Journal of Real-Time Image Processing (JRTIP), available online, Springer, 2013.

Real-Time Image Processing manuscript No.
(will be inserted by the editor)

Jörg Stückler · Benedikt Waldvogel · Hannes Schulz · Sven Behnke

Dense Real-Time Mapping of Object-Class Semantics from RGB-D Video

Received: date / Revised: date

Abstract We propose a real-time approach to learning semantic maps from moving RGB-D cameras. Our method models geometry, appearance, and semantic labeling of surfaces. We recover camera pose using simultaneous localization and mapping while concurrently recognizing and segmenting object classes in the images. Our object-class segmentation approach is based on random decision forests and yields a dense probabilistic labeling of each image. We implemented it on GPU to achieve a high frame rate. The probabilistic segmentation is fused in octree-based 3D maps within a Bayesian framework. In this way, image segmentations from various view points are integrated within a 3D map which improves segmentation quality. We evaluate our system on a large benchmark dataset and demonstrate state-of-the-art recognition performance of our object-class segmentation and semantic mapping approaches.

1 Introduction

Robots that perform complex tasks in unstructured environments require the ability to categorize surfaces into semantic classes. Made persistent in a semantic map this knowledge is available for reasoning about tasks and for communication with humans. In this paper, we propose a real-time approach to learning semantic maps on-the-fly from a moving camera. Our semantic mapping system integrates efficient simultaneous localization and mapping (SLAM) with real-time object-class segmentation of RGB-D images. Our SLAM approach is based on rapid registration of RGB-D images using multi-resolution surfel maps (Stückler and Behnke 2013). It extracts key views along the camera trajectory which are aligned through graph optimization. Our implementation is highly efficient and performs SLAM at about 10 Hz on a CPU. Its frame rate is sufficient to map an environment in real-time from a handheld camera moved at moderate speeds. Each RGB-D frame is segmented for object classes using random decision forests (RF, Breiman 2001) concurrently and in real-time on a GPU. Our image segmentation approach uses depth for scale-invariance and incorporates shape and texture cues seamlessly to provide a probabilistic labeling into object classes. The probabilistic image labeling are fused in 3D within a Bayesian framework given the trajectory estimate of SLAM. By this, segmentation evidence from various view points improves the overall segmentation quality in the map.

We assess run-time and recognition performance of our object segmentation method and demonstrate the benefits of fusing recognition information from multiple views in a 3D map. We evaluate our approach on a large RGB-D dataset for object-class segmentation that contains image sequences in a variety of indoor scenes.

All authors are with
Computer Science Insitute VI, Autonomous Intelligent Systems, University of Bonn, Germany
Contact E-mail: stueckler@ais.uni-bonn.de

2 Related Work

Many mapping approaches build geometric representations of the environment, e.g., using sensors like 2D and 3D laser scanners, monocular and stereo cameras. Since the commercial introduction of affordable, high-resolution RGB-D cameras, several methods have been explored that process images of such sensors to acquire 3D maps efficiently. A prominent example is KinectFusion (Newcombe et al 2011) which aligns depth images on the GPU towards a map that is incrementally accumulated from the registered images. The map is represented by 3D voxels that store signed distance towards the measured surface. Whelan et al (2012) extend this approach towards larger scenes through moving volumes. In own work, we apply rapid dense registration of RGB-D images (Stückler and Behnke 2013) and graph optimization to learn multi-resolution surfel maps.

In contrast to these dense methods, some approaches map sparse interest points. Henry et al (2012), for example, extract interest points and textured surface patches,

register them using ICP (Besl and McKay 1992), and apply graph-optimization to obtain an accurate map. Endres et al (2012) match SURF features between RGB-D frames and refine the registration estimate using ICP. While RGB-D SLAM recovers scene geometry and appearance, it does not incorporate valuable semantic information like place or object labels into the map.

Some systems map semantics. While most approaches use SLAM as a front-end to obtain a sensor trajectory estimate (Zender et al 2008; Vasudevan et al 2007; Meger et al 2008; Nüchter and Hertzberg 2008; Castle et al 2010; Civera et al 2011), some methods also incorporate the spatial relation of objects into SLAM. Tomono and Shin'ichi (2003), for example, detect polyhedral object models in images and perform SLAM in 2D maps using the detected objects as landmarks. In contrast to our approach, this method is restricted to objects with clearly visible linear edges. Zender et al (2008) apply SLAM in 2D maps using laser scanners, recognize objects using SIFT features, and map their locations in the 2D map. In addition to SIFT-based recognition, Vasudevan et al (2007) also detect doors in laser scans since they are important topological objects that connect rooms. Meger et al (2008) combine semantic 2D mapping of objects with attention mechanisms. In contrast, we build 3D semantic maps with dense object information. Nüchter and Hertzberg (2008) use ICP, plane segmentation, and reasoning to label planar segments in 3D maps that they acquire using 3D laser scanners. They apply AdaBoost on Haar wavelets and SVM classifiers on contour descriptions to detect objects and persons in the 3D maps. In our approach, we segment the original image data and fuse segmentation evidence from multiple views. Castle et al (2010) and Civera et al (2011) propose vision-based mapping of objects. In both approaches, SLAM is solved through feature-based monocular EKF-SLAM. Objects are recognized using SIFT features and persistently maintained in the 3D feature map. The approach of Ranganathan and Dellaert (2007) learns 3D constellation models of places composed of objects using SIFT features. In this approach, the map consists of a set of places with associated models. The aforementioned approaches, however, do not build dense 3D semantic maps. Closely related to our approach are the works by Lai et al (2012), Sengupta et al (2013), and Salas-Moreno et al (2013). Lai et al (2012) use the confidence score of an object detector to generate a dense soft labeling of an image and integrate the labelings in a voxel representation. The approach requires about 4 seconds per frame and, to the best of our knowledge, has not yet been implemented to perform in real-time with SLAM in the loop. In urban scenes, Sengupta et al (2013) label stereo images using conditional random fields and fuse the information in 3D stereo sequences. The run-time of this method is reported to be within seconds per frame. The approach by Salas-Moreno et al (2013) recognizes specific object instances in a scene and estimates the pose of the objects in a map using SLAM techniques. Our method provides dense semantic classification of the surfaces in a map.

We integrate image-based object-class segmentation with SLAM from RGB-D images into a semantic 3D mapping framework. Each image is segmented pixel-wise into object classes and background. Based on the SLAM estimate, this information is then projected into 3D to fuse object recognition results from multiple views. This not only provides 3D segmentations of objects, but also improves classification accuracy.

RFs have been applied to a variety of image segmentation problems such as object-class segmentation (Shotton et al 2008; Stückler and Behnke 2010) and human body part labeling (Shotton et al 2011). Semantic texton forests, proposed by Shotton et al (2008), use simple features of luminance and color at single pixels or comparisons between two pixels in a RF classifier. Using image-level priors and a second stage of RFs, local and scene context is incorporated into the classification framework. Recently, RFs have been successfully applied for segmenting human body parts and tracking body pose in real-time using depth images. Shotton et al (2011) propose to normalize feature queries with the available depth to obtain scale-invariant recognition. We extend RFs for object-class segmentation by incorporating both depth and color features. As in previous own work (Stückler et al 2012), we use region features in color and depth and normalize for scale changes to gain an efficient classifier for RGB-D images. For the problem of object-class segmentation, we also need to consider that objects may vary strongly in size between the classes. We propose a stratification mechanism to present sufficient amounts of training pixels of each object class.

We implemented our object-class segmentation method on GPU to classify images in real-time. Sharp (2008) presented a RF implementation on GPU using Direct3D that accelerates classification around 100 times, compared to a CPU implementation. It is used by Microsoft for human body part recognition from single depth images in the XBox Kinect (Shotton et al 2011). Their system operates in real-time and classifies all image pixels in less than 5 ms on a XBox 360 GPU. In contrast to our work, they only evaluate depth information and use simplified features that compare single pixels instead of region averages, in order to reduce computational complexity. We implemented training on GPU to enable efficient optimization of hyper parameters.

3 RGB-D Object-Classes Segmentation using Random Decision Forests

3.1 Structure of Random Decision Forests

RFs $\mathcal{F}$ are ensembles of K binary decision trees $\mathcal{T}_k$. Each node n in a tree classifies an example by a binary decision on a scalar feature function that quantifies local appear-

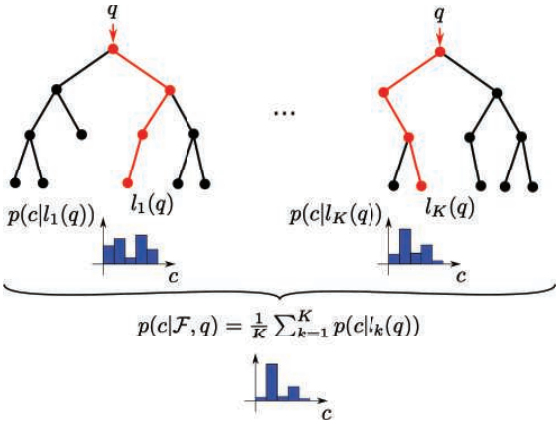

Fig. 1 Random decision forests (RFs). Query pixels q are classified in a binary decision cascade in each tree. Nodes in a tree cast binary decisions on the pixels. Query pixels are soft classified by the empirical class probability $p(c \mid l(q))$ of training pixels that arrive at a leaf $l(q)$. The posterior classification probability for the RF is determined by the average over trees.

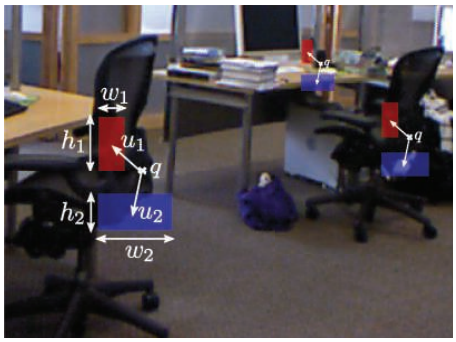

Fig. 2 Random decision forest features. Local shape and appearance at a query pixel q is calculated from the difference of average values in two offset regions. We exploit dense depth to normalize for scale changes and scale relative offset locations u_i and region extents w_i, h_i with the inverse of the depth $d(q)$ at the query pixel.

ance or shape in the image. In addition, each node is associated with a distribution $p(c \mid n)$ over class labels $c \in \mathcal{C}$ that arrived at the node during training. Randomness is injected into the classifier by considering only a random subset of the training data for generating a tree and by sampling node functions from only a random subset of the available binary decision functions. In this way, trees are decorrelated and generalization performance increases.

The probabilistic labeling at a query pixel q is determined as the posterior distribution over class labels encoded in the forest (illustrated in Fig. 1). In this process, the example pixel is passed down each decision tree $\mathcal{T}_k$, branching at each node according to its binary decision criterion until a leaf node l is reached. The posterior distribution is computed by averaging over the individual distributions at the leaf nodes $l_k(q)$ that the example reaches, i.e.,

$$p(c \mid \mathcal{F}, q) = \frac{1}{K} \sum_{k=1}^{K} p(c \mid l_k(q)).$$

3.2 RGB-D Image Features

As scalar feature functions (i.e., features) we determine differences in local regions of depth or color. Dense depth is used to normalize the features for scale changes (see Fig. 2). More formally, we parametrize a feature evaluated at pixel q by

$$f_\theta(q) := \frac{\sum_{p \in R_1(q)} \phi_1(p)}{|R_1(q)|} - \frac{\sum_{p \in R_2(q)} \phi_2(p)}{|R_2(q)|}, \tag{1}$$

where $R_j(q) := R\left(q + \frac{u_j}{d(q)}, \frac{w_j}{d(q)}, \frac{h_j}{d(q)}\right)$ is the rectangular image region at the offset u that is normalized in offset position and size by the depth $d(q)$ measured at the query pixel. The features are configured by parameters θ that comprise unnormalized offset positions u_j, region extents w_j, h_j, and image channels ϕ_j. Note, that we restrict comparisons to either two depth regions or between any two regions in color channels, and represent color in the CIE Lab color space. In the depth image, the region size $|R_j(q)|$ counts the number of valid depth readings in the region. If an offset region contains no valid depth measurement or lies beyond the image, the pixel traverses to the right child node. We efficiently implement region features using integral images.

Each node in the decision tree decides on the query pixels with a threshold τ to either pass the pixel further to its left or right child. Individually, each feature gives only small information about the object class at a pixel. Within the cascades in the decision trees, however, the tests describe complex texture and shape patterns which allows accurate classification of pixels.

3.3 Training Procedure

We train each of the K decision trees with a subset $\mathcal{D}$ of images from the training dataset. From each image we extract N pixels randomly for training. We stratify the training examples by resampling to a uniform distribution in class labels in order to normalize the amount of training examples for object size. We will, however, have to consider the actual distribution of class labels in the training images at later stages in order to incorporate the prior probability of each class into the classifier.

We train the decision trees in a depth-first manner by choosing feature parameters θ and a threshold τ at each node and splitting the pixel set Q accordingly into left and right subsets Q_l and Q_r:

$$\begin{aligned} Q_l(\theta, \tau) &:= \{q \in Q \mid f_\theta(q) < \tau\} \text{ and} \\ Q_r(\theta, \tau) &:= \{q \in Q \mid f_\theta(q) \geq \tau\}. \end{aligned} \tag{2}$$

Since the parameter space cannot be evaluated analytically, we sample P random parameter sets and thresholds (e. g., $P = 2000$) and select feature and threshold that yield maximal information gain

$$I(\theta,\tau) := H(Q) - \sum_{s\in\{l,r\}} \frac{|Q_s(\theta,\tau)|}{|Q|} H\left(Q_s(\theta,\tau)\right), \qquad (3)$$

where $H(Q) := -\sum_{c\in\mathcal{C}} p(c \mid Q) \log_2 \left(p(c \mid Q)\right)$ is the Shannon entropy of the distribution of training class labels in pixel set Q. This splitting criterion finds feature parameters and threshold that most distinctively separate the pixel set at a node. Each node is split until a maximum depth is reached in the tree, or the number of pixels lies below a minimum support threshold.

At each leaf node l, we want to maintain the distribution $p(c \mid l, D)$ of pixels of class c that arrive at the node from the original training set. Since we train the decision tree from pixels with equally distributed class labels, we actually measure the class distribution $p(c \mid l, Q)$ of training pixels Q at the leaf, i.e.,

$$p(c \mid l, Q) := p(c_q \mid l, q \in Q) = p(c_q \mid l, q \in Q, q \in \mathcal{D}). \qquad (4)$$

The distribution of interest can be obtained by applying Bayes rule:

$$\begin{aligned} p(c \mid l, Q, \mathcal{D}) &= \frac{p(q \in Q \mid c_q, l, q \in \mathcal{D})\ p(c_q \mid l, q \in \mathcal{D})}{p(q \in Q \mid l, q \in \mathcal{D})} \\ &= \frac{p(q \in Q \mid c_q, q \in \mathcal{D})\ p(c_q \mid l, q \in \mathcal{D})}{p(q \in Q \mid q \in \mathcal{D})}. \end{aligned} \qquad (5)$$

For the desired distribution we obtain

$$p(c_q \mid l, q \in \mathcal{D}) = \frac{p(c_q \mid l, q \in Q)\ p(q \in Q \mid q \in \mathcal{D})}{p(q \in Q \mid c_q, q \in \mathcal{D})}. \qquad (6)$$

We further reformulate the probability of a pixel of class c to be included in the class-equalized training data Q to

$$p(q \in Q \mid c_q, q \in \mathcal{D}) = \frac{p(c_q \mid q \in Q)\ p(q \in Q \mid q \in \mathcal{D})}{p(c_q \mid q \in \mathcal{D})} \qquad (7)$$

and obtain

$$p(c_q \mid l, q \in \mathcal{D}) = \frac{p(c_q \mid l, q \in Q)\ p(c_q \mid q \in \mathcal{D})}{p(c_q \mid q \in Q)}. \qquad (8)$$

By design, $p(c_q \mid q \in Q)$ is uniform among class labels and, hence, we incorporate the distribution of classes in the complete training set into the leaf distributions through

$$p(c \mid l, D) = \eta\ p(c \mid l, Q)\ p(c \mid \mathcal{D}), \qquad (9)$$

where $\eta^{-1} := p(c \mid Q) = 1/|\mathcal{C}|$.

We found that if there is a large imbalance of class pixel occurrences in the image, single training pixels from frequent classes that reach a leaf may outweigh many pixels from less frequent classes, and hence degrade segmentation accuracy dramatically. In such unbalanced datasets we subtract a fraction ρ of the total pixels that reached the leaf from each class count.

3.4 Real-Time Classification on GPU

We use CUDA (NVIDIA 2013) to implement random decision forest classification on GPU inspired by Sharp (2008). Figure 3 depicts how a binary tree is mapped in breadth-first order to a 2D layered texture on GPU, using one layer per tree and one row per node. Left child node IDs, threshold, feature parameters and histogram values are stored in columns. Nodes in the same tree level are located in consecutive rows. We do not store IDs of right child nodes, as they directly follow left child nodes due to breadth-first order.

Integral images are calculated and transferred to a 2D layered texture on GPU, using one layer per channel. To calculate a region average, we only need to query four pixels from the integral image.

Depth images can contain undefined values in case the camera cannot measure distance. In consequence, our implementation needs to distinguish between depth and color features. To calculate region averages in depth images, we need an additional channel that stores the number of valid pixels. Region sizes and offsets are normalized by depth of the query pixel. Color features require nine pixel queries in either one or two different image channels. Depth features require nine pixel queries in the depth channel and eight queries in the additional channel to count the number of valid measurements.

The image pixels are processed in parallel on GPU. We use thread blocks with 256 threads to process 16×16 pixel patches. The spatial locality of image queries leads to a high texture cache hit rate, which is crucial for performance. Each thread traverses the tree from root to leaf in a branchless loop, such that all threads access a tree level at the same time. The spatial locality of nodes in one level leads to high cache hit rate in the tree data texture.

Threads need to branch on feature type which leads to two different code paths. However, we measure only a slight performance decrease, as the GPU executes threads that follow the same code path in parallel.

Leaf node histogram values of all trees are combined and returned as $H \times W \times C$ matrix for C classes and $H \times W$ pixels images.

Our implementation scales linearly in the number of trees, tree depth and the number of pixels per image. Feature complexity and choice of parameters have significant impact on overall performance. Larger region offsets, for instance, lower cache hit rate which causes additional global memory fetches.

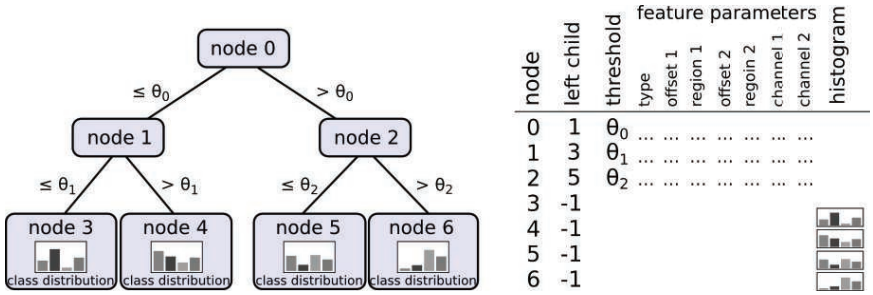

Fig. 3 Mapping of a binary tree (left) to a 2-dimensional texture on GPU (right). Every node is mapped to a row in breadth-first order. Decision nodes contain left child node ID, threshold and feature parameters. Leaf nodes are indicated with a negative child ID and contain the class distribution as determined during training.

4 View-based SLAM using Multi-Resolution Surfel Maps

We perform SLAM using our framework for dense RGB-D SLAM based on multi-resolution surfel maps (MRSMap, Stückler and Behnke (2013)). Our framework converts RGB-D images into a compact multi-resolution representation which can be efficiently aggregated from images and registered with other maps. The maps represent the 3D data at multiple resolutions in an octree. Each voxel maintains the 6D normal distribution of colored points that fall into the voxel which we denote as surfels (from surface elements).

Image aggregation is made efficient through exploitation of image neighborhood: Nearby points in 3D project to close-by points in the 2D image. We scan row-wise through the image to find connected components of pixels that project to the same octree leaf and accumulate surfel statistics for these leaves. Afterwards, we can efficiently build the octree with only a few thousand node insertions instead of up to 307,200 individual pixel insertions. Our map representation also allows for integrating evidence from images from multiple view points. In this case, the shape and color statistics of pixels from several images are accumulated within the voxels.

For registration, we optimize for the relative pose between two MRSMaps in a dual iterative refinement process. In each iteration, we establish associations between surfels in both maps under the current pose estimate. We exploit efficient neighborhood look-ups in the octree to find initial closest surfel matches. In further iterations, we efficiently bootstrap the associations from previous iterations by searching in the direct voxel neighborhood of associated nodes, while for unassociated nodes we again search with local volume queries. We establish associations first on the finest resolution possible to achieve high accuracy and save redundant associations on coarser resolutions. Accurate pose optimization is performed using a combination of Levenberg-Marquardt and Newton's method on the log-likelihood of the surfel matchings.

Our SLAM method extracts key views that are represented by MRSMaps along the camera trajectory. The motion of the camera is tracked through registration towards the closest key view in the map, while new key views are generated after sufficient camera motion. We build a graph of spatial relations in a hypothesize-and-test scheme, in which nearby key views are tested for successful pair-wise registrations. At each frame, we optimize the camera poses within the key view graph using the g2o framework (Kuemmerle et al 2011).

Our efficient implementation allows real-time SLAM on standard multi-core CPUs. Typically, we achieve frame-rates between 5 Hz and 10 Hz for VGA resolution (640×480) images, which is sufficient to perform mapping in real-time for moderately fast camera motion.

5 Dense Real-Time Semantic Mapping of Object-Classes

5.1 Probabilistic 3D Mapping of Object-Class Image Segmentations

Our online SLAM approach provides an estimate for the motion of the camera $\mathcal{S}$, while object segmentation yields a probabilistic labeling $\mathcal{Z}$ of the image according to the RGB-D images. Our aim is to fuse the object segmentations from individual images into a 3D semantic map. We use our efficient image aggregation techniques in MRSMaps to generate multi-resolution voxel maps that store beliefs on object classification in each voxel.

Formally, we store the belief $Bel(c_v)$ in each voxel v to be labeled as one of the object classes c_v,

$$Bel(c_v) = p(c_v \mid \mathcal{Z}, \mathcal{S}). \tag{10}$$

The labeled image pixels are projected into 3D to find corresponding voxels in the map. The beliefs in each voxel v are then updated in a Bayesian framework with the pixel observations $q_{1:N} := \{q_1, q_2, \ldots, q_N\}$ that fall into a voxel:

$$p(c_v \mid q_{1:N}, \mathcal{S}) = \sum_{c_{q,1},\ldots,c_{q,N}} p(c_v, c_{q,1}, \ldots, c_{q,N} \mid q_{1:N}, \mathcal{S}). \tag{11}$$

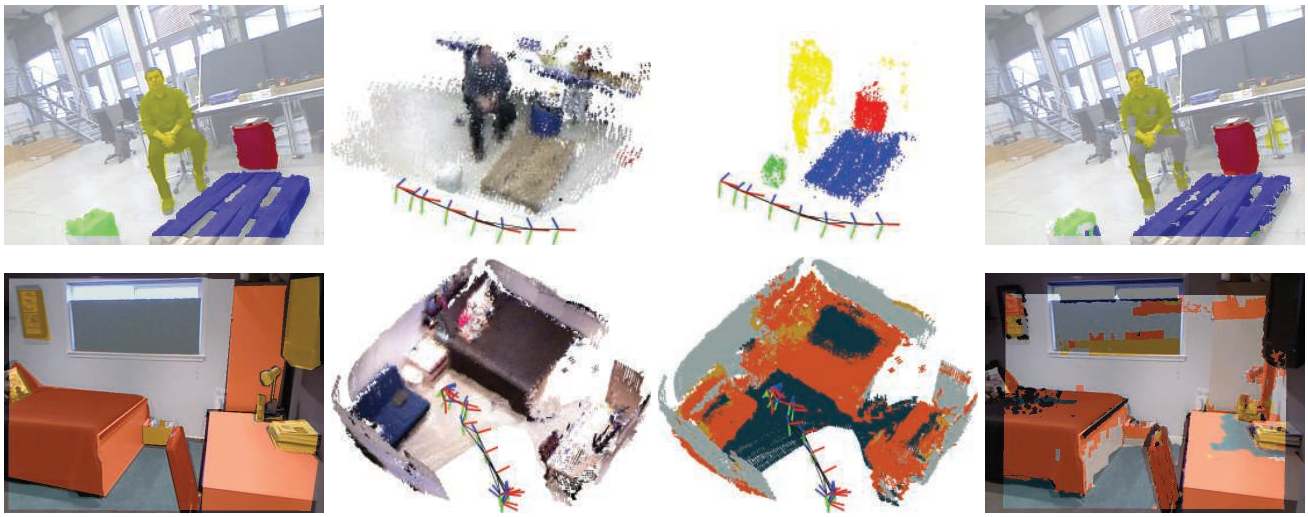

Fig. 4 Semantic mapping. From left to right: Ground-truth overlay on RGB image of a scene; Samples from MRSMaps overlayed in 3D and SLAM key view graph; Class belief for MRSMap samples in semantic 3D map; Object-class segmentation backprojected from semantic 3D map into image. Top: AIS Large Objects scene. Bottom: NYU Depth v2 scene.

Note that the known trajectory can be neglected in the further derivation to ease notation. Bayes rule yields

$$p(c_v \mid q_{1:N}) = \sum_{c_{q,1},\ldots,c_{q,N}} p(c_v \mid c_{q,1},\ldots,c_{q,N},q_{1:N}) \; p(c_{q,1},\ldots,c_{q,N} \mid q_{1:N}). \quad (12)$$

The left term is further factored using Bayes rule, while for the right term we impose independence between pixel observations. This yields

$$p(c_v \mid q_{1:N}) = p(c_v) \sum_{c_{q,1},\ldots,c_{q,N}} \prod_i \eta_i \, p(c_{q,i} \mid c_v) \, p(c_{q,i} \mid q_i), \quad (13)$$

where $\eta_i := 1/p(c_{q,i} \mid c(q_{i+1}),\ldots,c(q_N))$ are normalization factors for each observation. The RF classifier provides the likelihood $p(c_{q,i} \mid q_i)$ through $p(c_{q,i} \mid q_i, \mathcal{F})$, while the probability $p(c_v) =: Bel_0(c_v)$ incorporates prior knowledge on the belief which we set to uniform in our experiments. For the distribution $p(c_{q,i} \mid c_v) = \mathbf{1}_{\{c_v\}}(c_{q,i})$ we assume a deterministic one-to-one mapping such that

$$p(c_v \mid q_{1:N}, \mathcal{S}) = Bel_0(c_v) \prod_i \eta_i \, p(c_{q,i} = c_v \mid q_i, \mathcal{F}). \quad (14)$$

This belief update can be performed recursively in a time-sequential manner which is applied in our online semantic SLAM system.

5.2 Integrated Real-Time Semantic Mapping

We integrate object-class segmentation, SLAM, and semantic 3D fusion into a real-time operating semantic mapping system (see Fig. 5). Since object-class segmentation and SLAM are performed on GPU and CPU, respectively, we can execute both components in parallel. Once pose estimate and semantic labeling of the RGB-D image is available, we fuse the labeling into the semantic map of the reference key view.

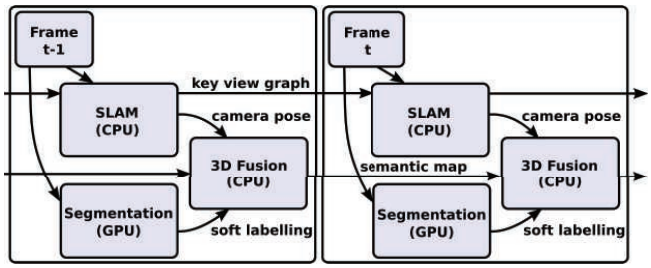

Fig. 5 Online semantic SLAM system. Each frame is segmented for object-classes on GPU and processed for SLAM (CPU) in parallel. Results are fused in 3D semantic maps.

Each key view in the map maintains a local aggregated semantic map in our approach, since the relative poses of the key views are subject to graph optimization in each frame and, hence, a single global map cannot be maintained. Global segmentation beliefs at a 3D position $Bel(p)$ can be easily obtained by combining the beliefs $Bel(c_v^k)$ of individual key views $k \in \mathcal{K}$ in a single estimate according to

$$Bel(p) = \eta \prod_k Bel(c_v^k), \quad (15)$$

where η is a normalization constant and c_v^k is the classification of the maximum resolution node v that contains p in key view k. Note that this process can be significantly sped up by restricting the set of key views $\mathcal{K}$ to the

views that contain query pixels or have sufficient frustum overlap with whole query images.

6 Experiments

We evaluate run-time and recognition performance of our semantic SLAM method in extensive experiments. We used two datasets to demonstrate our approach on two different qualities of object classes. Both datasets have been recorded using Microsoft Kinect cameras at VGA (640×480) RGB and depth image resolutions. Since ground truth for the camera trajectory is not available for the datasets, we kindly refer the reader to (Stückler and Behnke 2013) for an assessment of the accuracy of our real-time image registration and SLAM approach.

The NYU Depth v2 dataset (Silberman et al 2012) contains 590 RGB-D sequences recorded in 464 scenes with 408,473 frames in total. It comes with 1449 images with manual ground-truth labeling of object-classes. We evaluate on the four abstract object-classes *ground*, *structure*, *furniture*, and *props* that distinguish all 35,064 object instances in the dataset. The dataset has been split into disjunct training and test sets comprising 795 training and 654 test images with ground truth in 359 and 231 sequences, respectively.

We also use the AIS Large Objects dataset introduced in Stückler et al (2012) to classify four fine-grained object classes (large objects of *props*-type) from background. It consists of 40 sequences in different scene configurations and has been split into 30 training and 10 test sequences with 500 ground-truth labeled images each (50 per test sequence). The test sequences comprise 5,234 frames ranging between 254 and 858 frames per sequence.

We process the test sequences in real-time on a notebook PC with Intel Core i7-3610QM CPU (2.3 GHz) equipped with an NVIDIA GeForce GTX 675M GPU. Since our method does not process images at full 30 Hz image acquisition rate, it is required to skip frames. For assessing the segmentation, we compare segmentation accuracy of the direct RF maximum likelihood (ML) labeling with the ML labeling obtained by back-projecting the belief in the maps into the test images. Each pixel in the test image queries its corresponding node at maximum resolution in each key view. During SLAM, the image has been registered towards a reference key view. We require that the image pixel was visible in a key view and only consider those key views for which the corresponding node's resolution is equal or finer to the resolution in the reference key view. The belief for the pixel is then queried from this set of key views according to Eq. (15). We determine two kinds of labelings from the map: an instantaneous segmentation that is retrieved from the map in its current state when the test image is processed, and a final segmentation after the whole sequence has been mapped.

Table 1 RF parameters used in our experiments.

parameter	NYU Depth v2	AIS Large Objects
no. of trees	3	3
pixel samples per image	4537	2000
feature samples per node	5729	2000
threshold samples per node	20	50
max. offset radius (pixel m)	111	120
max. region size (pixel m)	3	10
max. tree depth	18	15
min. sample count in leaf	204	100
histogram bias ρ	0	0.2

Table 2 Run-time per frame on the NYU Depth v2 dataset in ms.

processing step	min	avg	max
image preprocessing	12.0	13.0	29.0
RF segmentation	32.0	44.4	67.0
SLAM	8.0	60.5	346.0
total	51.0	78.0	366.0

6.1 NYU Depth v2 Dataset

For the NYU Depth v2 dataset, we train RFs on average class accuracy for the four abstract structural object-classes. We optimize the hyper parameters of the RF, such as maximum tree depth, using the Hyperopt (Bergstra et al 2011) framework and 5-fold cross validation on the training set. Hyperopt performs informed search on the parameter space to efficiently find an optimal solution within the parameter range specified. Still, to optimize the RF in a feasible amount of time, rapid training is required. We therefore accelerate computationally expensive parts of RF training on GPUs. Our implementation is able to train and test about 350 trees per day on a single NVIDIA GeForce GTX TITAN. See Table 1 for resulting parameters. On this dataset, the distribution of pixels attributed to each object class is well-balanced, for which a setting of $\rho = 0$ is found through hyper-parameter optimization. While a region size of 3 appears to be small, most features that are selected by the RF are 3×3 regions.

6.1.1 Segmentation Accuracy

Table 3 shows average per-class, class, and pixel accuracy achieved on the test set. Example segmentations are illustrated in Fig. 6. Note that the NYU Depth v2 dataset provides a tool for in-filling missing depth readings that is too time-expensive for real-time processing, but has been used in related work on object-class segmentation (Silberman et al 2012; Couprie et al 2013). For comparison, we also show results of our RF segmentation method on in-filled depth images. Since we trained our RF method

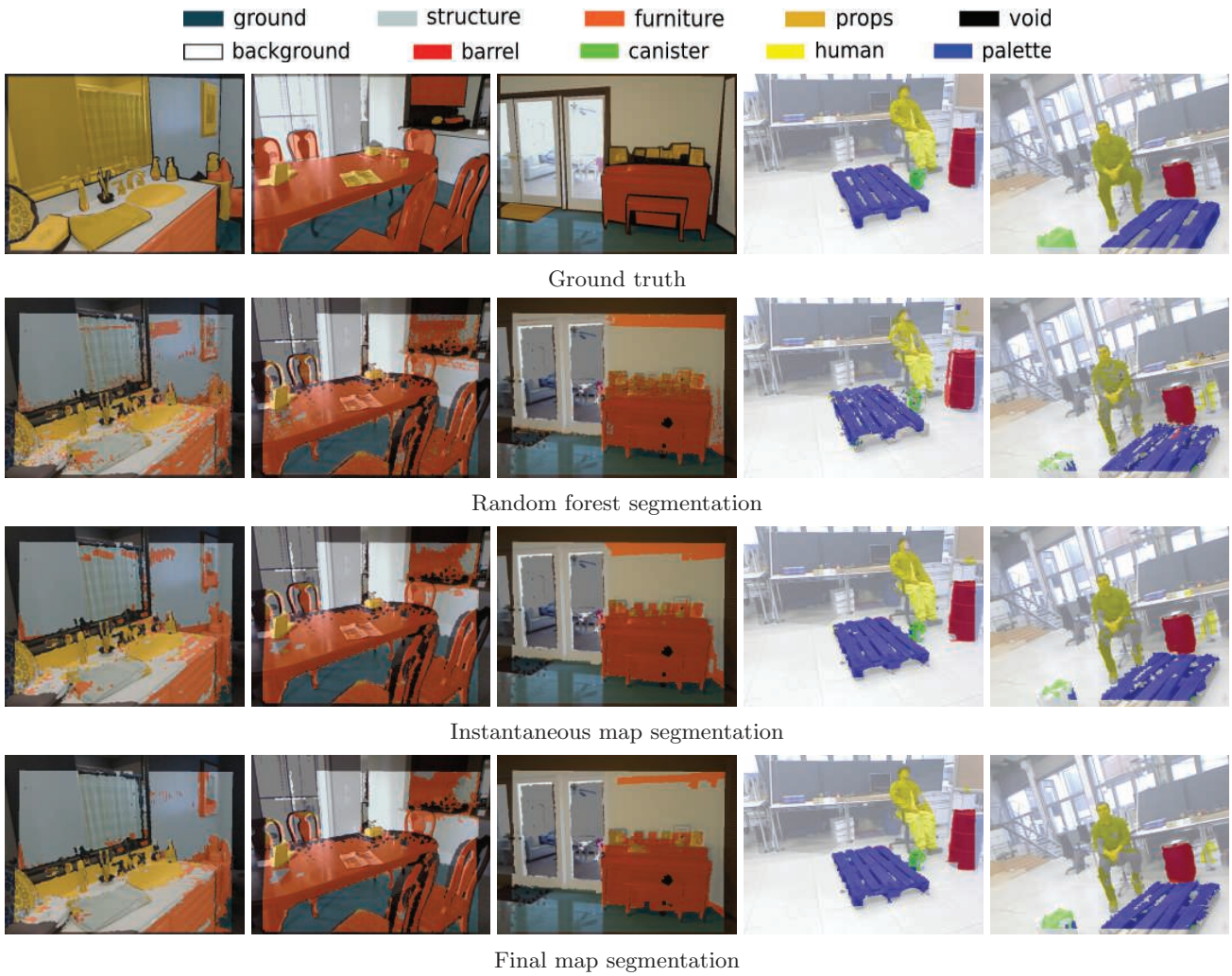

Fig. 6 Example labelings on the NYU Depth v2 (3 left) and the AIS Large Objects datasets (2 right).

Table 3 Segmentation accuracy on the NYU Depth v2 test set for 4 structural object-classes. (*) Comparison of segmentation results for in-filled depth images.

method	ground	structure	furniture	props	class avg.	pixel avg.
RF	93.7	80.0	69.4	**20.5**	65.7	68.4
instantaneous map	95.1	82.3	74.5	14.7	66.4	70.2
final map	**95.6**	**83.0**	**75.1**	14.2	**66.8**	**70.6**
*Silberman et al (2012)	68	59	70	**42**	59.6	58.6
*Couprie et al (2013)	87.3	**86.1**	45.3	35.5	63.5	64.5
*RF (ours)	**90.7**	81.4	**68.1**	19.8	**65.0**	**68.1**

on in-filled images, we fill-in the depth images during real-time experiments by constantly continuing depth from the right, the left, the top, and the bottom in the specified order. In-filling from the right first is motivated by the extrinsic setup of RGB and depth camera. Pixels without valid depth reading cannot be labeled in the 3D map. Hence, we discard them in the segmentation accuracy measure for the real-time experiments.

The results clearly demonstrate that our RF approach already achieves state-of-the-art performance on this dataset. The probabilistic fusion of the individual image segmentations into 3D further boosts segmentation performance by about 2.2% for pixel accuracy and ca. 1.1% for average class accuracy. The larger structural classes improve in segmentation accuracy, while the performance of the smallest object-class (*props*) is decreased

Table 4 Run-time per frame on the AIS Large Objects dataset in ms.

processing step	min	avg	max
image preprocessing	10.9	11.2	17.9
RF segmentation	30.4	33.0	42.9
SLAM	19.5	49.2	175.3
total	43.3	64.6	190.5

in the filtering process. The *props* class was already difficult to segment by our image-based RF approach. We attribute this to the fact that it has the most diversity in appearance, contains difficult objects, and is in parts inconsistently labeled. For instance, in bath room scenes, mirrors are labeled as *props*, which are difficult to distinguish from the reflected surface. Also, carpets on the floor are difficult to distinguish from the ground without considering the overall scene context. Our 3D fusion method reduces the segments to the consistently reoccuring parts in the RF segmentation. We note that few of the sequences could only be locally consistently mapped by our RGB-D SLAM approach, mainly due to the fact that only far distance measurements were available in frames or mostly a planar textureless region was visible.

6.1.2 Run-Time

Minimum, average, and maximum run-time per frame in milliseconds for individual processing steps and the overall semantic mapping pipeline are shown in Table 2. The average performance of semantic mapping is ca. 78 ms, i.e., 12.8 Hz. The largest part of the processing time is consumed by the SLAM method which is 60.5 ms on average. The time spent for the SLAM method strongly depends on the detail present in the image. If scenes are imaged from close distance, finer resolutions will be represented in the MRSMaps. If new spatial constraints need to be tested, a second image registration is performed which can further increase SLAM run-time to at most 346 ms. Nevertheless, the overall run-time of our approach has not been larger than 366 ms for the 231 test sequences. Note that the overall run-time is not a simple sum of the parts since object-class segmentation and SLAM run in parallel.

6.2 AIS Large Objects Dataset

Table 1 lists RF parameters used for the AIS Large Objects dataset. The dataset contains object classes of different sizes such as canisters, barrels, and palettes, while large parts of the scene are attributed to the background class. A histogram bias of $\rho = 0.2$ performs well on the dataset. The trained RF prefers large region sizes. In fact, most selected features have region sizes with 10 pixels width or height.

6.2.1 Segmentation Accuracy

This dataset has been trained and real-time processed without depth in-filling. From Table 5 we find that fusion into 3D strongly improves per-class accuracy as well as overall class and pixel accuracy.

6.2.2 Run-Time

In these test sequences, our semantic mapping achieved high frame-rates of about 15.5 Hz in average (64.6 ms). Similar to the NYU Depth v2 dataset, most processing time is spent for SLAM. The maximum overall run-time here is much less, since less close-by scenery has been recorded than in NYU Depth v2.

7 Conclusion

We proposed a novel real-time capable approach to semantic mapping. Our approach combines state-of-the-art object-class segmentation of RGB-D images with accurate RGB-D SLAM. Both methods have been implemented to perform real-time on GPU and CPU, respectively. They have been integrated into an online semantic mapping system.

Our object-class segmentation method is based on random decision forests (RF) and makes use of the dense depth available for scale-invariant recognition. The GPU implementation of our object-class segmentation approach reduces run-time by two order of magnitude, compared to a CPU implementation. Our SLAM method builds on multi-resolution surfel maps, a compact representation of RGB-D images that supports rapid 6-DoF registration. Using the camera pose estimates of SLAM, the probabilistic labelings of individual images by our RF approach are fused in multi-resolution voxel maps within a Bayesian framework.

In experiments, we demonstrate run-time efficiency and segmentation accuracy of our approach. We evaluated performance on two datasets with different qualities of object classes. The NYU Depth v2 dataset consists of 590 sequences recorded in indoor scenes, which we segment for structural object classes. Our approach outperforms state-of-the-art approaches to object-class segmentation on this massive dataset. Probabilistic fusion into 3D further increases segmentation accuracy. The whole processing pipeline operates online at approx. 12.8 Hz on average. The second dataset contains large objects that are segmented at good accuracy with our approach. It also performs real-time on these sequences at about 15.5 Hz on average. The semantic information made persistent in our maps could be used in many robotics applications such as object search and manipulation, exploration, or navigation.

Directions for further research include augmenting the RF classifier with concepts such as auto-context or

Table 5 Segmentation accuracy on the AIS Large Objects test set for 5 large object-classes.

method	background	barrel	canister	human	palette	class avg. (no bg)	pixel avg. (no bg)
RF	97.2	89.5	44.2	58.8	83.3	74.6 (55.2)	92.9 (73.8)
instantaneous map	97.8	93.9	46.5	**65.6**	88.1	78.4 (58.8)	94.4 (79.1)
final map	**98.0**	**94.0**	**47.5**	65.4	**88.9**	**78.8 (59.2)**	**94.6 (79.4)**

hierarchical segmentation. The accuracy and robustness of the underlying SLAM approach also influences segmentation accuracy. Semantic information could also be incorporated into SLAM to improve data association.

References

Bergstra J, Bardenet R, Bengio Y, Kégl B, et al (2011) Algorithms for hyper-parameter optimization. In: 25th Annual Conference on Neural Information Processing Systems (NIPS 2011), URL https://github.com/jaberg/hyperopt

Besl PJ, McKay ND (1992) A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 14(2):239–256

Breiman L (2001) Random forests. In: Machine Learning, pp 5–32

Castle RO, Klein G, Murray DW (2010) Combining monoSLAM with object recognition for scene augmentation using a wearable camera. Image Vision Computing 28(11):1548 – 1556

Civera J, Galvez-Lopez D, Riazuelo L, Tardos D, Montiel JMM (2011) Towards semantic SLAM using a monocular camera. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)

Couprie C, Farabet C, Najman L, LeCun Y (2013) Indoor semantic segmentation using depth information. The Computing Resource Repository (CoRR) abs/1301.3572

Endres F, Hess J, Engelhard N, Sturm J, Cremers D, Burgard W (2012) An evaluation of the RGB-D SLAM system. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)

Henry P, Krainin M, Herbst E, Ren X, Fox D (2012) RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. The International Journal of Robotics Research 31(5):647–663

Kuemmerle R, Grisetti G, Strasdat H, Konolige K, Burgard W (2011) G2o: A general framework for graph optimization. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), pp 3607–3613

Lai K, Bo L, Ren X, Fox D (2012) Detection-based object labeling in 3D scenes. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), pp 1330–1337

Meger D, Forssén PE, Lai K, Helmer S, McCann S, Southey T, Baumann M, Little JJ, Lowe DG (2008) Curious george: An attentive semantic robot. Robotics and Autonomous Systems 56(6):503–511

Newcombe RA, Izadi S, Hilliges O, Molyneaux D, Kim D, Davison AJ, Kohli P, Shotton J, Hodges S, Fitzgibbon A (2011) KinectFusion: real-time dense surface mapping and tracking. In: Proc. of the 10th Int. Symposium on Mixed and Augmented Reality (ISMAR), pp 127–136

Nüchter A, Hertzberg J (2008) Towards semantic maps for mobile robots. Robotics and Autonomous Systems 56(11):915–926

NVIDIA (2013) Parallel programming and computing platform — CUDA — NVIDIA. URL http://www.nvidia.com/object/cuda_home_new.html, visited on 2013-05-11

Ranganathan A, Dellaert F (2007) Semantic modeling of places using objects. In: Proc. of Robotics: Science and Systems

Salas-Moreno RF, Newcombe RA, Strasdat H, Kelly PHJ, Davison AJ (2013) Slam++: Simultaneous localisation and mapping at the level of objects. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)

Sengupta S, Greveson E, Shahrokni A, Torr P (2013) Semantic modelling of urban scenes. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)

Sharp T (2008) Implementing decision trees and forests on a GPU. Computer Vision–ECCV 2008 pp 595–608

Shotton J, Johnson M, Cipolla R (2008) Semantic texton forests for image categorization and segmentation. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)

Shotton J, Fitzgibbon A, Cook M, Sharp T, Finocchio M, Moore R, Kipman A, Blake A (2011) Real-time human pose recognition in parts from single depth images. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), IEEE, pp 1297–1304, DOI 10.1109/CVPR.2011.5995316

Silberman N, Hoiem D, Kohli P, Fergus R (2012) Indoor segmentation and support inference from RGBD images. In: ECCV

Stückler J, Behnke S (2010) Combining depth and color cues for scale- and viewpoint-invariant object segmentation and recognition using random forests. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)

Stückler J, Behnke S (2013) Multi-resolution surfel maps for efficient dense 3D modeling and tracking. In: Journal of Visual Communication and Image Representation

Stückler J, Biresev N, Behnke S (2012) Semantic mapping using object-class segmentation of RGB-D images. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)

Tomono M, Shin'ichi Y (2003) Object-based localization and mapping using loop constraints and geometric prior knowledge. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)

Vasudevan S, Gächter S, Nguyen V, Siegwart R (2007) Cognitive maps for mobile robots-an object based approach. Robotics and Autonomous Systems 55(5):359–371

Whelan T, Kaess M, Fallon M, Johannsson H, Leonard J, McDonald J (2012) Kintinuous: Spatially extended KinectFusion. In: RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras, Sydney, Australia

Zender H, Mozos OM, Jensfelt P, Kruijff GJ, Burgard W (2008) Conceptual spatial representations for indoor mobile robots. Robotics and Autonomous Systems 56(6):493 – 502