Dense 3D Semantic Mapping of Indoor Scenes from RGB-D Images

Alexander Hermans, Georgios Floros and Bastian Leibe

***Abstract*— Dense semantic segmentation of 3D point clouds is a challenging task. Many approaches deal with 2D semantic segmentation and can obtain impressive results. With the availability of cheap RGB-D sensors the field of indoor semantic segmentation has seen a lot of progress. Still it remains unclear how to deal with 3D semantic segmentation in the best way. We propose a novel 2D-3D label transfer, based on Bayesian updates and dense pairwise 3D Conditional Random Fields. This approach allows us to use 2D semantic segmentations to create a consistent 3D semantic reconstruction of indoor scenes. To this end, we also propose a fast 2D semantic segmentation approach based on Randomized Decision Forests. Furthermore, we show that it is not needed to obtain a semantic segmentation for every frame in a sequence in order to create accurate semantic 3D reconstructions. We evaluate our approach on both NYU Depth datasets and show that we can can obtain a significant speed-up compared to other methods.**

I. INTRODUCTION

3D scene understanding is required in many robotics applications. For scenarios such as autonomous navigation or general object interaction, knowledge of the surrounding scene is vital. While semantic segmentation can be an important cue for scene understanding, many approaches are not efficient enough and only focus on 2D images.

In order to truly utilize semantic information we want to create a semantically annotated 3D reconstruction of a surrounding scene, where every 3D point is assigned a semantic label. Furthermore, we want to enforce spatial and temporal consistency in such reconstructions. However, there is no clear-cut method for the transfer of 2D labels into a globally consistent semantic 3D reconstruction yet.

In this paper we address the task of dense semantic 3D scene understanding of indoor scenes. We build a point cloud reconstruction of the scene and assign a semantic label to each 3D point. As an initial step, we perform an efficient 2D semantic segmentation of the RGB-D frames. Our main contribution is a novel way to transfer the 2D image labels into a 3D reconstruction based on Bayesian updates [1] and dense pairwise Conditional Random Fields (CRFs) [2], that allows us to enforce temporal and spatial constraints.

Semantic segmentation of 2D images recorded from indoor scenes has been shown to be an especially challenging task [3]. This is caused by the large variability of both object and scene types, varying illumination and unconstrained scene layouts. Current methods achieve good results, but they typically need more than one minute per image [4], [5], [6]. As a first step towards a more efficient method, we propose a semantic segmentation approach based on Randomized Decision Forests (RDFs). Due to their parallel nature, RDFs

All authors are with Computer Vision Group, RWTH Aachen University. Email: {hermans,floros,leibe}@vision.rwth-aachen.de

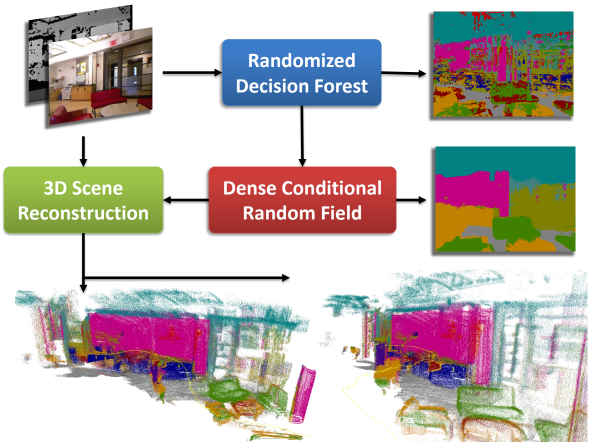


Fig. 1: A high level overview of our approach. We classify RGB-D images using a Randomized Decision Forest and refine the result using a dense CRF. Based on a sequence we create a 3D point cloud reconstruction of a scene. Based on our novel 2D-3D label transfer approach we can assign a class label to each 3D point, giving us a dense semantic reconstruction.

are well suited for efficient semantic segmentation. It has previously been shown that they can obtain impressive results [7]. However, the more expressive variants tend to suffer from expensive feature computations [6]. We propose a set of features for RDFs to keep them efficient, without loosing too much accuracy. When compared to fast semantic segmentation methods we obtains state-of-the-art results for both speed and classification accuracy.

Considering that in most typical scenarios the movement between two consecutive frames is small, the question arises as to whether it is necessary to segment every frame in a sequence. An important insight we gain is that semantic segmentation does indeed not need to be performed for every frame. We analyze how our semantic segmentation approach, coupled with the 2D-3D label transfer, behaves when processing only a subset of the frames. Based on this we show that it is not critical that each part is real-time capable, but that it still suffices if certain components of our approach run at far lower frame rates.

This paper is structured as follows. The next section gives an overview of related work. Section III describes the general idea of our complete approach. Sections IV, V and VI then describe the components of our approach in more details. Finally, Section VII discusses our experimental results.

II. RELATED WORK

Several 3D semantic segmentation approaches exist for both outdoor [8], [9], [10] and indoor [11], [12], [13], [5] scenes. Floros *et al.* [8] create a sterro 3D reconstruction and enforce temporal consistency through an additional potential in a CRF ranging over several frames in a sequence. Triebel *et al.* [9] label scene reconstructions in an unsupervised

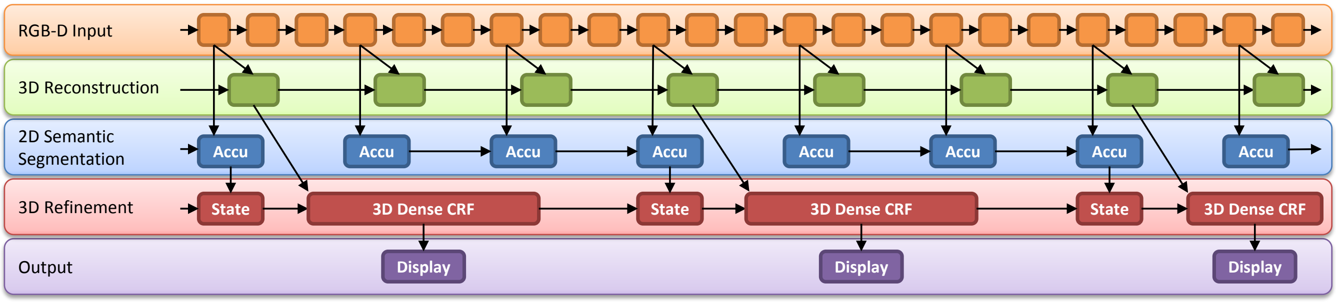


Fig. 2: **Overview of our method.** Between the input ( ), where each box represents an input frame, and output ( ) our approach is split into three processes. The *3D Reconstruction* ( ) fuses new frames into the point cloud reconstruction. The *2D Semantic Segmentation* ( ) classifies the RGB-D image and accumulates the results for each 3D point. Finally, the *3D Refinement* ( ) fuses this accumulated information with the previous state of a point. It can clearly be seen that the different processes can be executed at different rates. The visualized frequencies do not correspond to actual values.

way. They focus on the actual learning of different classes based on online clustering and incremental belief updates. Hu *et al.* [10] create a labeled 3D scene reconstruction. They aim to speed up the fusion of streamed laser scan data for outdoor scenes, based on an efficient representation of the point cloud. In contrast, we would also like to use the valuable information obtained from color images. Nüchter *et al.* [11] create 3D maps from laser scan data and reason about planes they find in the scene. They use heuristics to decide if a plane belongs to a wall, floor, ceiling or a door and furthermore detect some objects in the scene based on additional classifiers. This approach is limited to certain classes, while our approach can learn a larger number of classes using the RDF classifiers. Stückler *et al.* [12] obtain a 3D voxel reconstruction of a scene from an RGB-D slam approach and fuse the labels of different RGB-D frames using Bayesian updates. They focus on a subset of object classes and label the remaining scene as background, while we use classes covering general indoor scenes. Anand *et al.* [13] stitch together a small set of RGB-D frames to an indoor point cloud reconstruction. They segment the point cloud into patches and use both visual and geometrical cues for labeling the scene. They propose a relaxed version of their initially slow inference method, but it is unclear how long the feature computation takes. Valentin *et al.* [5] create a 3D mesh for the scene and obtain label hypotheses for every face based on appearance and geometric properties using a boosted classifier. A CRF is defined over the 3D mesh to get a globally consistent segmentation. They only perform inference on a full scene once, instead of building the semantic segmentation incrementally.

For 2D image semantic segmentation, common pipelines use a Textonboost framework [14], [15], which is optimized using a CRF [15], [2]. While these approaches obtain impressive results for outdoor scenes, different approaches are needed for the more complex indoor scene scenes. Silberman and Fergus [3] use a neural network classifier, based on SIFT features extracted from both the color and depth images and optimize the result using a grid CRF. Ren *et al.* [4] use a complex approach based on kernel descriptors and linear support vector machines evaluated on different levels of a segmentation tree. For the second NYU Depth dataset, Silberman *et al.* [16] focus on structural labels and support relationships. These are jointly inferred, formulated as an integer programming problem. Couprie *et al.* [17] use a multiscale Convolutional Neural Network (CNN) to infer both semantic and structural classes. To label video sequences, they enforce a simple temporal consistency based on temporally smoothed superpixels in consecutive frames. Gupta *et al.* [6] obtain state-of-the-art performance by creating superpixels based on depth and appearance contours and classifying these using either SVMs or RDFs. Apart from the CNN approach [17], all of the above 2D semantic segmentation methods need around a minute to obtain semantic segmentation results for a single image. The CNN approach needs approximately 0.7 seconds per image.

III. 2D-3D LABEL TRANSFER

We focus on 3D point cloud reconstructions with dense and consistent label information. Our approach is decoupled into three separately running processes (see Fig. 2). The *3D reconstruction* process takes a new image from a sequence of RGB-D images and adds it to the currently existing 3D reconstruction. The *2D semantic segmentation* process creates a soft classification for each pixel that corresponds to a 3D point in the cloud and accumulates this information for the points. The *3D refinement* process, which is the core of our approach, takes the newly accumulated information for each point and fuses it into the point's current state, which is represented by a class distribution. To obtain a globally optimal class labeling for the 3D reconstruction, we distribute the information over neighboring points depending on the distance, color similarity, and normal orientation. Fig. 2 also shows that the input data arrives at a higher rate than the other processes are executed. This is justified by the observation that the camera position tends to be very similar from frame to frame, so it suffices to run both the reconstruction and the semantic segmentation process only for a subset of the input frames. This also means we can accumulate information from several frames before we update a 3D point's state in the refinement process. Consequently we need to store a 3D point's current state and its currently accumulated information.

The next sections will explain each process in detail. As the 3D refinement does not depend on the type of 2D semantic segmentation and 3D reconstruction we will start with the 3D refinement, which is the core of our approach.

IV. 3D REFINEMENT

The 3D refinement process consists of two steps. The first step takes newly accumulated classification data and fuses it with the current state. We model the first step as a Bayesian update [1] that takes the current belief for a 3D point and updates it with the new predictions. The second step enforces spatial consistency which we achieve by applying a dense pairwise CRF over the 3D point cloud. A new point's state is initialized with an equal class distribution.

A. Fusing New Classification

Our goal is to obtain a class distribution for each 3D point. At time t, we denote the class distribution of a single point as c_t and all the pixel measurements that contribute to this point are denoted as $x_0^t = \{x_0, x_1, ...x_t\}$, meaning we are interested in $p(c_t|x_0^t)$. Applying Bayes' rule to this gives us:

$$p(c_t|x_0^t) = \frac{1}{Z_t} p(x_t|x_0^{t-1}, c_t) p(c_t|x_0^{t-1}), \tag{1}$$

where $Z_t = p(x_t|x_0^{t-1})$. Here we can use a first order Markov assumption and assume that $p(x_t|x_0^{t-1}, c_t) = p(x_t|c_t)$ because the measurement x_t is conditionally independent of previous measurements given the class c_t. Furthermore, we assume a smoothness of the posterior and thus use $p(c_t|x_0^{t-1}) \approx p(c_{t-1}|x_0^{t-1})$. This gives us:

$$p(c_t|x_0^t) = \frac{1}{Z_t} p(x_t|c_t) p(c_{t-1}|x_0^{t-1}). \tag{2}$$

Based on this we can take the the 3D point's current state $p(c_{t-1}|x_0^{t-1})$ and update it with the accumulated new predictions from the 2D semantic segmentation. Considering we use a RDF classifier, we get a prediction for the posterior $\widehat{p}(c_t|x_t)$. Again using Bayes' rule, we reformulate Eq. 2 to:

$$p(c_t|x_0^t) = \frac{1}{Z_t} \frac{p(c_t|x_t)p(x_t)}{p(c_t)} p(c_{t-1}|x_0^{t-1}). \tag{3}$$

While we cannot estimate the actual prior probability $p(x_t)$, we can simply fuse it with the normalization factor Z_t as it does not depend on c_t, giving us our final update equation:

$$p(c_t|x_0^t) \leftarrow \frac{1}{Z'_t} \frac{\widehat{p}(c_t|x_t) p(c_{t-1}|x_0^{t-1})}{p(c_t)}, \tag{4}$$

with $Z'_t = p(x_0^t|x_0^{t-1}) \cdot p(x_t)^{-1}$. This can now be computed as both the old state, the class prior and the new prediction are known and we do not need to compute Z_t explicitly, but can instead normalize the probability distribution.

B. Enforcing Spatial Consistency

After the new data has been fused into the state, we want to both distribute this information over neighboring points and also obtain information from neighbors. Influence between neighbors should be proportional to their distance and their visual and geometrical similarity. For this purpose, we use a dense CRF. Let $\mathbf{C} = \{C_1, C_2, ..., C_N\}$ be a set of random variables corresponding to the 3D points $i \in \{1, ...N\}$. Each random variable C_i takes a label from $\mathcal{L} = \{l_1, l_2, ..., l_k\}$ when considering k different classes. Based on a CRF, the probability distribution for a possible labeling $\mathbf{c} \in \mathcal{L}^N$ given a point cloud $\mathbf{X}$ is defined by:

$$P(\mathbf{C} = \mathbf{c}|\mathbf{X}) = \frac{1}{Z(\mathbf{X})} \exp\left(-E(\mathbf{c}|\mathbf{X})\right), \tag{5}$$

where $E(\mathbf{c}|\mathbf{X})$ is the Gibbs energy defined over the CRF graph and $Z(\mathbf{C})$ is the partition function. The dense CRF graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ contains vertices v_i corresponding to the random variables C_i, with edges between each pair of vertices ($\mathcal{E} = \mathcal{V} \times \mathcal{V}$). The Gibbs energy is defined over the unary and pairwise cliques in the graph given by:

$$E(\mathbf{c}|\mathbf{X}) = \sum_{i \in \mathcal{V}} \psi_u(c_i|\mathbf{X}) + \sum_{(i,j) \in \mathcal{E}} \psi_p(c_i, c_j|\mathbf{X}). \tag{6}$$

As the unary potential we use $\psi_u(c_i|\mathbf{I})$ the negative logarithm of the current state of point i. The pairwise potential $\psi_p(c_i, c_j|\mathbf{X})$ is given by a linear combination of Gaussian kernels:

$$\psi_p(c_i, c_j|\mathbf{X}) = \mu(c_i, c_j) \sum_{m=1}^{K} w^{(m)} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j), \tag{7}$$

where $\mu(x_i, x_j)$ is a simple Potts model, $w^{(m)}$ are weights for the different kernels, $\mathbf{f}_i$ and $\mathbf{f}_j$ are feature vectors for the points i and j, and $k^{(m)}$ are Gaussian kernels, given by:

$$k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) = \exp\left(\frac{1}{2}(\mathbf{f}_i - \mathbf{f}_j)^{\mathbf{T}} \Lambda^{(m)} (\mathbf{f}_i - \mathbf{f}_j)\right). \tag{8}$$

These are characterized by the symmetric positive-definite precision matrix $\Lambda^{(m)}$, which defines the shape of the kernel. When limiting the pairwise potential to a combination of Gaussian kernels, the efficient mean-field approximation inference approach introduced by Krähenbühl and Koltun [2] can be applied. The probability distribution $P(\mathbf{C})$ is approximated by a distribution $Q(\mathbf{C})$ that minimizes the KL-divergence $\mathbf{D}(Q||P)$, such that Q is a product over its marginals $Q(\mathbf{X}) = \prod_i Q_i(C_i)$. This distribution is approximated by an iterative approach with linear complexity in the number of points. After a number of iterations, the globally optimal labeling can be obtained for each point by setting the label $C_i = \arg\max_l Q_i(l)$.

In our case we use two Gaussian kernels for the pairwise potential. The first is an appearance kernel,

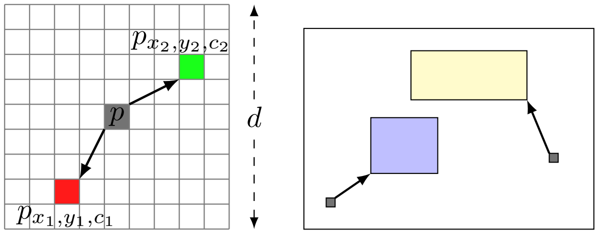
$$w^{(1)} \exp\left(-\frac{|\mathbf{p}_i - \mathbf{p}_j|^2}{2\theta_\alpha^2} - \frac{|\mathbf{l}_i - \mathbf{l}_j|^2}{2\theta_\beta^2}\right), \tag{9}$$

where $\mathbf{p}$ are the 3D positions of the points and $\mathbf{l}$ are color vectors in LAB colorspace. The parameters θ_α and θ_β, specify the range in which points with similar coordinates and colors affect each other, respectively. This kernel is used to model longer range connections between points with a similar appearance. The second Gaussian kernel is a smoothness kernel,

$$w^{(2)} \exp\left(-\frac{|\mathbf{p}_i - \mathbf{p}_j|^2}{2\theta_\gamma^2} - \frac{|\mathbf{n}_i - \mathbf{n}_j|^2}{2\theta_\delta^2}\right), \tag{10}$$

where $\mathbf{n}$ are the 3D normals of the points. This kernel operates on a smaller range, specified by θ_γ, thus enforcing a local, appearance-agnostic smoothness amongst points whose normal vectors are similar. These parameters can be obtained using piece-wise learning.

These models tend to converge very fast ($\sim$10 iterations) in which case the probability distribution of the possible classes collapses to a single class. As we want to use the



(a) (b)

Fig. 3: **Visualization of different features. (a)** A pixel value comparison, used by several features. **(b)** Patches relative to a pixel. The feature value is the average of the patch.

probability distribution in further Bayesian updates, we only execute a few iterations of the dense CRF. However, since the distributions can already be very focused on a single class after one iteration, we do not set the new distribution as the new state directly. Instead we use the distribution as a prediction in another Bayesian update step as defined in Eq. 4. This gives us the final class distribution after the 3D refinement step.

V. 2D SEMANTIC SEGMENTATION

To create 2D semantic segmentations we use a RDF. As each pixel is evaluated separately by a RDF, the results can be rather noisy. To obtain a spatially smoother result, we apply a dense pairwise CRF in 2D on the RDF output.

A. Randomized Decision Forest

We use a typical RDF framework, containing a number of binary trees. A pixel traverses the tree and the path is decided based on a feature and a threshold stored in every node. Once the pixel reaches a leaf node, it is assigned the corresponding class distribution. The final result is obtained by averaging the distributions of the different leaf nodes.

1) Features: The RDFs draw their strength from the features used to make splitting decisions. More expressive features can lead to higher classification accuracies [6]. But often these features are based on computationally expensive superpixel and segment computations. Only using simple features such as pixel comparisons can already yield impressive results [7], however the classification accuracies are not comparable to state-of-the-art classifiers. We have evaluated a large set of computationally feasible features to find a good balance between speed and accuracy. Among others we tried pixel and patch comparisons in the image or depth map, normal vectors, and more complex geometrical features. This section will describe the seven features in our final selection.

Following [7], we use a simple color feature. It returns the value of a color channel c_1, from a pixel p_{x_1,y_1,c_1}, relative to the current pixel p within a $d \times d$ patch surrounding p (as shown in Fig. 3a). Our second feature takes two relative pixels and compares their depth values by subtraction [18]. Furthermore, the relative distances are normalized by the depth of the current pixel, which makes the feature more robust to camera translations. The third feature is the same as the second, with the difference that now again color values are compared instead of depth values. Both color features are evaluated on the LAB colorspace. The two most expressive features are the height of a point in 3D and the relative depth in the scene. The relative depth is defined as the depth of a pixel, normalized by the furthest depth in the corresponding image column [3]. The last two features we use are based on patches in the image, relative to a pixel (see Fig. 3b). We calculate oriented gradients over either the L color channel or the depth map and perform soft binning similar to [19]. Then we extract average values of patches within these bins. Empirically we have found this feature set to work best for our purposes of indoor semantic segmentation.

2) Training: Our training procedure is similar to that of [7]. Initially, we sample our training samples I uniformly from the training images. Each of the trees is trained on a subset of these training samples $I' \subseteq I$. A set of samples I' is assigned to the root node of a tree and the set is recursively split up and distributed to the child nodes. For each node, 500 splitting features and for each of these 13 thresholds are considered. Unlike [7], we use the normalized information gain [20] to select the features with the best score. In our experiments this gave slightly better classification accuracies than the unnormalized information gain [7]. Splitting is stopped once either the maximum tree depth is reached, the leaf nodes are pure, or they contain less than a minimum number of samples n_{min}.

The used training data contains a large bias towards certain classes, which is reflected by a bias towards these classes in the resulting segmentations. In [7], this effect is alleviated by weighting each training pixel with its inverse class frequency during information gain computation. However, this re-weighting scheme only partially alleviates the problem and the bias is still clearly visible. To overcome this problem, we sample a set of training pixels, with an equal class distribution for each tree. This set is passed through the tree to learn new class distributions in the leaf nodes. Although this slightly decreases the global accuracy of the RDF, it gives a significant improvement of the class-average accuracy. Furthermore, we can assume an equal class prior in the Bayesian update steps as all used probabilities are based on results of RDFs trained with this scheme.

B. 2D Dense Pairwise CRF

To obtain smoother 2D semantic segmentation results, we apply a Dense Pairwise CRF as described in section IV-B. The difference being that the unary potential is now directly obtained from the RDF and that the pairwise potentials are different. We again use a smoothness kernel and an appearance kernel, but the feature vectors differ. For the smoothness kernel we only use the 2D pixel locations, which simply enforces smoothness in the image and for the appearance kernel we replace the 3D point location by the 2D pixel location. As we only evaluate the RDF for pixels with a valid depth, all other pixels are initialized using a class equal unary potential.

VI. 3D SCENE RECONSTRUCTION

Our 3D reconstruction approach is based on our previous work for stereo depth reconstruction [8]. We slightly adjusted

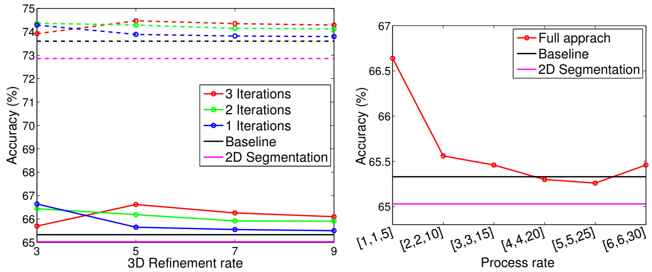


Fig. 4: **Quantitative reconstruction results. (left)** The colored lines each represent results for different numbers of 3D dense CRF iterations. The different measurements represent after how many new frames the 3D refinement is executed. Dashed lines represent the global accuracy and full lines class-average accuracy. **(right)** Results for different rates of the processes. The first number represents the execution rate of the 3D Reconstruction, the second that of the 2D Semantic segmentation and the third that of the 3D Refinement.

it to handle RGB-D frames, but the idea remains similar. We use Visual Odometry (VO) information to merge the point clouds of each depth map to a global reconstruction of the scene. For the VO estimation we use the fovis library [21], which finds 2D feature matches across images and computes 3D transforms based on the correspondences. The VO information is computed for every frame in a sequence in order to be able to correctly fuse frames into the reconstruction. However, this step is fairly inexpensive as it only takes ~25ms for each frame on a single core.

Given the camera position and orientation, we project points into global 3D space and associate each point with a zero-velocity Kalman filter, which tracks the point's spatial uncertainty. We initialized the Kalman filter using forward error propagation, which uses the image level uncertainties and propagates them into 3D space, based on the projection matrix. The forward error propagation for the Kinect sensor is explained in [22]. To find correspondences between new points and existing points, we project every existing point onto the camera plane, giving us a hypothetical depth value at a certain pixel location. If the difference between the actually measured depth value and the hypothetical depth is below a threshold we fuse the new point with the existing point. We then use the new point as a measurement to update the Kalman filter, giving us an updated spatial uncertainty and 3D position. If no matching point exists in the point cloud, we create a new point. In order to obtain clean point cloud reconstructions we only create a new point if the spatial uncertainty is below a threshold, if it has a valid normal vector, and if this normal vector is not close to perpendicular to the vector between the camera center and the 3D point. We assign each point a color and a normal vector and track the pixel-point correspondences to transfer the 2D class probabilities to the 3D points once available.

VII. EXPERIMENTAL RESULTS

A. Datasets

We evaluate our 2D semantic segmentation approach on the two NYU Depth datasets [3], [16]. Both contain images from indoor scenes, recorded with a Kinect sensor. The first dataset contains 2284 labeled RGB-D images, for which missing depth values in the depth maps have been filled. We evaluate our approach on the 12 core classes provided in [3]. The second dataset contains 1449 labeled images with the same preprocessing. While it contains fewer annotated images, it contains a larger variety of scenes and objects. We use the default training and test set splits. Furthermore, both datasets contain a large amount of raw images, which we use for the reconstructions. The test images from the first NYU Depth dataset are taken from 29 scenes. We created reconstructions for all of these scenes. For four scenes the visual odometry failed to produce correct trajectories, caused by missing frames in the sequences. We excluded these scenes and evaluated on the remaining 25 scenes, containing bathrooms, bedrooms, a bookstore, kitchens, living rooms and offices. In total we used over 25k images, with 652 ground truth annotations. Unless specified differently, results are based on the first dataset, as our main evaluation is done on this dataset. While the second dataset has a higher annotation quality, fewer images are labeled per scene, rendering it less suitable for the reconstruction experiments.

B. 3D Scene Experiments

We evaluate our approach in a causal way, meaning after fusing a new image into the 3D reconstruction we track the 2D-3D correspondences and for each pixel, we take the most likely label of the corresponding 3D point. This gives us a label for every pixel with a valid depth value. We then evaluate based on the 2D annotations. We run the 3D reconstruction and 2D semantic segmentation for each frame in a sequence and accumulate the predictions for each 3D point. For the 2D semantic segmentation we use the RDF described in Section V-A and refine the segmentation using a 2D dense CRF which runs for 3 iterations. The 3D refinement is only executed every n^{th} frame. To limit the computational cost of the refinement step, we only update points that accumulated new predictions in the past 10 frames.

As a baseline for our approach we take our single image semantic segmentation, which evaluates a RDF and then applies a dense pairwise CRF in 2D, using 12 iterations. Here we also only consider pixels with a valid depth to obtain a fair comparison (Fig. 4, shown in black). We also show the results when only evaluating the 2D semantic segmentation for every frame and simply accumulating the output in the 3D points (Fig. 4, shown in magenta).

The left graph in Fig. 4 shows both class-average (continuous lines and global accuracy (dashed lines) for different values of n and for different numbers of 3D dense CRF iterations. As it can be seen, our approach outperforms the baseline in every tested configuration. The improvement mainly depends on the number of 3D dense CRF iterations. The best results are obtained using 3 iteration and running the 3D refinement every 5^{th} frame. This gives a class average accuracy of 66.62%, which improves the baseline of 65.03% by more than 1.5%. Qualitative results for three scenes using this configuration can be seen in Fig. 5. While numerically the improvement seems rather small, it should be noted that it is purely based on spatial and temporal consistency. We did not consider a higher number of iterations due to the

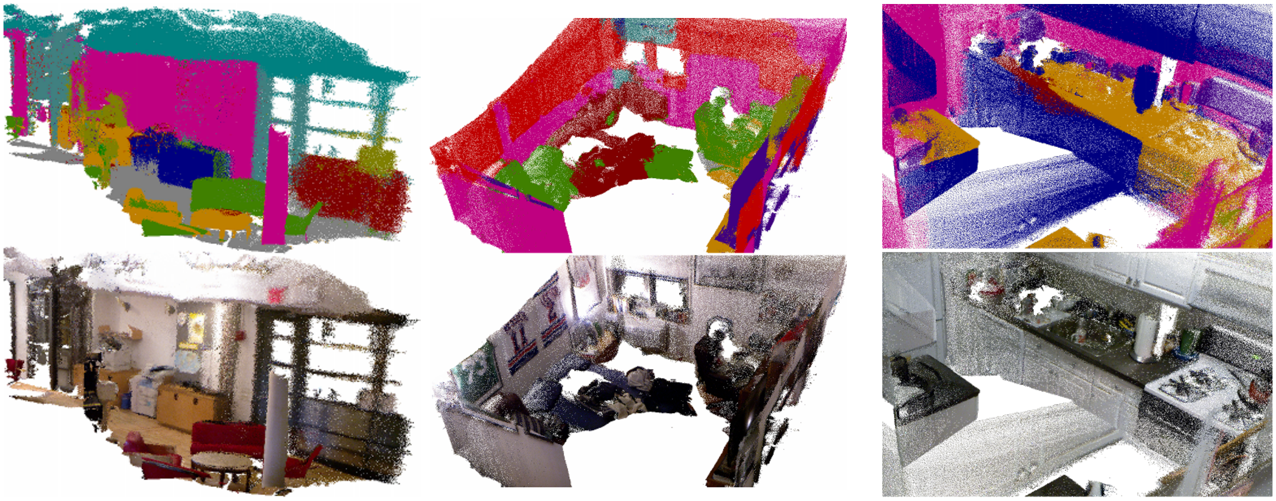


Fig. 5: **Qualitative results of our 3D reconstructions. (top row)**) Semantic reconstructions **(bottom row)** Corresponding RGB reconstructions. For these reconstructions a semantic segmentation was created for every frame and the 3D refinement was done every five frames, using three 3D dense CRF iterations. All scenes are from the first NYU Depth dataset. **(left)** Office 14, **(middle)** bedroom 4 and **(right)** kitchen 3. Label colors are listed in Table II.

TABLE I: **Timing results.** The upper part of the table lists the time needed to execute different components of our approach. Times are averaged over the different scenes and a range is listed if the times differ due to the different point cloud sizes. The lower part shows different frames rate of different configurations. All timings are based on a Intel i7 3.4GHz CPU.

Component	Consumed time (ms)		
General Preprocessing	$\sim$110		
Gradient calculation	$\sim$70		
Visual Odometry	$\sim$25		
Point cloud fusion	70 - 120		
RDF evaluation	$\sim$200		
2D CRF 3 or 12 Iter.	$\sim$360	$\sim$960	
3D CRF 1, 2 or 3 Iter.	400 - 1800	500 - 2200	600 - 2500

Setup	Frame rate (Hz)
Single Image baseline	$\sim$0.75
[1,1,5] 1 3D CRF Iter.	0.8 - 1.0
[3,3,15] 1 3D CRF Iter.	2.2 - 2.5
[6,6,30] 1 3D CRF Iter.	3.9 - 4.6

consequences for the runtime.

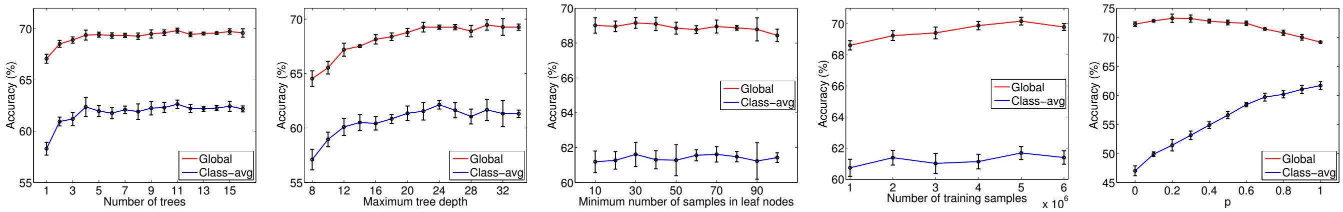
It can also be seen that the results are relatively stable regarding the number of executed refinement steps. In fact for a higher number of 3D dense CRF iterations it is better to execute the 3D refinement less often. We believe this is due to the fact that the class distributions will collapse if the dense CRF is applied too often, which in turn does not work well with the Bayesian updates.

C. Runtime Analysis

Table I shows the runtime of several components in our approach. The upper part gives a detailed time for different steps in our pipeline. The general preprocessing includes color conversions, normal, and basic feature computations. Together with the gradient calculation, the RDF evaluation, and a 2D CRF this makes up the basic 2D segmentation. The 3D reconstruction consists of the VO and the point cloud fusion. The point cloud fusion tends to vary based on the number of points in a scene, as we loop over all points to find corresponding pixels. The 3D refinement step consists of a 3D dense CRF and two Bayesian updates. The later do not contribute to the actual runtime significantly. The lower part of Table I show the frame rates for the different configurations. The single image baseline, which uses 12 dense CRF iterations, runs at $\sim$0.75Hz. Our full 3D reconstruction approach, running the 3D refinement every 5^{th} frame, runs slightly faster while obtaining better results and creating a reconstruction of the scene.

Motivated by the fact that we do not need to run the 3D refinement every frame to improve results, we wanted to know how the results are affected if we do not run the 3D reconstruction and 2D semantic segmentation for every frame either. Starting out from the configuration where the 3D refinement is executed every 5 frames, we ran several experiments where we increased the number of completely skipped frames. Results for these experiments can be seen in the right graph of Fig. 4. On the x-axis we plot the different configurations, the first number specifies after how many new frames the 3D reconstruction is executed, the second after how many the 2D semantic segmentation and the last after how many frames the 3D refinement is done. For the 3D dense CRF we only update points that accumulated new predictions in the previous 10 frames that we actually evaluated. Meaning the window size is twice the number of the skipped frames. As we can see, results decrease rapidly at first, but then seem to be somewhat stable. The accuracy when only generating semantic segmentations for each 6^{th} frame is very similar to that of the single image baseline. However, the speed improves by a factor of more than five. Improving the frame rate of $\sim$0.75Hz to a frame rate around 4Hz (see the lower part of Table I). Also lower subsampling rates result in a significantly higher frame rate. To our knowledge no other methods create semantic 3D reconstructions of indoor scenes at such speed.

Our code can further be optimized and currently does not exploit parallelism to its full potential. The three processes currently still run sequential and thus block the progress of the other processes. By actually running the processes in parallel, we are optimistic that we can achieve higher frame rates. This is especially motivated by the fact that even though the RDF uses eight threads in our current



(a) (b) (c) (d) (e)

Fig. 6: **RDF parameter experiments.** The graphs show both the class average and global classification accuracy. **(a)** The number of trees in the RDF. **(b)** The maximum tree depth. **(c)** The minimum number of samples needed further splitting of a node. **(d)** The number of training samples used during training. (Each tree is trained on a subset). **(e)** The weight for the new and old class distribution, when repassing equally sampled data through the trees.

implementation, it fails to fully use the capacity of the CPU, most likely due to many memory accesses. These currently wasted clock cycles might otherwise be used by the 3D refinement or the reconstruction of the next frame. So even while our current implementation does not run at a video frame rate, we believe this is within reach, even if the actual components of our pipeline run at a lower frame rate.

D. RDF Experiments

To gain a better understanding on how RDFs work for indoor semantic segmentation we investigate the influence of several parameters. We do this by varying a single parameter at a time. As seen in Fig. 6 certain parameters have a big influence. Both the number of trees and the maximum tree depth converge to a stable range. We limited our experiments to 8 trees for efficiency reasons (Fig. 6a). The performance of the RDF changes significantly with the allowed tree depth (Fig. 6b). Generally trees are limited in depth to prevent overfitting, this could not be confirmed in our experiments; even when growing the trees without depth constraints, the results did not degrade significantly. Closely related is the minimum number of samples required in a node, to consider further splitting (Fig. 6c). This parameter has a less significant effect. The number of training samples has a small effect on the performance. Fig. 6d suggests that larger training sets increase classification accuracy slightly, however the training time increases linearly as well. The repassing of the equally sampled training data has the most significant impact. To show the effect, we add the original and the new class distribution, weighted with $(1-p)$ and p respectively, where $p \in [0, 1]$ (Fig. 6e). While the global classification accuracy is reduced by $\sim 3\%$, the class average accuracy increases by $\sim 15\%$ when using $p = 1$. We did similar experiments for other parameters, such as the number of considered splits or specific features parameters. In our experiments these parameters were less significant. In general the RDF parameters seem robust to minor changes when values from the stable ranges are used.

E. 2D Semantic Segmentation Experiments

Finally we show how our single image segmentation approach (RDF followed by 12 iterations of the 2D dense CRF) compares to other 2D semantic segmentation approaches, we evaluated it on both NYU Depth datasets. Here we use the preprocessed images, thus being able to label every pixel.

1) NYU Depth version 1: We compare our 2D segmentation approach to the Extended Textonboost approach (based on the publicly available ALE library [15]). Here we only use the 12 core classes defined in [3]. We clearly improve the results, as can be seen in Table II. Even when we only use the RDF, the quantitative results are better for several classes already. This yields rather noisy results, but the dense CRF removes most of the noise (see Figs. 7d and 7e). Compared to the Extended Textonboost approach, the results are better aligned to actual image contours (see Figs. 7c and 7e).

Two other approaches provide results on this datasets, but they use an additional background class. This contains all labels in the dataset that could not be mapped to one of the 12 classes. We did not use this class in our experiments, as we think it contains no useful information. For a fair comparison we train an additional classifier using 13 classes. Silberman and Fergus [3] only provide their class average accuracy of 56.6%. Which we improve by almost 3%. Ren *et al.* [4] obtain better results for all classes, but their complex approach takes over a minute per image.

2) NYU Depth version 2: For all experiments we use the same parameters as for the first dataset. Couprie *et al.* [17] group all classes in the dataset to 13 semantic classes. Silberman *et al.* [16] provide results for the segmentation based on 4 structural classes. Quantitative results for both class sets are shown in Table III. We obtain better results for 9 of the 13 semantic classes and improve both the global and class average classification accuracy. Gupta *et al.* [6] also report results on the second NYU Depth dataset, however they use different evaluation measurements. Similar to Ren *et al.* [4], their approach is more accurate than our 2D segmentation, however also runs significantly slower. To our current knowledge, only Couprie *et al.* [17] evaluated their approach with regards to its runtime. We outperform their approach both in accuracy and speed. On downscaled images (320×240) they need 0.7 seconds per frame. This means that our single image approach is almost three times faster.

VIII. CONCLUSION

We have introduced a novel way to create a semantic 3D reconstruction of indoor scenes. We show how 2D semantic segmentation of RGB-D images can be achieved efficiently, using RDFs and 2D dense CRFs. We then apply our 2D-3D label transfer to obtain a semantic segmentation of a 3D reconstruction. Finally, we show that for our approach it is not needed to label every frame in a sequence and analyze how this affects the quality of the 3D reconstruction. In future work we plan to change our implementation to fully exploit parallelism, which we believe will further increase the speed of our approach.

TABLE II: **NYU Depth dataset version one. (upper half)** Comparison to the Extended Textonboost approach [15]. **(lower half)** Comparison of our single image semantic segmentation to two other approaches. We improve the class average accuracy of Silberman and Fergus [3]. However, we cannot compete with the method of Ren *et al.* [4] yet.

Label color	Bed	Blind	Booksh.	Cabinet	Ceiling	Floor	Picture	Sofa	Table	TV	Wall	Window	Backgr.	Global	Average
Extended Textonboost (no CRF)	14.6	3.5	56.1	34.5	58.8	73.8	47.7	31.8	45.7	53.3	88.7	12.2	-	67.1	43.3
Extended Textonboost (CRF)	16.4	1.2	53.3	38.4	64.8	80.6	30.8	33.3	52.8	53.4	**92.1**	13.4	-	70.4	44.2
Extended Textonboost (dense CRF)	14.1	3.2	57.2	34.5	59.2	75.9	48.6	33.8	**47.0**	56.3	90.3	11.4	-	68.2	44.3
Ours (RDF 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**	-	65.0	57.1
Ours (full 2D segmentation)	**57.6**	**57.3**	**67.5**	**58.2**	**92.7**	**88.5**	56.6	**66.7**	45.7	**82.0**	77.6	17.2	-	**71.5**	**64.0**
Silberman and Fergus [3]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	56.6
Ren *et al.* [4]	**85**	**80**	**89**	**66**	**93**	**93**	**82**	**81**	**60**	**86**	**82**	**59**	**35**	-	**76.1**
Ours (full 2D segmentation)	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	44.4	59.5

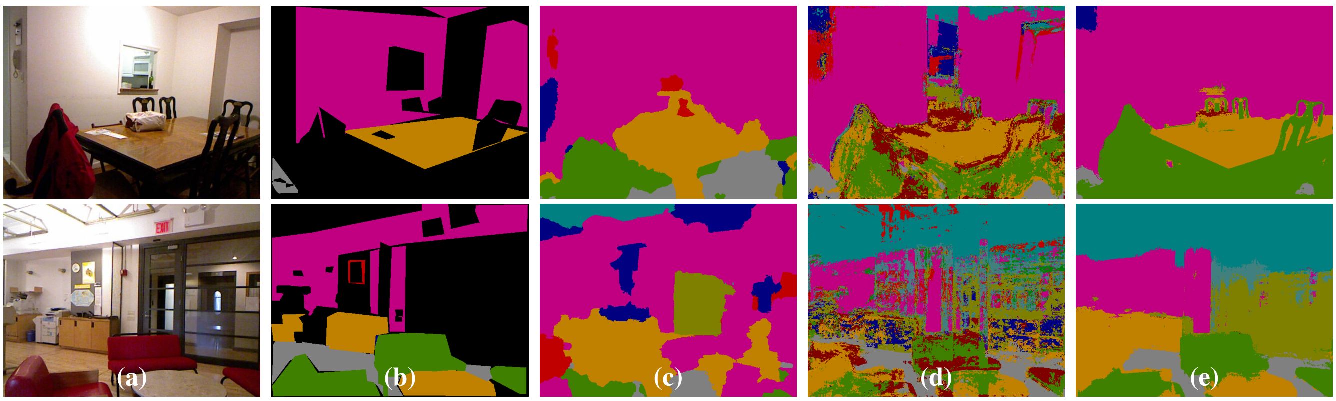


Fig. 7: **Qualitative results for the first NYU Depth dataset. (a)** Color Images. **(b)** Ground truth annotations. **(c)** Extended Textonboost results, using the dense CRF. **(d)** Intermediate RDF results. **(e)** Final results, after applying the 2D dense CRF. See Table II for label colors.

TABLE III: **NYU Depth dataset version 2. (left)** Using classes defined by [17]. **(right)** Using the structural classes defined by [16]. We improve both the results from Silberman *et al.* [16] and Couprie *et al.* [17]. Especially when using the semantic classes, we get improved results for several classes.

Label color	Bed	Obj.	Chair	Furnit.	Ceiling	Floor	Deco.	Sofa	Table	Wall	Window	Booksh.	TV	Global	Average	Ground	Furnit.	Props	Struct.	Global	Average
Silberman *et al.* [16]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	68	**70**	**42**	59	58.6	59.6
Couprie *et al.* [17]	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	52.4	36.2	87.3	45.3	35.5	**86.1**	63.5	64.5
Ours (full 2D segmentation)	**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**	**54.2**	**48.0**	**97.4**	61.8	40.9	76.1	**68.1**	**69.0**

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

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

[3] N. Silberman and R. Fergus, "Indoor scene segmentation using a structured light sensor," in *ICCV Workshop on 3D Repr. and Recognition*, 2011.

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

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

[6] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images," *CVPR*, 2013.

[7] M. Johnson and J. Shotton, "Semantic texton forests," in *Computer Vision: Detection, Recognition and Reconstruction*. Springer, 2010.

[8] G. Floros and B. Leibe, "Joint 2D-3D temporally consistent semantic segmentation of street scenes," in *CVPR*, 2012.

[9] R. Triebel, R. Paul, D. Rus, and P. Newman, "Parsing Outdoor Scenes from Streamed 3D Laser Data Using Online Clustering and Incremental Belief Updates," in *AAAI*, 2012.

[10] H. Hu, D. Munoz, J. A. Bagnell, and M. Hebert, "Efficient 3-D Scene Analysis from Streaming Data," in *ICRA*, May 2013.

[11] A. Nüchter and J. Hertzberg, "Towards semantic maps for mobile robots," *RAS*, vol. 56, no. 11, pp. 915–926, 2008.

[12] J. Stückler, N. Biresev, and S. Behnke, "Semantic mapping using object-class segmentation of RGB-D images," in *IROS*, 2012.

[13] A. Anand, H. S. Koppula, T. Joachims, and A. Saxena, "Contextually Guided Semantic Labeling and Search for 3D Point Clouds," *IJRR*, vol. 32, no. 1, pp. 19–34, 2013.

[14] J. Shotton, J. M. Winn, C. Rother, and A. Criminisi, "TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context," *IJCV*, vol. 81, no. 1, pp. 2–23, 2009.

[15] L. Ladicky, C. Russell, P. Kohli, and P. H. S. Torr, "Associative hierarchical CRFs for object class image segmentation," in *ICCV*, 2009.

[16] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *ECCV*, 2012.

[17] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor Semantic Segmentation using depth information," *CoRR*, vol. abs/1301.3572, 2013.

[18] J. Shotton, A. W. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *CVPR*, 2011.

[19] J. Gall, A. Yao, N. Razavi, L. J. V. Gool, and V. S. Lempitsky, "Hough Forests for Object Detection, Tracking, and Action Recognition," *PAMI*, vol. 33, no. 11, pp. 2188–2202, 2011.

[20] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[21] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera," in *ISRR*, 2011.

[22] J.-H. Park, Y.-D. Shin, J.-H. Bae, and M.-H. Baeg, "Spatial Uncertainty Model for Visual Features Using a Kinect$^{\text{TM}}$ Sensor," *Sensors*, vol. 12, no. 7, 2012.