Efficient 3D Scene Labeling Using Fields of Trees

Olaf Kähler
Dept. of Engineering Science
University of Oxford
olaf@robots.ox.ac.uk

Ian Reid
School of Computer Science
University of Adelaide
ian.reid@adelaide.edu.au

Abstract

We address the problem of 3D scene labeling in a structured learning framework. Unlike previous work which uses structured Support Vector Machines, we employ the recently described Decision Tree Field and Regression Tree Field frameworks, which learn the unary and binary terms of a Conditional Random Field from training data. We show this has significant advantages in terms of inference speed, while maintaining similar accuracy. We also demonstrate empirically the importance for overall labeling accuracy of features that make use of prior knowledge about the coarse scene layout such as the location of the ground plane. We show how this coarse layout can be estimated by our framework automatically, and that this information can be used to bootstrap improved accuracy in the detailed labeling.

1. Introduction

Interacting with the world requires both an understanding of the 3D geometry of a scene and of its semantic meaning. Remarkable achievements have been made in both of these directions, with systems like PTAM [8], DTAM [9] and KinectFusion [10] able to provide 3D information in real time. Learning and inference models such as Support Vector Machines and Random Forests [3] have been used in combination with Conditional Random Fields [2, 11] to infer semantic labels in a range of problems. However only recently has work focused on the combination of 3D data and semantic labeling. For a successful integration of the two parts, the 3D representation has to be rich enough to provide an actual benefit for the task of scene labeling, and the scene labeling system has to be fast enough to allow interaction with the inferred labels.

We contribute in four different ways to the combined treatment of reconstruction and interpretation of scenes. First, we present a framework to employ Decision Tree Fields [11] and Regression Tree Fields [6] for the 3D scene labeling task, where the dependency structure is dynamically determined from the scene instead using grid structures as in the original works. Second, we show that both of these classifiers are very competitive or even outperform state-of-the-art methods with the additional benefit of significantly faster inference steps. Third, we compare the two classifiers in the context of scene labeling. And finally we use an adaptation of the framework to estimate the coarse scene layout and ground plane, which are a prerequisite for high level features commonly used for 3D scene labeling.

1.1. Related Work

With the introduction of the Kinect sensor a significant number of works has been published on semantic labeling of RGB images with additional depth information [15, 13]. While these works achieve impressive results, they focus on semantic segmentation of single images merely using depth as an additional information source. They do not consider mechanisms for consistent labels in sequences of images from cameras progressively exploring an environment.

A similar segmentation task was investigated in [12], where multiple segmentations are computed independently for each image in a sequence. However, a significant improvement of the performance is observed by enforcing temporal consistency, which the authors achieve with a Markov Random Field that links segments in different images if they occupy the same location in 3D space.

The closest work to ours is presented in [2]. The authors first integrate the 3D information from multiple images into a single 3D point cloud and then oversegment and label this point cloud using full 3D geometry information. The downside of their SVM-based approach is the processing time, which is far from real-time for their full scale classification model and an accelerated, approximate solution only comes at the expense of reduced labeling performance.

We pose the problem instead within the framework of the recently introduced Decision Tree Fields [11] and Regression Tree Fields [6]. We show that this enables us to discover consistent scene labels for a full 3D model at interactive rates while still maintaining the high precision and recall of the current state-of-the-art methods.

Like many scene labeling systems, the performance of

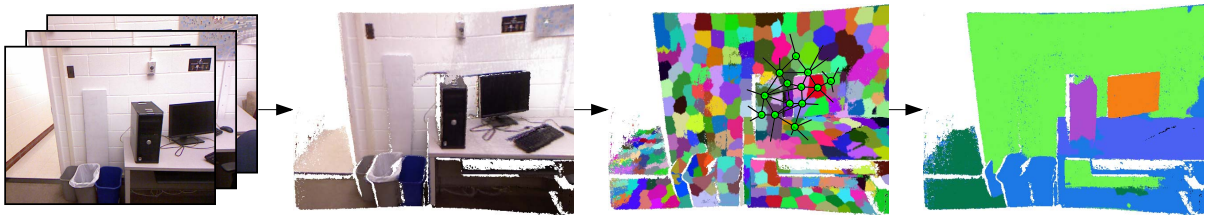

Figure 1. Outline of the pipeline for semantic labeling. Starting with a dense 3D reconstruction we compute an oversegmentation and construct a CRF before we finally arrive at the inferred labels for the scene.

our framework crucially relies on knowledge of the coarse scene layout comprising of a ground plane and the walls, which allows us to exploit rich geometric constraints that place other objects within the context of the overall scene. In the experimental section we empirically quantify the benefit of prior knowledge of these coarse features. However previous closely related works either pre-calibrate [2] and feed the information to the algorithm as prior knowledge, or it is estimated in a separate ad-hoc step [15]. A separate stream of work has explicitly considered inferring the coarse scene layout, usually by restricting it to some simple form such as a cuboid [16] or an indoor Manhattan world [5]. In contrast we show that the very same method we use for scene labeling can also be adapted to label the floor and walls in indoor environments, which then allows to bootstrap a fine grained labeling of the scene.

1.2. Method Overview

The main steps of our proposed system are illustrated in Figure 1. Starting from given depth images, we compute a dense, volumetric representation of the scene using our own implementation of the KinectFusion algorithm [10] that in addition takes the RGB information into account in aligning the RGB-D images. We then compute an oversegmentation of this dense representation and instantiate a graph over the segments as explained in Section 2. For each of the segments and for each neighborhood relation between the segments we then extract a feature vector which we will detail in Section 3. Finally we compute labels for the segments using adaptations of Decision Tree Fields [11] and Regression Tree Fields [6]. Section 4 will go into the details and revise the learning and inference steps. An experimental evaluation of our approach follows in Section 5 and we discuss it in some concluding remarks in Section 6.

2. Oversegmentation

Given the 3D input volumes from KinectFusion we first want to reduce the complexity from the millions of points on the 3D surface to a few thousand, and we want to pre-group points that are likely to be part of the same object. We assume that the boundaries between objects are represented by discontinuities in appearance, depth or surface orientation. We achieve the desired pre-grouping by computing an oversegmentation.

Inspired by the very successful SLIC superpixels [1] for 2D images, we develop a method for the oversegmentation of a 3D volume. We start by distributing seeds points C_i in a regular grid over the volume. Each of the seeds is defined by a centroid $\mathbf{p}_{C_i}$, a color $\mathbf{c}_{C_i}$ and a normal $\mathbf{n}_{C_i}$. Next a label $l_{\mathbf{x}}$ is assigned to each surface point $\mathbf{x}$ by finding the seed C_i in a neighborhood d, that minimizes the distance

$$D(\mathbf{x}, C_i) = w_p \|\mathbf{x} - \mathbf{p}_{C_i}\| + w_c \|\mathbf{c}_{\mathbf{x}} - \mathbf{c}_{C_i}\| + w_n \arccos \mathbf{n}_{\mathbf{x}}^T \mathbf{n}_{C_i},$$

where w_p, w_c and w_n weight the individual contributions of distances in space, color and normal. Once each point has a label, the centroids $\mathbf{p}_{C_i}$, mean colors $\mathbf{c}_{C_i}$ and average normals $\mathbf{n}_{C_i}$ of each seed C_i are recomputed from the points $\mathbf{x}$ that have been assigned the label $l_{\mathbf{x}} = C_i$. These two steps are iterated until convergence to a stable oversegmentation of the scene. As this approach does not enforce connectivity of the segments a post-processing step is applied after the iterations finish [1], in which individual stray points or small connected parts of the scene are merged with their most closely matching neighboring segment.

Given the oversegmentation we define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ over the segments. Each segment i is used as a node $v_i \in \mathcal{V}$ and for each pair of segments i, j with the distance between their centroids satisfying $\|\mathbf{p}_{C_i} - \mathbf{p}_{C_j}\| < d_{\text{context}}$ we add an edge $e_{i,j} \in \mathcal{E}$, where d_{context} is chosen such that it covers about 20% of the overall extent of the scene.

3. Feature Extraction

For each node v_i we extract a feature vector describing the segment appearance and its shape characteristics. Similarly, we extract a feature vector for each pair of segments that have an edge $e_{i,j}$ linking them, which will describe the contextual relation of the two segments. These descriptors primarily rely on the color and geometry information accumulated in the 3D reconstruction process and on the original input images that were used in this process. If no prior knowledge on the coarse layout of the scene is given, distances and the overall scale can only be measured in terms of voxels and the arrangement of segments within the scene context is unknown. An upgrade to metric distances and an

Average HSV values	3
Histogram of HSV values (6/2/2 bins)	10
HOG descriptor for projection	31
Eigenvalues of the Scatter Matrix	3
Linearity and Planarity ($\lambda_1 - \lambda_2$, $\lambda_2 - \lambda_3$)	2
Histogram of angles of the normals (8 bins)	8
Plane fit errors (mean, median and maximum)	3
Spin Image (4×4 bins)	16
Angle with ground plane	1
Height above ground plane	1
Footprint area	1
Total	**79**

Absolute difference of average HSV values	3
Absolute difference of HOG descriptors	31
Angle between mean normals	1
Distance between centroids	1
Convexity	2
Coplanarity (mean, median and max distances)	6
Connectivity	1
Vertical and horizontal displacement	2
Footprint overlap	1
Total	**48**

Table 1. List and dimensionality of the used features. The upper set of features is used for the unary terms and the lower one for the binary relations. The three subgroups are extracted from color, shape and global scene information, respectively.

additional set of very powerful features, such as the height above the ground plane, can be extracted once this prior knowledge is given. The list of the features we use in our current setup is given in Table 1. While most of them are self-explanatory and have been used before [2, 15] more details on some of them are given in the following.

As a typical appearance based feature we compute a HOG descriptor [4] for each segment. We follow the approach in [2] and go through the original input image sequence to find the image where the camera is looking most closely to orthogonal onto the segment. We then project the centroid of the segment into this image and compute a standard HOG descriptor for the respective cell, normalized using the L2-norm over four different neighborhoods of 2×2 cells each. We use 9 bins for a histogram of unsigned gradient orientations, 18 bins for signed orientations and 4 general texturedness features, which are simply the sums of all normalized histogram entries.

We also have 3D geometry information and can hence compute features describing the shape of the segments. Apart from the three sorted Eigenvalues $(\lambda_1, \lambda_2, \lambda_3)$ of the Scatter Matrix we also use $\lambda_1 - \lambda_2$ as a measure for the linearity of a segment and $\lambda_2 - \lambda_3$ as a measure for the planarity [2]. A measure of the flatness, that to our knowledge has not been used before, is given by a histogram of angles between the average normal of a segment and the surface

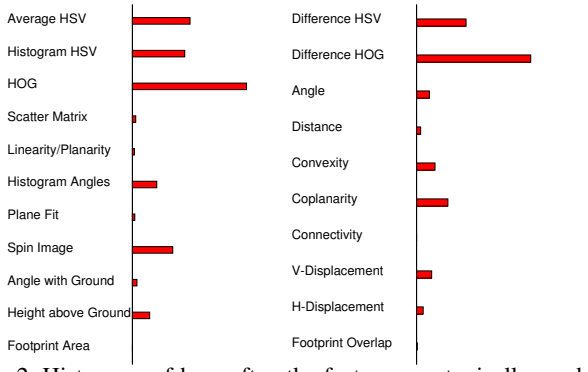

Figure 2. Histogram of how often the features are typically used for splitting decision trees in the labeling phase. The left and right plots cover the unary and binary features, respectively. This gives a rough estimate of the relevance of features, but it does not take the depth within the trees into account where the features are used.

normals at individual points. Finally we compute the mean, median and maximum plane fit errors as in [15] and spin images [7], that have not been used in either of [2, 15].

To describe the relation between two segments, a first obvious feature is the angle between the surface normals and the distance between the two centroids. Given the centroids and surface normals we can also check whether the centroid of one segment is in front of the other segment and vice versa, which provides an indication of convexity as used in [2]. We also compute the mean, median and maximum distances of the points of one segment from a plane fit to the other segment, which gives a measure of coplanarity of the two [15], and we check whether two segments are connected as neighbors in the oversegmentation.

A final set of descriptors is based on higher level knowledge of a ground plane. In our experiments in Section 5.3 these features significantly improve the performance of the overall system. While in [2] this global contextual information was given a priori, we show in Section 5.4 that it can be estimated automatically from the data using the very same system that we propose for fine grained scene labeling. Once given, this knowledge allows us to compute the verticality of a segment, i.e. the angle of the surface normal with the ground plane, the height above ground and the horizontal and vertical displacements between two segments, all of which have also been used in [2]. We also compute a projection of each segment onto the ground plane and use the area of this footprint and the percentage of overlap of two footprints as additional features [15].

Figure 2 gives an assessment of the relevance of individual features. Appearance based features like HOG dominate particularly for the unary terms. The histogram of normal angles and spin images appear to be the most important descriptors of shape, and convexity and coplanarity are highly relevant to describe the relation of segments. A detailed

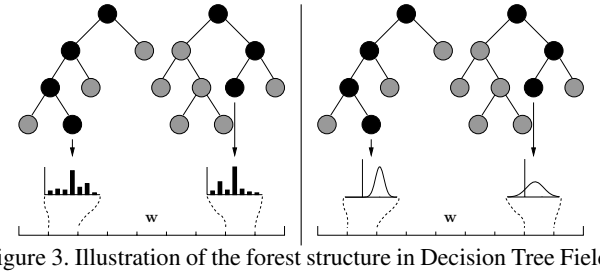

Figure 3. Illustration of the forest structure in Decision Tree Fields (left), where tables of costs are selected according to the input data, and Regression Tree Fields (right), where multi-dimensional quadratic cost functions or Gaussian likelihoods are selected.

evaluation of the features incorporating knowledge of the scene layout follows in Section 5.3, but it is already apparent that they contribute significantly.

4. Semantic Labeling

We model the relation between the collection of feature vectors $\mathbf{x}$ extracted for the nodes and edges and the label vector $\mathbf{y}$ for the scene as a Conditional Random Field. In our current setup we only consider unary terms E_N and binary terms E_E, leading to the overall relation:

$$P(\mathbf{y}|\mathbf{x},\mathbf{w}) = \frac{\exp(-E(\mathbf{y},\mathbf{x},\mathbf{w}))}{\int \exp(-E(\mathbf{y},\mathbf{x},\mathbf{w}))\mathrm{d}\mathbf{y}} \tag{1}$$

$$E(\mathbf{y},\mathbf{x},\mathbf{w}) = \sum_{i\in\mathcal{V}} E_N(\mathbf{y}_i,\mathbf{x},\mathbf{w}) + \sum_{i,j\in\mathcal{N}} E_E(\mathbf{y}_i,\mathbf{y}_j,\mathbf{x},\mathbf{w}) \tag{2}$$

with a parameter vector $\mathbf{w}$. Given that we have multiple classes and the labels for individual nodes in the graph are related to each other, this is a classical problem for structured learning and inference techniques. We address it using the two closely related Decision Tree Fields (DTFs) [11] and Regression Tree Fields (RTFs) [6].

For both the DTF and RTF formulations the energies E_N and E_E are determined using structures akin to Random Forests [3]. As illustrated in Figure 3, the input data $\mathbf{x}$ is passed down a set of trees and eventually selects a single leaf from each tree. The leaves then determine the energies required to assign the individual labels $\mathbf{y}_i$ to nodes v_i or pairs of labels $\mathbf{y}_i$ and $\mathbf{y}_j$ to nodes v_i and v_j. The definition of these energies are slightly different for DTFs and RTFs.

For DTFs the labels are discrete $y_i \in [1,\ldots,K]$, where K is the number of distinct classes, and the parameter vector $\mathbf{w}$ stores tables of energy values for each leaf. The label y_i selects a single entry from the table selected by the input data $\mathbf{x}$, and this entry represents the energy for assigning the label. As there are multiple trees in the forest and hence multiple leaves, the overall energies E_E and E_N are defined as the sums over the energies in the individual leaves:

$$E_N^{\text{DTF}}(y_i,\mathbf{x},\mathbf{w}) = \sum_{q\in\mathrm{L}(i,\mathbf{x})} w_{q,y_i} \tag{3}$$

$$E_E^{\text{DTF}}(y_i,y_j,\mathbf{x},\mathbf{w}) = \sum_{q\in\mathrm{L}(i,j,\mathbf{x})} w_{q,y_i,y_j}, \tag{4}$$

where the functions $\mathrm{L}(i,\mathbf{x})$ and $\mathrm{L}(i,j,\mathbf{x})$ return the set of leaves reached in the respective forests and w_{q,y_i} and w_{q,y_i,y_j} are the individual energy values.

For RTFs the labels are vectors $\mathbf{y}_i \in \mathbb{R}^K$. Each entry of $\mathbf{y}_i$ encodes the confidence that the node v_i should be labeled as the corresponding class. For the unary terms the parameter vector $\mathbf{w}$ stores a symmetric, positive definite matrix $\boldsymbol{\Theta}_{u,q} \in \mathbb{S}^K$ and a vector $\boldsymbol{\theta}_{u,q} \in \mathbb{R}^K$ for each leaf q, and the energy required to assign a label $\mathbf{y}_i$ to node v_i is determined by the quadratic energy function defined by $\boldsymbol{\Theta}_{u,q}$ and $\boldsymbol{\theta}_{u,q}$. For the binary terms, the quadratic energy functions stored in the leaves are $2K$ dimensional and defined by $\boldsymbol{\Theta}_{b,q} \in \mathbb{S}^{2K}$ and $\boldsymbol{\theta}_{b,q} \in \mathbb{R}^{2K}$. They determine the energy required for the concatenated label vector $\mathbf{y}_{i,j} = (\mathbf{y}_i^T, \mathbf{y}_j^T)^T$. The overall energy terms for the ensemble of trees in the forest are again sums over the individual contributions, resulting in:

$$E_N^{\text{RTF}}(\mathbf{y}_i,\mathbf{x},\mathbf{w}) = \sum_{q\in\mathrm{L}(i,\mathbf{x})} \left(\frac{1}{2}\mathbf{y}_i^T\boldsymbol{\Theta}_{u,q}\mathbf{y} - \boldsymbol{\vartheta}_{u,q}^T\mathbf{y}_i\right) \tag{5}$$

$$E_E^{\text{RTF}}(\mathbf{y}_i,\mathbf{y}_j,\mathbf{x},\mathbf{w}) = \sum_{q\in\mathrm{L}(i,j,\mathbf{x})} \left(\frac{1}{2}\mathbf{y}_{i,j}^T\boldsymbol{\Theta}_{b,q}\mathbf{y}_{i,j} - \boldsymbol{\vartheta}_{b,q}^T\mathbf{y}_{i,j}\right) \tag{6}$$

These quadratic energy functions can also be interpreted as Gaussians with covariance matrices $\boldsymbol{\Sigma}_{\cdot,q} = \boldsymbol{\Theta}_{\cdot,q}^{-1}$ and mean vectors $\boldsymbol{\mu}_{\cdot,q} = \boldsymbol{\Theta}_{\cdot,q}^{-1}\boldsymbol{\theta}_{\cdot,q}$. In that sense the leaves in the unary regression forests store K-dimensional Gaussian distributions over the label vectors $\mathbf{y}_i$, and the binary forests store $2K$-dimensional distributions over the concatenations of $\mathbf{y}_i$ and $\mathbf{y}_j$. Accordingly the combined energy E from equation (2) is a $K|\mathcal{V}|$-dimensional Gaussian.

4.1. Learning

In the learning phase the features $\mathbf{x}$ with corresponding labels $\mathbf{y}$ are given as training examples, and we have to find the model parameters $\mathbf{w}$. As in [11] we determine the tree structures in a first step and then optimize the parameters $\mathbf{w}$ in a separate, second step. A two step approach is necessary as the parameters $\mathbf{w}$ are continuous whereas the tree structures form a large, combinatorial space, and a simultaneous optimization of both is intractable.

In the first stage the tree structures are determined using standard methods [3]. We pick a random subset of the training data to train each tree in the forest, the binary decision rules at the internal nodes select a random element of

the feature vector and split it at a random value. Decision rules are selected to maximize the information gain and tree splitting is stopped, once a certain depth is reached or the entropy of the remaining labels is below a threshold.

In the second stage of learning we ideally want to maximize the likelihood from equation (1) w.r.t. $\mathbf{w}$. For DTFs this involves evaluating the normalization factor $\int \exp(-E(\mathbf{y}, \mathbf{x}, \mathbf{w}))\mathrm{d}\mathbf{y}$ and for RTFs the overall energy function is of dimension $K|\mathcal{V}|$, both of which are computationally too demanding. Instead a pseudolikelihood approximation of the true likelihood is used [11, 6]:

$$P(\mathbf{y}|\mathbf{x}, \mathbf{w}) \approx \prod_{i \in \mathcal{V}} P(\mathbf{y}_i|\mathbf{y}_{\mathcal{V}\setminus\{i\}}, \mathbf{x}, \mathbf{w}) \tag{7}$$

Taking the negative log-likelihood we arrive at a non-linear optimization problem. For DTFs this problem is unconstrained and can be solved using the standard L-BFGS method. For RTFs an additional constraint has to be met which enforces that the $\mathbf{\Theta}_{u,q}$ and $\mathbf{\Theta}_{b,q}$ remain positive definite matrices. As proposed in [6] we further impose that the eigenvalues of $\mathbf{\Theta}_{u,q}$ and $\mathbf{\Theta}_{b,q}$ are constrained to the range $[e_{\min}, e_{\max}]$ with e.g. $e_{\min} = 10^{-4}$ and $e_{\max} = 10^4$ to ensure well conditioned matrices throughout. This constrained optimization problem is solved using a projected L-BFGS method based on [14]. In both cases it typically takes about 100-200 iterations to find the minimum.

4.2. DTF Inference

In the inference problem we want to find a label vector $\mathbf{y}$ for given inputs $\mathbf{x}$ and $\mathbf{w}$. For DTFs this is a discrete optimization problem and in our current implementation we solve it using simulated annealing with a Gibbs sampler as proposed in [11]. We define the unnormalized temperized distribution $P_\tau(\mathbf{y}|\mathbf{x}, \mathbf{w})$ as:

$$P_\tau(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \prod_{i \in \mathcal{V}} \exp(-\frac{1}{\tau} E_N(y_i, \mathbf{x}, \mathbf{w})) \prod_{i,j \in \mathcal{N}} \exp(-\frac{1}{\tau} E_E(y_i, y_j, \mathbf{x}, \mathbf{w})). \tag{8}$$

Starting from a random initialization $\mathbf{y}^{(0)}$ and a high temperature coefficient of e.g. $\tau^{(0)} = 20$ we repeatedly sample a new label vector $\mathbf{y}^{(t+1)}$ from the above distribution P_τ conditioned on the previous $\mathbf{y}^{(t)}$ and reduce the temperature coefficient by a fixed factor. After typically 100 to 400 iterations we arrive at a final temperature of e.g. $\tau^{(T)} = 0.01$ and an approximation $\mathbf{y}^{(T)}$ of the maximum likelihood estimator of the labels for the given scene.

4.3. RTF Inference

For RTFs an efficient, exact inference method is presented in [6]. Given that all the individual contributions to the likelihood from Equation (1) are Gaussian, the overall likelihood is Gaussian as well and the negative log likelihood takes the form

$$-\ln P(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{1}{2}\mathbf{y}^T\tilde{\mathbf{\Theta}}\mathbf{y} - \tilde{\boldsymbol{\vartheta}}^T\mathbf{y} + \ln Z. \tag{9}$$

For the inference problem we can ignore the normalization constant Z and only have to find the mean of the corresponding Gaussian, i.e. $\hat{\mathbf{y}} = \tilde{\mathbf{\Theta}}^{-1}\tilde{\boldsymbol{\vartheta}}$. Due to the large number of dimensions an iterative method is required to solve this linear equation system. Starting from an arbitrary label vector $\mathbf{y}^{(0)} = \mathbf{0}$ we find $\hat{\mathbf{y}}$ using linear Conjugate Gradients for typically around 10-20 iterations until convergence.

5. Experimental Evaluation

We experimentally evaluate our method primarily using the Cornell-RGBD-Dataset [2]. Additional results on the NYU Depth dataset [15] are presented as supplementary material. Both of these contain image sequences recorded with a Kinect and pose multi-class labeling problems. The Cornell-RGBD-Dataset provides 24 office scenes with 17 classes of objects including tables, monitors and printers, and excerpts of this dataset are shown in Figures 1 and 4. Along with the dataset, the authors also provide a set of extracted feature vectors optimized for this task.

In Section 5.1 we show that both Decision Tree Fields and Regression Tree Fields have a very competitive performance compared to state-of-the-art classifiers at much faster inference times. We then evaluate the effects of varying the number of trees in Section 5.2 showing that a forest with more trees increases the performance of DTFs and RTFs at the expense of higher computational effort. In a third experiment in Section 5.3 we evaluate the performance gained by using global knowledge of a ground plane, and finally we investigate the performance of the presented methods at finding such a ground plane in Section 5.4.

5.1. Compared to State-of-the-Art

In [2] a labeling method based on structured Support Vector Machines is presented. This method comes with two different inference algorithms, a slow but accurate one and a much faster, but less accurate one. The authors evaluated their approach by computing macro- and micro-averaged precision and recall scores on the accompanying Cornell-RGBD-Dataset. Note that the micro-averaged precision and recall are identical if a label has to be assigned to each of the segments, but the fast and approximate inference method is allowed to reject segments hence leading to different values for micro-averaged precision and recall.

We evaluate the DTF and RTF classifiers using 5-fold cross validation on the feature vectors coming along with the dataset and a comparison is given in Table 2. As a baseline we also include a Random Forest classifier (RF) on

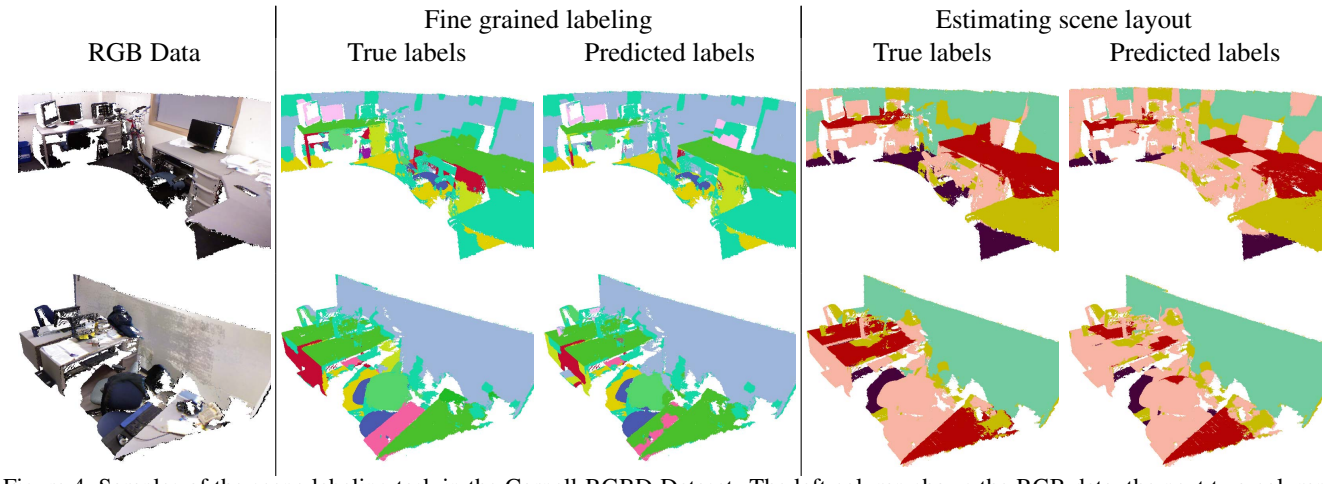

Figure 4. Samples of the scene labeling task in the Cornell-RGBD-Dataset. The left column shows the RGB data, the next two columns show the ground truth and prediction results for fine grained scene labeling and the right two columns show the same for coarse scene layout estimation.

		Macro P	Macro R	Micro P	Micro R	Training	Inference
w/o ground plane	RF	41.04	34.80	44.33		<1sec	<10msec
	DTF	70.02	46.11	60.07		10-20sec	50-300msec
	RTF	65.80	49.65	62.58		1.5-2h	50-300msec
w/ ground plane	RF	64.86	55.53	70.00		<1sec	<10msec
	DTF	85.43	67.63	81.48		10-20sec	50-300msec
	RTF	85.16	69.65	81.43		3h	50-300msec
	SVM [2]	80.52	72.64	84.06			20-30min
	SVM approx. [2]	82.95	38.14	87.41	56.82		50msec

Table 2. Experimental evaluation of macro- and micro-averaged precision and recall as well as typical timings for different classification methods using the feature vectors extracted in [2].

the unaries, but as expected it performs significantly worse than any of the methods exploiting contextual information. Both the DTF and RTF methods achieve almost identical macro averaged precision scores (85.43 and 85.16), which ranks slightly above the best SVM-based method (82.95). At recall RTFs slightly outperform DTFs (69.65 vs. 67.63), but they are again outperformed by the slow but accurate SVM-based method (72.64). However, the inference algorithms for the DTF and RTF methods are orders of magnitudes faster and comparable to the approximate, fast method based on SVMs, which only achieves a very low recall score (38.14). In summary both DTFs and RTFs achieve roughly the same performance as the slow but accurate SVM-based inference method in the same time as the fast but approximate SVM-based method. These approaches are therefore highly relevant for scene labeling, particularly if predicted labels are required at interactive rates.

We also evaluate our overall pipeline of oversegmentation, feature extraction and labeling. Note that the Cornell-RGBD-Dataset only comes with annotated 3D point clouds and the ground truth labels for these point clouds were originally created by annotating the oversegmentations from [2]. For our experiments we therefore try to find suitable ground truth labels by reprojecting the ground truth point cloud and our oversegmentation into the original camera images and we reject segments, where the label is not clear.

The results achieved with our proposed pipeline are shown in Table 3, and in this case the RTFs appear to perform better than DTFs both in labeling performance and inference time. Furthermore, the overall performance is slightly lower compared to the features extracted in [2] and evaluated in Table 2. As mentioned, the ground truth for this dataset was obtained by annotating the specific oversegmentations from [2] and differences in the segment boundaries will invariably degrade the performance. The oversegmentation of [2] also results in less complex CRFs with about 50-100 nodes per scene, whereas ours have about 1000-3000 segments of much smaller and much more regular size. This difference explains the differences in run time, but on the other hand the samples in Figures 1 and 4 show that the CRF structure strongly encourages contextual consistency for the many small segments.

5.2. Number of Trees

In Section 4 we presented formulations of DTFs and RTFs using multiple trees per term. With more trees, an increased accuracy can be expected at the cost of higher computational complexity. We investigate this effect using

		Macro P	Macro R	Micro P/R	Training	Inference
w/o ground plane	RF	42.84	14.25	49.29	3-5sec	50-100msec
	DTF	64.44	17.93	53.13	5-10min	50-120sec
	RTF	63.11	31.37	66.12	8h-10h	10-45sec
w/ ground plane	RF	50.03	29.06	69.83	3-5sec	50-100msec
	DTF	69.29	41.47	78.14	5-10min	50-120sec
	RTF	69.16	43.83	78.86	8-10h	10-45sec

Table 3. Experimental evaluation of macro- and micro-averaged precision and recall as well as typical timings for DTFs and RTFs using the pipeline and feature vectors as explained in this work.

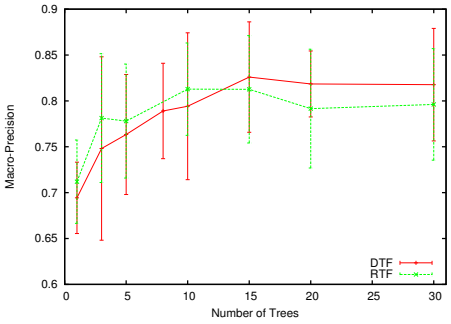

Figure 5. Effect of number of trees on classification performance.

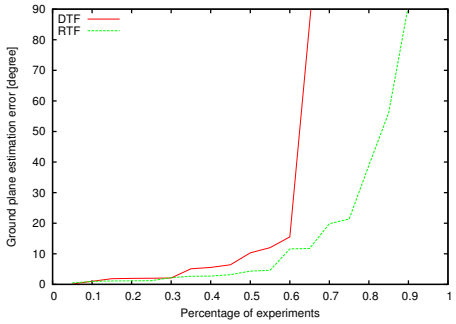

Figure 6. Errors in the estimation of the ground plane normal.

the features shipped with the Cornell-RGBD-Dataset, and the resulting macro-averaged precision scores are shown in Figure 5. As expected the performance increases with the number of trees in both the DTF and RTF formulations and saturates at about 15 trees. Similar, but slightly less pronounced increases are also observed for macro-recall and the micro-averaged values, but are omitted for clarity.

In this experiment we use the same number of trees for the unary and binary terms. We have also investigated varying the numbers of trees independently and found that the number of binary trees impacts the results more significantly than the number of unary trees. We attribute this to the greater diversity in the binary terms, where pairs of labels have to be predicted instead of a single label per segment. In the remaining experiments, we therefore typically use 10 unary trees and 15 binary trees, which appears to saturate the performance for most of our tasks.

5.3. Knowledge of Scene Layout

Prior context information such as knowledge of a ground plane and the absolute scale of a scene are important hints for the labeling task, and they are thus heavily used in the feature set we presented in Section 3. To assess their importance we re-run our scene labeling methods without using these features and compare the impact. The resulting precision and recall scores are shown in the rows entitled *w/o ground plane* in Tables 2 and 3. The significant drop in precision and recall scores underlines the relevance of such global knowledge for scene labeling and we next aim to find this knowledge automatically.

	Macro P	Macro R	Micro P/R	Training	Inference
DTF	73.94	44.10	59.27	10-20min	30-100sec
RTF	74.65	61.16	68.93	30-40min	5-20sec

Table 4. Experimental evaluation of precision and recall for estimating the coarse scene layout using DTFs and RTFs.

5.4. Estimating Scene Layout

For finding the scene layout we try to infer one of the labels $\{floor, wall, tableTop, clutter\}$ for each of the segments in the scene, and achieve this using the very same scene labeling approach as before. We reduce the set of labels to the given four classes and re-run the training and inference steps. Sample results of this labeling task are shown on the right hand side of Figure 4. To evaluate the performance we again compute the precision and recall values as a first evaluation criterion. As a second criterion we compute robust plane fits to the segments labeled as $floor$ by our system and in the ground-truth data and compute the angle between the two recovered normals.

In Table 4 we present the labeling precision and recall thus achieved with our system. While this metric gives a first impression of the performance, a much more relevant criterion is the final estimation of the ground plane, and a quantile-plot of the angular errors is given in Figure 6. From both evaluations it appears that RTFs perform better in this task than DTFs and the proposed method estimates the ground plane to within 20° in 80% of the cases.

For an overall system it is straight forward to apply a two stage approach. First, the coarse prediction is used to estimate the ground plane, and second, this ground plane is used in the computation of a fine grained scene labeling.

While this can be expected to fail in a few cases, it eliminates the need for prior knowledge about the camera setup.

It is also worth looking at the discrepancy in learning times for DTFs and RTFs as shown in Table 4 and compare it to Tables 2 and 3. The extra computational effort for RTFs is mostly due to the constraints on the eigenvalues of the matrices $\mathbf{\Theta}_{u,q}$ and $\mathbf{\Theta}_{b,q}$ in the leaves of the Regression Trees. In the case of 17 classes as in Tables 2 and 3 there is a much larger overhead compared to DTFs than for the 4 class problem in Table 4, which is obviously due to the larger eigenvalue decompositions. This of course inherently limits the number of classes the method can deal, but we have not experimented with pushing this boundary.

6. Conclusions

We have introduced a structured learning approach to 3D scene labeling that takes advantage of the recently described Decision Tree Field [11] and Regression Tree Field [6] classifiers. We show that both DTFs and RTFs achieve an almost identical, if not superior, prediction accuracy to state-of-the-art SVM-based methods, but allow for much more efficient inference steps. The input to our method are volumetric representations of the 3D scene, which can be computed in real time using KinectFusion [10]. In our current implementation the oversegmentation typically takes 2-3s and the feature extraction 5-10s. The combination with the efficient classifiers allows the system to provide annotated 3D scenes at interactive rates and renders the methods very attractive for scene labeling.

There are a number of practical measures that could improve the system. Our oversegmentation step from Section 2 computes a dense set of small segments. While this leads to a fine grained segmentation of object boundaries and while the CRF formulation does an excellent job at grouping the small segments into semantically consistent units, their sheer number poses a high computational burden on the CRF. It would be interesting to improve upon this by e.g. pre-grouping similar segments or even using hierarchical approaches. Also we currently impose an empirical threshold on the context range in an attempt to manage the complexity of the CRF. Although this works well in practice, and certainly is much more flexible than traditional 4- or 8-connected MRF/CRF models, long range interactions such as the relationship between the bounding walls of a room may not be captured. Again, a hierarchical approach with long-range contextual relations between objects and short range relations between individual parts of the objects might be beneficial. Finally our current system relies on a Kinect sensor and the KinectFusion system, but volumetric 3D scene representations can recently also be acquired with standard RGB cameras and DTAM [9]. While we expect the 3D shape information to be less reliable in this case, it would be interesting to compare the performance.

Acknowledgments We gratefully acknowldge the support of the UK Engineering and Phyiscal Sciences Research Council (grant EP/H050795), and the Australian Research Council (grant DP130104413), and Laureate Fellowship FL130100102 to IDR).

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *PAMI*, 34(11):2274–2282, 2012.

[2] 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*, 32(1):19–34, 2013.

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

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893, 2005.

[5] A. Flint, D. Murray, and I. Reid. Manhattan scene understanding using monocular, stereo, and 3d features. In *ICCV*, pages 2228–2235, 2011.

[6] J. Jancsary, S. Nowozin, and C. Rother. Regression tree fields an efficient, non-parametric approach to image labeling problems. In *CVPR*, pages 2376–2383, 2012.

[7] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *PAMI*, 21(5):433–449, 1999.

[8] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *ISMAR*, pages 83–86, 2009.

[9] R. Newcombe, S. Lovegrove, and A. Davison. Dtam: Dense tracking and mapping in real-time. In *ICCV*, pages 2320–2327, 2011.

[10] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136, 2011.

[11] S. Nowozin, C. Rother, S. Bagon, T. Sharp, B. Yao, and P. Kohli. Decision tree fields. In *ICCV*, pages 1668–1675, 2011.

[12] I. Posner, M. Cummins, and P. Newman. A generative framework for fast urban labeling using spatial and temporal context. *Autonomous Robots*, 26(2):153–170, 2009.

[13] X. Ren, L. Bo, and D. Fox. Rgb-(d) scene labeling: Features and algorithms. In *CVPR*, pages 2759–2766, 2012.

[14] M. Schmidt, E. Van Den Berg, M. Friedlander, and K. Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. In *Conference on Artificial Intelligence and Statistics*, pages 456–463, 2009.

[15] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *ICCV Workshops*, pages 601–608, 2011.

[16] H. Wang, S. Gould, and D. Koller. Discriminative learning with latent variables for cluttered indoor scene understanding. In *ECCV*, pages 435–449, 2010.