ISPRS Journal of Photogrammetry and Remote Sensing 202 (2023) 13–29

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

Pruning for image segmentation: Improving computational efficiency for large-scale remote sensing applications

Xianwei Lv [a,b], Claudio Persello [b], Wufan Zhao [b,c,*], Xiao Huang [d], Zhongwen Hu [e,f], Dongping Ming [g], Alfred Stein [b]

[a] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China
[b] Department of Earth Observation Science, Faculty ITC, University of Twente, 7500AE Enschede, The Netherlands
[c] Geomatics Section, Department of Civil Engineering, Faculty of Engineering Technology, KU Leuven, Belgium
[d] The Department of Geosciences, University of Arkansas, Fayetteville, AR, USA
[e] School of architecture and urban planning, Shenzhen University, Shenzhen 518060, China
[f] MNR key laboratory for Geo-environmental monitoring of great bay area, Shenzhen university, Shenzhen 518060, China
[g] School of Information Engineering, China University of Geosciences (Beijing), 29 Xueyuan Road, Haidian, Beijing 100083, China

ARTICLE INFO

Keywords:
Image segmentation
Region-merging
Region adjacency graph
Nearest neighbour graph
Pruning

ABSTRACT

Image segmentation is a fundamental step in object-based image analysis and other workflows. However, high-efficiency remains a challenge, especially for the analysis of large-scale Earth observation images. In recent years, considerable effort has been paid to designing merging criteria, automatic scale selection, and object-specific optimisation. These segmentation methods usually rely on the region-adjacency graph (RAG) model and the nearest neighbour graph (NNG) model, which provide acceptable merging performance. Low efficiency occurs due to many redundant edge weight updates in the RAG model. In this study, we propose a generic dynamic pruning framework to improve the efficiency of existing region-merging-based segmentation algorithms, opening the door for large-scale applications in remote sensing. The proposed pruning framework includes intra-object and inter-object pruning modules for the RAG model. Inter-object pruning divides the RAG model into multiple sub-RAG models to reduce the redundant edge weight updates between adjacent objects. Intra-object pruning iteratively divides the sub-RAG into smaller RAGs. In our experimental analysis, we employ the proposed pruning framework with six region-merging segmentation methods and validate the effectiveness on four 10–20M pixel images and a 100M pixel data set. The pruning framework improves the performance of various segmentation algorithms by reducing computational complexity while maintaining segmentation accuracy. We observed a significant improvement in efficiency, with various achieving super-linear speed-up while maintaining the stability of segmentation accuracy. In single-core mode, the computation time of tested algorithms is enhanced by two to ten times on the four test images. In the multicore mode, speed-up increased up to 40 times with eight CPU cores. The computational cost was reduced by 36.15% to 95.77% in the number of weight updates, which is independent of hardware characteristics. On the large-scale image, two modes achieved speed-ups of 36.07 and 102.74, respectively.

1. Introduction

The rapid advances in remote sensing technology have led to increased numbers of very high resolution (VHR) remote sensing imageries, allowing for precise geometrical analysis and semantic interpretation of objects at fine scales (Liu and Abd-Elrahman, 2018; Zhao et al., 2021, 2022). Object-based image analysis (OBIA) has been proven reliable for VHR image analysis thanks to its capability to utilise geometric and spatial contextual features while also reducing the salt-and-pepper noise often introduced by pixel-based methods (Blaschke, 2010; Van den Bergh et al., 2012; Lv et al., 2019). Consequently, OBIA has a wide range of applications, including land cover mapping, ecological wetland surveying, and crop type identification (Zhang et al., 2018b; Orynbaikyzy et al., 2019; Bhatnagar et al., 2021; Lv et al., 2021). Accurate image segmentation effectively solves the difficulty posed by intra-object heterogeneity, serving as a preconditional step in OBIA methods.

Traditional segmentation methods are developed based on models such as mean-shift (Ming et al., 2015), watershed (Shafarenko et al.,

* Corresponding author at: Geomatics Section, Department of Civil Engineering, Faculty of Engineering Technology, KU Leuven, Belgium.
E-mail address: wufan.zhao@kuleuven.be (W. Zhao).

https://doi.org/10.1016/j.isprsjprs.2023.05.024
Received 11 January 2023; Received in revised form 26 April 2023; Accepted 23 May 2023
Available online 8 June 2023

1997), region merging (Hossain and Chen, 2019), and Markov Random Field (Zhang et al., 2017). Among these studies, the region merging method has been widely used for generating segments via different threshold settings. Built upon these studies, numerous improved segmentation algorithms have been developed by estimating the optimal scale parameter, designing merging rules, or automating the determination of object scale (Hossain and Chen, 2019; Zhang et al., 2020). Although the research focus has largely shifted towards deep learning models in the last decade, traditional image processing and analysis like region-merging algorithms can still play an important role in several remote sensing and computer vision applications. Deep learning models are popular for addressing several image analysis tasks, such as semantic segmentation, instance segmentation, object detection, and so on. However, they typically require large amounts of labelled data and training of such models comes at significant computational costs. Conversely, region-merging methods can process large images quickly without relying on massive training data. Most region-merging segmentation methods are in fact completely unsupervised, while the recently introduced DeepMerge algorithm exploits the advantage of deep learning for learning the similarity between adjacent segments considering labelled pairs of segments, i.e., a form of weak supervision. As a part of the overall image processing workflow, region merging methods output image segments (or super-pixels), an essential step of the object-based image analysis workflow to finally produce a semantic segmentation. Super-pixel segmentation can be combined with convolutional networks or other deep learning methods to perform semantic segmentation and semi-supervised segmentation (Lv et al., 2023; Martins et al., 2020; Zhang et al., 2018b; Tong et al., 2018; Chen et al., 2023). In such cases, a quick and reliable segmentation can assist deep learning to achieve better performance (Zhang et al., 2018b; Lv et al., 2018). Therefore, research on region-merging methods is still relevant in the deep learning era, in particular for applications where limited training data is available and the processing time is critical.

One critical challenge in segmentation is computational efficiency, especially in processing large-scale remote-sensing images. For example, it takes at least 9 million steps (50 min processing time on the tested computer) to update weights in segmenting an image of 100M pixels (Lv et al., 2023). Hierarchical Stepwise Optimisation (HSWO) iteratively optimises pixel-wised image segmentation and merges pixels using the region adjacent graph (RAG) model (Beaulieu and Goldberg, 1989), which forms a prototype for region-merging-based optimisation methods. The nearest neighbour graph (NNG) was developed to record the cycles created by the lowest-weight edges from node edges so that the least weighted edge can be found in a rapid manner (Haris et al., 1998). To speed up the processing efficiency, efforts have been made to create segmentation models suitable for multithreading or parallel data processing. Algorithm parallelism differentiates algorithms into independently runnable modules and executes them on multicore CPUs, while data parallelism achieves parallel acceleration by processing data in chunks (Wassenberg et al., 2009; Hu et al., 2018).

The merging criterion and the merging order in region-merging procedures are the two main elements that affect the efficiency of region-merging-based segmentation. Scholars have adopted a variety of merging features, including segment area, mean band values, spectral homogeneity, compactness, and shape features (Chopra et al., 2005; Paris and Durand, 2007; Liu et al., 2012; Zhang et al., 2013; Drăguţ et al., 2014). A deep learning-based region-merging was recently proposed to combine deep learning, handcrafted features, and segmentation for high-accuracy segmentation with less than 0.2% of sample data (Lv et al., 2023). Besides, various merging criteria have been handcrafted to reduce weight updating computation complexity. However, the segmentation accuracy is often inversely proportional to the complexity of the merging criterion design. The global best merging strategy (Beaulieu and Goldberg, 1989), the best local-merging strategy (Baatz, 2000), and the local mutual-best merging strategy (Castilla et al., 2008) are the three mostly used merging strategies, with varying segmentation efficiencies (Zhang et al., 2019). These handcrafted merging criteria, however, are limited by a tradeoff between segmentation precision and segmentation efficiency. In addition, after the NNG method, most methods that aim to increase computational efficiency are generally designed for special cases with poor generalisability.

To address the aforementioned challenges, we propose a generic dynamic pruning framework to systematically improve the efficiency of remote sensing image segmentation algorithms. Pruning is an effective tool in many data analysis fields such deep learning (Liu et al., 2021; Lin et al., 2022), machine-learning (Gelfand et al., 1989; Salembier and Foucher, 2016; Tochon et al., 2015), top-down image segmentation methods (Wassenberg et al., 2009), searching algorithms (Harabor and Grastien, 2011), compressing models (Vo et al., 2009), and so on. Pruning is utilised to reduce the size of models (Liu et al., 2021; Lin et al., 2022; Vo et al., 2009), improve the efficiency (Liu et al., 2021; Lin et al., 2022; Wassenberg et al., 2009), optimise classification models (Gelfand et al., 1989; Salembier and Foucher, 2016; Harabor and Grastien, 2011). In spired by the pruning and divide-and-conquer strategy, we designed the pruning framework for the image segmentation. To the best of our knowledge, we are the first to apply it in region-merging methods, which are typically bottom-up methods. In our preliminary study, we found that the frequent update of edge weights is one of the main factors affecting segmentation efficiency. Thus, reducing the number of edge weight updates is the main objective of the proposed framework. The basic principle is to reduce the number of weight updates by pruning the RAG model, thus significantly improving the segmentation efficiency while maintaining the segmentation accuracy. As a result of pruning, a RAG will be partitioned into many sub-models, allowing the method to be executed on multicore devices. The distribution of individual sub-RAGs over many CPU cores enhances the algorithm's performance.

We implemented the pruning framework for six region-merging-based segmentation methods and validated the performance on four test sites. We utilised the F value (Zhang et al., 2015) and qualitative assessment to validate the effectiveness of the proposed framework to preserve the original segmentation accuracy, and we used the speed-up indicator to evaluate the enhanced computational efficiency of segmentation algorithms. The proposed pruning framework contains intra-object and inter-object pruning modules with two main contributions:

- Generic dynamic pruning framework for bottom-up segmentation methods, whose scales rely on the similarity (or discrepancy) between segments. The proposed framework improves segmentation efficiency without sacrificing segmentation accuracy. For the single-core mode, the computational time of each tested algorithm is enhanced by two to ten times, and the speed-up in the multicore mode can increase up to 40 times with eight CPU cores. On the large-scale image, the single- and multicore modes achieve speed-ups of 36.07 and 102.74, respectively.
- A improvement of the performance of different segmentation algorithms by reducing time complexity while maintaining space complexity, a measure of the memory size temporarily used by the algorithm. The proposed framework can partition a RAG model into sub-RAG models and allows computation on multiple cores, thus fully exploiting the performance of multicore machines.

2. Methodology

The workflow of the proposed dynamic-pruning framework for remote sensing image segmentation is presented in Fig. 1. Fig. 1(a) depicts the processing of conventional region-merging-based segmentation methods, where remote sensing images are first partitioned into primitive segments as initial segmentation via a standard segmentation

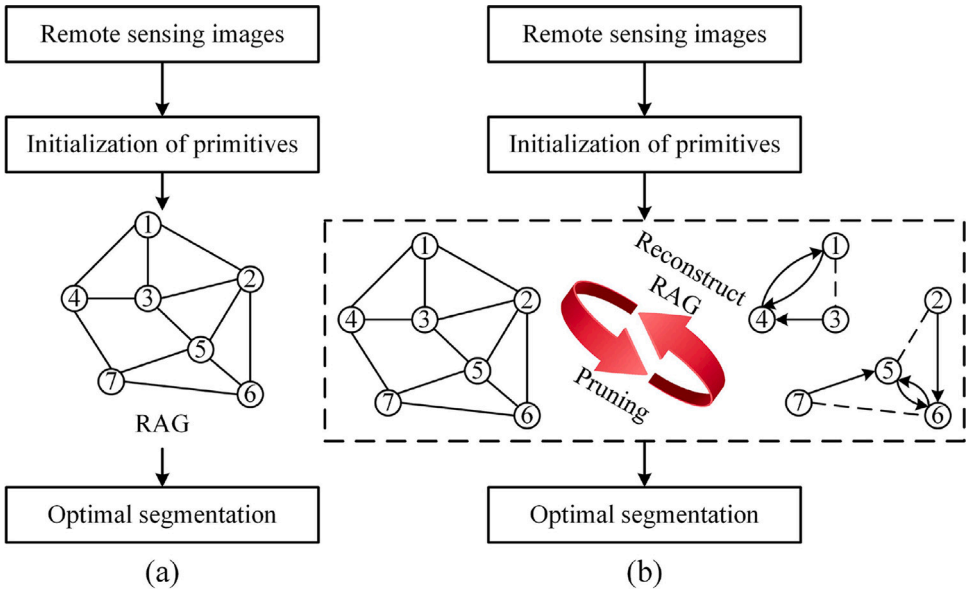

Fig. 1. The workflow of a typical segmentation and pruning framework. (a) Conventional workflow of segmentation methods; (b) Workflow of the proposed segmentation framework using inter-object and intra-object pruning.

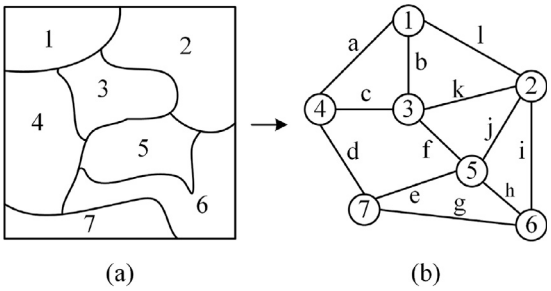

Fig. 2. The concept of the RAG model. (a) the primitive segments; (b) the non-directional graph corresponding to the primitive segments in (a).

method. Traditionally, a RAG-NNG model, usually with low efficiency, can be built based on primitive segments to output optimised segmentation based on the handcrafted merging criteria. The proposed dynamic pruning depicted in Fig. 1(b) improves the segmentation efficiency, as it serves as a replacement for the inefficient RAG-NNG model in typical region-merging processing modes. The proposed dynamic pruning process contains several major steps (Fig. 1(c)), including constructing global-RAG, pruning, generating local-RAG, and reconstructing RAG, where $\times L$ means that the module has to be applied L times iteratively. We describe the principles of the RAG and NNG models and the steps of the proposed method in the following sections.

2.1. The rationale of the RAG-NNG model

RAG is a graph data structure used to record segments and their spatial relationships. Fig. 2 depicts the RAG model for a sample over-segmentation case with seven primitive segments. An un-directional graph $G=(V, E)$ can be constructed considering the seven segments as nodes and the connections between them as twelve edges (Fig. 2(b)). Nodes are represented by v_i (i=0,1, …,7), and edges are symbolised by e_j (j=a,b, …,l).

The weight of each edge denotes the similarity between nodes at both ends. Thus, the region-merging-based segmentation methods can be implemented by merging the most similar segment pairs via searching for the least weighted edge iteratively. After each merge, the weights of the edges connected to the new node need to be updated. For example, assuming e_a is the least weighted edge in the graph, v_1 and v_4 are the most similar segments to be merged. The weights of edges e_b, e_c, e_d, and e_l connected to the new node must be updated. Such a protocol causes two inefficiency issues. One is the time complexity of searching the least weighted edge, which demands extensive computation resources in a large graph. Given a set of edges with size n and a total number of m regions to be merged, the time complexity of the least weighted edge search using existing methods is $O(n \times mlogn)$, where the first term is the cost to construct the graph, while the second term is the cost to maintain the priority queue (Wu, 1993). The region merging with this time complexity usually fails to meet the high-efficiency requirements of many image-segmentation tasks. The other issue is the unnecessary weight update of some edges, which also leads to additional computational demand. Haris et al. (1998) proposed the NNG model to address the least weighted edge search issue. Fig. 3 presents the construction of the NNG model.

The NNG is a directed graph where the edge with the lowest weight among the edges connected to a node define the node's direction edge. For instance, assuming e_c is the least weighted edge among the edges connected to v_3 the edge direction of v_3 is e_c from v_3 to v_4 (Fig. 3(a)). Each node has only one edge direction. Edge directions of each node can be defined by the lowest-weight edge connection (Zhang et al., 2014). As we can see in Fig. 3(b), in the data structure composed of nodes connected by their directional edges, only one cycle eventually remains in one NNG model such as node cluster of v_1, v_3, v_4 and the other cluster of v_2, v_5, v_6, v_7. Therefore, the weights of the cycles are the weights of related edges. The data structures depicted in Fig. 3(b) result in two cycles stored in a minimum priority queue by the weights of the cycle (Haris et al., 1998; Zhang et al., 2014). According to the generation principle of cycles, the least weighted edge in the RAG must be stored in these cycles. Hence, the size of the edge set to be searched is reduced from an initial twelve edges to only two edges (Fig. 3(c)). After searching for the least weighted edge, a new node $v_{1,4}$ is merged from the corresponding nodes v_1 and v_4. By updating the weights of edges connected to $v_{1,4}$, and calculating the direction of $v_{1,4}$, a new cycle is extracted and pushed into the minimum priority queue. Although the NNG model addresses the least weighted edge search issue, the unnecessary weight updating remains an issue which will be addressed in the following sections.

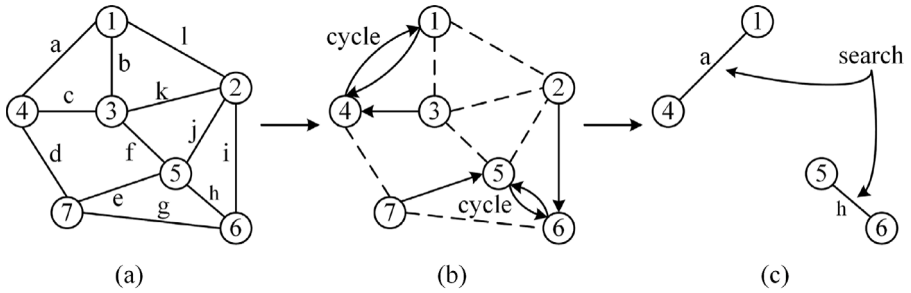

Fig. 3. Comparing NNG and RAG models. (a) the RAG model; (b) the construction process of NNG from RAG; (c) the NNG model.

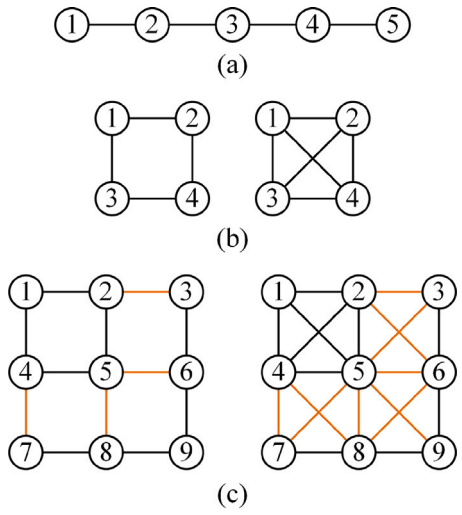

Fig. 4. Ideal state examples of RAG models. (a) a linear RAG model; (b) a RAG model with cycles; (c) a complex RAG model with two hypothetical objects (object 1 contains nodes v_1, v_2, v_4, v_5 and object 2 contains nodes v_3, v_6, v_7, v_8, and v_9). The orange lines mean that the nodes at their ends owe to different objects and thus should not be merged. On the contrary, the black lines indicate the end nodes belong to the same object, and should thus be merged. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

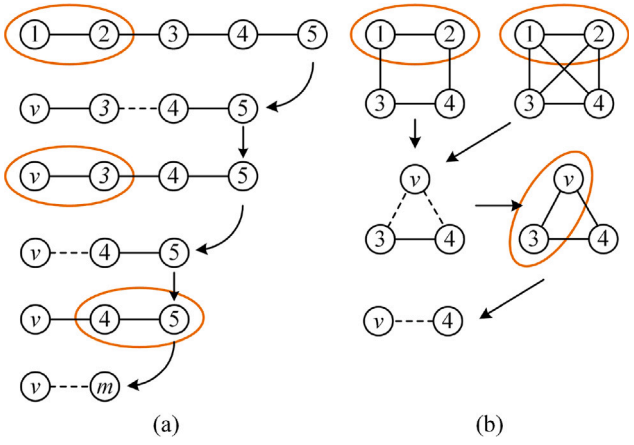

Fig. 5. Example merging steps in RAG models. The nodes in the circle are candidate nodes to be merged. The weights of dash lines require weight updating after each merging. (a) is the merging process of Fig. 4(a) requiring three weight updates. (b) is the merging process of Fig. 4(b), require three weight updates.

2.2. Computational complexity of the RAG model

The complexity of the RAG models varies depending on the data types, segmentation algorithms, and parameter settings; nevertheless, their inefficiency is due to frequent updating of the edge weights. We chose three examples of the RAG models to analyse and discuss their computational complexity (Fig. 4).

The sample RAG model in Fig. 4 (a) contains five nodes and four edges. The graph on the left of Fig. 4(b) consists of four nodes and four edges that create a ring. On the right of Fig. 4(b), each node is connected with others in the ring. The two RAG models in Fig. 4(c) are combinations of basic graphs from Fig. 4(a) and (b), demonstrating high structural complexity.

The standard RAG model includes inter-object weight update and intra-object weight update redundancies. The inter-object weight updates refer to the weight updates between adjacent segment pairs forming different objects in the final segmentation results. These weights are often higher than the user-defined threshold, and participate in weight updates in the standard RAG model, but unnecessary. The intra-object weight updates refer to that the weight updates between segment pairs composing the same object in the final segmentation. These weights are usually lower than the threshold, but their updates are too frequent to be merged efficiently.

Assuming that nodes connected by black edges are to be merged into one object, the graph in Figs. 4(a), 4(b) left, and 4(b) right will be merged into the same object, respectively. The graphs shown in Fig. 4(c) will be merged into two objects (one object contains nodes v_1, v_2, v_4, v_5, and the other contains nodes v_3, v_6, v_7, v_8, and v_9). The steps of initialising weight for each edge correspond to the edge number in the RAG model. Updating steps represent the number of weight updates required within the merged segments. Excess steps are the number of unnecessary weight updates between merged segments. In Fig. 4(c), notably, the weight update of orange edges is excessive. Different merging orders result in varying merging efficiencies. The merging process in Fig. 4(a) and (b) is exhibited in Fig. 5(a) and (b), respectively. We can conclude that the updating steps of the RAG mode in Fig. 4(a) are three (Fig. 5(a)). Similarly, both the left and right updating steps in Fig. 4(b) are three (Fig. 5(b)).

The RAG on the left in Fig. 4(c) is formed of Figs. 4(a) and 4(b) left, whereas the RAG on the right in Fig. 4(c) is composed of Figs. 4(a) and 4(b) right. Thus, the merging steps of RAGs on the left and right in Fig. 4(c) are both six. However, the unnecessary steps of edge weight updating are quite different in RAGs. The unnecessary steps in Figs. 4(a) and 4(b) are zero. The orange edges in the RAGs in Fig. 4(c) are connected by two objects, indicating that the weight updating of these edges is unnecessary and redundant. Table 1 shows the details regarding the initial steps, updating steps, unnecessary steps, and total steps of edge weight updates in Fig. 4. We notice that the unnecessary steps account for a large ratio of the total steps (12/30 and 16/32). In other words, the inter-object weight updates are responsible for the low computational efficiency of the RAG model.

On the other hand, intra-object weight update also reduces the efficiency of the RAG model. For example, an object in the sample segmentation is a data structure consisting of a complete binary tree, whose leaf nodes are initial segments (Hu et al., 2017; Zhang et al.,

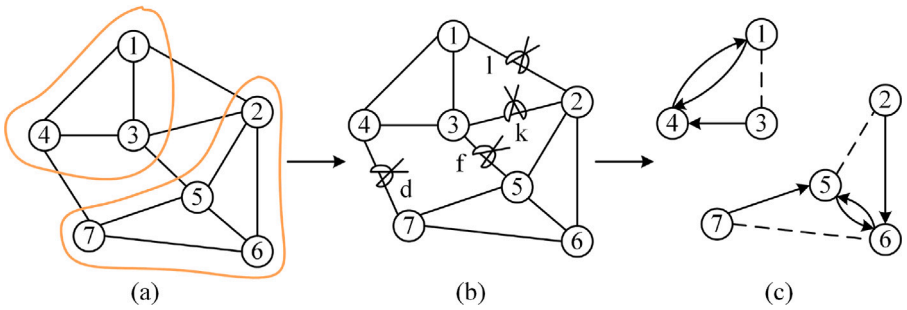

Fig. 6. Process of inter-object pruning. (a) the RAG model and the yellow circles mean two objects; (b) inter-object pruning; (c) the NNG model. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Statistics about weight updates in scenarios that are reported in Fig. 4.

RAG	Initial steps	Updating steps	Unnecessary steps	Total steps
Fig. 4(a)	4	3	0	7
Fig. 4(b) left	4	3	0	7
Fig. 4(b) right	6	3	0	9
Fig. 4(c) left	12	6	12	30
Fig. 4(c) right	20	6	12	30

2020). Under ideal conditions, the optimal weight updating frequency is only once each merge (for constructing a complete binary tree). Supposing that n denotes the node number and l denotes the leaf node number, the merging steps is $(n - l)$. However, given the complex spatial topological relationships between segments, such a condition is impractical. Thus, the weight updates in the object composed of multiple segments would contain many redundancies. For an object, the more segments it has, the more serious the problem is. In other words, the intra-object weight updates are also responsible for the low computational efficiency of the RAG model.

To address the two issues, we proposed a dynamic-pruning framework, which includes inter-object pruning and intra-object pruning modules, to optimise the standard RAG model. The following sections introduce the pruning framework in detail.

2.3. Inter-object pruning

The rationale of inter-object pruning is to prune the edge connecting two segments whose weight (the similarity between two neighbour segments) exceeds a user-defined threshold. When the global RAG model is updated, the edge weight frequently increases following user-defined region-merging criteria. Therefore, following the initialisation of the RAG model, the update of the weights higher than the defined scale becomes redundant. We prune the edges with weights higher than the defined scale. Fig. 6 exhibits the concept of inter-object pruning. Fig. 6(a) shows a global RAG model the same as Fig. 2. Assuming that the RAG is finally merged into two segments (indicated by the nodes in orange circles) and the weights of edge e_d, e_f, e_k, and e_l are larger than the scale, we prune these edges in Fig. 6(b) without future weight updates. In the final step, we construct the local RAG models on the resulting subgraph respectively (Fig. 6(c)). As a result of the inter-object pruning, the updating times in Fig. 6(a) significantly reduce.

The local RAGs in Fig. 4(c) are divided into two local RAGs after pruning the orange edges. The updating excess times related to the global RAG drop from twelve and sixteen to zero. The theoretical merging efficiency increases by 40% and 50%, respectively. However, the RAG in the actual situation is more complex than the ideal state. For instance, the weights of edges connecting to the new node $v_{5,6}$ merged by v_5 and v_6 must be updated. The edge weights between v_2 and $v_{5,6}$ or v_7 and $v_{5,6}$ may be greater than the scale. In addition, the edge weight between v_3 and $v_{5,6}$ can also become a lower value,

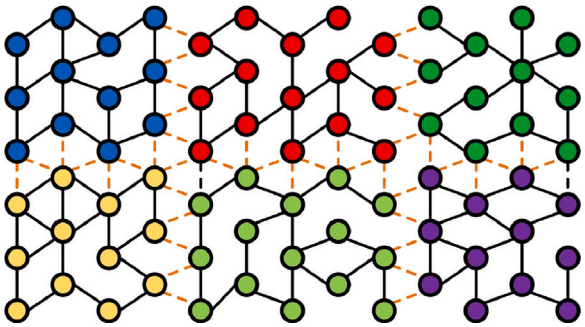

Fig. 7. The division of local RAGs using breadth-first search. The six colours in nodes demonstrate six sub-RAGs. The dash lines mean the related edges are pruned by intra-object pruning.

despite its pruning. Thus, the initial inter-object pruning sometimes fails to produce the optimal merging outputs. We will reconstruct a global RAG model composed of the local RAG models based on known spatial relationships to improve the inter-object pruning performance. Then, we iterate the pruning steps described above for the new global RAG model. Such a design improves the merging efficiency and ensures optimal merging results.

2.4. Intra-object pruning

The actual steps of weight updates are substantially more than the theoretical minimum in an object according to Section 2.2. After inter-object pruning for a global RAG, the local RAGs have potential to become objects in the final segmentation. For example, a large local RAG may finally become one large object. The object contains many initial segments, thus lots of redundant edge weight updates occur in the merging process. Therefore, these large RAGs hinder significantly the merging efficiency. We therefore design an intra-object pruning to reduce the redundant weight updates of edges, i.e., the optimisation of the local RAG. Let us note that the greater the compactness of an object, the shorter its perimeter, e.g., the perimeter of the circle is the shortest among the objects with the same area. A shorter perimeter means fewer connections to other objects. According to this principle, we are going to divide large RAGs into small RAGs with high compactness and the same size. We utilise the breadth-first search (BFS) algorithm (Zhang et al., 2018a) to divide the large local-RAG into more equal-size RAGs. In this way, computing resources can be evenly allocated to each local RAG, and in the next iteration, because of the reduced number of connections, the number of weight updates to rebuild the global RAG is smaller. The BFS searches over the local-RAG until the count of searched nodes reaches a user-defined threshold. These nodes are used to construct a new local-RAG, and the next round of BFS can start on the unsearched nodes until all nodes are searched. Fig. 7 shows the division of a local RAG with six clusters of equal size. Thus, the local native RAG

Fig. 8. Four testing sites with very high-resolution images of T1 (a), T2 (b), T3 (c), and T4 (d) shown with red, green, and blue bands. The regions with green outlines are reference objects. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

is divided into six new local RAGs, resulting in the six segments in the native RAG after this iteration. The six segments are further constructed in one local RAG in the next iteration.

In the intra-object pruning, lots of edges with weights lower than the user-defined threshold are pruned. However, the nodes at the ends of these edges can have high similarity to be merged. Therefore, both intra- and inter-object pruning require re-constructing RAG and iterate the pruning process. In addition, a global RAG is divided into even local RAGs via inter-object pruning and intra-object pruning, thus demonstrating the possibility to parallelise the algorithm for multicore implementation. The Pseudo-1 codes of the framework are reported below:

The inputs include the primitive segments S_0, the scale parameter, the number of iteration L, the number of cores T, and the searching size of BFS. First, a *global-RAG* can be constructed based on the S_0 using RAG constructing function Construct-RAG(*). *Scale-series* are defined by input scale, and are applied in each iteration. The *global-RAG* is divided into multiple *local-RAGs* via Inter-object-pruning(*) function at the *begin-scale*, where some large *local-RAGs* continue to be separated using intra-object pruning (BFS(*) function). We distribute the *local-RAGs* evenly over T worker cores based on Multi-cores(*) function. Further, we can obtain the optimal merging results using the *local-RAGs* by conducting Merging(*) function. We then reconstruct *global-RAG* based on the *local-RAGs* by Re-construct-RAG (*) function and iterate the above processing until the iteration times meet the required parameter L. The spatial complexities of the RAG model before and after the addition of pruning remain $S(n+k)$, where n is the number of nodes and k is the number of edges. The time complexity after pruning becomes $O(n-r+m\log n)$, where r is the redundancy of weight updating related to the segmentation algorithms.

2.5. Tested methods

We apply the proposed dynamic-pruning to prune six existing segmentation methods based on region merging to exhibit its robustness and efficiency. They involve unsupervised segmentation, supervised segmentation, standard methods, deep learning, and various merging criteria. The hierarchical step-wise optimisation (HSWO) method is an earlier region-merging based optimisation method and is widely applied in remote sensing images (Beaulieu and Goldberg, 1989). Best merging region-growing segmentation (HSeg) first integrates nonadjacent geo-objects aggregation via bottom-up strategy (Tilton et al., 2012). The Boundary-Constrained Multi-Scale Segmentation (BCMS) method utilises the edge penalty to improve the segmentation performance (Zhang et al., 2013). Fast hierarchical segmentation (FHS) adopts a linear nearest neighbour graph and adaptive edge strength (Zhang et al., 2014) method, employs a local spectral angle threshold for region-merging (Yang et al., 2017). The deep region merging (DeepMerge) method integrates a deep learning model and region-merging based segmentation on a large scale dataset of high spatial

Algorithm 1: Pruning framework

input : Primitive segments S_0, *scale*, iteration times L, core number T, *size* of the BFS
output: The optimisation result S_L based on the region-merging

```
global-RAG ← Construct-RAG(S_0);
scale-series←[scale_0,scale_1,...,scale_k] where k ← L-1, scale_0 <= scale_1 <=...<= scale_k = scale;
i←0;
while i<L do
    begin-scale←scale-series[i];
    local-RAGs←Inter-object-pruning(global-RAG,begin-scale);
    new-local-RAGs←[];
    j←0;
    node-count←Count(local-RAGs);
    while j<L do
        size←Count(local-RAGs[j]);
        if size>node-count then
            sub-RAGs←BFS(local-RAGs[j],size);
            new-local-RAGs.Add(sub-RAGs);
        else
            new-local-RAGs.Add(local-RAGs[j]);
        j=j+1;
    local-RAGs←Multi-cores(new-local-RAGs,T);
    S_{i+1},local-RAGs←Merging(local-RAGs,scale);
    global-RAG←Re-construct-RAG(local-RAGs);
    i=i+1;
final;
return S_L;
```

resolution remote sensing images (Lv et al., 2023). The aforementioned methods are selected as competing methods to test our proposed dynamic-pruning framework. The details involving merging criteria and description of these methods can be found in Table 2.

MC is the edge weight between two adjacent segments. a_1 and a_2 are the areas in pixels of the segment pair, and u_1 and u_2 are the mean spectral values of the pair (Beaulieu and Goldberg, 1989). b is the number of bands. v_{1i} and v_{2i} are the mean value of ith band (Tilton et al., 2012). $CComp$ is the change of compactness. ES is the edge strength and $CStd$ is the change of homogeneity defined by standard deviations Std_1, Std_2 and areas a_1 and a_2 of two regions. Std is the standard deviation of the hypothetical merged region (Zhang et al., 2013). L_1 and L_2 are the perimeters of two regions, and L is the perimeter of the hypothetical merged region. ε is a user-defined parameter adjusting the strength of edge penalty (Zhang et al., 2014). H_I is the relative value of the initialised area weight of the segment. $s_1 s_2$ is the angle of the segment pair, the same as HSWO. $P(x, y)$ is the pixel value in the segments' location (x, y). $stdv$ is the standard deviation of the segments (Yang et al., 2017). f_l and f_r represent the feature vector in the two segments, and i indicates the ith feature item from the vector (Lv et al., 2023).

The parameter updating, such as area, perimeter, mean value, and standard deviation value after merging two regions, are also important factors that affect segmentation efficiency. The parameters in the newly merged region updated directly based on the native pixels can cause low efficiency in the whole merging process. Therefore, we exhibit the efficient parameter updating methods based on the native parameters before merging in the following equations:

$$a = a_1 + a_2 \tag{1}$$

$$L = L_1 + L_2 - 2l \tag{2}$$

$$u = \frac{1}{a_1 + a_2}\left(a_1 u_1 + a_2 u_2\right) \tag{3}$$

$$Std = \sqrt{\frac{a_1 Std_1^2 + a_2 Std_2^2}{a_1 + a_2} + \frac{a_1 a_2 \left(u_1 - u_2\right)^2}{\left(a_1 + a_2\right)^2}} \tag{4}$$

$$f_i = \frac{1}{M + N}\left(M f_i^l + N f_i^r\right) \tag{5}$$

$$ES = \frac{1}{l_1 + l_2}\left(l_1 ES_1 + l_2 ES_2\right) \tag{6}$$

where a, L, u, Std, and f_i are the area, perimeter, mean value, standard deviation value, and features of the merged region, respectively. l is the length of the intersected edge between two regions. M and N are the number of feature vectors in two regions, respectively. ES_1 and ES_2 are the edge strength between two regions and the third region. ES is the edge strength between the merged region and the third region, in which l_1 and l_2 are the edge length between the third and the first two regions, respectively.

2.6. Performance metrics

To validate the segmentation performance with pruning and no pruning, we choose to employ the F values (Zhang et al., 2015), used in the superpixel segmentation estimation, which is presented in the following. S is the set of polygon segmentation results containing M segments $\{S_1, S_2, \ldots, S_M\}$, and R is the set of polygons containing N reference geo-objects $\{R_1, R_2, \ldots, R_N\}$. $|*|$ function outputs the area of a segment. $R_{i,max}$ denotes the largest area reference geo-object related to the segment S_i, and $S_{i,max}$ denotes the largest area segment related to the reference geo-object R_i. R_{ij} denotes the set of segments related to R_i. $S_i \cap R_{i,max}$ is the intersection of S_i and $R_{i,max}$. α is often set as 0.5.

$$|S| = \sum_{i=1}^{M} |S_i| \tag{7}$$

$$precision = \sum_{i=1}^{M} \left|S_i \bigcap R_{i,\max}\right| / |S| \tag{8}$$

$$|R| = \sum_{i=1}^{N} \left|R_i\right| \tag{9}$$

$$recall = \sum_{i=1}^{N} \left|R_i \bigcap S_{i,\max}\right| / |R| \tag{10}$$

$$\mathrm{F} = \frac{1}{\alpha \frac{1}{precision} + (1 - \alpha)\frac{1}{recall}} \tag{11}$$

To validate the performance of the proposed dynamic-pruning algorithm, we chose to employ the speed-up factor, often applied in the field of parallel processing, to quantify the efficiency improvement of segmentation methods before and after the addition of the proposed pruning framework (BalkanskiF et al., 2019). The speed-up factor is defined as follows:

$$S_p = \frac{T_0}{T_p} \tag{12}$$

where T_0 is the runtime of the native algorithm without pruning, and T_P is the runtime of the segmentation methods using the dynamic-pruning framework. P is the number of the running cores. S_P is the modified speed-up factor, with a higher value indicating the higher efficiency of the dynamic-pruning framework. When S_P is equal to P, S_P becomes linear speed-up. If S_P is larger than P, it is called super-linear speed-up. Super-linear speed-up rarely occurs during the improvement of the algorithm efficiency. In addition, the count of weight update (merging steps plus excess steps in Section 2.6) except for initialising weight steps is applied to measure the computational volume of the segmentation methods. The code of the proposed pruning framework is available at https://pan.baidu.com/s/1YlCV7gLWyJNxcdK05dAgA?pwd=1234 (extraction code: 1234).

Table 2
Merging criteria of the tested methods.

Methods	Merging criteria	Citations
HSWO	$MC = a_1 a_2/(a_1 + a_2)(u_1 - u_2)^2$	Beaulieu and Goldberg (1989)
HSeg	$MC = \arccos\left(\sum_{i=1}^{b} v_{1i} v_{2i} / \left(\left\|\sum_{i=1}^{b} v_{1i}\right\|_2 \left\|\sum_{i=1}^{b} v_{2i}\right\|_2\right)\right)$	Tilton et al. (2012)
BCMS	$MC = (a_1 + a_2)(CStd + CComp)\, ES$ $CStd = Std - (a_1 Std_1 + a_2 Std_2)/(a_1 + a_2)$ $CComp = L/\sqrt{a} - \left(a_1 L_1/\sqrt{a_1} + a_2 L_2/\sqrt{a_2}\right)/(a_1 + a_2)$	Zhang et al. (2013)
FHS	$MC = (a_1 + a_2) \cdot CStd \cdot \exp(-\epsilon/ES)$	Zhang et al. (2014)
Local-SA	$MC = 1/H_I \cdot \theta_{s1s2} \underset{x,y \in (s1,s2)}{stdv} \left(\frac{1}{b}\sum_{i=0}^{b} P^i_{(x,y)}\right)$	Yang et al. (2017)
DeepMerge	$MC = \left\|f_i^l - f_i^r\right\|_2$	Lv et al. (2023)

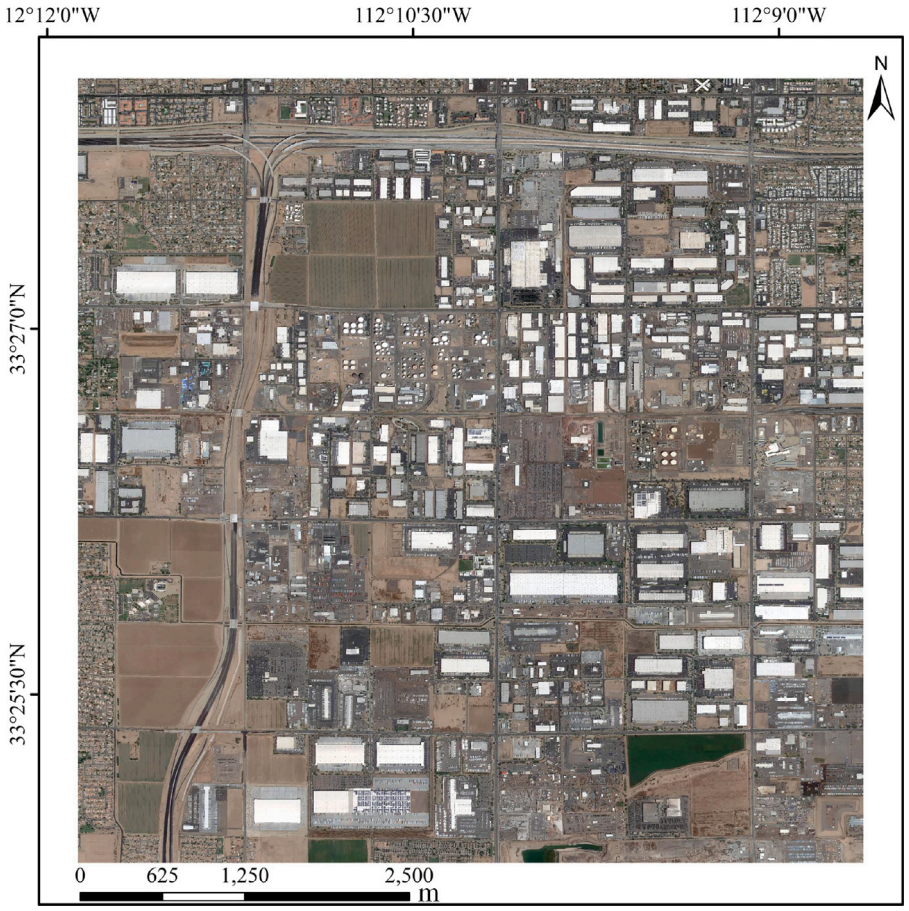

Fig. 9. The study area for large-scale test with 100M pixels and 0.55 m resolution.

3. Study area and experimental results

The pruning segmentations are tested to validate the effectiveness and robustness of the proposed dynamic-pruning framework. In particular, quantitative metrics and visual analysis are used to validate the performance and efficiency of the six segmentation methods before and after adding the dynamic-pruning framework.

3.1. Study area

The segmentation methods for pruning have been proven to be effective and robust in various remote sensing images from different satellite sensors with medium resolution (Beaulieu and Goldberg, 1989; Tilton et al., 2012; Zhang et al., 2013; Chen et al., 2015; Yang et al., 2017). To make the experiment more challenging for the proposed dynamic-pruning and other competing algorithms, we chose to use very high resolutions images. Four Google Earth images (Yu and Gong, 2012) with diverse landscapes in Phoenix City, Arizona, U.S., are selected as test sites, resulting in RAG models with widely varying internal structures (Fig. 8). The four images, named T1 (Fig. 8(a)), T2 (Fig. 8(b)), T3 (Fig. 8(c)), and T4 (Fig. 8(d)) are composed of red, green, and blue bands with 0.55-meter resolution and encoded as 8-bit integral values. The sizes of T1, T2, T3, and T4 are 16.6M, 11.0M, 19.8M, and 8.0M pixels, respectively. The dominant land covers in T1 are continuous bare soils, residential sites, and roads, with small patches of vegetation in the rural zone. The landscapes in T2 cover residential houses with various textures, asphalt roads, and vegetation in the urban zone. The land covers in T3 reflect factory buildings of different sizes and textures, residential buildings, and roads. The dominant land cover categories in T4 are factory buildings, as it mainly covers an industrial zone.

The green polygons in the four images are manually digitised as reference geo-objects. We manually digitised the reference geo-objects according to the dominant categories that contain roads, residential

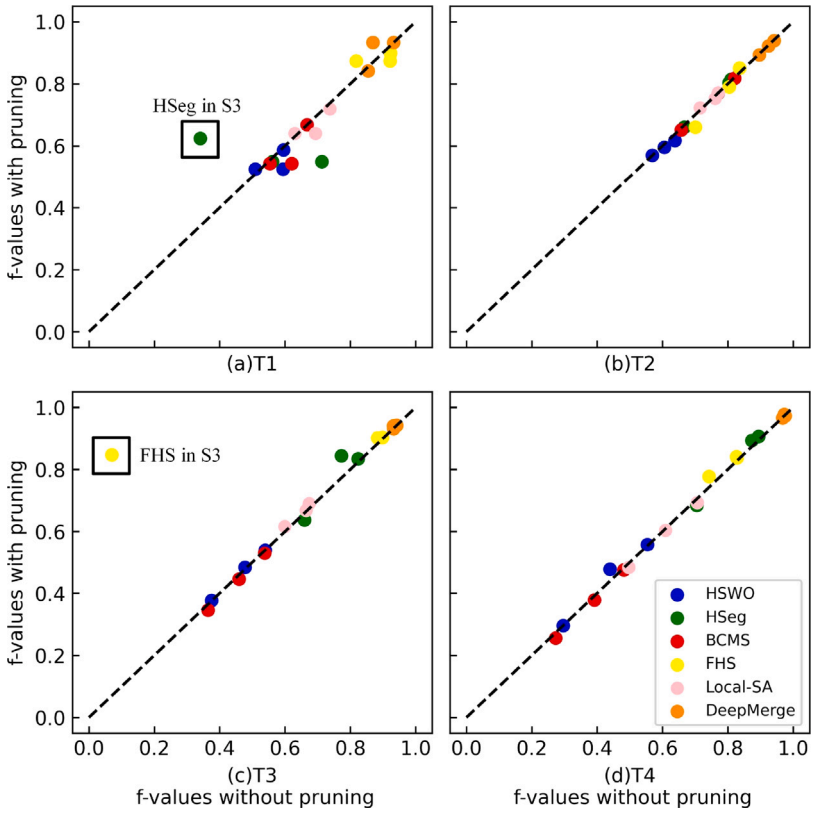

Fig. 10. The scatter points of segmentation results with- and without pruning. The black dashed line denotes the 1:1 reference line. The two points in the black rectangles are two extreme cases, which are the results of HSeg in S3 and FHS in S3, respectively.

houses, vegetation, water areas, factory buildings, bare soils, and side roads in each image. The number of digitised reference objects in T1, T2, T3, and T4 is 89, 305, 270, and 108, respectively.

Multi-resolution segmentation (MRS), as an effective standard segmentation method, is applied to initialise the primitive segments. To generate suitable over-segmentation results, the scale parameter of MRS is set to 25, and the shape and compactness are set to 0.5. The RAG in T1 contains 43,707 nodes (primitive segments) and 128,798 edges (connections between neighbour segments). The RAG constructed in T2 involves 42,030 nodes and 122,096 edges. The RAG initialised in T3 is composed of 76,061 nodes and 222,215 edges. The initial RAG used in T4 consists of 29,802 nodes and 87,117 edges.

To test the performance of the pruning framework, we compared the six pruning methods on a large image with 100M pixels and 0.55 m resolution. The image contains various objects such as residential houses, factory buildings, large-area bare soils, vegetations, and continuous roads, demonstrating the high complexity of the image. Fig. 9 shows the details of the image for very large-scale test.

Table 3
Scale parameters of each method in this study.

Method	S1	S2	S3	Citations
HSWO	8,700	67,745	195,317	Beaulieu and Goldberg (1989)
HSeg	0.2	0.5	0.7	Tilton et al. (2012)
BCMS	2,990	5,768	9,620	Zhang et al. (2013)
FHS	50	600	1210	Chen et al. (2015)
Local-SA	1.0000	1.5161	2.3374	Yang et al. (2017)
DeepMerge	0.3	0.5	0.7	Lv et al. (2023)

Table 4
Comparison of average F values of segmentation results in study areas.

Method	S1	S1+pruning	S2	S2+pruning	S3	S3+pruning
HSWO	0.437	0.442	0.529	0.533	0.582	0.575
HSeg	0.648	0.633	0.809	0.806	0.699	0.794
BCMS	0.462	0.449	0.561	0.557	0.627	0.623
FHS	0.820	0.851	0.863	0.866	0.630	0.812
Local-SA	0.624	0.628	0.682	0.691	0.708	0.706
DeepMerge	0.932	0.931	0.928	0.945	0.925	0.925

3.2. Segmentation accuracy

To reveal the capability of the proposed dynamic pruning in maintaining segmentation accuracy, we compared the segmentation results of six existing region-merging-based segmentation methods (mentioned in Section 2.5) at three scales from quantitative and qualitative perspectives. The scale parameters of segmentation methods vary under different merging criteria, shown in Table 3. S1, S2, and S3, denoting three different scales, are determined by trial-and-error or author recommendation. The proposed Dynamic-pruning is implemented using C# programming language and tested on a desktop computer with Windows 10 OS, an intel i7-10700 CPU (2.9 GHz, 8 Cores), and 32 GB RAM.

Within three scales, we obtained six segmentation results (pruning and no pruning) for each method. Table 4 presents the average F values of the six segmentation results of each method on study areas. "Pruning" denotes the application of dynamic pruning. From Table 4, we notice that the proposed dynamic pruning has achieved similar performance with no-pruning in the study areas. The segmentation accuracy fluctuates with and without pruning, but a little.

According to the above quantitative measures, we notice that the average F values improved by 0.0029 on average for algorithms that exhibit improved performance after adding the proposed dynamic pruning ignoring the two extreme cases shown in Fig. 10. We observe that the addition of dynamic pruning leads to a positive impact on performance for some algorithms. The standard deviation of the F

Fig. 11. The local comparison of segmentation results with and without dynamic-pruning in T1.

value changing is 0.0303, demonstrating the ability of the proposed dynamic pruning to maintain the segmentation accuracy. Fig. 10 presents scatter plots of F values with- and without pruning in four testing areas, i.e., T1, T2, T3, and T4. We observe that most scatter points are distributed around the reference line, suggesting the high stability of the segmentation accuracy of the algorithm after adding the proposed dynamic pruning strategy.

We further compared segmentation results with and without dynamic-pruning in a qualitative manner. We selected four local areas in each test image to exhibit the segmentation details. Figs. 11, 12, 13, and 14 present the local segmentation details with and without dynamic-pruning in T1, T2, T3, and T4, respectively. We observe that the segmentation results after adding the dynamic-pruning strategy are similar to the ones without dynamic pruning, with some exceptions emphasised by red circles in the four figures. In some cases, the involvement of dynamic pruning enhances the segmentation accuracy.

3.3. Computational speed-up

The segmentation time (in seconds) of each method in three different scales in T1, T2, T3, and T4 are shown in Fig. 15. We notice that the proposed dynamic pruning greatly improves the segmentation efficiency of all methods, on all scales, in all tested images. We observe different levels of improvement in segmentation methods with pruning. The average Sps of DeepMerge with dynamic pruning are 7.25, 3.88, and 7.97, achieving the highest S_p in T1, T2, and T4 (the first, the second, and the last rows in Fig. 15). In T3, the FHS achieves the highest S_p with 12.31 in S3. However, it achieves the lowest S_p with 1.36 in T4. The Sp values range from 1.36 to 12.31 in the single-core mode, demonstrating that the proposed dynamic-pruning can greatly improve the efficiency of image segmentation while maintaining the segmentation accuracy. The proposed dynamic-pruning is proven to maintain the accuracy of segmentation methods with pruning, with a significant improvement in running efficiency.

Both the algorithm and the hardware determine the segmentation time. In order to assess the efficiency gain irrespectively of hardware characteristics, we counted the number of weight updates during the merging process with and without pruning. Fig. 16 shows the statistics about the number of weight updates, which are substantially decreased by the proposed dynamic pruning framework. The main reason for the efficiency enhancement is the significant reduction in the number of

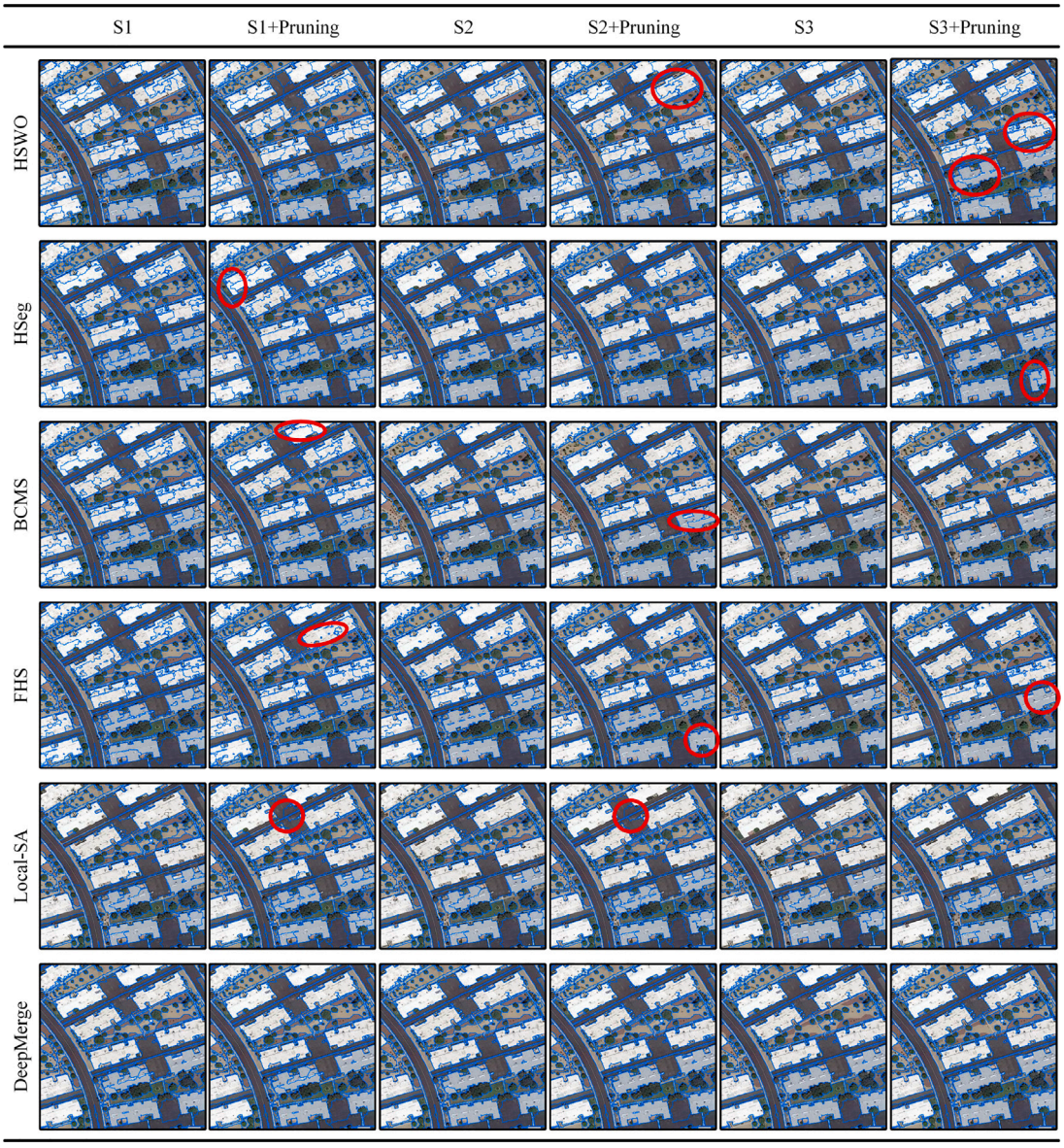

Fig. 12. The local comparison of segmentation results with and without dynamic-pruning in T2.

weight updates in the merging process. The reduction of weight updates of all results in Fig. 16 shows the same trend as in Fig. 15.

4. Discussion

The experimental results are encouraging, as our method preserves the segmentation accuracy and highly improves segmentation efficiency. It can be observed that the pruning method can result in accuracy fluctuations (Fig. 10). This is because the pruning operator can change the region-merging order. In the last iteration of the pruning framework, the global RAG is divided into sub-RAGs, possibly resulting in some adjacent segments with similarity smaller than the scale parameter to be located on different sub-RAGs, thus stopping the merging process. If the hypothetically-merged segment is under-segmented, the accuracy will improve because of the cancellation of the merging step by the pruning operator. On the contrary, the accuracy will drop if the hypothetically-merged segment belongs to one object or is over-segmented. Under-segmentation is serious on the large scales, e.g., S3. The pruning operator can stop merging some neighbouring primitives into under-segmented segments, resulting in a large accuracy distance between before and after the pruning, especially in the two extreme cases: HSeg and FHS on S3 shown in red dashed rectangles in Fig. 10. However, the standard deviation of the F value changing is 0.0303, demonstrating the ability of the proposed dynamic pruning to maintain the segmentation accuracy. Therefore, it will rarely affect subsequent analysis steps. The results demonstrate the effectiveness of the proposed dynamic-pruning framework for region-merging based segmentation methods. The results indicate that the proposed framework has great potential for taking advantage of multicore computers. Based on the rationales of dynamic pruning and region-merging-based methods, three factors affect the segmentation efficiency with dynamic-pruning: (1) the scale parameters of segmentation methods, (2) the number of CPU cores during algorithm execution, and (3) the iteration times of dynamic-pruning frameworks. In this section, we discuss in detail these three crucial factors.

The scale parameter directly affects the efficiency of segmentation methods under pruning. Assuming that only the one merging step was conducted, the provided scale parameter requirement is satisfied

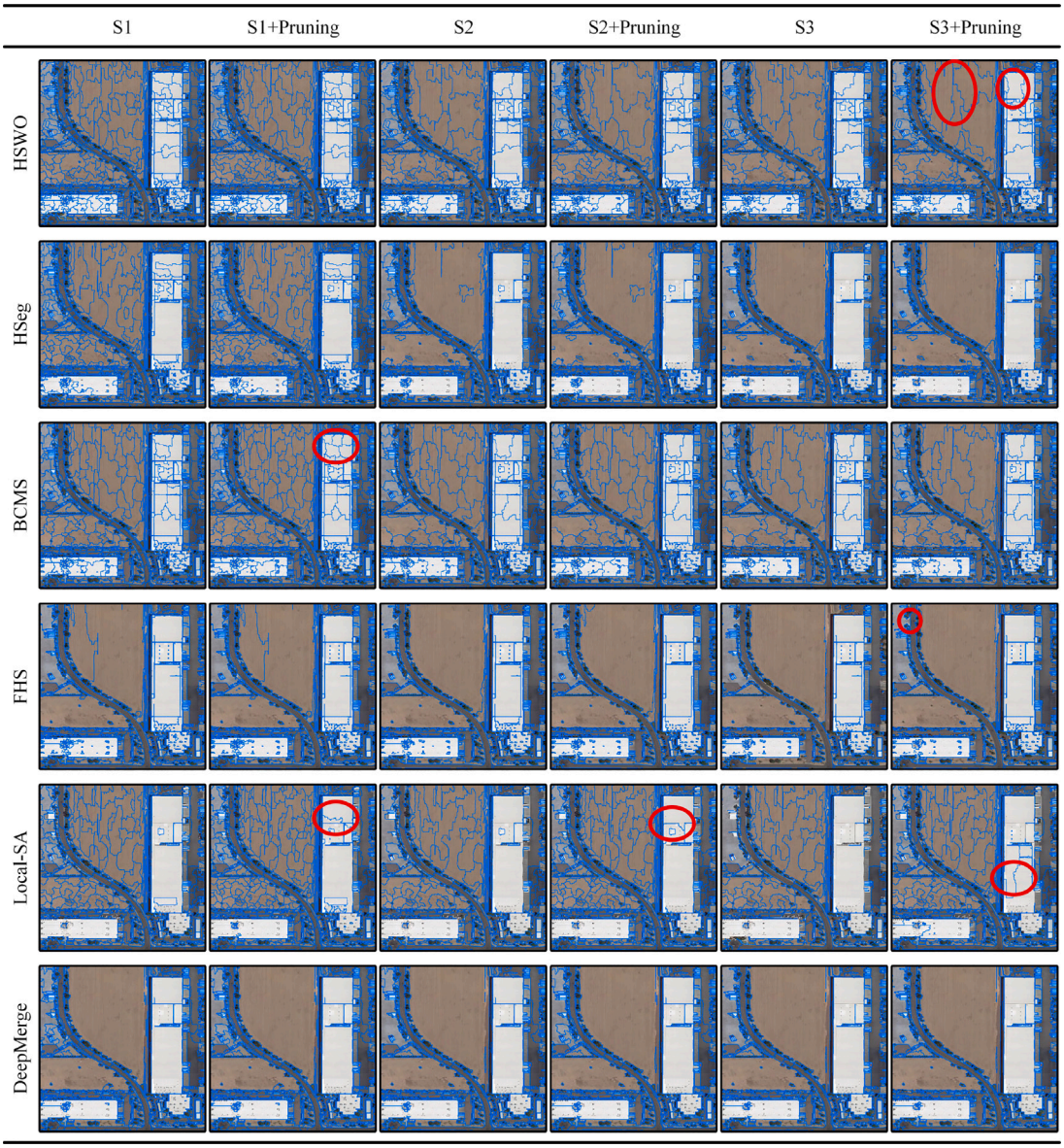

Fig. 13. The local comparison of segmentation results with and without dynamic-pruning in T3.

(with a very low scale parameter). In this case, the pruning framework becomes redundant. Fortunately, in real-world scenarios, the number of segments to be merged is usually large, and these segments are often distributed across the whole image. Thus, the importance of dynamic-pruning can be well-demonstrated. We present the S_p boxplots based on three scales in Fig. 17.

The boxplots in Fig. 17(a) and (b) demonstrate that the segmentation speed-ups of the six pruned segmentation algorithms are very stable on three different scales. Nonetheless, FHS in T3 and DeepMerge in T4 in Fig. 17(c) and (d) exhibit substantial volatility due to the large geo-objects in their segmentation results on fine scales. However, the minimum S_p of both methods reaches 2.0. Overall, scale parameters have a trivial impact on segmentation efficiency. The segmentation efficiency of some specific methods fluctuates wildly in certain testing images.

The proposed data parallel strategy yields identical segmentation results. However, there are significant differences in efficiency between the two modes. We compared the S_p values of dynamic-pruning utilising 1 to 8 CPU cores. Fig. 18 exhibits the S_p values of dynamic-pruning in the multicore mode. The first, second, and third columns in the figure denote the S_p results on scales S1, S2, and S3, respectively. Since the DeepMerge with dynamic pruning achieves significantly higher S_p values than other methods, we put the S_p values of DeepMerge in the fourth column in the figure. The results in Fig. 18 above the dotted line are all super-linear speed-up values. We notice that more than half of the cases in the testing results fall into the category of super-linear speed-up, which is rare to occur in many parallel speeding domains. The S_p trend of all methods increases dramatically with the increase in the number of utilised CPU cores, and this trend gradually slows down and remains stable at a certain number of CPU cores. The number of CPU cores where a steady state is reached is often method-dependent. From Fig. 18, when the number of CPU cores is three, the S_p values of BCMS become stable. The pruning strategy performs best for DeepMerge, where almost all cases achieve super-linear speed-up, as shown in the last column in Fig. 18. The S_p is also sensitive to the

Fig. 14. The local comparison of segmentation results with and without dynamic-pruning in T4.

study areas. Almost all cases in T3 (the third row in Fig. 18) achieve super-linear speed-up.

The merging criteria partially cause the differences in S_p values. Given that the merging criteria of HSWO are simple, the segmentation algorithm presents high efficiency (Figs. 15 and 16), resulting in the low potential for obtaining high S_p values with dynamic-pruning. Based on the spectral angles, HSeg and Local-SA show a noticeable efficiency improvement. DeepMerge owns the most complex merging criteria, involving spatial Euclidean distance between two one-dimensional vectors with 100 values, causing the lowest efficiency without pruning. We notice that the highest S_p values are obtained by DeepMerge, which uses the most complex merging criteria compared to all other selected algorithms. The segmentation results of DeepMerge reach the highest F values, demonstrating the ability to correctly segment both small and large objects while requiring many steps of weight updates in the merging process. DeepMerge has, therefore, low efficiency without pruning. With pruning, many redundant weight updates are avoided, resulting in higher efficiency gains compared to other methods. In addition, the number of iterations L is also merging criteria-dependent. We found that L=3 is sufficient for the segmentation methods in this study, whereas one iteration is enough for Local-SA. The starting scale at each iteration also affects the segmentation efficiency. The beginning scale at the last iteration is a user-selected scale parameter. We recommend 30% of the scale parameter used in the first iteration and 40% in the second iteration.

The pruning results on the large-scale test area are shown in Table 5. The S_p values on single- and multicore modes are higher than in the four test images, demonstrating that the larger the image, the greater the efficiency gain of pruning. The highest S_p values in single- and multicore modes are 36.07 and 102.74, respectively, achieved by HSWO. The S_p values also indicate a large improvement in the segmentation efficiency of other methods. The pruning framework achieves high efficiency and high robustness with all six methods. Larger images result in higher efficiency gains because they contain large objects, such as bare soils and factory buildings in Table 5 , which require many weight updates to be merged, resulting in low segmentation efficiency.

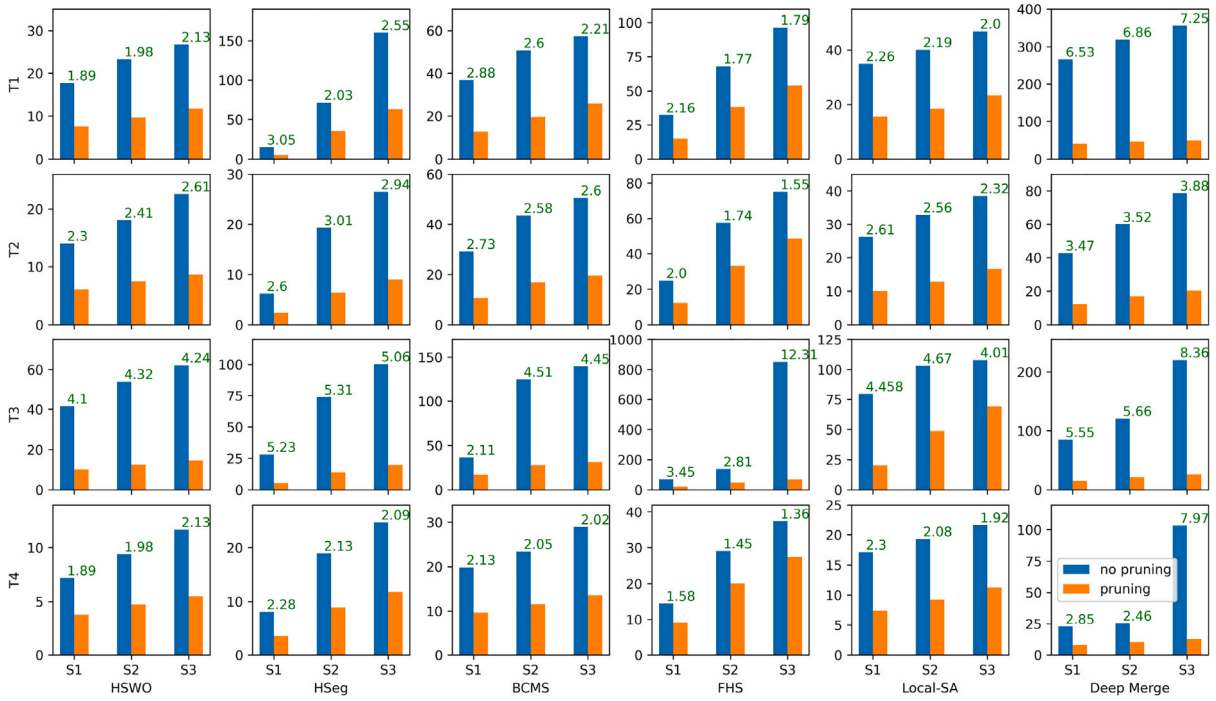

Fig. 15. Segmentation time of tested methods on three scales in the tested images. The X- and Y- axis are scales and processing time, respectively. Four rows are related to the four tested images (T1, T2, T3, and T4). Blue bars represent the segmentation time with no pruning, and orange bar with pruning. Green numbers are the S_p related to each scale. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

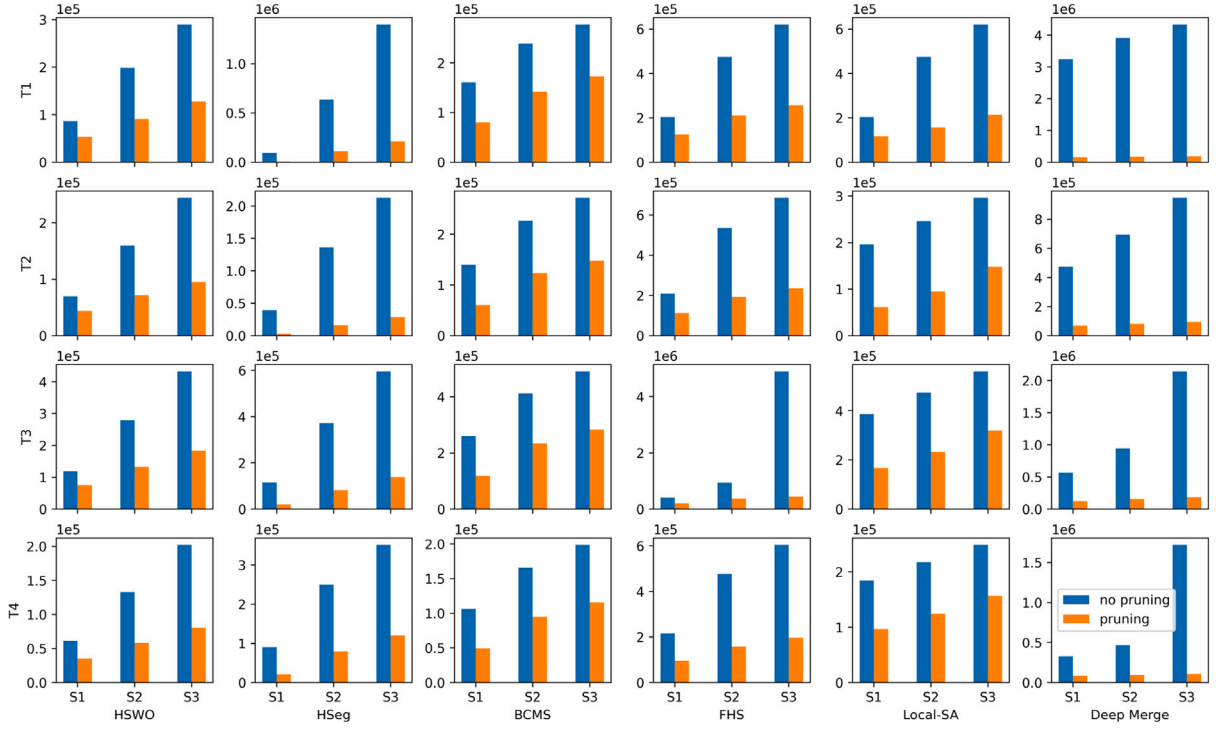

Fig. 16. Steps of weight updates of the tested methods on three scales in the tested images. The X- and Y- axis are scales and number of weight updates. The layout of methods and scales is the same as in Fig. 15.

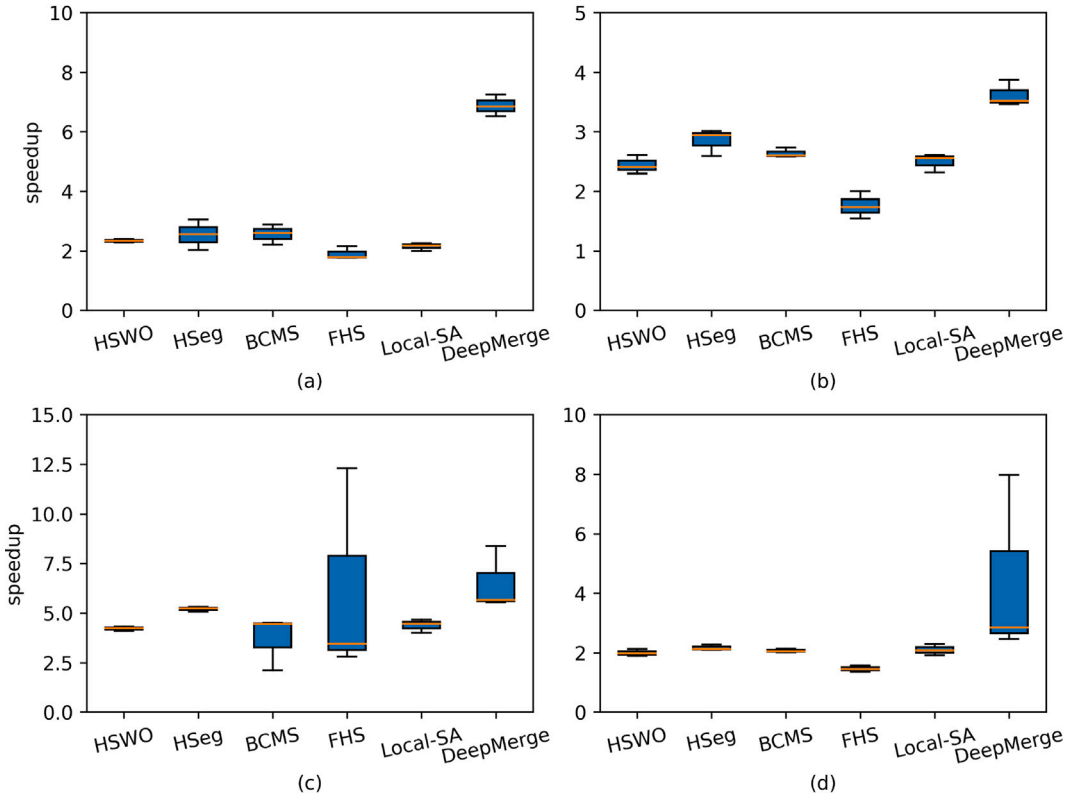

Fig. 17. The impact of scales on the segmentation efficiency in single-core mode. The boxplot of speed-ups in (a) T1; (b) in T2; (c) T3; and (d) T4. The boxplots are based on the three scale results. The height of the boxplot can reflect the degree of change in the S_p of the three scales. The yellow line in the boxplots is the average S_p value of three scales. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 5
The pruning results on the large-scale test data.

Method	No pruning (s)	1-core+pruning (s)	8-cores+pruning (s)	Sp in 1-core	Sp in 8-cores	Cost (steps)	Cost+pruing (steps)
HSWO	1,854.48	51.42	18.05	36.07	102.74	3,257,005	947,950
HSeg	2,553.07	122.79	29.43	20.79	86.75	4,225,712	1,091,663
BCMS	2,925.81	308.54	96.59	9.48	30.29	7,041,757	2,231,039
FHS	2,991.43	152.85	62.17	19.57	48.12	2,181,987	1,282,490
Local-SA	2,047.88	91.14	37.96	22.47	53.95	2,656,107	1,521,081
DeepMerge	3,035.4	105.80	33.30	28.69	91.15	9,010,402	1,046,585

The pruning framework solves the problem caused by these objects and dramatically decreases the number of weight updates. Therefore, large images often mean higher efficiency improvement than small images.

5. Conclusion

In this study, we proposed a dynamic-pruning framework to improve segmentation efficiency on remote sensing images. Specifically, a general acceleration framework was designed for the RAG model to improve segmentation efficiency based on intra-object pruning and inter-object pruning, where the handcrafted framework prunes the edges in the RAG-NNG to reduce redundant weight updates. Utilising the user-defined scale parameters, the framework prunes the scale-related edges in the RAG model to generate many local RAGs and further prunes BFS partitioned edges in the local RAGs. We employed the proposed dynamic-pruning framework on six region-merging-based segmentation methods and tested them on four remote sensing images and one large-scale image. The results of four images and one large-scale image demonstrate that the larger the image, the greater the efficiency gain of pruning. In addition, the data structure of local-RAGs can release the accelerating potential of multicore computers. The proposed dynamic-pruning framework has the potential to revamp image segmentation by significantly improving the efficiency of region-merging-based methods. In future work, we will design a pixel-based bottom-up segmentation method using the pruning framework and deeply optimise the pruning framework to achieve further improvement in efficiency.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by the Foundation of Anhui Province Key Laboratory of Physical Geographic Environment, P.R. China (Grant No. 2022PGE012) and China Scholarship Council.

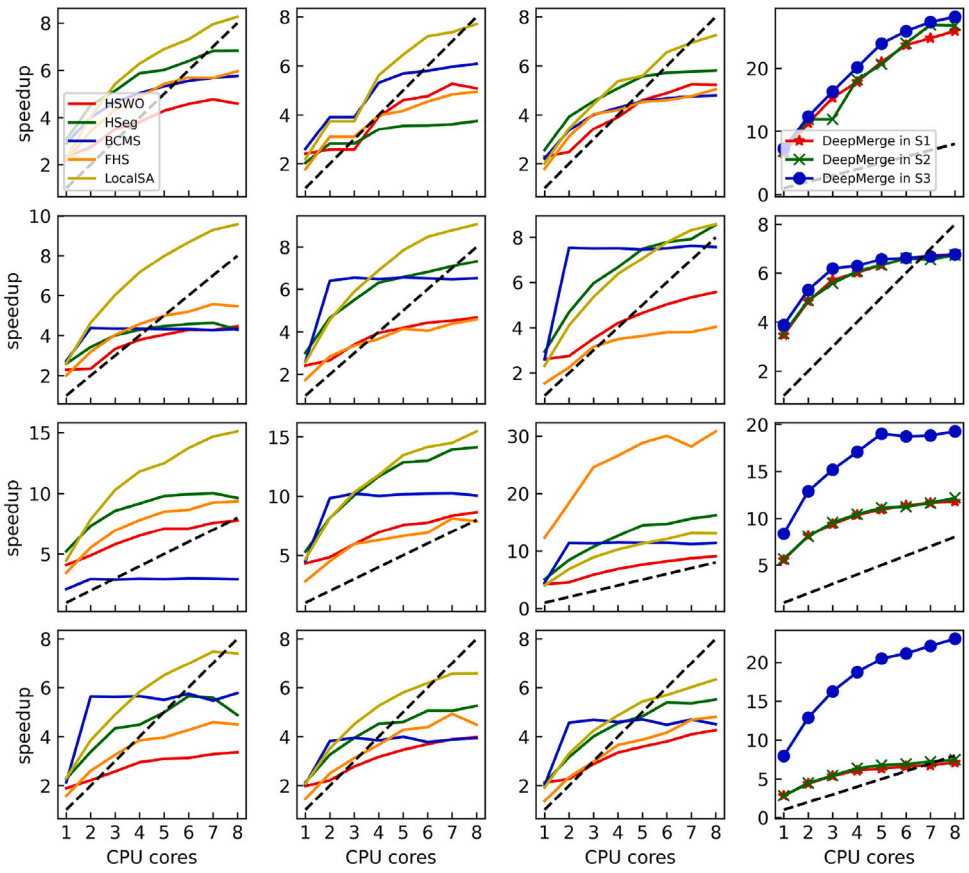

Fig. 18. The S_p values of six methods under Dynamic-pruning in the study areas. The first, second, and third columns in the figure denote the S_p results on scales S1, S2, and S3, respectively. The fourth column shows the results of DeepMerge. The block dotted line denotes the 1:1 reference line, i.e., a proportional function with a slope of one.

References

Baatz, M., 2000. Multi resolution segmentation: an optimum approach for high quality multi scale image segmentation. In: Beutrage Zum AGIT-Symposium. Salzburg, Heidelberg, 2000. pp. 12–23.

BalkanskiF, E., Rubinstein, A., Singer, Y., 2019. An exponential speedup in parallel running time for submodular maximization without loss in approximation. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM, pp. 283–302.

Beaulieu, J.-M., Goldberg, M., 1989. Hierarchy in picture segmentation: A stepwise optimization approach. IEEE Trans. Pattern Anal. Mach. Intell. 11 (2), 150–163.

Bhatnagar, S., Gill, L., Regan, S., Waldren, S., Ghosh, B., 2021. A nested drone-satellite approach to monitoring the ecological conditions of wetlands. ISPRS J. Photogramm. Remote Sens. 174, 151–165.

Blaschke, T., 2010. Object based image analysis for remote sensing. ISPRS J. Photogramm. Remote Sens. 65 (1), 2–16.

Castilla, G., Hay, G.G., Ruiz-Gallardo, J.R., 2008. Size-Constrained Region Merging (SCRM). Photogramm. Eng. Remote Sens. 74 (4), 409–419.

Chen, B., Qiu, F., Wu, B., Du, H., 2015. Image segmentation based on constrained spectral variance difference and edge penalty. Remote Sens. 7 (5), 5980–6004.

Chen, Y., Zhang, G., Cui, H., Li, X., Hou, S., Ma, J., Li, Z., Li, H., Wang, H., 2023. A novel weakly supervised semantic segmentation framework to improve the resolution of land cover product. ISPRS J. Photogramm. Remote Sens. 196, 73–92. http://dx.doi.org/10.1016/j.isprsjprs.2022.12.027, URL: https://www.sciencedirect.com/science/article/pii/S0924271622003422.

Chopra, S., Hadsell, R., LeCun, Y., 2005. Learning a similarity metric discriminatively, with application to face verification. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 1. CVPR'05, IEEE, pp. 539–546.

Drăguţ, L., Csillik, O., Eisank, C., Tiede, D., 2014. Automated parameterisation for multi-scale image segmentation on multiple layers. ISPRS J. Photogramm. Remote Sens. 88, 119–127.

Gelfand, S.B., Ravishankar, C., Delp, E.J., 1989. An iterative growing and pruning algorithm for classification tree design. In: Conference Proceedings., IEEE International Conference on Systems, Man and Cybernetics. IEEE, pp. 818–823.

Harabor, D., Grastien, A., 2011. Online graph pruning for pathfinding on grid maps. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 25, no. 1. pp. 1114–1119.

Haris, K., Efstratiadis, S.N., Maglaveras, N., Katsaggelos, A.K., 1998. Hybrid image segmentation using watersheds and fast region merging. IEEE Trans. Image Process. 7 (12), 1684–1699.

Hossain, M.D., Chen, D., 2019. Segmentation for Object-Based Image Analysis (OBIA): A review of algorithms and challenges from remote sensing perspective. ISPRS J. Photogramm. Remote Sens. 150, 115–134.

Hu, Z., Li, Q., Zhang, Q., Zou, Q., Wu, Z., 2017. Unsupervised simplification of image hierarchies via evolution analysis in scale-sets framework. IEEE Trans. Image Process. 26 (5), 2394–2407.

Hu, Z., Zhang, Q., Zou, Q., Li, Q., Wu, G., 2018. Stepwise evolution analysis of the region-merging segmentation for scale parameterization. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 11 (7), 2461–2472.

Lin, M., Zhang, Y., Li, Y., Chen, B., Chao, F., Wang, M., Li, S., Tian, Y., Ji, R., 2022. 1Xn pattern for pruning convolutional neural networks. IEEE Trans. Pattern Anal. Mach. Intell..

Liu, T., Abd-Elrahman, A., 2018. Multi-view object-based classification of wetland land covers using unmanned aircraft system images. Remote Sens. Environ. 216, 122–138.

Liu, Y., Bian, L., Meng, Y., Wang, H., Zhang, S., Yang, Y., Shao, X., Wang, B., 2012. Discrepancy measures for selecting optimal combination of parameter values in object-based image analysis. ISPRS J. Photogramm. Remote Sens. 68, 144–156.

Liu, J., Zhuang, B., Zhuang, Z., Guo, Y., Huang, J., Zhu, J., Tan, M., 2021. Discrimination-aware network pruning for deep model compression. IEEE Trans. Pattern Anal. Mach. Intell. 44 (8), 4035–4051.

Lv, X., Ming, D., Chen, Y., Wang, M., 2019. Very high resolution remote sensing image classification with SEEDS-CNN and scale effect analysis for superpixel CNN classification. Int. J. Remote Sens. 40 (2), 506–531.

Lv, X., Ming, D., Lu, T., Zhou, K., Wang, M., Bao, H., 2018. A new method for region-based majority voting CNNs for very high resolution image classification. Remote Sens. 10 (12), 1946.

Lv, X., Persello, C., Huang, X., Ming, D., Alfred, S., 2023. DeepMerge: Deep learning-based region-merging for image segmentation. ISPRS J. Photogramm. Remote Sens. (submitted for publication) URL: https://paperswithcode.com/paper/deepmerge-deep-learning-based-region-merging.

Lv, X., Shao, Z., Ming, D., Diao, C., Zhou, K., Tong, C., 2021. Improved object-based convolutional neural network (IOCNN) to classify very high-resolution remote sensing images. Int. J. Remote Sens. 42 (21), 8318–8344.

Martins, V.S., Kaleita, A.L., Gelder, B.K., da Silveira, H.L., Abe, C.A., 2020. Exploring multiscale Object-Based Convolutional Neural Network (multi-OCNN) for remote sensing image classification at high spatial resolution. ISPRS J. Photogramm. Remote Sens. 168, 56–73. http://dx.doi.org/10.1016/j.isprsjprs.2020.08.004, URL: https://www.sciencedirect.com/science/article/pii/S0924271620302124.

Ming, D., Li, J., Wang, J., Zhang, M., 2015. Scale parameter selection by spatial statistics for GeOBIA: Using mean-shift based multi-scale segmentation as an example. ISPRS J. Photogramm. Remote Sens. 106, 28–41.

Orynbaikyzy, A., Gessner, U., Conrad, C., 2019. Crop type classification using a combination of optical and radar remote sensing data: A review. Int. J. Remote Sens. 40 (17), 6553–6595.

Paris, S., Durand, F., 2007. A topological approach to hierarchical segmentation using mean shift. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, pp. 1–8.

Salembier, P., Foucher, S., 2016. Optimum graph cuts for pruning binary partition trees of polarimetric SAR images. IEEE Trans. Geosci. Remote Sens. 54 (9), 5493–5502.

Shafarenko, L., Petrou, M., Kittler, J., 1997. Automatic watershed segmentation of randomly textured color images. IEEE Trans. Image Process. 6 (11), 1530–1544.

Tilton, J.C., Tarabalka, Y., Montesano, P.M., Gofman, E., 2012. Best merge region-growing segmentation with integrated nonadjacent region object aggregation. IEEE Trans. Geosci. Remote Sens. 50 (11), 4454–4467.

Tochon, G., Feret, J.-B., Valero, S., Martin, R.E., Knapp, D.E., Salembier, P., Chanussot, J., Asner, G.P., 2015. On the use of binary partition trees for the tree crown segmentation of tropical rainforest hyperspectral images. Remote Sens. Environ. 159, 318–331.

Tong, X.Y., Xia, G.S., Lu, Q., Shen, H., Li, S., You, S., Zhang, L., 2018. Land-cover classification with high-resolution remote sensing images using transferable deep models.

Van den Bergh, M., Boix, X., Roig, G., de Capitani, B., Van Gool, L., 2012. Seeds: Superpixels extracted via energy-driven sampling. In: European Conference on Computer Vision. Springer, pp. 13–26.

Vo, D.T., Sole, J., Yin, P., Gomila, C., Nguyen, T.Q., 2009. Selective data pruning-based compression using high-order edge-directed interpolation. IEEE Trans. Image Process. 19 (2), 399–409.

Wassenberg, J., Middelmann, W., Sanders, P., 2009. An efficient parallel algorithm for graph-based image segmentation. In: International Conference on Computer Analysis of Images and Patterns. Springer, pp. 1003–1010.

Wu, X., 1993. Adaptive split-and-merge segmentation based on piecewise least-square approximation. IEEE Trans. Pattern Anal. Mach. Intell. 15 (8), 808–815.

Yang, J., He, Y., Caspersen, J., 2017. Region merging using local spectral angle thresholds: A more accurate method for hybrid segmentation of remote sensing images. Remote Sens. Environ. 190, 137–148.

Yu, L., Gong, P., 2012. Google earth as a virtual globe tool for earth science applications at the global scale: Progress and perspectives. Int. J. Remote Sens. 33 (12), 3966–3986.

Zhang, X., Feng, X., Xiao, P., He, G., Zhu, L., 2015. Segmentation quality evaluation using region-based precision and recall measures for remote sensing images. ISPRS J. Photogramm. Remote Sens. 102, 73–84.

Zhang, F., Lin, H., Zhai, J., Cheng, J., Xiang, D., Li, J., Chai, Y., Du, X., 2018a. An adaptive breadth-first search algorithm on integrated architectures. J. Supercomput. 74 (11), 6135–6155.

Zhang, C., Sargent, I., Pan, X., Li, H., Gardiner, A., Hare, J., Atkinson, P.M., 2018b. An Object-Based Convolutional Neural Network (OCNN) for urban land use classification. Remote Sens. Environ. 216, 57–70.

Zhang, X., Xiao, P., Feng, X., 2017. Toward combining thematic information with hierarchical multiscale segmentations using tree Markov random field model. ISPRS J. Photogramm. Remote Sens. 131, 134–146.

Zhang, X., Xiao, P., Feng, X., 2020. Object-specific optimization of hierarchical multiscale segmentations for high-spatial resolution remote sensing images. ISPRS J. Photogramm. Remote Sens. 159, 308–321.

Zhang, X., Xiao, P., Feng, X., He, G., 2019. Another look on region merging procedure from seed region shift for high-resolution remote sensing image segmentation. ISPRS J. Photogramm. Remote Sens. 148, 197–207.

Zhang, X., Xiao, P., Feng, X., Wang, J., Wang, Z., 2014. Hybrid region merging method for segmentation of high-resolution remote sensing images. ISPRS J. Photogramm. Remote Sens. 98, 19–28.

Zhang, X., Xiao, P., Song, X., She, J., 2013. Boundary-constrained multi-scale segmentation method for remote sensing images. ISPRS J. Photogramm. Remote Sens. 78, 15–25.

Zhao, W., Persello, C., Stein, A., 2021. Building outline delineation: From aerial images to polygons with an improved end-to-end learning framework. ISPRS J. Photogramm. Remote Sens. 175, 119–131.

Zhao, W., Persello, C., Stein, A., 2022. Extracting planar roof structures from very high resolution images using graph neural networks. ISPRS J. Photogramm. Remote Sens. 187, 34–45.