An Efficient Recursive Shortest Spanning Tree Algorithm Using Linking Properties

Sai Ho Kwok, A. G. Constantinides, *Fellow, IEEE*, and Wan-Chi Siu, *Senior Member, IEEE*

***Abstract*—Speed is a great concern in the recursive shortest spanning tree (RSST) algorithm as its applications are focused on image segmentation and video coding, in which a large amount of data is processed. Several efficient RSST algorithms have been proposed in the literature, but the linking properties are not properly addressed and used in these algorithms and they are intended to produce a truncated RSST. This paper categorizes the linking process into three classes based on link weights. These linking processes are defined as the linking process for link weight equal to zero (LPLW-Z), the linking process for link weight equal to one (LPLW-O), and the linking process for link weight equal to real number (LPLW-R). We study these linking properties and apply them to an efficient RSST algorithm. The proposed efficient RSST algorithm is novel, as it makes use of linking properties, and its resulting shortest spanning tree is truly identical to that produced by the conventional algorithm. Our experimental results show that the percentages of links for the three classes are 17%, 27%, and 58%, respectively. This paper proposes a prediction method for LPLW-O, as a result of which the vertex weight of the next region can be determined by comparing sizes of the merging regions. It is also demonstrated that the proposed LPLW-O with prediction approach is applicable to the multiple-stage merging. Our experimental results show that the proposed algorithm has a substantial improvement over the conventional RSST algorithm.**

***Index Terms*—Edge detection, graph theory, image segmentation, linking property, recursive shortest spanning tree (RSST).**

I. Introduction

MANY recursive shortest spanning tree (RSST) applications have been proposed in the past decades. Its applications include image segmentation and edge detection for two-dimensional (2-D) image data, and temporal segmentation, temporal decimation, and object segmentation and tracking for three-dimensional (3-D) video data. Efficiency is always a major concern in applying RSST in these applications. Constructing the shortest spanning tree in a recursive way differentiates it from the well-known shortest spanning tree (SST) problem, and, therefore, many existing fast and parallel solutions for SST are not applicable.

Manuscript received April 2, 2002; revised March 3, 2003. This paper was recommended by Associate Editor E. Izquierdo.

S. H. Kwok is with the Department of Information and Systems Management, The Hong Kong University of Science and Technology, Kowloon, Hong Kong (e-mail: jkwok@ust.hk).

A. G. Constantinides is with the Department of Electrical and Electronic Engineering, Imperial College of Science, Technology, and Medicine, London SW7 2BT, U.K. (e-mail: a.constantinides@ee.ic.ac.uk, a.constantinides@ic.ac.uk).

W.-C. Siu is with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: enwcsiu@polyu.edu.hk).

Digital Object Identifier 10.1109/TCSVT.2004.828334

Morris *et al.* [30], [31] applied graph theory to image processing applications, including image analysis, image segmentation, and edge detection problems, which resulted in a RSST representation of image data. The RSST was proven to be highly accurate to define regions. Due to its hierarchical representation, any number of region representations of an image can be chosen as desired. Zeng further modified the RSST algorithm so that the hierarchical representation can be formed in consideration of perceptually significant regions [41] and the saliency of region contours [40]. In addition, Kwok [19] proposed a hierarchical structure for 3-D data using RSST.

The RSST was later extended to both segmented image coding [8], [29], [42] and segmented video coding [2], [28]. Morris and Constantinides [27] also proposed a progressive image coding scheme incorporating with it. For video processing applications, Kwok *et al.* extended the RSST to temporal segmentation application [26] and temporal decimation application [24]. Alatan *et al.* [3] applied RSST to motion-vector-based segmentation in interactive multimedia services. Other video applications are object extraction [11] and object segmentation and tracking , [35], [37]. Recently, the RSST algorithm was also applied to 2-D affine motion modeling by Tuncel and Onural [36] and video content representation using its multiresolution implementation by [14] and [15]. Another branch of RSST algorithm is due to the multiresolution RSST (M-RSST) segmentation algorithm. The M-RSST has been applied for a content-based face detection [5], an active contour-based video object segmentation scheme for stereoscopic video sequences [32], an efficient unsupervised content-based segmentation in stereoscopic video sequences [13], and video database [6]. Prior to Vlachos and Constantinides [38] proposal of RSST for color images, in which color components of red, green, and blue are translated to the cost function, RSST algorithms and its applications were based on grayscale images.

Studies on efficient implementation of RSST were reported by Kwok and Constantinides [20]–[22], [25]. In [21] and [23], they proposed a tailored data partition strategy to assign jobs to processing elements in their parallel algorithm, and proved their parallel implementation is cost-optimal. In the fast RSST implementation, they expedite the RSST algorithm from the complexity of $O(n^2)$ to $O(n)$ in the worst case, which is the new lower bound for algorithms of this type, where n is the number of vertices in the graph [20], [25]. The fast implementation is achieved by removing the sorting algorithm from the conventional algorithm. However, the outputs of these efficient RSST algorithms [20], [21], [23], [25] are truncated versions of RSST, which is different from the RSST generated by [30] and [31]. However, these algorithms did not address a potential approach of using the linking property of RSST. The original RSST algo-

rithm [30], [31] consists of two functional blocks, namely, initialization stage and linking process. The linking process is computationally expensive. By understanding the linking properties of RSST, we can streamline the linking process and construct the RSST in a more efficient way. In this paper, we investigate the linking properties and propose an efficient RSST algorithm using the linking properties.

A close relative of RSST is SST or minimum spanning tree (MST). Numerous efficient SST algorithms [7], [9], [12], [16]–[18], [33], [34], [39] have been reported. Most of these algorithms are based on a property of the minimum spanning trees which is generally known as the minimum spanning tree property [1]. Unfortunately, these approaches cannot be adapted to RSST as RSST is constructed in a recursive manner and there is no report addressing an efficient RSST algorithm using the RSST properties. This motivates the present work.

The rest of the paper is organized as follows. Section II provides background information about graph theory, which defines necessary terminologies used in RSST, and delineates the conventional RSST algorithm for 2-D image data. In Section III, we study and analyze the linking properties for various linking processes. Proposals of efficient realization for these linking processes are also given. Section IV outlines the efficient RSST algorithm using the linking properties. Section V evaluates the performance of the proposed RSST algorithm. Finally, Section VI summarizes the contributions of our study.

II. RSST

A. Definition

Graph theory is the study of graphs and their applications. A graph $G = (V, E)$ is made up of a set of *vertices* V_i and V_j connected to each other by *links* $E_{i,j}$, for $i \neq j$, and where V_i and V_j are the terminal vertices that the link connects. In a weighted graph the vertices and links have weights associated with them, namely, v_i and $e_{i,j}$, respectively (v_i and $e_{i,j}$ are also known as vertex weight and link weight, respectively). Each vertex is not necessarily linked to every other, but if the vertices are linked together then the graph is complete. A *partial graph* has the same number of vertices but only a subset of the links of the original graph.

A *chain* is a list of successive vertices in which each vertex is connected to the next by a link in the graph. A *cycle* is a chain whose end links meet at the same vertex. A *tree* is a connected set of chains such that there are no cycles. A *spanning tree* is a tree, which is also a partial graph. The *shortest spanning tree* of a weighted graph is a spanning tree such that the sum of its link weights, or some other monotonic function of its link weights, is a minimum for any possible spanning tree. It is *trivial* if it consists of a single vertex, and it is *nontrivial*, otherwise. A *forest* is a set of trees, and a *spanning forest* is a forest that is also a partial graph. A full explanation of these terms can also be found in [10].

B. Conventional RSST Algorithm

The conventional RSST algorithm [30], [31] consists of two functional blocks, namely, the initialization stage and the linking process. The flowchart of the RSST algorithm is depicted in Fig. 1. The RSST starts with a mapping of an image onto a weighted graph at the initialization stage of IS-1. Each region or vertex initially contains only one pixel. The pixel intensity values of regions are used to evaluate vertex weights and link weights of the graph. A vertex weight is defined as the average intensity value of the corresponding region, while a link weight is evaluated by a link weight function or a cost function, which is basically a function of the vertex weights and the sizes of the connected regions. All links are then sorted in order according to their link weights, and stored in a heap at IS-2. In entering to the linking process, a link with the least link weight in the graph is chosen to be the next link of the SST. The chosen link is saved and the two connecting or merging regions are merged in LP-1. The vertex weight of the newly merged region is updated, hence, all surrounding links need to be recalculated in LP-2 and all loop-forming links, also known as duplicated links, will be removed in LP-3. Subsequently, all remaining links are sorted in LP-4. Thus, the number of regions is progressively reduced from $x \times y$ in an x pixel by y pixel image, down to just one if desired. Those saved links form a spanning tree representation of the image. By noting the order in which the links are saved, a hierarchical representation of the original image is created.

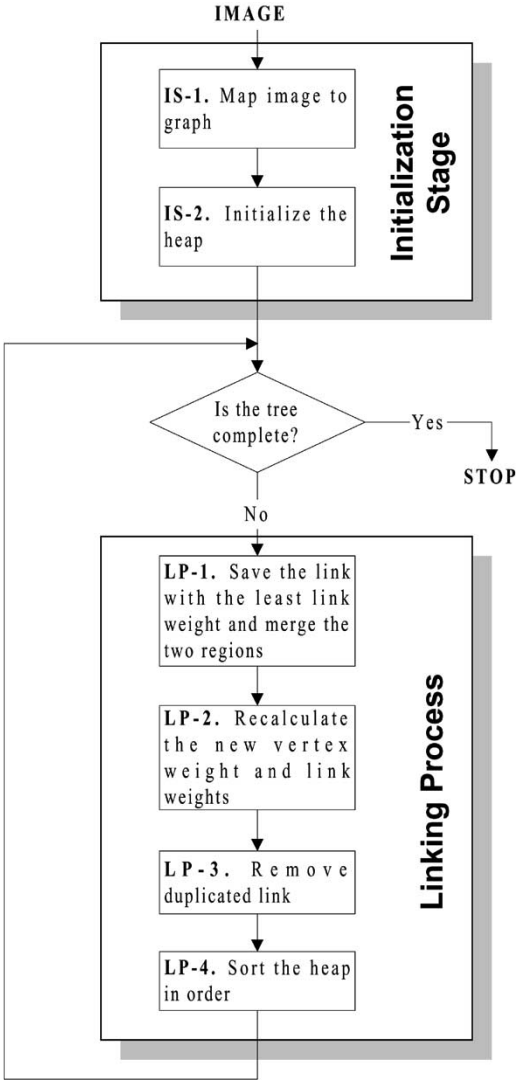


Fig. 1. Flowchart of the conventional RSST algorithm.

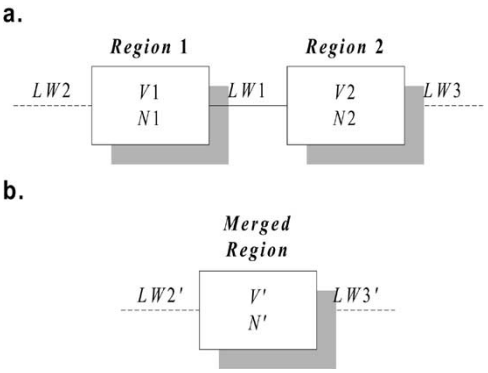


Fig. 2. Merging two regions. (a) Before merging. (b) After merging.

III. Linking Property

In constructing an RSST, the linking process is executed $n-1$ times recursively, where n is the total number of vertices in the graph. The linking process is composed of processes of saving the link with the least link weight, recalculating the new vertex and link weights, removing the duplicated link, and sorting the link. The decision for linking two vertices is solely based on link weight, which is equal to the distance between two connected vertices evaluated by a cost function.

An important procedure in the linking process is to merge two regions and it is regarded as a merging process, as shown in Fig. 2. Fig. 2(a) shows two regions, *Region 1* and *Region 2*, with three links. The link with link weight LW1 connects the two regions, while the other two links connect to two other adjacent regions. Each region or vertex in the graph is represented by a vertex weight V and the size of the region N. As a region may consist of more than one vertex, $N1$ and $N2$ are used to indicate the numbers of already merged vertices in the *Region 1* and *Region 2*, respectively. Link weights LW1, LW2, and LW3 are evaluated by the cost function $\mathrm{LW}(v)$. The absolute difference function (ADF) was the cost function in the conventional RSST algorithm [30], [31] and is also used in this paper.

The ADF cost function is defined as follows:

$$\mathrm{LW}(v) = |Va - v| \tag{1}$$

where Va is the vertex weight of an adjacent region (for example, $\mathrm{LW1} = \mathrm{LW}(V1) = |V2 - V1|$). Assume LW1 is the smallest link weight in the graph. Then the two regions at both ends of the link are merged and become one as shown in Fig. 2(b). Link weights of the two connecting links, LW2 and LW3, are altered due to the change of the vertex weight of the merged region, V'. After the merging process, they become LW2$'$ and LW3$'$. The newly merged region is represented by V' and N'. V' is the new vertex weight of the region determined by (2), while N' is the sum of $N1$ and $N2$ as expressed in (3). The adjustment of link weight leads to the rearrangement of links in the heap.

$$V' = \frac{V1 \times N1 + V2 \times N2}{N1 + N} \tag{2}$$

$$N' = N1 + N2. \tag{3}$$

Comparing the current least link weight with the adjusted link weights, e.g., LW2$'$ and LW3$'$, the next link to be joined to the resulting RSST can be identified instantly, even without going through the expensive sorting algorithm. Therefore, by studying the linking property of how the merging process affects link weights of adjacent links, we could predict and identify the next link after merging.

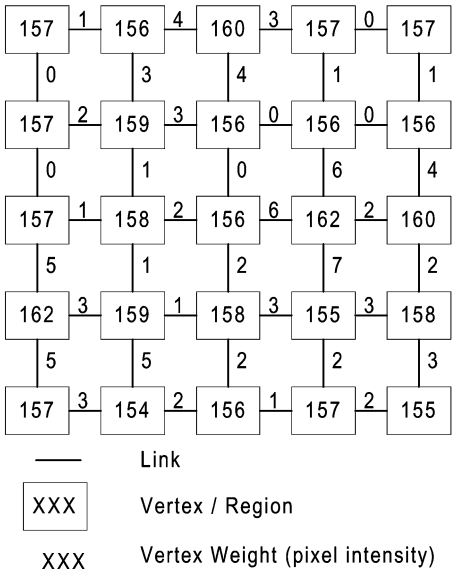


Fig. 3. Initial graph.

Due to the importance of the linking process, we study and analyze the properties of the linking process in this paper. Our analysis starts with a classification of linking process based on link weight value, which can be basically divided into three groups: zero, one, and others (i.e., real numbers). With the use of the classification, we introduce three terms to represent these linking processes. They are: 1) linking process for link weight equal to zero (LPLW-Z); 2) linking process for link weight equal to one (LPLW-O); and 3) linking process for link weight equal to real number (LPLW-R). Analyses of these linking processes are given in the following sections. Our focus is to realize an efficient RSST algorithm using these linking processes and their linking properties. Possible solutions include prediction techniques for identifying next links in these three linking processes.

A. LPLW-Z

If two adjacent regions are of the same vertex weight, the link connecting them is zero link weight. Thus, the linking process to handle this type of link is classified as LPLW-Z. This is usual and common for image data that contains a number of pixels with identical vertex weight clustering together. An example of 5×5 vertices extracted from the testing image *House* is given in Fig. 3.

A link with zero link weight is of the highest priority to be used as zero is the smallest positive integer number. Since the graph contains many links with zero link weights, these links are saved to the representation of the resulting RSST if and only if they are not duplicated links. Although RSST [30], [31] relies on the linking order for the hierarchical representation of the graph, the linking order in LPLW-Z has no impact on the resulting RSST. It is because the conventional RSST algorithm has no explicit rule to distinguish the importance of links with identical link weights. This is a very important property for the

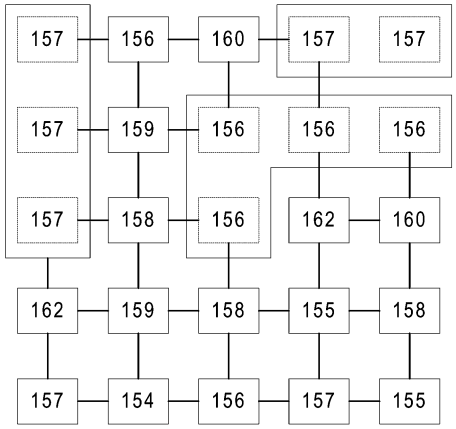


Fig. 4. After LPLW-Z.

development of efficient RSST algorithm. This linking property is also extended to LPLW-O.

The LPLW-Z is the simplest among the three linking processes due to the following reasons.

- Associated vertex weights and link weights remain unchanged after the linking process.
- Recalculation of associated vertex weights and link weights are not required, but the size of the merged region is required.
- The sorting algorithm is also not required.

Therefore, this is just a straightforward merging process with duplicated links removal. It is desirable and recommended to include this linking process in the initialization stage.

After applying the LPLW-Z to the example, the resulting shortest spanning forest is depicted in Fig. 4. The LPLW-Z usually finishes with a shortest spanning forest, except the entire RSST is completed at the end of the linking process. Our empirical results show that 15.4% of the entire RSST is completed after the LPLW-Z.

B. LPLW-O

The LPLW-O is important to the RSST since a significant amount of linking of this type is found in typical image- and video-processing applications. In our experiment, 26.8% of links in the resulting RSST were contributed by the LPLW-O. However, this linking process is not as simple as the LPLW-Z. It requires an update on corresponding vertex and link weights, duplicated link removal, and sorting link. We analyze the merging process in LPLW-O, and apply the characteristic of the merging process to develop a linking process for link weight equal to one with prediction, also known as the LPLW-O with prediction. The proposed technique could expedite the linking process by: 1) providing guidelines for choosing the next region and the next link without recalculating vertex weight and link weight; (2) limiting the number of vertices to be affected in the linking process; (3) minimizing the cost of sorting; and (4) eliminating the process of duplicated link removal.

A crucial parameter in the merging process, V', the vertex weight of the merged region, determines the link weights of adjacent links after merging. By knowing the impact of V' to the link weight, an accurate prediction about the next link can be made. Our analysis starts with the formula of V'. Referring to Fig. 2, assume $\mathrm{LW1} = 1$ and, therefore, it is selected by the LPLW-O. Since V' is the average value of the vertex weights $V1$ and $V2$ as stated in (2), V' value has to be between $V1$ and $V2$.

Hence

$$V1 < V' < V2, \quad \text{if } V1 < V2 \quad \text{or}$$
$$V1 < V' < V2, \quad \text{if } V1 < V2.$$

V' can be represented in terms of region parameters $V1, V2, N1$, and $N2$ used in (2) and (3).

Therefore

$$V' = V1 + \frac{N2}{N1+N2} \times (V2 - V1) \tag{4}$$

or

$$V' = V2 + \frac{N1}{N1+N2} \times (V1 - V2). \tag{5}$$

Generalizing (4) and (5), we produce

$$V' = \min(V1, V2) + \frac{f(\min(V1, V2))}{N1+N2} \times |V1 - V2|, \tag{6}$$

$$\text{where } f(V) = \begin{cases} N1 \text{ for } V = V2 \\ N2 \text{ for } V = V1. \end{cases} \tag{7}$$

Simplifying (6) by applying the fact that $|V1 - V2| = 1$ for LPLW-O

$$V' = \min(V1, V2) + \frac{f(\min(V1, V2))}{N1+N2}. \tag{8}$$

Therefore, (8) represents a generalized equation for V' in terms of the parameters of *Region 1* and *Region 2*. We then explore how these parameters are affected by V'.

Let

$$\mathrm{DF}\ (\text{Driven Factor}) = \frac{f(\min(V1, V2))}{N1+N2}, \quad \text{where } 0 < \mathrm{DF} < 1. \tag{9}$$

Applying (7), DF can also be expressed as follows: $\mathrm{DF} = (N1)/(N1+N2)$ or $\mathrm{DF} = (N2)/(N1+N2)$ and

$$V' = \min(V1, V2) + \mathrm{DF}. \tag{10}$$

Let us partition the range of the DF value into three parts in our analysis, $0 < \mathrm{DF} < 0.5, \mathrm{DF} = 0.5$, and $0.5 < \mathrm{DF} < 1$. In extending Fig. 2, Fig. 5 defines terms for the prediction technique in LPLW-O. Let *Rmerged* be the merged region, *Rnext* be a region connecting to *Rmerged*, and *LWnext* be the link weight between *Rnext* and *Rmerged*. The vertex weight and the size of *Rnext* are represented as v and n, respectively. The *LWnext* is evaluated by the cost function of ADF, defined in (1).

It is noted that after merging *Region 1* and *Region 2*, the vertex weight of *Rnext*, v, stays unchanged but the corresponding *LWnext* is updated due to the change of the connected region, *Region 1* or *Region 2*. For instance, the vertex weight is changed from $V1$ to V' in *Region 1*.

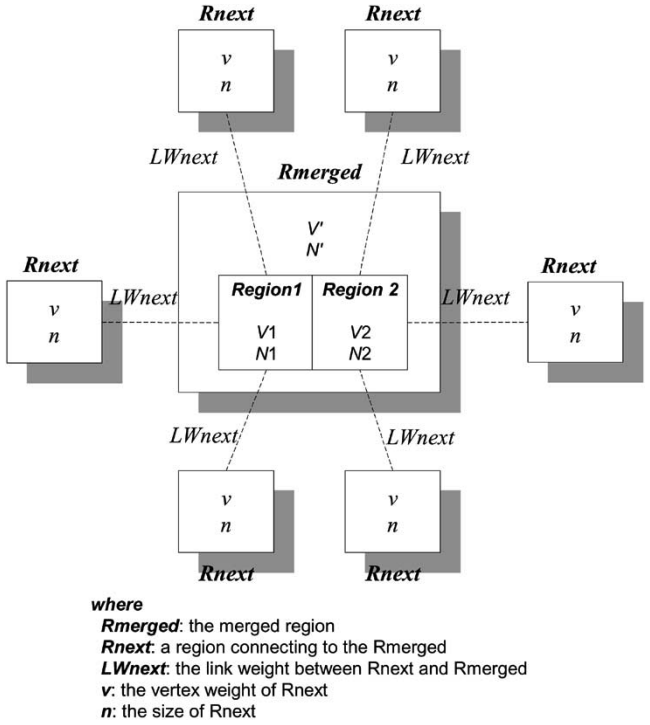


Fig. 5. Term definitions for the LPLW-O with prediction.

TABLE I
THE CHANGE OF *LWnext*. (NOTE: SINCE THE PROCESS OF DUPLICATED LINK REMOVAL IS NO LONGER REQUIRED IN THE LPLW-O WITH PREDICTION, THE CASE OF HAVING DUPLICATED LINK CAN BE IGNORED)

	Link Weight (*LWnext*)	
	Between *Region 1* and *Rnext*	Between *Region 2* and *Rnext*
Before Merging	$LW(v)=\lvert V1-v\rvert$	$LW(v)=\lvert V2-v\rvert$
After Merging	$LW(v)=\lvert(min(V1,V2)+DF)-v\rvert$	

TABLE II
RELATIONSHIPS OF DF, V', AND *LWnext*

	$0<DF<0.5$	$DF=0.5$	$0.5<DF<1$
V'	tends to $min(V1,V2)$	$min(V1,V2)+0.5$	tends to $max(V1,V2)$
	Link Weight		
$LW(v=V1)$	**less than 1**	**less than 1**	**less than 1**
$LW(v=V2)$	**less than 1**	**less than 1**	**less than 1**
$LW(v\neq\{V1,V2\})$	more than 1	more than 1	more than 1

Table I shows that the change of *LWnext* is dependent upon the vertex weight of *Rmerged*. To estimate the change of *LWnext*, we analyze the relationships among DF, V', and *LWnext*.

Table II tabulates the resulting vertex weights of V' evaluated using (10) for various ranges of DF. We apply results of V' to three possible cases of vertex weight of *Rnext*, v, namely, $v=V1, v=V2$, and $v\neq\{V1,V2\}$. The results in Table II show that when two regions are merged in LPLW-O, *LWnext* will be decreased and less than 1, if and only if the vertex weight of *Rnext*, v, is equal to either $V1$ or $V2$. Thus, these regions are possible next regions. Vertex weights other than $V1$ and $V2$ must produce *LWnext* greater than one. As a result, LPLW-O cannot process these links since this is beyond the scope of the LPLW-O.

When the merging process is completed, the next link will be the one with the least link weight. If there are links with link weight less than one generated during the last linking process, the one among them with the least link weight will be the next link. Otherwise, any links with link weight equal to one is used. When all links with link weight equal to one stored in the heap are consumed, the LPLW-O is completed.

Thus, a technique that can estimate adjacent link weights and identify the one with the least link weight is a prediction method for LPLW-O. Results in Table II state that the potential next link has to be of the vertex weight either $V1$ or $V2$. Therefore, the prediction technique is first to verify the vertex weight of *Rnext* and then estimate V' in order to predict the next link.

Table II gives a rough estimation of V'. When DF is in between 0 and 0.5, V' tends to be close to the smaller value of $V1$ and $V2$. Thus, the next region would be the region with vertex weight of $\min(V1,V2)$. For the case where DF is greater than 0.5 and less than 1, the next region will be the one with the vertex weight of $\max(V1,V2)$. However, there is no difference from choosing a *Rnext* with vertex weight $V1$ or $V2$, when DF is equal to 0.5.

Since the DF value plays an important role in determining the vertex weight of the next region, we state the necessary conditions for various ranges of DF in terms of V and N. This can be regarded as a prediction method for the next region. This prediction method is not computationally intensive, as many parameters are precomputed and, therefore, it solely requires comparison operation.

For the case of $0<\mathrm{DF}<0.5$, using (9)

$$\mathrm{DF}=\frac{f(\min(V1,V2))}{N1+N2}<0.5.$$

We obtain

$$f(\min(V1,V2))<0.5N1+0.5N2. \tag{11}$$

Referring to (7) and assuming $V1<V2$

$$f(\min(V1,V2))=f(V1)=N2. \tag{12}$$

If (12) holds, the following also holds:

$$f(\max(V1,V2))=f(V2)=N1. \tag{13}$$

Another pair of equations is obtained for the case of $V1>V2$, as follows:

$$f(\min(V1,V2))=f(V2)=N1 \tag{14}$$

$$f(\max(V1,V2))=f(V1)=N2. \tag{15}$$

For the case of $V1<V2$, substitute (12) and (13) to (11). Therefore, (11) becomes

$$f(\min(V1,V2))<f(\max(V1,V2)) \tag{16}$$

$$f(V1)<f(V2) \tag{17}$$

$$N2<N1. \tag{18}$$

It is noted that (16) also holds when $V1>V2$. Equation (16) is a general inequality for the case of $0<\mathrm{DF}<0.5$ regardless of the values of $V1$ and $V2$. Equation (16) states the condition for $0<\mathrm{DF}<0.5$, where $f(\min(V1,V2))$ must be smaller than $f(\max(V1,V2))$. Referring to (10) $V'=\min(V1,V2)+\mathrm{DF}$, the vertex weight of the next region has to be $\min(V1,V2)$ when DF is less than 0.5. Applying the above condition, the sizes of the regions, $N1$ and $N2$, which are expressed in (18), can be used as a prediction criterion in practice to select the next

TABLE III
LOOKUP TABLE FOR *Rnext* IN LPLW-O

Condition of *Rmerged*			Vertex Weight of *Rnext*		
General case	$V1<V2$	$V1>V2$	General case	$V1<V2$	$V1>V2$
For $0<DF<0.5$, $f(min(V1,V2))<f(max(V1,V2))$	$N2<N1$	$N1<N2$	$min(V1,V2)$	$V1$	$V2$
For $DF=0.5$, $f(min(V1,V2))=f(max(V1,V2))$	$N2=N1$	$N1=N2$	$min(V1,V2)$ or $max(V1,V2)$	$V1$ or $V2$	$V1$ or $V2$
For $0.5<DF<1$, $f(min(V1,V2))>f(max(V1,V2))$	$N2>N1$	$N1<N2$	$max(V1,V2)$	$V2$	$V1$

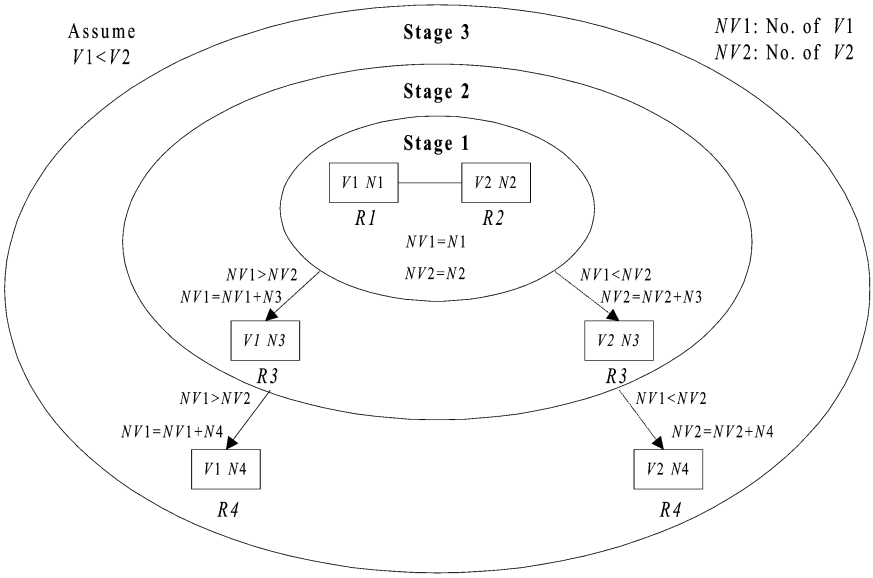


Fig. 6. Multiple-stage merging in LPLW-O with prediction.

region. Therefore, the link connecting the chosen next region and the merging region is the next link.

Following the above analysis for $0 < \mathrm{DF} < 0.5$, we analyze the other two cases of DF. Results are presented in Table III. It is used as a reference lookup table for the next region. In Table III, we summarize general cases for various DF values in terms of parameters of the merging regions. For instance, in the case of $V1 > V2$, if $N1$ is greater than $N2$, we know that the DF must be in the range of 0.5–1 as shown in Table III. V' tends to $\max(V1, V2)$, which is $V1$. We must go for the *Rnext* with the vertex weight of $V1$ because its *LWnext* is less than one and will be the smallest after the merging process. If there is more than one possible next region available, randomly pick one as they are of equal priority to be the next region.

With the prediction method given in Table III, the next region and the next link can be identified using precomputed parameters. However, a multiple-stage merging that merges multiple regions in a linking process and results in a resulting tree is usual in applications. Therefore, the merging process in LPLW-O with prediction is required to perform recursively in this situation. The following is an example to demonstrate how to apply the LPLW-O with prediction to the multiple-stage merging. Fig. 6 shows a graphical representation of the multiple-stage merging using LPLW-O with prediction. At stage 1, regions $R1$ and $R2$ are merged, as the link connecting them is of the link weight equal to one. Assume $V1$ is smaller than $V2$. Let $NV1$ and $NV2$ be the numbers of vertices being merged with vertex weights equal to $V1$ and $V2$, respectively. It is noted that $NV1 = N1$ and $NV2 = N2$ are initialized after the first stage of merging, and these two parameters are recursively updated at the end of each merging process. Based on the foundation we developed and presented in Table III, these parameters are used to determine the vertex weight of the next region, which has to be either $V1$ or $V2$. After verifying the condition of the merged region in terms of $NV1$ and $NV2$, the range of the DF value is determined. Using DF, the vertex weight of the *Rnext* can be found using Table III. Fig. 6 only includes two cases of DF values, $0 < \mathrm{DF} < 0.5$ and $0.5 < \mathrm{DF} < 1$. When the case of $\mathrm{DF} = 0.5$ occurs, the vertex weight of *Rnext* can be either $V1$ or $V2$ regardless of $NV1$ and $NV2$, and, therefore, the prediction method is simpler. At stage 2, either $NV1$ or $NV2$ is increased dependent on the vertex weight of $R3$. If the vertex weight of $R3$ is $V1$, then update $NV1$ to $NV1 = NV1 + N3$. Otherwise, update $NV2$ to $NV2 = NV2 + N3$. The condition of the newly merged region is verified by comparing the updated $NV1$ and $NV2$. Then choose the vertex weight of *Rnext* for the next round of merging. At stage 3, the same prediction method is executed and again either $NV1$ or $NV2$ is accumulated. This process can go on until no more connecting regions with vertex weight $V1$ or $V2$ are found.

Throughout the multiple-stage merging, recalculation of vertex weight and link weight, together with the process of sorting link weight, is not required. The final vertex weight of the merged region and link weights for adjacent links are evalu-

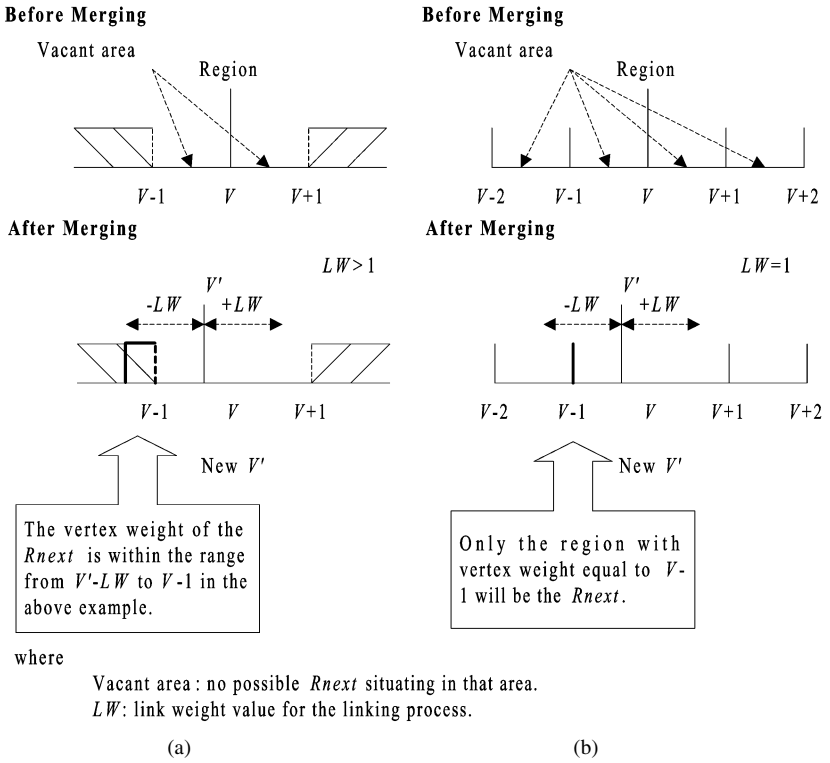


Fig. 7. Linking processes for real link weight and integer link weight. (a) LPLW-R (Real link weight). (b) LPLW-O (Integer link weight).

ated when the multiple-stage merging is complete. The process of saving the resulting tree is invoked at the end but it is to save a tree instead of a link. Therefore, the overall performance of the algorithm is enhanced and better optimization can be achieved. The greatest achievement of the LPLW-O with prediction is that the cost of the sorting algorithm is reduced significantly. Another advantage of the LPLW-O with prediction is that it does not involve any approximation, therefore, the resulting tree is truly identical to [30] and [31].

C. LPLW-R

The LPLW-R is devoted to link weight in real number. It is applied after the linking processes for integer link weight, LPLW-Z and LPLW-O. The approach in LPLW-R is completely different from LPLW-Z and LPLW-O. It is far more important than LPLW-Z and LPLW-O, as our experimental results show that it occupies 57.8% of the entire RSST. Unfortunately, the linking process for these links is unlikely to be expedited due to its complex linking interrelationship.

A comparison of LPLW-R and LPLW-O is given in Fig. 7. The LPLW-O is chosen as representative of the linking process for integer link weight in our analysis, because the LPLW-Z is trivial. Fig. 7(a) shows an example of the linking process for real link weight, LPLW-R, while Fig. 7(b) shows an example of the linking process for integer link weight, LPLW-O. Each example contains a region with its initial vertex weight V represented by a vertical line. As LPLW-R is executed after LPLW-Z and LPLW-O, there must be no region with vertex weight between $V-1$ and $V+1$. Those regions must have been completely consumed by LPLW-Z and LPLW-O. Unlike LPLW-O in which vertex weights of possible next regions situate at positions $V+1, V+2, V+3, V-1, V-2, V-3$, etc., as depicted in Fig. 7(b), vertex weights in LPLW-R could be any real numbers outside that vacant area. This is the major difference between the two types of linking processes.

In addition, the V' value is bounded between $V-1$ and $V+1$, and possible next regions must be with vertex weight either $V-1$ or $V+1$ in LPLW-O. However, in the case of LPLW-R, the vertex weight V' can be any real numbers from $V+\mathrm{LW}$ to $V-\mathrm{LW}$ and vertex weights of possible next regions are bounded within the range of $V'-\mathrm{LW}$ and $V-1$ if $V'<V$ after merging. As the range of $V'-\mathrm{LW}$ and $V-1$ is continuous, the prediction method is required to examine vertex weights of all *Rnext* and, therefore, it is no longer applicable to this type of linking process. Thus, the linking process of the conventional RSST algorithm is used when accuracy is a concern of the application. Kwok *et al.* [20], [21], [23], [25] suggested the use of truncated link weight throughout the linking process, so that the LPLW-R can be simplified. However, this introduces errors into the resulting RSST. As the objective of this paper is to produce an RSST the same as that generated by the conventional algorithm, the truncated version or the error-prone RSST is not acceptable.

IV. Linking Properties for RSST

Applying the linking processes and linking properties presented in Section IV, we propose an efficient RSST algorithm using linking properties. Fig. 8 depicts the flowchart of the proposed algorithm. The proposed algorithm is modified from the conventional RSST with the inclusion of LPLW-Z, LPLW-O, and LPLW-R. It consists of two stages: initialization stage and linking process. A number of additional modules are introduced to implement the three linking processes. Functions of these additional modules are given in Table IV.

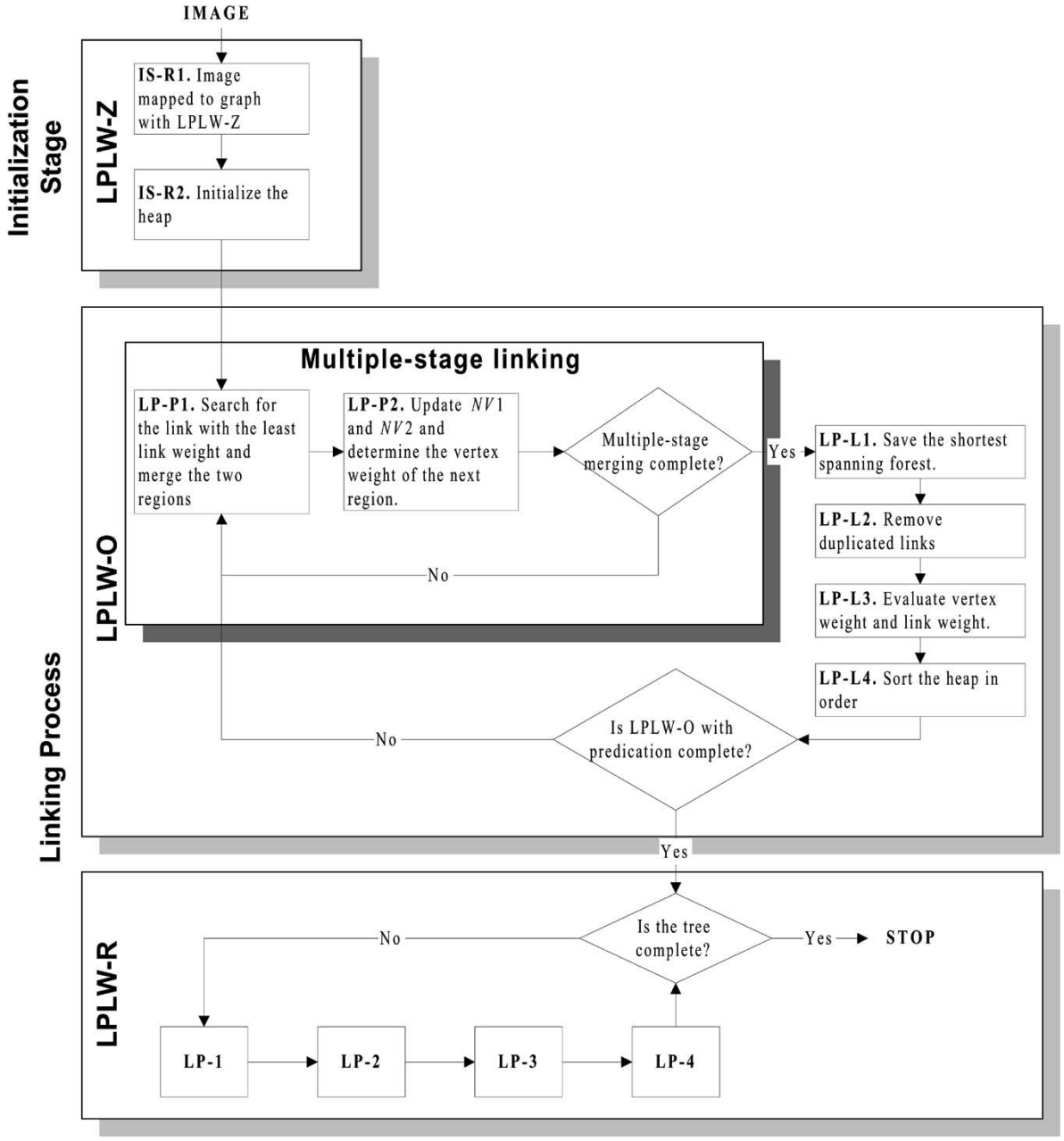
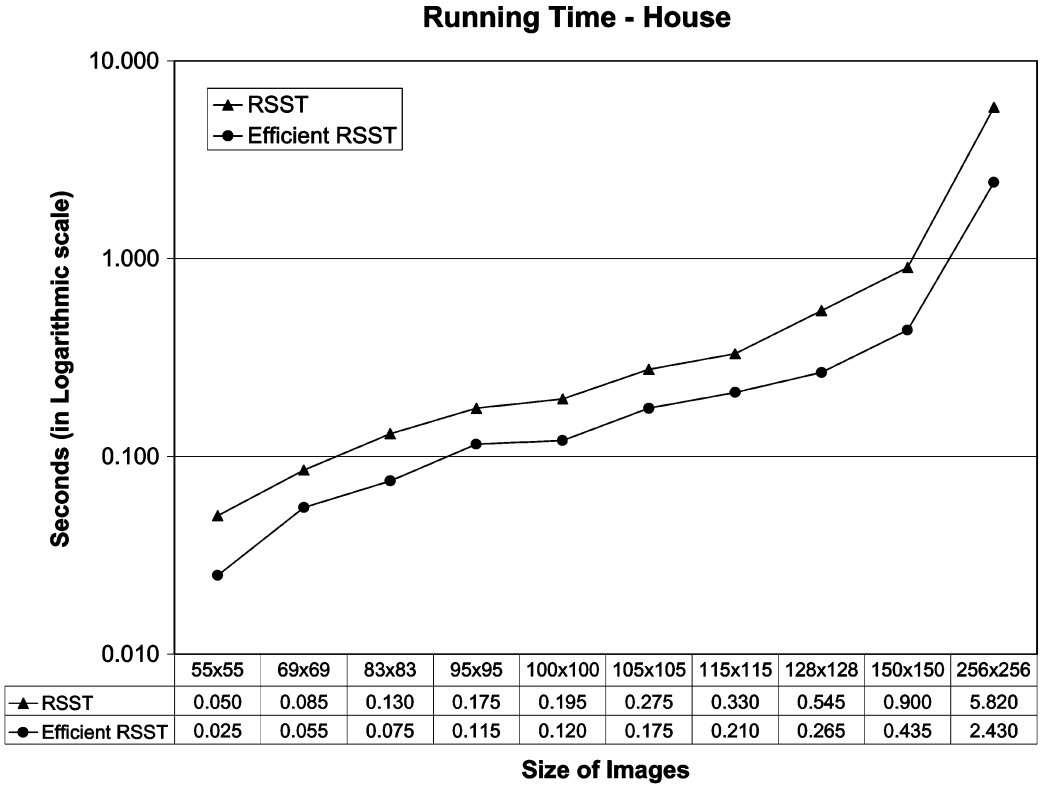


Fig. 8. Flowchart of the efficient RSST algorithm using linking properties.

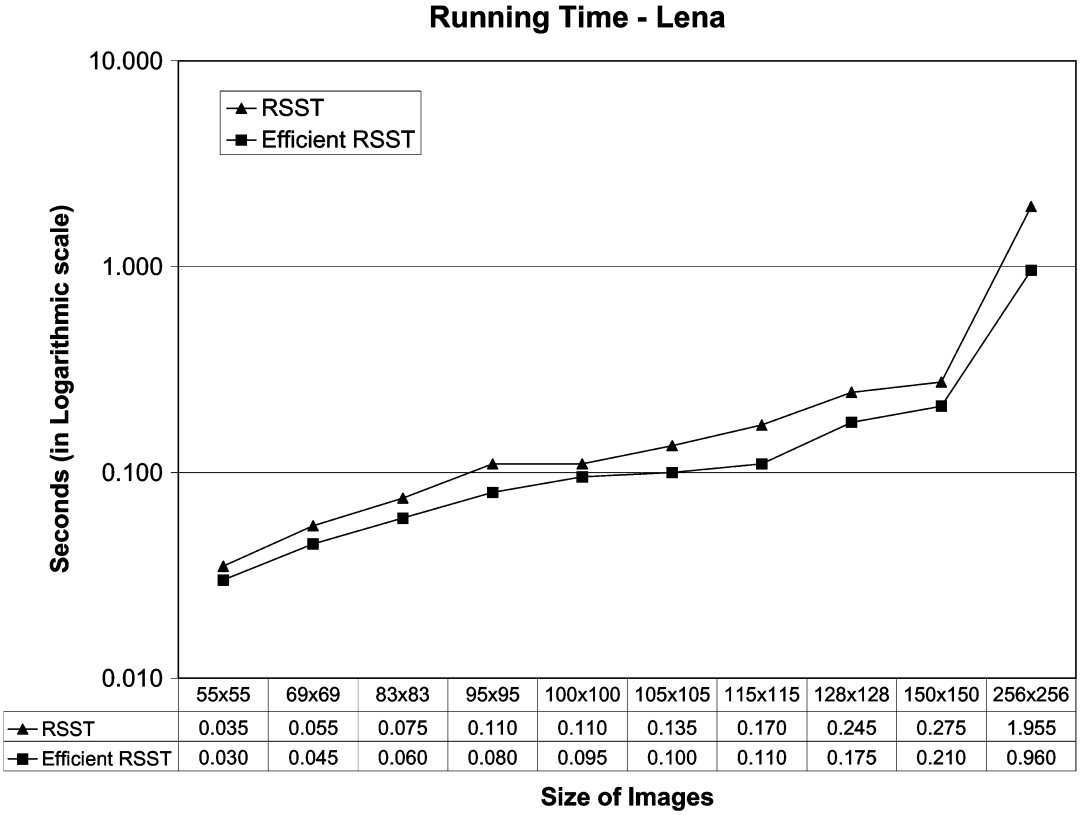
TABLE IV
ADDITIONAL MODULES IN THE EFFICIENT RSST ALGORITHM

	Linking Process	Module	Function
Initialization stage	LPLW-Z	IS-R1	Collect links with zero link weight and add them to the resulting shortest spanning tree.
	LPLW-Z	IS-R2	Initialize the heap and sort links in order (same as Figure 1, IS-2).
Linking process	LPLW-O	LP-P1	Identify the next link either by fetching a link from the heap or a link suggested by the prediction method, and then merge its connecting two regions.
	LPLW-O	LP-P2	Operations may refer to Table 3 and Figure 6.
	LPLW-O	LP-L1	Save the resulting tree.
	LPLW-O	LP-L2	Remove duplicated links (same as Figure 1, LP-3).
	LPLW-O	LP-L3	Evaluate vertex weight based on Equ. (2) and link weight based on Equ. (3).
	LPLW-O	LP-L4	Sort links in the heap (same as Figure 1, LP-4).



	55x55	69x69	83x83	95x95	100x100	105x105	115x115	128x128	150x150	256x256
RSST	0.050	0.085	0.130	0.175	0.195	0.275	0.330	0.545	0.900	5.820
Efficient RSST	0.025	0.055	0.075	0.115	0.120	0.175	0.210	0.265	0.435	2.430

Fig. 9. Running times of the RSST and the efficient RSST algorithms with various image sizes of the *House* image.



	55x55	69x69	83x83	95x95	100x100	105x105	115x115	128x128	150x150	256x256
RSST	0.035	0.055	0.075	0.110	0.110	0.135	0.170	0.245	0.275	1.955
Efficient RSST	0.030	0.045	0.060	0.080	0.095	0.100	0.110	0.175	0.210	0.960

Fig. 10. Running time of RSST and efficient RSST algorithms with various image size of the *Lena* image.

The LPLW-Z is embedded in the initialization stage. The two functional modules involved are IS-R1 and IS-R2. The implementation of the module IS-R1 is similar to its counterpart in the conventional algorithm, IS-1, but IS-R1 is also required to manage links with link weight equal to zero. The output of this stage is a nontrivial shortest spanning forest.

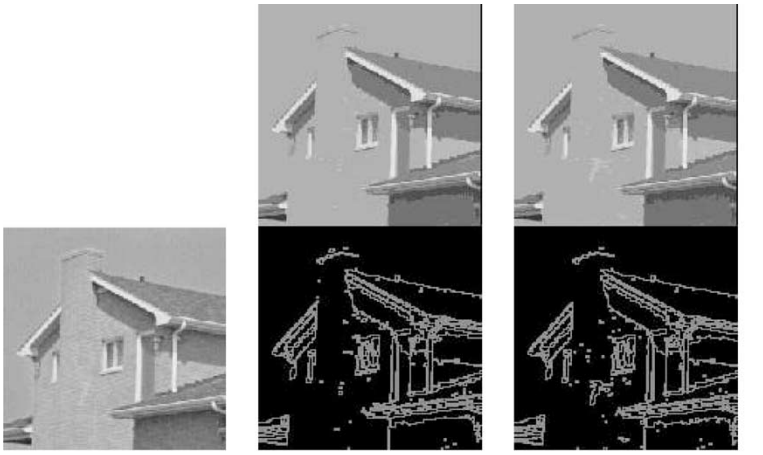


Fig. 11. Segmented images and edge maps of *House* using the RSST and the efficient RSST.

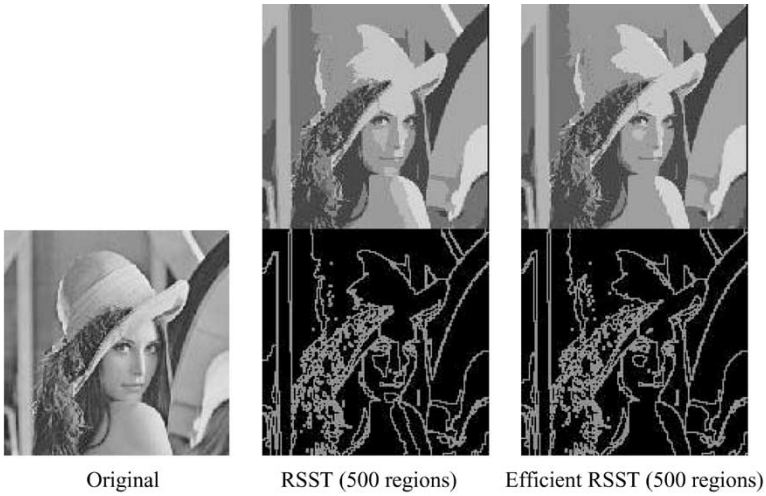


Fig. 12. Segmented images and edge maps of *Lena* using the RSST and the efficient RSST.

The linking process of the proposed RSST algorithm is divided into two parts, the LPLW-O with prediction and the LPLW-R. It is because link weights of the remaining links are greater than or equal to one. The LPLW-O with prediction outperforms the conventional LPLW-O since expensive processes (vertex weight, link weight evaluation, and the sorting algorithm) are no longer required for each merging process. Besides, it is no longer a link-based process, but it uses the tree as a basic joining element instead. These greatly improve the efficiency and performance of the RSST algorithm. Other modules, such as LP-L1, LP-L2, LP-L3, and LP-L4 are similar to their corresponding modules in the conventional algorithm. They are used for updating the resulting tree and preparing for the next round of the linking process, after the multiple-stage merging process.

V. Experimental Results

Two testing images, namely, *Lena* and *House*, with various image sizes were used to examine the performance of the proposed algorithm and the conventional RSST algorithm by Morris *et al.*. The experiments were conducted on a SUN Ultra 1200 machine (with a 200 MHz Ultra SPARC-I processor and 1G RAM). Our running-time experimental results show that the proposed RSST algorithm outperforms the conventional RSST algorithm, as shown in Figs. 9 and 10.

The segmented images and edge maps of both the *House* and *Lena* images are shown in Figs. 11 and 12, respectively. Although the proposed algorithm and the RSST algorithm basically follow the same rules in building the RSST representation, their segmented images and edge maps are different as the RSST representation is not deterministic.

The proposed efficient RSST algorithm outperforms the conventional RSST algorithm because our approach takes redundant links into consideration in LPLW-Z and applies prediction in LPLW-O. Any redundant links can always be found in images; for example, there are redundant zero links wherever a region exists. Table V gives percentages of various linking processes in building RSST representations for *House* and *Lena* with a resolution of 150×150 pixels when the proposed algorithm is in use.

The experimental results show that about 40% of links (in both LPLW-Z and LPLW-O) were accelerated by using the proposed efficient RSST algorithm, while the rest of the links (in LPLW-R) rely on the conventional RSST algorithm for both testing images.

TABLE V
PERCENTAGES OF LINK ACHIEVED BY LPLW-Z, LPLW-O, AND LPLW-R LINKING PROCESSES

	"House" image (150x150) - Percentage of links	"Lena" image (150x150) - Percentage of links
LPLW-Z	15.4%	16.8%
LPLW-O	26.8%	26.6%
LPLW-R	57.8%	56.6%

VI. CONCLUSION

In this paper, we presented an efficient RSST algorithm using linking properties. This paper categorizes the linking process into three classes based on the link weight value. The three linking processes are LPLW-Z, LPLW-O, and LPLW-R. In LPLW-Z, which handles links with the link weight equal to zero, it is noted that this linking process is trivial and can be included in the initialization stage. In our experiment, 15.4% and 16.8% of the resulting RSST are completed after the LPLW-Z is applied to the testing images of *House* and *Lena*, respectively. For the LPLW-O, we explore the linking properties of this linking process and propose a LPLW-O with prediction to expedite the LPLW-O. The proposed linking process includes a prediction method, which is basically a reference lookup table for the next region. The lookup table transforms the merging process to a process that solely relies on comparison operation, comparing sizes of the merging regions N. The prediction method successfully eliminates the sorting algorithm and the process of evaluating link weight and vertex weight from the merging process. As the multiple-stage merging is more practical and desirable, we demonstrate the applicability of the LPLW-O with prediction in this paper. Using the same testing images, it is noted that 26.8% and 26.6% of the resulting RSST are contributed by the LPLW-O with prediction. It is evident that LPLW-R cannot be expedited due to its complex linking interrelationship. Therefore, it is the most expensive linking process among the three. Unfortunately, this occupies about 57% of links to the overall RSST for both testing images. Further studies on the enhancement and optimization can be achieved in LPLW-Z and LPLW-O, but not in LPLW-R.

Future works include the exploration of potential applications of linking properties to RSST applications. For example, in the application of the multiscale implementation of RSST [14], the LPLW-Z and the LPLW-O with prediction can be applied to the highest resolution of the testing image, and the partial RSST may be applied to lower resolutions of images. It is expected that the results could be useful to the development of progressive coding and video coding.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*. Reading, MA: Addison-Wesley, 1983.

[2] A. A. Alatan, "A technical report on very low bit-rate region segmentation based videophone systems," Imperial College, London, U.K., Technical Report, Sept. 1992.

[3] A. A. Alatan, L. Onural, M. Wollborn, R. Mech, E. Tuncel, and T. Sikora, "Image sequence analysis for emerging interactive multimedia services—The European COST 211 framework," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 802–813, Nov. 1998.

[4] A. A. Alatan, E. Tuncel, and L. Onural, "A rule-based method for object segmentation in video sequences," in *Proc. IEEE. Int. Conf. Image Processing*, vol. 2, Oct. 1997, pp. 522–525.

[5] Y. Avrithis, N. Tsapatsoulis, and S. Kollias, "Color-based retrieval of facial images," presented at the 10th Eur. Signal Processing Conf. Signal Processing Theories and Applications, Tampere, Finland, 2000.

[6] Y. S. Avrithis, A. D. Doulamis, N. D. Doulamis, and S. D. Kollias, "A stochastic framework for optimal key frame extraction from MPEG video databases," *Comput. Vis. Image Understanding*, vol. 75, pp. 3–24, July 1999.

[7] C. Berge and A. Ghouila-Houri, *Programming, Games, and Transportation Networks*. New York: Wiley, 1965.

[8] M. J. Biggar, O. J. Morris, and A. G. Constantinides, "Segmented-image coding: Performance comparison with the discrete cosine transform," in *Proc. IEE (Part F)*, vol. 135, Apr. 1988, pp. 121–132.

[9] D. Cheriton and R. E. Tarjan, "Finding minimum spanning trees," *SIAM J. Computing*, vol. 5, no. 4, pp. 724–742, 1976.

[10] N. Christofides, *Graph Theory: An Algorithmic Approach*. New York: Academic, 1975.

[11] S. Cooray, N. O'Connor, S. Marlow, N. Murphy, and T. Curran, "Hierarchical semi-automatic video object segmentation for multimedia applications," in *Proc. SPIE*, vol. 4519, 2001, pp. 10–19.

[12] E.W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[13] A. Doulamis, N. Doulamis, K. Ntalianis, and S. Kollias, "Efficient unsupervised content-based segmentation in stereoscopic video sequences," *Int. J. Artificial Intell. Tools (Architectures, Languages, Algorithms)*, vol. 9, no. 2, pp. 277–303, 2000.

[14] A. D. Doulamis, N. Doulamis, and S. Kollas, "Non-sequential video content representation using temporal variation of feature vectors," *IEEE Trans. Consumer Electron.*, vol. 46, pp. 758–768, Aug. 2000.

[15] N. D. Doulamis, A. D. Doulamis, Y. S. Avrithis, K. S. Ntalianis, and S. D. Kollias, "Efficient summarization of stereoscopic video sequences," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 501–517, June 2000.

[16] S. E. Goodman and S. T. Hedetniemi, *Introduction to the Design and Analysis of Algorithms*. New York: McGraw-Hill, 2002.

[17] Z. Hai, S. Narendra, and W. Nicholls, "Efficient minimum spanning tree construction without Delaunay triangulation," *Inform. Processing Lett.*, vol. 81, no. 5, pp. 271–276, 2002.

[18] A. Kershenbaum and R. V. Slyke, "Computing minimum spanning trees efficiently," in *Proc. 25th ACM Nat. Conf.*, Boston, MA, 1972, pp. 518–527.

[19] S. H. Kwok, "Hierarchical structure of time window for video processing and coding," presented at the Proc. 3rd World Multiconf. Systemics, Cybernetics and Informatics and 5th Int. Conf. Information Systems Analysis and Synthesis, Brisbane, Australia, 1999, pp. 392–397.

[20] S. H. Kwok and A. G. Constantinides, "A fast recursive shortest spanning tree for image segmentation and edge detection," *IEEE Trans. Image Processing*, vol. 6, pp. 328–332, Feb. 1997.

[21] ——, "An optimal parallel algorithm of recursive shortest spanning tree for image segmentation and edge detection," in *Proc. IEEE Singapore Int. Conf. Signal Processing, Circuits and Systems*, 1995, pp. 215–220.

[22] ——, "A parallel recursive shortest spanning tree algorithm for image segmentation in distributed computing environment," *J. Parallel Distrib. Comput.*, vol. 56, no. 3, pp. 181–207, Mar. 1999.

[23] ——, "A parallel recursive shortest spanning tree in distributed computing environment for image segmentation," *J. Parallel Distrib. Comput.*, vol. 56, no. 3, pp. 181–207, Mar. 1999.

[24] S. H. Kwok, W. C. Siu, and A. G. Constantinides, "Adaptive temporal decimation algorithm with dynamic time window," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 104–111, Feb. 1998.

[25] ——, "A fast recursive shortest spanning tree for image segmentation," in *Proc. Int. Conf. Neural Networks and Signal Processing*, Nanjing, China, 1995, pp. 1386–1389.

[26] ——, "A scaleable and adaptive temporal segmentation algorithm for video coding," *Graphical Models and Image Processing*, vol. 59, no. 3, pp. 128–138, 1997.

[27] O. J. Morris and A. G. Constantinides, "Progressive image coding from a spanning tree image representation," presented at the Onzieme Colloq., Nice, France, 1987.

[28] ——, "Segmented video coding," in *Proc. IEEE. Int. Conf. Acoustics, Speech, and Signal Processing*, 1988, pp. 1108–1111.
[29] ——, "Thin link coding techniques," presented at the Int. Conf. Digital Signal Processing, Florence, Italy, 1987.
[30] O. J. Morris, J. M. Lee, and A. G. Constantinides, "Graph theory for image analysis: An approach based on the shortest spanning tree," in *Proc. Inst. Electr. Eng. (Part F)*, vol. 133, 1986, pp. 146–152.
[31] ——, "A unified method for segmentation and edge detection using graph theory," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processingc*, 1986, pp. 2051–2055.
[32] K. S. Ntalianis, N. D. Doulamis, A. D. Doulamis, and S. D. Kollias, "An active contour-based video object segmentation scheme for stereoscopic video sequences," in *Proc. 10th Mediterranean Electrotechnical Conf. Information Technology and Electrotechnology for the Mediterranean Countries*, May 2000, pp. 554–557.
[33] S. Pettie and V. Ramachandran, "An optimal minimum spanning tree algorithm," in *J. ACM*, vol. 49, Jan. 2002, pp. 16–34.
[34] F. Suraweera, "A fast algorithm for the minimum spanning tree," *Computers in Industry*, vol. 13, no. 2, pp. 181–185, Nov. 1989.
[35] D. Tancharoen, S. Jitapunkul, S. Triamlumlerd, P. Kittipanya-Ngam, and S. Chompun, "Object segmentation based on multiple features for low bit rate video coding," in *Proc. 5th Int. Conf. Signal Processing and the 16th World Computer Congress*, 2000, pp. 975–978.
[36] E. Tuncel and L. Onural, "Utilization of the recursive shortest spanning tree algorithm for video-object segmentation by 2-D affine motion modeling," *IEEE Trans. Circuits Syst. Video Tech.*, vol. 10, pp. 776–781, Aug. 2000.
[37] ——, "Video object segmentation by extended recursive-shortest-spanning-tree method," in *Proc. IEEE-EURASIP Workshop Nonlinear Signal and Image*, 1999, pp. 23–27.
[38] T. Vlachos and A. G. Constantinides, "A graph-theoretic approach to color image segmentation and contour classification," presented at the Int. Conf. Image Processing and its Applications, Maastricht, The Netherlands, 1992.
[39] A. C. Yao, "An $O(|E|\log\log|V|)$ algorithm for finding minimum spanning trees," *Inform. Processing Lett.*, vol. 4, no. 1, pp. 21–23, 1975.
[40] Z. Yongqin and A. G. Constantinides, "Perceptual saliency weighted segmentation algorithm," in *7th Int. Conf. Image Processing and Its Applications*, 1999, pp. 562–566.
[41] Y. Zeng, "Perceptual segmentation algorithm and its application to image coding," in *Proc. Int. Conf. Image Processing*, 1999, pp. 820–824.
[42] ——, "Region coding and detail compensation," in *Proc. Int. Conf. Signal Processing*, New York, NY, 1996, pp. 1227–1230.

Sai Ho Kwok received the B.Eng. degree (with honors) in electronic and communications engineering from the University of North London, London, U.K. He received the Diploma of Imperial College (DIC) from the Imperial College of Science, Technology and Medicine, London, and the Ph.D. degree in digital image processing from the University of London.

He is currently an Assistant Professor at the Department of Information and Systems Management, Hong Kong University of Science and Technology (HKUST), Hong Kong. He was a Visiting Scholar and a Research Assistant in the Department of Electronic and Information Engineering at the Hong Kong Polytechnic University. His research has been published in *Communications of the ACM (CACM), International Journal of Electronic Commerce (IJEC), Decision Support Systems (DSS), Electronic Markets (EM), IEEE Transactions on Circuits and Systems for Video Technology*, IEEE TRANSACTIONS ON IMAGE PROCESSING, *Graphical Models and Image Processing, Journal of Parallel and Distributed Computing, Optical Engineering, International Journal of Information Technology and Decision Making (IJITDM), Journal of Applied Systems Studies (JASS), and ACM SIGecom Exchanges*. He is a Senior Editor of the *Journal of Information Technology Theory and Application* (JITTA) and is on the Editorial Board of the *Journal of Database Management* (JDM). He has served as a member of several technical program committees and organizing committees. His research interests include image segmentation, video coding, watermarking, peer-to-peer technology, digital rights management, knowledge management, and electronic commerce applications.

A. G. Constantinides (S'68–M'78–SM'78–F'98) is the Professor of Signal Processing and the Head of the Signal Processing and Digital Systems Section of the Department of Electrical and Electronic Engineering at Imperial College, London, U.K. He has been actively involved with research in various aspects of digital filter design, digital signal processing, and communications for more than 30 years. Professor Constantinides' research spans a wide range of Digital Signal Processing, from the theoretical, as well as the practical, points of view. His recent work has been directed toward the demanding signal processing problems arising from the various areas of telecommunication. This work is supported by research grants and contracts from various government and industrial organisations. He has published several books and papers in learned journals in the area of Digital Signal Processing and its applications.

Professor Constantinides has served as the First President of the European Association for Signal Processing (EURASIP). He has been on, and is currently serving as, a member of many technical program committees of the IEEE, the IEE and other international conferences. He has organized the first international series of meetings on Digital Signal Processing, initially in London, in 1967, and in Florence since 1972. In 1985, he was awarded the Honour of Chevalier, Palmes Academiques, by the French government, and in 1996, the promotion to Officier, Palmes Academiques. He holds honorary doctorates from European and Far Eastern Universities, several Visiting Professorships, Distinguished Lectureships, Fellowships and other honors around the world. He served as a Member of the Board of Governors of the IEEE Signal Processing Society, a member of several Technical Committees of the IEEE and the Institution of Electrical Engineers (IEE), U.K., and is on the Editorial Boards of many professional journals. He is a Fellow of IEE and an honorary member of Eta Kappa Nu.

Wan-Chi Siu (S'77–M'77–SM'90) received the Ph.D. degree from the Imperial College of Science, Technology and Medicine, London, U.K., in 1984.

He was with the Chinese University of Hong Kong from 1975 to 1980. He joined the Hong Kong Polytechnic University, Hong Kong, as a Lecturer in 1980 and has been Chair Professor since 1992. He was also Associate Dean of the Engineering Faculty from 1992 to 1994, Head of the Department of Electronic and Information Engineering from 1994 to 2000, and Dean of the Engineering Faculty from 2000 and 2002. Since September 1998, he has been Director of Centre for Multimedia Signal Processing. He has published over 100 research papers in international journals, including IEEE transactions and IEE proceedings. He is an Editor of the recent book, *Multimedia Information Retrieval and Management* (Berlin, Germany: Springer-Verlag, 2003). He has held the position of General Chair or Technical Program Chair of many international conferences. He is a member of the editorial board of the *Journal of VLSI Signal Processing Systems for Signal, Image, Video Technology*, the *EURASIP Journal on Applied Signal Processing*, and other journals. From 1991 to 1995, he was a member of the Physical Sciences and Engineering Panel of the Research Grants Council (RGC), Hong Kong Government, and in 1994, he chaired the first Engineering and Information Technology Panel to assess the research quality of 19 cost centers (departments) from all universities in Hong Kong. His research interests include digital signal processing, fast computational algorithms, transforms, wavelets, image and video coding, and computational aspects of pattern recognition and neural networks.

Prof. Siu was a Guest Editor and Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: ANALOG AND DIGITAL SIGNAL PROCESSING from 1995 to 1997. He was a Technical Program Chair of the IEEE International Symposium on Circuits and Systems (ISCAS'97), and is the General Chair of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'2003).