Multi-thread Solution of Permutohedral Refined UNet for Cloud/Shadow Detection in High-resolution Remote Sensing Images

Libin Jiao[1], Jibo Wang[1], Zhen Bao[2,3,4,5]
{jiaolibin@cumtb.edu.cn, sqt2410405036@student.cumtb.edu.cn, 12079985@ceic.com}
[1]School of Artificial Intelligence, China University of Mining and Technology-Beijing, Beijing, 100083, China
[2]CHN Energy Science and Technology and Environment Co., Ltd., China
[3]CHN Energy Zhi Shen Control Technology Co., Ltd., China
[4]State R&D Center of Control System and Information Security Technologies for Energy Industry, China
[5]Beijing Engineering Research Center of Power Station Automation, Beijing, 102211, China
Corresponding author: Zhen Bao

Abstract. Boundary-aware high-resolution segmentation aims to partition a high-resolution image into regions in terms of both semantic information and low-level visual features, which has been applied to the discovery of fine-grained objects of interest, for example, cloud/shadow detection in remote sensing images. Their computational cost, on the other hand, has to be carefully considered due to the quadratic time complexity of their naive implementations. Such limitations to practical applications motivate us to try to improve the efficiency performance from a practical perspective by distributing independent computations into multiple CPU threads. We therefore present a multi-thread implementation for our Permutohedral Refined UNet to achieve global boundary refinement for cloud/shadow detection. Specifically, the bilateral/spatial feature generations, a part of filter initialization, a part of filter computation, and CRF iterations can be computed in parallel, which allows us to distribute such computations into multiple CPU threads. The left computations still run sequentially. We then evaluate the efficiency performance of our multi-thread implementations in statistics, and find that a significant efficiency gain is achieved by our multi-thread implementations. The source codes are publicly available at https://github.com/jiaolobel/perm-refined-unet-efficient-impls.

Keywords: Boundary-aware segmentation, high-resolution images, cloud/shadow detection, multi-thread implementations, efficiency improvement

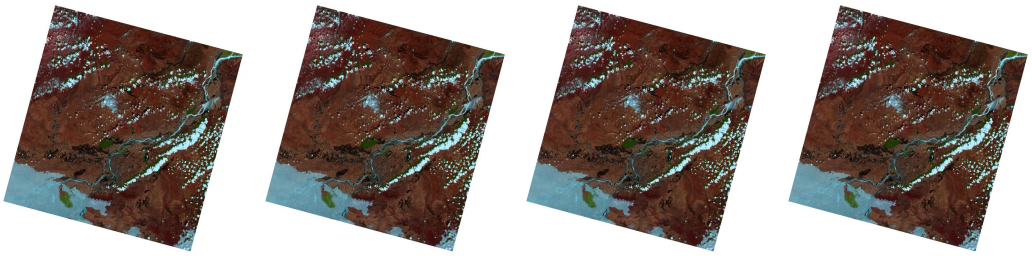
1 Introduction

Boundary-aware high-resolution segmentation aims to partition an image into regions in terms of semantic information and low-level visual features [1, 2, 3, 4]. Such visual techniques can be

IIKI 2025, December 18-21, Chengdu, People's Republic of China

DOI 10.4108/eai.18-12-2025.2365264

applied to fine-grained object discovery in a straightforward way, for instance, cloud/shadow detection in remote sensing images [5, 4]. Our prior research, including Refined UNet [4], v2 [6], v3 [7], v4 [8], and the Permutohedral version [9], has achieved visually significant boundary-refined segmentation performance, demonstrating the effectiveness of boundary-aware applications.



Prototypical	*SeqCompute*	*MtCompute*	*MtFt + MtCompute*
276.11 ± 1.94 s	198.48 ± 2.24 s	120.86 ± 0.74 s	114.97 ± 0.62 s

Fig. 1. Visual inspection from the global perspective. There seems to be no visually significant differences but time consumptions differ significantly. This confirms efficiency gains from our multi-thread implementations.

Unfortunately, such visual techniques bring expensive computational cost when boundary-refined results are produced, leading to limitations to practical applications: the CRF inference, for instance, requires computations with quadratic time complexity if explicit boundary-sensitive terms are applied [2]. Such limitations to practical applications, especially in a GPU-unavailable context, motivate us to try to improve the efficiency performance from a practical perspective by distributing independent computations into multiple threads. We therefore present a multi-thread implementation for our Permutohedral Refined UNet to achieve global boundary refinement for cloud/shadow detection. Specifically, the bilateral/spatial feature generations, a part of filter initialization, a part of filter computation, and CRF iterations can be computed in parallel, which allows us to distribute such computations into multiple threads. The other computations still run sequentially. We then evaluate the efficiency performance of our multi-thread implementations in statistics, and find that significant improvements exist. This confirms efficiency gains from our multi-thread implementations. The source codes are publicly available at https://github.com/jiaolobel/perm-refined-unet-efficient-impls.

The main contributions are as follows.

1. *Potential efficient treatments for the implementations of Permutohedral Refined UNet*: Presented is a multi-thread implementation for our Permutohedral Refined UNet to achieve global boundary refinement for cloud/shadow detection, which is demonstrated to be efficient in a GPU-unavailable context.

2. *Significance evaluation in statistics*: The efficiency significance of such treatments is statistically evaluated, and the efficiency gain is confirmed if $p < 0.05$.

The rest of this report is as follows. Section 2 investigates related works. Section 3 introduces our multi-thread implementation for the Permutohedral Refined UNet. Section 4 presents the experiments, the results, and the discussions supporting our claims. Section 5 concludes this report.

2 Related Works

Boundary-aware segmentation has been constantly considered and explored [2, 3, 10]. Typical solutions try to improve their boundary-aware performance by introducing particular boundary-sensitive modules or processing [10, 11], by adding boundary-sensitive terms to objective functions [12], and by iteratively performing boundary refinements [13]. Such solutions provide confirmations of the validity and prospect of boundary-aware segmentation.

On the other hand, efficient implementations come to our attention after boundary-aware performance has been fully achieved. Neural variants try to reduce their time consumption by deploying their computations on GPUs in practice, while formal researches take steps towards efficient data structures or approximations. For example, formal researches apply efficient permutohedral lattices [14, 2, 15], Gaussian KD trees [16], or approximations [2, 17]. Fast approximations for bilateral message-passing steps are done by applying Taylor expansion [18], trigonometric range kernel approximation [19], linearization with fast Fourier transformation [20], fast high-dimensional filter [17, 14, 16], and bilateral grid [17, 21, 22]. Such approximations are used to significant reduce the time complexity.

In particular, our efficient implementations of Refined UNet are constantly considering end-to-end formulations on GPUs, in order to achieve efficiency gains in practice. This includes Refined UNet v2 [6] / v3 [7] using the guided filters, Refined UNet v4 [8] applying the bilateral grid, Permutohedral Refined UNet [9] with the Eigen library in the CRF inference. Such efficient implementations provide a particular insight to improve the efficiency performance of boundary-aware segmentation, which motivate us to pursue efficient treatments to accelerate the CRF computations, and then to give our multi-thread implementation in this report.

3 Multi-thread Implementations for Global Permutohedral Refined UNet

In general, our Permutohedral Refined UNet follows an iterative mean-field approximation for CRF inference, in the form

$$Q_{t+1}(x_n = l) \leftarrow \frac{1}{Z_n} \exp\left(-\psi_u(x_n = l) - \sum_{l' \in L} \mu(l, l') \sum_i w_i \sum_{n' \neq n} k_i(\boldsymbol{f}_n, \boldsymbol{f}_{n'}) Q_t(x_{n'} = l') \right) \quad (1)$$

where the unary potentials $\psi_u(x_n = l)$ are partially given in terms of the prior probability from our pretrained UNet backbone and the pairwise potentials, bilateral features $\boldsymbol{f}_n$, together with the label compatibility μ are given in our prior researches.

Such an iterative procedure has a nearly linear time complexity in terms of the image size N apart from the $O(N^2)$ bilateral message-passing step. This bilateral step remains a challenging bottleneck for the efficient application of the CRF inference, especially in a GPU-unavailable context.

Fortunately, the pixel-independent property of such a mean-field approximation can be found from (1) because the iteration of $Q(x_n)$ does not depend on its spatial neighbors $Q(x_{n+1})$. A sequentially efficient implementation of such an approximation can be given by distributing computations into multiple threads in the multi-thread CPU context.

The sequential computations of the CRF inference of our Permutohedral Refined UNet can be partitioned into the following parts: feature generation, filter initialization, Q initialization, and mean-field iterations (message-passing steps, compatibility transformations, local updates, and normalizations). In the mean-field iterations, filter computations can be performed in a partially parallel way: the filter blurs inputs in a lattice structure, which is hardly converted into parallel computation. Other computations, including the feature generation, the Q initialization, and the left computations in the iterations, are straightforward to convert into a parallel implementation because of the pixel independence. Specifically, the distributive properties of such computations are as follows.

- **Feature generation**: The feature generation takes the form

$$\boldsymbol{f}_n = \left[\frac{p_{n,\mathrm{X}}}{\theta_\alpha}, \frac{p_{n,\mathrm{Y}}}{\theta_\alpha}, \frac{I_{n,\mathrm{R}}}{\theta_\beta}, \frac{I_{n,\mathrm{G}}}{\theta_\beta}, \frac{I_{n,\mathrm{B}}}{\theta_\beta} \right]^{\mathrm{T}} \tag{2}$$

  and naturally can be computed in a multi-thread way, because $\boldsymbol{f}_n$ is independent of its spatial neighbors and the pixel-independent property therefore holds.

- **Filter initialization**: The filter initializations can be partitioned into distributive and sequential steps, in the form

$$\boldsymbol{v}_n, \boldsymbol{\xi}_n, \boldsymbol{\beta}_n = \mathrm{mtInit}(\boldsymbol{f}_n) \tag{3}$$

$$\boldsymbol{\xi}_m^-, \boldsymbol{\xi}_m^+ = \mathrm{seqInit}(\boldsymbol{v}_m). \tag{4}$$

  We can see that the filter initialization can run in part in a multi-thread way because the permutohedral lattice can be spanned pixel-wise, but the conversion from the sparse lattice to the dense structure should be performed sequentially.

- Q **initialization**: Also, the Q initialization takes the form

$$Q_0(\boldsymbol{x}_n) = -\frac{\exp\left(\psi_u(\boldsymbol{x}_n)\right)}{\sum_{l'} \exp\left(\psi_u(x_n = l')\right)} \tag{5}$$

  and likewise can run in a multi-thread way.

- **Message-passing steps**: Splatting, blurring, and slicing take the forms

$$q'_{\xi_{nd},c} \leftarrow q'_{\xi_{nd},c} + \beta_{nd} \cdot q_{nc} \tag{6}$$

$$q'_{\xi_{m'd},c} \leftarrow \frac{1}{2}\left(q'_{\xi^-_{m'd},c} + q'_{\xi^+_{m'd},c} \right) \tag{7}$$

$$q^*_{nc} \leftarrow q^*_{nc} + \alpha \cdot \beta_{nd} \cdot q'_{\xi_{nd},c}. \tag{8}$$

Likewise, the filter computation can also run in part in a multi-thread way at the splatting and slicing stages because the pixel-independent property holds; multi-thread blurring is still in progress as its blurring step is dynamically dependent on the lattice scales.

- **Compatibility transformations, local updates, and normalizations**: These steps take the form

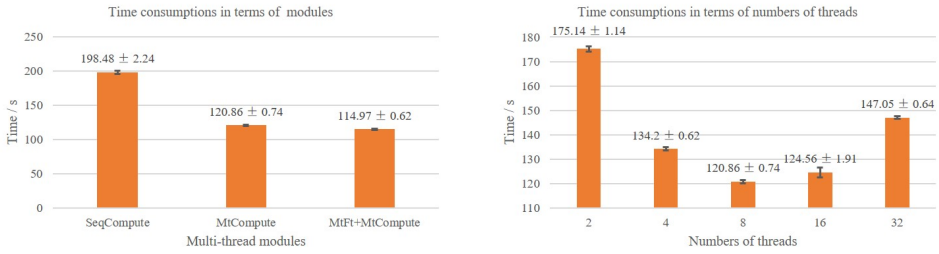
$$Q_{t+1}(x_n = l) \leftarrow \frac{1}{Z_n} \exp\left(-\psi_u(x_n = l) - \sum_{l' \in L} \mu(l, l') \sum_i w_i \tilde{Q}_t(x_{n'} = l') \right) \tag{9}$$

and naturally can be computed in a multi-thread way, because it is straightforward to find that the pixel-independent property holds.

Please kindly refer to https://github.com/jiaolobel/perm-refined-unet-efficient-impls for more implementation details.

4 Experiments and Discussions

We verify if there are efficiency gains by distributing computations into multiple threads for our prior Permutohedral Refined UNet for cloud/shadow detection. The typical test case used in our experiments is from the Landsat 8 OLI dataset [23], and the visual results are illustrated in Figs. 1, 3, and 4. Specifically, we evaluate the significance in statistics: running these variants on one typical case ten times and then performing the paired sample t-tests implemented in SciPy [24] to see if there are differences; efficient gains are acquired if $p < 0.05$. Also, we use the visual check to verify if there are significant differences in the boundary-refined performance. In particular, *Prototypical* refers to the original sequential computation, and *SeqCompute* the multi-thread CRF inference apart from feature initialization and filter computation; *MtCompute* includes partial multi-thread filter computation on the basis of *SeqCompute*; *MtFt+MtCompute* introduces multi-thread feature initialization to *MtCompute*. All the experiment results are illustrated in Figs. 1, 2, 3, and 4.



(a) (b)

Fig. 2. Evaluations with respect to time consumptions. (a) Time consumptions in terms of the multi-thread modules. There are significant differences between *SeqCompute* and *MtCompute* ($p < 0.05$) and between *MtCompute* and *MtFt + MtCompute* ($p < 0.05$). (b) Time consumptions in terms of the numbers of threads. There are significant differences between the implementations with the adjacent numbers of threads ($p < 0.05$).

We can see that there possibly exists no significant difference from the visual perspective from Figs. 1 and 3. This confirms the boundary-refined segmentation effectiveness of such variants. However, we can find significant differences both between *SeqCompute* and *MtCompute* ($p < 0.05$) and between *MtCompute* and *MtFt + MtCompute* ($p < 0.05$). This confirms statistically significant efficiency gains from our multi-thread implementations. Such efficient gains, in our opinion, are possibly attributed to the computational multi-thread distribution in the context of the permutohedral lattices.

We also evaluate the visual and efficiency performances in terms of the numbers of threads. We can see that there possibly exists no significant difference from the visual perspective from Fig. 4. However, we can find significant differences both between the implementations with the adjacent numbers of threads ($p < 0.05$). Such efficiency performances, in our opinion, are possibly attributed to the computational resources (8 physical cores in CPU) and thread scheduling on our hardware.

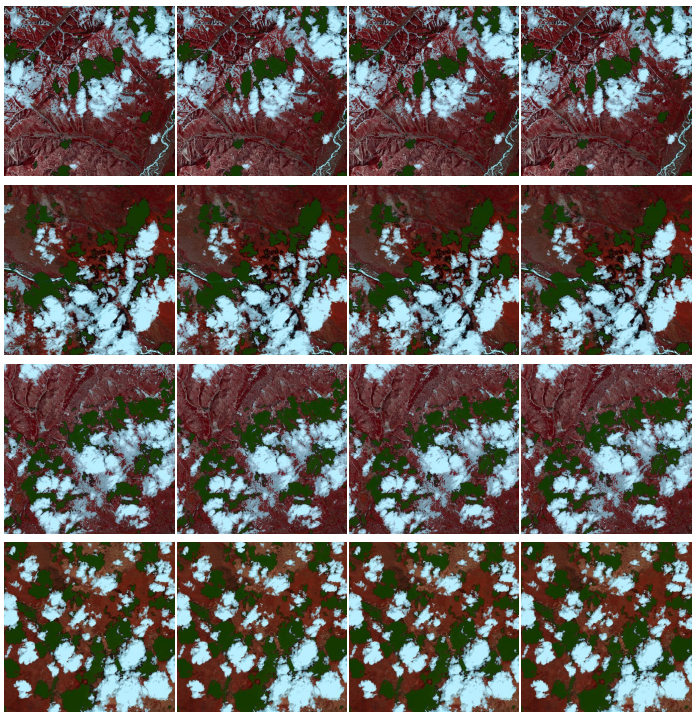


Fig. 3. Visual inspection from the local perspective. It is used to further confirm that there seems no significant differences with respect to the inclusions of multi-thread modules from the visual perspective.

# threads = 2	# threads = 4	# threads = 8	# threads = 16	# threads = 32
175.14 ± 1.14 s	134.2 ± 0.62 s	120.86 ± 0.74 s	124.56 ± 1.91 s	147.05 ± 0.64 s

Fig. 4. Visual inspection from the local perspective. It is used to further confirm that there seems no significant differences with respect to the numbers of threads from the visual perspective.

5 Conclusions

In this report, we aim to improve the efficiency performance of our prior Permutohedral Refined UNet from a practical perspective and therefore present the multi-thread implementations given by distributing CRF computations into multiple threads. This distributive computation can be developed since the pixel-independent property of such a mean-field approximation holds in the iteration of $Q(x_n)$ independent of its spatial neighbors $Q(x_{n+1})$. Specifically, the CRF inference of our Permutohedral Refined UNet can be partitioned into four parts: feature generation, filter initialization, Q initialization, and mean-field iterations, and most of these parts are straightforward to convert into a parallel implementation, other than partial message-passing steps. Also, the efficiency gains are verified from the experiments. We can see that no significant difference exists from the visual

perspective, but there are significant efficiency differences in terms of both multi-thread modules and the number of used threads ($p < 0.05$). This confirms efficiency gains from our multi-thread implementations. Such efficiency improvements are possibly attributed to the computational multi-thread distribution in the context of the permutohedral lattice. In the future, we would like to further improve the efficiency of our Permutohedral Refined UNet by enabling other high-performance computational techniques.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 52404180, in part by the Fundamental Research Funds for the Central Universities of China under Grant 2024ZKPYZN01, and in part by CHN Energy Science and Technology and Environment Co., Ltd., China "Boundary-aware Intelligent Coal Separation Technology based on Weakly Supervised Training". We thank Associate Professor Ming GU from China University of Mining and Technology-Beijing for language editing.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] Bishop CM. Pattern Recognition and Machine Learning. New York: Springer; 2007.

[2] Krähenbühl P, Koltun V. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In: Advances in Neural Information Processing Systems; 2011. p. 109-17.

[3] Zheng S, Jayasumana S, Romera-Paredes B, Vineet V, Su Z, Du D, et al. Conditional random fields as recurrent neural networks. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV); 2015. p. 1529-37.

[4] Jiao L, Huo L, Hu C, Tang P. Refined UNet: UNet-based refinement network for cloud and shadow precise segmentation. Remote Sensing. 2020;12(12):2001.

[5] Chai D, Newsam S, Zhang HK, Qiu Y, Huang J. Cloud and cloud shadow detection in Landsat imagery based on deep convolutional neural networks. Remote Sensing of Environment. 2019;225:307-16.

[6] Jiao L, Huo L, Hu C, Tang P. Refined UNet V2: End-to-End Patch-Wise Network for Noise-Free Cloud and Shadow Segmentation. Remote Sensing. 2020;12(21):3530.

[7] Jiao L, Huo L, Hu C, Tang P. Refined UNet v3: Efficient end-to-end patch-wise network for cloud and shadow segmentation with multi-channel spectral features. Neural Networks. 2021;143:767-82.

[8] Jiao L, Huo L, Hu C, Tang P, Zhang Z. Refined UNet V4: End-to-End Patch-Wise Network for Cloud and Shadow Segmentation with Bilateral Grid. Remote Sensing. 2022;14(2):358.

[9] Jiao L, Huo L, Hu C, Tang P, Zhang Z. Permutohedral Refined UNet: Bilateral Feature-Scalable Segmentation Network for Edge-Precise Cloud and Shadow Detection. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 2024;17:10468-89.

[10] Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2018;40(4):834-48.

[11] Tang C, Chen H, Li X, Li J, Zhang Z, Hu X. Look closer to segment better: Boundary patch refinement for instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2021. p. 13926-35.

[12] Borse S, Wang Y, Zhang Y, Porikli F. Inverseform: A loss function for structured boundary-aware segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2021. p. 5901-11.

[13] Huynh C, Tran AT, Luu K, Hoai M. Progressive semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2021. p. 16755-64.

[14] Adams A, Baek J, Davis MA. Fast High-Dimensional Filtering Using the Permutohedral Lattice. Computer Graphics Forum. 2010;29(2):753-62.

[15] Krähenbühl P, Koltun V. Parameter learning and convergent inference for dense random fields. In: Proceedings of the International Conference on Machine Learning (ICML); 2013. p. 513-21.

[16] Adams A, Gelfand N, Dolson J, Levoy M. Gaussian KD-Trees for Fast High-Dimensional Filtering. Acm Transactions on Graphics. 2009;28(3):1-12.

[17] Paris S, Durand F. A Fast Approximation of the Bilateral Filter Using a Signal Processing Approach. International Journal of Computer Vision. 2009;81(1):24-52.

[18] Porikli F. Constant time O(1) bilateral filtering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2008. p. 1-8.

[19] Chaudhury KN, Sage D, Unser M. Fast $O(1)$ Bilateral Filtering Using Trigonometric Range Kernels. IEEE Transactions on Image Processing. 2011;20(12):3376-82.

[20] Durand F, Dorsey J. Fast Bilateral Filtering for the Display of High-Dynamic-Range Images. ACM Transactions on Graphics. 2002;21(3):257-66.

[21] Chen J, Paris S, Durand F. Real-time edge-aware image processing with the bilateral grid. ACM Transactions on Graphics. 2007;26(3):103.

[22] Chen J, Adams A, Wadhwa N, Hasinoff SW. Bilateral guided upsampling. Acm Transactions on Graphics. 2016;35(6):203.

[23] Vermote E, Justice C, Claverie M, Franch B. Preliminary analysis of the performance of the Landsat 8/OLI land surface reflectance product. Remote Sensing of Environment. 2016;185:46-56.

[24] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods. 2020;17:261-72.