is among the top N class labels of the distribution for at least one of the candidate segments, the feature evaluates to `true`. A visualization of this feature is shown in Figure 4.5.

3. **Inverse TopN Segment Feature**
   For this feature, the binary test is the same as for the regular *TopN Segment Feature*, but the geometric constraints are applied differently: First, the angle difference of the candidate segment s_i is calculated *w.r.t. the ground plane* instead of the target segment s_t. Consequently, regarding the applied geometric constraints, this feature is the only one which is not split into a "horizontal" and a "vertical" version. And second, the point-to-plane distance is measured *inversely*, thus, along the surface normal of s_i, from the centroid of s_i to the closest point of s_t. Not using the surface normal of s_t as a reference (as all other

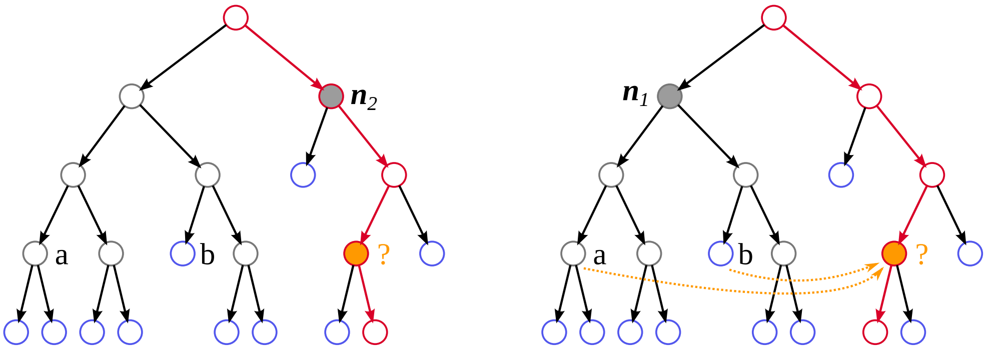


Figure 4.6: General principle of the node descendant feature during inference. Evaluated in the split node marked orange, the feature's learned geometric constraints yield two candidate segments, which have reached the nodes denoted a and b. In the left example, the feature learned a node index $k = 2$, such that it evaluates to `false`, because neither node a or b are direct or indirect descendants of node $\boldsymbol{n}_2$, marked in gray. On the contrary, in the right example the feature learned $k = 1$ and evaluates to `true`, because node $\boldsymbol{n}_1$ is an indirect parent of both nodes a and b.

features do), this feature has been designed for classes without a meaningful mean surface normal, i.e. classes that are not composed by mainly planar surfaces (as exemplified in Figure 4.4c). A visualization of this feature is shown in Figure 4.5.

4. **Node Descendant Feature**
   This feature considers the path candidate segments s_i take through the tree during classification. With a node index k as additionally learned parameter, the feature evaluates to `true` if the current tree node reached by at least one candidate segment is a (direct or indirect) descendant of tree node $\boldsymbol{n}_k$. Figure 4.6 illustrates the general idea behind this feature.

5. **Common Ancestor Feature**
   This feature can be considered as the pairwise equivalent of the *Node Descendant Feature*, because it takes the classification path of the target segment s_t into account as well. As a parameter it learns a maximum number of steps m. Starting at the current tree nodes reached by s_t and s_i, the tree is traversed backwards until a common ancestor node has been reached. If this node is reached in a maximum of m steps, the feature evaluates to `true`. A visualization is provided in Figure 4.7.

Applying the geometric constraints, the *topN segment feature*, the *node descendant feature* and the *common ancestor feature* are 3D extensions of [61]. The *existing segment feature* and the *inverse topN segment feature* are new, specifically designed 3D entangled features. A summary of all new features and their learned parameters is given in Table 4.2.

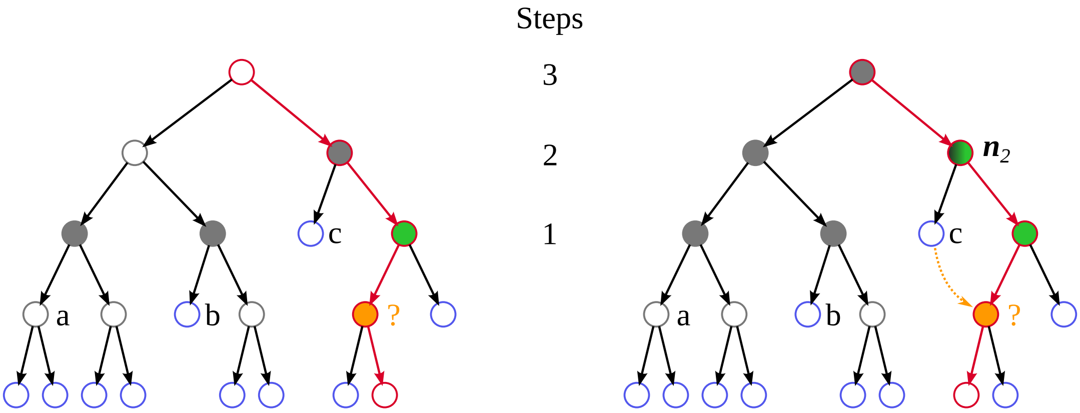


Figure 4.7: The functionality of the common ancestor feature during inference. Evaluated in the split node marked orange, the feature's learned geometric constraints yield three candidate segments, which have reached the nodes denoted a, b and c. In the left example, the feature learned a maximum number of steps $m = 1$. Going only one level up from all nodes of the candidate segments and the node of the current segment reaches the nodes indicated in gray respectively green. There is no common ancestor between one of the candidate segments and the current segment reached, such that the feature evaluates to `false`. In the right example, the feature learned to go back $m = 2$ steps. In this case, there is a common ancestor reached (node $\boldsymbol{n}_2$, indicated gray/green), and the feature evaluates to `true`.

Table 4.2: List of all learned parameters for each of the 3D entangled features. Note that the Inverse TopN Segment feature is the only one for which there is no separate horizontal and vertical version, because the geometric constraints are applied differently.

Feature	Dimensions	Learned Parameters
Existing Segment	4	d_{min}, d_{max}, $\alpha_{h/v,min}$, $\alpha_{h/v,max}$
TopN Segment	6	d_{min}, d_{max}, $\alpha_{h/v,min}$, $\alpha_{h/v,max}$, label l, limit N
Inverse TopN Segment	6	d_{min}, d_{max}, α_{min}, α_{max}, label l, limit N
Node Descendant	5	d_{min}, d_{max}, $\alpha_{h/v,min}$, $\alpha_{h/v,max}$, node index k
Common Ancestor	5	d_{min}, d_{max}, $\alpha_{h/v,min}$, $\alpha_{h/v,max}$, max. steps m

4.3.5 Implementation Considerations

To be able to apply the new 3D entangled features, a few general considerations for the implementation of the decision tree algorithm have to be taken into account. First, the structure of the trees has to be learned in breadth-first instead of depth-first order to ensure that every node has all data of the previous tree level available. And second, a mechanism needs to be in place that keeps track of the currently reached tree nodes for all data points to provide the basis for the binary tests of all entangled features. This lookup table is updated every time a data point traverses to the next depth level, during training as well as during inference.

In the next section, we outline the details of the applied training and inference procedures for our 3DEF.

4.3.6 Training and Inference

Because the architecture of a 3DEF is based on an RF classifier, we can generally apply the standard training and inference procedures as described in chapter 2.

To prevent overfitting and decorrelate the trees in the forest from each other, we take several measures. First, we limit the number of available features in every node. In particular, for each node, a random subset containing only half of the entire feature set is drawn. Second, we apply a bagging technique, such that each tree is only trained on 30 % of the available training data. This also helps leveling the highly unequal distribution of training data per class label in the datasets (refer to section 5.1). However, even in the smaller bags there remain a few classes which clearly dominate (e.g. *Wall*, *Bookshelf*). To compensate for the imbalance, we weight each data point with a class weight inversely proportional to the number of points available for the corresponding class label in the bag. This weighting is applied twice in the split evaluation procedure, which randomly samples and evaluates different feature settings to determine the best split feature in a new node of the tree. First, it reweighs the contribution of the different labels to the entropy of the label distribution in a node. And second, we also reweigh the contribution of the entropies in the left and right child nodes towards the information gain of a proposed split. This is a different approach to the weighting proposed in [10], where the respective entropies are only weighted by the number of points in the left and right child node (as already defined in equation 2.7):

$$\xi = H(\mathcal{X}) - \sum_{k\in[l,r]} \frac{|\mathcal{X}_k|}{|\mathcal{X}|} H(\mathcal{X}_k) \tag{4.4}$$

The idea behind the original scheme from [10] is to avoid splitting off child nodes with very few points. However, if some classes are only represented by a few data points compared to dominant classes, such splits definitely make sense, such that we alter the definition of the information gain to:

$$\xi = H(\mathcal{X}) - \sum_{k\in[l,r]} \frac{|\hat{\mathcal{X}}_k|}{|\hat{\mathcal{X}}|} H(\mathcal{X}_k), \tag{4.5}$$

where $\hat{\mathcal{X}}_k$ is the weighted number of points and $\hat{\mathcal{X}}$ the corresponding normalization term. This weighting scheme allows for splitting off only a few data points if their respective class weights are large.

Following the feature selection analysis of [61] and also the work of [40], in the first few levels of the forest only our unary feature set is active. Entangled features depend on previously learned tree nodes, such that we do not enable them until depth level 6, when the learned tree structure and the intermediate label predictions in the nodes are already more meaningful.

For some classes, especially for those with a small number of available training data, the applied geometric constraints can be too specific, such that they tend to overfit. To countersteer this effect, we partly relax the constraints for a fraction of the sampled feature configurations. In particular, for 25 % of the sampled parameter sets for an entangled feature, we loosen the constraints and only enforce either the point-to-plane or the angular constraint. In this case, the geometric constraints are randomly selected from one of the following reduced definitions:

$$\mathcal{S}_C^t = \{s_i \in \mathcal{S} \mid i \neq t \wedge d^{t,i} \in [d_{min}, d_{max}] \wedge d_e^{t,i} \leq d_{e,max}\}, \tag{4.6}$$

$$\mathcal{S}_C^t = \{s_i \in \mathcal{S} \mid i \neq t \wedge \alpha_h^{t,i} \in [\alpha_{h,min}, \alpha_{h,max}] \wedge d_e^{t,i} \leq d_{e,max}\}, \tag{4.7}$$

$$\mathcal{S}_C^t = \{s_i \in \mathcal{S} \mid i \neq t \wedge \alpha_v^{t,i} \in [\alpha_{v,min}, \alpha_{v,max}] \wedge d_e^{t,i} \leq d_{e,max}\}. \tag{4.8}$$

The remaining hyperparameters of the forest are presented in the next chapter.

Chapter 5

Evaluation

In this chapter, we present a thorough evaluation of both of our pipelines in various settings and on different challenging indoor datasets for semantic segmentation. Comparing our results to several other approaches, we examine the strengths and weaknesses of our methods. Furthermore, we provide an in-depth analysis how different parameters influence the overall performance and which features provide the most information. Finally, we prove the capability of both of our methods to infer and learn contextual information of a scene and show how this improves the results.

In the next sections, we present several popular datasets of indoor scenes and describe their suitability for various different tasks in the field of semantic segmentation. We then introduce the exact setup of our methods and the used metrics for the following evaluation.

5.1 Datasets

Obtaining densely labeled data for semantic segmentation is a very time-consuming and expensive process, particularly for 3D point cloud data. However, in recent years, several different datasets for indoor scenes have been published. We give a brief description of the major datasets, highlighting the most important differences, and explain our choice to evaluate our algorithms on the two NYU Depth Datasets [81, 82].

Cornell Dataset [41]

Published in 2011, the Cornell Dataset consists of 24 office and 28 home scenes recorded with a Microsoft Kinect sensor. The scenes were reconstructed from about 550 single frames, where each scene has been reconstructed from $8-9$ different views using RGBDSLAM [15]. In many areas, the obtained reconstructions are rather sparse, such that our segmentation method based on pointwise adjacency does not produce usable results. Furthermore, the established label sets used for evaluation on this dataset are very specific and divide objects into several parts, e.g. there are labels such as *tableTop*, *tableDrawer*, *tableLeg* or *cpuFront*, *cpuSide* and *cpuTop*. In contrast to the complex hierarchical graphical model presented with the dataset, for our scope with computational efficiency as a priority, this division is too granular and is not suitable for a meaningful evaluation.

SUN3D Database [98]

The SUN3D Database has been introduced in 2013 and contains a huge collection of tens of thousands of frames recorded by an ASUS Xtion Pro Live sensor. More precisely, at the time of publication of this thesis, 415 sequences have been captured for 254 spaces, in 41 different buildings. The objects in the database are full 3D reconstructions, annotated with semantic labels. For annotation, first single frames are labeled with polygons in 2D. Using a reconstruction method, this annotation is then further propagated to other frames of a scene. At the same time, annotations of the same object in different frames are used in the reconstruction to establish correspondences and to refine the 3D model.

Although the amount of data is large and dense annotations are available, for our single frame semantic segmentation approach, however, the dataset is not very suitable. Errors in the reconstructions cause the projections of the 2D polygons to be far off in the next frame, which leads to substantial annotation errors. Furthermore, the 2D polygons are drawn by external users, such that the labeling quality varies a lot and objects labeled in one frame are sometimes not labeled in another one.

Altogether, these issues make it very difficult for a context-based semantic segmentation approach to extract consistent and meaningful information.

SceneNet [29]

SceneNet is a very new addition to the collection of semantic scene datasets, motivated by the recent impressive rise of Deep Learning methods (e.g. Convolutional Neural Networks or CNNs) applied to various computer vision problems. A key factor for these methods to perform well is an enormous amount of available labeled training data. Therefore, they are very well suited for problems where a theoretically infinite amount of data can be synthetically generated.

Tapping into this field, the SceneNet dataset is using synthetic 3D renderings of indoor scenes. Constraints and heuristics ensure that the randomly created scenes contain realistic and meaningful arrangements of furniture and other objects. Of course, the huge advantage of the dataset is that perfectly labeled ground truth data is acquired for free. Furthermore, the different scene types can be distributed in a way such that the dataset is approximately balanced and there are no clearly dominant classes.

However, the largest drawback so far is that the entire dataset contains 3D data only, without any color or texture information. Therefore, we cannot use the dataset to test our methods.

Very recently, an extension to the dataset has been published [56], where textures are rendered on top of the scenes as well. While the resulting renderings are visually pleasing, it remains to be seen how the artificially added color information contributes to the value of the dataset.

NYU Depth Datasets [81, 82]

The NYU Depth Dataset is divided into two versions. Version 1 has been presented in 2011 [81], version 2 followed 2012. Both versions, in the remainder referred to as NYUv1 and NYUv2, have been recorded with a Microsoft Kinect sensor. The first version consists of more than 100,000 frames from 64 different scenes of 7 scene types (bathroom, bedroom, bookstore, cafe, kitchen, living room and office). From each scene, several frames have been manually labeled, such that in total 2,347 densely annotated single frames are available. The most widely used set of labels, introduced by the authors, contains 13 different labels, with the 13th label being an additional "background" label containing all points which do not belong to one of the other 12 labels: bed, blind, bookshelf, cabinet, ceiling, floor, picture, sofa, table, tv, wall and window. Thus, the background label is mixing hundreds of different classes together, which makes it very hard for a classifier to capture any distinct properties of this class. Furthermore, the background label makes the dataset very imbalanced, since it contains points from so many different classes. Another very dominant label of the dataset is bookshelf, since large portions of the dataset have been recorded in a bookstore.

The second version of the dataset contains some selected scenes from the first version, extended by 464 new scenes, resulting in a total number of over 400,000 unlabeled frames. The subset of labeled single frames consists of 1,449 frames. While the initial label set proposed by the authors contains only 4 coarse class labels, the most commonly used label set for this dataset, using the following 13 labels, has been introduced in a later work by Couprie et al. [9]: bed, object, chair, furniture, ceiling, floor, wall deco., sofa, table, wall, window, bookshelf, tv.

Containing many more different scene types and no mixed background label, the second version of the NYU dataset, used with the reasonable label set of [9], is much more balanced than the first version. Moreover, the large amount of available densely labeled single frames makes NYUv2 a very suitable dataset for the scope of our methods, where a reliable annotation is key to be able to learn a meaningful contextual model.

Therefore, the majority of the evaluation of both of our pipelines is conducted on the set of 13 labels of the NYUv2 dataset. However, we also list experimental results obtained on different label sets, as well as the imbalanced first version NYUv1.

5.2 Experimental Setup

NYUv1 provides 10 different splits of the labeled data into a training and a test set, and the listed results are averaged over all of them. For the second version, only one split in training and test set is given. We list the results for this particular split, however, to mitigate the influence of the randomness of the RF architecture on the results, we learn 10 different forests on this split and list the averaged results.

Since the CRF pipeline requires two disjunct training sets for the RF classifier and the dense CRF, we randomly divide the training set, where 2/3 of the data are used for the classifier and the remaining 1/3 is used to train the dense CRF.

5.2.1 Parameter settings

For all of the conducted experiments, a few hyperparameters remain fixed:

CRF Pipeline

For the oversegmentation, we set the voxel size r_v to 1.5 cm and the maximum segment size r_c to 30 cm. These settings are a suitable trade-off between speed and accuracy, as the segments can capture enough information while smaller objects can still be accurately segmented. The weights for the influence of the color distance, surface normal difference and spatial regularity have been empirically set to to 0.7, 2.0 and 0.1, respectively.

For the RF classifier, we fix the number of trees t and the maximum tree depth d to 20. During training, in each split node 200 feature/threshold combinations are evaluated. The minimum information gain h and the minimum number of available points n for a valid split is set to 0.02 and 10, respectively.

In the CRF model, we only define the parameters specifying the similarity of color and normal features and the range on which the kernels operate, all other parameters are learned from training data. For the smoothness kernel, we set the range parameter $\theta_{p,s} = 20\,\text{cm}$ and the surface normal similarity $\theta_n = 0.05\,\text{rad}$. The appearance kernel operates in a larger range of $\theta_{p,l} = 1\,\text{m}$ and the color similarity parameters are set to $\theta_{c,L} = 12$ for the L and $\theta_{c,ab} = 3$ for the a and b channels. For all experiments, we set the number of executed CRF iterations to 3.

To prove the advantageousness of using a dense CRF with learned parameters, we evaluate our framework in three different configurations: First, we compare the labeling result directly after the RF classifier, without any further processing by the CRF. Second, we add a CRF, but only use simple Potts models for the label compatibility terms with manual tuning of the kernel weights. Finally, we evaluate the performance using our full CRF model, where we jointly learned the kernel weights and the full label compatibility matrix from training data.

3D Entangled Forest

For all our experiments with the 3DEF, we use the same settings for our segmentation stage. Compared to the CRF pipeline, we slightly adapt the weights for VCCS to 0.6 for color distance, 1.0 for surface normal difference and 0.3 for spatial regularity. We set $w_c = w_n = w_p$ and empirically defined $\tau_{merge} = 0.06$. Those settings showed to be a good trade-off between acquiring large, homogeneous segments while still separating classes of similar appearance, e.g. pictures and a wall.

We fixed the number of trees in the 3DEF to 60 and limit each tree to a maximum depth of 20 levels. Our experiments showed that, with the available amount of training data, no significant performance gain could be achieved by adding more trees or developing them deeper. The remaining forest parameters have been selected using grid search on a hold-out validation set of the NYUv2 dataset. Besides reaching the maximum depth level, we also stopped the expansion of the trees during training if the best information gain achieved dropped below 0.02 or if less than 5 points were left in a node to avoid overfitting. In each split node we randomly sampled 100 thresholds for each available unary feature and 1,000 random combinations of geometric constraints and binary test parameters for each available 3D entanglement feature. The larger number of samples for the latter is necessary due to the larger parameter space of the 3D entangled features (refer to Table 4.2).

As a final step after the learning procedure of the forest is done and the tree structure and split features have been learned, we further refine the label predictions in all leaf nodes by passing all available training data (instead of small bags) through the trees. The predictions are again weighted and we get the final label distributions in the leaf nodes of the 3DEF. While causing only a small decrease in overall global accuracy, this last step significantly improves the class accuracy score of the classifier. More details about the used performance metrics are given in the next section.

5.2.2 Performance Metrics

The two most widely used metrics to evaluate semantic segmentation methods are the *global accuracy (GA)* or *pixel accuracy* and the *average class accuracy (CA)*. Considering a label set with n class labels and based on the elements of the confusion matrix (true positives tp, false positives fp and false negatives fn), the metrics are defined as follows:

Global Accuracy

The global accuracy is defined by the number of all correctly labeled points of a scene over the total number of points in the scene:

$$GA = \frac{\sum_{i=1}^{n} tp_i}{\sum_{i=1}^{n} (tp_i + fn_i)} \tag{5.1}$$

Average Class Accuracy

The average class accuracy is the mean of the fractions of correctly labeled points per class.

$$CA = \frac{1}{n} \sum_{i=1}^{n} \frac{tp_i}{tp_i + fn_i} \tag{5.2}$$

Following these definitions, the global accuracy tends to be more biased towards dominant classes with a lot of points, while the average class accuracy is more sensitive to classes with a very strong or a very weak performance.

Jaccard Score

An alternative useful metric is the *Jaccard score*, also referred to as *intersection-over-union score*:

$$J = \frac{1}{n} \sum_{i=1}^{n} \frac{tp_i}{tp_i + fn_i + fp_i} \tag{5.3}$$

Weighted Jaccard Score

The Jaccard score can be modified to account for imbalanced test sets by weighting it with the individual class frequencies, yielding the weighted Jaccard score defined as follows:

$$J_w = \frac{1}{\sum_{i=1}^{n} (tp_i + fn_i)} \sum_{i=1}^{n} \frac{(tp_i + fn_i)\, tp_i}{tp_i + fn_i + fp_i} \tag{5.4}$$

However, most evaluations of semantic segmentation approaches only list the individual class accuracies as well as their average and the global pixel accuracy. Therefore, in this work, we focus on these two metrics for our comparisons.

5.2.3 Ground Truth Mapping

Both versions of the NYU Depth Dataset contain pointwise ground truth labels. However, both the RF classifier of the CRF pipeline and the 3DEF classify feature vectors computed for segments of points and not individual points. Consequently, in order to be able to train and evaluate the components, a mapping from pointwise ground truth labels to one label per segment needs to be established.

To that end, we generate new ground truth data corresponding to both segmentation stages. For each input point cloud, we first compute the respective segmentation. Then for each acquired segment, the distribution of all contained ground truth labels is calculated. Because of imperfections of the segmentation stage and also due to ambiguities in the ground truth labelings, some segments will contain multiple labels. To detect and handle these ambiguous segments, we apply the following scheme: From the label distribution calculated for each segment, we only consider the two labels with the most points in the segment. If the most frequent label contains more than twice as many point than the second-most frequent label in the segment, the top label is the ground truth label assigned to this segment. If the ratio between most frequent and second-most frequent label is less than 2, the segment ground truth label is considered too ambiguous and the segment is discarded from the training set.

5.3 Overall Quantitative Evaluation

We evaluate our methods against several other approaches. The work of Silberman et al. [81] represents the baseline for the NYUv1 dataset, for which they trained a neural network classifier, followed by a CRF. The CRF, however, does not incorporate specific class label relations and is only used to smooth the classification results. Also evaluated only NYUv1, Ren et al. [72] combine a superpixel Markov Random Field (MRF) with kernel descriptors and a segmentation tree. Their framework is very complex and takes over a minute per frame to compute, such that its results might be interpreted as a benchmark, but are achieved in a scope not directly comparable to ours.

When Silberman et al. presented the second version of their dataset [82], they focused only on a few structural labels to infer physical support relations between different classes. While their original label set of 4 classes (ground, struct, furniture and props) is not very meaningful for our robotics scope, nevertheless we list results on this set for completeness. The more informative evaluation on the second dataset is conducted on the label set consisting of 13 labels, introduced by Couprie et al. [9]. Their work represents one of the first approaches of applying Convolutional Neural Networks (CNNs) to the problem of semantic segmentation.

The fifth method we compare our methods against on both datasets and all label sets is the most similar. In a carefully engineered pipeline, Hermans et al. [32] also apply RF classification, followed by an efficient dense CRF implementation. In contrast to our method, however, they only use simple Potts models with hand-tuned parameters.

To evaluate the contributions of the individual stages of our methods, we report results for different configurations:

CRF Pipeline

1. **RF only**
   Apply only the RF classifier, without a refinement step using the dense CRF.

2. **Manual**
   Apply the RF classifier, followed by the refinement step with the dense CRF, but only with manually defined parameters and a simple Potts model for the label compatibility terms.

3. **Learned**
   Our full pipeline including the dense CRF with a full label compatibility matrix with learned parameters.

3D Entangled Forest

1. **Unary only**
   Only use locally computed unary features in the 3DEF classifier, which then corresponds to a regular RF.

2. **Full**
   Use the full available feature set including the new 3D entangled features to model contextual information.

5.3.1 NYU Depth Dataset v1

Quantitative evaluation results for NYUv1 are shown in Table 5.1. The complex framework of Ren et al. [72] shows by far the best performance compared to all of the other methods. However, its computation time of more than a minute per frame is not suitable for a real-time application on a mobile system.

Comparing the CRF pipeline to the more computationally efficient works of [81] and [32], we observe that the class accuracy scores of our method are generally lower. While enabling the dense CRF, particularly in the full configuration with learned compatibilities, significantly improves the overall global accuracy, the initial results obtained by the RF classifier seem to limit the performance of the whole pipeline on this dataset. Notably, two particular classes, namely *Blind* and *Window*, appear to be the most difficult classes which cannot be solved by the classifier, and therefore also not recovered by the CRF. Although we apply methods to counter the imbalance of the dataset, unarguably the RF is unable to properly distinguish underrepresented classes using the feature vectors calculated on the similar sized segments.

On the contrary, it can be observed that in our second pipeline, the features calculated on the larger segments are much more expressive. Using unary features only (which turns the 3DEF into a regular RF), our second pipeline already achieves competitive results. Enabling the 3D entangled features further boosts the average class and the global accuracy. Nevertheless, the average class accuracy is still about 7 % below [32], while our method shows a 6 % improvement regarding global accuracy.

5.3.2 NYU Depth Dataset v2

On the more balanced NYUv2 dataset, the results for both of our pipelines, presented in Table 5.2, clearly support our claim that being able to learn and incorporate contextual scene information is beneficial for semantic segmentation. Both of our methods improve the average class accuracy as well as the global accuracy compared to [9] and [32].

Based on strong results obtained by the RF classifier (we match the performance of the full pipeline of [32] only using the RF), the dense CRF with learned parameters of our first pipeline further improves the performance to a average class accuracy of 48.9 % and a global accuracy of 62.4 %, which is an improvement of more than 8 %. Analyzing the individual class accuracies, it can be observed that the dense CRF tends to be biased towards classes contributing the most to the global accuracy, i.e. classes representing large objects such as bed, furniture and wall. Nevertheless, the dense CRF also boosted the result of the more fine-grained *object* class by 15 %. On the other hand, the class with the largest performance decrease caused by the CRF is *TV*. A likely cause for this issue is the bad data quality for this class. The shiny surfaces of TVs barely reflect the infrared pattern of structured light sensors, such that the point cloud is very sparse and noisy in these areas, causing the segmentation method to produce unreliable results. In Figure 5.1, we further show the confusion matrix of our approach, once showing the results without the dense CRF and once for the full pipeline with learned parameters. Comparing the matrix for the full pipeline

Table 5.1: Evaluation scores for the NYUv1 dataset using 13 labels. The 13th label is an additional *Background* label which contains all points which do not belong to one of the first 12 labels. We show the individual class accuracies as well as their average and the global accuracy. RF/CRF refers to our first, 3DEF to our second pipeline. The results for [81, 32, 72] are directly taken from the respective papers.

Method	Bed	Blind	Bookshelf	Cabinet	Ceiling	Floor	Picture	Sofa	Table	TV	Wall	Window	Background	Class Avg.	Global
Silberman [81]	-	-	-	-	-	-	-	-	-	-	-	-	-	56.6	-
Hermans [32]	50.7	57.6	59.8	57.8	92.8	89.4	55.8	70.9	48.4	81.7	75.9	18.9	13.5	59.5	44.4
Ren [72]	**85**	**80**	**89**	**66**	**93**	93	**82**	**81**	60	**86**	82	**59**	35	**76.1**	-
RF/CRF (RF only)	17.6	0.6	56.5	12.6	82.6	94.5	40.9	47.3	42.7	28.4	62.7	12.5	24.4	40.3	45.3
RF/CRF (manual)	17.4	0.1	59.8	11.5	84.4	95.1	39.8	49.5	45.9	28.3	66.3	11.8	27.2	41.3	48.0
RF/CRF (learned)	1.4	0.0	34.3	4.3	71.7	93.3	18.3	27.2	20.7	10.5	58.6	0.1	**76.1**	32.0	**60.6**
3DEF (unary only)	39.6	12.1	47.0	44.3	88.8	95.2	27.6	56.1	53.4	46.4	88.4	11.1	17.4	48.3	49.1
3DEF (full)	54.2	13.5	62.9	48.8	88.7	**95.3**	29.1	59.9	**62.3**	46.3	**89.1**	11.9	14.1	52.0	50.2

Table 5.2: Class and global accuracy scores for the NYUv2 dataset, using 13 different semantic classes defined in [9]. RF/CRF refers to our first, 3DEF to our second pipeline. The results for [9, 32] are directly taken from the respective papers.

Method	Bed	Object	Chair	Furniture	Ceiling	Floor	Wall Deco.	Sofa
Couprie [9]	38.1	8.7	34.1	42.4	62.6	87.3	40.4	24.6
Hermans [32]	68.4	8.6	41.9	37.1	83.4	91.5	35.8	28.5
RF/CRF (RF only)	44.6	19.0	47.4	38.5	76.9	97.9	46.7	49.2
RF/CRF (manual)	47.7	18.8	51.1	41.4	75.4	98.3	**47.3**	56.2
RF/CRF (learned)	63.3	**34.1**	42.1	**59.2**	64.2	97.5	40.9	49.7
3DEF (unary only)	60.3	17.8	49.2	43.2	**86.0**	**98.7**	23.2	50.8
3DEF (full)	**74.6**	17.6	**62.2**	47.9	82.4	**98.7**	26.4	**69.4**

Method	Table	Wall	Window	Bookshelf	TV	Class Avg.	Global
Couprie [9]	10.2	**86.1**	15.9	13.7	6.0	36.2	52.4
Hermans [32]	27.7	71.8	**46.1**	45.4	38.4	48.0	54.2
RF/CRF (RF only)	37.4	55.5	35.2	36.6	46.7	48.6	53.8
RF/CRF (manual)	38.9	60.2	36.2	39.7	**47.1**	50.6	56.5
RF/CRF (learned)	35.4	72.7	23.7	41.7	10.8	48.9	62.4
3DEF (unary only)	46.5	82.4	25.8	27.8	31.5	49.5	60.5
3DEF (full)	**48.6**	83.7	25.6	**54.9**	31.1	**55.6**	**64.9**

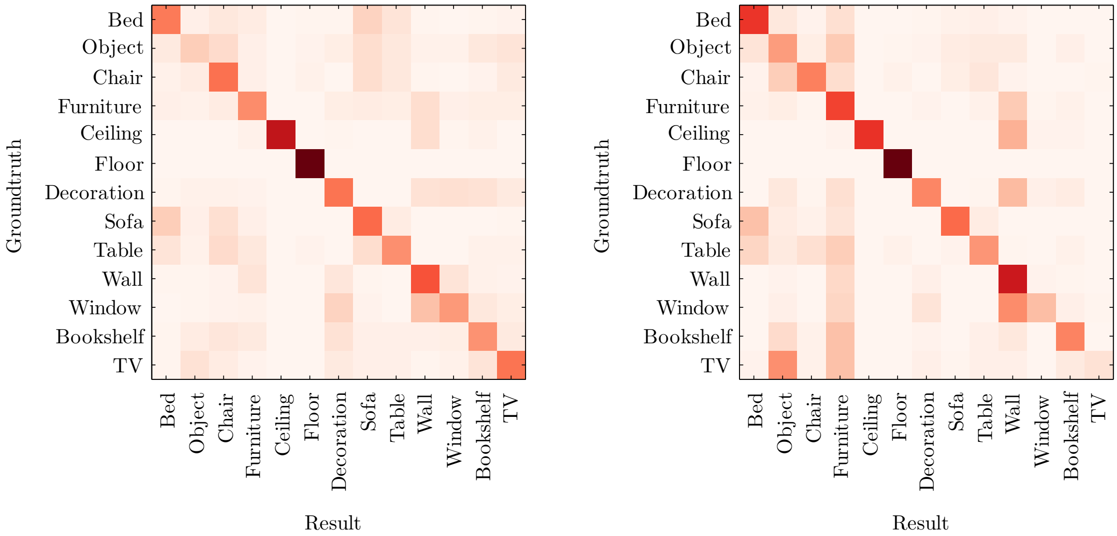


Figure 5.1: Confusion matrices of our first pipeline for the NYUv2 dataset with 13 labels. Left: Results after RF only. Right: Result for full pipeline with dense CRF with learned parameters.

to the system without the CRF, it can be observed that while the CRF reduces some ambiguities (e.g. for the decoration class), it also tends to increase the dominance for some labels. For instance, the furniture label shows a lot of false positives.

Analyzing the quantitative results for our 3DEF pipeline, it can be seen to further outperform the good results of our first method. It achieves a class accuracy score of 55.6 % and a global accuracy score of 64.9 %, which is a further improvement of 7 % and 3 %, respectively. On this dataset, which offers comprehensive labelings of many different scenes, the 3DEF unfolds its whole potential. While our classifier already outperforms [9] and [32] using unary features only, adding our context sensitive feature set boosts performance by another $5 - 6\,\%$ in average class and global accuracy. Analyzing the individual class accuracies, the classes profiting the most from the 3D entangled features are *Bed*, *Chair*, *Sofa* and *Bookshelf*, with improvements of up to 27 % compared to using unary features only.

This improvement is also visible in the confusion matrices, shown in Figure 5.2. It can be seen that the full 3DEF further strengthens the main diagonal of the confusion matrix, reducing the ambiguity between certain classes. The most remaining confusions appear between the labels *wall*, *wall decoration* and *window*, where the former is the dominant one. These confusions are mainly due to the segmentation stage, which sometimes wrongly merges segments of a picture or a window frame with the adjacent wall segments. These undersegmentations cannot be recovered by the 3DEF.

For the set of four structural labels of [82], the results of the quantitative evaluation are given in Table 5.3. Again, we compare our methods to the works of [9] and [32] and also add the baseline results for this label set, presented in [82]. Furthermore, we include the results from [85] and [63], which have only been evaluated on this label set of NYUv2.

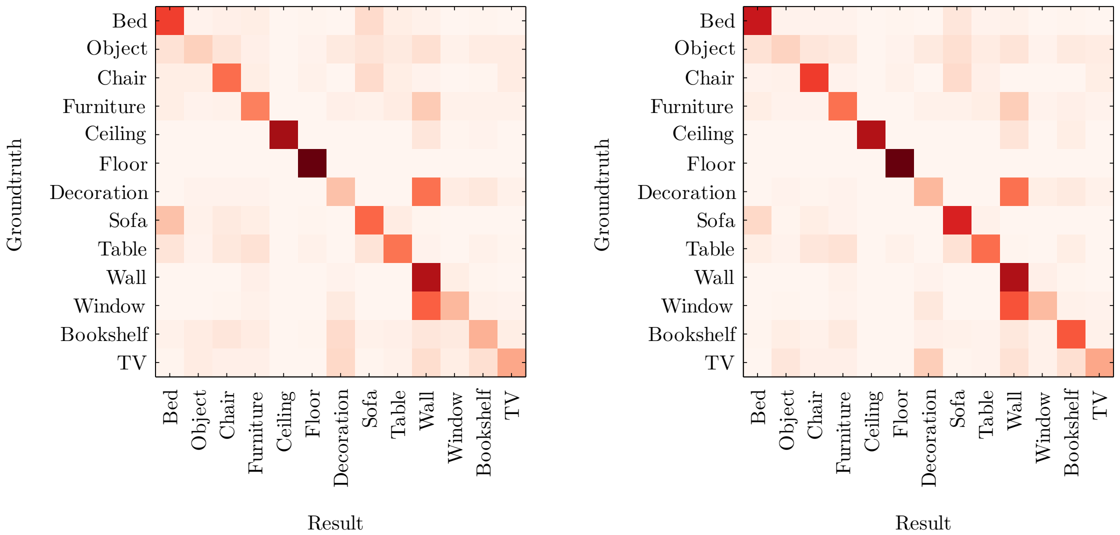


Figure 5.2: Confusion matrices of the 3DEF for the NYUv2 dataset with 13 labels. Left: Results for unary features only. Right: Results for full feature set including our new 3D entangled features.

Table 5.3: Class and global accuracy scores for the NYUv2 dataset using the smaller set of four structural classes defined by [82]. RF/CRF refers to our first, 3DEF to our second pipeline. The results for [82, 9, 85, 32, 63] are directly taken from the respective papers.

Method	Ground	Struct	Furniture	Props	Class Avg.	Global
Silberman [82]	68	59	70	42	59.6	58.6
Couprie [9]	87.3	86.1	45.3	35.5	64.5	63.5
Stückler [85]	95.6	83.0	**75.1**	14.2	66.8	70.6
Hermans [32]	97.4	76.1	61.8	40.9	69.0	68.1
Müller [63]	94.9	78.9	71.1	42.7	71.9	72.3
RF/CRF (RF only)	97.5	71.9	69.0	43.6	70.5	70.8
RF/CRF (manual)	97.8	74.2	72.1	39.5	70.9	72.1
RF/CRF (learned)	95.7	75.7	72.2	**43.8**	71.8	73.0
3DEF (unary only)	**98.8**	86.0	70.2	41.4	74.1	75.3
3DEF (full)	**98.8**	**87.0**	73.9	40.4	**75.0**	**76.7**

Similar to the larger label set, we can observe that for both pipelines, the classification results using the RF only already outperform all of the full pipelines of the other methods. As an interesting detail it can be seen that the new segmentation method of the second pipeline alone leads to a performance increase of $4-5\,\%$ from our first to the second method. Considering the nature of the label set, only including large structures, consisting of mainly planar segments, this is not surprising.

The performance of the CRF pipeline gradually increases with growing complexity of the pipeline, peaking at 71.8 % average class accuracy and 73.0 % global accuracy, using the full CRF model with learned label compatibilities.

Enabling the 3D entangled features in the 3DEF of the second pipeline further improves the good RF classification results, leading to the best overall result on this dataset with 75.0 % average class accuracy and 76.7 % global accuracy. For 3 out of 4 classes, the 3DEF achieves the highest individual class accuracy.

5.4 Qualitative Evaluation

We show some exemplary results of the different stages of both of our semantic segmentation pipelines for different scene types of the NYUv2 dataset in Figures 5.3 and 5.4.

5.4.1 CRF Pipeline

Examining the results of the CRF pipeline in Figure 5.3, we observe that globally, our method performs very well. Thus, it succeeds in labeling the major contents of the scene and correctly resolves different arrangements, containing severe occlusion between objects and various viewpoints. However, confirming the findings from the confusion matrix in the previous section (Figure 5.1), the results still exhibit some ambiguities, sometimes confusing the classes object and TV, bed and sofa or bookshelf and furniture. Nevertheless, analyzing the particular scenes, these confusions are reasonable in the sense that also the ground truth labels contained in the dataset are not always consistent and show the same ambiguities.

If we compare the results after the RF stage and after the dense CRF, the positive impact of the dense CRF on the overall result is clearly visible. While the RF classifier outputs a reasonable first result, due to the only local features it is very noisy and inconsistent. The dense CRF visibly smoothes the result, but is also capable of completely overturning misguided decisions by the RF, e.g. as it can be seen for the clothes rack in the third, the shelf full of objects in the fourth, and the bed in the last row of Figure 5.3.

5.4.2 3D Entangled Forest

The qualitative results of our 3DEF classifier, shown in Figure 5.4, confirm the quantitative improvement in classification accuracy compared to the CRF pipeline. Because the 3DEF operates on much larger individual segments of the input point cloud, the

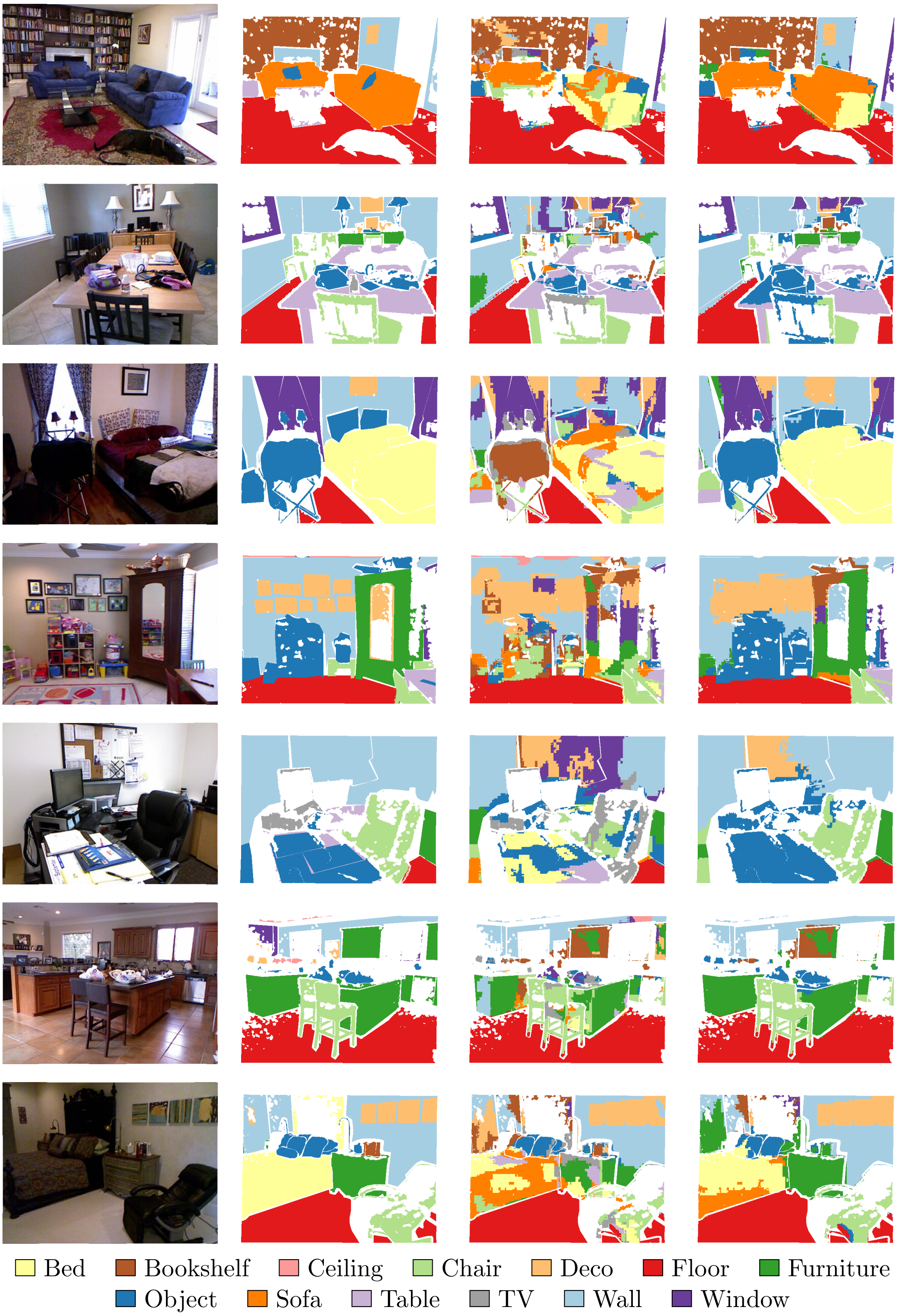


Figure 5.3: Example results of the CRF pipeline on version 2 of the NYU Depth dataset. First column: Input point cloud. Second column: Ground truth. Third column: Results after the RF, without the dense CRF. Last column: Results after the full pipeline including the dense CRF with learned parameters. Note that unlabeled areas or points without depth information are not shown.

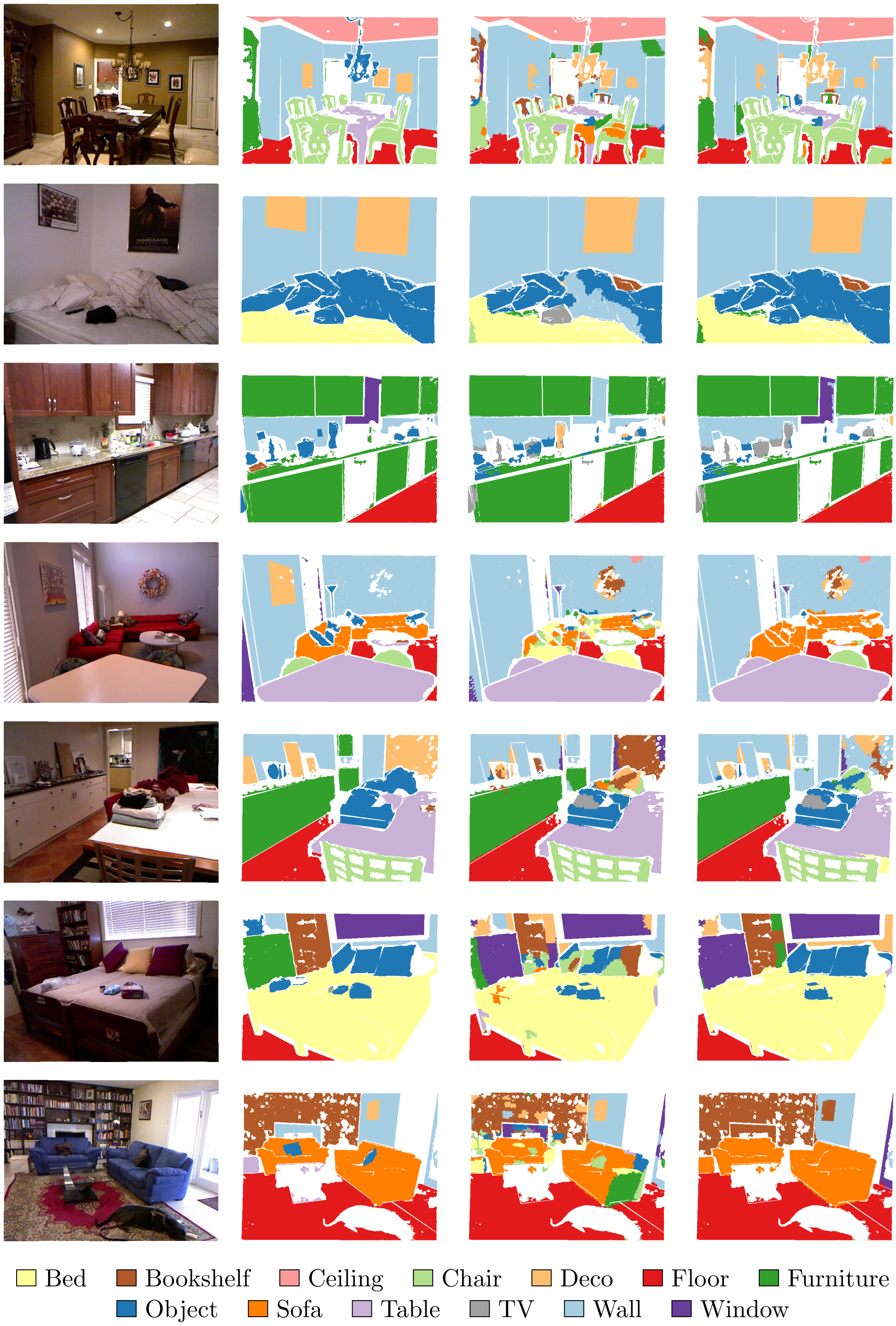


Figure 5.4: Qualitative results of our 3DEF pipeline for the NYUv2 Dataset. First column: Input point cloud. Second column: Ground truth. Third column: Standard RF with unary features only. Last column: Full 3DEF. Note that unlabeled areas or points without depth information are not shown.

calculated labelings exhibit much less classification noise than our first approach. The results also adhere much better to object boundaries, compared to the partly fuzzy labelings of the output of the dense CRF. Nevertheless, the applied segmentation stage has its limits, which is exemplified in rows two and four of Figure 5.4. For those scenes, the segmentation stage wrongly merges the supervoxels, corresponding to the pictures hanging on the wall, with the adjacent wall segments to one large segment. Because the 3DEF only assigns one label per segment, it then cannot distinguish between the pictures and the wall any more, such that the pictures are labeled as wall. This issue is also reflected in the confusion matrix shown in Figure 5.2, where the most frequent confusion is shown to be exactly *decoration* misinterpreted as *wall*.

Overall, however, these results underline that our new 3DEF classifier is capable of calculating accurate and smooth semantic segmentations of complex scenes, without the need of an additional refinement stage.

5.5 Parameter Analysis

In this section, we provide an in-depth analysis of the performance of both of our methods. First, we illustrate their capability of inferring and learning of contextual information of a scene. And second, we thoroughly examine the influence of different parameter settings on the overall performance and study, which of the calculated features provide the most information to the classifier.

5.5.1 CRF Pipeline

Learned Label Compatibilities

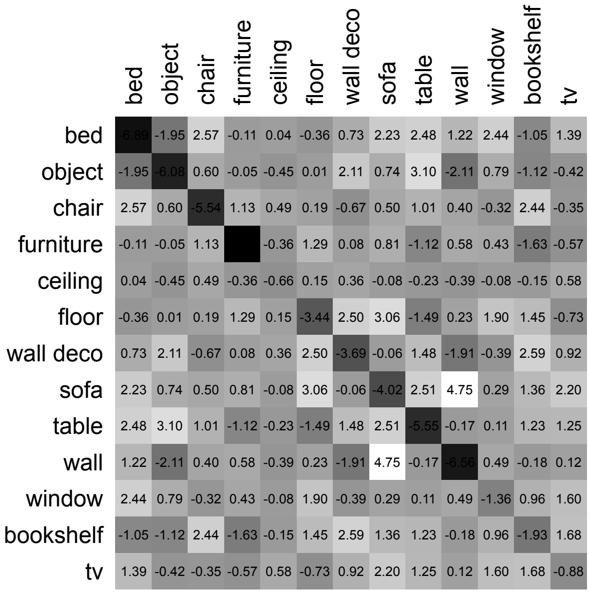
To prove the effectiveness of the learning approach of the dense CRF applied in our first pipeline, we analyze the learned parameters of the full label compatibility matrix on NYUv2, which is shown in Figure 5.5.

It can be seen that the main diagonal is very strong, showing that the dominant property of the dense CRF is indeed to smooth the noisy result from the RF classifier. Off the main diagonal, we observe various other strong relations between different class labels, which have been are identified by the algorithm. The most noticeable entries are wall/wall-deco, wall/object and bed/object.

We further analyze the influence of different parameters during the training stage of the RF, in particular the maximum tree depth d_{max} and the number of trees t.

Influence of Random Forest Parameters

In Figure 5.6, we show how the evaluated maximum tree depth influences the performance of the RF, while we fix the number of used trees to the maximum of 60. Please note that all results have been averaged across 10 different learned trees to reduce the influence of the randomness injected during the learning procedure.



	bed	object	chair	furniture	ceiling	floor	wall deco	sofa	table	wall	window	bookshelf	tv
bed	-6.89	-1.95	2.57	-0.11	0.04	-0.36	0.73	2.23	2.48	1.22	2.44	-1.05	1.39
object	-1.95	-6.08	0.60	-0.05	-0.45	0.01	2.11	0.74	3.10	-2.11	0.79	-1.12	-0.42
chair	2.57	0.60	-5.54	1.13	0.49	0.19	-0.67	0.50	1.01	0.40	-0.32	2.44	-0.35
furniture	-0.11	-0.05	1.13		-0.36	1.29	0.08	0.81	-1.12	0.58	0.43	-1.63	-0.57
ceiling	0.04	-0.45	0.49	-0.36	-0.66	0.15	0.36	-0.08	-0.23	-0.39	-0.08	-0.15	0.58
floor	-0.36	0.01	0.19	1.29	0.15	-3.44	2.50	3.06	-1.49	0.23	1.90	1.45	-0.73
wall deco	0.73	2.11	-0.67	0.08	0.36	2.50	-3.69	-0.06	1.48	-1.91	-0.39	2.59	0.92
sofa	2.23	0.74	0.50	0.81	-0.08	3.06	-0.06	-4.02	2.51	4.75	0.29	1.36	2.20
table	2.48	3.10	1.01	-1.12	-0.23	-1.49	1.48	2.51	-5.55	-0.17	0.11	1.23	1.25
wall	1.22	-2.11	0.40	0.58	-0.39	0.23	-1.91	4.75	-0.17	-6.56	0.49	-0.18	0.12
window	2.44	0.79	-0.32	0.43	-0.08	1.90	-0.39	0.29	0.11	0.49	-1.36	0.96	1.60
bookshelf	-1.05	-1.12	2.44	-1.63	-0.15	1.45	2.59	1.36	1.23	-0.18	0.96	-1.93	1.68
tv	1.39	-0.42	-0.35	-0.57	0.58	-0.73	0.92	2.20	1.25	0.12	1.60	1.68	-0.88

Figure 5.5: Label compatibility terms learned for version 2 of the NYU dataset. The darker the entry in the matrix (or the smaller the value), the more likely the two corresponding labels occur close to each other.

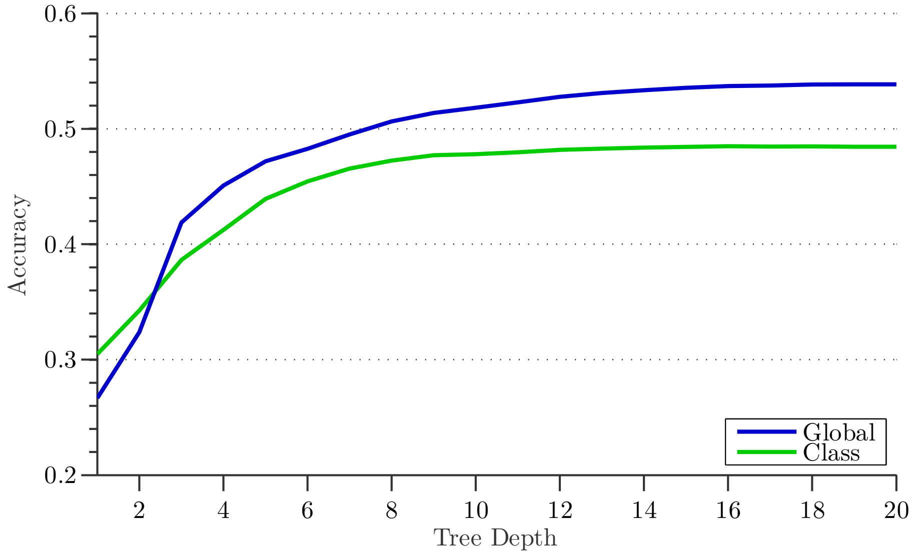


Figure 5.6: Influence of the maximal evaluated tree depth d_{max} on the RF classification accuracy. Evaluating the tree for deeper levels than 12 does not result in any more significant performance gains. For this experiment, the number of trees has been fixed to 60.

The average class accuracy already reaches its best value at a tree depth of 9, while the global accuracy saturates at a depth of 12. Thus, on one hand it is not necessary to run the inference procedure for deeper levels, which saves processing time. On the other hand, however, if we take the number of different classes into account (in this case 13), it is also is an indicator that the expressiveness of the used feature vector is limited.

In the next experiment we fix the maximum depth value to 20 and examine, how the performance scales with the number of trees in the forest. The results, presented in Figure 5.7 show that the highest accuracies are already reached after about 20 trees, with only very minor improvements afterwards. Adding too many trees to a forest does not only increase processing time. Furthermore, the resulting label probabilities for each data point are “washed out”, since they are averaged across more and more individual tree results.

5.5.2 3D Entangled Forest

Performance Gain by 3D Entangled Features

In a first experiment, we examine the impact of the 3D entangled features on the overall performance of the 3DEF in more detail. To this end, in Figure 5.8, we compare both the average class accuracy and the global accuracy on the NYUv2 dataset, depending on the maximal evaluated tree depth, for two different forest configurations. To reduce the effect of the injected randomness in the forests, all presented results are averages over the results of 10 different learned forests.

In the first configuration, the 3DEF has been learned using unary features only, which turns it into a regular RF classifier, similar to the one used in the CRF pipeline. For the second configuration, we activate the new 3D entangled features, which capture more global contextual information, at depth level 6. It can be clearly observed that the activation of the additional features immediately improves the accuracy of the classifier. With increasing tree depth, the new features build up a significant performance gain, which saturates at a $5-6\,\%$ increase compared to the unary feature set.

This experiment nicely illustrates the capability of the new 3D entangled features to overcome some of the limitations of a local RF classifier by incorporating contextual information. Another interesting finding is that the combination of the segmentation method and the unary features calculated in this pipeline compares favorably to their counterparts of our first pipeline. Observing that both the global and the average class accuracy reach their maximum values for deeper depth levels than for our first pipeline, the features appear to be more descriptive. Adding our new 3D entangled features, the accuracies peak even later, which is another indicator for their expressiveness.

In the following experiments, we will further evaluate which features in particular are the most informative ones, and examine learned contextual relations between different labels.

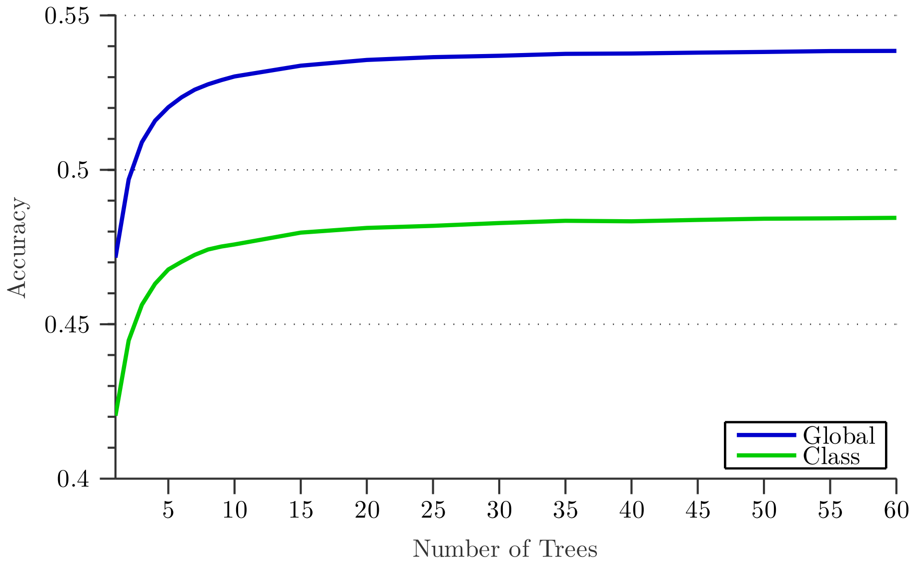


Figure 5.7: Performance of the RF classifier of the CRF pipeline depending on the number of trees of the forest. The accuracy quickly increases but saturates already at approximately 10 trees. For this experiment, the maximum tree depth has been fixed to 20.

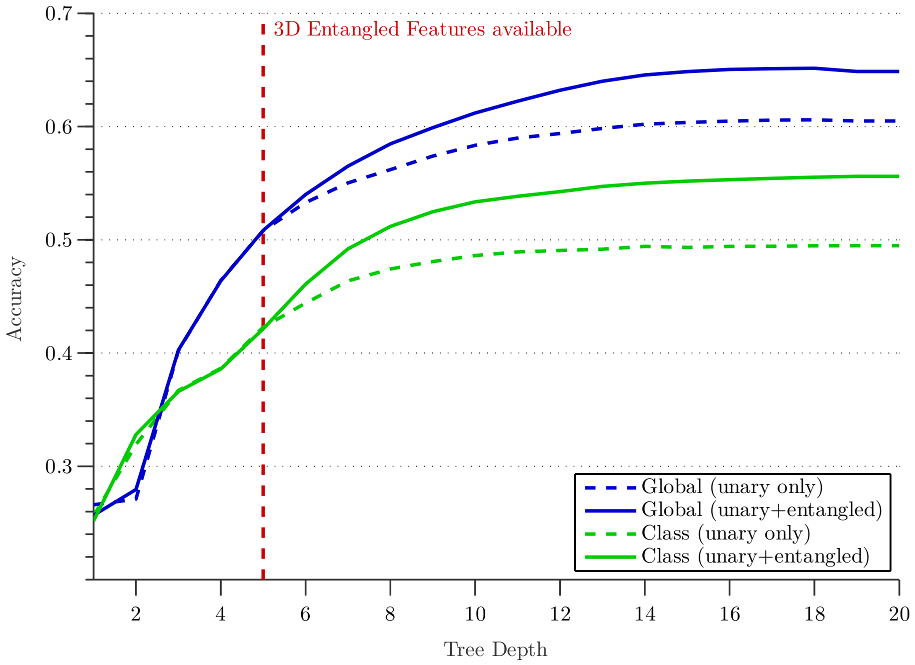


Figure 5.8: The proposed 3D entangled features, activated at depth level 6, significantly improve the performance regarding the average class accuracy as well as the overall global accuracy of the classifier. For this experiment, the number of trees has been fixed to 60.

Feature Analysis

To examine the importance of the different features in more detail, we analyze a 3DEF of 60 trees, learned on the NYUv2 dataset. For each depth level, we calculate a separate histogram over all available features types which have been selected by the training procedure at the respective level. Fig. 5.9 shows a visualization of these histograms as a heat map, normalized across all 60 trees. The features which have been selected the most at each depth level and are therefore the most suitable ones to separate the classes are shown in bright colors. Again, the new 3D entangled features are activated at depth level 6, such that in the first 5 levels only unary features are highlighted.

Clearly, the most distinctive of the unary features are the height and surface normal features, namely, the minimum and maximum height of a segment, its bounding box dimensions and its mean surface angle with respect to the ground plane. This is an intuitive result, since many of the classes can generally be distinguished from each other using height and orientation. E.g. walls are vertical and likely to be the highest structures in a scene, whereas the floor and table planes are horizontal and located at particular heights above the ground plane.

Other unary features which show to carry important information are the vertical elongation and the thickness of a segment. Regarding the different color channels of the CIELAB space, the L channel, capturing the illumination, appears to be the most discriminative one for our application.

As soon as the 3D entanglement features are activated (level 6), they immediately become the most frequently selected features of the whole set, with the *TopN Segment Feature* being on top. The least distinctive of the 3D entanglement features appears to be the *Node Descendant Feature* with a slight drop in importance with increasing tree depth.

In general, this analysis agrees with the findings for the 2D case of [61] and demonstrates the feasibility of our new feature set, with the 3DEF preferring all 3D entanglement features over the unary feature set at deeper tree levels.

Influence of the Forest Size

We also evaluate how the size of the 3DEF, i.e. the number of used trees, influences the overall performance of the classifier. In Figure 5.10 it can be seen that, similar to the results for the CRF pipeline, the accuracy quickly increases until approximately 10 trees have been added, before it finally saturates after adding about 20 trees to the forest.

Contextual Relations Learned by 3D Entangled Features

In our last evaluation we demonstrate the specific capability of the 3D entanglement features to learn informative spatial and contextual relations between objects and structures in the scene. To that end, we directly analyze the parameters of the *TopN Segment Feature*, learned on the NYUv2 dataset.

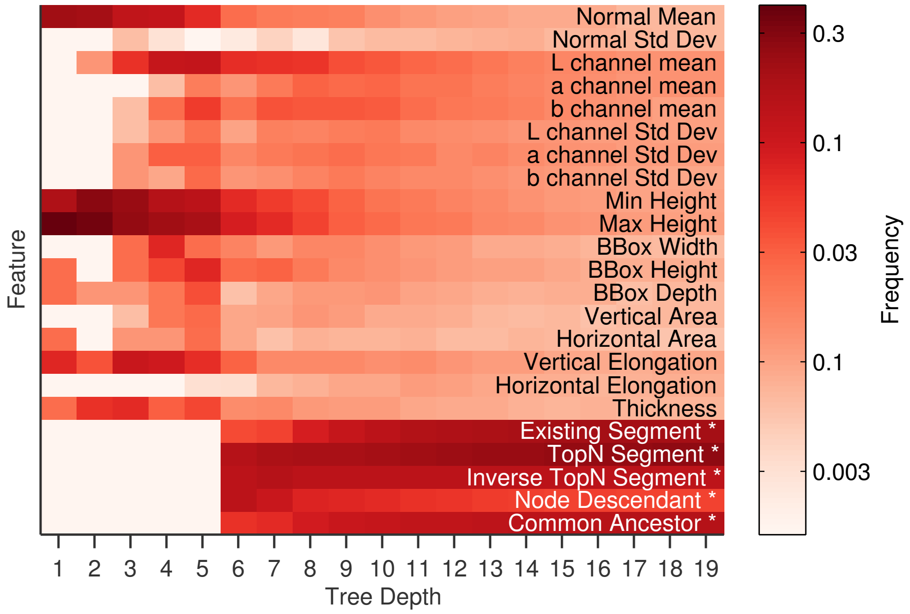


Figure 5.9: Distribution of learned split features, depending on the depth level of the trees (3D entangled features marked with *). The darker the cell, the more often the feature has been selected at a particular depth (data accumulated over 60 trees; logarithmic scale for better readability).

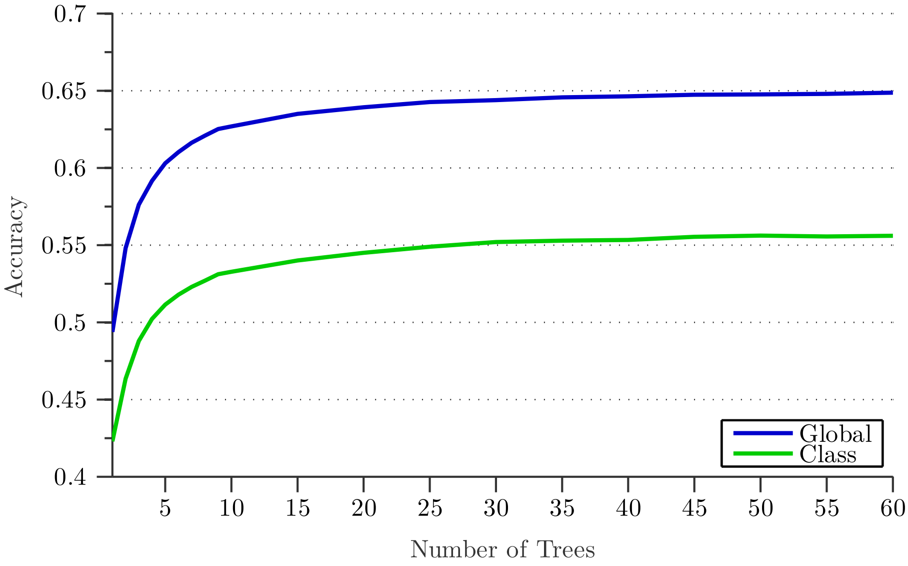


Figure 5.10: Performance of the 3DEF depending on the number of trees of the forest. After a steep increase both average class and global accuracy saturate after adding about 20 trees. For this experiment, the maximum tree depth has been fixed to 20.

Traversing all trees in the learned forest, each time the feature is used in a split node, we calculate which class label "profits" the most from the split, i.e. which class label's probability encounters the largest increase, comparing the right child node of the split node to the split node itself (since all points, for which the feature evaluates to `true`, move to the right child node). Consequently, we can relate this class label (which we denote *target label*), to the label which is learned to be most informative in the TopN binary test of the feature (denoted *split label*). Finally, for each pairwise target and split label combination, we compute histograms of the learned geometric constraint parameters. Observing the most prominent bins, the most informative geometric relation in combination with a specific contextual relation can be found.

In Fig. 5.11, we show three examples of learned correlations, proving the capability of our features to explicitly capture and exploit frequently observed relations between class labels. With peaks around 0°, 70° and 290° in the vertical angle histogram, the first example illustrates that the features learned that table planes are generally parallel to the seat or approximately perpendicular to the back rest of a chair. The second example visualizes that finding a *Decoration* segment (e.g. a picture) parallel to the target segment is a strong indicator of the target segment belonging to a wall. The last example demonstrates that the features also learn relations between segments of the same class and therefore perform context-sensitive smoothing. In this case, the peaks at $0°, 90°, 180°$ and 270° in the horizontal angle histogram nicely show that the perpendicularity of furniture has been correctly picked up by the 3DEF.

5.6 Runtime Analysis

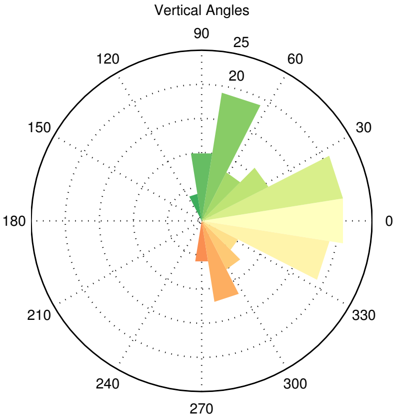
In a robotics scope, resources such as computation power and available processing time are limited. Therefore, it is very important that algorithms do not only obtain good results, but are also computationally efficient. In this section, we evaluate the runtimes of the different components of our pipelines and further analyze, how our new 3DEF classifier scales with different parameters.

5.6.1 Full Pipelines

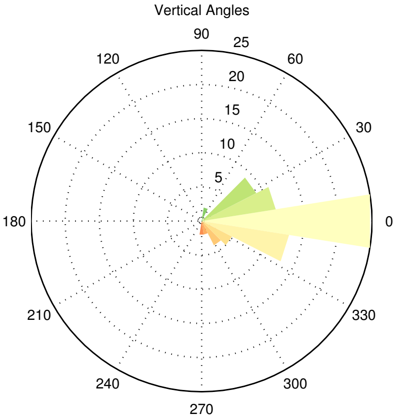
In Tables 5.4 and 5.5, we give an overview on the approximate runtimes of the different components of our two pipelines. The numbers are based on the experiments conducted on our test machine, an i7 laptop with 8 cores clocked with 2.4 GHz.

In our first pipeline, the feature calculation for each patch and the RF classification are fully parallelized, as well as parts of the oversegmentation stage. For single Kinect point clouds with a resolution of 640 × 480 points we achieve an average processing time of approximately 500 ms. Thus, compared to the similar approach of [32], our framework is more than twice as fast.

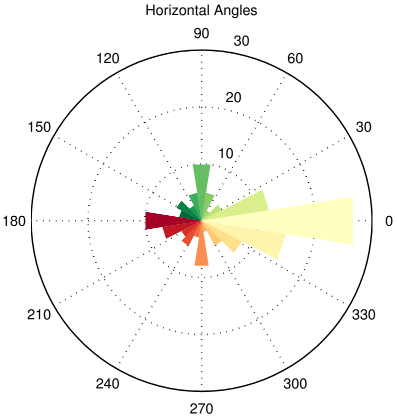
The slowest part of the pipeline is the oversegmentation stage, which performance also depends on the input point cloud. Because the seeding density of the VCCS segmentation method scales inversely proportional with the distance to the camera, point



(a) Chair → Table



(b) Decoration → Wall



(c) Furniture → Furniture

Figure 5.11: Examples for strong contextual and spatial relations between classes, learned by the *TopN Segment Feature*. The first class name is the learned informative label of the feature (split label), the second is the class which has been split best by the feature using the depicted angular constraints (target label).

clouds with larger distances to the camera will be segmented in less, but larger segments. However, in the VCCS implementation available in the Point Cloud Library, this setting is slower than calculating more, but smaller segments on a point cloud with short distances from the camera.

Table 5.4: Approximate runtimes of the separate stages of the CRF pipeline. Timings have been measured on an i7 laptop ($8 \times 2.4\,\mathrm{GHz}$) using the experimental settings described in section 5.2. On average, 2 Kinect point clouds per second can be processed.

Processing Stage	Runtime
Oversegmentation	$200 - 300\,\mathrm{ms}$
Feature extraction	$120\,\mathrm{ms}$
RF classification	$< 5\,\mathrm{ms}$
CRF, setup and inference (3 iterations)	$100\,\mathrm{ms}$
Total processing time	$\approx 500\,\mathrm{ms}$

Table 5.5: Approximate runtimes of the separate stages of our second pipeline with a 3DEF. Timings have been measured on an i7 laptop ($8 \times 2.4\,\mathrm{GHz}$) using the experimental settings described in section 5.2.

Processing Stage	Runtime
Preprocessing	$45\,\mathrm{ms}$
Segmentation	$580\,\mathrm{ms}$
Feature extraction	$20\,\mathrm{ms}$
Classification	$85\,\mathrm{ms}$
Total processing time	$730\,\mathrm{ms}$

For our second pipeline, the imbalance in processing time of the separate steps is even larger. The segmentation stage takes approximately 7 times longer than the next most time-intensive stage, which is the 3DEF classification. We are convinced that there is still a lot of room for improvement in the segmentation stage, however, this stage not the focus of this work. Nevertheless, the overall processing time of our pipeline is still far below 1 second per frame, which makes it very suitable for an online application on a mobile robot.

5.6.2 Scalability of the 3DEF

In our last set of experiments, we evaluate how the classification time of a 3DEF scales with different parameters of the forest.

First, we empirically analyze the connection between classification time and forest size, that is, number of trees t. Since the inference procedure for RFs treats all trees of a forest independently from each other, inference time is expected to scale linearly, which is confirmed in the results shown in Figure 5.12.

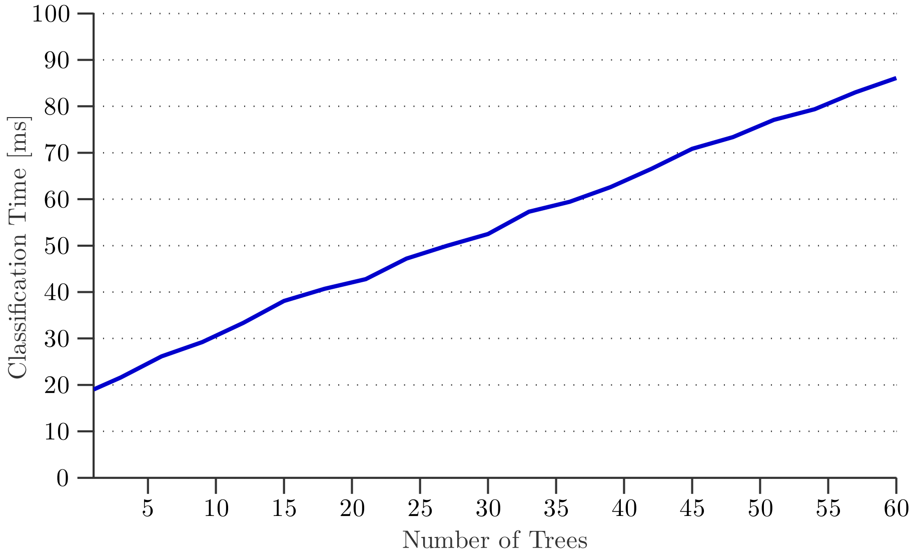


Figure 5.12: Influence of the number of trees on the classification time of the 3DEF. Results are averages over 10 forests.

Figure 5.13 illustrates the dependency on the maximally evaluated depth level. Two interesting findings can be observed:

1. Compared to unary features, the 3D entangled features, activated at depth level 6, take significantly more computation time during inference. This can be explained by the geometric constraints, which have to be applied during the evaluation of the 3D entangled features, filtering the candidate segments from all available segments in the scene. This procedure includes the calculation of all pairwise point-to-plane distances and angular differences between the segments, which is the computationally most expensive part.

2. Although theoretically the number of nodes in the forest and therefore the computation time should grow exponentially with increasing depth level, it actually only grows sublinearly. This is due to the fact that during learning at each depth level a growing fraction of the newly learned nodes are turned into leaf nodes because the maximal achievable information gain decreases with increasing depth.

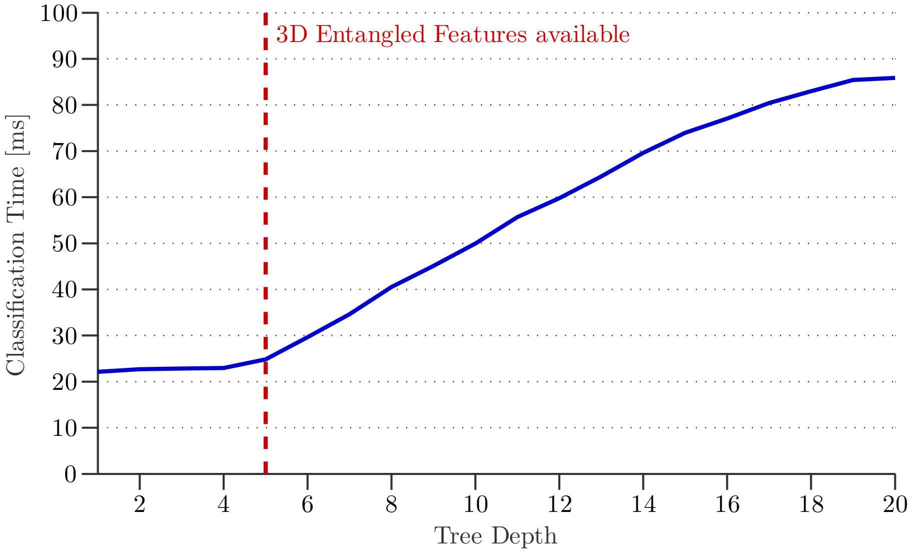


Figure 5.13: Influence of the maximal evaluated tree depth on the classification time of the 3DEF with 3D entangled features activated at depth level 6. Results are averages over 10 forests.

Chapter 6

Conclusion and Outlook

Holistic scene understanding is a capability where human visual perception is still far superior to any computer vision system. However, breaking down this large complex problem into smaller subtasks gradually enables autonomous systems to comprehend and intelligently react to their surroundings.

One important cornerstone on the way to more complete and advanced scene understanding systems are methods for accurate, but at the same time efficient semantic segmentation, which is the key contribution of this thesis.

6.1 Summary

Semantic segmentation of images or point clouds of indoor scenes is a very challenging problem. Scenes are composed of hundreds of different types and variations of objects which, observed from a single camera, mutually occlude each other and exhibit arbitrary appearances and shapes. Consequently, processing different parts of a scene individually leads to ambiguous and noisy predictions for semantic labels. On the other hand, however, most of the environments of our everyday life show common patterns of how different objects are arranged, providing valuable information about the scene. Exploiting this fact, in this thesis we have presented two different approaches for semantic segmentation of 3D point clouds, which are able to learn and model this global, contextual knowledge to improve semantic label predictions for indoor scenes.

Our first approach, described in chapter 3, applies a fully connected dense CRF to refine the coarse and noisy classification result of an RF classifier. Taking geometry and appearance properties into account, long-range connections in the CRF allow contextual information, modeled by a label compatibility matrix, to be propagated across large distances in 3D space. These pairwise compatibilities are entirely learned from training data and reflect frequently appearing object combinations and arrangements. By optimizing the resulting energy formulation of the CRF model, these common object patterns are emphasized in the final result and overturn ambiguous predictions from the classifier.

The pairwise terms of the CRF are modeled as linear combinations of Gaussian kernels, such that we can apply a very efficient inference method based on mean field approximation. Combined with a compact unary feature vector and the easily parallelizable RF classifier, the entire pipeline is able to process two Kinect point clouds per second, which is significantly faster than comparable methods.

The evaluation on the two competitive NYU Depth datasets, presented in chapter 5, underlines the strengths, but also shows some weaknesses of our first approach. On one hand, it achieves superior results to comparable methods on the second version of the dataset. On the other hand, for the first, very imbalanced version, the calculated features and the resulting quality of the RF prediction appears to be a limiting factor.

Tackling the shortcomings of our first pipeline, in chapter 4 we have introduced 3D Entangled Forests, a different approach for context learning and modeling. We have unified the classification and refinement step, resulting in a very compact and efficient framework. Extending the standard RF architecture with a concept called *entanglement* enables the classifier to directly incorporate global contextual information of a scene, such as frequent geometric arrangements of different classes.

In the conducted evaluation, the 3DEF achieved competitive results on the NYUv1 dataset and clearly outperformed comparable methods, as well as our first pipeline, on two different label sets of the NYUv2 dataset. We further showed the efficiency of the new 3D entangled features in several experiments, highlighting their ability to explicitly learn and model useful contextual and geometric information, e.g. between tables and chairs. Finally, in a detailed runtime analysis we also demonstrate the computational efficiency of our new classifier, underlining its suitability for the application on an autonomous mobile system.

6.2 Outlook

In this thesis, we have presented two different approaches to improve semantic segmentation of indoor scenes by incorporating contextual information from their respective constituent objects. Both of our methods focus on a computationally efficient calculation of class label predictions for each point of a single frame point cloud. However, considering a mobile robot accumulating semantic knowledge about its environment, the question arises how to translate these pointwise semantic predictions into a more abstract semantic representation as a basis for further exploitation.

6.2.1 From Single Frame Predictions to Semantic Maps

To provide a robot with a more complete model of its immediate surroundings, the single frame predictions calculated by our pipelines could be integrated in a full 3D model, which is also referred to as a *semantic map*. Generally, our 3DEF is not limited to single frame point clouds but can also directly be applied to full 3D reconstructions of a scene. In Figures 6.1 and 6.2 we show example results of the 3DEF on reconstructions of two indoor scenes, calculated by two different reconstruction methods. While the overall qualitative results seem reasonable, they also exhibit several confusions between class labels. This is mainly due to the fact that because of the lack of fully annotated 3D reconstructions, the classifier has only been trained on single frame point clouds, which have different properties compared to reconstructions obtained from multiple frames. Besides the limited field of view, in contrast to full reconstructions, single frame point clouds have less point density, and the density also drastically decreases with increasing distance from the camera. Consequently, a different initial segmentation method should be provided if the 3DEF were applied on reconstructions instead of single frames. Nevertheless, it would be interesting to analyze the performance of a 3DEF which has been trained on a dataset containing dense labelings of full 3D reconstructions.

In contrast to labeling an already obtained full reconstruction with the 3DEF, an iterative approach could fuse single frame predictions of the classifier online during the reconstruction. An interesting method tackling this problem has recently been presented in [89] and [90].

6.2.2 From Pointwise Predictions to Object Instances

Obtaining a pointwise semantic labeling of a scene is a first step to establish semantic knowledge about an environment in an autonomous system. However, while a pointwise representation is useful to fuse multiple frames into a full 3D reconstruction and create a semantic map, we claim that it is not the most suitable model to deduct informed decisions from. First and foremost, a pointwise representation does not include a notion of object instances. For example for a dining room, a pointwise semantic map contains hundreds or thousands of points labeled “chair” or “table”, but no information about the exact number of chairs and their spatial arrangement. However, these properties are essential for a system to be able to fully understand and interpret an observed scene.

Consequently, we consider the transfer from a pointwise representation to a more abstract, high-level model of a scene as a crucial next step towards fully autonomous holistic scene understanding.

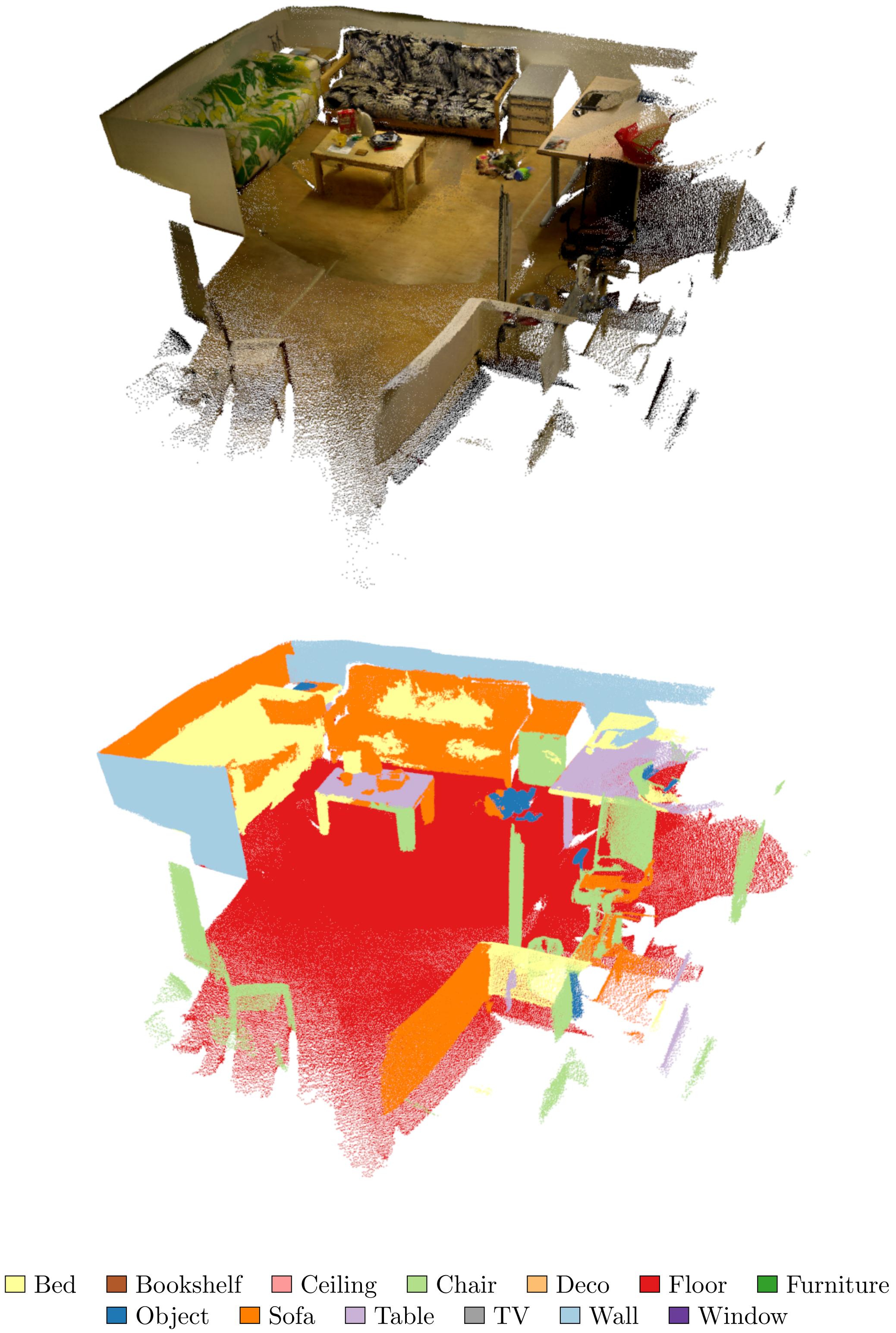


Figure 6.1: Example result of our 3DEF, trained on the NYUv2 dataset, on a 3D reconstruction of a living room. The reconstruction has been obtained using ElasticFusion [93, 94]. The most confusion appears between the sofa and bed label.

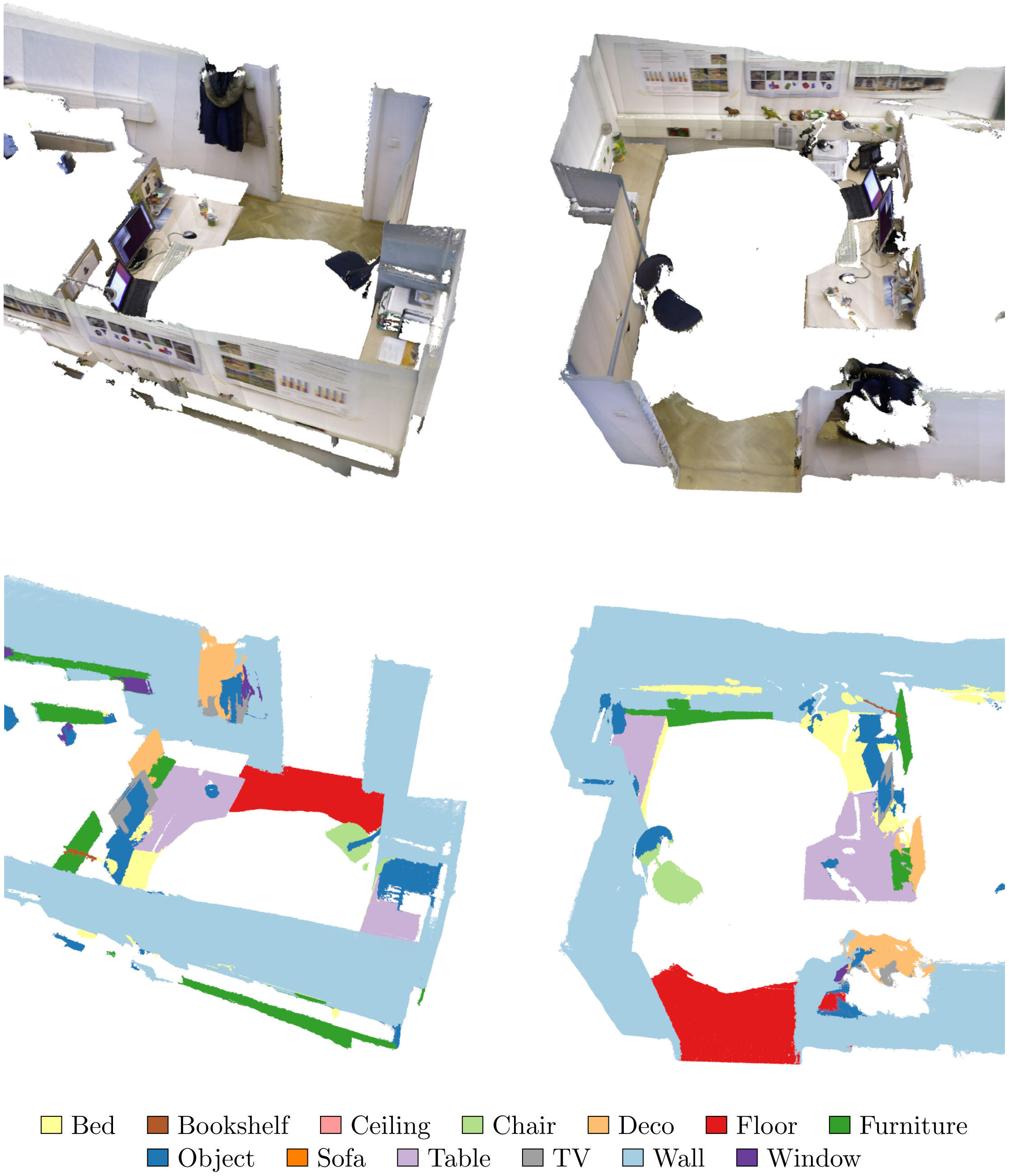


Figure 6.2: Example result of our 3DEF, trained on the NYUv2 dataset, on a 3D reconstruction of an office scene. The reconstruction has been obtained using a reconstruction method based on a KLT tracker [70].

List of Figures

1.1 Semantic decomposition of a dining scene distinguishing different types of furniture (chairs, table, cabinet, lamp, pictures) and the structural components of the room (floor, wall, ceiling). 2
1.2 Challenging scenes for semantic segmentation of 3D point clouds using a structured light sensor. 4
1.3 Overview of both presented methods for context learning for semantic segmentation. 6

2.1 Example of a trained decision tree. 16
2.2 Visualization of a decision tree split function. 17
2.3 RF inference procedure. 21
2.4 Different methods to dynamically prune an RF at inference time. 23

3.1 Overview of the semantic segmentation pipeline based on an RF and a dense CRF. . 26
3.2 Example result of the Voxel Cloud Connectivity Segmentation method. 27
3.3 Intermediate labeling result from the RF classifier. 30

4.1 Overview of the second pipeline using a 3DEF for classification and context learning. 36
4.2 Effect of bilateral filtering applied to an input point cloud. 37
4.3 Example results of the segmentation stage of our 3DEF pipeline. 39
4.4 Examples of learned geometric constraints of a 3DEF. 42
4.5 TopN and Inverse TopN Segment Features. 43
4.6 Node Descendant Feature. 44
4.7 Common Ancestor Feature. 45

5.1 Confusion matrices of our first pipeline for the NYUv2 dataset with 13 labels. 59
5.2 Confusion matrices of the 3DEF for the NYUv2 dataset with 13 labels. 60
5.3 Example results of the CRF pipeline on version 2 of the NYU Depth dataset. 62
5.4 Qualitative results of our 3DEF pipeline for the NYUv2 Dataset. 63
5.5 Label compatibility terms of the dense CRF learned for version 2 of the NYU dataset. 65
5.6 Influence of the maximal tree depth on the RF performance. 65
5.7 Influence of the number of evaluated trees on the RF performance. 67
5.8 Performance increase of the 3DEF due to 3D entangled features. 67
5.9 Evaluation of most informative features of 3DEF. 69
5.10 Influence of the number of evaluated trees on the 3DEF performance. 69
5.11 Strong contextual and spatial relations between classes, learned by the 3DEF. 71
5.12 Classification time of the 3DEF depending on the number of trees. 73
5.13 Classification time of the 3DEF depending on the tree depth. 74

6.1 Result of a 3DEF on a 3D reconstruction of a living room. 78
6.2 Result of a 3DEF on a 3D reconstruction of an office scene. 79

List of Tables

3.1 Unary feature vector calculated in the CRF pipeline. 28

4.1 Unary features calculated for the 3DEF. 40
4.2 Summary of 3D entangled features 45

5.1 Quantitative results of our methods on NYUv1 with 13 different classes. 57
5.2 Quantitative results of our methods on NYUv2 with 13 different classes. 58
5.3 Quantitative results of our methods on NYUv2 with four structural classes. 60
5.4 Runtimes of the CRF pipeline. 72
5.5 Runtimes of the 3DEF pipeline. 72

Bibliography

[1] A. Anand, H. S. Koppula, T. Joachims, and A. Saxena. Contextually guided semantic labeling and search for three-dimensional point clouds. *International Journal of Robotics Research (IJRR)*, 32(1):19–34, 2012.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.

[3] M. Bajones, D. Wolf, J. Prankl, and M. Vincze. Where to look first? Behaviour control for fetch-and-carry missions of service robots. In *Austrian Robotics Workshop (ARW)*, 2014.

[4] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE, 2007.

[5] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[6] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *International Conference on Machine Learning (ICML)*. ACM, 2014.

[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *arXiv preprint arXiv:1606.00915*, 2016.

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

[10] A. Criminisi and J. Shotton, editors. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer London, 2013.

[11] A. Criminisi, J. Shotton, D. Robertson, and E. Konukoglu. Regression forests for efficient anatomy detection and localization in CT studies. In *International MICCAI Workshop on Medical Computer Vision*, pages 106–117. Springer, 2010.

[12] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893. IEEE, 2005.

[13] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[14] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2650–2658, 2015.

[15] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1691–1696. IEEE, 2012.

[16] G. Fanelli, J. Gall, and L. Van Gool. Real time head pose estimation with random regression forests. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 617–624. IEEE, 2011.

[17] C. Farabet, C. Couprie, L. Najman, and Y. Lecun. Scene parsing with multiscale feature learning, purity trees, and optimal covers. In *International Conference on Machine Learning (ICML)*, pages 575–582. ACM, 2012.

[18] Li Fei-Fei, A. Iyer, C. Koch, and P. Perona. What do we perceive in a glance of a real-world scene? *Journal of Vision*, 7(1):10–10, 2007.

[19] Li Fei-Fei, C. Koch, A. Iyer, and P. Perona. What do we see when we glance at a scene? *Journal of Vision*, 4(8):863–863, 2004.

[20] U. Franke and A. Joos. Real-time stereo vision for urban traffic scene understanding. In *IEEE Intelligent Vehicles Symposium*, pages 273–278. IEEE, 2000.

[21] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *Decision forests for computer vision and medical image analysis*, pages 143–157. Springer, 2013.

[22] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun. 3D traffic scene understanding from movable platforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 36(5):1012–1025, 2014.

[23] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (6):721–741, 1984.

[24] E. Geremia, O. Clatz, B. H. Menze, E. Konukoglu, A. Criminisi, and N. Ayache. Spatial decision forests for MS lesion segmentation in multi-channel magnetic resonance images. *NeuroImage*, 57(2):378–390, 2011.

[25] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[26] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, Sept 2009.

[27] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[28] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *European Conference on Computer Vision (ECCV)*, 2014.

[29] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla. SceneNet: Understanding real world indoor scenes with synthetic data. *arXiv preprint arXiv:1511.07041*, 2015.

[30] G. Heitz, S. Gould, A. Saxena, and D. Koller. Cascaded classification models: Combining models for holistic scene understanding. In *Advances in Neural Information Processing Systems (NIPS)*, pages 641–648, 2009.

[31] G. Heitz and D. Koller. Learning spatial context: Using stuff to find things. In *European Conference on Computer Vision (ECCV)*, pages 30–43. Springer, 2008.

[32] A. Hermans, G. Floros, and B. Leibe. Dense 3D semantic mapping of indoor scenes from RGB-D images. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[33] N. Höft, H. Schulz, and S. Behnke. Fast semantic segmentation of RGB-D scenes with GPU-accelerated deep neural networks. In *KI 2014: Advances in Artificial Intelligence*, pages 80–85. Springer, 2014.

[34] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2137–2144, 2006.

[35] D. Hoiem, A. A. Efros, and M. Hebert. Closing the loop in scene interpretation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008.

[36] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(5):433–449, 1999.

[37] O. Kähler and I. D. Reid. Efficient 3D scene labeling using fields of trees. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

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

[39] B. Kim, P. Kohli, and S. Savarese. 3D scene understanding by voxel-CRF. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[40] P. Kontschieder, P. Kohli, J. Shotton, and A. Criminisi. GeoF: Geodesic forests for learning coupled predictors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 65–72. IEEE, 2013.

[41] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3D point clouds for indoor scenes. In *Advances in neural information processing systems (NIPS)*, pages 244–252, 2011.

[42] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. In *Conference and Workshop on Neural Information Processing Systems (NIPS)*, 2011.

[43] P. Krähenbühl and V. Koltun. Parameter learning and convergent inference for dense random fields. In *International Conference on Machine Learning (ICML)*, 2013.

[44] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, pages 1097–1105, 2012.

[45] J. Lafferty, A. McCallum, F. Pereira, et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International conference on machine learning (ICML)*, volume 1, pages 282–289, 2001.

[46] C. Leistner, A. Saffari, J. Santner, and H. Bischof. Semi-supervised random forests. In *IEEE International Conference on Computer Vision (ICCV)*, pages 506–513. IEEE, 2009.

[47] V. Lempitsky, M. Verhoek, J. A. Noble, and A. Blake. Random forest classification for automatic delineation of myocardium in real-time 3D echocardiography. In *International Conference on Functional Imaging and Modeling of the Heart (FIMH)*, pages 447–456. Springer, 2009.

[48] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(9):1465–1479, 2006.

[49] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision (IJCV)*, 43(1):29–44, 2001.

[50] L.J. Li, R. Socher, and Li Fei-Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2036–2043, June 2009.

[51] D. Lin, S. Fidler, and R. Urtasun. Holistic scene understanding for 3D object detection with RGBD cameras. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1417–1424, 2013.

[52] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[53] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[54] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[55] R. Maree, P. Geurts, J. Piater, and L. Wehenkel. Random subwindows for robust image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 34–40. IEEE, 2005.

[56] J. McCormac, A. Handa, S. Leutenegger, and A. Davison. SceneNet RGB-D: 5M photorealistic images of synthetic indoor trajectories with ground truth. *arXiv preprint arXiv:1612.05079*, 2016.

[57] D. Meagher. Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129–147, 1982.

[58] G. Medioni, M.-S. Lee, and C.-K. Tang. *A computational framework for segmentation and grouping*. Elsevier, 2000.

[59] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht. On oblique random forests. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 453–469. Springer, 2011.

[60] A. Montillo and H. Ling. Age regression from faces using random forests. In *IEEE International Conference on Image Processing (ICIP)*, pages 2465–2468. IEEE, 2009.

[61] A. Montillo, J. Shotton, J. Winn, J. E. Iglesias, D. Metaxas, and A. Criminisi. Entangled decision forests and their application for semantic segmentation of CT images. In *Biennial International Conference on Information Processing in Medical Imaging*, pages 184–196. Springer, 2011.

[62] F. Moosmann, B. Triggs, F. Jurie, et al. Fast discriminative visual codebooks using randomized clustering forests. In *Conference and Workshop on Neural Information Processing Systems (NIPS)*, volume 2, page 4, 2006.

[63] A. C. Müller and S. Behnke. Learning depth-sensitive conditional random fields for semantic segmentation of RGB-D images. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6232–6237. IEEE, 2014.

[64] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1520–1528, 2015.

[65] S. Nowozin, C. Rother, S. Bagon, T. Sharp, B. Yao, and P. Kohli. Decision tree fields. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1668–1675. IEEE, 2011.

[66] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter. Voxel cloud connectivity segmentation - supervoxels for point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[67] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. In *European Conference on Computer Vision (ECCV)*, pages 568–580. Springer, 2006.

[68] N. Payet and S. Todorovic. (RF)^2–random forest random field. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1885–1893, 2010.

[69] P. H. O. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In *International Conference on Machine Learning (ICML)*, pages 82–90, 2014.

[70] J. Prankl, A. Aldoma, A. Svejda, and M. Vincze. RGB-D object modelling for object recognition and tracking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 96–103. IEEE, 2015.

[71] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, Oct 2007.

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

[73] R. B. Rusu and S. Cousins. 3D is here: Point cloud library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[74] I. Saleemi, L. Hartung, and M. Shah. Scene understanding by statistical modeling of motion patterns. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2069–2076. IEEE, 2010.

[75] R. Shapovalov, D. Vetrov, and P. Kohli. Spatial inference machines. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2985–2992, June 2013.

[76] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, et al. Efficient human pose estimation from single depth images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(12):2821–2840, 2013.

[77] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008.

[78] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[79] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision (ECCV)*, pages 1–15. Springer, 2006.

[80] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision (IJCV)*, 81(1):2–23, December 2007.

[81] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2011.

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

[83] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[84] J. Stückler, N. Biresev, and S. Behnke. Semantic mapping using object-class segmentation of RGB-D images. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3005–3010. IEEE, 2012.

[85] J. Stückler, B. Waldvogel, H. Schulz, and S. Behnke. Dense real-time mapping of object-class semantics from RGB-D video. *Journal of Real-Time Image Processing*, 10(4):599–609, 2015.

[86] P. Sturgess, K. Alahari, L. Ladicky, and P. H. S. Torr. Combining appearance and structure from motion features for road scene understanding. In *British Machine Vision Conference (BMVC)*. BMVA, 2009.

[87] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky. Learning hierarchical models of scenes, objects, and parts. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1331–1338, Oct 2005.

[88] C. Szegedy, L. Wei, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[89] Tombari F. Navab N. Tateno, K. Real-time and scalable incremental segmentation on dense SLAM. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[90] Tombari F. Navab N. Tateno, K. When 2.5D is not enough: Simultaneous reconstruction, segmentation and recognition on dense SLAM. *IEEE International Conference on Robotics and Automation (ICRA)*, May 2016.

[91] J. P. C. Valentin, S. Sengupta, J. Warrell, A. Shahrokni, and P. H. S. Torr. Mesh based semantic modelling for indoor and outdoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[92] H. Wang, S. Gould, and D. Koller. Discriminative learning with latent variables for cluttered indoor scene understanding. *Communications of the ACM*, 56(4):92–99, April 2013.

[93] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A.J. Davison. ElasticFusion: Dense SLAM without a pose graph.

[94] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research (IJRR)*, page 0278364916669237, 2016.

[95] D. Wolf, M. Bajones, J. Prankl, and M. Vincze. Find my mug: Efficient object search with a mobile robot using semantic segmentation. In *Workshop of the Austrian Association for Pattern Recognition (ÖAGM)*, 2014.

[96] D. Wolf, J. Prankl, and M. Vincze. Fast semantic segmentation of 3D point clouds using a dense CRF with learned parameters. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4867–4873. IEEE, 2015.

[97] D. Wolf, J. Prankl, and M. Vincze. Enhancing semantic segmentation for robotics: The power of 3-D entangled forests. *Robotics and Automation Letters, IEEE*, 1(1):49–56, 2016.

[98] J. Xiao, A. Owens, and A. Torralba. Sun3D: A database of big spaces reconstructed using sfm and object labels. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1625–1632, 2013.

[99] J. Ya and S. Geman. Context and hierarchy in a probabilistic image model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2145–2152, 2006.

[100] P. Yin, A. Criminisi, J. Winn, and I. Essa. Tree-based classifiers for bilayer video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2007.

[101] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1529–1537, 2015.

Curriculum Vitae

Personal Data

Full Name: **Daniel Matthias Wolf**
Academic Degree: Dipl.-Ing. B.Sc.
Address: Kohlgasse 47/36, 1050 Wien (Austria)
Date and place of birth: 02-07-1987, Feldkirch (Austria)
Citizenship: Austrian

Education

2013 - 2017: Doctoral studies - Technische Universität Wien (ACIN), Austria
2007 - 2012: Diploma and Bachelor's studies in Electrical Engineering and Information Technology - Technical University of Munich, Germany
2001 - 2006: Secondary Technical College for Electrical Engineering - Rankweil, Austria

Work Experience

2016: Research Internship at Apple Inc. (USA)

Teaching Experience

2013 - 2017: Machine Vision and Cognitive Robotics
Process Control Engineering
Selected Topics: Mobile Robotics and Computer Vision
Selected Topics: Robot Vision

Research Projects

2015 - 2017: Autonomous Learning of the Meaning of Objects (ALOOF)
`https://project.inria.fr/aloof`
2013 - 2015: HOBBIT - The Mutual Care Robot
`http://hobbit.acin.tuwien.ac.at`

Reviewing

International Journal of Computer Vision (IJCV)

IEEE Robotics and Automation Letters (RA-L)

IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)

Publications

Daniel Wolf, Johann Prankl and Markus Vincze. **Enhancing semantic segmentation for robotics: The power of 3-D entangled forests**. In IEEE Robotics and Automation Letters (RA-L), vol. 1., no. 1, January 2016.

Markus Vincze, Markus Bajones, Markus Suchi Daniel Wolf, Astrid Weiss, David Fischinger and Paloma da la Puente. **Learning and detecting objects with a mobile robot to assist older adults in their homes**. In European Conference on Computer Vision Workshops (ECCVW), 2016.

Markus Vincze, David Fischinger, Markus Bajones, Daniel Wolf, Markus Suchi, Lara Lammer, Astrid Weiss, Jürgen Pripfl, Tobias Körtner and Christoph Giesinger. **What Older Adults would like a robot to do in their homes - first results from a user study in the homes of users**. Proceedings of ISR 2016: 47st International Symposium on Robotics, 2016.

Daniel Wolf, Johann Prankl and Markus Vincze. **Fast semantic segmentation of 3D point clouds using a dense CRF with learned parameters**. In IEEE International Conference on Robotics and Automation (ICRA), 2015.

Daniel Wolf, Markus Bajones, Johann Prankl and Markus Vincze. **Find my mug: Efficient object search with a mobile robot using semantic segmentation**. In Annual Workshop of the Austrian Association for Pattern Recognition (ÖAGM), 2014.

Markus Bajones, Daniel Wolf, Johann Prankl and Markus Vincze. **Where to look first? Behaviour control for fetch-and-carry missions of service robots**. In Austrian Robotics Workshop (ARW), 2014.

Paloma de la Puente, Markus Bajones, Peter Einramhof, Daniel Wolf, David Fischinger and Markus Vincze. **RGB-D sensor setup for multiple tasks of home robots and experimental results**. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2014.

Martin Hofmann, Daniel Wolf and Gerhard Rigoll. **Hypergraphs for joint multi-view reconstruction and multi-object tracking**. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

Martin Hofmann, Daniel Wolf and Gerhard Rigoll. **Identification and reconstruction of complete gait cycles for person identification in crowded scenes**. International Conference on Computer Vision Theory and Applications (VISAPP), 2011.

Vienna, May 2017

Daniel Wolf