Vision, Modeling, and Visualization (2021)
B. Andres, M. Campen, and M. Sedlmair (Eds.)

SuBloNet: Sparse Super Block Networks for Large Scale Volumetric Fusion

Darius Rückert[1] and Marc Stamminger[1]

[1]Visual Computing Lab, University of Erlangen Nurenberg, Germany

Abstract

Training and inference of convolutional neural networks (CNNs) on truncated signed distance fields (TSDFs) is a challenging task. Large parts of the scene are usually empty, which makes dense implementations inefficient in terms of memory consumption and compute throughput. However, due to the truncation distance, non-zero values are grouped around the surface creating small dense blocks inside the large empty space. We show that this structure can be exploited by storing the TSDF in a block sparse tensor and then decomposing it into rectilinear super blocks. A super block is a dense 3d cuboid of variable size and can be processed by conventional CNNs. We analyze the rectilinear decomposition and present a formulation for computing the bandwidth-optimal solution given a specific network architecture. However, this solution is NP-complete, therefore we also a present a heuristic approach for fast training and inference tasks. We verify the effectiveness of SuBloNet and report a speedup of 4x towards dense implementations and 1.7x towards state-of-the-art sparse implementations. Using the super block architecture, we show that recurrent volumetric fusion is now possible on large scale scenes. Such a systems is able to reconstruct high-quality surfaces from few noisy depth images.
(see https://www.acm.org/publications/class-2012)

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Reconstruction; Mixed / augmented reality;*** *3D imaging;*

1. INTRODUCTION

Deep learning has achieved major breakthroughs in various image processing tasks. However, a direct application of 2d models to 3d problems is often not feasibly due to increased memory consumption and compute requirements. The underlying sparsity of 3d input data has driven researchers to develop architectures that operate directly on point clouds or indirectly on sparse voxel grids. This significantly improves 3d classification tasks, for example, a pedestrian detection in LiDAR data of a self-driving vehicle. In some fields, such as 3d reconstruction, sparse architectures have not been explored yet because the underlying truncated singed distance field (TSDF) requires many non-zero samples around the surface. The additional overhead of sparse convolutions outweighs the increased memory consumption of traditional dense neural networks resulting in similar or worse performance for moderately sized scenes.

In this work, we present SuBloNet, a novel architecture designed for efficiently processing block sparse features arising in real-time TSDF-based reconstruction. As a first step, we search *super blocks* in the input data, which are large rectangular non-zero blocks of features. These super blocks are extracted and passed through a fully convolutional neural network and then copied back to its original block sparse structure. We can therefore leverage the high efficiency of dense volumetric CNN implementations while also supporting large scale scenes. We validate our approach by comparing it to state-of-the-start methods showing a speedup of 1.7x on room-scale indoor environments. Using our method we also present a novel incremental depth fusion approach that is able to reconstruct clean surface from a stream of noisy depth images in real time.

In summary, the contributions of our work are:

- A novel neural network architecture for processing block sparse input data
- An efficient algorithm for decomposing block sparse tensors into super blocks
- An incremental depth fusion approach that is able to reconstruct large scenes in high resolution

2. RELATED WORK

Sparse tensor architectures for deep learning have been explored extensively in the recent years. We usually differentiate between weight sparsity, activation sparsity, and feature sparsity. These domains require different implementation techniques and are used for different applications.

Sparsity in the weight domain is the most explored form and builds on the idea of removing weights close to zero [LDS*89]. It has been shown that this approach significantly reduces model size [HPTD15] and energy consumption [YCS17] with only a small hit to accuracy. However, expected run-time improvements are often only theoretical because consumer hardware is optimized for dense filter operations [NUD17]. Special hardware has then

DOI: 10.2312/vmv.20211375

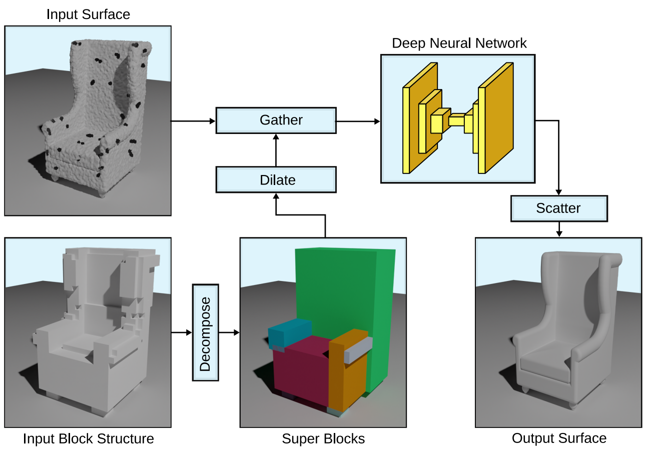

Figure 1: *Overview of our super block processing pipeline. The block sparse TSDF (left) is decomposed into super blocks. These blocks are dilated by the neural network's receptive field size and collected by a gather operation. After that, every block is processed by the network, and scattered back into the original structure. This concept can be used for all problems with block sparse input data.*

been introduced to process the sparse models with impressive performance gains over CPUs and GPUs [HLM*16, PRM*17]. To avoid the need of custom hardware, recent work focuses on structured pruning, which involves removing groups of weights instead of individual edges [SCYE17, AHS17, WWW*16]. In the general case, this results in a block-sparse weight matrix, which improves computational efficiency of linear layers [GRK17], convolutional layers [MHP*17, WGH*21, VK*19], and recurrent neural networks [NUD17]. Further constraining block-sparsity to a single dimension, for example, channel pruning [LLS*17, HZS17], neuron pruning [HPTT16], and filter pruning [LKD*16, LWL17, HLW*19, IRCC17], allows the reduced models to be executed using the highly optimized dense matrix operations. Other approaches of reducing the number of weights exploit the sparsity of the ReLu activation function [SC17, DHYY17], apply quantization [ZYG*17, CEKL16], or approximate the weight matrix using a low rank decomposition [LWF*15, DZB*14, LGR*14, KPY*15].

Sparsity in the feature domain is significant for applications where only a fraction of the input space contains relevant data. The sparsity is exploited by either running dense neural networks on partial point clouds [QSMG17, QYSG17] or using special convolution operations on sparse tensors [GvdM17, CGS19, TLZ*20, TQD*19]. The latter approach has proven to be more successful on common segmentation and classification tasks, though the sparse convolutions introduce additional overhead compared to dense operations. If the input consist of dense non zero blocks inside large empty space, the block sparse approach of SBNet [RPYU18] outperforms the more general sparse networks. They subdivide the input space into a uniform grid of blocks and define a binary mask that captures the non-zero pattern. Convolutions are then only applied on the selected blocks using a gather, convolve, scatter approach. Our method, which is presented in Section 3, also processes block sparse input data. However, we further combine neighboring non-zero blocks and store the sparsity in a coordinate format instead of mask. This reduces memory consumption, as well as, increases performance if the input contains large dense regions.

Rectilinear decomposition is the task of finding a set of rectangles that covers a rectilinear polygon [SHIF12]. This will be used in our work to increase neural network performance by partitioning block sparse tensors into super blocks. In previous work, many different 2d decomposition algorithms have been proposed. They range from quadtree decompositions [WHL01] over heuristic approximations [NS88] to the optimal graph-based solution [Fra86, SG93, AKR99, SHF12]. These algorithms have been successfully used in various applications, for example, binary image compression [MF95], VLSI mask fabrication [LTL89], and optimizing two dimensional databases [LLL*79]. In the three dimensional case, only few researchers have approached this problem [Jai02, HIF19] because computing the optimal decomposition is known to be NP-hard [DK91].

In the application domain of volumetric fusion from depth images, the seminal work of Curless et al. [CL96] is now used in many real time reconstruction systems. KinectFusion [IKH*11] incrementally constructs a truncated signed distance field (TSDF) on a uniform grid while tracking the RGB-D camera. This was later improved by implementing a block sparse architecture to store the TSDF values only around the surface [NZIS13, DNZ*17, PKG*17, RMO*19]. Other approaches, which focus on high-quality offline reconstruction, use hierarchical tree-based structures to capture variable scale [FG11, SKC13]. Recent advancements in deep learning also show good results for surface completion, filtering and reconstruction [DRB*18]. Various approaches have been explored that make use of a uniform TSDF grid [MvAB*20], octrees [ROUG17, RUBG17], or an implicit surface representation [PFS*19, CZ19, MON*19]. The recent method RoutedFusion [WSPO20] uses deep neural networks to integrate depth images into a uniform TSDF of fixed size. Their pipeline consists of a routing stage, which preprocesses the input, and a fusion stage, which predicts the new TSDF values. Our approach (see Section 3.3) improves on this concept by fusing the depth images in latent space. We also use the novel super block architecture to improve the efficiency and enable interactive reconstruction of large scale scenes.

3. SUPER BLOCK ARCHITECTURE

The most common data structure for real-time 3d reconstruction from depth images is a truncated signed distance field (TSDF). Each voxel stores the signed distance to the surface up to a threshold t_d. Voxels with a distance larger than t_d are not relevant and can be discarded. It has been shown, that a sparse structure of voxel blocks can reduce memory consumption significantly and allows real-time operation on large-scale scenes. This structure consists of N blocks of d^3 voxels, in our case $d = 8$, with C channels each and is stored using a data tensor D and a structure tensor S.

The goal of this work is to build a neural network architecture that is able to efficiently process block sparse TSDFs. A high-level overview of our pipeline is shown in Figure 1. The first step is to decompose the structure S into a list of rectilinear super blocks, which are 3-dimensional cuboids of arbitrary size. For example, a super block with dimensions $3 \times 3 \times 3$ contains 27 blocks and $27 \cdot d^3$ voxels. We then dilate each super block by the receptive field size of the neural network and extract each cuboid by a gather operation. The extracted super blocks are passed through the dense network and scattered back into the original sparse structure. Since we have to add the network's receptive field, it is sometimes more efficient to create fewer overlapping blocks than more disjunct blocks. For example, the blue armrest in Figure 1 intersects the red seat and the green backrest covers multiple empty cells.

In the following section, we present the bandwidth-optimal decomposition for a given receptive field size. After that, we derive an approximation algorithm, which is able to produce a high-quality solution in polynomial time.

3.1. Optimal Rectilinear Cover for CNNs

Every binary three dimensional discrete shape S can be decomposed into $K \geq 1$ rectilinear cuboids $\beta = \{B_1, B_2, \ldots B_K\}$ where each cuboid is defined by the two opposing vertices $B_i = (a, b)$ with $a, b \in \mathbb{Z}^3$. We call the decomposition a *bijective cover* if the partitions B_i do not overlap and the union of all cuboids is identical to S:

$$\begin{aligned} B_i \cap B_j &= \emptyset \quad \forall i, j \\ S &= \bigcup_{k=1}^{K} B_k \end{aligned} \tag{1}$$

Due to the fact that processing empty and redundant cells does not change the final result of a neural network, a bijective cover is not required in our case. It is enough to ensure that the input structure S is a subset of the decomposition.

$$S \subseteq \bigcup_{k=1}^{K} B_k \tag{2}$$

Applying a CNN to the cuboid list β requires each element of β to be dilated by the CNN's receptive field radius r. Since we operate on a block granularity of 8^3 voxels, a CNN with four $5 \times 5 \times 5$ convolutional layers has a receptive field radius of $r = (4 \cdot 2)/8 = 1$ block. The memory consumption after gathering the super blocks V depends on r and can be defined using the *Dilate* function which expands every block by the receptive field radius.

$$\begin{aligned} V_r(B) &= \text{Volume}(\text{Dilate}(B, r)) \\ \text{Volume}(B) &= (b_x - a_x) \cdot (b_y - a_y) \cdot (b_z - a_z) \\ \text{Dilate}(B, r) &= (a - r, b + r) \end{aligned} \tag{3}$$

Figure 2 shows a 2-dimensional shape, which is decomposed in two different ways. The decomposition A (middle image) is bijective. Hence V_0 is identical to the input volume. The decomposition β (right image) consists of a single rectangle which conceals the hole in the middle. It is not bijective, because empty space is occluded. However, dilating each rectangle by $r = 1$ shows that the latter decomposition requires around 25% less memory.

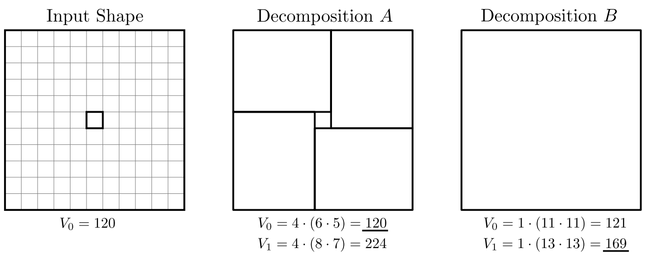
Input Shape
Decomposition A
Decomposition B
$V_0 = 120$
$V_0 = 4 \cdot (6 \cdot 5) = \underline{120}$
$V_1 = 4 \cdot (8 \cdot 7) = 224$
$V_0 = 1 \cdot (11 \cdot 11) = 121$
$V_1 = 1 \cdot (13 \cdot 13) = \underline{169}$

Figure 2: *A rectilinear input shape is decomposed into a bijective cover A and the V_1-optimal decomposition B.*

We will now use the volume formulation of Eq. (3) to define the optimal decomposition in terms of memory consumption for a receptive field radius r.

$$\underset{\beta}{\text{argmin}} \sum_{B \in \beta} \varepsilon + V_r(B) \tag{4}$$

If we are able to solve (4) w.r.t. the constraint (2), we obtain a list of super blocks β which require the least amount of memory after dilation. The constant ε is introduced to favor decompositions with less super blocks if two solutions have the same volume. For a better control over the result, we extend Eq. (4) by computing the volume for multiple radii $V_0, V_1, \ldots$ and scale them by a scalar weight vector $w = (w_0, w_1, ..)$.

$$\begin{aligned} &\underset{\beta}{\text{argmin}} \sum_{B \in \beta} C(B) \\ &C(B) = \varepsilon + \sum_j w_j V_j(B) \end{aligned} \tag{5}$$

A recurrent neural network, for example, benefits from non-overlapping blocks in the training stage because less gradient memory is required. Therefore we can manually adjust the decomposition to use a weight of $w = (4, 2, 1)$ during training and a weight $w = (0, 0, 1)$ during inference. For standard feed-forward networks, we have found that few large blocks are preferred. Hence we default to a zero weight vector w with only the r-th element set to one.

Unfortunately, it has been shown by reduction that computing the bijective solution $w = (1)$ is NP-complete for non-convex input shapes [DK91]. Our less restrictive problem defined in Eq. (5) is at least as hard to solve because the integer search space is further enlarged. In the next section, we present an approximation algorithm to compute a non-optimal solution in polynomial time.

3.2. Approximate Decomposition Algorithm

Starting from a list of 3d integer coordinates the goal is to find a decomposition β that minimizes (5) under the constraint (2). Our algorithm is visualized in Figure 3 and will be explained in the following section.

We start the decomposition process by constructing an octree on the sparse input [Mea82]. For every element, the 64-bit Morton Code is computed and the list is sorted according to this value [Mor66]. Then, the tree is assembled in a bottom-up fashion by combining neighboring elements with identical binary substrings

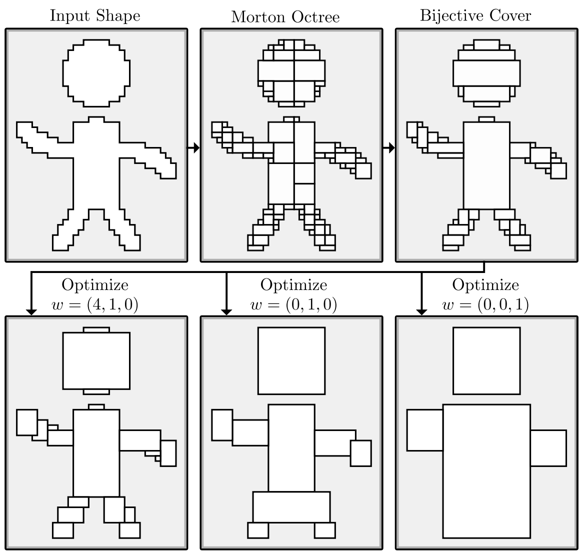

Figure 3: *Overview of our rectilinear decomposition algorithm. The input shape is decomposed into a bijective cover, which is then optimized by a heuristic merging algorithm that reduces the weighted volume cost (Eq. (5)). In the bottom row, the locally optimal cover is shown for a different sets of weights.*

[LGS*09]. It has been demonstrated that this method can be parallelized and shows good results in fast construction of bounding volume hierarchies (BVH) [Kar12].

After the initial octree-based solution, we seek to refine the current decomposition according to Eq. (5). First, we construct a BVH on the initial decomposition. Using this BVH, we can efficiently compute a neighbor list for every rectangle by detecting all elements with a distance of less than 1 voxel. For every neighbor pair (B_i, B_j), we compute the current cost $C_c = C(B_i) + C(B_j)$ and the cost of the smallest cuboid that contains both of them $C_m = C(B_i \cup B_j)$. Additionally, we check if $B_i \cup B_j$ overlaps a boundary slice of the other neighbors. We erase this slice and subtract the difference from C_m. If the total cost is reduced $(C_m < C_c)$, the merge is executed and the BVH is updated. The neighbor list is rebuild for all elements that have been changed. We stop the merged-based optimization if the decomposition has converged or a maximum number of iterations is reached.

In our experiments, we have found that directly merging the octree cuboids based on (5) results in a non-optimal decomposition because the error accumulates over multiple merges. Therefore we improve the initial guess by separating the merging pass into two stages. The first stage only accepts operations that keep the volume V_0 constant. The second stage then uses the user-defined cost function to refine the result. Figure 3 shows an example shape that is decompose using our algorithm. In the first row, we compute the octree and merge matching neighbors to get the bijective cover. After that we execute the second merging pass resulting in different solutions based on the weight vector w.

3.3. Volumetric Fusion Model

In the previous sections, we have presented the idea of super block decomposition for processing block sparse TSDFs. We will now use this architecture to build an incremental 3d reconstruction system, which fuses a stream of depth images. An overview of our pipeline is shown in Figure 4. First, the depth images are converted to a local TSDF I and are integrated into a latent-space voxel grid L using a recurrent fusion network. After the integration of all images, we pass L through a second reconstruction network to extract the fused surface from the neural features. We use our super block architecture to improve the efficiency and memory consumption of the fusion and reconstruction stage.

The input depth map is converted to a block sparse TSDF of 8^3 voxel blocks by projecting the point cloud into the grid and setting each voxel to the local signed surface distance. We then compute a rectilinear decomposition on the input structure and gather the respective super blocks from the input I and latent space L. If I contains previously unseen voxels, the gather operation sets the respective blocks of L to zero. The extracted blocks of I and L are then concatenated along the channel dimension and processed by the fusion network. This network is a three layer 3d U-net [RFB15] with a receptive field size of 32 voxels. The U-net outputs are fused super blocks which are written into the latent space data structure by a scatter operation. If a block already exists in latent space, it is overwritten by the novel block to guide the network to learn a reasonable fusion function. After any number of depth images have been fused into the latent space, the user can issue a surface reconstruction. For this task, we decompose the structure of L into super blocks, gather them, and process them by a 3d reconstruction U-net. The output is a block sparse TSDF with a confidence value for every element, which can be converted to a mesh by ISO-surface extraction [LC87]. Additional outputs are also possible, for example, classified labels, surface gradients, and colors.

4. EXPERIMENTS

We have conducted two experiments to verify the effectiveness of our method. First, we compare the incremental fusion pipeline to other methods and show that our approach outperforms the current state-of-the-art. Then, we evaluate the performance of the underlying super block architecture by measuring training time on a room-scale block sparse TSDF (see Section 4.2). Finally, we discuss the results and show current limitations of our work (Section 4.3).

4.1. Incremental Fusion Results

In Section 3.3, we have presented an incremental fusion pipeline using our SuBloNet architecture. A stream of depth images is fused into a latent space data structure and then converted to a truncated signed distance field. The complete system is trained end-to-end from a sequence of local TSDFs as input to a fused voxel grid as output. For every optimizer step, 8 images are fused into the latent space and then converted to the output TSDF once. Both networks are optimized simultaneously using the Adam algorithm [KB14] with a learning rate of 10^{-4}. Due to memory limitations of the recurrent architecture, we randomly crop the input and ground-truth to a maximum size of 300 voxel blocks. In our tests, the network

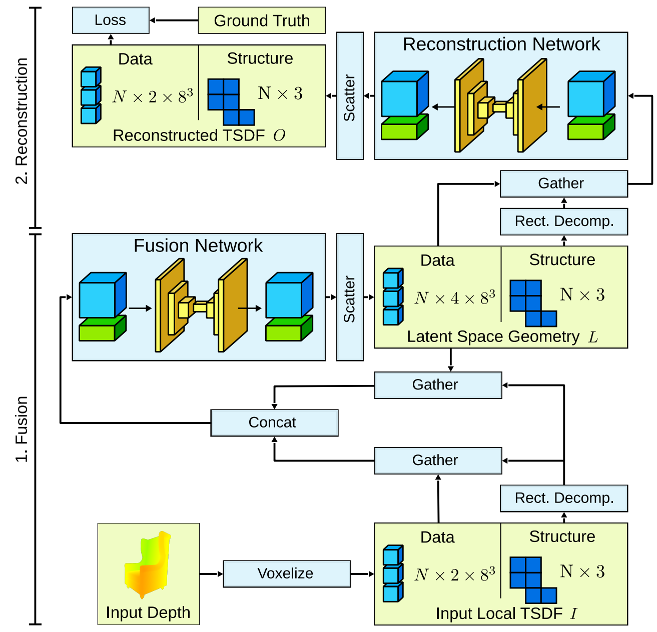

Figure 4: *The incremental 3d reconstruction pipeline using the super block architecture. In the fusion stage (1.), the input images are integrated into a latent space feature grid. After multiple images have been fused, the block sparse TSDF is extracted from the latent space using a reconstruction network (2.).*

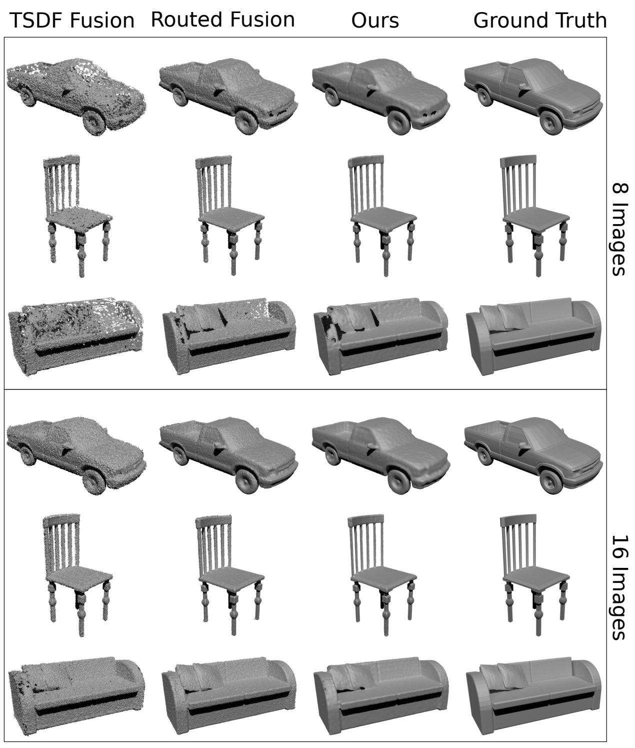

Figure 5: *Reconstructed surface after fusing 8 (top) and 16 (bottom) depth images. TSDF Fusion (left column) is the traditional approach based on [CL96].*

converges in around 40 epochs when training on 500 models and 16 depth images per model. On a single NVidia RTX 2080 GPU one epoch takes around 7 minutes to complete.

Now we compare the results of our approach to traditional volumetric fusion [CL96] and RoutedFusion [WSPO20], a state-of-the-art deep learning-based method. We have trained the network to incrementally reconstruct the shape from virtual depth images captured in random locations around 3d models from the ShapeNet dataset [CFG*15]. Similar to the work of [WSPO20], we add additional noise to the measurements and invalidate random patches of pixels.

Figure 5 shows the fusion result after 8 (top half) and 16 (bottom half) depth images have been processed. As you can see in the left most column, the traditional volumetric fusion approach is not able to reconstruct a high quality surface from few input images. Noise is not sufficiently removed and holes are not closed. Both RoutedFusion and our method are able to close most of the holes in the surface. Noise is also drastically reduced, though our approach produces a slightly cleaner mesh due to the U-net's good smoothing properties. This is especially visible when only few depth images are present. For example, the chair's backrest is sharper and the sofa contains visibly less artifacts and noise.

4.2. Block Sparse Network Benchmark

Our super block architecture, which we have used for high-quality surface reconstruction, can also be used for various other tasks with block sparse input data. Therefore, we benchmark the most basic configuration, which is a single 3d U-net processing a block sparse input tensor. Such a network can be, for example, trained to smooth a reconstructed surface or predict a class label for every voxel. We test two differently sized networks with a receptive field of 16^3 and 32^3 respectively. As input data, we use a real-world scene from the Scannet dataset [DCS*17] that was reconstructed in different resolutions. Figure 6 shows the block sparse structure, where each cube on the left is a 8^3 voxel block. On the right hand side, our approximate super block decomposition is shown. In the high resolution example (bottom) $n = 377854$ voxel blocks are decomposed into $N = 1258$ cuboids.

We have evaluated the training time of the U-nets when implemented using our method and other approaches. In Table 1 the run-time measurements of SuBloNet, the dense implementation of Pytorch [PGM*19], SBNet [RPYU18] with 32^3, 64^3, and 128^3 blocks, and Torchsparse [TLZ*20] are presented. For low resolution input, the dense implementation shows promising results outperforming the other sparse methods. However, if we increase resolution, performance of the dense neural network degrades compared to the other approaches. On high resolutions, SBNet64 slightly outperforms SBNet32, SBNet128, and Torchsparse but falls behind our super block architecture. In total, our approach is between 1.7x and 4x more efficient than the other approaches on block sparse TSDF data. This validates that the core idea of super block decomposition is well suited for neural processing of truncated signed distance fields. The memory consumption of SuBloNet is also slightly lower than that of Torchsparse, because they require one coordinate for every feature and an additional ker-

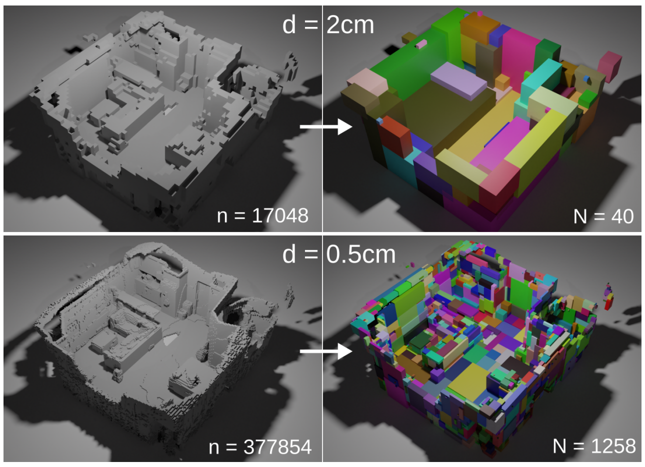

Figure 6: *Our rectilinear decomposition converts block sparse data (left) into sparse super blocks (right). Every block on the left is further subdivided into* 8^3 *voxels resulting in over 150M feature vectors for the bottom row.*

Resolution (cm)	4	2	1	0.8	0.6	0.5
# Non Zeros	1.4M	6.7M	31M	53M	104M	160M
Density	0.29	0.19	0.12	0.11	0.09	0.07
	U-net Receptive Field = 16					
Torch Dense	0.34	1.75	11.6	21.64	49.21	85.32
SBNet32	0.52	1.85	7.73	13.28	23.9	37.91
SBNet64	0.55	1.47	7.22	11.56	20.52	31.3
SBNet128	0.45	1.94	8.56	13.66	26.34	40.26
Torchsparse	0.39	1.69	7.61	12.41	25.26	40.14
SuBloNet (ours)	0.33	1.03	4.04	6.25	11.81	18.78
	U-net Receptive Field = 32					
Torch Dense	0.52	2.5	16.02	30.01	67.77	115.37
SBNet32	1.07	3.81	17.16	27.27	50.6	77.93
SBNet64	0.9	2.42	11.37	18.13	33.21	50.1
SBNet128	0.62	2.7	11.54	18.33	35.97	54.64
Torchsparse	0.55	2.39	10.77	18.11	38.02	59.1
SuBloNet (ours)	0.47	1.44	6.37	9.7	18.23	28.03

Table 1: *Training time in seconds for one iteration on a room scale sparse TSDF. The scene has been reconstructed in different voxel resolutions ranging from* 4 cm *to* 5 mm.

nel map to compute the convolutions. SBNet is not sparse in memory. They store the data in a dense volume and process blocks which are marked using a sparsity map.

The time required for the rectilinear decomposition is included in Table 1. For a resolution of 4 cm, the decomposition takes around 0.01 s and for a resolution of 0.5 cm it requires approximately 0.5 s on the CPU. Inside the decomposition, more than 90% of the time is spent on the two merge-based optimization stages. The time required for the initial octree-based decomposition is insignificant.

4.3. Limitations and Discussion

We have shown that the performance of deep learning on volumetric data structures can be improved by the use of our super block architecture. However, this approach also has a few limitations that have to be considered when using SuBloNet. First, the rectilinear decomposition adds additional computational overhead, which might be significant for real-time applications or if the network is very small. This is not a problem for moderate and large networks because the efficiency gain outweighs the overhead cost (see Section 4.2). The second limitation is that batching multiple inputs during network training is limited, because the super blocks can have arbitrary dimension. In some cases this might reduce training performance, however inference efficiency is unchanged. This problem could be circumvented by forcing the generation of uniform super blocks during training stage.

5. CONCLUSION

We have presented a novel deep learning architecture for efficiently processing block sparse volumetric input data. The core idea is the rectilinear decomposition of the input structure to extract dense blocks inside the sparse volume. We have analyzed this decomposition and presented the bandwidth-optimal solution, which is defined as a minimization problem. Since computing the optimal solution is a NP-hard problem, we have also derived a fast approximation algorithm that can readily be used in interactive applications. In our experiments, the SuBloNet architecture outperforms dense implementations and other sparse methods by a factor of 1.7x - 4x on room-scale indoor scenes. For the task 3d reconstruction of depth images, we have shown a novel fusion pipeline that makes use of the super block architecture. This reconstruction approach operates in latent space and shows high-quality results even when the input is very noisy and contains many holes. If we look beyond the scope of this paper, we think that SuBloNet can be used in a multitude of applications that exhibit a block sparse input pattern. The complete source code is available on GitHub.

https://github.com/darglein/SuBloNet

References

[AHS17] Anwar S., Hwang K., Sung W.: Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC) 13*, 3 (2017), 1–18. 2

[AKR99] Anil Kumar V., Ramesh H.: Covering rectilinear polygons with axis-parallel rectangles. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing* (1999), pp. 445–454. 2

[CEKL16] Choi Y., El-Khamy M., Lee J.: Towards the limit of network quantization. *arXiv preprint arXiv:1612.01543* (2016). 2

[CFG*15] Chang A. X., Funkhouser T., Guibas L., Hanrahan P., Huang Q., Li Z., Savarese S., Savva M., Song S., Su H., et al.: Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* (2015). 5

[CGS19] Choy C., Gwak J., Savarese S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 3075–3084. 2

[CL96] Curless B., Levoy M.: A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 303–312. 2, 5

[CZ19] CHEN Z., ZHANG H.: Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 5939–5948. 2

[DCS*17] DAI A., CHANG A. X., SAVVA M., HALBER M., FUNKHOUSER T., NIESSNER M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 5828–5839. 5

[DHYY17] DONG X., HUANG J., YANG Y., YAN S.: More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017). 2

[DK91] DIELISSEN V. J., KALDEWAIJ A.: Rectangular partition is polynomial in two dimensions but np-complete in three. *Information Processing Letters 38*, 1 (1991), 1–6. 2, 3

[DNZ*17] DAI A., NIESSNER M., ZOLLHÖFER M., IZADI S., THEOBALT C.: Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG) 36*, 4 (2017), 1. 2

[DRB*18] DAI A., RITCHIE D., BOKELOH M., REED S., STURM J., NIESSNER M.: Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2018). 2

[DZB*14] DENTON E., ZAREMBA W., BRUNA J., LECUN Y., FERGUS R.: Exploiting linear structure within convolutional networks for efficient evaluation. *arXiv preprint arXiv:1404.0736* (2014). 2

[FG11] FUHRMANN S., GOESELE M.: Fusion of depth maps with multiple scales. *ACM Transactions on Graphics (TOG) 30*, 6 (2011), 1–8. 2

[Fra86] FRANKLIN P.: *Optimal rectangle covers for convex rectilinear polygon*. PhD thesis, Theses (School of Computing Science)/Simon Fraser University, 1986. 2

[GRK17] GRAY S., RADFORD A., KINGMA D. P.: Gpu kernels for block-sparse weights. *arXiv preprint arXiv:1711.09224 3* (2017). 2

[GvdM17] GRAHAM B., VAN DER MAATEN L.: Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307* (2017). 2

[HIF19] HÖSCHL IV C., FLUSSER J.: Close-to-optimal algorithm for rectangular decomposition of 3d shapes. *Kybernetika 55*, 5 (2019), 755–781. 2

[HLM*16] HAN S., LIU X., MAO H., PU J., PEDRAM A., HOROWITZ M. A., DALLY W. J.: Eie: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News 44*, 3 (2016), 243–254. 2

[HLW*19] HE Y., LIU P., WANG Z., HU Z., YANG Y.: Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). 2

[HPTD15] HAN S., POOL J., TRAN J., DALLY W. J.: Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02626* (2015). 1

[HPTT16] HU H., PENG R., TAI Y.-W., TANG C.-K.: Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250* (2016). 2

[HZS17] HE Y., ZHANG X., SUN J.: Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Oct 2017). 2

[IKH*11] IZADI S., KIM D., HILLIGES O., MOLYNEAUX D., NEWCOMBE R., KOHLI P., SHOTTON J., HODGES S., FREEMAN D., DAVISON A., ET AL.: Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (2011), pp. 559–568. 2

[IRCC17] IOANNOU Y., ROBERTSON D., CIPOLLA R., CRIMINISI A.: Deep roots: Improving cnn efficiency with hierarchical filter groups. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017). 2

[Jai02] JAIN A.: *Partitioning 3-D Regions Into Cuboids*. PhD thesis, University of Florida, 2002. 2

[Kar12] KARRAS T.: Maximizing parallelism in the construction of bvhs, octrees, and k-d trees. In *Proceedings of the Fourth ACM SIGGRAPH/Eurographics conference on High-Performance Graphics* (2012), pp. 33–37. 4

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 4

[KPY*15] KIM Y.-D., PARK E., YOO S., CHOI T., YANG L., SHIN D.: Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530* (2015). 2

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics 21*, 4 (1987), 163–169. 4

[LDS*89] LECUN Y., DENKER J. S., SOLLA S. A., HOWARD R. E., JACKEL L. D.: Optimal brain damage. In *NIPs* (1989), vol. 2, Citeseer, pp. 598–605. 1

[LGR*14] LEBEDEV V., GANIN Y., RAKHUBA M., OSELEDETS I., LEMPITSKY V.: Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553* (2014). 2

[LGS*09] LAUTERBACH C., GARLAND M., SENGUPTA S., LUEBKE D., MANOCHA D.: Fast bvh construction on gpus. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 375–384. 4

[LKD*16] LI H., KADAV A., DURDANOVIC I., SAMET H., GRAF H. P.: Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710* (2016). 2

[LLL*79] LIPSKI W., LODI E., LUCCIO F., MUGNAI C., PAGLI L.: On two dimensional data organization ii. *Fundanmenta Informaticae 2* (1979), 245–260. 2

[LLS*17] LIU Z., LI J., SHEN Z., HUANG G., YAN S., ZHANG C.: Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Oct 2017). 2

[LTL89] LIOU W., TAN J., LEE R.: Minimum partitioning simple rectilinear polygons in o (n log log n)-time. In *Proceedings of the fifth annual symposium on Computational geometry* (1989), pp. 344–353. 2

[LWF*15] LIU B., WANG M., FOROOSH H., TAPPEN M., PENSKY M.: Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015). 2

[LWL17] LUO J.-H., WU J., LIN W.: Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (Oct 2017). 2

[Mea82] MEAGHER D.: Geometric modeling using octree encoding. *Computer graphics and image processing 19*, 2 (1982), 129–147. 3

[MF95] MOHAMED S. A., FAHMY M. M.: Binary image compression using efficient partitioning into rectangular regions. *IEEE Transactions on Communications 43*, 5 (1995), 1888–1893. 2

[MHP*17] MAO H., HAN S., POOL J., LI W., LIU X., WANG Y., DALLY W. J.: Exploring the regularity of sparse structure in convolutional neural networks. *arXiv preprint arXiv:1705.08922* (2017). 2

[MON*19] MESCHEDER L., OECHSLE M., NIEMEYER M., NOWOZIN S., GEIGER A.: Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 4460–4470. 2

[Mor66] MORTON G. M.: A computer oriented geodetic data base and a new technique in file sequencing. 3

[MvAB*20] MUREZ Z., VAN AS T., BARTOLOZZI J., SINHA A., BADRINARAYANAN V., RABINOVICH A.: Atlas: End-to-end 3d scene reconstruction from posed images. *arXiv preprint arXiv:2003.10432* (2020). 2

[NS88] NAHAR S., SAHNI S.: Fast algorithm for polygon decomposition. *IEEE transactions on computer-aided design of integrated circuits and systems 7*, 4 (1988), 473–483. 2

[NUD17] NARANG S., UNDERSANDER E., DIAMOS G.: Block-sparse recurrent neural networks. *arXiv preprint arXiv:1711.02782* (2017). 1, 2

[NZIS13] NIESSNER M., ZOLLHÖFER M., IZADI S., STAMMINGER M.: Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG) 32*, 6 (2013), 1–11. 2

[PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 165–174. 2

[PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., ET AL.: Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703* (2019). 5

[PKG*17] PRISACARIU V. A., KÄHLER O., GOLODETZ S., SAPIENZA M., CAVALLARI T., TORR P. H. S., MURRAY D. W.: InfiniTAM v3: A Framework for Large-Scale 3D Reconstruction with Loop Closure. *arXiv e-prints* (Aug. 2017), arXiv:1708.00783. `arXiv:1708.00783`. 2

[PRM*17] PARASHAR A., RHU M., MUKKARA A., PUGLIELLI A., VENKATESAN R., KHAILANY B., EMER J., KECKLER S. W., DALLY W. J.: Scnn: An accelerator for compressed-sparse convolutional neural networks. *ACM SIGARCH Computer Architecture News 45*, 2 (2017), 27–40. 2

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017). 2

[QYSG17] QI C. R., YI L., SU H., GUIBAS L. J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413* (2017). 2

[RFB15] RONNEBERGER O., FISCHER P., BROX T.: U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), Springer, pp. 234–241. 4

[RMO*19] REIJGWART V., MILLANE A., OLEYNIKOVA H., SIEGWART R., CADENA C., NIETO J.: Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps. *IEEE Robotics and Automation Letters 5*, 1 (2019), 227–234. 2

[ROUG17] RIEGLER G., OSMAN ULUSOY A., GEIGER A.: Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 3577–3586. 2

[RPYU18] REN M., POKROVSKY A., YANG B., URTASUN R.: Sbnet: Sparse blocks network for fast inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 8711–8720. 2, 5

[RUBG17] RIEGLER G., ULUSOY A. O., BISCHOF H., GEIGER A.: Octnetfusion: Learning depth fusion from data. In *2017 International Conference on 3D Vision (3DV)* (2017), IEEE, pp. 57–66. 2

[SC17] SHI S., CHU X.: Speeding up convolutional neural networks by exploiting the sparsity of rectifier units. *arXiv preprint arXiv:1704.07724* (2017). 2

[SCYE17] SZE V., CHEN Y.-H., YANG T.-J., EMER J. S.: Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE 105*, 12 (2017), 2295–2329. 2

[SG93] SOLTAN V., GORPINEVICH A.: Minimum dissection of a rectilinear polygon with arbitrary holes into rectangles. *Discrete & Computational Geometry 9*, 1 (1993), 57–79. 2

[SHF12] SUK T., HÖSCHL C., FLUSSER J.: Rectangular decomposition of binary images. In *International Conference on Advanced Concepts for Intelligent Vision Systems* (2012), Springer, pp. 213–224. 2

[SHIF12] SUK T., HÖSCHL IV C., FLUSSER J.: Decomposition of binary images—a survey and comparison. *Pattern Recognition 45*, 12 (2012), 4279–4291. 2

[SKC13] STEINBRUCKER F., KERL C., CREMERS D.: Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *Proceedings of the IEEE International Conference on Computer Vision* (2013), pp. 3264–3271. 2

[TLZ*20] TANG H., LIU Z., ZHAO S., LIN Y., LIN J., WANG H., HAN S.: Searching efficient 3d architectures with sparse point-voxel convolution. In *European Conference on Computer Vision* (2020), Springer, pp. 685–702. 2, 5

[TQD*19] THOMAS H., QI C. R., DESCHAUD J.-E., MARCOTEGUI B., GOULETTE F., GUIBAS L. J.: Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (October 2019). 2

[VK*19] VARMA G., KOTHAPALLI K., ET AL.: Dynamic block sparse reparameterization of convolutional neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops* (2019), pp. 0–0. 2

[WGH*21] WEN N., GUO R., HE B., FAN Y., MA D.: Block-sparse cnn: towards a fast and memory-efficient framework for convolutional neural networks. *Applied Intelligence 51*, 1 (2021), 441–452. 2

[WHL01] WU C.-H., HORNG S.-J., LEE P.-Z.: A new computation of shape moments via quadtree decomposition. *Pattern Recognition 34*, 7 (2001), 1319–1330. 2

[WSPO20] WEDER S., SCHONBERGER J., POLLEFEYS M., OSWALD M. R.: Routedfusion: Learning real-time depth map fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 4887–4897. 2, 5

[WWW*16] WEN W., WU C., WANG Y., CHEN Y., LI H.: Learning structured sparsity in deep neural networks. *arXiv preprint arXiv:1608.03665* (2016). 2

[YCS17] YANG T.-J., CHEN Y.-H., SZE V.: Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017). 1

[ZYG*17] ZHOU A., YAO A., GUO Y., XU L., CHEN Y.: Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044* (2017). 2