The following manuscript was published in

Hidden Deletable Pixel Detection Using Vector Analysis in Parallel Thinning to Obtain Bias-Reduced Skeletons

Yung-Sheng Chen

Department of Electrical Engineering, Yuan-Ze Institute of Technology
135 Yuan-Tung Road, Nei-Li, Taoyuan, Taiwan 320, ROC
FAX: +886-3-4639355
E-mail: eeyschen@ee.yzit.edu.tw
ABSTRACT

The improvement of producing skeletons that preserve the significant geometric features of patterns is of great importance. One of feasible approaches is to develop a method embedded in a known parallel algorithm to produce bias-reduced skeletons since a bias skeleton usually degrades the preservation of significant geometric features of patterns. From our observations, bias skeletons always appear in the junction of lines which form an angle less than or near equal to 90°. In this paper, the hidden deletable pixel (HDP) which influences the speed of deleting the boundary pixels on a concave side is newly defined. Based on the comparable performance of our pseudo 1-subcycle parallel thinning algorithm (CH), a reduced form of larger support (two L-pixel vectors, which is not the form of \( k \times k \) support) operated by an intermediate vector analysis about the deleted pixels in each thinning iteration is developed for HDP detection to obtain bias-reduced skeletons, which can be purchased by a reasonable computation cost. HDP restoration and parallel implementation are further considered to formulate an improved algorithm (CYS), where the connectivity preservation is guaranteed by the use of CH’s operators and HDP restoration. A set of synthetic images are used to quantify the skeleton from the geometry viewpoint and investigate the skeleton variations of using different (L)s. Based on the analyzing results, \( 3 \leq L \leq 9 \) are suggested for the current algorithm of CYS. CYS is evaluated in comparison with two small support algorithms (AFP3 and CH) and one larger support algorithm (VRCT) using the same patterns. Performance are reported by number of iterations (NI), CPU time (TC), and number of unmatched pixels (\( N_{unmatch} \) for bias-reduced measure). Results show that on the measure of TC, CYS is approximately 2 to 3 times slower than the others, while on the measure of NI, the four algorithms have approximately identical performance. And on the measure of \( N_{unmatch} \), CYS is approximately 2 to 3 times less than the others. One-pixel boundary noise is also considered for exploring the noise immunity. The results suggest that the noise immunity of CYS and CH is identical and is better than that of AFP3. As a result, the better bias-reduced skeletons produced by CYS may be purchased by a reasonable computation cost.
1 INTRODUCTION

Thinning has been an important technique in image processing for many pattern recognition systems, e.g. OCR (optical character recognition), schematic diagram interpretation, document analysis and recognition, etc. For example, several practical OCR systems have been in use from the earliest years of this technique to the immediate present [1-4]. Since 1966 [5], many thinning algorithms or the modified versions have been proposed. A comprehensive survey of thinning algorithms can be found in [6]. Thinning algorithms may be classified into two types: sequential [7-9] and parallel [9-27] algorithms. Sequential thinning operates on one pixel at a time and the operation depends on previously processed results, while parallel thinning tends to operate on all the pixels simultaneously. There has been a growing interest in parallel thinning algorithms in anticipation of a greater availability of parallel image processing systems [28-30]. A fully parallel thinning algorithm applies the same thinning operator over the entire image for each iteration. Operators with $3 \times 3$ support are often used in parallel thinning; however, the fully parallel application of a thinning operator based on $3 \times 3$ support cannot provide adequate thinning\footnote{Fully parallel $3 \times 3$ support reducing operators can preserve topology but the combination of geometric requirements and topology preservation cannot be done with $3 \times 3$ support for thinning.} [25]. Thus in the past, parallel thinning algorithms are often designed in 4-subcycle [28] or in 2-subcycle [10-14] manners to preserve the connectivity of an image. To improve the efficiency of parallel thinning, some investigators have worked around the topic to reduce the parallel computation time in number of iterations [13, 15-22]. An optimally small support for fully (also called 1-subcycle or one-pass) parallel thinning has been shown to require 11 pixels [25]. Certain issues regarding topological property (e.g. 4-curve and 8-curve), geometric results (e.g. L-shape, medial curve thinness and representation), and noise immunity when using 1-subcycle parallel thinning can be found in [13, 18, 20, 22, 24]. The use of thinning templates and connectivity-preserving functions has also been discussed in [13, 31]. As a note, the 1-subcycle parallel thinning algorithm is of considerable attention up to date.

In addition, in order to examine the effects of different parallel thinning algorithms on an OCR system, the performance of 10 parallel thinning algorithms from this perspective by gathering statistics from their performance on large sets of data were reported by Lam and
Suen [32]. They supported the conclusion of [20] that: There is very little further progress to
make in improving parallel computation time, since the operational speeds of these thinning
algorithms can be further increased through implementation by hardware, processing speed is
no longer a matter of serious concern (Of course, not only software but also hardware im-
plementation for a thinning algorithm, the actual computing time depends on the operation
complexity in the chosen platform), and attention should be focused on other improvements
such as producing skeletons that preserve significant geometric features of patterns [32]. Follow-
ed by this conclusion, the usual problem of bias skeleton (or skeleton shape distortion
[27]) which influences the preservation of significant geometric features of patterns is further
addressed in this paper. Our goal is to develop a method based on our original parallel
thinning algorithm [13] to reduce this problem (i.e. to obtain a bias-reduced skeleton) with
a reasonable computation cost.

Consider skeletons obtained by 1-subcycle parallel thinning algorithms with small support
operators. It is known that a highly biased skeleton is produced by Chin et al.’s algorithm
[15]. Such a highly bias skeleton has been noted by several investigators, e.g. Chen and
Hsu [13], Guo and Hall [20], whose thinning algorithms are denoted by CH2 and AFP33,
respectively. Figures 1(a) and 1(b) show the thinning results obtained by AFP3 and CH,
respectively. From these results, the bias skeletons are also presented noticeably in the
angular shape regardless of the branch yielded after thinning. According to the opinion of Li
and Basu [27] on bias skeleton, it is due to the lack of global information (or larger support).
They developed a so-called variable-resolution character thinning algorithm (denoted by
VRCT), which is parallel and uses a 9×9 window with a high resolution 3×3 center and low
resolution periphery, to reduce the skeleton shape distortion. However, by our investiga-
tion on this method (see Section 4.1), the current algorithm of VRCT still produces a noticeably
bias skeleton on our test patterns shown in Fig. 1(c). It also seems that other larger support
algorithms, such as k × k thinnings [9, 26], can help to solve this problem. However, based
on their results, their algorithms produce “coarseness” (or noisy) skeletons for larger k,

2 The pseudo 1-subcycle parallel thinning algorithm is used in this paper. It will be briefly reviewed in
Section 2.

3 This algorithm included here is one of a closely related family of three such algorithms, and it was
selected because it was considered superior by its authors.
and the acceptable cases are $k = 3$ or 4. This returns to the small support algorithm. Since the computational complexity (the number of comparisons for template matching) may increase exponentially with increasing template size, and the number of possible templates may increase exponentially, the larger support up to $9 \times 9$ template algorithm seems to be impractical and therefore has never been used so far [27].

Furthermore, a desirable skeleton in an 8-4 image [33] is perfectly 8-connected\(^4\) [32,34,35]. It has shown that skeletons produced by (CH) are perfectly 8-connected regardless of T-junctions (see pattern “Z” in Fig. 1(b) against that in Fig. 1(a)). That is, based on the designed local connecting function in [13] the 8-deletable point does not appear in thin curve but its appearance in a T-junction is allowed. This is due to the original consideration that the identification of a junction among thin curves (it usually is a 3-fork point in skeleton) is always easy regardless of the 8-deletable point in T-junction. In addition, in accordance with Lam and Suen’s report [32], only algorithms based on the 8-connectivity number [12, 28] have consistently produced perfectly 8-connected skeletons. It has been proved that the local connecting function used in designing CH is equivalent to 8-connectivity number [23]. Thus in the evaluation [32], except for algorithms in [12, 28], CH has the least number of “extra pixels” which is used to measure the deviation of skeletons from unit width. Where the “extra pixels” yielded in CH are always located on T-junctions.

In this paper, based on the comparable performance of CH, another reduced form of larger support (which is not the form of $k \times k$ support) operated by an intermediate vector analysis about the deleted pixels in each thinning iteration is developed on CH to obtain a bias-reduced skeleton, which is purchased by a reasonable computation cost. The purpose of vector analysis is mainly to find hidden deletable pixels (HDPs) which influence the speed of deleting the boundary pixels on the concave side [15]. In addition, HDP restoration and parallel implementation are further considered to formulate an improved algorithm, where the connectivity preservation is guaranteed by the use of CH’s operators and HDP restoration.

Section 2 briefly reviews the pseudo 1-subcycle parallel thinning algorithm (CH). Sec-

\(^4\)According to the strict definition on 8-curve [34], we call it the perfect 8-(connected) curve if it has no 8-deletable points; otherwise we call it imperfect.
tion 3 gives the analysis of bias skeletons in thinning, defines the HDP which is a major factor to yield bias skeletons, describes the HDP detection by vector analysis, and formulates the improved parallel thinning algorithm. Section 4 gives the further analyses and experimental results, and conclusions are drawn in Section 5.

2 PSEUDO 1-SUBCYCLE PARALLEL THINNING ALGORITHM

A binary pattern matrix consists of only two-level pixels: one is the pattern pixel (denoted by “1”) and the other is the background pixel (denoted by “0”). To facilitate the following expressions, 13-pixel support is defined in Fig. 2.

Thinning may be defined as the successive deletions of the outermost layers of a pattern until only a connected “skeleton” or “medial curve” of unit width remains. Deletion is defined as transforming a “1” pixel to “0” pixel. In accordance with our systematic approach [13], we found that three sets of thinning templates, defined respectively in Figs. 3(a), 3(b) and 3(c), are effective to design a parallel thinning algorithm. Note that the thinning templates in Fig. 3(c) are those in Fig. 3(b) rotated by 180°. Three logical functions $T_c(p)$, $T_1(p)$ and $T_2(p)$ are defined as the following:

$$T_c(p) = \begin{cases} 0 & \text{if the 8-neighbor of } p \text{ matches any template in Fig. 3(a),} \\ 1 & \text{otherwise,} \end{cases}$$

$$T_1(p) = \begin{cases} 0 & \text{if the 8-neighbor of } p \text{ matches any template in Fig. 3(b),} \\ 1 & \text{otherwise,} \end{cases}$$

and

$$T_2(p) = \begin{cases} 0 & \text{if the 8-neighbor of } p \text{ matches any template in Fig. 3(c),} \\ 1 & \text{otherwise.} \end{cases}$$

In practice, these logical functions may be implemented in a table-lookup manner if the thinning templates are predesigned and saved in a table [11].

Consider a 1-subcycle parallel thinning algorithm, parallel deletion of all “1” pixels satisfying $T_c(p) \land T_1(p) \land T_2(p) = 0$ may destroy the pattern connectivity in $p$’s 8-neighborhood, since $p$ and its “1” neighbors may be deleted together; thus additional conditions are required
[13]. A so-called on-line extended local connecting function (ELC-function) was designed to check the connectivity in parallel processing. The ELC-function, or $E(p)$, can be simply defined as the following:

$$E(p) \equiv (\bar{p}_{11} \land p_6 \land \bar{p}_2) \lor (\bar{p}_{10} \land p_4 \land \bar{p}_0) \lor (\bar{p}_9 \land p_2 \land \bar{p}_6) \lor (\bar{p}_8 \land p_0 \land \bar{p}_4).$$ (4)

If $E(p) = 1$, any “1” pixel $p$ satisfying $T_c(p) \land T_1(p) \land T_2(p) = 0$ should be preserved (not deleted). The four ORed terms of $E(p)$, sometimes called restoring templates [15], are illustrated in Fig. 4. In AFP3 algorithm [20], only two restoring templates given in Fig. 4(a) and 4(b) are used, the considered neighborhood is thus an optimally small 11-pixel support [15, 20, 25].

Based on the previous definitions, the 1-subcycle parallel thinning algorithm can be formally stated below:

**Algorithm 1 1-subcycle parallel thinning algorithm.** Apply the following operators in parallel over the image space. A “1” pixel $p$ is deleted when the following condition is satisfied:

$$(T_c(p) \land T_1(p) \land T_2(p)) \lor E(p) = 0.$$ 

Thinning stops when no further deletions occur. Figure 5 shows the thinning result of pattern “Z” given in Fig. 1, by this 1-subcycle parallel thinning algorithm. It takes 11 iterations (In this paper, one iteration is defined that the image space is examined once regardless of which operator applied over). Note that thinning the pattern “Z” by AFP3 also takes 11 iterations, its result is shown in Fig. 1(a). Since the result of Fig. 5 is not a unit-width thin line, some modifications of Algorithm 1 by the embedded 2-subcycle scheme were made to overcome this drawback. Let $I$ be a flip – flop, its value is changed as “010101...” corresponding to the first iteration, second iteration, third iteration, and so on. When $E(p) = 1$ occurs, the following two cases are considered: (1) When $I = 0$, $p$ is deleted if $T_c(p) \land T_1(p) = 0$, otherwise not. (2) When $I = 1$, $p$ is deleted if $T_c(p) \land T_2(p) = 0$, otherwise not. Since the most iterations of thinning perform in 1-subcycle manner until $E(p) = 1$ occurs (see Fig. 5 for example), the modified algorithm is called pseudo 1-subcycle parallel thinning algorithm [13], or CH in this paper.
Algorithm 2 Pseudo 1-subcycle parallel thinning algorithm (CH). Apply the following operators in parallel over the image space. A “1” pixel $p$ is deleted when the following condition is satisfied:

$$
[(T_e(p) \land T_1(p) \land T_2(p)) \lor E(p)] \land [(T_e(p) \lor T_1(p)) \land (E(p) \land I)] \land [(T_e(p) \land T_2(p)) \lor (E(p) \land I)] = 0.
$$

Thinning stops when no further deletions occur. The thinning result of pattern “Z” obtained by CH is shown in Fig. 1(b), it also takes 11 iterations. Compared this result by CH to that by AFP3, the later still has one 8-deletable pixel on the thin line. The better aspect of CH is due to the consideration of the other two restoring templates shown in Fig. 4(c) and 4(d), the embedded 2-subcycle scheme, and the rotation of the operation. This is why the 13-pixel support defined in Fig. 2 is used in our algorithms.

3 THE PROPOSED METHOD

3.1 Analysis of Bias Skeletons in Thinning

Two reasons for generating bias skeletons have been mentioned in Chin et al.’s paper [15]. One is that the restoring templates are even size templates, and hence the pixel deletion process is not symmetrical. The other is due to the side effect of the thinning templates, that is, they delete pixels from convex corners faster than from concave corners. From Chin et al.’s observations [15], and our comparisons [31], these two reasons are really the major factors leading to the bias in a skeleton obtained by Chin et al.’s algorithm.

However, in a systematic design of thinning algorithm, the first reason can be ignored since the restoring templates are only used in maintaining connectivity of the “final” skeleton. If this term is further considered, a little improvement of more symmetrical skeletons may be achieved by using the symmetry of restoring templates shown in Fig. 4. Thus the bias-skeleton problem can be reduced to the side-effect of the thinning templates. In addition, we observed that bias skeletons always appear in the junction of lines which forms “an angle” less than or near equal to 90°. This phenomenon can be illustrated as follows.

Figures 6(a), 6(b), and 6(c) show the thinning iterations (denoted by numerals) of a 135° angle, a 90° angle, and a 45° angle pattern, respectively, by CH. Where Figs. 6(b) and 6(c)
show the skeletons (denoted by “@”s) are not medial near sharp corners. The iterations of deleting pixels show that the speed of deleting the convex side is faster than that of deleting the concave side. To further observe the deleting speed in thinning both sides, we focus on the first iteration of thinning the patterns shown in Figs. 6(d), 6(e), and 6(f), where “D” denotes a deleted pixel in the first iteration. To balance the deleting speed of both side, it is apparent that the circled pixels should be deleted in the first iteration. The circled pixel in Fig. 6(d) has been deleted since it matches a thinning template in Fig. 3(c), whereas the circled pixels in Figs. 6(e) and 6(f) are not deleted since they do not match any thinning template in Fig. 3. Under the consideration of connectivity preservation for a thin curve, the circled pixel can be deleted and referred to as hidden deletable pixel (or HDP) in the later of this paper. As a note, if the circled pixels in each iteration may be detected and deleted, then the bias skeleton may be reduced. In the following, we will now develop the approaches for HDP detection and restoration to enhance our original thinning algorithm CH such that the bias-reduced skeleton can be produced.

3.2 Hidden Deletable Pixel Detection by Vector Analysis

From the observation of bias skeletons, the circled pixels are located at the concave corner whose angle \( \angle(p) \leq 90^\circ \) consists of a “1” pixel \( p \) as well as two vectors \( \vec{V}_f \) and \( \vec{V}_s \) satisfying a straightness criterion [36]. Because of the use of square grids for representing a digital image, the angle representation may be illustrated in Fig. 7(a). Where “f”s and “s”s are the deleted pixels (i.e., “D”s shown in Figs. 6(d)-(f) for example) in the considered thinning iteration, and \( L \) is a pre-defined vector length (number of pixels) for both \( \vec{V}_f \) and \( \vec{V}_s \). Thus “D” pixels \( f_1, f_2, ..., f_L \) belong to vector \( \vec{V}_f \) and those \( s_1, s_2, ..., s_L \) belong to vector \( \vec{V}_s \), respectively. For tracing the vectors, direction code (originally termed chain code [37]) from a pixel to one of its neighbors is defined in Fig. 8. Three local measures are used: (1) \( D(p) \) is the number of “D” neighbors of \( p \); (2) \( DN(p) \) is the number of D1 or 1D patterns in the ordered set \( p_0, p_1, ..., p_7, p_0 \) as shown in Fig. 2; and (3) \( ON(p) \) is the number of 01, 0D, 10, or D0 patterns in the ordered set of (2). The “1” pixel \( p \) is called angle pixel of \( \angle(p) \) if \( D(p) = 2, \ DN(p) = 2, \) and \( ON(p) = 2 \). Only the angle pixel will be further analyzed.

The straightness determination of \( \vec{V}_f \) and \( \vec{V}_s \), and \( \angle(p) \leq 90^\circ \) are formulated as follows:
Let \( v \) be a general variable to represent the variable \( f \) or \( s \). Consider the “1” pixel \( p \) in Fig. 7(a), for \( \vec{V}_f \) the chain of direction code is obtained from \( f_1 \rightarrow f_2 \rightarrow \ldots \rightarrow f_L \rightarrow f_{L+1} \) and saved in an array \( CODE_f(i), i = 1, 2, \ldots, L \). And for \( \vec{V}_s \) the chain of direction code is obtained from \( s_1 \rightarrow s_2 \rightarrow \ldots \rightarrow s_L \rightarrow s_{L+1} \) and saved in an array \( CODE_s(i), i = 1, 2, \ldots, L \). The code change (CC) from \( CODE_v(i) \) to \( CODE_v(i + t) \) is represented by:

\[
CC_v(i, t) \equiv CODE_v(i) \odot CODE_v(i + t). \tag{5}
\]

Then the accumulator of code change (ACC) and the corner measurement (CM) of a vector are given by:

\[
ACC_v = \sum_{i=1}^{L} CC_v(i, 1), \text{ and} \tag{6}
\]

\[
CM_v = \begin{cases} 
1 & \text{if } \exists CC_v(i, t) \geq 2, i = 1, \ldots, L - t, \text{ where } t = 2, \ldots, \lfloor \frac{L}{2} \rfloor, \text{ and } L \geq 4, \\
0 & \text{otherwise.} 
\end{cases} \tag{7}
\]

A vector \( \vec{V}_v \) satisfies straightness if \( ACC_v \leq 2 \) and \( CM_v = 0 \). Where \( \lfloor x \rfloor \) denotes truncation to the smaller integer part of \( x \). Note that if \( L < 4 \), \( CM_v \) is set to 0.

The vector direction \( \phi_v \) of \( \vec{V}_v \) in the \( x - y \) coordinate system can be determined by unit variations of \( \delta x_v \) and \( \delta y_v \) calculated by:

\[
\delta x_v = \frac{\sum_{i=1}^{L}(x_{v_i} - x_{v_{i+1}})}{L}, \text{ and } \delta y_v = \frac{\sum_{i=1}^{L}(y_{v_i} - y_{v_{i+1}})}{L}, \tag{8}
\]

where \(-1 \leq \delta x_v, \delta y_v \leq 1\). Based on the \( \delta x_v \) and \( \delta y_v \), \( \phi_v \) of the considered vector starting from \((x_{v_1}, y_{v_1})\) is assigned to the nearest direction code. Therefore if \( \vec{V}_v \) satisfies straightness, the vector form of \( \vec{V}_v \) may be represented by the starting point \((x_{v_1}, y_{v_1})\) and the vector direction \( \phi_v \). That is, under satisfied straightness, \( \vec{V}_f \) is composed of \((x_{f_1}, y_{f_1})\) and \( \phi_f \); and \( \vec{V}_s \) is composed of \((x_{s_1}, y_{s_1})\) and \( \phi_s \), shown in Fig. 7(a) for example. Thus \( \angle(p) \leq 90^\circ \) is functionally equivalent to the code change from \( \phi_f \) to \( \phi_s \leq 2 \), and both \( \phi_f \) and \( \phi_s \) belong to the same quadrant in the \( x - y \) coordinate system.

Based on the mentioned characteristics of the \( \angle(p) \leq 90^\circ \) consisting of two vectors satisfying straightness, there exists an intersection point \((x_c, y_c)\) where the vectors, \( \vec{V}_f \) and \( \vec{V}_s \), intersect as illustrated in Fig. 7(a). We further consider the “1” pixels in the triangular region, \( \Delta(p) \), constructed by the points \((x_{f_1}, y_{f_1}), (x_{s_1}, y_{s_1})\), and \((x_c, y_c)\), since they belong to the pixels of both contour and intersection of an angle. Note that the triangular regions in
Fig. 7 are painted by grey. A “1” pixel belonging to $\Delta(p)$ is called **Hidden Deletable Pixel** or **HDP**. Now assume $\vec{V}_f$ and $\vec{V}_s$ in Figs. 7(b)-(d) satisfying straightness for example. Because the “1” pixel $p$ on the contour belonging to such a case (or those obtained from it by multiples of $90^\circ$ rotations) shown in Fig. 7(b) is usually deleted with a thinning template (refer to Fig. 3), it is impossible to be an HDP. In Fig. 7(c), the angle formed by $\vec{V}_f$ and $\vec{V}_s$ is $90^\circ$, the found intersection pixel $p$ is just the angle pixel is an HDP located at the triangular region. In Fig. 7(d), the angle formed by $\vec{V}_f$ and $\vec{V}_s$ is $45^\circ$, the found intersection pixel $q$ and the angle pixel $p$ are HDPs located at the triangular region. In Fig. 7(e), for example, the partial pixels in Fig. 6(f) are used to actually enumerate the HDPs considering $L = 3$. Here $p$ and $q$ are “1” pixels. Now consider $p$, $D(p) = 2$, $DN(p) = 2$, and $ON(p) = 2$, thus $p$ is an **angle pixel**.

For $\vec{V}_f$, the chain sequence is obtained from $(9,8) \rightarrow (10,7) \rightarrow (11,6) \rightarrow (12,5)$, where chain code is 111. For $\vec{V}_s$, the chain sequence is obtained from $(10,9) \rightarrow (11,9) \rightarrow (12,9) \rightarrow (13,9)$, where chain code is 000. Thus we have $ACC_f = 0$, $CM_f = 0$, $\delta x_f = -1$, $\delta y_f = 1 \Rightarrow \phi_f = 5$, and $ACC_s = 0$, $CM_s = 0$, $\delta x_s = -1$, $\delta y_s = 0 \Rightarrow \phi_s = 4$. The found angle $\angle(p)$ is $45^\circ$ since the code change from $\phi_f$ to $\phi_s = 1$. The intersection point by vectors $\vec{V}_f$ and $\vec{V}_s$ is $(8,9)$. The found triangular region (painted by grey), $\Delta(p)$, covers pixels $p$ and $q$, thus the two pixels are HDPs.

**HDP Detection.** After performing an iteration of thinning, the angle pixel $p$ is first detected to form $\angle(p)$. If $\angle(p) \leq 90^\circ$ consisting of two vectors satisfying straightness, then a $\Delta(p)$ is constructed. The “1” pixel belonging to $\Delta(p)$ is an HDP.

For more clearness of HDP detection, three points in Fig. 9(a) are used for illustration. Point (3,9) is not an angle pixel since its $D(p) = 3$ (here we regard a point as a pixel). Point (6,6) is an angle pixel since its $D(p) = 2$, $DN(p) = 2$, and $ON(p) = 2$. The chain code starting from point (5,6), a “D” pixel, is 555 (here $L = 3$) for measuring straightness of the first vector ($\vec{V}_f$). And the chain code starting from point (6,5), a “D” pixel, is 111 for measuring straightness of the second vector ($\vec{V}_s$). Both the vectors satisfy straightness since $ACC_f = 0$, $CM_f = 0$, $ACC_s = 0$, and $CM_s = 0$. However the code change from $\phi_f$ to $\phi_s = 4$ (where $\phi_f = 1$ and $\phi_s = 5$), this does not satisfy $\angle(p) \leq 90^\circ$ and point (6,6) is thus not an HDP. Similarly point (9,9) is an angle pixel. The chain code starting from point (9,8) is 111 for measuring straightness of the first vector ($\vec{V}_f$). And the chain code
starting from point (10,9) is 000 for measuring straightness of the second vector (\(\overrightarrow{V}_s\)). Both the vectors also satisfy straightness. The code change from \(\phi_f\) to \(\phi_s = 1\) (where \(\phi_f = 5\) and \(\phi_s = 4\)), this does satisfy \(\angle(p) \leq 90^\circ\). The intersection point by the two vectors is (8,9) and \(\Delta(p)\) is formed by points (9,8), (10,9), and (8,9). Since points (8,9) and (9,9) belong to \(\Delta(p)\), they are HDPs and denoted by “H”s. Figures 9(b) and 9(c) show the second iteration and the final result, respectively. Note that HDP detection strongly depends on the image resolution and the pattern boundary. This is because the vectors for HDP detection is determined by the deleted pixels (boundary pixels). If there exist strange unusual cases, such as irregular boundaries, the HDP detection may be ineffective. Fortunately, this problem may be overcome by an iterative smoothing of the original boundary before thinning [26]. The further detailed analysis of \(L\) will be given in Section 4.3.

3.3 HDP Restoration and Thinning Algorithm

An important criterion of deleting an HDP is that connectivity should be preserved. If deletion of an HDP destroys connectivity, then this HDP should be not deleted and be restored to “1” pixel.

**HDP Restoration.** The found HDP should be restored (that is, set “H” pixel return to “1” pixel) if the neighborhood of \(H\) satisfy any of the templates shown in Fig. 10.

Figure 11 shows the effect of HDP restoration. In Fig. 11(a), the connectivity is destroyed without using HDP restoration after the deletion of pixels labeled by “D” and “H”. If HDP restoration is involved, two HDPs are restored and the connectivity is preserved as the illustration shown in Fig. 11(b).

Following the previous introduction to HDP detection using vector analysis and HDP restoration, we now enhance our original thinning algorithm CH described in Section 2 by the following algorithm.

**Algorithm 3** Thinning Algorithm (CYS). For each iteration apply the following steps in parallel over the image space. They are iteratively performed until no “1” pixels are changed.
1. Execute the operators of CH on a “1” pixel and label the deletable pixel as “D”. Note here that in this step the “D” pixel is not changed to “0” since it is useful for the following HDP detection.

2. Find HDPs and label them as “H”s.

3. Do HDP restoration. If the deletion of an “H” pixel destroys the connectivity, then restore it to “1”.

4. Change all the pixels labeled by “D” and “H” to “0” (deletions performed here).

Where the connectivity preservation of CYS is guaranteed by the use of CH’s operators and HDP restoration.

The steps 1, 3 and 4 may be operated in parallel processing due to the use of 3 × 3 local window [28-30]. From the presentation of vector tracing for HDP detection, it seems that step 2 is operated in sequential manner. However if we could trace each detail in HDP detection, we would find its intrinsic characteristics of parallel processing: The direction codes collected in a local memory for each vector with the same L are decided by 3 × 3 window, the “D” values in image space are not modified, the computations of detecting HDPs can be implemented in a local processor, and the found “H”s can be labeled in the corresponding memory location of the other same-size image space. That is, if a suitable HDP detection processor (HDP processor) was designed, step 2 may also be operated in parallel manner. Based on the characteristics of vector tracing in HDP detection, a “D” pixel may be searched by two or more HDP processors. The collision in sharing communication lines (or accessing the same data) among HDP processors should be avoided when such a parallel processing system is implemented. As a note, the parallel implementation of CYS may be summarized as follows:

**Parallel Implementation of CYS.** Let the size of adopted matrices be n × n, matrices $MT_1$ and $MT_2$ be used for thinning operator of CH, matrix $MH$ be used for HDP detection, $n \times n$ thinning processors implement CH, and $n \times n$ HDP processors implement the functions of HDP detection and HDP restoration. The original binary image is stored in $MT_1$. 

12
1. Thinning processors simultaneously search $MT_1$ and generate the results (“D”s and other no changes) stored in $MT_2$.

2. HDP processors simultaneously search $MT_2$ for HDP detection and generate the results (“H”s and other no changes) stored in $MH$.

3. HDP processors simultaneously search $MH$ for HDP restoration and generate the results (deletions and other no changes) stored in $MT_1$.

4. Goto step 1 until no further deletions occur.

Note that these three parallel processing steps are done in sequence. In the following, we illustrate a possible model for such a parallel implementation. Consider a point $(x, y)$, the corresponding processing elements (PEs) are $PE_T(x, y)$ and $PE_{HDP}(x, y)$ for thinning processor and HDP processor, respectively. For each $PE_T(x, y)$, there are local memory arranged as Fig. 2 shows. This memory arrangement allows each $PE_T(x, y)$ to look at parts of the domain of its neighbors. The current pixel corresponds to $MT_1(x, y)$ in Step 1, and the result (“D” or no change) is written on $MT_2(x, y)$ (considered in Step 2). Similarly, for each $PE_{HDP}(x, y)$, there are two $(2L + 1) \times (2L + 1)$ local memories (One is for $MT_2$, the other is for $MH$). The use of these local memories is to fit any possible case when tracing a vector), as well as a general-purpose CPU with its own program and data memory for data computations. In Step 2, the current pixel corresponds to $MT_2(x, y)$ and based on it for HDP detection. If the point $(x, y)$ is identified by $PE_{HDP}(x, y)$ or other (PEHDP)s as “H” pixel, then “H” is written on the corresponding memory location in $MH(x, y)$. Otherwise, the original value in $MT_2(x, y)$ is directly written on $MH(x, y)$. In Step 3, the current pixel corresponding to $MH(x, y)$ as well as its neighbors in local memory are used for HDP restoration, and the result is written on $MT_1(x, y)$. This step is similar to Step 1. By such a design for $PE_T$ and $PE_{HDP}$, the problem of having two or more PEs trying to write on the same memory location is overcome, and the scheme of intercommunication among PEs thus becomes simple. Of course obviously, the trade-off is to increase a large amount of memory and interconnection lines, as well as the cost of copying one result to many memory locations in different PEs when each step is finished.
For the time complexity of parallel implementation of CYS, a rough evaluation may be stated as follows. Let $T_{\text{local}}$ be the processing time of a local operator in general case assuming some fair unit of computation is used. Then both thinning operator (in Step 1) and HDP restoration (in Step 3) takes $T_{\text{local}}$. The vector tracing in $L$ dependent HDP detection will take about $2L \times T_{\text{local}}$, since there are $2L$ local window are countered in vector tracing (for two $L$-pixel vectors), and deciding next move direction of the local window may be regarded as a local operator. However, to complete the HDP detection, the additional steps in Eqs. (5)-(8) and the decision of an HDP are needed. The additional processing time is denoted as $T_{\text{compute}}$ which is larger than $T_{\text{local}}$. Furthermore let the time of copying data from one memory to the other be $T_{\text{copy}}$. Thus the overall processing time in one iteration of the parallel version of CYS will be

$$T_{\text{CYS}} = 2T_{\text{local}} + 2L \times T_{\text{local}} + T_{\text{compute}} + 3 \times T_{\text{copy}}$$

$$= 2(L + 1)T_{\text{local}} + T_{\text{compute}} + 3T_{\text{copy}}. \quad (9)$$

Based on the defined time units, we have $T_{\text{CH}} = T_{\text{local}} + T_{\text{copy}}$ for the parallel version of CH. Compare $T_{\text{CYS}}$ to $T_{\text{CH}}$, the additional processing time is $(2L + 1)T_{\text{local}} + T_{\text{compute}} + 2T_{\text{copy}}$. Although the iteration number of CYS may be smaller than or equal to that of CH (See Section 4.4 for reference), the increasing time complexity is indeed very significant in parallel implementation. Accordingly, the parallel implementation of CYS is feasible but the cost of hardware is expensive and the increasing time complexity is significant even the parallel processors can operate very fast. Thus we suggest that CYS is implemented in sequential machine, e.g. Intel Pentium-based personal computer, in which the addition computational cost is reasonably accepted. The detail performance comparisons will be given in Section 4.4.

The same angle patterns shown in Fig. 6 are now used to demonstrate the behaviour of the new improved algorithm, CYS. The resulting skeletons are shown in Figs. 12(a)-(c). Compared with those in Figs. 6(a)-(c), we find that the bias skeleton has been reduced. The results of the first iteration are also shown in Figs. 12(d)-(f), where HDP is labeled by “H”. Since not only “D” but also “H” pixels are deleted in the same iteration, the deleting speed of both convex and concave side does be nearly the same. This is just the main contribution of CYS to produce bias-reduced skeletons. Similarly Figure 13 shows that the bias-reduced skeleton can be effectively obtained by CYS (with $L = 5$) using the same patterns shown.
in Fig. 1.

The effectiveness of HDP detection depends on the selected value of $L$. Compared to the illustration ($L = 3$) shown in Fig. 9, the HDP detection is ineffective when $L = 5$, whose results are shown in Fig. 14. Where HDPs detected in Fig. 9(a) are not shown in Fig. 14(a), for example. In Fig. 14(a), point (9,9) is also an angle pixel, the chain code starting from point (9,8) is 11133 for measuring straightness of the first vector ($\vec{V}_f$). And the chain code starting from point (10,9) is 00000 for measuring straightness of the second vector ($\vec{V}_s$). In this case, the vector $\vec{V}_f$ does not satisfy straightness since $CM_f = 1$ so that the HDP is not located. Similar comparisons of the second iteration are also given in between Fig. 9(b) and 14(b). The final result of Fig. 9(c) shows a bias-reduced skeleton, but that of Fig. 14(c) does not. That is, the effectiveness of HDP detection fails when $L = 5$ in the current example. As a note, the proper $L$ selected may produce a better bias-reduced skeleton, otherwise the result by CYS may be returned to that by CH since the HDP detection may be ineffective. This property will be further analyzed and discussed in next section.

4 ANALYSES AND EXPERIMENTS

In this section, the properties of other larger support algorithms [9, 26, 27] with respect to CYS and the problem of noise immunity are first discussed. Then, $L$ used in CYS is analyzed, and experimental results are given, by a set of synthetic patterns. Finally, performance are reported by number of iterations, CPU time, and bias-reduced measure for comparisons and discussion.

4.1 Other Larger Support Algorithms

The use of two $L$-pixel vectors is the main contribution of CYS for obtaining bias-reduced skeletons. It seems that this work of reducing the skeleton shape distortion may also be done by other larger support algorithms, e.g. $k \times k$ thinning [9, 26]. However, according to the opinion of Li and Basu [27], the computational complexity (the number of comparisons for template matching) may increase exponentially with increasing template size, and the number of possible templates may also increase exponentially. Thus the $9 \times 9$ template
algorithms seems to be impractical and has never been used so far. Although the use of larger $k$ can reduce the number of iterations due to $(k-2) \times (k-2)$ pixels deleted simultaneously, it can also be detrimental to processing speed [6] and this is often at the expense of “coarseness” results (i.e. noisy skeletons) for larger $k$ [6, 9]. In addition, based on the experimental results of [9, 26], it seems that the acceptable cases are $k = 3$ or 4. Accordingly such a larger support algorithm based on the current form is unsuitably applied to the concerned topic in this paper since it returns to the small support version.

Variable-resolution character thinning (VRCT) proposed by Li and Basu [27] is a feasible principle to reduce skeleton shape distortion. The principle of VRCT is that a “rough” look of a larger neighborhood of the contour pixel being examined is used to help preserve the shape of the output skeleton. The algorithm is simply described as follows [27]: All “1” pixels are first examined by a small support parallel thinning algorithm (In our experiment for comparison, CH is adopted.) If certain conditions (e.g. straight border) are satisfied, another set of $9 \times 9$ templates are used. Each of these templates is partitioned into 9 equal square-shaped parts. The center part is a regular $3 \times 3$ template. Each one of the eight peripheral parts also covers a $3 \times 3$ region, but at a reduced resolution. Two binary values, $\alpha$ and $\beta$, are generated from each peripheral region of 9 pixels. $\alpha = 1$ if there are > 5 “1” pixels in a periphery, $\alpha = 0$ otherwise. $\beta = 1$ if this periphery forms a straight line, $\beta = 0$ otherwise. When $\alpha = 0$ and $\beta = 1$ for a periphery, the larger template specifies that the pattern possibly has a long straight stroke. A contour pixel should not be deleted if its neighborhood matches any of these templates.

Figure 15(a) shows the thinning results of three patterns used in [27], where straight vertical strokes are preserved. However, a problem is raised: If the line-width of a pattern is larger than that given in Fig. 15(a), the $9 \times 9$ template used in VRCT should be not enough for the further test of the “certain conditions”. Figure 15(b) illustrates this drawback. Patterns “B”, “N”, and “Z” are thinned respectively to the 6th, 7th, and 6th iteration by VRCT. Obviously, the $\alpha$ and $\beta$ tests used in VRCT may fail in further iterations on some regions of “1” pixels. And the unbalanced speed of deleting the convex side and concave side is very apparent. Thus bias skeletons are produced as shown in Fig. 1(c). If we want to solve this problem based on the principle of VRCT, it needs $> 9 \times 9$ larger template and more
complete set of testing functions. This returns to the original challenge point, expensive computational complexity, of Li and Basu. In addition, the line-width of a pattern is usually unknown, the determination of larger template may be very difficult.

Based on the above discussions of \( k \times k \) thinning and \textbf{VRCT}, they can be categorized as a larger support algorithm. \( k \times k \) thinning is only feasible in smaller \( k \) (3 or 4) and is not suitable for reducing bias effect. The reduction of skeleton shape distortion by the current algorithm of \textbf{VRCT} is limited due to the adopted low resolution periphery and simple testing functions. Unlike these algorithms, the proposed algorithm \textbf{CYS} uses two \( L \)-pixel vectors instead of the larger support or template to determine the HDPs for reducing the skeleton shape distortion. Although the HDP detection is not implemented in table-lookup manner as the thinning implementation using small support, it takes only a reasonable time (which will be confirmed in Section 4.4) for searching \( 2L \) direction codes and computing the local functions defined in Section 3.2. Hence, in the following, the experiments and comparisons will mainly focus on \textbf{AFP3}, \textbf{CH}, and \textbf{CYS} to explore the characteristics of involving HDP detection. Some performance of \textbf{VRCT} will also be given properly for comparisons.

\subsection{4.2 Noise Immunity}

The noise immunity is sometimes discussed in the literatures [15, 24, 26]. However, the considered noise is often focused on the one-pixel boundary noise because of the use of small support. In Chin \textit{et al.}'s algorithm [15], they use a set of trimming templates at each iteration in parallel with the thinning operation to suppress the growth of spurious segments induced by boundary noise. In using the trimming templates, endpoints of objects are no longer preserved. Although a measure of boundary noise sensitivity was presented by Jain and Chin [24], the noise immunity does not be further improved. A conclusion of noise immunity made by Poty and Ubeda [26] may be generally stated as that: The up-to-date thinning operators cannot differentiate an end pixel of a skeleton from a noisy pixel of the boundary. This kind of problem cannot be avoided by using neighborhood at unit distance except with an iterative smoothing of the original boundary before thinning.

Even though such a claim of tradeoff between noise immunity and endpoint preservation was made, it is possible to obtain a better performance of noise immunity for a one-pixel
boundary noise presented if a more complete set of thinning templates is developed as CH [13, 31]. A simple comparison illustrating this opinion is given in Fig. 16. Because CYS is improved from CH for obtaining bias-reduced skeletons, the noise immunity of CH and CYS is identical and is better than that of AFP3.

We give a comment for noise immunity to close this section: Since in the daily life, not all the line patterns are well-defined. They may be noisy (not the one-pixel boundary noise) or specially designed. Such a noisy digital pattern is impossible to be processed by the present thinning algorithms if the skeleton information of the pattern is desired. Based on principles of human visual perception [38], we proposed an approach for thinning noisy digital patterns [39]. The feasibility of this approach is fundamentally achieved by means of (1) finding the effective circular range (or adaptive support) containing the maximum pixel information; (2) computing the symmetry score of the pixel distribution in the circular range and converting the symmetry information into a gray-scale image; and (3) performing a gray-scale thinning algorithm on the gray-scale image. Obviously, the computational complexity is expensive for this approach. But if a more precise skeleton with good noise immunity is desired to be explored in the near future, a further improvement involving this approach may be one of possible solutions since the “adaptive support” instead of “$k \times k$” [9, 26] and “variable-resolution” [27] support is operated based on maximum pixel information, symmetry information, or other useful information.

4.3 Analysis of $L$ and Experimental Results

Since the interesting goal of this study is to develop a method for obtaining the bias-reduced skeleton, it is necessary to quantify the skeleton from the geometry viewpoint and investigate the skeleton variations of using different ($L$)s. Since the geometric property is not easily quantifiable by means other than human vision, Lam and Suen used three shape matching methods to develop distance measures and applied them to the comparison between skeletons and references [40]. Because the reference was prepared by human subjects and the skeleton is approximated by a sequence of polygons (referred to as *fragments*) for shape matching, the details of skeleton variations might be lost. Therefore we develop a simple matching method to measure the skeleton variation as follows: For any thick pattern, we define that
there exists a “truly medial curve”. Because the “truly medial curve” for a thick pattern is always unknown, the matching degree between the thinned result and the “truly medial curve” is only approximately evaluated by human vision such as the bias skeleton mentioned in [15] and previously. To quantify the matching degree, it can be done by counting the number of unmatched pixels between in the resulting medial curve and in the original thin pattern. That is, an original thin pattern instead of the “truly medial curve” is first given, then thicken it to obtain a thick pattern. Apply a thinning algorithm to the thick pattern and obtain the thinned result. Then by matching the original one and the thinned one, the number of unmatched pixels can thus be adopted for the evaluation of biasing degree of skeletons.

Let set $F$ denote the considered original thin pattern consisting only “1” pixels, and $G$ denote the thinned pattern (consisting also only “1” pixels) obtained from thinning the thickened version of $F$ by a thinning algorithm. Let the distance $d_{fg}$ between two pixels $f \in F$ and $g \in G$ be chessboard distance. And the distance $d_{FG}$ between the pixel $f \in F$ and $G$ is defined as the minimum $d_{fg}$, $\forall g \in G$. For a pixel $f$, it is called a match case if $d_{FG} \leq 1$. Let $N_{total}$ and $N_{match}$ be total number of pixels $\in F$, and number of match cases, respectively. Thus the number of unmatched cases, $N_{unmatch}$, is obtained by $N_{total} - N_{match}$.

We apply the proposed matching method to analyze $L$ used in CYS and evaluate skeletons obtained by different algorithms. Two types of synthetic images are designed for experiments: symbolic patterns and angular patterns. The symbolic test sets include sets of 3 star symbols (star1, star2, star3), sets of 3 Chinese characters (sé, pi, yung), and sets of 3 numerals (4, 5, 7). The original thin patterns of the symbolic test sets are shown in Fig. 17. The angular test sets include 16 angle patterns varying from $15^\circ$ to $90^\circ$, denoted by angle_15 to angle_90, respectively. The original thin patterns of the angular test sets are shown in Fig. 18.

We use the symbolic test sets to explore the skeleton variations of using different ($L$s). The thickened version of an original thin pattern is obtained by the following procedure:

**Thickening Procedure.** For each original thin pattern, the other same-size image space is created. A circular disk (diameter = 19 pixels in the present case) consisting “1” pixels is applied over the created image space to form a 19-pixel thickness pattern.
Where for each “1” pixel belonging to the original thin pattern, let the coordinate of the “1” pixel be the center of the disk and set all pixels covered by the disk to “1”’s in the created image space. In other words, the new pattern is the union of same-size disks centered at points of the original thin curve.

Ideally, based on such a thickening procedure, the thinned result should fit the original thin pattern as close as possible if the algorithm provides a balance speed of deletions. The thickened patterns of the symbolic test sets are thinned by CYS using different \((L)\)s to explore the skeleton variations. The experiments are summarized in Fig. 19(a) for star symbols, Fig. 19(b) for Chinese characters, and Fig. 19(c) for numerals, plotting number of unmatched pixels against \(L\). The best selection of \(L\) is considered by two factors: (1) The corresponding measured number of unmatched pixels should be minimum; and (2) the smallest \(L\) is selected if there are many \((L)\)s having the same minimum number of unmatched pixels. The first is to obtain the best bias-reduced skeleton, while the second is to minimize the computational cost under the use of the proposed CYS algorithm. Based on this selection principle, \(L = 5\) for “star1”, \(L = 21\) for “star2”, and \(L = 7\) for “star3”, in Fig. 19(a); \(L = 3\) for “sê” and “yung” and \(L = 5\) for “pi”, in Fig. 19(b); and \(L = 3\) for the three numerals in Fig. 19(c). From the plots in Fig. 19, the other two facts may be observed: The first fact is that the number of unmatched pixels may increase when \(L\) increases, and it eventually reaches at a upper bound. This means that the large \(L\) is unnecessary and/or ineffective for our HDP detection\(^5\), and it may degrade the performance of CYS to that of CH for the concern of bias-reduced skeletons. This fact also provides the response for the illustration given in the end of Section 3.3 (Fig. 14). The upper bounds, i.e. the maximum number of unmatched pixels, of these plots except for “4” and “7” in Fig. 19(c) (Here their upper bounds reach at after \(L > 100\)) are exactly obtained by CH (see Table 2 discussed in next section for reference.) The second fact in the current example is \(L \geq 3\) for the proper operation of HDP detection. Based on these analyses and observations, \(3 \leq L \leq 9\) are suggested.

\(^{5}\)The main reason is that the vector directions, \(\phi_{f}\) and \(\phi_{s}\), with large \(L\) for an angle pixel \(p\) may differ with the effective ones such that the code change from \(\phi_{f}\) to \(\phi_{s} > 2\), or \(\angle(p) > 90^\circ\), and the detection of HDP pixels fails. Thus \(p\) is not an HDP in this case.
Figures 20-22 show the thinning results obtained by **AFP3**, **CH**, and **CYS**, respectively, for comparisons. To apparently display the thinning result, the thick pattern is only displayed by its contour. The image size of the symbolic test sets is 256 × 256 pixels. The 15-pixel thickness patterns thickened from angular test sets are also used for thinning comparisons. The thickening procedure is the same as that performed on the symbolic test sets. Figures 23-25 show the thinning results obtained by **AFP3**, **CH**, and **CYS** (with \( L = 5 \)), respectively. The image size of the angular test sets is 60 × 60 pixels. To further confirm the feasibility of **CYS** (with \( L = 5 \)), three handwritten Chinese characters “CHEN”, “YUNG”, and “SHENG” (256 × 256 pixels) were scanned, and their thinning results are shown in Fig. 26 for comparisons. All the test patterns were also thinned by **VRCT**, the corresponding performance will be reported in next subsection.

### 4.4 Comparisons and Discussion

The proposed thinning algorithm, **CYS**, is evaluated in comparison with two small support algorithms (**AFP3** and **CH**) and one larger support algorithm (**VRCT**). These algorithms were written in WATCOM C/C++\(^3\)_, run on the same Intel 90 MHz Pentium-based personal computer with MS-DOS (version 6.22), and tested with the same patterns. Number of iterations (including the final iteration computing the stopping condition that no more pixels can be deleted based on the corresponding algorithm), CPU time, and number of unmatched pixels are reported for comparisons and discussion. The measures for the same-size pattern sets are put into the same table for the convenience of statistics. Thus the measures for 256 × 256 pattern sets are reported in Tables 1 and 2; and those for 60 × 60 pattern sets are reported in Tables 3 and 4.

Tables 1 and 3 report the parallel computation time in number of iterations (\(NI\)) and CPU time (\(TC\), in msecs). Their averages are also given at bottom of these tables. Let \(R_{NI}^{\text{algorithm}}\) be the ratio of average (\(NI\))s between **CYS** and the given algorithm, and \(R_{TC}^{\text{algorithm}}\) be the ratio of average (\(TC\))s between **CYS** and the given algorithm, respectively. Then we have \(R_{NI}^{\text{AFP3}} = 0.94\), \(R_{NI}^{\text{CH}} = 0.94\), and \(R_{NI}^{\text{VRCT}} = 0.93\) for Table 1; and \(R_{NI}^{\text{AFP3}} = 0.97\), \(R_{NI}^{\text{CH}} = 0.95\), and \(R_{NI}^{\text{VRCT}} = 0.92\) for Table 3. And we have \(R_{TC}^{\text{AFP3}} = 2.27\), \(R_{TC}^{\text{CH}} = 2.69\), and \(R_{TC}^{\text{VRCT}} = 2.46\) for Table 1; and \(R_{TC}^{\text{AFP3}} = 1.85\),
$R_{TC}^{CH} = 3.19$, and $R_{TC}^{VRCT} = 2.63$ for Table 3. Based on these results, on the measure of $TC$, CYS is approximately 2 to 3 times slower than the others ($R_{TC}^{algorithm} \approx 1.85 - 3.19$), while on the measure of $NI$, the four algorithms have approximately identical performance ($R_{NI}^{algorithm} \approx 0.92 - 0.97$). The additional computation cost paid in CYS is mainly due to HDP detection.

Tables 2 and 4 report the number of unmatched pixels, $N_{unmatch}$, between the resulting medial curve and the original thin pattern. Their averages are also given at bottom of these tables. Because the test patterns, “CHEN”, “YUNG”, “SHENG”, “B”, “N”, and “Z”, are not synthetic images, there are no measures on these patterns. Let $R_{N_{unmatch}}^{algorithm}$ be the ratio of average ($N_{unmatch}$) between the given algorithm and CYS. Then we have $R_{N_{unmatch}}^{AFP3} = 3.13$, $R_{N_{unmatch}}^{CH} = 3.18$, and $R_{N_{unmatch}}^{VRCT} = 3.12$ for Table 2; and $R_{N_{unmatch}}^{AFP3} = 1.95$, $R_{N_{unmatch}}^{CH} = 2.07$, and $R_{N_{unmatch}}^{VRCT} = 1.95$ for Table 4. On the measure of $N_{unmatch}$, CYS is approximately 2 to 3 times less than the others ($R_{N_{unmatch}}^{algorithm} \approx 1.95 - 3.18$). The better bias-reduced skeleton produced by CYS is also due to the HDP detection.

In these reports, we found that the improvement of skeletons by the larger support algorithm VRCT is very little. This is because only a rough test for the $3 \times 3$ peripheral regions outside of the $3 \times 3$ neighborhood when certain conditions exist in central $3 \times 3$ region are embedded in a small support thinning algorithm. Thus it fails when a more thick pattern is processed by this algorithm as mentioned in Section 4.1. There is an apparent tradeoff between computation cost and bias-reduced skeleton among these algorithms. When low computation cost is a first priority, the small support algorithms (AFP3 or CH) may be considered. Otherwise, when the bias-reduced skeleton is strongly desirable, we recommend the proposed CYS to be used because the better bias-reduced skeleton may be purchased by a reasonable computation cost (refer to the previous results of $R_{TC}^{algorithm}$ and $R_{N_{unmatch}}^{algorithm}$).

Another comparison from the geometry viewpoint is also important. From the compared results (see Figs. 1 and 13 for “B”, “N”, “Z”; Figs. 20-22 and Figs. 23-25 for synthetic images; and Fig. 26 for scanned images), the best performance is achieved by CYS. That is, the bias skeleton obtained using the conventional algorithms has been reduced exactly by CYS. In Fig. 25, it shows a fact that the effectiveness of bias-reduced skeletons depends on not only the line thickness but also the angle. If the line pattern is so thick and the angle is so small,
then the shape of the “angular pattern” may disappear (see angle_15 and angle_20 in Fig. 25 compared with the corresponding original thin patterns in Fig. 18 for example) and thus result in an analogous line-segment pattern. Of course, the result of thinning such a pattern is also similar to a line-segment. However, if the angle is apparent enough even the line pattern is very thick, the angle shape of resulting thin line obtained by CYS is also apparent from the geometry viewpoint (see angle_25, angle_30, and angle_35 in Fig. 25 for example). Others (angle_40 - angle_90) show a good bias-reduced skeleton. Furthermore, observing the results in Fig. 22, we also found another fact that the length of branch appearing in the junction among lines is reduced by CYS. Thus we have a conclusion that as long as the angle is apparently reflected on the given line pattern regardless of its thickness, a better geometric approximation of bias-reduced skeleton can be achieved by the proposed method.

Thinning results shown in Fig. 26 also present a view of boundary “noise” points against skeleton. There are some one-pixel or two-pixel boundary “noise” points appearing on the contours of the scanned patterns. It seems that the skeletons are insensitive more or less to such a boundary point which arises in scanning. However, according to the discussion in Section 4.2, such a noise immunity is only effective on the one-pixel boundary noise (not the endpoint) because of the use of small support, and a better noise immunity for this kind of thinning may be obtained by designing a more complete set of thinning templates such as the comparison illustrated in Fig. 16. The tradeoff between noise immunity and endpoint preservation should also be considered. Finally, based on the work of this paper and the comment given in the end of Section 4.2, the more precise skeleton preservation and the more effective noise immunity in parallel thinning design under a reasonable computation cost paid may be a potential research topic in the future.

5 CONCLUDING SUMMARY

A common conclusion of recent investigators on thinning algorithms (systematic approach [13], fully parallel approach [20], evaluation approach [32], etc.) is given that: The processing speed of a parallel thinning algorithm using the reduction operators with small support is no longer a matter of significant concern because it is easily implemented by hardware using table-lookup scheme for example. The research direction may be focused on the improvement
of producing skeletons that preserve the significant geometric features of patterns. A major topic of this research is to develop a method embedded in a known parallel algorithm to produce bias-reduced skeletons (such as VRCT proposed by Li and Basu [27]) since a bias skeleton usually degrades the preservation of significant geometric features of patterns. The limitation of VRCT to reduce skeleton shape distortion has been explored in Section 4.1. The larger support algorithm is another possible solution to this topic. However the large support (up to $9 \times 9$ for example) algorithm seems to be impractical and has never been used so far [27]. Further, the useful version of the known $k \times k$ thinnings [9, 26] is $k = 3$ or 4 since “coarseness” (or noisy) skeletons for larger $k$ are produced based on their results. Similar to the work of Li and Basu, in this paper based on the comparable performance of our pseudo 1-subcycle parallel thinning algorithm (CH), a reduced form of larger support (two $L$-pixel vectors, which is not the form of $k \times k$ support) operated by an intermediate vector analysis about the deleted pixels in each thinning iteration has been developed for HDP detection to obtain bias-reduced skeletons, which can be purchased by a reasonable computation cost. HDP restoration and parallel implementation are further considered in Section 3.3 to formulate the improved algorithm (CYS), where the connectivity preservation is guaranteed by the use of CH’s operators and HDP restoration.

One-pixel boundary noise has been further considered for exploring the noise immunity. The results suggest that the noise immunity of CYS and CH is identical and is better than that of AFP3. A set of synthetic images are used to quantify the skeleton from the geometry viewpoint and investigate the skeleton variations of using different ($L$)s. Based on the analyzing results, $3 \leq L \leq 9$ are suggested for the current algorithm of CYS. CYS is further evaluated in comparison with two small support algorithms (AFP3 and CH) and one larger support algorithm (VRCT) using the same patterns. Performance are reported by number of iterations ($NI$), CPU time ($TC$, in msecs), and number of unmatched pixels ($N_{unmatch}$ for bias-reduced measure). Results show that on the measure of $TC$, CYS is approximately 2 to 3 times slower than the others, while on the measure of $NI$, the four algorithms have approximately identical performance. And on the measure of $N_{unmatch}$, CYS is approximately 2 to 3 times less than the others. Based on the tradeoff between computation cost and bias-reduced skeleton, we conclude that: when low computation cost is
a first priority, the small support algorithms (AFP3 or CH) may be considered. Otherwise, when the bias-reduced skeleton is strongly desirable, we recommend the proposed CYS to be used because the better bias-reduced skeleton may be purchased by a reasonable computation cost.

A few further works may be continued on the proposed approach. For example, the design of adaptive selection scheme for $L$, the possible solution of combining thinning operators, HDP detection, and/or HDP restoration in one step, the exploration on the more precise skeleton preservation and on the more effective noise immunity. As a summary, based on the work of this paper and the comment on noise immunity given in Section 4.2, the more precise skeleton preservation and the more effective noise immunity in parallel thinning design under a reasonable computation cost paid may be a potential research topic in the future.

ACKNOWLEDGMENTS

This work was partially supported by National Science Council of Republic of China under grant NSC 86-2213-E-155-005. The author is grateful to the referees; their thoughtful in-depth detailed comments have been very helpful in the revision of this article.

References


Figure 1: Thinning results obtained by (a) AFP3, (b) CH, and (c) VRCT algorithms. Where small dots represent the deleted pixels, large dots represent pixels of the medial curve.
Figure 2: Neighborhood definition.

\[
\begin{array}{ccc}
  p_7 & p_0 & p_1 \\
  p_6 & p & p_2 \\
  p_5 & p_4 & p_3 \\
  p_{10} & & \\
\end{array}
\]

And their 90° rotations.

(a)

\[
\begin{array}{ccc}
  q & 1 & 1 \\
  0 & p & 1 \\
  0 & 0 & r \\
\end{array} \quad \begin{array}{ccc}
  0 & q & 1 \\
  0 & p & r \\
  0 & 0 & 0 \\
\end{array} \quad \begin{array}{ccc}
  0 & 1 & 0 \\
  0 & p & 1 \\
  0 & 0 & 0 \\
\end{array} \quad \begin{array}{ccc}
  0 & 0 & 1 \\
  0 & p & 1 \\
  0 & 0 & 1 \\
\end{array}
\]

(b)

\[
\begin{array}{ccc}
  1 & 1 & x \\
  1 & p & 0 \\
  1 & 1 & x \\
\end{array} \quad \begin{array}{ccc}
  1 & 1 & 1 \\
  1 & p & 1 \\
  x & 0 & x \\
\end{array} \quad \begin{array}{ccc}
  0 & 0 & r \\
  0 & p & 1 \\
  q & 1 & 0 \\
\end{array} \quad \begin{array}{ccc}
  q & 1 & 0 \\
  0 & p & 1 \\
  0 & 0 & r \\
\end{array}
\]

(c)

\[
\begin{array}{ccc}
  x & 1 & 1 \\
  0 & p & 1 \\
  x & 1 & 1 \\
\end{array} \quad \begin{array}{ccc}
  x & 0 & x \\
  1 & p & 1 \\
  1 & 1 & 1 \\
\end{array} \quad \begin{array}{ccc}
  0 & 1 & r \\
  1 & p & 0 \\
  q & 0 & 0 \\
\end{array} \quad \begin{array}{ccc}
  q & 0 & 0 \\
  1 & p & 0 \\
  0 & 1 & r \\
\end{array}
\]

Figure 3: Three sets of thinning templates [13]. Where \(q \lor r = 1\), and \(x\) denotes don’t care.
Figure 4: Four restoring templates.

Figure 5: Thinning result obtained by Algorithm 1 (1-subcycle parallel thinning algorithm) in Section 2.
Figure 6: (a), (b), and (c) show the thinning iterations of a 135°, 90°, and 45° angle pattern, respectively, by CH, where (b) and (c) show the skeletons are not medial near sharp corners. (d)-(f) illustrate the first iteration of thinning these patterns, where “D” denotes a deleted pixel.
Figure 7: Illustration of HDP detection. (a) Representation of an angle consisting of a “1” pixel $p$ as well as two vectors $\overrightarrow{V_f}$ and $\overrightarrow{V_s}$. (b) $p$ is not an HDP. (c) $p$, and (d) $p$ and $q$ are HDPs. Where the found triangular regions are painted by grey. (e) The partial pixels in Fig. 6(f) are used to actually enumerate the HDPs considering $L = 3$ (see text for reference).
Figure 8: Direction code.

Figure 9: (a) Illustration of HDP detection (refer to text) after the first iteration of thinning the pattern shown in Fig. 6(c) by CYS with $L = 3$. (b) The second iteration. (c) The final result.
Figure 10: Restoring templates used in HDP restoration. Where \( w \) may be “0”, “D”, or “H” pixel; and \( x \) denotes don’t care.

(a)

(b)

Figure 11: Illustration of HDP restoration. (a) Connectivity is destroyed without using HDP restoration. (b) Two HDPs are restored and the connectivity is preserved. Note that in each iteration, the pixel labeled by “D” or “H” is changed as “0” at the end of that iteration.
Figure 12: (a)-(c) The bias-reduced skeletons obtained by CYS (with $L = 3$). (d)-(f) illustrate the first iteration of thinning these patterns. Where “H” denotes an HDP.
Figure 13: Thinning results obtained by **CYS** (with $L = 5$) algorithm.

Figure 14: (a) The first iteration of thinning the pattern shown in Fig. 6(c) by **CYS** with $L = 5$. In this illustration, the HDP detection is ineffective. (b) The second iteration. (c) The final result is the same as that shown in Fig. 6(c).
Figure 15: Thinning results obtained by VRCT algorithm. (a) The results of three patterns used in [27] are given for testing. Where straight vertical strokes are preserved. (b) The reason of bias skeletons produced by this algorithm (see Fig. 1(c)) is illustrated by these intermediate results. The $\alpha$ and $\beta$ tests used in VRCT may fail in further iterations on some regions of “1” pixels.
Figure 16: Thinning results of test pattern “H” with some one-pixel boundary noise. The noise immunity of (b) CH and (c) CYS is identical and is better than that of (a) AFP3.
Figure 17: Original thin patterns for analysis and comparison.
Figure 18: Original thin patterns for analysis and comparison.

angle_15  angle_20  angle_25  angle_30
angle_35  angle_40  angle_45  angle_50
angle_55  angle_60  angle_65  angle_70
angle_75  angle_80  angle_85  angle_90
Figure 19: $L$ analysis.
Figure 20: Thinning results obtained by AFP3 algorithm.
Figure 21: Thinning results obtained by CH algorithm.
Figure 22: Thinning results obtained by CYS algorithm.
Figure 23: Thinning results obtained by AFP3 algorithm.
Figure 24: Thinning results obtained by CH algorithm.
Figure 25: Thinning results obtained by CYS (with $L = 5$) algorithm.
Figure 26: Thinning results for the handwritten Chinese characters “CHEN”, “YUNG”, and “SHENG”. (a) AFP3; (b) CH; (c) CYS (with $L = 5$).
Table 1: Parallel computation time in number of iterations (NI) and CPU time (TC, in msecs).

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Sets</th>
<th>Algorithms</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>AFP3</td>
</tr>
<tr>
<td>star1</td>
<td></td>
<td></td>
<td>NI</td>
</tr>
<tr>
<td>star2</td>
<td></td>
<td></td>
<td>NI</td>
</tr>
<tr>
<td>star3</td>
<td></td>
<td></td>
<td>NI</td>
</tr>
<tr>
<td>sè</td>
<td></td>
<td></td>
<td>NI</td>
</tr>
<tr>
<td>pi</td>
<td></td>
<td></td>
<td>NI</td>
</tr>
<tr>
<td>yung</td>
<td></td>
<td></td>
<td>NI</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>NI</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>NI</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>NI</td>
</tr>
<tr>
<td>CHEN</td>
<td></td>
<td></td>
<td>NI</td>
</tr>
<tr>
<td>YUNG</td>
<td></td>
<td></td>
<td>NI</td>
</tr>
<tr>
<td>SHENG</td>
<td></td>
<td></td>
<td>NI</td>
</tr>
</tbody>
</table>

**Average** 16.67 1655.92 16.75 1397.08 16.83 1528.58 15.67 3758.75

^a L = 5   ^b L = 21   ^c L = 7   ^d L = 3
Table 2: Number of unmatched pixels between the resulting medial curve and the original thin pattern.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Original thin pattern size</th>
<th>Algorithms</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets</td>
<td></td>
<td>AFP3</td>
<td>CH</td>
<td>VRCT</td>
<td>CYS</td>
</tr>
<tr>
<td>star1</td>
<td>978</td>
<td>267</td>
<td>270</td>
<td>264</td>
<td>106$^b$</td>
</tr>
<tr>
<td>star2</td>
<td>939</td>
<td>238</td>
<td>243</td>
<td>243</td>
<td>66$^c$</td>
</tr>
<tr>
<td>star3</td>
<td>978</td>
<td>276</td>
<td>279</td>
<td>273</td>
<td>117$^d$</td>
</tr>
<tr>
<td>sè</td>
<td>853</td>
<td>149</td>
<td>150</td>
<td>143</td>
<td>9$^e$</td>
</tr>
<tr>
<td>pi</td>
<td>750</td>
<td>106</td>
<td>112</td>
<td>111</td>
<td>44$^b$</td>
</tr>
<tr>
<td>yung</td>
<td>542</td>
<td>62</td>
<td>63</td>
<td>63</td>
<td>19$^e$</td>
</tr>
<tr>
<td>4</td>
<td>510</td>
<td>52</td>
<td>53</td>
<td>52</td>
<td>14$^e$</td>
</tr>
<tr>
<td>5</td>
<td>505</td>
<td>31</td>
<td>31</td>
<td>29</td>
<td>7$^e$</td>
</tr>
<tr>
<td>7</td>
<td>342</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>0$^e$</td>
</tr>
</tbody>
</table>

Average: 725.22  132.89  135.11  132.56  42.44

$^a$ The total number of “1” pixels in the original thin pattern.

$^b$ $L = 5$  $^c$ $L = 21$  $^d$ $L = 7$  $^e$ $L = 3$
Table 3: Parallel computation time in number of iterations (NI) and CPU time (TC, in msecs).

<table>
<thead>
<tr>
<th>Pattern Sets</th>
<th>Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AFP3</td>
</tr>
<tr>
<td></td>
<td>NI</td>
</tr>
<tr>
<td>angle_15</td>
<td>13</td>
</tr>
<tr>
<td>angle_20</td>
<td>13</td>
</tr>
<tr>
<td>angle_25</td>
<td>14</td>
</tr>
<tr>
<td>angle_30</td>
<td>14</td>
</tr>
<tr>
<td>angle_35</td>
<td>14</td>
</tr>
<tr>
<td>angle_40</td>
<td>14</td>
</tr>
<tr>
<td>angle_45</td>
<td>13</td>
</tr>
<tr>
<td>angle_50</td>
<td>13</td>
</tr>
<tr>
<td>angle_55</td>
<td>13</td>
</tr>
<tr>
<td>angle_60</td>
<td>12</td>
</tr>
<tr>
<td>angle_65</td>
<td>12</td>
</tr>
<tr>
<td>angle_70</td>
<td>12</td>
</tr>
<tr>
<td>angle_75</td>
<td>11</td>
</tr>
<tr>
<td>angle_80</td>
<td>11</td>
</tr>
<tr>
<td>angle_85</td>
<td>11</td>
</tr>
<tr>
<td>angle_90</td>
<td>11</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
</tr>
<tr>
<td>N</td>
<td>13</td>
</tr>
<tr>
<td>Z</td>
<td>11</td>
</tr>
</tbody>
</table>

<sup>a</sup> \( L = 5 \)

Average: 12.37 88.89 12.53 51.74 13.05 62.79 11.95 164.89
Table 4: Number of unmatched pixels between the resulting medial curve and the original thin pattern.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Original thin pattern size(^a)</th>
<th>Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets</td>
<td></td>
<td>AFP3</td>
</tr>
<tr>
<td>angle,15</td>
<td>77</td>
<td>43</td>
</tr>
<tr>
<td>angle,20</td>
<td>76</td>
<td>42</td>
</tr>
<tr>
<td>angle,25</td>
<td>74</td>
<td>39</td>
</tr>
<tr>
<td>angle,30</td>
<td>74</td>
<td>33</td>
</tr>
<tr>
<td>angle,35</td>
<td>72</td>
<td>24</td>
</tr>
<tr>
<td>angle,40</td>
<td>70</td>
<td>19</td>
</tr>
<tr>
<td>angle,45</td>
<td>67</td>
<td>16</td>
</tr>
<tr>
<td>angle,50</td>
<td>70</td>
<td>14</td>
</tr>
<tr>
<td>angle,55</td>
<td>72</td>
<td>14</td>
</tr>
<tr>
<td>angle,60</td>
<td>74</td>
<td>13</td>
</tr>
<tr>
<td>angle,65</td>
<td>75</td>
<td>11</td>
</tr>
<tr>
<td>angle,70</td>
<td>77</td>
<td>11</td>
</tr>
<tr>
<td>angle,75</td>
<td>78</td>
<td>9</td>
</tr>
<tr>
<td>angle,80</td>
<td>78</td>
<td>7</td>
</tr>
<tr>
<td>angle,85</td>
<td>79</td>
<td>9</td>
</tr>
<tr>
<td>angle,90</td>
<td>79</td>
<td>7</td>
</tr>
</tbody>
</table>

Average 74.50 19.31 20.44 19.25 9.88

\(^a\) The total number of “1” pixels in the original thin pattern.
\(^b\) \(L = 5\)