Improved Block Merging for 3D Point Cloud Instance Segmentation

Leon Denis, Remco Royen and Adrian Munteanu
Vrije Universiteit Brussel, Departement of Electronics and Informatics, 1050 Brussels, Belgium
Email: leon.denis@vub.be, remco.royen@vub.be, adrian.munteanu@vub.be

***Abstract*—This paper proposes a novel block merging algorithm suitable for any block-based 3D instance segmentation technique. The proposed work improves over the state-of-the-art by allowing wrongly labelled points of already processed blocks to be corrected through label propagation. By doing so, instance overlap between blocks is not anymore necessary to produce the desirable results, which is the main limitation of the current art. Our experiments show that the proposed block merging algorithm significantly and consistently improves the obtained accuracy for all evaluation metrics employed in literature, regardless of the underlying network architecture.**

I. Introduction

Interest in 3D data is growing at a record pace. This is largely fuelled by the advancements in 3D data capturing technologies in combination with the AI breakthrough seen in recent years. One particularly important aspect related to this field is 3D scene understanding. Domains such as virtual reality, autonomous driving, and drone exploration require a deep understanding of the captured 3D scene. This includes semantic segmentation, which classifies the class of each object in the 3D point cloud, and instance segmentation, which performs semantic segmentation in combination with instance labelling for each point.

The domain of 3D scene instance segmentation can be subdivided in two different groups, each targeting a different application. Belonging to the first group, one can distinguish the neural networks operating on the full scene at once. Those *full-scene methods* are particularly useful for non-real-time applications where general scene understanding is not necessary during capturing or when enough resources are available during runtime. Inspection of office buildings can serve as an example. Important full-scene methods include [1], [2], [3], [4], [5]. In the first work, a network named PointGroup [1] learns offsets that aid a point clustering algorithm by increasing the intra-instance distances. Redundant clusters are subsequently removed by non-maximum suppression and a dedicated network that evaluates each obtained instance candidate. In DyCo3D [3] dynamical convolutions are employed to ultimately achieve instance segmentation. In a later work, a hierarchical aggregation is proposed to progressively generate instance proposals [4]. SoftGroup [5] further improves upon this work and introduces soft semantic labelling prior to clustering.

The second group consist of *block-based methods*. Unlike their full-scene counterparts, they operate on parts of the scene, and thus, are able to analyse 3D environments during point cloud capturing. This is particularly useful for applications such as autonomous driving and drone exploration, which typically need to interpret the currently captured scene in order to navigate. Recent works in this field include [6], [7], [8], [9], [10], [11], [12], [13]. Similar as the full-scene methods, they often rely on clustering techniques. SGPN [6], for example, computes a similarity matrix which is subsequently used for grouping similar points to one particular instance. ASIS [7], JSNet [8] and JSPNet [10] jointly perform semantic and instance segmentation through specifically designed modules connecting both branches of the network. The resulting feature vector once again serves as input for a clustering algorithm. In [9], each point is represented as a tri-variate normal distribution and a novel loss function is employed for clustering. Lastly, MP-Net [14] proposes a memory-augmented network which learns and memorizes feature prototypes for scene analysis.

As mentioned previously, all aforementioned methods belonging to this group process the scene in a block-based manner. Full scene analysis is obtained by combining the information of each block using a merging algorithm. As will be clarified in the remainder of this paper, the employed algorithm can greatly affect the obtained results. Yet, to the best of our knowledge, only a single block merging algorithm to date exists, which we will henceforth refer to as 'BlockMerging v1' [6]. It is used by *all* aforementioned block-based segmentation techniques. The proposed work has one major contribution. It proposes a novel block merging algorithm that vastly improves over the current art. Unlike the original method, the improved algorithm considers historical information and allows already labelled points to be altered to correct wrongly labelled instances from past blocks. Consequently, the improved block merging algorithm consistently and simultaneously improves performance with respect to multiple evaluation metrics, regardless of the employed architecture.

II. BlockMerging v1

Block-based approaches for 3D scene instance segmentation partition a given scene in blocks prior to processing. These blocks serve as input for the block merging algorithm. The area of the block size is commonly chosen as $1m \times 1m$, which corresponds to the original proposal of [6] and which is also retained in this work. We note that blocks are constructed based on the ground level. Infinite height is assumed. We define the i^{th} block as B_i from the set of blocks B. In addition to block-based partitioning, the input scene is also voxelized. The size of each cell is chosen such that the input scene fits a $400 \times 400 \times 400$ grid. The voxelization, thus, operates on all 3 spatial dimensions, unlike block partitioning. Each point of the scene is assigned to the voxel, which spatially embeds said point. We mathematically express this as:

$$c_j = C(p_n), \tag{1}$$

with c_j the corresponding voxel (or cell) of input point p_n and C the voxelization operation. The general idea behind block merging is defining a label for each cell based on the predicted labels of its population. The method starts by defining for each cell c_j of the first block B_1 the label k selected from the set of instances identifiers K where

$$\sum_{n=1}^{N} [\sigma(p_{j,n}) = k] \tag{2}$$

is maximal. σ is the inference operator of the neural network operating on the n^{th} input point $p_{j,n}$ belonging to cell c_j. We denote $l_{c_j} = k$ as the label belonging to c_j. Next, blocks are processed in an overlapping manner using a sliding window with snake pattern and stride 0.5 meter, as shown in Figure 1. We note that other scanning methods can improve performance. For each block, all instances I_m are traversed and the labels l_{c_j} are assigned to the cells $C_{j,m}$ belonging to I_m if the number of occurrences of l_{c_j} is larger than a given threshold τ_1. If not, the cells $C_{j,m}$ are labelled with a new monotonically increasing grouping identifier, which represents a new instance. This processes is repeated until all blocks are processed after which all $p_{j,n}$ in c_j are labelled as l_{c_j}.

A. *Limitations*

Though performing well in most cases, the original block merging algorithm has one major limitation [6]. The underlying snake pattern does not ensure instance overlap when processing blocks. That is, it is possible that only not-yet-labelled points of a particular instance identified in the past are present in a given block. This, combined with the monotonically increasing grouping identifier results in the failure of grouping numerous instances. This is demonstrated by Figure 1. The dashed lines indicate the position of all blocks and are drawn with 0.5 meter offset. Note that the blocks themselves are 1m^2. The first and second blocks are depicted in red and green, respectively. They process the top part of the sofa, and label the associated points with a given instance number. The problem arises when processing the 8th block, depicted in blue. That particular block is only able to process the portion of the sofa that is marked. As there is no overlap with other points

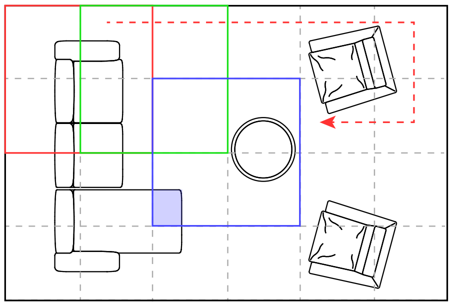
Fig. 1: Top down view of a room divided in blocks. The portion of the sofa marked in blue will be wrongly labelled using current block merging techniques [6] as no overlap is present.

associated with the sofa, the currently processed points will be wrongly labelled as a new instance.

III. Proposed method

To alleviate this issue, we propose a novel block merging algorithm that is able to alter past labelled cells while processing any given block. This is done by allowing the merging of two or more different labels when object confusion is detected as more information of different blocks becomes available. The pseudocode of our algorithm is shown in Algorithm 1. Red lines indicate the lines of code introducing the new concepts. We will refer to this method as 'BlockMerging v2'. Instead of permanently labelling processed cells, labels $l_{c_{j,m}}$ associated with cells from instance I_m are stored in the list L_{I_m}. Note that the overlapping windows allow multiple $l_{c_{j,m}}$ to be associated with the same instance I_m, which is exactly the root of the problem, resulting in subdivided instances. Once a block has been processed, labels are propagated on a per-instance basis using the instances resulting from the block-based prediction. That is, should there exist multiple $l_{c_{j,m}}$ for any given instance I_m, then the labels in L_{I_m} are redefined as $l_{I_{max}}$ such that

$$\sum_{m=1}^{M} [l_{c_{j,m}} = L_{I_{max}}] \tag{3}$$

is maximal. In other words, the ambiguity of the labels associated with a given instance is removed by labelling each cell $C_{j,m}$ with the most occuring label $L_{I_{max}}$. We note that the labels $l_{c_{j,m}}$ are altered on a global level while instances I_m are processed in a sequential, but not necessarily successive, order. This allows propagating labels farther than a single block-size, ultimately leading to full-scene label propagation.

To provide more control of the merging process, we also introduce a control parameter τ_2, which prevents merging when the number of cells with a given label is above a threshold for a particular instance. Larger thresholds allow more label ambiguity and thus, instance partitioning, indirectly resulting in smaller instances being detected, hence, improving recall. On the other hand, a smaller τ_2 will faster enforce label merging, which in turn increases overall precision. Using the control parameter τ_2, one is, thus, able to fine-tune the overall inference results and balance recall and precision to a limited, but nonetheless useful, extend. More importantly, empirical results, and an ablation study presented later in this paper, show that recall and precision can both be improved simultaneously when using the proposed method compared to the original block merging algorithm [6].

IV. Experimental assessments

As a first experiment, we demonstrate the influence of the threshold τ_2 with respect to the obtained performance, which is shown in Figure 2. For this specific experiment, the publicly available model of JSNet was employed [8]. From the figure, one can notice similar behaviour for all voxel sizes. Importantly, mean precision and recall *both* benefit at low thresholds. As expected, degraded performance is noticeable when choosing τ_2 too large. It is also noteworthy that better overall accuracy is obtained with increasing resolution of the voxel grid. Peak performance is obtained with thresholds around 50 and 100. Similar observations were made for the

Algorithm 1 - BlockMerging v2

```
Input: Blocks: $B$, Instances per block: $I$, Voxel grid: $V$
Output: Point instance labels for the whole scene: L
Initialize all cells $c_j$ of $V$ as $-1$
group $\leftarrow 0$
for every block $B_i$ do
    if $B_i$ is the first block then
        for every point $p_n$ in $B_i$ belonging to $I_m$ do
            Define $c_j = C(p_n)$
            $l_{c_j} \leftarrow \sigma(p_n)$
            Add $l_{c_j}$ to the instance mapping table $H_{I_m}$
            if group $<= l_{c_j}$ then
                group $\leftarrow l_{c_j} + 1$
            end
        end
    else
        for every instance $I_m$ do
            Define cells $c_j$ which embed points of $I_m$
            Define $l'_{I_{max}}$ as the mode of $I_m$ with cells not $-1$
            if $l_{c_j}$ is not -1 then
                if frequency of $l'_{I_{max}} < \tau_1$ then
                    $l_{c_j} \leftarrow$ group
                    group $\leftarrow$ group $+ 1$
                else
                    $l_{c_j} \leftarrow l'_{I_{max}}$
                end
                Add $l_{c_j}$ to the instance mapping table $H_{I_m}$
            end
        end
        for every instance $I_m$ do
            for every label $l_{I_j}$ belonging to $I_m$ do
                Use $H_I$ to define $l_{I_{max}}$ as the mode of $I_m$
                if the frequency of $l_{I_j} > \tau_2$ then
                    $l_{I_j} = l_{I_{max}}$
                end
            end
        end
    end
    for every point $p_n$ in the whole scene do
        Define $c_j$ as $C(p_n)$
        $L_{p_n} \leftarrow l_{c_j}$
    end
end
```

mean coverage and mean weighted coverage. In terms of network accuracy, we have compared the proposed method to the original BlockMerging algorithm of [6] using numerous instance segmentation architectures [7], [8], [10], [1], [3], [4]. Besides block-based methods, we have also incorporated networks normally operating on full scenes. Their models have been retrained specifically for blocks. For the other networks, publicly available models were employed. The only exception is the most recent state-of-the-art block-based method of [10], for which the results of the original work are reported. We were unable to apply the proposed block merging on the output of [10] as its source code is not made public fully. However, we report their results to allow making a comparison with the state-of-the-art in block-based instance segmentation.

Table I and Table II summarize the findings of our experiments. For grid size 400^3 a τ_2 of 110 was employed. For

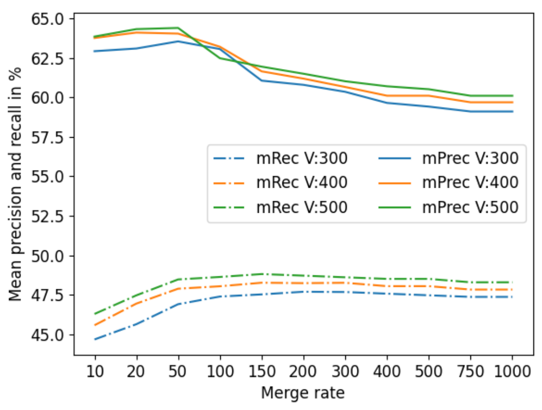

Fig. 2: The effect of the parameter τ_2 with respect to the mean precision and mean recall for different voxel sizes.

500^3 50 was selected for all architectures except for ASIS [7], for which 110 was used. From the tables, it is clear that the proposed method significantly improves upon BlockMerging v1 [6]. This is consistently true for *all* evaluated architectures and for *all* metrics. It is also noteworthy that the obtained results for JSNet are slightly lower than reported in their original work [8]. However, even so, employing BlockMerging v2 with JSNet significantly outperforms the state-of-the-art method of [10], improving mean recall with 0.2% and mean precision with a significant 3.5% when taking the highest available values for both metrics into account. When using a grid of 500^3 even better results are obtained, improving mean recall and precision with 0.5% and 4.8%, respectively. The best mean and weighted coverage are still produced by JSPNet. However, given the outcome of the current experiments, it is fair to assume better results would be obtained when utilising the proposed block merging for that architecture as well.

Looking at individual improvements, compared to BlockMerging v1 [6], the proposed method increases mean recall up to 2.8%, and mean precision up to 7.2%. For mean coverage and weighted coverage, the additional performance increase adds up to 1.8% and 2.5%, respectively. These results clearly indicate the benefits of the proposed block merging algorithm.

Visual results for grid sizes 400^3 and 500^3 are shown in Figure 3 and Figure 4, respectively. The figures demonstrate the improved accuracy when using the proposed method. Less object confusion is noticed, and instances that were wrongly subdivided in multiple instances are now correctly fused. This is especially noticeable for the bottom row in Figure 4. Though some object confusion is still present, the proposed method clearly and greatly improves the overall segmentation results.

Regarding the time complexity, the newly introduced functionality increased the execution time with roughly 29%, on a laptop with an Intel i9-12900h.

V. Conclusion

This paper presents a novel block merging algorithm that significantly improves the prediction accuracy compared to the reference merging method that is currently employed by all

TABLE I: Instance segmentation results on S3DIS-blocks for Area-5 using the original [6] and proposed block merging methods. The models denoted with * are full-scene methods retrained on blocks.

Method	BlockMerging v1 [6]				Proposed				Delta			
	mRec	mPrec	mCov	mWCov	mRec	mPrec	mCov	mWCov	mRec	mPrec	mCov	mWCov
ASIS [7]	35.2	46.4	34.7	41.2	35.3	49.0	35.0	42.2	0.1	2.6	0.3	1.0
JSNet [8]	47.8	59.7	44.0	49.6	**48.2**	**63.2**	44.6	50.6	0.4	3.5	0.6	1.0
JSPNet [10]	48.0	59.6	**50.7**	**53.5**	-	-	-	-	-	-	-	-
PointGroup* [1]	45.4	48.1	41.6	46.9	46.9	53.4	42.6	48.6	1.5	5.3	1.0	1.7
DyCo3D* [3]	47.2	55.2	43.8	49.1	47.6	57.9	44.6	50.5	0.4	2.7	0.8	1.4
HAIS* [4]	42.9	48.3	39.6	45.3	44.7	52.9	41.0	47.5	1.8	4.6	1.4	2.2

TABLE II: Same table as Table I, but for grid size $500 \times 500 \times 500$.

Method	BlockMerging v1 [6]				Proposed				Delta			
	mRec	mPrec	mCov	mWCov	mRec	mPrec	mCov	mWCov	mRec	mPrec	mCov	mWCov
ASIS [7]	35.1	46.6	34.7	41.6	35.3	48.8	35.1	42.3	0.2	2.2	0.4	0.7
JSNet [8]	48.3	60.1	44.4	49.9	**48.5**	**64.4**	**44.9**	**51.0**	0.2	4.3	0.5	1.1
PointGroup* [1]	45.5	52.2	41.6	46.9	47.1	59.4	42.8	48.9	1.6	7.2	1.2	2.0
DyCo3D* [3]	47.4	56.6	43.6	49.0	48.3	60.4	44.8	50.5	0.9	3.8	1.2	1.5
HAIS* [4]	42.7	48.2	39.5	45.3	45.5	54.3	41.3	47.8	2.8	6.1	1.8	2.5

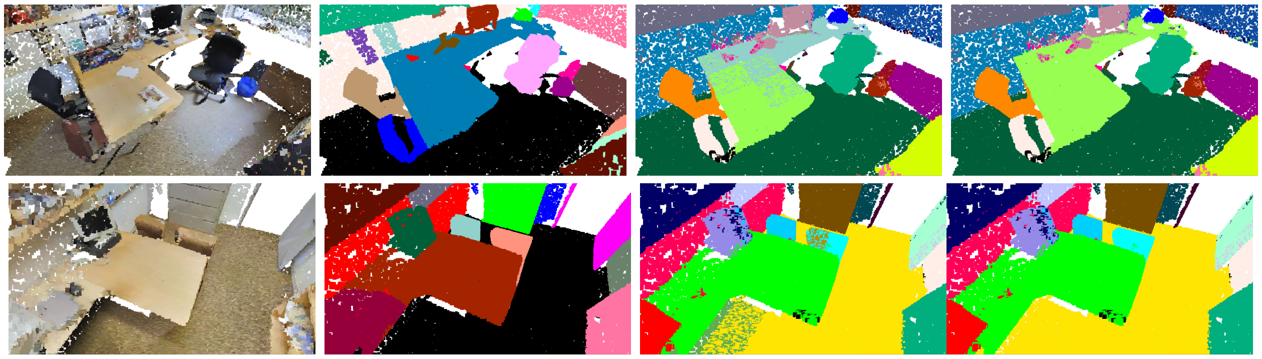
Fig. 3: Results for grid resolution 400^3. From left to right: input, ground truth, BlockMerging v1 [6], proposed.

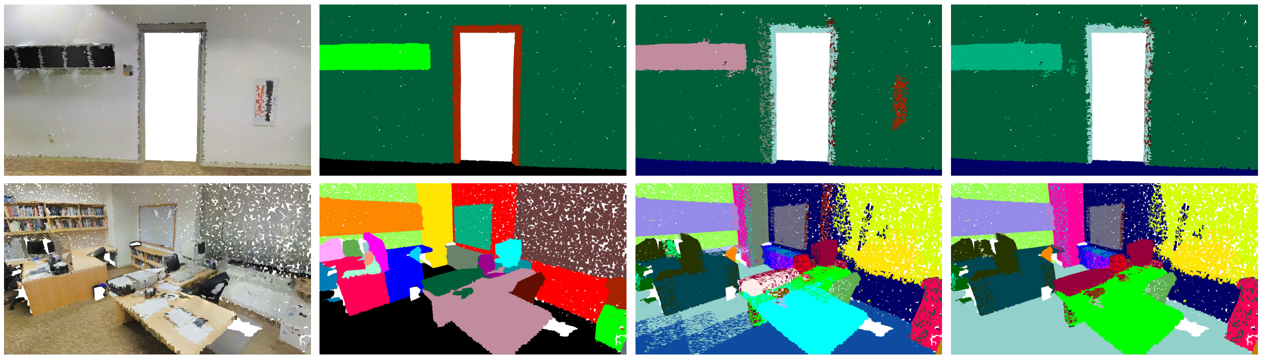
Fig. 4: Visual results for grid size 500^3.

block-based instance segmentation techniques in the literature. The increased performance is obtained by enabling wrongly labelled points from past processed blocks to be corrected through label propagation. By doing so, instance overlap when processing blocks is not anymore required to achieve consistent results, which is the main limitation of the original method. Experiments consistently show that state-of-the-art performance is obtained when adopting the proposed block merging algorithm. For mean recall and precision, improvements up to 2.8% and 7.2% are obtained. For mean coverage and weighted coverage, the improvements add up to 1.8% and 2.5%, respectively. Visually, the additional performance is translated to less object confusion, ultimately leading to clearer and more refined instance segmentation.

Conceptually, the proposed method will benefit all existing block-based instance segmentation methods in the literature. Investigating this aspect is left as topic of further investigation.

Acknowledgment

This research is funded by Innoviris (BRGRD66-TORRES) and FWO (1S89420N).

References

[1] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia, “Pointgroup: Dual-set point grouping for 3d instance segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4867–4876.

[2] L. Han, T. Zheng, L. Xu, and L. Fang, “Occuseg: Occupancy-aware 3d instance segmentation,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2940–2949.

[3] T. He, C. Shen, and A. van den Hengel, “Dyco3d: Robust instance segmentation of 3d point clouds through dynamic convolution,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 354–363.

[4] S. Chen, J. Fang, Q. Zhang, W. Liu, and X. Wang, “Hierarchical aggregation for 3d instance segmentation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 467–15 476.

[5] T. Vu, K. Kim, T. M. Luu, T. Nguyen, and C. D. Yoo, “Softgroup for 3d instance segmentation on point clouds,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 2708–2717.

[6] W. Wang, R. Yu, Q. Huang, and U. Neumann, “Sgpn: Similarity group proposal network for 3d point cloud instance segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2569–2578.

[7] X. Wang, S. Liu, X. Shen, C. Shen, and J. Jia, “Associatively segmenting instances and semantics in point clouds,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4096–4105.

[8] L. Zhao and W. Tao, “Jsnet: Joint instance and semantic segmentation of 3d point clouds,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 12 951–12 958.

[9] B. Zhang and P. Wonka, “Point cloud instance segmentation using probabilistic embeddings,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8883–8892.

[10] F. Chen, F. Wu, G. Gao, Y. Ji, J. Xu, G.-P. Jiang, and X.-Y. Jing, “Jspnet: Learning joint semantic & instance segmentation of point clouds via feature self-similarity and cross-task probability,” *Pattern Recognition*, vol. 122, p. 108250, 2022.

[11] C. Elich, F. Engelmann, T. Kontogianni, and B. Leibe, “3d bird's-eye-view instance segmentation,” in *German Conference on Pattern Recognition*. Springer, 2019, pp. 48–61.

[12] Q.-H. Pham, T. Nguyen, B.-S. Hua, G. Roig, and S.-K. Yeung, “Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8827–8836.

[13] T. He, Y. Liu, C. Shen, X. Wang, and C. Sun, “Instance-aware embedding for point cloud instance segmentation,” in *European Conference on Computer Vision*. Springer, 2020, pp. 255–270.

[14] T. He, D. Gong, Z. Tian, and C. Shen, “Learning and memorizing representative prototypes for 3d point cloud semantic and instance segmentation,” in *European Conference on Computer Vision*. Springer, 2020, pp. 564–580.