IEICE TRANS. INF. & SYST., VOL.E97–D, NO.12 DECEMBER 2014

LETTER

An Efficient Two-Scan Labeling Algorithm for Binary Hexagonal Images

Lifeng HE[†,††a)], ***Member***, **Xiao ZHAO**[†], **Bin YAO**[†], **Yun YANG**[†b)], ***and*** **Yuyan CHAO**[†††], ***Nonmembers***

SUMMARY This paper proposes an efficient two-scan labeling algorithm for binary hexagonal images. Unlike conventional labeling algorithms, which process pixels one by one in the first scan, our algorithm processes pixels two by two. We show that using our algorithm, we can check a smaller number of pixels. Experimental results demonstrated that our method is more efficient than the algorithm extended straightly from the corresponding labeling algorithm for rectangle binary images.

key words: *hexagonal image, labeling, connected component, computer vision, pattern recognition*

1. Introduction

Images are usually sampled on a rectangular lattice, and rectangular-pixel images are almost always used in image processing. However, having several important advantages, e.g., consistent connectivity, equidistance with all neighbors and resemblance with the arrangement of photoreceptors in the human eyes [1], images sampled on a hexagon lattice, i.e., hexagonal images, have attracted much attention [2]–[9].

Labeling connected components in a binary image is one of fundamental operations in image analysis, pattern recognition, computer (robot) vision, and machine intelligence. By use of the labeling operation, a binary image is transformed into a symbolic image in which all pixels belonging to a connected component are assigned a unique label. Labeling is required whenever a computer or a system needs to recognize independent objects (connected components) in binary images as separate objects. In other words, labeling is indispensable in almost all image-based applications [10], [11]. Many algorithms have been proposed for labeling connected components in a binary rectangular-pixel image [12]–[16].

As mentioned above, hexagonal images have been actively studied recently. However, no labeling algorithm for hexagonal images has been reported. In principle, labeling algorithms for binary rectangular images can be straightly extended to label binary hexagonal images. For example, the labeling algorithm proposed by He, Chao and Suzuki [17], called *the HCS algorithm*, which is a very efficient algorithm for labeling binary rectangle images, can be extended to label binary hexagonal images by processing pixels one by one. For convenience, we denote this algorithm as *SF algorithm*. However, different from the case for labeling binary rectangle images, where the mask for every foreground pixel is the same, there are two different masks for foreground pixels in different positions in binary hexagonal images. We show that, instead of processing pixels one by one, processing pixels two by two will be more efficient. Experimental results demonstrated that our proposed method is more efficient than the SF algorithm.

2. Preliminaries

For an $N \times M$-size binary image, we use $b(x, y)$ to denote the pixel as well as its value at (x, y) in the image. As in most image processing algorithms, we assume that the object (foreground) pixels and background pixels in a given binary image are represented by 1 and 0, respectively, and all pixels on the border of an image are background pixels.

The HCS algorithm is an equivalent-label-based two-scan labeling algorithm. In this algorithm, at any processing point in the first scan, all equivalent labels* are combined in an *equivalent label set*, and the smallest label in the set is called the *representative label* of the set as well as all provisional labels in the set. For convenience, an equivalent label set with m as its representative label is represented as $S(m)$, and the representative label of l is denoted as $r[l]$.

In the first scan, if the current pixel $b(x, y)$ is a background pixel, nothing needs to be done. Otherwise, i.e., the current pixel $b(x, y)$ is a foreground pixel, we process $b(x, y)$ in different ways according to the configuration of the mask, which consists of all of $b(x, y)$'s processed neighbor pixels (Fig. 1):

(1) If there is no foreground pixel in the mask, i.e., no label in the mask, $b(x, y)$ does not connect to any processed foreground pixel up to now, it belongs to a new connected component in the processed area. We assign it a new provisional label p by $b(x, y) \leftarrow p$, and establish an equivalent

Manuscript received June 10, 2014.
Manuscript revised August 8, 2014.
Manuscript publicized August 27, 2014.

†The authors are with Artificial Intelligence Institute, College of Electrical and Information Engineering, Shaanxi University of Science and Technology, Xi'an, Shaanxi 710021, China.
††The author is with the Graduate School of Information Science and Technology, Aichi Prefectural University, Nagakute-shi, 480–1198 Japan.
†††The author is with the Graduate School of Environment Management, Nagoya Sangyo University, Owariasahi-shi, 488–8711 Japan.
a) E-mail: helifeng@ist.aichi-pu.ac.jp
b) E-mail: yangyun11@163.com (Corresponding author)
DOI: 10.1587/transinf.2014EDL8119

*All provisional labels assigned to the same connected component are called equivalent labels.

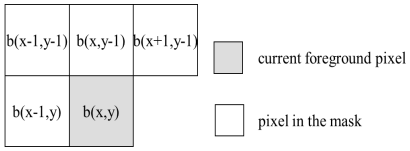

Fig. 1 The mask for the current foreground pixel in binary rectangular images.

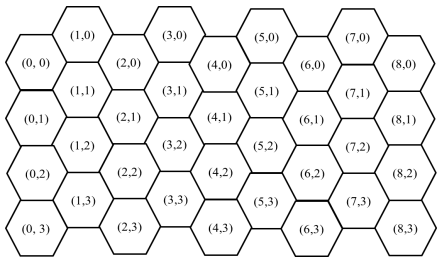

Fig. 2 The 2D array representation for hexagonal images.

label set for the new connected component by $S(p) = \{p\}$ and $r[p] = p$.

(2) There is only a foreground pixel block in the mask, thus, all foreground pixels in the mask belong to the same connected component, and $b(x, y)$ also belongs to the connected component. We assign $b(x, y)$ any label in the mask.

(3) There are two indispensable foreground pixel blocks in the mask, thus, the two blocks will connect by $b(x, y)$, we assign $b(x, y)$ any label in the mask. Let m and n be two provisional labels assigned to the two blocks respectively, then all provisional labels in $S(u)$ and $S(v)$ are equivalent labels, where $u = r[m]$ and $v = r[n]$. Thus, we need to combine $S(u)$ and $S(v)$. The pseudo codes for doing this work, denoted to $resove(u, v)$, can be given as follows.

if $u<v$
| $S(u)\leftarrow S(u)\cup S(v)$;
| $(\forall t\in S(v))(r[t] \leftarrow u)$;
else if $v < u$
| $S(v)\leftarrow S(u)\cup S(v)$;
| $(\forall t\in S(u))(r[t] \leftarrow v)$;
end of if

By the above strategy, as soon as the first scan is finished, all equivalent labels of each connected component will have been combined in an equivalent label set with a unique representative label. In the second scan, by replacement of each provisional label with its representative label, all foreground pixels of each connected component will be assigned a unique label.

3. The SF Algorithm

Suppose that a binary hexagonal image $b(x, y)$ is stored in a 2D array as shown in Fig. 2.

Unlike the case in rectangular images, where the mask, as shown in Fig. 1, is the same for all foreground pixels, there are two different masks for different type foreground pixels in binary hexagonal images. The mask for foreground pixels $b(2u + 1, v)$ consists of all of its four processed neighbor pixels, i.e., $b(2u, v)$, $b(2u, v - 1)$, $b(2u + 1, v - 1)$, and $b(2u + 2, v - 1)$ (Fig. 3 (a)), while the mask for foreground pixels $b(2u + 2, v)$, consists of all of its two processed neighbor pixels, i.e., $b(2u + 1, v)$ and $b(2u + 2, v - 1)$ (Fig. 3 (b)), respectively. Thus, For a foreground pixel $b(2u + 1, v)$, there are 16 patterns in the mask, as shown in Fig. 4, and for a foreground pixel $b(2u + 2, v)$, there are four pattern in the mask, as shown in Fig. 5.

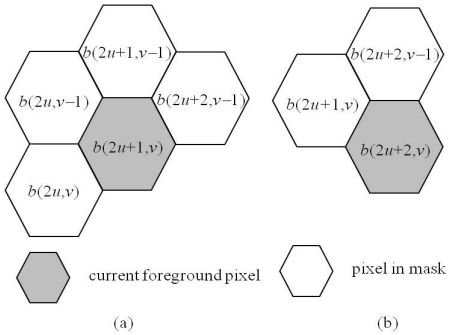

Fig. 3 The masks: (a) for the foreground pixel $b(2u + 1, v)$; (b) for the foreground pixel $b(2u + 2, v)$, in binary hexagonal image.

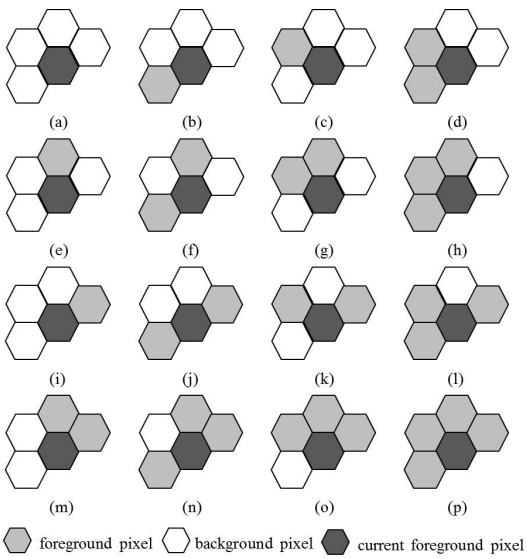

Fig. 4 16 patterns in the mask for a foreground pixel $b(2u + 1, v)$.

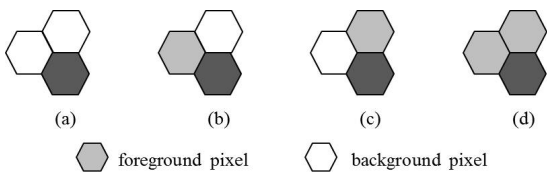

Fig. 5 Patterns in the mask for a foreground pixel $b(2u + 2, v)$.

Similar to the HCS algorithm, the SF algorithm labels a binary hexagonal image as follows: for each current pixel in the first scan, if it is a background pixel, nothing needs

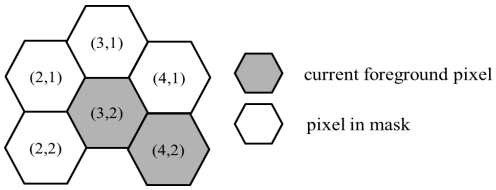

Fig. 6 An example.

to be done; otherwise, i.e., if it is a foreground pixel, we can assign to it any label in the mask if any, else a new label. Moreover, if there is two indispensable foreground pixel blocks in the mask (Fig. (f), (j), (k), (l), and (n)), they will be connected by the current foreground pixel, all provisional labels assigned to them are equivalent labels, and thus, should be combined together.

As the same in the HCS algorithm, after the first scan, all provisional labels assigned to the same connected component will be combined in an equivalent label set with the same representative label. Then, by the second scan, we can complete labeling by replacing each provisional label with its representative label.

4. Our Proposed Labeling Algorithm for Binary Hexagonal Images

By the SF algorithm, for processing a foreground pixel of the type of $b(2u+1, v)$, we need to check 4 pixels in the mask in maximum, and for a foreground pixel of the type of $b(2u+2, v)$, we need to check 2 pixels in the mask in maximum. Notice that, for example, as shown in Fig. 6, suppose that both of pixel $b(3,2)$ and pixel $b(4,2)$ are foreground pixels, then, they have the common processed neighbor pixel $b(4,1)$. Moreover, when processing pixel $b(4,2)$, pixel $b(3,2)$ is one of its processed neighbor pixels. If we process pixel $b(3,2)$ and pixel $b(4,2)$ independently, pixel $b(4,1)$ might be checked repeatedly, and pixel $b(3,2)$ will also be checked again when processing pixel $b(4,2)$.

To resolve this problem, we can process the two pixels together, although for processing pixel $b(3,2)$, we need to do as much as in the SF algorithm, however, for processing $b(4,2)$, we only need to assign the label of pixel $b(3,2)$ to pixel $b(4,2)$ without checking any pixel. Moreover, in the case where pixel $b(3,2)$ is background and pixel $b(4,2)$ is a foreground pixel, we only need to check the processed neighbor pixel $b(4,1)$. Thus, we can complete the process by checking less pixels, this leads a more efficient processing. The pseudo codes of our algorithm for processing the yth row ($1 \le y \le M-2$) can be shown as follows, where for convenience, we assume that the number of pixels in a row is even.

```
x←1;
while x ≤ N−2
| if b(x, y) =1
| | if b(x−1, y−1)=1   //Fig.4 (c), (d), (g), (h), (k),(l), (o) or (p)
| | | b(x, y) ← b(x−1, y−1);
| | | if b(x, y−1)=0 and b(x+1, y−1)=1   //Fig.4 (k) or (l)
| | | | resolve(r[b(x−1, y−1)], r[b(x+1, y−1)]);
| | | end of if
| | else if b(x, y−1)=1   //Fig.4 (e), (f), (m), or (n)
| | | b(x, y) ← b(x, y−1);
| | | if b(x−1, y)=1   //Fig.4 (f) or (n)
| | | | resolve(r[b(x, y−1)], r[b(x−1, y)]);
| | | end of if
| | else if b(x+1, y−1)=1 //Fig.4 (i) or (j)
| | | b(x, y) ← b(x+1, y−1);
| | | if b(x−1, y)=1   //Fig.4 (j)
| | | | resolve(r[b(x, y−1)], r[b(x−1, y)]);
| | | end of if
| | else if b(x−1, y)=1   //Fig.4 (b)
| | | b(x, y) ← b(x−1, y);
| | else   //Fig.4 (a)
| | | assigning a new label to b(x, y);
| | end of if
| | x ← x+1;
| | if b(x, y) =1 //Fig. 5 (b) or (d)
| | | b(x, y) ← b(x−1, y);
| | end of if
| else
| | x ← x+1;
| | if b(x, y) =1 //Fig.5 (a) or (c)
| | | if b(x, y−1) =1 //Fig. 5 (c)
| | | | b(x, y) ← b(x, y−1);
| | | else //Fig. 5 (a)
| | | | assign a new label to b(x, y);
| | | end of if
| | end of if
| end of if
| x ← x+1;
end of while
```

5. Experimental Results

Nine 512×512-sized noise images with densities from 0.1 to 0.9 were used for test. Because connected components in noise images have complicated geometrical shapes and complex connectivity, serious comparison can be made on them. The speed-up of our algorithm on the SF algorithm versus the density of an image is shown in Table 1. We can find that our algorithm is faster than the SF algorithm for all of noise images.

Table 1 Experimental results on noise images.

Density	Ours	SF	Speed-up (%)
	t_1	t_2	$100\times(t_2-t_1)/t_2$
0.1	0.94	0.94	0
0.2	1.89	1.93	1.97
0.3	2.68	2.79	4.14
0.4	3.45	3.71	6.94
0.5	3.90	4.37	10.80
0.6	3.82	4.30	10.91
0.7	3.27	3.67	8.38
0.8	2.54	2.67	4.87
0.9	1.56	1.59	2.01

Other images used for testing were composed of four types: artificial images, natural images, texture images, and medical images. All of these images are originally rectangle images and are transformed into corresponding hexagonal images [†].

Artificial images contain specialized patterns (stair-

Table 2 Various execution times (*ms*) for various types of images.

Image Type		SF	Ours
Natural	Max.	1.91	1.77
	Mean	0.97	0.91
	Min.	0.43	0.37
Medical	Max.	1.08	0.98
	Mean	0.74	0.69
	Min.	0.59	0.54
Textural	Max.	1.64	1.53
	Mean	1.06	1.02
	Min.	0.55	0.49
Artificial	Max.	1.22	1.09
	Mean	0.56	0.51
	Min.	0.16	0.13

like, spiral-like, saw-tooth-like, checker-board-like, and honeycomb-like connected components) [16].

On the other hand, 50 natural images, including landscape, aerial, fingerprint, portrait, still-life, snapshot, and text images, obtained from the Standard Image Database (SIDBA) developed by the University of Tokyo [††] and the image database of the University of Southern California [†††], were used for realistic testing of labeling algorithms. In addition, seven texture images, which were downloaded from the Columbia-Utrecht Reflectance and Texture Database [††††], and 25 medical images obtained from a medical image database of The University of Chicago, were used for testing. All of these images were 512×512 pixels in size, and they were transformed into binary images by means of Otsu's threshold selection method [17].

6. Conclusion

Due to several important advantages, hexagonal images have attracted many attentions. This paper discussed the labeling problem on binary hexagonal images for the first time, and proposed an efficient labeling algorithm. The experimental results demonstrated that our algorithm is more efficient than the straightforward one. For future work, we will extend our method to labeling 3D hexagonal images [18].

Acknowledgments

We thank the anonymous referee for his/her valuable comments that improved this paper greatly. We are grateful to the associate editor Dr. Daisuke Deguchi for his kind cooperation. This work was supported in part by the National Natural Science Foundation of China under Grant No. 61471227, and the Grant-in-Aid for Scientific Research (C) of the Ministry of Education, Science, Sports and Culture of Japan under Grant No. 26330200.

[†]Because there is no dataset for hexagonal images.
[††]http://sampl.ece.ohio-state.edu/data/stills/sidba/index.htm
[†††]http://sipi.usc.edu/database/
[††††]http://www1.cs.columbia.edu/CAVE/software/curet/index.php

References

[1] A. Fayas, H. Nisar, and A. Sultan, "Study on hexagonal grid in image processing," The 4th International Conference on Digital Image Processing, pp.7–8, 2012.
[2] E.G. Rajan, T. Sanjay, and S.K. Pramod, "Hexagonal pixel grid modeling and processing of digital images using CLAP algorithms," International Conference on Systemics, Cybernetics and Informatics, pp.12–15, 2004.
[3] M. Lee and S. Jayanthi, "Hexagonal Image Processing – A Practical Approach," Springer-Verlag, 2005.
[4] L. Middleton and J. Sivaswamy, "Edge detection in a hexagonal-image processing framewor," Image and Vision Computing, vol.19, no.14, pp.1071–1081, 2001.
[5] S. Veni, K.A. Narayanankutty, and P. Vidya, "Performance analysis of edge detection methods on hexagonal sampling grid," International Journal of Electronics Engineering and Research (IJEER), Research India publications, vol.1, no.4, pp.313–328, 2009.
[6] L. Middleton and J. Sivaswamy, "Framework for practical hexagonal image processing," Journal of Electronic Imaging, vol.11, no.1, pp.104–114, 2002.
[7] P. Sheridan, T. Hintz, and D. Alexander, "Pseudo-invariant image transformations on a hexagonal lattice," Image and Vision Computing, vol.18, pp.907–917, 2000.
[8] D. Van De Ville, W. Philips, and I. Lemahieu, "Least-squares spline resampling to a hexagonal lattice," Signal Processing: Image Communication, vol.17, no.5, pp.393–408, May 2002.
[9] D. Van De Ville, T. Blu, M. Unser, W. Philips, I. Lemahieu, and R. Van DeWalle, "Hex-spline: A novel family for hexagonal lattices," IEEE Trans. Image Processing, vol.13, no.6, pp.758–772, June 2004.
[10] D.H. Ballard, Computer Vision, Prentice-Hall, Englewood, New Jesey, 1982.
[11] R.C. Gonzalez and R.E. Woods, Digital Image Processing, Addison Wesley, 1992.
[12] A. Rosenfeld, "Connectivity in digital pictures," Journal of ACM, vol.17, no.1, pp.146–160, Jan. 1970.
[13] A. Rosenfeld and J.L. Pfalts, "Sequential operations in digital picture processing," Journal of ACM, vol.13, no.4, pp.471–494, 1966.
[14] L. He, Y. Chao, and K. Suzuki, "A run-based two-scan labeling algorithm," IEEE Trans. Image Process., vol.17, no.5, pp.749–756, 2008.
[15] F. Chang, C.J. Chen, and C.J. Lu, "A linear-time component-labeling algorithm using contour tracing technique," Computer Vision and Image Understanding, vol.93, pp.206–220, 2004.
[16] L. He, Y. Chao, and K. Suzuki, K. Wu, "Fast connected-component labeling," Pattern Recognition, vol.42, pp.1977–1987, 2009.
[17] N. Otsu, "A threshold selection method from gray-level histograms," IEEE Trans. Syst. Man Cybern., vol.SMC-9, pp.62–66, Jan. 1979.
[18] L. He, Y. Chao, and K. Suzuki, "Two efficient label-equivalence-based connected-component labeling algorithms for 3D binary images," IEEE Trans. Image Process., vol.20, no.8, pp.2122–2134, 2011.