

Correspondence

A Fast Recursive Shortest Spanning Tree for Image Segmentation and Edge Detection

S. H. Kwok and A. G. Constantinides

Abstract—This correspondence presents a fast recursive shortest spanning tree algorithm for image segmentation and edge detection. The conventional algorithm requires a complexity of $O(n^2)$ for an image of n pixels, while the complexity of our approach is bounded by $O(n)$, which is a new lower bound for algorithms of this kind. The total memory requirement of our fast algorithm is 20% smaller.

I. INTRODUCTION

Image segmentation is considered to be a preprocessing for image analysis, image understanding, robotics vision, etc. To a certain extent, it is also an essential processing tool for segmented image coding and video coding. It subdivides an image into its constituent parts or objects. The level to which this subdivision is carried out depends on the problem being solved. Hierarchical image segmentation approaches can be easily adapted to various applications. Segmentation algorithms of this type will have many more selling points. Edge detection is used to outline the boundaries of objects.

The first recursive shortest spanning tree (RSST) algorithm proposed by Morris *et al.* [1] was used for image segmentation and edge detection problems. Morris *et al.*, proved that segmentation and edge detection are dual problems. Two functions are included in one algorithm. Their algorithm is proven to be highly accurate in defining regions. Because of its hierarchical structure, any number of region representations of an image can be chosen as desired. This feature is beneficial to both segmented image coding [2], [3] and segmented video coding [4]. A progressive image coding scheme [5] using the RSST was also proposed. Alatan [6] used it for video-phone systems. It can be adapted to some segmented and object oriented based video coding systems [7]–[10] effectively. Vlachos *et al.* [11] extended the RSST algorithm to color images by introducing red–green–blue (RGB) components into the cost function. The RSST is a powerful tool for various image processing applications. Speed is a great concern for some applications therefore this fast algorithm study has been conducted.

II. SPEED CONSTRAINTS AND COMPLEXITY OF THE RSST

Given an image of size $M \times N$, there are $(M \times N - 1)$ required links in a shortest spanning tree. According to the conventional RSST algorithm [1], [2] shown in Fig. 1, each link found requires

Manuscript received March 3, 1995; revised May 10, 1996. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Patrick A. Kelly.

S. H. Kwok is with the Department of Electronic Engineering, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, and the Department of Electrical and Electronic Engineering, Imperial College of Science, Technology, and Medicine, London SW7 2BT, U.K. (e-mail: enshkwok@enepu01.en.hkpk.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.

Publisher Item Identifier S 1057-7149(97)00342-4.

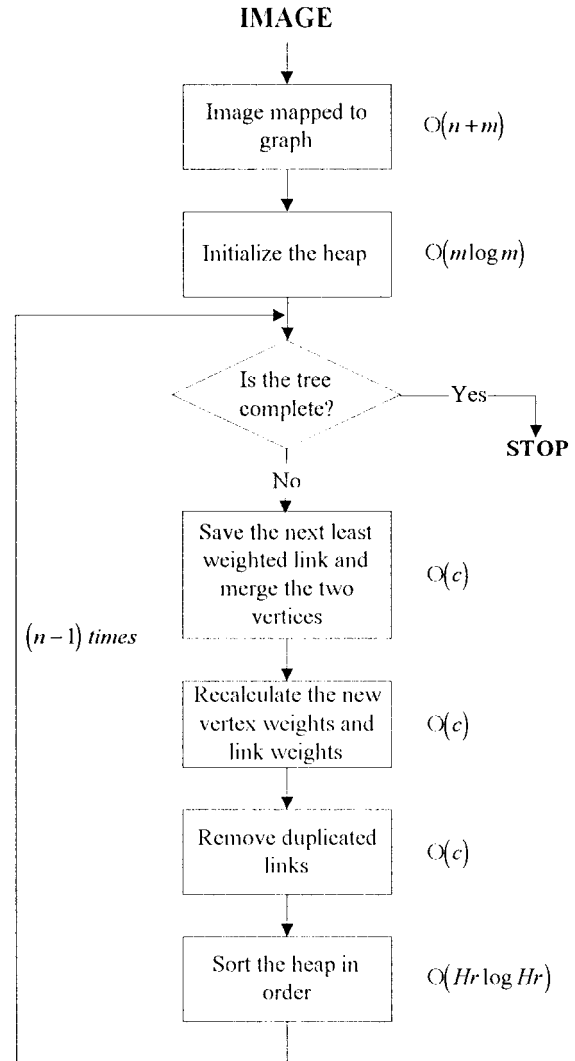


Fig. 1. Flowchart of the RSST.

a rearrangement of the heap, which maintains all remaining links in order. If there are now any links that are identical, then the redundant ones are removed. Due to the removal of duplicated links and saved links, a sorting algorithm is needed to shuffle the heap. It certainly slows down the construction process. Therefore, the execution time is heavily dependent upon the choice of the sorting algorithm, which is also true for the fastest applicable known sorting algorithm, heap sort with a complexity of $O(k \log k)$ for k keys. This is certainly a bottleneck of the algorithm.

Referring to Fig. 1, the complexity of the RSST is analyzed as follows. Assume that n is the total number of vertices, and m is the total number of the links (edges). For both Morris *et al.*'s algorithm and the proposed algorithm, the image is mapped into a graph that produces $m = 2n - 2\sqrt{n}$ links. The heap sort algorithm is used for the implementation of Morris *et al.*'s algorithm.

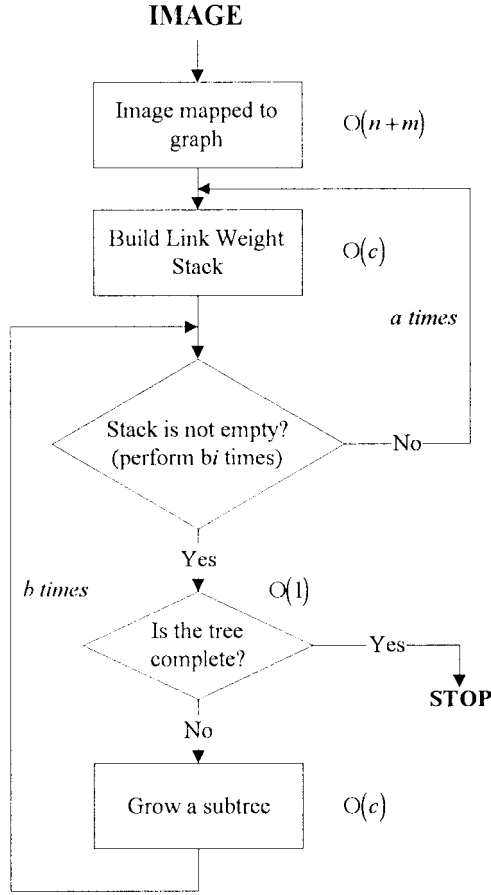


Fig. 2. Flowchart of the FRSST.

Obtaining from Fig. 1, the complexity of Morris *et al.*'s algorithm is

$$T_{\text{Morris}} = O(n + m) + O(m \log m) + \sum_{i=0}^{n-2} [3 \times O(c) + Hr(i) \log Hr(i)]$$

where Hr is the number of remaining links in the heap and c is a small constant.

After eliminating some constants and using the relation of $m = 2n - 2\sqrt{n}$

$$T_{\text{Morris}} = O(m \log m) + \sum_{i=0}^{n-2} Hr(i) \log Hr(i).$$

On average, duplicated links at stage i proportional to $b \times [i/s \times m/(n-1)]$ with $b \approx 1$ and s is the total number of stages; therefore, there are $m - b \times [i/s \times m/(n-1)] - \text{link}_{\text{saved}}$ remaining links left in the stack (note: $\text{link}_{\text{saved}}$ is the number of saved links). Then let $k = \{m - b[i/s \times m/(n-1)] - \text{link}_{\text{saved}}\}$, it yields

$$\sum_{i=0}^{n-2} Hr(i) \log Hr(i) = \sum_{i=0}^{n-2} (k \log k)$$

and, its upper bound is $\int k \log k dk = O(k^2) = O(m^2)$ Hence

$$T_{\text{Morris}} = O(m \log m) + O(m^2) = O(m^2) = O(n^2).$$

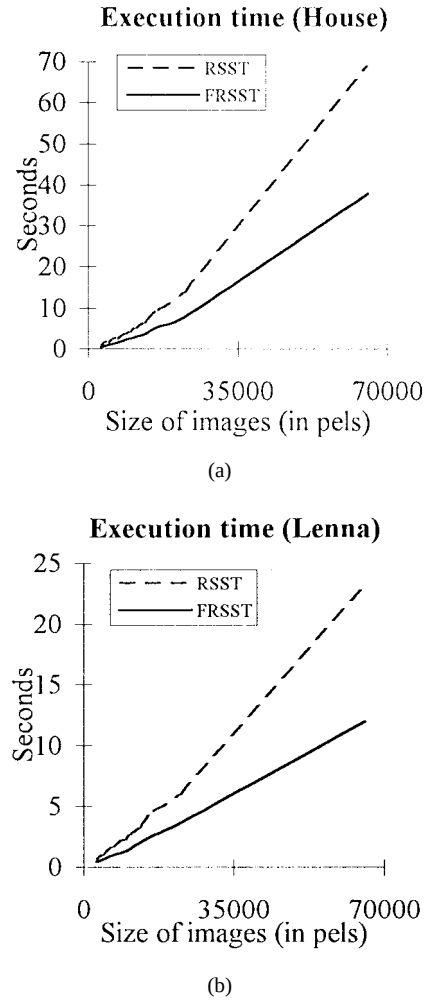


Fig. 3. Experimental results of the RSST and FRSST algorithms with House and Lenna images. (a) Execution time for House. (b) Execution time for Lenna.

III. THE FRSST ALGORITHM

According to the discussion in the last section, we can now present a more efficient algorithm in terms of speed, let us refer to it as the fast RSST (FRSST) algorithm. The flowchart of the FRSST is depicted in Fig. 2. The FRSST differs from the RSST by i) constructing the spanning tree in a region growing fashion instead of a single link; ii) removing the sorting algorithm from the construction process; and iii) approximately evaluating the link weight function in order to maintain best possible output quality.

The FRSST is designed in a region growing fashion. Each region can be represented by a recursive shortest spanning subtree. Two apparent reasons why the FRSST runs faster are that it i) does not need sorting and ii) is a region-based process. The speed can obviously be improved by removing the sorting algorithm. Moreover, the region-based mapping can improve the performance of the algorithm.

Instead of adding a link at a time to grow the spanning tree, a spanning subtree is used by the FRSST. It also takes the advantage of updating neighboring link weights. Each updated link is needed for rearranging the heap in the conventional algorithm. The FRSST does not update neighboring link weights but searches for the next link in the region. Next links must be either already stored in the link weight stack or created by previous merging. As link weights are changed

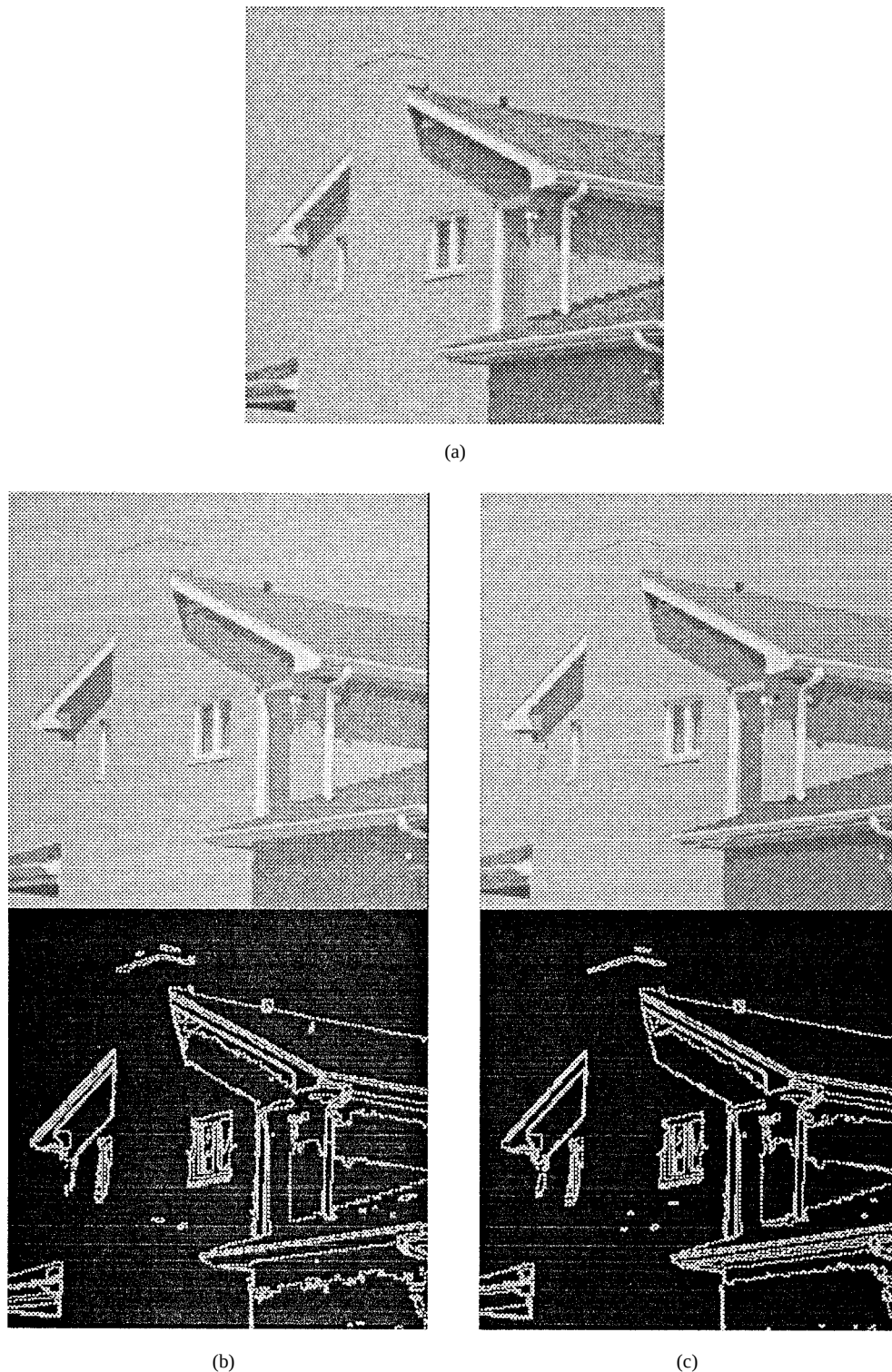


Fig. 4. Segmented images and edge maps of House using the RSST and the FRSST. (a) Original. (b) RSST (500 regions). (c) FRSST (500 regions).

after each merging, there is no need to keep them. The strategy of the FRSST is to measure link weights when they are needed. Due to these arrangements, empirical results show that the FRSST requires a space of $62n - 32\sqrt{n} - 1$ while Morris *et al.*'s algorithm requires $76n - 40\sqrt{n}$. They both hold the same order of space $O(n)$.

The definition of the next link is that a link connects to the merged region with link weight equal to or less than the link weight of the

initial link of the subtree. When more than one next links are found, the one with the lowest weight is used.

The link weight function used in the FRSST is a truncated version of an absolute difference function (TADF). It is basically an absolute difference function (ADF) but is truncated to integer values while the building link weight stacks and real number link weights are used within the region growing process. The output quality will be



Fig. 5. Segmented images and edge maps of Lenna using the RSST and the FRSST. (a) Original. (b) RSST (350 regions). (c) FRSST (350 regions).

degraded slightly due to the truncation error. The maximum possible number of stacks is equal to upper bound of the link weight function and it is very often up to one half of the range for head-and-shoulder type of images. The FRSST tends to use links in an ascending order, from 0 link weight stack to the largest possible link weight stack.

Prior to the subtree growing process, it is necessary to prepare a link weight stack, which is to store all links with link weights equal to i . Due to the fact that the shortest spanning tree is not deterministic,

links in the link weight stack $[i]$ can be used in an arbitrary order. A subtree is grown by means of one of these links. Very similar to the conventional algorithm, the processes for updating a vertex weight and removing duplicated links are also needed, but it is not necessary to update link weights. An additional scanning for the next link follows. A recursive loop will grow a subtree. Note that an increase in the size of the subtree will decrease the number of links b_i in the link weight stack $[i]$ (see Fig. 2).

IV. COMPLEXITY OF THE FRSSST ALGORITHM

The complexity of the FRSSST algorithm is analyzed below. Based on the flow chart of the FRSSST as shown on Fig. 2, we obtain

$$T_{\text{FRSSST}} = O(n + m) + \sum_{i=0}^{a-1} \left[O(c) + \sum_{j=0}^{b-1} O(c) \right].$$

The pixel domain mapping is used in this analysis. The variable a will start from 0 (a will start from 1 when region domain mapping is used). The value of the constant a , which has to be within the dynamic range of the link weight function is the maximum number of link weight stack $[i]$ used and b_i is the number of links in the stack $[i]$. The algorithm will be terminated when the total number $(n - 1)$ of links are saved. It can be expressed as follows:

$$b = \sum_{i=0}^{a-1} b_i, \quad \text{and} \quad a \times b \geq n - 1.$$

Applying the above relationship, the complexity of the FRSSST becomes

$$T_{\text{FRSSST}} = O(n).$$

Comparing with the complexity of Morris *et al.*'s RSST algorithm $O(n^2)$, a new lower bound for RSST algorithm is found, which is $O(n)$ for our fast algorithm.

V. EXPERIMENTAL RESULTS AND CONCLUSIONS

A number of standard images have been used to examine the performance of our approach, the FRSSST, and the RSST algorithm. Let us use the House and Lenna examples for our discussion. By comparing their execution times, it is shown that approximately 30% speedup has been achieved by our proposed algorithm. The execution time for Morris *et al.*'s algorithm and our fast algorithm with both House and Lenna images, ranging from 55×55 pixels to 256×256 pixels are shown on Fig. 3. Their segmented images and edge maps of both the House and Lenna images are shown on Figs. 4 and 5, respectively. The resulting images for the two algorithms are different, since their growing strategies relating to their link weight functions are different.

We have presented a fast RSST algorithm for which the running time is $O(n)$ in the worst case. This is achieved by removing the sorting algorithm from the conventional algorithm. The running time of the proposed algorithm is a new lower bound for algorithms of this type. The algorithm is a region-based design, which is beneficial to images with many objects and regions in which pixel intensities are very close to the neighbors. Such regions and objects can possibly be represented by a single subtree, thereby greater reduction in processing time can be obtained. It is indeed the usual case for typical head-and-shoulder images.

Although the FRSSST holds the same order of space $O(n)$ as Morris *et al.*'s algorithm, the coefficient of the highest order for the FRSSST is 20% smaller than that for Morris *et al.*'s algorithm. This figure is even more significant for images with larger sizes. Furthermore, the region-based image mapping can achieve further speedup on the overall performance of the algorithm.

ACKNOWLEDGMENT

The authors are grateful to Prof. W. C. Siu of Hong Kong Polytechnic University for his comments and careful reading of the draft of this paper.

REFERENCES

- [1] O. J. Morris, M. de J. Lee, and A. G. Constantinides, "Graph theory for image analysis: An approach based on the shortest spanning tree," in *IEE Proc. F., Communications, Radar & Signal Processing*, 1986, vol. 133, pp. 146–152.
- [2] M. J. Biggar, O. J. Morris, and A. G. Constantinides, "Segmented-Image coding: A performance comparison with discrete cosine transform," *IEE Proc.* vol. 135, pt. F, pp. 121–132, Apr. 1988.
- [3] O. J. Morris and A. G. Constantinides, "Thin link coding techniques," *Int. Conf. Digital Signal Processing*, Florence, Italy, 1987.
- [4] —, "Segmented video coding," in *Proc. ICASSP 88*, vol. 1–5, pp. 1108–1111.
- [5] —, "Progressive image coding from a spanning tree image representation," in *Proc. 11th Colloque*, 1987.
- [6] A. A. Alatan, "A technical report on very low bit-rate region segmentation based videophone systems," Imperial College, London, U.K., Tech. Rep., Sept. 1992.
- [7] S. B. Chae, J. S. Kim, and R. H. Park, "Video coding by segmenting motion vectors and frame differences," *Opt. Eng.*, vol. 32, pp. 870–876, Apr. 1993.
- [8] M. Soryani and R. J. Clarke, "Segmented coding of digital image sequences," in *Proc. Inst. Elect. Eng. I*, vol. 139, pp. 212–218, Apr. 1992.
- [9] M. Gilge, T. Engelhardt, and R. Mehlan, "Coding of arbitrarily image segments based on a generalized orthogonal transform," *Signal Processing: Image Commun.* 1, pp. 153–180, 1989.
- [10] P. Cicconi and H. Nicolas, "Efficient region-based motion estimation and symmetry oriented segmentation for image sequence coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 357–364, June 1994.
- [11] T. Vlachos and A. G. Constantinides, "A graph-theoretic approach to color image segmentation and contour classification," in *Proc. 4th. Int. Conf. Image Processing and Its Applications*, Maastricht, The Netherlands, Apr. 1992.

Color Image Retrieval Based on Hidden Markov Models

Hsin-Chih Lin, Ling-Ling Wang, and Shi-Nine Yang

Abstract—In this correspondence, a new approach to retrieving images from a color image database is proposed. Each image in the database is represented by a two-dimensional pseudo-hidden Markov model (2-D PHMM), which characterizes the chromatic and spatial information about the image. In addition, a flexible pictorial querying method is used, by which users can paint the rough content of the desired images in a query picture. Image matching is achieved by comparing the query picture with each 2-D PHMM in the database. Experimental results show that the proposed approach is indeed effective.

I. INTRODUCTION

Traditionally, we access an image database based on textual information. With the growth in volume and diversity of image databases, the access method encounters some problems: i) Uniform textual descriptions may be insufficient to characterize various types of images and their internal properties; and ii) Cognitive gaps between

Manuscript received April 15, 1995; revised May 17, 1996. This work was supported in part by the National Science Council, Taiwan, R.O.C., under Grant NSC85-2213-E-007-047. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. John Goutsias.

The authors are with the Department of Computer Science, National Tsing Hua University, Hsinchu 30043, Taiwan, R.O.C. (e-mail: ling@cs.nthu.edu.tw).

Publisher Item Identifier S 1057-7149(97)00339-4.