Fast Semantic Segmentation of 3D Point Clouds using a Dense CRF with Learned Parameters

Daniel Wolf, Johann Prankl and Markus Vincze

Abstract— **In this paper, we present an efficient semantic segmentation framework for indoor scenes operating on 3D point clouds. We use the results of a Random Forest Classifier to initialize the unary potentials of a densely interconnected Conditional Random Field, for which we learn the parameters for the pairwise potentials from training data. These potentials capture and model common spatial relations between class labels, which can often be observed in indoor scenes. We evaluate our approach on the popular NYU Depth datasets, for which it achieves superior results compared to the current state of the art. Exploiting parallelization and applying an efficient CRF inference method based on mean field approximation, our framework is able to process full resolution Kinect point clouds in half a second on a regular laptop, more than twice as fast as comparable methods.**

I. Introduction

Understanding the contents and the meaning of a perceived scene is one of the tasks in computer vision research, where human perception is still far ahead compared to the capabilities of artificial vision systems. However, being able to interpret and reason about the immediate surroundings is a crucial capability to enable more intelligent autonomous behavior. Semantic scene segmentation or semantic scene labeling addresses the core of this problem, namely to decompose a scene into meaningful parts and assign semantic labels to them. Especially for indoor scenes, this is a very challenging task, as they can contain a huge variety of different objects, possibly occluding each other, and are often poorly illuminated.

However, as most man-made environments, indoor scenes usually exhibit very distinctive structures and repetitive spatial relations between different classes of objects. A wall is normally enclosed by other walls, the floor and a ceiling; chairs are often placed next to tables, pictures hang on walls etc. Being able to capture, model and exploit these kinds of relations can drastically improve the quality of semantic segmentation results for indoor scenes and can be considered as the main contribution of research presented in this paper. Moreover, every step of the framework has been optimized for fast processing times, a crucial aspect to enable applications of semantic segmentation on mobile robots.

We propose a fast semantic segmentation framework, which processes 3D point clouds of indoor scenes. After oversegmenting the input point cloud in a first step, we calculate discriminative feature vectors for each segment and calculate conditional label probabilities using a Random Forest (RF) classifier. This output is then used to initialize the unary potentials of a densely interconnected Conditional Random Field (CRF), for which we learn the parameters for the pairwise potentials from training data. These potentials incorporate the likelihood of different class labels appearing close to each other, depending on different properties of the underlying 3D points such as color and surface orientation.

In a thorough evaluation on the popular NYU Depth datasets [14], [15], we show that our approach achieves superior results compared to the current state of the art.

Because we can apply an efficient CRF inference method based on mean field approximation [9] and the feature extraction and classification stage is fully parallelized, our framework is able to process a full resolution point cloud from a Microsoft Kinect in half a second, clearly outperforming comparable methods.

All authors are with the Vision4Robotics Group at the Automation & Control Institute, Technical University of Vienna, Austria. [wolf,prankl,vincze]@acin.tuwien.ac.at

II. Related Work

Particularly since the rise of cheap RGB-D sensors as the Microsoft Kinect, many different approaches have been presented to tackle the problem of semantic segmentation for indoor scenes [14], [12], [1], [16], [7], [8], [4], [6], [5], [17].

Silberman et al. presented the first large indoor RGB-D dataset [14], containing thousands of frames, from which over 2,000 have been densely annotated. The baseline algorithm they proposed trains a neural network as a classifier and then applies a CRF model which does not incorporate specific class label relations. On the same dataset, Ren et al. [12] presented impressive class accuracy scores. They use kernel descriptors to describe RGB-D patches and combine a superpixel Markov Random Field (MRF) with a segmentation tree to obtain the labeling results. However, their complex approach takes over a minute to compute for a single frame.

Valentin et al. [16] as well follow the common paradigm of a classification stage followed by a Conditional Random Field. They calculate an oversegmentation for a mesh representation of the scene and classifiy the mesh faces with a JointBoost classifier. Again, their CRF does not take individial label relations into account and only contains contrast-sensitive smoothing.

The segmentation pipeline of Hermans et al. [6] has a similar framework architecture compared to our work, since it also contains a Random Forest classifier, followed by a dense CRF model. However, their features are mostly based on 2D data, compared to our set of 3D features. Using the fast CRF inference method from [9], they achieve competitive runtimes, but they only use simple Potts potentials

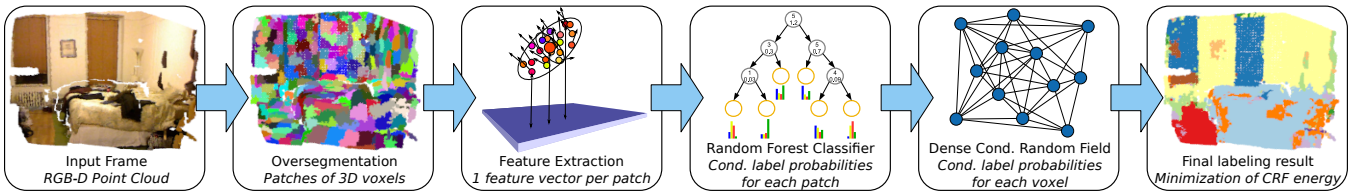

Fig. 1: Overview of our segmentation framework, which works on RGB-D point clouds. Details about the separate steps are given in section III.

in the CRF model. Also using a Random Forest classifier, Wolf et al. [17] achieve good results on the NYU datasets, but on a very reduced set of labels, which simplifies the problem. They formulate a Markov Random Field, which only performs spatial smoothing on the classification result.

Gupta et. al presented a framework based on a bottom-up segmentation method using amodal completion to group segments [4], which they extended in [5] by RGB-D object detectors to improve the feature set. Achieving state-of-the-art results on a 40-class labeling task, their method does not focus on processing time and no measurements are given in this respect.

In contrast to all of the mentioned approaches, several methods have already been proposed which try to learn and model some sort of contextual relation between class labels:

Anand et al. [1] model the labeling task in a huge graphical model, capturing spatial relations of class labels depending on different features. They show very accurate results for their dataset, however, solving the resulting mixed integer problem takes 2 minutes per frame. They claim that a relaxed formulation, with a small decrease in labeling accuracy, can be solved in 50 ms, but they do not mention the total runtime of their pipeline including pre-processing, feature extraction etc.

Combining some advantages of binary trees and CRFs for labeling problems, Kähler et al. [7] proposed a method using Decision Tree Fields (DTF) or Regression Tree Fields (RTF) for semantic scene labeling. Their framework is able to learn unary and pairwise terms of the CRF from training data, which results in comparable performance to the complex model of [1]. With inference times of $50 - 300$ ms their method is also very fast, but again timings do not include all stages of the pipeline.

Couprie et al. [2] train a large multiscale convolutional network to automatically determine features and spatial relations. Their approach is capable of processing an RGB-D frame in 0.7 s, but only if subsampled by a factor of 2 in the first place.

Similar to our work, Kim et al. [8] presented a framework with a CRF model defined on the 3D voxel level, capturing semantic and geometric relationships between classes. For a very coarse pre-defined voxel grid with $4.5\,\mathrm{cm}^3$ resolution, they achieve good results on the NYU datasets. Runtime has not been the focus of their approach and is not mentioned. However, since specific detectors for planes and objects are used to initialize the CRF, besides pre-calculated feature responses from [12], processing time cannot compete with our method.

In the following section, we describe our proposed semantic segmentation framework in detail. Section IV explains the setup used for the conducted experiments and section V discusses their results and contains a detailed runtime analysis, before we conclude in section VI.

III. Approach

A general overview on our proposed semantic segmentation framework is given in Figure 1. Our approach operates on 3D point clouds, generated by RGB-D sensors such as a Microsoft Kinect. In a first step, the raw input point cloud is transformed into a voxel representation with a defined minimum voxel size. For the created voxelized point cloud we compute an oversegmentation, such that the scene is clustered into small patches of voxels with similar appearance. In the next step, a feature vector, which captures shape and appearance properties, is extracted for each patch and then processed by a pre-trained Random Forest classifier. For each patch and label, the classifier outputs conditional label probabilities, which are used in the final step to initialize the unary potentials of a Conditional Random Field.

In the CRF model, we define pairwise smoothness potentials to locally reduce the noise of the classifier stage. Furthermore, long-range pairwise potentials incorporate the likelihood of different class labels appearing close to each other, enabling the CRF to resolve ambiguous classification results. The label compatibility terms used to calculate these potentials are completely learned from training data and capture the contextual relations between the class labels. The final labeling result is obtained by minimizing the CRF energy, which can be efficiently achieved using mean field approximation. The following sections describe the separate steps of our framework in more detail.

A. Oversegmentation

To be able to compose a robust and meaningful set of features for the classifier, we do not calculate a feature vector for every 3D point of the input point cloud, but for whole patches consisting of several points which likely belong to the same class. Assuming that the points of such a patch have a similar appearance, we calculate an oversegmentation of the input point cloud, which takes color and surface orientation into account to group adjacent points together. We make use of the oversegmentation method presented by Papon et al. [11], which is publicly available in the PointCloud Library (PCL) [13]. The method produces patches which adhere well to

object boundaries as it strictly enforces spatial connectivity in segmented regions. It works on a voxel representation of the input point cloud, its accuracy is specified by the voxel size r_v. As all subsequent steps in the framework also operate on the voxelized input cloud, this parameter defines the spatial accuracy of the whole framework. 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 result of the oversegmentation can be seen in Figure 1.

B. Feature Extraction

For each of the patches generated by the oversegmentation we calculate a feature vector $\mathbf{x}$, which captures color information as well as geometric properties of the patch. The choice of features is based on the work of Wolf et al. [17], as their feature vector is also calculated on 3D point cloud data and very efficient to compute. A list of all used features is given in Table I.

TABLE I: List of all features calculated for each 3D patch and their dimensionality. $\lambda_0 \leq \lambda_1 \leq \lambda_2$ are the eigenvalues of the scatter matrix of the patch.

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

C. Random Forest Classifier

To calculate label predictions $p(y|\mathbf{x})$ for each class label $y \in \mathcal{L} = \{l_1, \ldots, l_M\}$ and scene patch based on its feature vector $\mathbf{x}$, we use a standard Random Forest classifier [3]. 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, which in turn directly define the unary potentials in the CRF model described in section III-D.

1) Training: Since the available training data is very limited, we augment the dataset to train the RFs. Because the supervoxel clustering method is based on an Octree representation of the point cloud, it produces slightly different oversegmentation results if we mirror and rotate the training point clouds about each axis. This way, we create 10 additional oversegmentations per input point cloud to enlarge the training set.

We adapt the default training procedure for RFs used for classification, intensively discussed in [3], to our application. 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 the training sets. These weights are taken into account when the information gain is evaluated at each split node. We also recalculate the final label distributions in the leaf nodes of the trees according to the class weights. Training of each of the t trees in the forest is finished if the data points in each leaf node cannot be split up any further with a sufficient information gain, defined by a threshold h, if less than n data points are left in a node or if a specified maximum tree depth d is reached.

2) Classification: To classify a feature vector $\mathbf{x}$, it traverses through all trees in the forest—according to the learned split functions—until a leaf node is reached in each tree. The final class predictions $p(y|\mathbf{x})$ are then obtained by averaging the label distributions stored in the reached leaf nodes during training. After the classifier has been evaluated for the input scene, the intermediate result is a coarse label prediction on the patch level, suffering from classification noise, since the classifier only takes local information from single patches into account. The next section describes how we use a Conditional Random Field to smooth and refine the result on the finer voxel level and exploit learned contextual class relations to resolve ambiguous label predictions.

D. Dense Conditional Random Field

A Conditional Random Field can improve the labeling by the introduction of pairwise smoothness terms, which maximize the label agreement of similar voxels in the scene. Additionally, more elaborate pairwise potentials can be defined, such that contextual relations between different class labels can be modeled in order to further refine the classification results.

A CRF is defined over a set $\mathcal{X} = \{X_1, \ldots, X_N\}$ of random variables. The variables are conditioned on the model parameters $\boldsymbol{\theta}$ and, in our particular case, the voxelized input point cloud $\mathbf{P}$. Thus, each variable X_i corresponds to a voxel $\mathbf{v}_i \in \mathbf{P}$ and is assigned a label $y_i \in \mathcal{L}$. Furthermore, it is associated with a feature vector $\mathbf{f}_i$, determined by $\mathbf{P}$. Note that this is a different feature vector than the one defined and used for classification in sections III-B and III-C!

A complete label assignment $\mathbf{y} \in \mathcal{L}^N$ then has a corresponding Gibbs energy, composed by unary and pairwise potentials ψ_i and ψ_{ij}:

$$E(\mathbf{y}|\mathbf{P}, \boldsymbol{\theta}) = \sum_i \psi_i(y_i|\mathbf{P}, \boldsymbol{\theta}) + \sum_{i<j} \psi_{ij}(y_i, y_j|\mathbf{P}, \boldsymbol{\theta}) \quad (1)$$

with $1 \leq i, j \leq N$. The optimal label assignment $\mathbf{y}^*$ then corresponds to the configuration which minimizes the energy function:

$$\mathbf{y}^* = \underset{\mathbf{y} \in \mathcal{L}^N}{\arg\min}\, E(\mathbf{y}|\mathbf{P}, \boldsymbol{\theta}). \quad (2)$$

We define the unary potential as the negative log-likelihood of the label distribution output by the classifier:

$$\psi_i(y_i|\mathbf{P}, \boldsymbol{\theta}) = -\log(p(y_i|\mathbf{x_i})), \quad (3)$$

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

$$\psi_{ij}\left(y_i, y_j|\mathbf{P}, \boldsymbol{\theta}\right) = \sum_m \mu^{(m)}\left(y_i, y_j|\boldsymbol{\theta}\right) k^{(m)}\left(\mathbf{f}_i, \mathbf{f}_j\right), \quad (4)$$

where $\mu^{(m)}$ is a label compatibility function which 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 the kernel functions $k^{(m)}$ to Gaussian kernels:

$$k^{(m)}\left(\mathbf{f}_i, \mathbf{f}_j\right) = w^{(m)} \exp\left(-\frac{1}{2}\left(\mathbf{f}_i - \mathbf{f}_j\right)^T \Lambda^{(m)}\left(\mathbf{f}_i - \mathbf{f}_j\right)\right), \quad (5)$$

where $w^{(m)}$ are linear combination weights and $\Lambda^{(m)}$ 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. [6]. The first one is a smoothness kernel, which is only active in the local neighborhood of each voxel and reduces the classification noise by favoring the assignment of the same label to two close voxels with a similar surface orientation:

$$k^{(1)} = w^{(1)} \exp\left(-\frac{|\mathbf{p}_i - \mathbf{p}_j|}{2\theta_{p,s}^2} - \frac{|\mathbf{n}_i - \mathbf{n}_j|}{2\theta_n^2}\right), \quad (6)$$

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

$$k^{(2)} = w^{(2)} \exp\left(-\frac{|\mathbf{p}_i - \mathbf{p}_j|}{2\theta_{p,l}^2} - \frac{|\mathbf{c}_i - \mathbf{c}_j|}{2\theta_c^2}\right), \quad (7)$$

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

In contrast to [6], we define separate label compatibility functions $\mu^{(m)}$ for both kernels. For the smoothness kernel we use a simple Potts model: $\mu^{(1)}(y_i, y_j|\boldsymbol{\theta}) = 1_{[y_i \neq y_j]}$. For the appearance kernel, however, we use a more expressive model, since it should capture contextual relations between different classes across larger distances. Consequently, $\mu^{(2)}$ is a full, symmetric $M \times M$ matrix, where all class relations are defined indvidually. But instead of manually assigning the compatibility values, we automatically learn them from training data. More details about the parameter learning for our CRF model are given in section III-D.2.

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 [9], based on mean field approximation. Their method is able to cope with a large number of variables and allows all pairs of variables to be connected by pairwise potentials ("dense" CRF). For our application this has two key advantages: First, it enables us to define the CRF on the finer voxel level instead of the patch level while maintaining fast inference times. Consequently, the CRF improves the labeling on a finer scale and across patch 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 contextual relations between different classes, helping to resolve ambiguities in the labeling result.

2) Parameter Learning: To be able to fully exploit the capabilities of the complex CRF model, its numerous parameters have to be well defined. Since they are often depending on each other, this is a difficult task. Recently, Krähenbühl and Koltun extended their CRF model with a learning framework, based on the optimization of a marginal-based loss function [10]. Their approach captures the parameter dependencies by jointly estimating all parameters from training data. Adapting their framework to our application, we are able to 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 section V 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 manually defined parameters.

IV. Experimental Setup

We conducted all of our scene labeling experiments on the popular NYU Depth datasets introduced by Silberman et al. [14], [15]. Both datasets contain thousands of RGB-D frames from indoor scenes, recorded with a Microsoft Kinect sensor. In version 1 2,284 and in version 2 1,449 of the frames have been densely labeled. For both versions, postprocessed data, where missing depth values have been automatically filled in using an inpainting algorithm, is provided as well as the original sensor data. We train and evaluate our framework on the original data to be independent of any postprocessing algorithms.

For both datasets, we create 5 splits into training, validation and test sets, where we use 60% for the training, 20% for the validation and the remaining 20% for the test set. The RF classifier is then trained on the training set and the CRF parameters are learned on the validation set. In all experiments, we use the same parameter settings in the whole framework:

For the oversegmentation, we set the voxel size r_v to 1.5 cm and the maximum cluster size r_c to 30 cm. These settings are a suitable trade-off between speed and accuracy, as the patches can capture enough information while smaller objects can still be accurately segmented. In 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.

V. Results and Discussion

For the quantitative evaluation, we compare our framework to other approaches using the common multi-label segmentation metrics of *global accuracy* and *class average accuracy*. The global accuracy is defined as the overall point-wise accuracy over the whole test set, whereas the class average acuracy is calculated as the mean of the main diagonal of the confusion matrix. All of the given numbers are averaged values over all 5 splits of the cross-validation.

We compare our work to the methods of [14], [12], [15] and [2], as well as to the recently presented approach of [6], which is the most similar and achieves not only competitive results on the NYU Depth datasets, but also shows fast processing times.

A. NYU Depth Dataset v1

Quantitative evaluation results for version 1 of the NYU Depth dataset are shown in Table II. First, we trained our framework to distinguish between 12 semantic classes defined in [14] and directly compare our results to [6]. We can observe that our RF outperforms their RF implementation by more than 12% regarding class average accuracy as well as global accuracy. Apparently, it is beneficial to calculate the feature vector on patches of 3D data including features such as the surface normal angles instead of using pixel-wise features. Enabling the CRF in our framework, without learned label compatibility terms, boosts the class average accuracy to 71.9% and the global accuracy to 85.5%. If we apply the complex CRF model with the fully learned label compatibility matrix, class average accuracy is similar to the RF performance, but the global accuracy further increases to 87.8%, outperforming the current state of the art in [6] by a 16%-margin.

To be able to compare our framework also with other approaches, we add a separate *background* class for a second series of experiments on the v1 dataset. This class contains all of the $1{,}418$ available labels which could not be mapped to the first 12 classes. Obviously, this setup comes with a drastically reduced overall labeling performance, since the background class is mixing hundreds of class labels, whose different properties can hardly be captured and distinguished by the feature vector and the classifier. In this configuration, we cannot compete with the method of [12], but their approach, combining a superpixel MRF with kernel descriptors and a segmentation tree, takes over a minute per frame to compute. Again, we achieve better results than [6] with both of our CRF models, the simpler manual model performing well with regards to class average accuracy, and the complex learned model achieving a 23% improvement in global accuracy.

B. NYU Depth Dataset v2

For the second version of the NYU Depth dataset, we also conducted two series of experiments. The quantitative results are presented in Table III and Table IV. The first evaluation contains 13 different semantic labels, specified by Couprie et al. [2]. The label set is slightly different from the set used for version 1, e.g. it contains a particular *object* class. Besides [6], which to our knowledge represents the current state of the art for this dataset, we also compare our results to the method presented in [2], where a multi-scale convolutional network is trained to perform segmentation.

TABLE IV: Class and global Accuracy scores for NYU v2 on the small set of structural classes defined by [15].

Method	Ground	Struct	Furniture	Props	Class Avg.	Global
Silberman [15]	68	59	70	42	59.6	58.6
Couprie [2]	87.3	**86.1**	45.3	35.5	64.5	63.5
Hermans [6]	97.4	76.1	61.8	40.9	69.0	68.1
Ours (RF only)	97.4	73.6	65.6	**51.1**	71.9	72.0
Ours (manual)	**97.7**	78.7	67.7	46.0	72.5	73.7
Ours (learned)	96.8	77.0	**70.8**	45.7	**72.6**	**74.1**

For 11 out of the 13 class categories, our framework achieves the best results. Both the manually defined and the learned CRF model clearly outperform both of the compared methods, the former achieving 56.9% class average and 63.2% global accuracy, the latter with 55.5% class average respectively 64.9% global accuracy. Thus, we improve the current state of the art on this dataset by more than 8 respectively 10% regarding class average and global accuracy. Some qualitative example results are given in Figure 3.

The second label set on which we evaluated our framework for this dataset is defined by [15] and only contains 4 rather structural class labels *ground*, *structure*, *furniture*, and *props*. Again, our approach achieves superior results compared to other methods, where the learned CRF model performs slightly better in class average accuracy (72.6%) as well as global accuracy (74.1%) than the manually defined model with no individual label compatibility terms.

C. Learned Label Compatibilities

To show the effectiveness of the learning approach used in our framework, Figure 2 depicts an example result of a

TABLE II: Class and global accuracy scores for NYU v1. The upper half shows results for the label set containing 12 classes defined in [14], the results in the lower half have been achieved after adding a separate background class.

Method	Bed	Blind	Bookshelf	Cabinet	Ceiling	Floor	Picture	Sofa	Table	TV	Wall	Window	Background	Class Avg.	Global
Hermans [6] (RF only)	51.5	41.6	48.5	54.1	88.3	87.2	62.1	50.0	40.0	73.4	69.6	18.9	-	57.1	65.0
Hermans [6]	57.6	57.3	67.5	**58.2**	**92.7**	88.5	56.6	66.7	45.7	82.0	77.6	17.2	-	64.0	71.5
Ours (RF only)	62.5	63.7	75.9	39.9	91.5	97.2	**65.2**	71.8	71.5	86.7	76.2	34.3	-	69.7	79.6
Ours (manual)	**66.1**	**64.4**	87.6	40.0	90.8	**97.9**	59.9	**73.6**	75.0	**88.8**	83.0	**35.4**	-	**71.9**	85.5
Ours (learned)	52.8	62.6	**93.9**	38.7	89.4	97.5	58.3	71.5	**75.9**	84.8	**86.2**	28.9	-	70.1	**87.8**
Silberman [14]	-	-	-	-	-	-	-	-	-	-	-	-	-	56.6	-
Hermans [6] (full)	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 [12]	**85**	**80**	89	**66**	**93**	93	**82**	**81**	60	86	**82**	**59**	35	**76.1**	-
Ours (RF)	67.3	63.2	80.4	39.6	87.7	96.0	67.1	67.7	**68.4**	85.3	75.5	29.6	27.5	65.8	58.0
Ours (manual)	70.7	65.9	**91.9**	40.6	87.3	**96.8**	61.9	70.9	72.5	**87.0**	81.4	29.7	30.1	68.2	62.5
Ours (learned)	53.5	57.9	86.4	34.0	85.7	95.7	61.1	64.4	60.1	81.9	79.7	18.1	**48.3**	63.6	**67.8**

TABLE III: Class and global accuracy scores for NYU v2, using 13 different semantic classes defined in [2].

Method	Bed	Object	Chair	Furniture	Ceiling	Floor	Wall deco.	Sofa	Table	Wall	Window	Bookshelf	TV	Class Avg.	Global
Couprie [2]	38.1	8.7	34.1	42.4	62.6	87.3	40.4	24.6	10.2	86.1	15.9	13.7	6.0	36.2	52.4
Hermans [6]	**68.4**	8.6	41.9	37.1	83.4	91.5	35.8	28.5	27.7	71.8	46.1	45.4	**38.4**	48.0	54.2
Ours (RF only)	49.8	24.4	55.6	41.4	92.5	96.8	**43.6**	54.2	47.3	58.6	44.2	43.9	31.9	52.6	58.0
Ours (manual)	57.1	24.3	**63.0**	47.8	**93.3**	**97.5**	42.7	**64.7**	50.2	68.5	**46.3**	**52.6**	31.2	**56.9**	63.2
Ours (learned)	58.2	**37.4**	54.7	**57.3**	92.8	**97.5**	32.3	49.8	**51.8**	**74.4**	43.2	45.3	26.4	55.5	**64.9**

learned label compatibility matrix for the second version of the dataset. Besides the main diagonal of the matrix, it can be seen that various other strong relations between different class labels are identified by the algorithm, the most noticeable entries being wall/wall-deco, wall/object and bed/object.

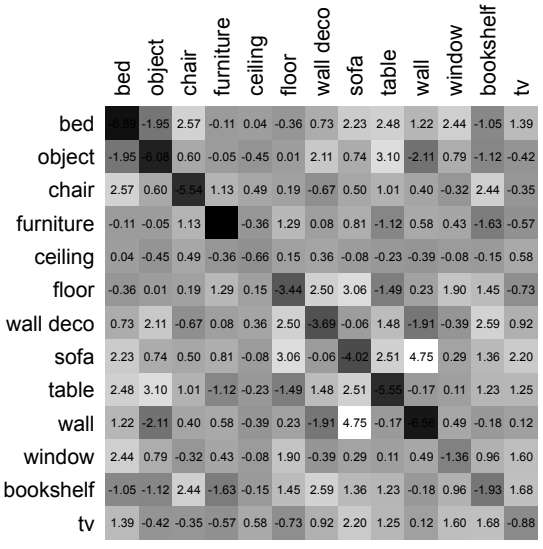

Fig. 2: Label compatibility terms learned for version 2 of the NYU dataset. The darker the entry in the matrix (or the smaller the value), the more likely the two corresponding labels occur close to each other.

D. Runtime Analysis

In Table V, we give an overview on the approximate runtimes of the different components of our framework. The numbers are based on the experiments conducted on our test machine, an i7 laptop with 8 cores clocked with 2.4 GHz. 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 [6], our framework is more than twice as fast.

TABLE V: Approximate runtimes of the separate stages of our framework on our test machine (i7 laptop, 8×2.4 GHz), using the experimental settings described in section IV. On average, 2 Kinect point clouds per second can be processed.

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

VI. Conclusions and Outlook

We introduced a fast semantic segmentation framework for 3D point clouds, which combines a Random Forest classifier with a highly efficent inference method for a dense Conditional Random Field. Furthermore, it incorporates a

Fig. 3: Example results for version 2 of the NYU Depth dataset. Top to bottom row: Input point cloud, groundtruth, results after RF, results after full CRF. Labels which are not part of the label set are not shown in the groundtruth. Notice the inconsistent and therefore missing groundtruth labeling, e.g. for the pole in the second column or the bookshelf in the fourth column), where our method assigns correct labels.

learning method to jointly estimate all CRF parameters from training data, enabling the model to capture and exploit the strong contextual relations between different class labels, often exhibited especially in indoor scenes. Our method achieves state-of-the-art results for two challenging datasets while being more than twice as fast as comparable methods. This performance paves the way to achieve complex higher level scene understanding and reasoning, crucial for intelligent autonomous systems.

For future work, we plan to further exploit the learning capabilities of the dense CRF. Besides the class relations, we intend to simultaneously estimate all kernel parameters of the model as well, leading to a purely data driven semantic segmentation system.

VII. Acknowledgement

The research leading to these results has received funding from the European Community, Seventh Framework Programme (FP7/2007-2013), under grant agreement No. 288146, HOBBIT, and the Austrian Science Foundation (FWF) under grant agreement No. I1856-N30, ALOOF.

References

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

[2] C. Couprie, C. Farabet, L. Najman, and Y. LeCun. Indoor Semantic Segmentation using depth information. In *ICLR*, 2013.

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

[4] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images . In *CVPR*, 2013.

[5] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning Rich Features from RGB-D Images for Object Detection and Segmentation. In *ECCV*, 2014.

[6] A. Hermans, G. Floros, and B. Leibe. Dense 3D Semantic Mapping of Indoor Scenes from RGB-D Images. In *ICRA*, 2014.

[7] O. Kähler and I. D. Reid. Efficient 3D Scene Labeling Using Fields of Trees. In *ICCV*, 2013.

[8] B. Kim, P. Kohli, and S. Savarese. 3D Scene Understanding by Voxel-CRF. In *ICCV*, 2013.

[9] P. Krähenbühl and V. Koltun. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In *NIPS*, 2011.

[10] P. Krähenbühl and V. Koltun. Parameter Learning and Convergent Inference for Dense Random Fields. In *ICML*, 2013.

[11] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter. Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds. In *CVPR*, 2013.

[12] X. Ren, L. Bo, and D. Fox. RGB-(D) scene labeling: Features and algorithms. In *CVPR*, 2012.

[13] R. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *ICRA*, 2011.

[14] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *ICCVW*, 2011.

[15] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012.

[16] J. P. Valentin, S. Sengupta, J. Warrell, A. Shahrokni, and P. H. Torr. Mesh Based Semantic Modelling for Indoor and Outdoor Scenes. In *CVPR*, 2013.

[17] D. Wolf, M. Bajones, J. Prankl, and M. Vincze. Find my mug: Efficient object search with a mobile robot using semantic segmentation. In *ÖAGM*, 2014.