Hierarchical Sparse Coded Surface Models

Michael Ruhnke Liefeng Bo Dieter Fox Wolfram Burgard

***Abstract*— In this paper, we describe a novel approach to construct textured 3D environment models in a hierarchical fashion based on local surface patches. Compared to previous approaches, the hierarchy enables our method to represent the environment with differently sized surface patches. The reconstruction scheme starts at a coarse resolution with large patches and in an iterative fashion uses the reconstruction error to guide the decision as to whether the resolution should be refined. This leads to variable resolution models that represent areas with few variations at low resolution and areas with large variations at high resolution. In addition, we compactly describe local surface attributes via sparse coding based on an overcomplete dictionary. In this way, we additionally exploit similarities in structure and texture, which leads to compact models. We learn the dictionary directly from the input data and independently for every level in the hierarchy in an unsupervised fashion. Practical experiments with large-scale datasets demonstrate that our method compares favorably with two state-of-the-art techniques while being comparable in accuracy.**

I. Introduction

There is an increasing interest in accurate and textured 3D models of real-world environments or objects as they are relevant for various applications including surveillance, environmental monitoring and virtual reality. Also in the context of robotics many applications rely on accurate 3D models like navigation, object recognition, and mobile manipulation. Recently, dense 3D reconstruction with RGBD cameras has become popular due to the availability and low costs of such cameras. However, especially large-scale environments or high resolution models ask for compact representations of dense 3D models.

In this paper, we propose a hierarchical method to construct compact and textured 3D models of an environment from RGBD data. We describe local surface attributes with sparse codes that refer to an overcomplete dictionary. Sparse Coding [12] is a flexible and adaptive toolkit to compactly encode similarities in large data collections. A sparse code describes data with up to $n \in \mathbb{N}$ dictionary entries and we use it to encode local depth and texture data. In this way we exploit the redundancy in typical human made environments to build highly compact models. By doing this in a hierarchical fashion with local surface descriptions of different sizes and resolutions, we are able to capture similarities at different scales. This leads to even more compact models with high resolution in areas with high variation and low resolution in areas with minor variations. Fig. 1 shows an example obtained with our approach. The red surfaces in the left picture correspond to the largest surface patches and the blue surfaces to the smallest patches. The right picture shows the full model with RGB information.

Michael Ruhnke and Wolfram Burgard are with the Autonomous Systems Lab, University of Freiburg. Liefeng Bo is with the Intel Science and Technology Center for Pervasive Computing (ISTC-PC). Dieter Fox is with the Robotics and State Estimation Lab, University of Washington. This work was partially funded by the EC under ERC-AG-PE7-267686-LifeNav and FP7-610603-EUROPA2, by the German Research Foundation under grant number EXC 1086 and by the ISTC-PC.

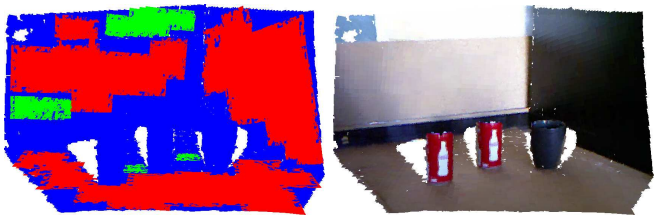

Fig. 1. This figure illustrates our hierarchical surface description scheme. The large areas in the left picture correspond to the regions that are represented at the highest level using the large patches. The green areas are represented with smaller patches at level two and the blue area is represented with small high resolution patches. The other picture shows the corresponding model with a size of 343 kB compared to 4.7 MB of the original point cloud.

We construct such hierarchical models in a greedy fashion by starting at the highest level of our hierarchy with the largest surface descriptions and use the standard deviation in depth and RGB-space to decide as to whether a local surface should be used at the current level. The idea is it to describe everything that cannot be accurately represented at the current level at a higher resolution. To guarantee coverage we describe all remaining data at the lowest level. Once we know the level of detail for all data, we apply K-SVD to learn a dictionary for every level in the hierarchy and calculate a sparse code for every local surface.

II. Related Work

There exists a wide variety of different representations for textured 3D data. Depending on the desired application the individual approaches either focus on accuracy, compactness, or rendering performance. Recent developments in simultaneous localization and mapping (SLAM) and the availability of RGBD-cameras made it possible to obtain large colored metric models. Whereas a major advantage of colored point-clouds lies in their accuracy, their drawback lies in their storage requirement, since we need to store every single data item in both 3D and RGB-space. Therefore, RGBD-based SLAM systems often use only a subset of features internally [7].

Most recent GPU driven RGBD-SLAM systems utilize the Truncated Signed Distance Function (TSDF) [2] as representation. The main advantages of this approach are the smoothness and the resolution of the estimated surface. However, this approach requires to maintain a cubic voxel

grid in memory, which restricts the size of such a grid to fit into GPU memory. Recent extensions use either a moving TSDF volume representation [17] and maintain data outside of the GPU or use local TSDF volumes or patches to represent only local aspects of the environment [6]. Since TSDF volumes have a limited compactness and require ray casting for visualization, the volumes are typically converted into meshes for that purpose. Meshes, however, are a highly accurate representation but lack compactness and furthermore cannot be easily updated.

Recently, Ma *et al.* [10] presented a planar simplification scheme for large meshes to drastically reduce the number of vertices on planar surfaces. In contrast to our work the quadtree-based triangulation scheme produces a mesh without overlap, while our method produces overlapping surfaces. On the other side it reduces only the structural information and does not exploit repetitive structures or textures. It furthermore is limited to planar surfaces.

Another compact volumetric representation for 3D data are octrees. Octrees represent occupancy and implicitly free space in a hierarchical grid structure. Fairfield *et al.* [4] used octrees in the context of SLAM for underwater vehicles. Recently Wurm *et al.* [18] introduced an open source implementation of octrees called Octomap which can store additional RGB information. In the context of compression Kammerl *et al.* [9] used octrees to compactly transmit point cloud streams by sending only the differences in successive frames. Especially for large voxel sizes, octrees are compact but suffer from discretization errors due to the alignment to the voxel structure. On the other hand, they are accurate for small voxel size models but not compact anymore.

In our previous work on Sparse Coded Surface Models (SCSM)[14] we introduced a surface-patch-based representation that uses Sparse Coding to compactly describe the surface attributes in both, 3D and RGB space. Sparse Coding [12] is a machine learning technique used for signal approximation [13], image denoising [8], [3] and for learning features [1], [19]. In this paper we present a hierarchical extension that better adapts to the level of detail needed to accurately describe local surfaces. This extension leads to even more compact models. Additionally we present a novel keypoint method driven by maximizing coverage and alignment to substantially reduce the number of patches needed to represent the input data. Furthermore, we introduce a distance-based weighting scheme to compute weighted means for the surface descriptions. This approach has been applied successfully in the past especially to TSDF-based SLAM methods [11], [17], [6]. As a result, the represented surfaces suffer less from noisy far range measurements of RGBD cameras.

III. Hierarchical Sparse Coded Surface Models

The main idea of our method is to represent colored 3D data with a set of surface patches and to encode every of these local surfaces using Sparse Coding based on a reference dictionary. In this way, we can exploit the repetition of structures and textures and build highly compact models.

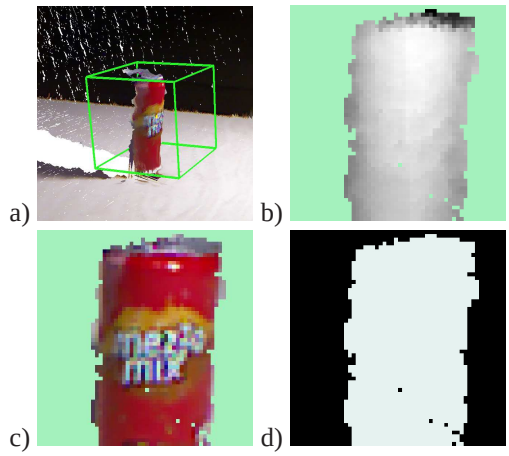

Fig. 2. This figure shows a colored point cloud with a green cube that will be represented by a patch (a), the depth description (b), the color description (c) and the bitmask (d). The light green areas in (b) and (c) correspond to undefined areas, which are marked as zeroes in the patch bitmask (d).

Note that we need more than one surface patch to describe a volume with this strategy. One major challenge in this context is that the scale at which structure or texture reappears can be arbitrary. Therefore it seems desirable to introduce surface patches of different sizes to capture similarities at different scales.

Similar to SCSM we encode local colored 3D data at a well-defined location as set of 2D images for three channels, which are depth, RGB, and bitmask (see Fig. 2). The grey value on the depth channel (b) corresponds to the weighted mean of the distance to the surface in direction of the normal. We compute a RGB value for the texture channel (c) as the weighted mean of the RGB values that fall into a pixel. Note that we can compute a local range image and a corresponding texture at every location in an unorganized point cloud. In contrast to SCSM, we use the error model described in [15] to weight the error according to the error of the maximum range of the sensor. The bitmask (d) encodes which pixel have a valid value. In this way, we can encode surfaces that are smaller than the given patch size and also deal with occlusions (see Fig. 2). In our current implementation we start with a maximum patch size and a minimum resolution at the highest level and halve the size in every patch dimension and double the resolution for every following level.

The main intuition of the hierarchical modeling is that we try to explain everything on the current level and use coverage and error distributions to guide the decision as to whether a surface is well represented or should be represented at a lower level and higher resolution. Starting with a pre-registered set of colored point clouds as input data we build a model using large patches with low resolution to represent the full data set. If a patch does not cover more than 90% we can easily describe it at the next lower level. The standard deviation ρ_c of a pixel in a patch gives us the information how good the patch resolution can represent the data at the current level of the hierarchy. We introduce a maximum standard variation ρ_{max}^{depth} and ρ_{max}^{rgb} for both channels to control how accurate a model will represent details. If the standard deviation of a pixel is above one of

Fig. 3. This figure illustrates the greedy-based hierarchical modeling scheme we apply. The left picture shows the input point cloud and the right picture shows all points that could not be accurately represented at the first level of the hierarchy. These points are the input for the next level.

these thresholds we set the bitmask of this pixel to 0 and postpone this part of the surface to be represented at the next level. Fig. 3 illustrates the procedure. The left picture shows the input point cloud and the right picture shows the point cloud of all remaining points after modeling the first level. The latter corresponds to the input for the next level. At the lowest level we accept all patches independent of the computed standard deviation to ensure that the model is complete and represents all data.

We define a Hierarchical Sparse Coded Surface Model (HSCSM) as $\mathcal{M}(\mathcal{H}, \mathcal{D}^{depth}, \mathcal{D}^{rgb}, \mathcal{I})$ with the hierarchy description $\mathcal{H} = \{h_1, \ldots, h_{|\mathcal{H}|}\}$ consisting of a tuple $h_j = \langle ps_j, pr_j \rangle$ for the j-th level which defines the patch size ps_j and patch resolution pr_j, reference dictionaries for both channels, $\mathcal{D}^{depth}$ and $\mathcal{D}^{rgb}$, and a scene description $\mathcal{I} = \{\mathbf{i}_1, \ldots, \mathbf{i}_{|\mathcal{I}|}\}$. The reference dictionary of every channel has entries that correspond to a particular level of the hierarchy $\mathcal{D}^{depth} = \{\mathcal{D}_1^{depth}, \ldots, \mathcal{D}_{|\mathcal{H}|}^{depth}\}$ that we construct by concatenating the learned dictionaries for every level. All entries $\mathbf{d}_k^{depth}$ of a depth dictionary $\mathcal{D}_j^{depth} = \{\mathbf{d}_1^{depth}, \ldots, \mathbf{d}_n^{depth}\}$ with $n = |\mathcal{D}_j^{depth}|$ have the same number of rows and columns as the depth channels of the surface patches of their corresponding level. Currently we us the same dimensions for all levels, but this is not mandatory. Every $\mathbf{i}_j = \langle T_j, \mathbf{c}_j^{depth}, \mathbf{c}_j^{rgb}, \mathbf{b}_j, level_j \rangle$ stores a transformation T_j, consisting of the 3D pose and the orientation for the surface patch, one sparse code for the depth $\mathbf{c}_j^{depth}$ and one for the RGB channel $\mathbf{c}_j^{rgb}$, a bitmask $\mathbf{b}_j$ that is 0 for undefined pixels and 1 for defined pixels and the index of its level level_j. In the following we will discuss how we compute the positions for our surface patches and will give an overview of the dictionary learning scheme we apply.

A. Surface Patch Locations

To fully represent a set of input point clouds $\mathcal{P}$ we need to find locations for the surface patches such that all data is covered by the patches. In general, finding the minimum number of subsets that contain all elements of $\mathcal{P}$ is a set covering problem known to be NP-complete [5]. For SCSM we used a spatial subsampling strategy to uniformly distribute the patch positions on the data as approximate solution. Since we have different patch sizes for every level of our hierarchy, we cannot follow the same strategy here. Therefore, we propose to solve this by applying a greedy algorithm that searches for the best patch locations at every level starting with the largest patches.

Possible ways to define the quality of patch locations are the area they cover, the error introduced by the patch discretization, and the error of the reconstruction based on our dictionary. Since we compute the dictionary in a later step, we could only use the reconstruction error by either introducing a two pass encoding, which results in better patch locations in the second pass, or using a general pre-computed dictionary for every patch size and resolution. Therefore, we rely only on coverage and discretization error. Since we reject pixels in the 2D projection that introduce high discretization errors, this also influences the area covered by a patch. Therefore, we sub-sample $\mathcal{P}$ to a resolution close to the resolution of a patch pixel $\mathcal{P}'$ and calculate the approximate coverage value for $\mathcal{P}'$ by computing surface patches and counting the number of valid pixels. We use the coverage value to guide our greedy selection of possible locations and start with the location that corresponds to the best coverage value. Once selected, we update the coverage of the neighborhood of a location according to the area the corresponding patch covers. To compute a coordinate frame of a patch we compute the normal of the range data in the patch space and chose x- and y-axes from the two global coordinate axes that have the larger angle to the computed normal and make them orthogonal to the normal. In this way, we constrain the cubical surface patches in their rotation around the normal according to a global coordinate frame. Given constrained orientations for surface patches we can extend the greedy search to first select a maximum coverage location and then search in the neighborhood along the defined x, y-directions for possible candidate locations which are well-aligned to our cubical patch structure. As a result of this alignment the average distance between neighboring surface patch locations is increased compared to distance-only based methods while still covering the same surface.

B. Dictionary Learning with wK-SVD

The extracted surface patches $\mathcal{S}$ contain a lot of redundant information on RGB and depth channels. Thus, we intend to compute a sparse approximation to find a compact representation of the data. Let S be the data matrix that contains every $\mathbf{s}_i$ as i-th column vector. The idea of K-SVD is to learn an overcomplete dictionary D and a sparse description X to approximate S. We can formulate this as a minimization problem using the following equation:

$$\min \|S - (W \odot (DX))\|_F^2 \quad \text{s.t.} \quad \forall i \, \|x_i\|_0 \leq k. \qquad (1)$$

Here, $\|A\|_F$ denotes the Frobenius norm, $\odot$ denotes the element-wise matrix multiplication, and k is the maximum number of nonzero entries for each column x_i that approximates a corresponding surface patch $\mathbf{s}_i \approx Dx_i$.

To deal with undefined values we use weighted K-SVD (wK-SVD) which applies a binary weighting matrix W that contains a 1 for data pixels that were actually observed and 0 otherwise. This information is represented in the bitmask channel of the patches. In this way we ignore the reconstruction results for undefined values and focus the reconstruction accuracy on the observed values. Undefined

pixels store a value of zero in S and by multiplying W in an element-wise fashion we ensure that the reconstructed values of undefined pixels are ignored during the optimization. A more detailed description of wK-SVD and a comparison to regular K-SVD can be found here [14] give.

To learn our reference dictionary we apply wK-SVD independently on every level for the depth channel and the RGB channel and concatenate the dictionaries per channel and update the indices of the sparse codes accordingly. Since the impact of errors on the resulting model scales with the size of a patch, we require a low reconstruction error for the higher levels. We start with a maximum number of allowed dictionary entries or the maximum number of data entries for the current level. Since wK-SVD reports back the number of unused entries, we crop the dictionary afterwards and proceed with the next level.

We can decode a model $\mathcal{M}(\mathcal{H}, \mathcal{D}^{depth}, \mathcal{D}^{rgb}, \mathcal{I})$ back into a point cloud if needed. This can be done by iterating through all elements of $\mathcal{I}$ and decoding the j-th element with

$$\hat{s_j}^{depth} = \mathcal{D}^{depth} \cdot \mathbf{c}_j^{depth}. \tag{2}$$

With $\mathbf{c}_j^{depth} = [\lambda_j^1, \ldots, \lambda_j^N]$ this can be rewritten as

$$\hat{s_j} = \lambda_j^1 \cdot \mathbf{d}_1^{depth} + \cdots + \lambda_j^N \cdot \mathbf{d}_N^{depth}. \tag{3}$$

Note that the sparse code $\mathbf{c}_j$ is a sparse vector with only $k \in \mathbb{N}$ nonzero entries. We apply the same scheme for decoding the color channel and project the joint information into a colored 3D point cloud according to the scale information for all values defined in $\mathbf{b}_j$. Finally, we apply T_j to put the patch point cloud at the right location in the resulting $\hat{\mathcal{P}}$.

IV. Experiments

The interesting quantities for our models are accuracy, compactness and the time needed to compute a model. As measure of compactness, we take the file size of the resulting models. As measure of accuracy, we compute the Root Mean Squared Error (RMSE) between the input point clouds $\mathcal{P}$ and the point cloud reconstruction of our model $\hat{\mathcal{P}}$. Obviously, this depends on the sensor noise and the discretization scheme we apply in our models. Still, the order of magnitude of the error is a good indicator for the accuracy. We compute the error for every point in $\hat{\mathcal{P}}$ by searching for the nearest neighbor in $\mathcal{P}$ and vice-versa. In this way, the error measures inliers and outliers.

A. Influence of Maximum Standard Deviation

Crucial questions are how much we gain from introducing multiple levels with different patch sizes and if the maximum standard deviation is a good criterion to guide our decision what to model on a certain level. Therefore, we conducted a corresponding experiment on the small RGB-D frame shown in Fig. 3. We chose to use 4 levels with patch sizes of $24/12/6/3$ cm and corresponding resolutions of $24/12/6/3$ mm. In the first setting, we varied the maximum allowed standard deviation of the depth channel ρ_{max}^{depth} for a fixed maximum standard deviation of the RGB channel $\rho_{max}^{rgb} = 100$. In the second setting, we used a fix $\rho_{max}^{depth} = 1.0$ and varied ρ_{max}^{rgb}. We chose large values for ρ_{max} to reduce the impact of the dependency between the two parameters. Fig. 5 shows the resulting plots. The left plot shows the area covered for every level of the hierarchy and the resulting RMSE for the corresponding model. For small ρ_{max}^{depth} the models are fully covered with small patches of the lowest level. By increasing ρ_{max}^{depth} the contribution in coverage is shifted to higher levels. This drastically reduces the number of patches needed to describe the surface and results in more compact models. The right plot shows a similar evaluation for ρ_{max}^{rgb}. Again, the area covered with larger patches of higher levels increases while relaxing the parameter. We highlighted the most interesting parameter ranges with a light gray box. In this parameter range the file size is reduced around 25% while only moderately increasing the error.

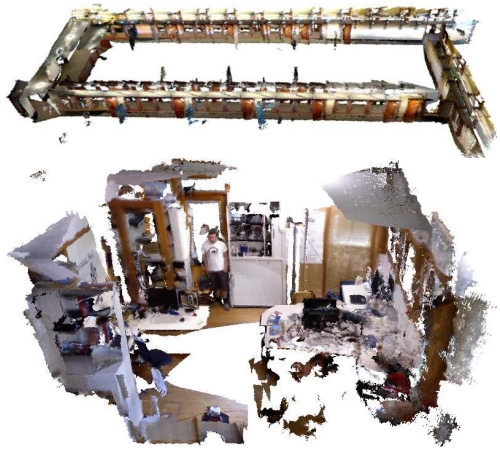

Fig. 4. This Figure shows the data sets used for comparison with Octomap and Sparse Coded Surface Models. The top picture shows an overview of a RGBD data set acquired in a typical corridor environment. The second picture shows the publicly available fr1/room data set.

B. Comparison to Octomap and SCSM

In this section, we compare our proposed method to Octomap [18] and the non-hierarchical SCSM. Therefore we applied our method on three different data sets, the example scene illustrated in Fig. 3, a SLAM solution[1] and the publicly available fr1/room data set [16], both shown in Fig. 4. Table I gives an overview of the relevant statistics for Octomap, SCSM and the proposed method. Note that we chose the minimum resolution according to the average distance between sensor and surfaces in the data set and to avoid over-fitting to sensor noise. On the three data sets, our method never extracted patches on level 4 or higher. Therefore, we provide timings for the larger data sets only with 3 levels.

For the corridor SLAM data set, we applied our method to the accumulated point cloud and built a model with 35,824 surface patches on three hierarchy levels (251 / 801 / $34,772$). As can be seen, the resulting models have no patches on the highest level. The full process of creating the model took 19 min including the dictionary learning with a dictionary size of 1,000 for the depth channel and $3,500$ for

[1]Courtesy of Peter Henry

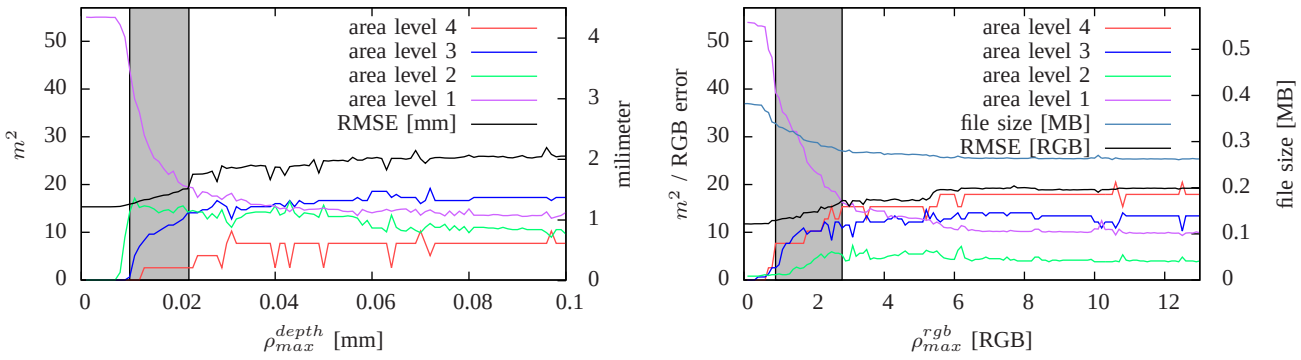

Fig. 5. Impact of the maximum standard deviation parameters ρ_{max}^{depth} and ρ_{max}^{rgb} on our hierarchical model. The left plot shows the area covered for every level of the hierarchy and the resulting RMSE. For small values of ρ_{max}^{depth} our model is fully covered with small patches of the lowest level. By increasing ρ_{max}^{depth} we shift the contribution in coverage to higher levels. This reduces the number of patches needed to describe the surface. The right plot shows a similar evaluation for ρ_{max}^{rgb}. Again, the area covered with larger patches at higher levels increases while relaxing this parameter. The most interesting parameter ranges are marked with light gray boxes. In this area the file size is reduced by 25% with only a moderate increase in error.

Data set	method	dict. (D/RGB)	patch size	res.(cm)	input	result	RMSE (D/RGB)	time
Scene Fig. 3	Octomap	- / -	-	0.3	4.7 MB	918 kB	0.0014 m / 8.2	0.25 s
Scene Fig. 3	SCSM	100 / 200	0.03 m	0.3	4.7 MB	423.9 kB	0.0016 m / 14.3	7 s
Scene Fig. 3	HSCSM	100 / 200	0.03 m	2.4 / 1.2 / 0.6 / 0.3	4.7 MB	343.6 kB	0.0017 m / 15.2	8 s
RGBD Corridor	Octomap	- / -	-	2	3.35 GB	44.5 MB	0.016 m / 25.1	56 s
RGBD Corridor	SCSM	100 / 500	0.2 m	2	3.35 GB	10.2 MB	0.017 m / 19.9	8 min
RGBD Corridor	HSCSM	1000 / 3000	0.8 / 0.4 / 0.2 m	8 / 4 / 2	3.35 GB	7.8 MB	0.013 m / 21.7	19 min
fr1/room	Octomap	- / -	-	1	2.7 GB	45 MB	0.006 m / 39.6	2 min
fr1/room	SCSM	500 / 3500	0.05 m	1	2.7 GB	8.1 MB	0.005 m / 29.9	31 min
fr1/room	HSCSM	500 / 3500	0.2 / 0.1 / 0.05 m	4 / 2 / 1	2.7 GB	7.2 MB	0.005 m / 30.0	36 min

TABLE I

EXPERIMENTAL EVALUATION FOR EACH DATA SET AND METHOD. THE DICTIONARY SIZE AND RMSE ERRORS ARE SPLIT INTO DEPTH AND RGB. WE MEASURED THE TIMINGS ON A STANDARD DESKTOP CPU WITH 3 GHZ.

the RGB channel. The model created with our hierarchical method outperforms Octomap in terms of accuracy and also visually as can be seen in Fig. 6 (b) and (d). Regarding runtime, Octomap is fastest with less than a minute but introduces a higher error in both depth and RGB space. Compared to SCSM, the hierarchical model is 24% more compact and more accurate in the depth channel. The error in RGB space is slightly higher. This is due to the fact that encoding errors in larger patches have a bigger impact on the overall error calculation. Therefore we had to increase the dictionary sizes compared to the non hierarchical method.

The fr1/room data set was captured in a cluttered office environment and is a challenging benchmark for our method. Clutter introduces a high entropy and makes it harder to find similar structures. Fig. 7 shows an example view for the raw point cloud, Octomap, SCSM and our method. Again, our method outperforms Octomap in terms of accuracy and compactness, while it requires more computation time. The model also looks smoother while being 84% smaller in size. In comparison to SCSM we gain 9% in terms of compactness while introducing a slightly higher error in RGB. In general, the gain in compactness is highly influenced by the amount of variations in structure and texture. In the worst case the full surface needs to be represented at the lowest level so that the resulting model performs similar to SCSM.

V. CONCLUSIONS

In this paper, we presented a novel approach to construct textured 3D models using a hierarchy of local surface patches. Our method employs differently sized surface patches thus leading to more compact models. Furthermore, our method exploits similarities in structure and texture by using Sparse Coding to describe surface attributes. It learns the required dictionaries during the model creation process in an unsupervised fashion. Practical experiments carried out with data sets of different scale demonstrate that our method compares favorably with two state-of-the-art techniques.

REFERENCES

[1] L. Bo, X. Ren, and D. Fox, "Unsupervised feature learning for rgb-d based object recognition," *Proc. of the Int. Symposium on Experimental Robotic(ISER)*, 2012.

[2] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 303–312.

[3] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[4] N. Fairfield, G. A. Kantor, and D. Wettergreen, "Real-time SLAM with octree evidence grids for exploration in underwater tunnels," *Journal of Field Robotics*, 2007.

[5] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM (JACM)*, vol. 45, no. 4, pp. 634–652, 1998.

[6] P. Henry, D. Fox, A. Bhowmik, and R. Mongia, "Patch volumes: Segmentation-based consistent mapping with RGB-D cameras," in *International Conference on 3D Vision (3DV)*, 2013.

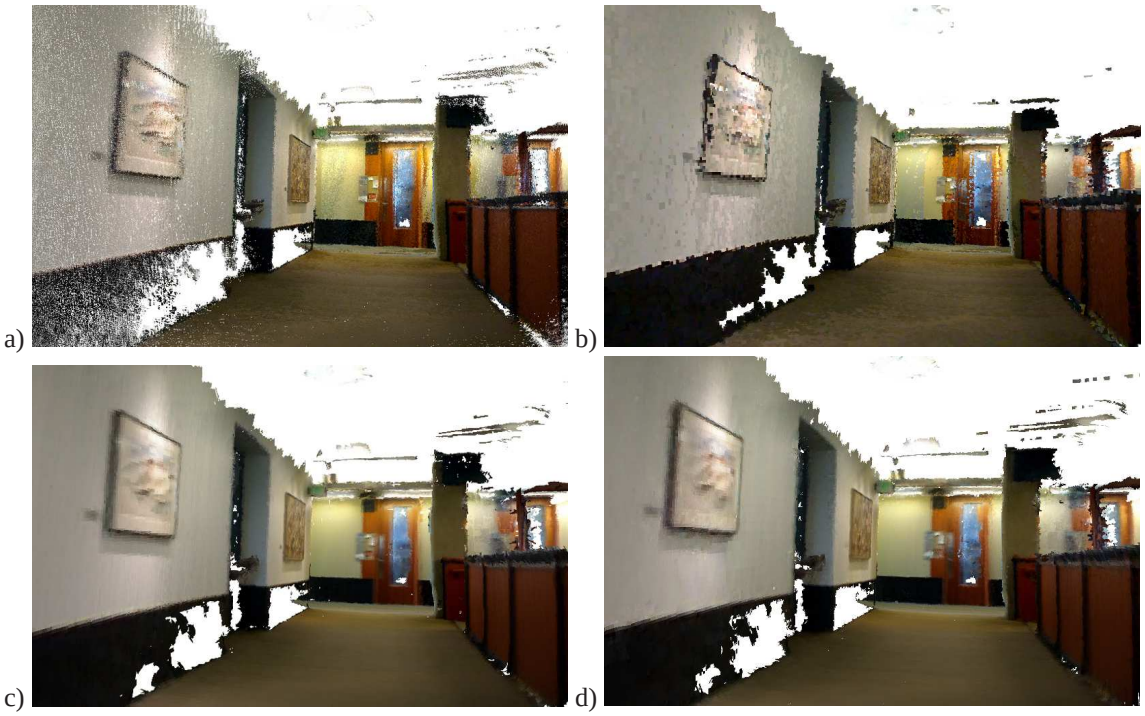

Fig. 6. Resulting model for the RGB Corridor data set, a representative part of the model as a raw point cloud (a), an octomap (b), SCSM (c) and the hierarchical variant we propose (d). The Octomap occupancy grid has a file size of 44.49 MB with a voxel size of 2 cm, an RMSE of 0.016 m on the depth channel and an RMSE of 25.1 on the RGB channel.The model without hierarchy has a file size of 10.2 MB and stores a depth dictionary with 100 entries and a RGB dictionary with 500 entries. The RMSE is 0.017 m for the depth channel and 19.9 for the RGB channel. The hierarchical model has an RMSE of 0.013 m for the depth and of 21.7 for the RGB channel. In comparison, the hierarchical model has the lowest error in the depth data and an error in the RGB data that is between the octomap and Sparse Coded Surface Model errors. Organizing the data with the hierarchical model reduces the required storage from 10.2 MB to 7.8 MB.

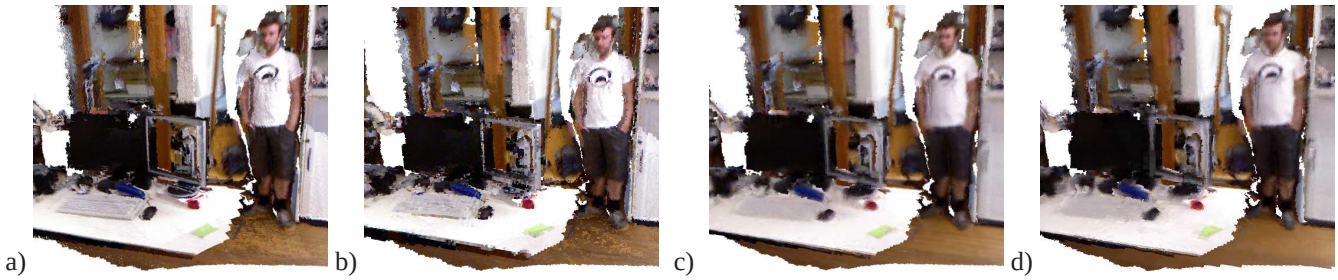

Fig. 7. Resulting model for the fr1/room data set, a representative part of the model as raw point cloud (a), an Octomap (b), SCSM (c) and HSCSM (d).

[7] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3d modeling of indoor environments," in *Proc. of the Int. Symposium on Experimental Robotic(ISER)*, vol. 20, 2010, pp. 22–25.

[8] A. Hyvarinen, P. Hoyer, and E. Oja, "Image denoising by sparse code shrinkage," *Intelligent Signal Processing*, pp. 554–568, 2001.

[9] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Minnesota, USA, May 2012.

[10] L. Ma, T. Whelan, E. Bondarev, P. H. N. de With, and J. McDonald, "Planar simplification and texturing of dense point cloud maps," in *Proc. of the European Conference on Mobile Robots (ECMR)*, Barcelona, Spain, September 2013.

[11] R. Newcombe, A. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *IEEE International Symposium Mixed and Augmented Reality (ISMAR)*, 2011.

[12] B. Olshausen, D. Field, *et al.*, "Sparse coding with an overcomplete basis set: A strategy employed by vi?" *Vision research*, vol. 37, no. 23, pp. 3311–3326, 1997.

[13] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1553–1564, 2010.

[14] M. Ruhnke, L. Bo, D. Fox, and W. Burgard, "Compact RGBD surface models based on sparse coding," in *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2013.

[15] M. Ruhnke, R. Kümmerle, G. Grisetti, and W. Burgard, "Highly accurate 3d surface models by sparse surface adjustment," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2012.

[16] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[17] T. Whelan, M. Kaess, J. Leonard, and J. McDonald, "Deformation-based loop closure for large scale dense RGB-D SLAM," in *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013, (to appear).

[18] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems," in *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, vol. 2, 2010.

[19] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 1794–1801.