DE GRUYTER OPEN

Int. J. Appl. Math. Comput. Sci., 2017, Vol. 27, No. 3, 637–658
DOI: 10.1515/amcs-2017-0045

amcs

ON THE FAREY SEQUENCE AND ITS AUGMENTATION FOR APPLICATIONS TO IMAGE ANALYSIS

SANJOY PRATIHAR [a,*], PARTHA BHOWMICK [b]

[a]Department of Computer Science and Engineering
National Institute of Technology Meghalaya, Bijni Complex, Shillong, India
e-mail: sanjoy.pratihar@gmail.com

[b]Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur, India

We introduce a novel concept of the *augmented Farey table* (AFT). Its purpose is to store the ranks of fractions of a *Farey sequence* in an efficient manner so as to return the *rank* of any query fraction in constant time. As a result, computations on the digital plane can be crafted down to simple integer operations; for example, the tasks like determining the extent of collinearity of integer points or of parallelism of straight lines—often required to solve many image-analytic problems—can be made fast and efficient through an appropriate AFT-based tool. We derive certain interesting characterizations of an AFT for its efficient generation. We also show how, for a fraction not present in a Farey sequence, the rank of the *nearest fraction* in that sequence can efficiently be obtained by the *regula falsi* method from the AFT concerned. To assert its merit, we show its use in two applications—one in polygonal approximation of digital curves and the other in skew correction of engineering drawings in document images. Experimental results indicate the potential of the AFT in such image-analytic applications.

Keywords: Farey sequence, Farey table, fraction rank, theory of fractions, image analysis.

1. Introduction

Given a positive integer n, the *Farey sequence* F_n of *order* n is the (ordered) sequence of all simple fractions starting from 0, ending at 1, and having denominators not exceeding n (Hardy and Wright, 1968). For example, the first five sequences are

$$F_1 = \left\langle \frac{0}{1}, \frac{1}{1} \right\rangle,$$
$$F_2 = \left\langle \frac{0}{1}, \frac{1}{2}, \frac{1}{1} \right\rangle,$$
$$F_3 = \left\langle \frac{0}{1}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{1}{1} \right\rangle,$$
$$F_4 = \left\langle \frac{0}{1}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{1}{1} \right\rangle,$$
$$F_5 = \left\langle \frac{0}{1}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{1}{1} \right\rangle.$$

If we disregard the ordering of F_n for sake of notational simplicity, then

$$F_n = \left\{ \frac{a}{b} : (0 \le a \le b \le n) \wedge (b > 0) \wedge (\gcd(a,b) = 1) \right\}. \quad (1)$$

The Farey sequences are named after John Farey, who first conjectured in 1816 that F_n can be obtained from F_{n-1}; a brief history following that can be found in the work of Hardy and Wright (1968). There are several works related with the Farey sequence, which mostly concern the *theory of fractions* (Hardy and Wright, 1968; Graham *et al.*, 1994; Neville, 1950; Pătraşcu and Pătraşcu, 2004; Pawlewicz and Pătraşcu, 2009; Schroeder, 2006). However, from the viewpoint of algorithms or computation, limited work has been done so far. A computationally interesting problem addressed in recent time is the *rank problem* and its associated *order statistic problem* (Pawlewicz and Pătraşcu, 2009). The *rank* $r_n(x)$ of a fraction x in a Farey sequence F_n is the

*Corresponding author

number of fractions less than or equal to x in F_n, i.e., $r_n(x) = |\{y : (y \leq x) \wedge (y \in F_n)\}|$. Given the order n, the rank problem is to find the rank of a given fraction in F_n, whereas the order statistic problem deals with finding the fraction in F_n for some given rank.

In this paper, we introduce some elementary properties of the Farey sequence, which are particularly useful for digital-geometric techniques related to image analysis. Based on these properties, we propose the concept of an *augmented Farey table* (AFT), from which the rank of a fraction in an *augmented Farey sequence* can be obtained by single memory access only. This, in turn, provides definite advantages of using the AFT as a tool while performing fraction-related computations in an image-analytic application.

An AFT is readily implementable as a 2D array whose rows are indexed by numerators and columns by denominators, and whose array entries store the fraction ranks. It becomes a natural choice in our applications, since the best-known algorithm can compute the rank of a fraction in no sooner than $O(n^{2/3}\log^{1/3} n)$ of time (Pawlewicz and Pătraşcu, 2009). Using the AFT, the fraction ranks are fetched almost instantaneously as they are queried very frequently in different procedural steps, such as comparing the slopes of two line segments with integer endpoints, checking the collinearity of three or more integer points, determining the type of turn (left or right) for three non-collinear integer points, etc. (cf. Das *et al.*, 2010; Pratihar and Bhowmick, 2010; 2011).

The motivation for our work on AFT owes to its particular advantage as a precomputed look-up table to support an application involving straight edge information and running on a large image dataset. Let us clarify at this early point its typical applicability through a simple example. Given three integer points (i.e., pixels) $p_1(i_1, j_1)$, $p_2(i_2, j_2)$, and $p_3(i_3, j_3)$ in succession, the metric used to decide on the deviation of p_2 from $\overline{p_1p_3}$ is given by $\Delta(p_1, p_2, p_3)/d_\infty(p_1, p_3)$, where $d_\infty(p_1, p_3) = \max(|i_1-i_3|, |j_1-j_3|)$ and $\Delta(p_1, p_2, p_3)$ denotes the area of the triangle with vertices p_1, p_2, p_3. This is a commonly followed practice (Bhowmick and Bhattacharya, 2007; Wall and Danielsson, 1984), which requires several multiplications for computing the value of $\Delta(p_1, p_2, p_3)$. Using an AFT provides a simpler and faster way to reach the solution by keeping away such multiplications, as shown in Section 4.

Our contributions are as follows:

(i) We show how a Farey sequence can be appropriately augmented for effectively using it in different applications. Theoretical results related to this are given in Sections 2.1 and 2.2. Further, as the existing algorithms for constructing a Farey sequence cannot readily be adapted for AFT generation, we propose an optimal-time algorithm for this in Section 2.3.

(ii) When a fraction has a large denominator and hence does not belong to an AFT of a lower order, the rank of its nearest fraction in that AFT can serve the requisite purpose in a computation involving fraction ranks. For this purpose, in Section 3, we derive certain theoretical results that aid designing an efficient technique for searching in an AFT for the closest fraction against any query fraction.

(iii) To emphasize the applicability of the AFT, we consider two well-known problems related to digital image analysis. One is polygonal approximation of digital curves (Section 4), and the other is skew correction of document images containing engineering drawings (Section 5). We show how an AFT can be used while determining the approximate collinearity of a sequence of 2D points having integer coordinates; the procedure involves only addition, comparison, and memory access, but no multiplication or division, thus reducing the overall runtime.

In essence, we show how the computation involved in the related algorithms can be crafted down to operations based on fraction ranks that are accessible from an AFT in real time. For the applications shown in the subsequent sections, extraction of straight edges is an important preprocessing task. For this purpose, we use the results of Bhowmick and Bhattacharya (2007) when the input image is a binary curve (Section 4), and Pratihar and Bhowmick (2009) when it is a gray-scale image (Section 5).

2. Farey sequence: Generation, augmentation, representation

Given two fractions $\frac{a}{b} \in F_n$ and $\frac{c}{d} \in F_n$, their *mediant* is defined as the fraction $\frac{a+c}{b+d}$, which always lies between $\frac{a}{b}$ and $\frac{c}{d}$ (Graham *et al.*, 1994). Based on this unique property, the sequence F_n can be computed from the sequence F_{n-1} by inserting in F_n the mediant of any two consecutive fractions of F_{n-1} unless the denominator of the mediant does not exceed n. To recursively generate F_n from F_{n-1} based on mediants, it is required to check all pairs of adjacent fractions in F_{n-1}. This calls for $\Theta((n-1)^2)$ steps, since F_n contains $f_n = \frac{3}{\pi^2}n^2 + O(n\log n) = \Theta(n^2)$ fractions (Hardy and Wright, 1968). Hence, the time complexity to generate F_n becomes $\Theta(n^3)$.

2.1. Generation of F_n. The mediant-based technique for generation of F_n, as explained above, is not optimal in runtime. As mentioned by Pătraşcu and Pătraşcu (2004) or Routledge (2008), and also by Graham *et al.* (1994), (Problem 4-61), the fractions of F_n can be generated using their adjacency relation in $O(n^2)$ iterations. We state this here in Theorem 1, which is proved using the following lemma.

Lemma 1. *For any fraction $\frac{a}{b} \in F_n$, the next fraction in F_n is $\frac{u+ka}{v+kb}$, where $k = \left\lfloor \frac{n-v}{b} \right\rfloor$ and (u, v) is any solution to $bx - ay = 1$.*

Proof. Since $\gcd(a, b) = 1$, there exist two integers u and v such that $bu-av = 1$ (Hardy and Wright, 1968). So, if $\frac{c}{d}$ lies next to $\frac{a}{b}$ in F_n, then $c = u+ka$ and $d = v+kb$, since $bc - ad = 1$ (Graham *et al.*, 1994). Also, the difference between these two fractions is $w = \frac{c}{d} - \frac{a}{b} = \frac{bc-ad}{bd} = \frac{1}{bd}$. Clearly, w is minimum when d is maximum. As $d \leq n$, we get $v + kb \leq n$, or, $k = \left\lfloor \frac{n-v}{b} \right\rfloor$, which completes the proof. ∎

Given any fraction $\frac{a}{b} \in F_n$, the above lemma says about the existence of the next fraction in F_n, but it does not provide an efficient way of solving it. Consequently, we extend this lemma to obtain an efficient way of finding the fraction lying immediately next to any two consecutive fractions in F_n. In particular, we have the following theorem. (Recall that $\frac{a}{b}$ is *simple* if $\gcd(a, b) = 1$; otherwise $\frac{a}{b}$ is a *compound* fraction and *equivalent* to its corresponding simple fraction, i.e., $\frac{a/g}{b/g}$, where $g = \gcd(a, b)$.)

Theorem 1. (Next fraction) *If $\frac{a}{b}$ and $\frac{c}{d}$ are two consecutive fractions in F_n, with $\frac{a}{b} < \frac{c}{d}$, then the next (simple or equivalent) fraction in F_n is given by*

$$\frac{e}{f} = \frac{kc - a}{kd - b}, \tag{2}$$

where $k = \left\lfloor \frac{n+b}{d} \right\rfloor$.

Proof. As $\frac{a}{b}, \frac{c}{d}$ are two adjacent fractions in F_n, we get $cb - ad = 1$, or, $d(-a) - c(-b) = 1$. Thus, by Lemma 1, the fraction next to $\frac{c}{d}$ is $\frac{kc-a}{kd-b}$, where $k = \left\lfloor \frac{n+b}{d} \right\rfloor$, which is our assertion. ∎

Using Theorem 1, the Farey sequence F_n can be generated, starting with $\langle \frac{0}{1}, \frac{1}{n} \rangle$, in $\Theta(n^2)$ time. However, to speed up the computation, we use the ranks of the leading fractions in F_n to obtain those of the trailing fractions. To this end, we define the lower half of F_n by $F_n^L = F_n \cap [0, \frac{1}{2})$, and the upper half by $F_n^U = F_n \cap (\frac{1}{2}, 1]$, so that $F_n = F_n^L \cup \{\frac{1}{2}\} \cup F_n^U$. Now we have the following theorem.

Theorem 2. (Complementary rank) *If $x \in F_n$, then $r_n(1 - x) = f_n + 1 - r_n(x)$, where $f_n = r_n\left(\frac{1}{1}\right) = 2r_n\left(\frac{1}{2}\right) - 1$.*

Proof. Observe that $x \in F_n^L$ if and only if $y := 1 - x \in F_n^U$, which implies that $F_n^L \mapsto F_n^U$ is a bijection. Hence, $\frac{1}{2} - x = y - \frac{1}{2}$, or $r_n\left(\frac{1}{2}\right) - r_n(x) = r_n(y) - r_n\left(\frac{1}{2}\right)$, or $r_n(x) + r_n(y) = 2r_n\left(\frac{1}{2}\right) = f_n + 1$. ∎

Theorem 2 is used in Section 2.3 for efficiently generating an AFT.

Numerator \ Denominator	-4	-3	-2	-1	0	1	2	3	4
4	19	18	16	14	13	12	10	8	7
3	20	19	17	15	13	11	9	7	6
2	22	21	19	16	13	10	7	5	4
1	24	23	22	19	13	7	4	3	2
0	25	25	25	25	–	1	1	1	1
-1	26	27	28	31	37	43	46	47	48
-2	28	29	31	34	37	40	43	45	46
-3	30	31	33	35	37	39	41	43	44
-4	31	32	34	36	37	38	40	42	43

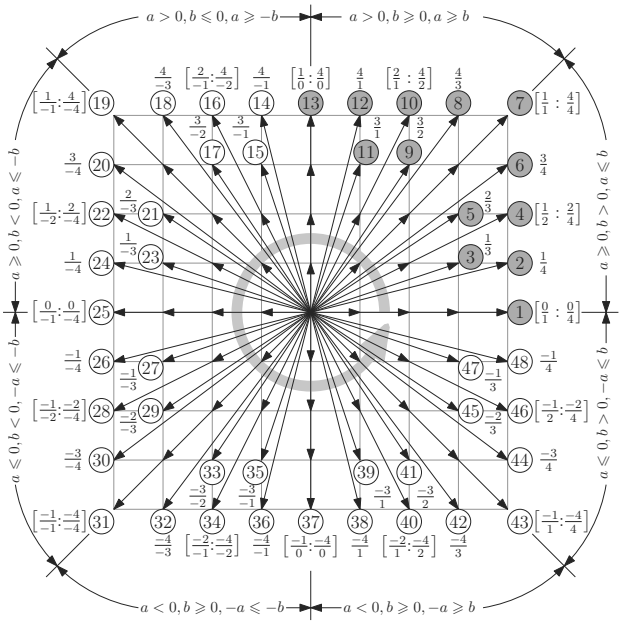


Fig. 1. Geometric interpretation (bottom) of the complete AFT of order 4 (top), i.e., $\mathbf{F}_4 := \mathbf{F}_4^{(1)} \cup \mathbf{F}_4^{(2)} \cup \mathbf{F}_4^{(3)} \cup \mathbf{F}_4^{(4)}$, containing the ranks of all fractions in $\overline{F}_4 := \{\frac{a}{b} : |a| \leq 4 \wedge |b| \leq 4\}$. Notice the counter-clockwise increase in ranks of the fractions in $\mathbf{F}_4$.

2.2. Augmentation of F_n. As the Farey sequence F_n is restricted only to simple and proper fractions, it needs an appropriate augmentation for geometric applications. The rationale is as follows. Let pq be a straight line segment with integer endpoints $p = (x_p, y_p)$ and $q = (x_q, y_q)$. Also, let $\frac{\max\{|x_q - x_p|, |y_q - y_p|\}}{\gcd(|x_q - x_p|, |y_q - y_p|)} \leq n$. Then the slope of pq equals some fraction in F_n if and only if $0 \leq y_q - y_p \leq x_q - x_p \leq n$. In other words, F_n corresponds to just one half of the first quadrant, as shown in Fig. 1. Hence, to overcome this limitation, we augment F_n to $\overline{F}_n$, so that it includes compound fractions and proper/improper fractions with negative numerators or/and denominators. We call $\overline{F}_n$ the *augmented Farey sequence*.

2.3. Farey table. For efficient storage and subsequent use of $\overline{F}_n$, we construct the *augmented Farey table* (AFT), $\mathbf{F}_n$, which is a 2D array of size $(2n+1) \times (2n+1)$.

Figure 1 shows an example for $n = 4$. For simplicity, we allow negative row and negative column indices in $\mathbf{F}_n$. Each row $i \in [-n, n]$ represents a numerator and each column $j \in [-n, n]$ a denominator. The value stored in $\mathbf{F}_n[i][j]$ contains the rank of the fraction $\frac{i}{j}$ in $\overline{F}_n$.

The array $\mathbf{F}_n$ comprises four sub-arrays, namely, $\{\mathbf{F}_n^{(k)} : k = 1, 2, 3, 4\}$, which store the ranks of fractions in $\overline{F}_n^{(k)}$ for $k = 1, 2, 3, 4$, and correspond to line slopes in all the four quadrants. For example, $\mathbf{F}_n^{(1)}$ stores the ranks of the fractions in $\overline{F}_n^{(1)}$, i.e., $\mathbf{F}_n^{(k)}[a][b] = r_n\left(\frac{a}{b}\right) \forall (a, b) \in [0, n]^2$. The following theorem shows how the ranks of fractions in $\mathbf{F}_n^{(k)}$ for $k \neq 1$ are obtainable from those in $\mathbf{F}_n^{(1)}$.

Theorem 3. (All ranks) *If* $(a, b) \in [0, n] \times [0, n] \setminus \{(0, 0)\}$, *then*

$$\mathbf{F}_n^{(1)}[a][b] = \begin{cases} r_n\left(\dfrac{a}{b}\right) \quad \text{if } a \leq \dfrac{b}{2} & \text{(3a)} \\ 2r_n\left(\dfrac{1}{2}\right) - r_n\left(1 - \dfrac{a}{b}\right) \\ \qquad \text{if } \dfrac{b}{2} < a \leq b & \text{(3b)} \\ 2f_n - r_n\left(\dfrac{b}{a}\right) \quad \text{if } a > b & \text{(3c)} \end{cases}$$

Otherwise, we have the following three cases:

$$a \in [0, n], b \in [-n, -1] \implies \mathbf{F}_n^{(2)}[a][b] = 4f_n - 2 - \mathbf{F}_n^{(1)}[a][-b], \quad \text{(4a)}$$

$$a \in [-n, -1], b \in [-n, -1] \implies \mathbf{F}_n^{(3)}[a][b] = 8f_n - 6 - \mathbf{F}_n^{(1)}[-a][b], \quad \text{(4b)}$$

$$a \in [-n, -1], b \in [0, n] \implies \mathbf{F}_n^{(4)}[a][b] = 12f_n - 10 - \mathbf{F}_n^{(1)}[a][-b]. \quad \text{(4c)}$$

Proof. The formula (3) follows from Theorem 2 and (4) from the reflection symmetry among the four quadrants. ■

Theorem 3 implies that we can store only $\mathbf{F}_n^{(1)}$ and use it to get the rank of any fraction in $\mathbf{F}_n$ in constant time. This is particularly useful when we need to work with a Farey table of a higher order.

Theorems 1–3 are used to design Algorithm 1 for generation of $\mathbf{F}_n^{(1)}$. An illustration of this algorithm for $n = 10$ is shown in Fig. 2. The cells highlighted in light gray contain the ranks of the fractions in $[0, \frac{1}{2}]$. The ranks of the smallest element (i.e., $0/1$ and its equivalent fractions) and the next element (i.e., $1/n$) are generated by the first **for** loop. Then, all the fractions in $[0, \frac{1}{2}]$ are generated by the **repeat-until** loop. Fractions in $[\frac{1}{2}, 1]$ are generated by the subsequent **for** loop; cells are highlighted in dark gray in Fig. 2). The last **for** loop generates the ranks of the fractions greater than 1.

Algorithm 1. GEN-AFT.

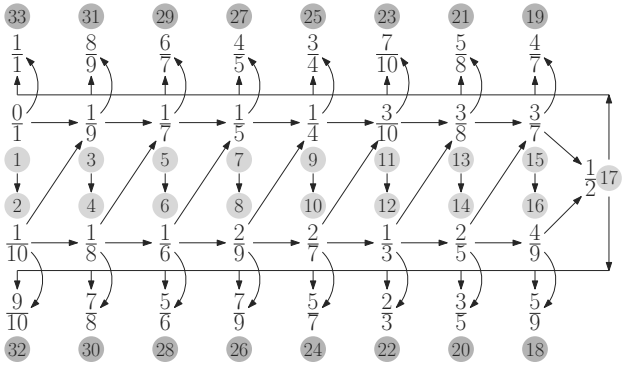
Input: Order of Farey sequence: n
Output: F_n, $\mathbf{F}_n^{(1)}$
int $a \leftarrow 0, b \leftarrow 1, c \leftarrow 1, d \leftarrow n, e, f, k, g, rank \leftarrow 2$
for $i \leftarrow 1, 2, \ldots, n$ **do**
  $\mathbf{F}_n^{(1)}[0][i] \leftarrow 1$
$F_n[1] \leftarrow 0/1, F_n[2] \leftarrow 1/n, \mathbf{F}_n^{(1)}[1][n] \leftarrow 2$
repeat
  $rank \leftarrow rank + 1$
  $k \leftarrow \lfloor (n + b)/d \rfloor$ ▷ Theorem 1
  $e \leftarrow k \cdot c - a,\ f \leftarrow k \cdot d - b$
  $g \leftarrow \gcd(e, f)$
  $e \leftarrow e/g, f \leftarrow f/g$
  $F_n[rank] \leftarrow e/f, \mathbf{F}_n^{(1)}[e][f] \leftarrow rank$
  for $i \leftarrow 1, 2, \ldots, \lfloor n/f \rfloor$ **do**
    $\mathbf{F}_n^{(1)}[e \cdot i][f \cdot i] \leftarrow rank$ ▷ equivalent fractions
  $a \leftarrow c, b \leftarrow d, c \leftarrow e, d \leftarrow f$
until $(e/f = 1/2)$
$f_n \leftarrow 2 \cdot rank - 1$ ▷ Theorem 2
for $a \leftarrow 1, 2, \ldots, n$ **do**
  for $b \leftarrow a, a + 1, \ldots, 2a - 1$ **do**
    $\mathbf{F}_n^{(1)}[a][b] \leftarrow f_n + 1 - \mathbf{F}_n^{(1)}[b - a][b]$ ▷ Theorem 3, Eqn. 3(b)
for $a \leftarrow 1, 2, \ldots, n$ **do**
  for $b \leftarrow 0, 1, \ldots, a - 1$ **do**
    $\mathbf{F}_n^{(1)}[a][b] \leftarrow 2 \cdot f_n - \mathbf{F}_n^{(1)}[b][a]$ ▷ Theorem 3, Eqn. 3(c)
return $\mathbf{F}_n^{(1)}$

Time complexity. The algorithm uses $O(n^2)$ iterations. The dominant operation in an iteration is for GCD computation, which runs with $O(\log n)$ *arithmetic operations* or $O(\log^2 n)$ *bit operations*, since the operands are bounded above by n (Cormen *et al.*, 2000). Hence, the time complexity of Algorithm 1 is $O(n^2 \log n)$ measured in terms of arithmetic operations and $O(n^2 \log^2 n)$ in terms of bit operations.

3. Finding the closest rank in $\mathbf{F}_n^{(1)}$

While processing fractions and their ranks, if some fraction $\frac{a}{b}$ appears with $a > n$ or $b > n$, then it does not belong to F_n, and so its rank cannot be found in $\mathbf{F}_n^{(1)}$. To tackle this situation, we find its *closest fraction* $\frac{a'}{b'}$ in F_n, and use the rank of $\frac{a'}{b'}$ in subsequent applications. We call this rank the *closest rank* corresponding to $\frac{a}{b}$.

To search for the closest fraction $\frac{a'}{b'}$, we use a range (i.e., subsequence) of fractions, namely $[\frac{a_1}{b_1}, \frac{a_2}{b_2}] \subset F_n$, to speed up the process. The following theorem provides this range.



		Denominator										
		0	1	2	3	4	5	6	7	8	9	10
Numerator	0	–	1	1	1	1	1	1	1	1	1	1
	1	65	33	17	12	9	7	6	5	4	3	2
	2	65	49	33	22	17	14	12	10	9	8	7
	3	65	54	44	33	25	20	17	15	13	12	11
	4	65	57	49	41	33	27	22	19	17	16	14
	5	65	59	52	46	39	33	28	24	21	18	17
	6	65	60	54	49	44	38	33	29	25	22	20
	7	65	61	56	51	47	42	37	33	30	26	23
	8	65	62	57	53	49	45	41	36	33	31	27
	9	65	63	58	54	50	48	44	40	35	33	32
	10	65	64	59	55	52	49	46	43	39	34	33

Fig. 2. Generation of $\mathbf{F}_{10}^{(1)}$ using Algorithm 1.

Theorem 4. (Fraction search) *For any proper fraction* $\frac{a}{b}$*, let*

$$\gcd(\lfloor an/b \rfloor, n) = g_1,$$

$$\gcd(\lceil an/b \rceil, n) = g_2,$$

$$a_1 = \frac{\lfloor an/b \rfloor}{g_1}, \quad b_1 = \frac{n}{g_1},$$

$$a_2 = \frac{\lceil an/b \rceil}{g_2}, \quad b_2 = \frac{n}{g_2},$$

such that

$$\gcd(a_1, b_1) = \gcd(a_2, b_2) = 1.$$

Then, for

$$\frac{a_1}{b_1} := \frac{\lfloor an/b \rfloor}{n}, \quad \frac{a_2}{b_2} := \frac{\lceil an/b \rceil}{n},$$

the following statements are true:

(i) $\dfrac{a_1}{b_1} \in F_n$, $\dfrac{a_2}{b_2} \in F_n$.

(ii) $\dfrac{a_1}{b_1} \le \dfrac{a}{b} \le \dfrac{a_2}{b_2}$.

Proof.

(i) As $0 \le a \le b \le n$, we have $0 \le \lfloor an/b \rfloor \le n$ and $0 \le \lceil an/b \rceil \le n$. This gives $\frac{a_1}{b_1} \in F_n$ and $\frac{a_2}{b_2} \in F_n$.

(ii) We have $\frac{a_1}{b_1} \le \frac{an/b}{n} \le \frac{a_2}{b_2}$, which implies $\frac{a_1}{b_1} \le \frac{a}{b} \le \frac{a_2}{b_2}$.

■

We use the result of Theorem 4 later in Algorithm 2. In particular, we use the following corollary, which follows from the two statements of Theorem 4 when taken together.

Corollary 1. *We have that* $\frac{a}{b} \in F_n$ *if* $\frac{a_1}{b_1} = \frac{a_2}{b_2}$*. However, if* $\frac{a_1}{b_1} < \frac{a_2}{b_2}$*, then* $\frac{a}{b}$ *may or may not belong to* F_n*.*

Also,

$$\frac{a_1}{b_1} < \frac{a_2}{b_2} \Leftrightarrow \frac{\lfloor an/b \rfloor}{n} < \frac{\lceil an/b \rceil}{n},$$

and hence the following corollary:

Corollary 2. *If* $\frac{a_1}{b_1} < \frac{a_2}{b_2}$*, then* $\frac{a_1}{b_1} = \frac{m}{n}$ *and* $\frac{a_2}{b_2} = \frac{m+1}{n}$*, where* $m = \lfloor \frac{an}{b} \rfloor$.

If we search for the closest fraction of $\frac{a}{b}$ in $[\frac{a_1}{b_1}, \frac{a_2}{b_2}]$ using binary search, it may take $O(\log n)$ time in the worst case. The following theorem accounts for this.

Theorem 5. (Search range) F_n *contains at most* $O(n)$ *fractions in the range* $[\frac{a_1}{b_1}, \frac{a_2}{b_2}]$*, where* a_1, b_1, a_2, b_2 *are defined as in Theorem 4.*

Proof. From Corollary 2, $\frac{a_1}{b_1} = \frac{m}{n}$ and $\frac{a_2}{b_2} = \frac{m+1}{n}$ when $\frac{a_1}{b_1} < \frac{a_2}{b_2}$. Let, for a given positive integer d, $\frac{c}{d}$ be a

fraction of F_n, if possible, lying between (and excluding) $\frac{m}{n}$ and $\frac{m+1}{n}$. Observe that $d < n$. We now show that no other fraction of F_n with denominator d can exist in $[\frac{a_1}{b_1}, \frac{a_2}{b_2}]$.

As $d < n$, we get $\frac{1}{d} > \frac{1}{n}$, or, $\frac{c}{d} - \frac{c-1}{d} > \frac{m+1}{n} - \frac{m}{n} > \frac{c}{d} - \frac{m}{n}$, or, $\frac{c-1}{d} < \frac{m}{n}$, which implies that any a fraction with denominator d and a numerator less than c lies outside $[\frac{a_1}{b_1}, \frac{a_2}{b_2}]$. Similarly, any fraction with denominator d and a numerator greater than c also lies outside this range, as $\frac{1}{d} > \frac{1}{n}$ implies $\frac{c+1}{d} - \frac{c}{d} > \frac{m+1}{n} - \frac{m}{n} > \frac{m+1}{n} - \frac{c}{d}$.

Thus, for every integer d in $[1, n-1]$, at most one fraction of F_n can be there in $[\frac{a_1}{b_1}, \frac{a_2}{b_2}]$. Therefore, there can be at most $n-1$ fractions of F_n lying in $[\frac{a_1}{b_1}, \frac{a_2}{b_2}]$. ■

3.1. Binary search and its limitation. To study the performance of binary search, we consider various cases for searching for some key fractions in Farey tables of small orders. For this, we define the *fraction difference* as the difference of a fraction $\frac{c}{d}$ of F_n from the key fraction $\frac{a}{b}$, given by $\frac{a}{b} - \frac{c}{d} = \frac{ad-bc}{bd}$. Computation of each such difference needs $O(\log^{\log 3} n)$ *bit operations* if we use the Karatsuba algorithm for multiplication, since the numerator and the denominator of a fraction in F_n is bounded above by n (Knuth, 1997). We compute the differences of all the fractions of F_n having rank in $[r_1, r_2]$ from the key fraction $\frac{a}{b}$.

These differences would be monotonically decreasing with the fraction ranks increasing from r_1 to r_2. Further, they are initially positive and gradually become negative, with a *zero-crossing* somewhere in between. Clearly, the closest fraction of $\frac{a}{b}$ in F_n is the fraction lying nearest to this zero-crossing, which needs $O(\log n)$ iterations for searching using binary search. In fact, for instances when the zero-crossing lies near one of the boundaries (r_1 or r_2), the binary search algorithm really takes $O(\log n)$ iterations (Fig. 3(b)). Hence, the number of bit operations is given by $O(\log n \log^{\log 3} n) \approx O(\log^{2.585} n)$.

3.2. Algorithm using *regula falsi*. The nature of the variation in the fraction difference is grossly linear with the fraction rank. This is evident from three different plots given in Fig. 3. Thus, as a better alternative, we use an iterative approach based on the *regula falsi* method to find the zero-crossing. The rationale is that the *regula falsi* method provides the exact solution for a linear function in constant time. For a nonlinear function, it provides an approximate solution that can be successively improved by iteration. Further, if a function is not exactly but grossly linear, then it provides an approximate solution very fast, and the number of iterations depends on the grossness of linearity.

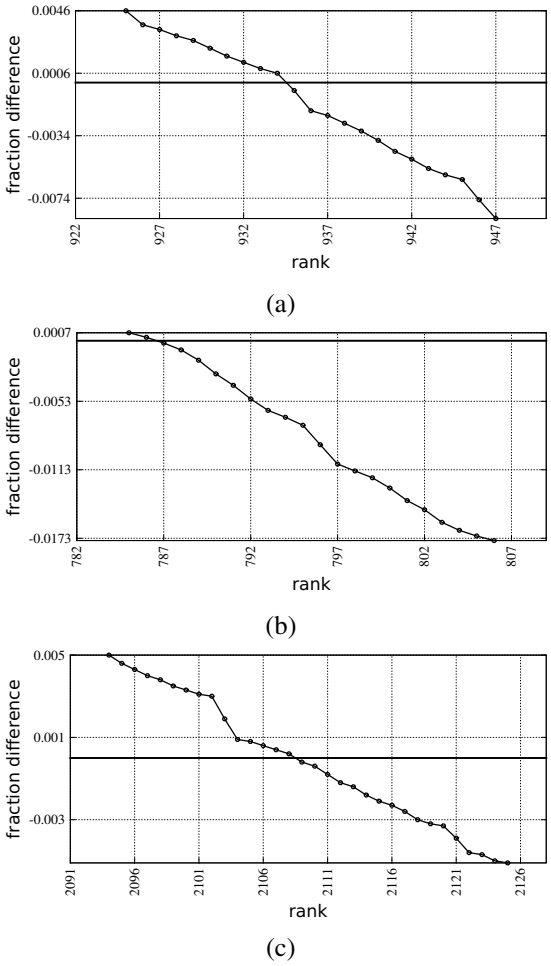


Fig. 3. Instances of various cases for searching for fractions in lower order Farey tables using Algorithm 2: searching for $a/b = 78/145$ in F_{75}, the closest fraction in $F_{75} = 7/13$, rank$= 935$ (a), searching for $a/b = 375/448$ in F_{55}, the closest fraction in $F_{55} = 36/43$, rank$= 787$(b), searching for $a/b = 1957/2788$ in F_{99}, the closest fraction in $F_{99} = 40/57$, rank$= 2108$ (c). The black solid lines indicate zero-crossing.

In our algorithm FIND-CLOSEST-RANK (Algorithm 2), after each iteration, we get two closer ranks between which the zero-crossing lies. The iteration continues depending on how close the ranks are, and it stops when the two ranks have unit a difference. The initialized search range is $[a_1/b_1, a_2/b_2]$, where $r_n(a_1/b_1) = r_1$ and $r_n(a_2/b_2) = r_2$. The fractions a_1/b_1 and a_2/b_2 are computed at the beginning of the algorithm (Algorithm 2). By Theorem 4, the query fraction a/b belongs to the range given by $[a_1/b_1, a_2/b_2]$. We have now the following three cases.

Case 1 ($r_1 = r_2$). The query fraction belongs to F_n, and it is equal to a_1/b_1 (Corollary 1). Therefore, the algorithm returns the rank of a_1/b_1.

Algorithm 2. FIND-CLOSEST-RANK.

Input: $\mathbf{F}_n^{(1)}$, F_n, $\frac{a}{b}$
Output: Rank of closest fraction of $\frac{a}{b}$ in $\mathbf{F}_n^{(1)}$

$g_1 \leftarrow \gcd(\lfloor an/b \rfloor, n), g_2 \leftarrow \gcd(\lceil an/b \rceil, n)$
$a_1 \leftarrow \lfloor an/b \rfloor / g_1, b_1 \leftarrow n/g_1$
$a_2 \leftarrow \lceil an/b \rceil / g_2, b_2 \leftarrow n/g_2$
$r_1 \leftarrow \mathbf{F}_n^{(1)}[a_1][b_1], r_2 \leftarrow \mathbf{F}_n^{(1)}[a_2][b_2]$
if $r_1 = r_2$ **then**
  return $\mathbf{F}_n^{(1)}[a_1][b_1]$ ▷ Corollary 1
if $r_2 - r_1 = 1$ **then**
  if $(a/b - a_1/b_1) < (a_2/b_2 - a/b)$ **then**
    return $\mathbf{F}_n^{(1)}[a_1][b_1]$
  else
    return $\mathbf{F}_n^{(1)}[a_2][b_2]$
$a_1'/b_1' \leftarrow a_1/b_1, a_2'/b_2' \leftarrow a_2/b_2$
repeat
  if $a/b < a_2'/b_2'$ **then**
    $a_2/b_2 \leftarrow a_2'/b_2'$
  else
    $a_1/b_1 \leftarrow a_2'/b_2'$
  $y_1 \leftarrow (a_1 b - b_1 a)/(b_1 b),\ y_2 \leftarrow (a_2 b - b_2 a)/(b_2 b)$
  $r \leftarrow (r_1 y_2 - r_2 y_1)/(y_2 - y_1)$ ▷ zero-crossing
  $a_1'/b_1' \leftarrow F_n[\lfloor r \rfloor]$
  if $a_1'/b_1' < a/b$ **then**
    $a_2'/b_2' \leftarrow F_n[\lfloor r + 1 \rfloor]$
  else
    $a_2'/b_2' \leftarrow F_n[\lfloor r - 1 \rfloor]$
until
$((a_1'/b_1' \le a/b \le a_2'/b_2') \vee (a_2'/b_2' \le a/b \le a_1'/b_1'))$
if $a/b - a_1'/b_1' < a_2'/b_2' - a/b$ **then**
  return $\mathbf{F}_n^{(1)}[a_1'][b_1']$
else
  return $\mathbf{F}_n^{(1)}[a_2'][b_2']$

Case 2 ($r_2 - r_1 = 1$). There are only two fractions in the range $[a_1/b_1, a_2/b_2]$. Therefore, the fraction between a_1/b_1 and a_2/b_2 lying closest to a/b is returned, using the **if**-**else** statements.

Case 3 ($r_2 - r_1 > 1$). The length of the search range is reduced iteratively for this case, until $r_2 - r_1 = 1$. This is done in the **repeat**-**until** loop. The loop terminates when $a_1'/b_1' \le a/b \le a_2'/b_2'$ or $a_2'/b_2' \le a/b \le a_1'/b_1'$. In either case, the ranks of a_1'/b_1' and a_2'/b_2' differ by unity.

The fractions a_1'/b_1' and a_2'/b_2' are used to update the range $[a_1/b_1, a_2/b_2]$. The fraction a_2'/b_2' gets the value of the fraction having rank $\lfloor x + 1 \rfloor$ when $a_1'/b_1' < a/b$; otherwise, it gets the value of the fraction having rank $\lfloor x - 1 \rfloor$.

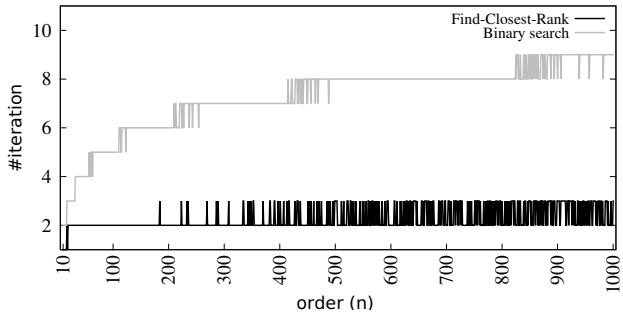


Fig. 4. Comparison between FIND-CLOSEST-RANK and binary search. The average number of iterations (rounded off to integers) is plotted vs. n.

3.3. Performance of FIND-CLOSEST-RANK. For studying the performance of FIND-CLOSEST-RANK and to compare it with binary search, a set of 100 randomly generated proper fractions in $[0, 1]$ is taken, and the *average number of iterations* required for each of these two methods is considered for $\mathbf{F}_n^{(1)}$, with its order n varying from 10 to 1000. We generate the set of random fractions such that no fraction belongs to F_n. The corresponding average number of iterations required for these randomly generated sets for different $\mathbf{F}_n^{(1)}$ is plotted against n in Fig. 4. From this plot, it can be noticed that FIND-CLOSEST-RANK runs significantly faster compared with binary search. Even for n of order of 1000, FIND-CLOSEST-RANK reports the closest rank in just two or three iterations.

3.4. Comparison with the existing technique. Recently, an algorithm has been proposed by Charrier and Buzer (2009) for determining the fraction $\frac{a'}{b'} \in F_n$ so that it is closest to any given fraction $\frac{a}{b}$. The technique is based on the principal convergent series of an even order and that of an odd order, denoted respectively by S_E and S_O, which are obtained by decomposition to a continued fraction. As a natural follow-up, we compare our FIND-CLOSEST-RANK with this algorithm, and some examples on the comparison are given below.

Example 1. To detect the closest fraction of $\frac{a}{b} = \frac{78}{145}$ where its denominator is less than or equal to $n = 75$.

We have

$$\frac{78}{145} = [0 \quad 1 \quad 1 \quad 6 \quad 11].$$

Therefore,

$$S_E = \{0/1, 1/2, 8/15, 15/28, 22/41, 29/54, 36/67, 43/80, 50/93, 57/106, 64/119, 71/132, 78/145\}$$

Table 1. Sub-cases of Case 3 (FIND-CLOSEST-RANK).

Sub-case 3a	Sub-case 3b	Sub-case 3c	Sub-case 3d
$\frac{a'_1}{b'_1} < \frac{a}{b}$		$\frac{a'_1}{b'_1} \geq \frac{a}{b}$	
$\frac{a'_2}{b'_2} < \frac{a}{b}$ $\implies \frac{a'_1}{b'_1} < \frac{a'_2}{b'_2} < \frac{a}{b}$	$\frac{a'_2}{b'_2} \geq \frac{a}{b}$ $\implies \frac{a'_1}{b'_1} < \frac{a}{b} < \frac{a'_2}{b'_2}$	$\frac{a'_2}{b'_2} < \frac{a}{b}$ $\implies \frac{a'_2}{b'_2} < \frac{a}{b} < \frac{a'_1}{b'_1}$	$\frac{a'_2}{b'_2} \geq \frac{a}{b}$ $\implies \frac{a}{b} < \frac{a'_2}{b'_2} < \frac{a'_1}{b'_1}$
action:			
$\frac{a_1}{b_1} \leftarrow \frac{a'_2}{b'_2}$	**return** closest rank	**return** closest rank	$\frac{a_2}{b_2} \leftarrow \frac{a'_2}{b'_2}$

and

$$S_O = \{1/1, 2/3, 3/5, 4/7, 5/9, 6/11, 7/13\}.$$

The closest fraction of $\frac{a}{b}$ in S_E is $\frac{36}{67}$ with error 0.000618 and in S_O is $\frac{7}{13}$ with error 0.000531. Hence, the reported closest fraction is $\frac{7}{13}$. Our algorithm also reports the same (see Fig. 3(a)). ♦

Example 2. Let $\frac{a}{b} = \frac{1957}{2788}, n = 99$. We have

$$\frac{1957}{2788} = [0 \quad 1 \quad 2 \quad 2 \quad 1 \quad 4 \quad 2 \quad 6 \quad 4].$$

Thus,

$$S_E = \{0/1, 1/2, 2/3, 7/10, 40/57, 73/104, 544/775, 1015/1446, 1486/2117, 1957/2788\}$$

and

$$S_O = \{1/1, 3/4, 5/7, 12/17, 19/27, 26/37, 33/47, 106/151, 179/255252/359, 325/463, 398/567, 471/671\}.$$

The closest fraction reported is $\frac{40}{57}$, which matches our result (see Fig. 3(c)). ♦

The time complexity of the algorithm is mainly determined by the number of iterations in the **repeat-until** loop. Since the fraction difference has a grossly linear relation with rank, this loop converges very fast within a small number of iterations because of the interpolation by the *regula falsi* method. Hence, the time complexity becomes a constant factor of the arithmetic operations used in different steps of the algorithm. The major operations are GCD computation and multiplication/division, whose respective time complexities in terms of bit operations are $O(\log^2 n)$ and $O(\log^{\log 3} n)$ (Cormen *et al*., 2000). As a result, the overall time complexity becomes $O(\log^2 n)$, which is asymptotically better than the algorithm based on binary search.

4. Application to polygonal approximation

Polygonal approximation is an important task in the paradigm of computer vision, since it can succinctly represent the boundary of an object (Melkman and O'Rourke, 1988). The result can subsequently be used in various applications, such as image analysis (Neumann and Teisseron, 2002), image registration (Wang *et al*., 2013; Zhang *et al*., 2015), image matching (Mikolajczyk and Schmid, 2005; Wang *et al*., 2009; Zhang and Koch, 2013), shape understanding (Attneave, 1954; Kumar *et al*., 2002; O'Connell, 1997), image and video retrieval (Mokhtarian and Mohanna, 2002; Van and Le, 2016). There exists a multitude of algorithms for polygonal (or poly-chain) approximation of a digital curve; see, for example, the works of Bhowmick and Bhattacharya (2007), Wall and Danielsson (1984), Chung *et al*. (2008), Koutroumbas (2012), Liu *et al*. (2008), Nguyen and Debled-Rennesson (2011), Parvez and Mahmoud (2010), Prasad *et al*. (2012), Rosin (1997), Teh and Chin (1989) or Yin (2004) and the references therein. These are based on various techniques like normalization of area deviation (Wall and Danielsson, 1984), curvature estimation (Liu *et al*., 2008; Teh and Chin, 1989), dominant point detection (Prasad *et al*., 2012), particle swarm optimization (Yin, 2004), approximate digital straightness (Bhowmick and Bhattacharya, 2007), etc. Working principles of these algorithms and their comparative study are given in Table 2.

We show here how the AFT can be used for polygonal approximation of a (thinned) digital curve, once the *approximately digital straight segments* (ADSSs) are extracted from the curve (Bhowmick and Bhattacharya, 2007). For this purpose, the endpoints of the ADSS are taken in order and stored in V. Each maximal subsequence of vertices in V, which are "almost collinear", is replaced by a single edge. That is, if $\langle e_h, e_{h+1}, \ldots, e_{h+i} \rangle$ is the maximal subsequence of straight edges that are almost collinear, i.e., they conform to some approximation criterion, then these $(i+1)$ edges are combined to a single edge. The process is repeated for all such maximal subsequences in succession to obtain a reduced sequence of (almost) straight edges

Algorithm 3. POLY-APPROX-FT.

Input: $V, m, \mathbf{F}_n^{(1)}, \tau$
Output: V'

$V' \leftarrow V$
for $h \leftarrow 1$ *to* $m-2$ **do**
  $i \leftarrow 1, iLoop \leftarrow$ TRUE
  while $((i < m-h) \wedge (iLoop =$ TRUE$))$ **do**
    $j \leftarrow 1$
    while $j \leq i$ **do**
      $dx_1 \leftarrow V[h+i+1].x - V[h].x$
      $dy_1 \leftarrow V[h+i+1].y - V[h].y$
      $dx_2 \leftarrow V[h+j].x - V[h+i+1].x$
      $dy_2 \leftarrow V[h+j].y - V[h+i+1].y$
      $l \leftarrow \max(|dx_2|, |dy_2|)$
      $r_1 \leftarrow \mathbf{F}_n^{(1)}[dy_1][dx_1]$ ▷ Eqns. 3, 4
      $r_2 \leftarrow \mathbf{F}_n^{(1)}[dy_2][dx_2]$ ▷ Eqns. 3, 4
      $r \leftarrow \delta_n(r_1, r_2)$
      $a/b \leftarrow F_n[r]$
      if $l \cdot |a| > \tau \cdot \sqrt{a^2+b^2}$ **then**
        $i \leftarrow i-1, iLoop \leftarrow$ FALSE
        break
      $j \leftarrow j+1$
    if $iLoop =$ TRUE **then**
      delete $V'[h+i]$
    $i \leftarrow i+1$
  $h \leftarrow h+i$
return V'

V' corresponding to the object boundary. The process is repeated for all such maximal subsequences in succession to obtain a reduced set of straight edges corresponding to the object boundary.

4.1. Approximation by the differential Farey index.

There are several approximation criteria available in the literature (Bhowmick and Bhattacharya, 2007; Rosin and West, 1988; 1995; Wall and Danielsson, 1984). As we use the AFT for polygonal approximation, we introduce here the notion of a *differential Farey index*. For ranks r_1 and r_2 of two fractions in F_n, it is defined as

$$\delta_n(r_1, r_2) = \begin{cases} \Delta r_{21} & \text{if } \Delta r_{21} \leq 4f_n - 4, \\ 8f_n - 8 - \Delta r_{21} & \text{otherwise.} \end{cases} \quad (5)$$

Here, $\Delta r_{21} = |r_2 - r_1|$. For example, in Fig. 1, $n = 4, f_n = 7$, which gives $4f_n - 4 = 24$. So, for $r_1 = 1$ and $r_2 = 25$, we get $\delta_n(r_1, r_2) = 24$; as r_2 increases to 26, we have $|r_2 - r_1| = 25$, and so $\delta_n(r_1, r_2) = 48 - 25 = 23$.

While checking the approximate collinearity of a subsequence of straight edges coming from the set of straight edges E, their slopes are considered ranks of the corresponding fractions stored in $\mathbf{F}_n$. These ranks are used to compute their differential Farey indices. Algorithm 3 shows the steps of the proposed algorithm, POLY-APPROX-FT. It takes the set V that stores m endpoints defining the sequence of edges. The parameter τ denotes the *approximation parameter*, and it signifies the amount of approximation. A high value of τ provides a loose approximation, whereas a low value gives a tight approximation,

Algorithm 3 has a **for** loop containing two nested **while** loops. Each iteration of the **for** loop finds a maximal subsequence of straight edges that are approximately collinear (within a tolerance τ). In particular, if $\langle e_h, e_{h+1}, \ldots, e_{h+i} \rangle$ is the maximal subsequence, then the distance of each vertex v_{h+j}, $1 \leq j \leq i$, from the straight line segment $\overline{v_h v_{h+i+1}}$ is at most τ. Note that the vertex v_{h+j} is shared by the edges e_{h+j-1} and e_{h+j}. For each straight line segment pq with endpoints $p(x_p, y_p)$ and $q(x_q, y_q)$, the rank of the fraction $\frac{y_q - y_p}{x_q - x_p}$ is obtained from $\mathbf{F}_n$ using single memory access.

Each iteration of the inner **while** loop computes the deviation (Euclidean distance) of a vertex v_{h+j} from the straight line segment $\overline{v_h v_{h+i+1}}$. This computation is done using the differential Farey index obtained from the ranks corresponding to the slopes of $\overline{v_h v_{h+i+1}}$ and $\overline{v_{h+i+1} v_{h+j}}$, as shown in Fig. 5. This differential Farey index is treated as a rank and its corresponding fraction, shown as $\frac{a}{b}$ in Fig. 5, is obtained from F_n. Consequently, the deviation of v_{h+j} from $\overline{v_{h+j} v_{h+i+1}}$ is estimated as

$$d_2(v_{h+j}, \overline{v_h v_{h+i+1}}) \approx \frac{a \cdot \|\overline{v_{h+j} v_{h+i+1}}\|}{\sqrt{a^2+b^2}}, \quad (6)$$

where $\|\overline{v_{h+j} v_{h+i+1}}\|$ denotes the length of $\overline{v_{h+j} v_{h+i+1}}$.

The inner **while** loop executes for j varying from 1 to i. If the deviation lies within the threshold τ for each vertex v_{h+j}, then $\langle e_h, e_{h+1}, \ldots, e_{h+i} \rangle$ is considered to be approximately straight; to include the next edge in this sequence for its maximality, this loop is again executed with the value of i incremented to $i+1$ in the outer **while** loop. If the deviation exceeds the threshold τ for some vertex v_{h+j} in the inner **while** loop, then the previous solution remains the maximal solution, and the algorithm finds the next maximal subsequence with initialization from the **for** loop.

The inner **while** loop runs in a time linear in the number of vertices from v_{h+1} to v_{h+1}. Hence the overall time complexity obtained by summing up this runtime becomes quadratic in the total number of input vertices. However, as mentioned earlier, the input sequence of vertices is obtained by extraction of the ADSS from a digital curve. If the digital curve consists of M pixels, then the input sequence of m vertices is obtained in $O(M)$ time (Bhowmick and Bhattacharya, 2007). If m is asymptotically smaller than M, then the overall time complexity becomes asymptotically less than $O(M^2)$.

Table 2. Comparative study on existing polygonal approximation algorithms (listed chronologically) and the proposed algorithm.

Principle	Non-Euclidean	Complexity	Error control	Non-recursive
Normalization of area deviation (Wall and Danielsson, 1984)	No	Low	Yes	No
Curvature estimation and non-maxima suppression (Teh and Chin, 1989)	No	High	No	No
Dominant point detection using trigonometric operations (Ray and Ray, 1992)	No	High	No	No
Particle swarm optimization (Yin, 2004)	No	High	Yes	No
Genetic algorithm (Sarkar *et al*., 2004)	No	High	No	No
Digital straightness properties and area deviation (Bhowmick and Bhattacharya, 2007)	Yes	Low	Yes	Yes
Curvature estimation based on local integral square error (Chung *et al*., 2008)	No	High	Yes	No
Chain code and removal of dominant points (Masood, 2008)	No	High	Yes	No
Statistical and geometric properties (Dinesh and Guru, 2009)	No	High	Yes	No
Curvature estimation and adaptive fitting (Parvez and Mahmoud, 2010)	No	High	Yes	No
Maximal blurred segments (Nguyen and Debled-Rennesson, 2011)	No	Low	Yes	No
Graph-theoretic tools and elementary geometry (Koutroumbas, 2012)	No	Low	Yes	No
Reverse engineering on Bresenham's algorithm (Ray and Ray, 2013)	Yes	Low	No	Yes
Proposed algorithm: digital straightness properties and differential Farey index	Yes	Low	Yes	Yes

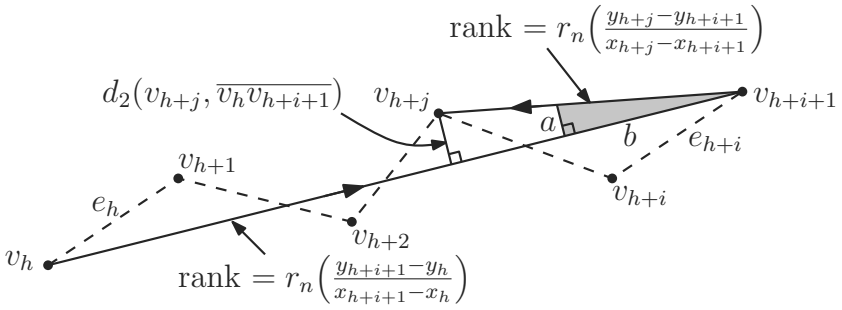
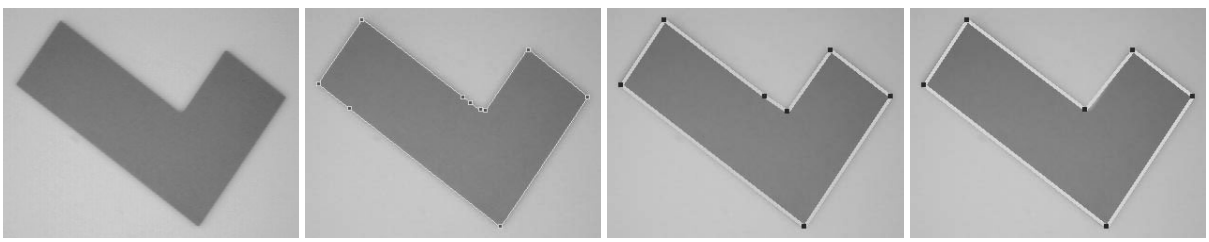


Fig. 5. Estimation of the deviation of a point v_{h+j} from $\overline{v_h v_{h+i+1}}$ using the differential Farey index.



(a) (b) (c) (d)

Fig. 6. Results using the proposed algorithm: input image (a), straight edges extracted (b), polygonal approximation: $\tau = 1 : m' = 7$ (c), polygonal approximation: $\tau = 2 : m' = 6$ (d).

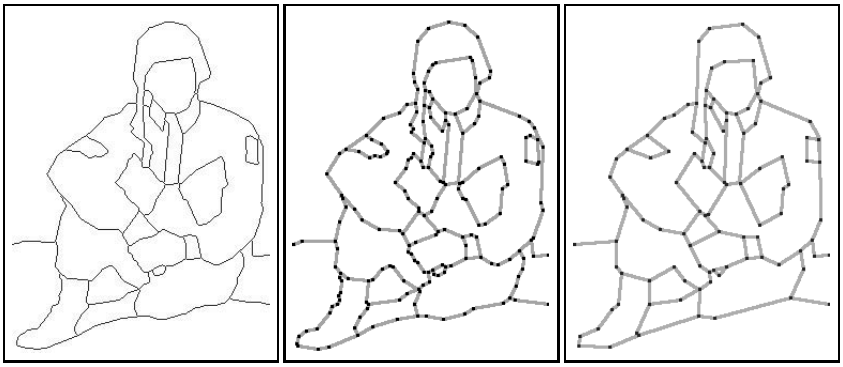
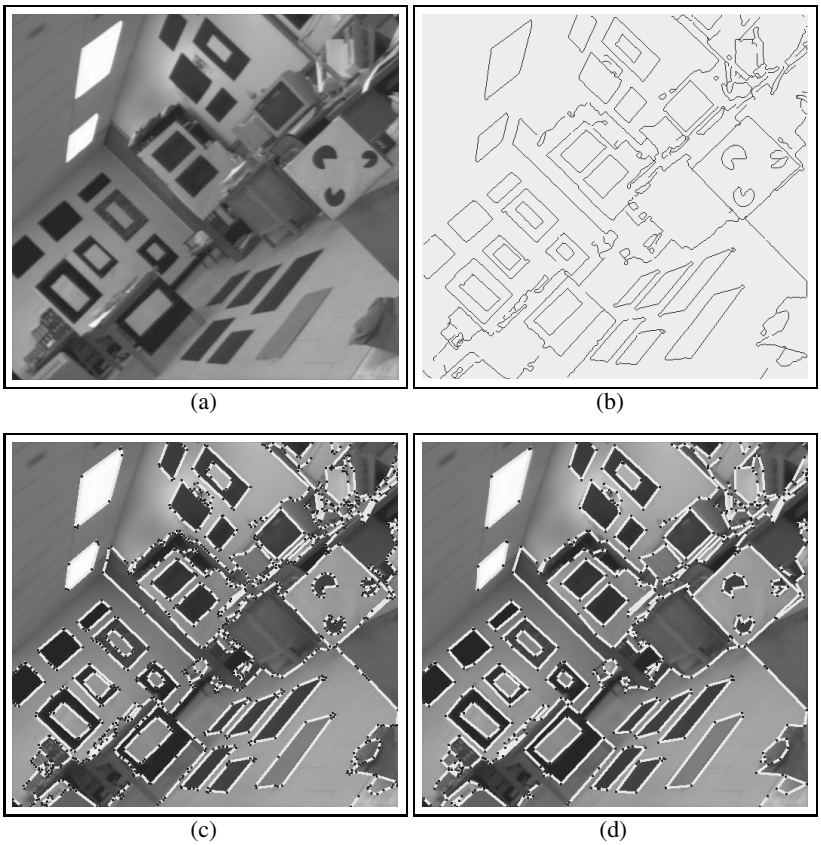


Fig. 7. Results on `soldier` (left) for $\tau = 1$ (middle) and $\tau = 4$ (right).



(a) (b) (c) (d)

Fig. 8. Polygonal approximation for the image `lab`, using the proposed algorithm (after detection of thin edge). Input image: `lab` (512×484 pixels) (a), edge map extracted using the approach of Pratihar and Bhowmick (2009), $M = 9349$ (b), $\tau = 1$, $m' = 1080$ (c), $\tau = 4$, $m' = 561$ (d).

For example, if $m = O(\sqrt{M})$, then the overall time complexity is linear in M. In general, the actual runtime would depend on the amount of variation in direction while traversing the input curve.

4.2. Results of polygonal approximation. We have implemented our algorithm in C in Ubuntu 12.04, Linux Kernel 3.2.0-54-generic 64-bit, Intel® Core™ i5-2310 CPU 2.90 GHz. Unless mentioned otherwise, all the results shown in this paper are obtained for $n = 200$. The significance of τ in our algorithm is evident from a set of experimental results shown in Fig. 6. The number of edge points, the number of vertices before approximation, and the number of vertices in the polygon or polychain after approximation are denoted here by M, m, and m', respectively. For $\tau = 1$, we get a tight approximation with $m' = 7$, whereas for $\tau = 2$, we have more relaxation with $m' = 6$. We notice the same pattern for other images , too, such as the ones shown in Figs. 7 and 8.

Table 3. Summary of results for some images.

			Existing algorithm (Bhowmick and Bhattacharya, 2007) m'					Proposed algorithm m'				
Image name	M	m	$\tau=1$	2	3	5	8	$\tau=1$	2	3	5	8
soldier	2494	302	209	156	120	90	73	202	143	115	87	72
bird-children	2813	267	166	115	93	76	69	163	107	88	72	66
crocodile	1123	100	67	49	35	28	18	66	48	34	27	17
spider	1760	135	78	69	52	42	38	78	66	50	42	38
diver	1719	153	103	74	55	40	30	101	67	55	38	27
drum-men	2835	374	257	178	144	112	92	240	170	135	104	88
vase	5755	576	438	322	266	215	181	414	308	258	209	174
India-map	2535	566	444	297	198	133	96	399	245	181	119	86
House	7037	349	275	192	166	145	131	261	192	163	145	132

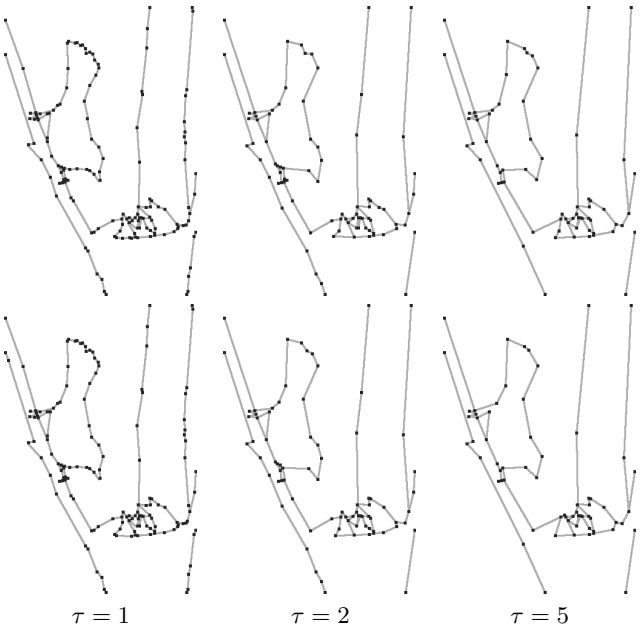


Fig. 9. Results of polygonal approximation on bird-children: Bhowmick and Bhattacharya (top row), proposed method (bottom row).

Algorithm 3 works with the differential Farey index, which is computed from the ranks of two fractions in $\mathbf{F}_n$. We have already shown an illustration of $\mathbf{F}_n$ in Fig. 1 for an order of $n = 4$, which has $f_n = 7$ and contains 48 fractions in total. As the order n becomes large, the number of fractions increases quadratically. For example, the total number of fractions in $\mathbf{F}_n$ is 24352 for $n = 100$, and 97856 for $n = 200$. These slopes/fractions divide the space of 360° into angular divisions, which are approximately, but not exactly, equal. However, with an increasing value of n, the disparity of these angular divisions decreases, as the number of divisions increases quadratically. For example, $n = 100$ means 24352 slopes and hence 24352 angular divisions, making each division approximately 0.014783°, whereas $n = 200$ means 97856 slopes, making each division approximately 0.003679°.

If some straight line segment has a slope a/b whose numerator (or/and denominator) is greater than the order n of $\mathbf{F}_n$, then the fraction would not be present in $\mathbf{F}_n$. Hence, its approximate rank is estimated as the rank of its closest fraction a'/b' in $\mathbf{F}_n$, using Algorithm 2, as explained in Section 3.

Table 3 shows results for some digital curves (two of them are shown in Figs. 7 and 9). It can be noticed from these results that, as τ increases, the number of vertices in the approximate polygon P gets reduced. For example, for the image soldier with $M = 2494$ and $m = 302$, we get $m' = 202, 143, 115, 87, 72$, for $\tau = 1, 2, 3, 5, 8$, using our algorithm. These results are quite similar to the ones obtained by the algorithm proposed by Bhowmick and Bhattacharya (2007), which produces $m' = 209, 156, 120, 90, 73$ for the same values of τ.

For an appropriately large value of τ, our algorithm is found to be stable with respect to rotation. This is evident from the set of results given in Fig. 10. Notice that the sequence of straight edges describing the rotated object remains almost same as that of the original one, especially when $\tau = 8$. For $\tau < 8$, there are some variations due to changes in the point sequence defining the edges, which occurred during rotation.

Quality of approximation. In our algorithm, the parameter τ provides a control over the approximation. Thus, the *compression ratio*, $\rho = m'/m$, is taken as a measure of approximation quality. This measure is also used in most of the related work (Bhowmick and Bhattacharya, 2007; Masood, 2008; Teh and Chin, 1989; Yin, 2004). As the control parameter τ increases, ρ decreases. As evident from Table 3, the compression ratio ρ reported by our algorithm complies with the results of Bhowmick and Bhattacharya (2007).

Another measure of approximation quality is defined as the maximum deviation of a point (x, y) lying on a

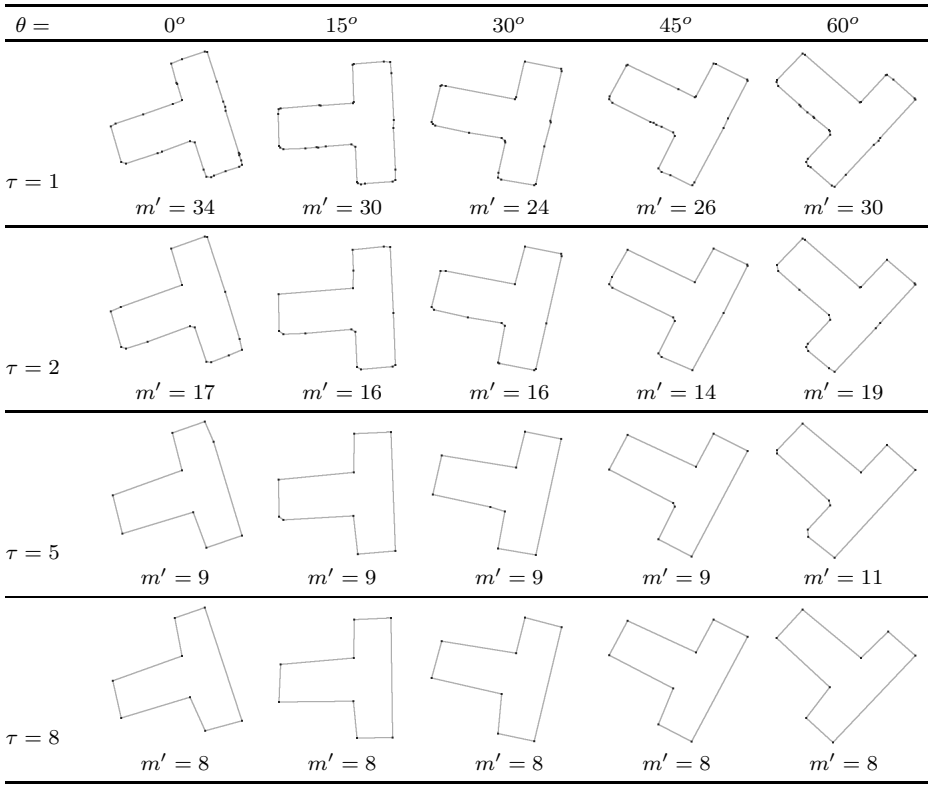


Fig. 10. Results on rotated `wdg3` for various τ.

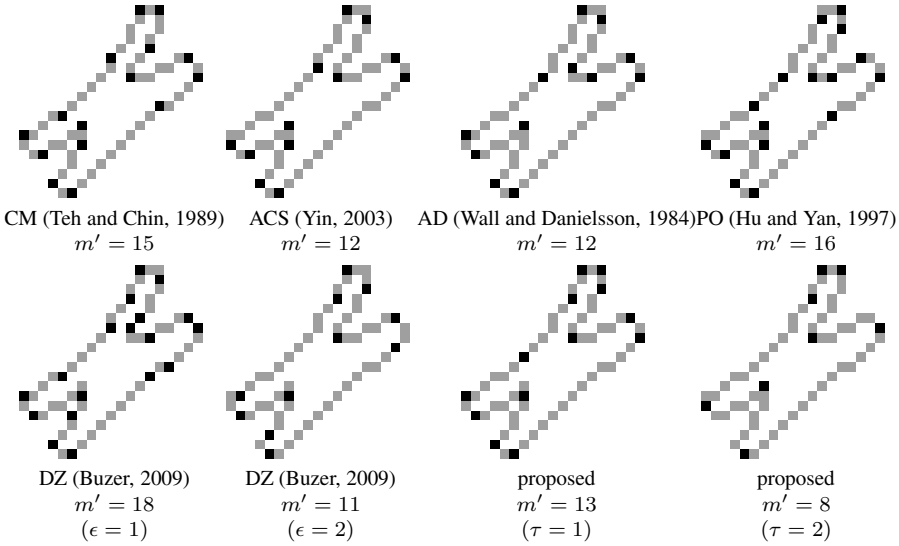


Fig. 11. Results of polygonal approximation obtained by existing algorithms and the proposed algorithm for the image `chromosome`.

straight edge $e \in E$ from the corresponding edge of the approximate polygon, P. If $p' = (x', y')$ is the closest point on an edge of P corresponding to a point $p = (x, y) \in e$, then the deviation of p from p' is given as

$$d_\infty(p, p') = \max\{|x - x'|, |y - y'|\}. \tag{7}$$

The total deviation of all the edge points of E is therefore

$$\delta_E = \sum_{e \in E} \sum_{p \in e} d_\infty(p, p'). \tag{8}$$

The value $d_\infty(p, p')$ depends on the provided value

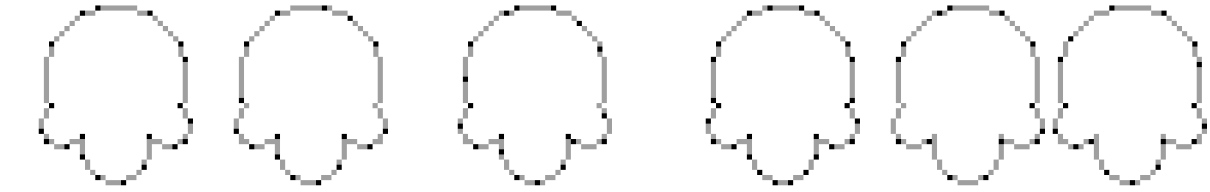


Fig. 12. Results of polygonal approximation obtained by existing algorithms and the proposed algorithm for the image `semi-circle` ($\epsilon = 1$ for DZ and $\tau = 1$ for the proposed).

Table 4. Comparative results on the number of operations for the image `soldier` for the polygonization part. (mul., add., comp., mem., assn., inr./dcr., AND denote the respective numbers of multiplications, additions, comparisons, memory accesses, assignments, increments/decrements, and logical `AND` operations.)

				Number of operations						
	τ	m	m'	mul.	add.	comp.	mem.	assn.	inr./dcr.	AND
Bhowmick and Bhattacharya (2007)	1	302	209	2296	2435	1767	0	2890	664	295
	2	302	156	2751	2707	1844	0	2979	729	242
	3	302	120	3178	2976	1930	0	3090	790	206
	4	302	99	3640	3285	2041	0	3246	856	185
	5	302	90	3871	3441	2098	0	3327	889	176
	6	302	82	4186	3658	2180	0	3446	934	168
	7	302	76	4382	3792	2230	0	3518	962	162
	8	302	73	4606	3949	2291	0	3608	994	159
Proposed method	1	302	202	332	2116	1932	582	3966	668	288
	2	302	143	409	2365	2181	659	4387	745	229
	3	302	115	459	2537	2353	709	4681	795	201
	4	302	95	528	2793	2609	778	5124	864	181
	5	302	87	563	2925	2741	813	5353	899	173
	6	302	79	602	3073	2889	852	5610	938	165
	7	302	74	645	3240	3056	895	5901	981	160
	8	302	72	672	3346	3162	922	6086	1008	158

of τ, and the average deviation is measured as

$$\overline{\delta} = \frac{\delta_E}{M}, \tag{9}$$

where M denotes the total number of pixels in E. Again, for the measure of the *error frequency*, for a chosen τ, we find the number of edge points in P deviated from the edges of E by a distance $\delta(= 0, 1, 2, 3, \ldots)$. Then the error frequency is defined as

$$\nu(\tau, \delta) = \frac{1}{M} \sum_{e \in E} |\{p \in e : d_\infty(p, p') = \delta\}|, \tag{10}$$

where the variation of $\nu(\tau, \delta)$ versus τ and δ provides the error distribution for polygonal approximation.

Since the focus of our work is on the AFT and its applicability to practical problems, we have compared our algorithm with one of those proposed by Bhowmick and Bhattacharya (2007), which has a fast execution owing to integer operations. Plots for *average deviation* caused by approximation are shown in Fig. 13 for the images `soldier` and `bird-children`. It can be noticed from these plots that the output produced by the proposed algorithm complies with the one produced using the algorithm given by Bhowmick and Bhattacharya (2007). Comparative results on the error frequency for these two images are also shown in Figs. 14 and 15. It yields high frequency for low deviation, and low frequency for high deviation, which indicates its qualitative merit in the process of approximation. Table 4 shows a comparative study on the number of operations required for polygonal approximation of `soldier` image. It can be noticed in this table that, in comparison with the algorithm of Bhowmick and Bhattacharya (2007), the number of multiplications is significantly lower in the proposed algorithm. However, the proposed algorithm needs memory access for accessing ranks from the AFT. Also, it needs some more comparisons and assignments compared with that of Bhowmick and Bhattacharya (2007). As the number of multiplications is reduced significantly, we notice a marginal improvement in CPU time (see Table 5).

Also, we have compared the performance of our method with that of some of the existing methods. These are *curvature maximization* (CM (Teh and Chin, 1989)),

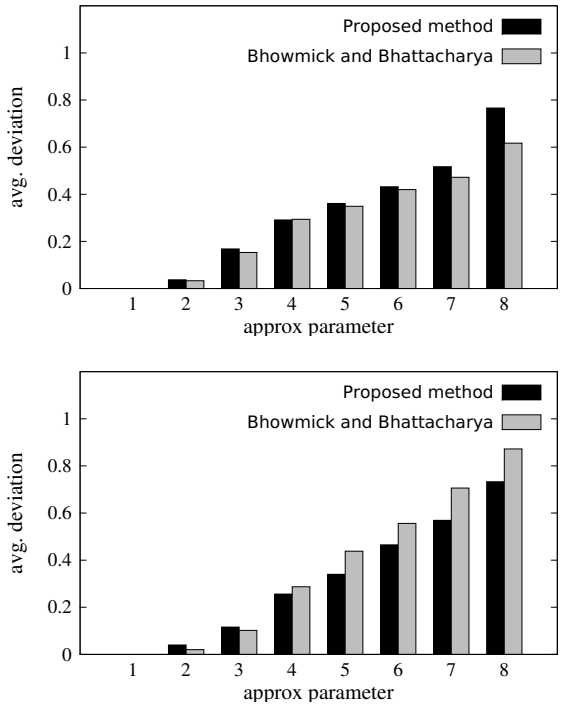


Fig. 13. Average deviation for various values of approximation parameter (τ) for the image `soldier` (top) and `bird-children` (bottom).

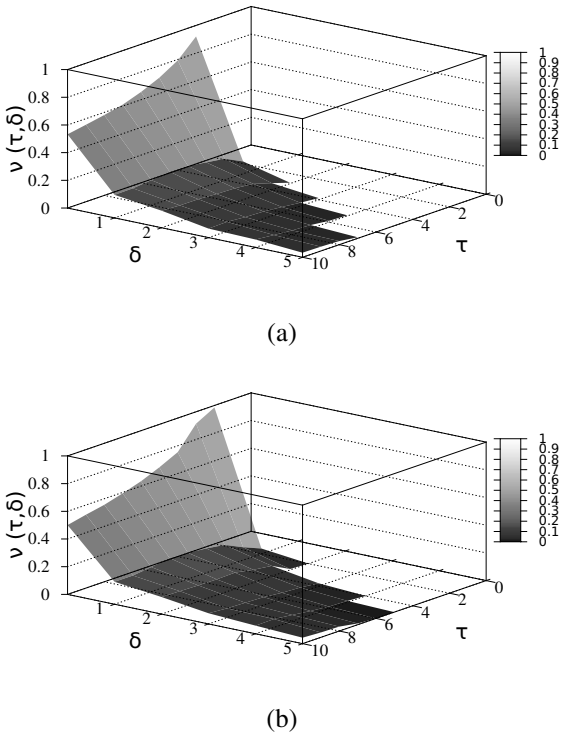


Fig. 14. Error frequency distribution (ν versus τ and δ) for the image `soldier`: algorithm of Bhowmick and Bhattacharya (2007) (a), proposed algorithm (b).

ant colony search (ACS (Yin, 2003)), *area deviation* (AD (Wall and Danielsson, 1984)), *perceptual organization* (PO (Hu and Yan, 1997)), and *digital zone criteria* (DZ (Buzer, 2009)). Results obtained for these methods and the proposed approach for two benchmark curves, namely, `chromosome` and `semi-circle`, are presented in Figs. 11 and 12, respectively. It is evident from these that the sequence of vertices reported by the proposed method for $\tau = 1$ closely matches those obtained by the existing methods.

4.3. Order of the AFT. For this application, we have taken Farey order $n = 200$. Figure 1 shows the AFT, $\mathbf{F}_4$, of order 4, where 48 possible slopes exist. As the order n of $\mathbf{F}_n$ becomes large, the number of slopes increases appreciably. For example, for $n = 200$, the number of fractions in the first quadrant is 24465. Hence, the total number of fractions in $\mathbf{F}_{200}$ is $4 \times 24465 = 97856$. These 97856 slopes (fractions) partition the space of 360° into 97856 divisions. Theoretically, not all the divisions will be of an equal degree. But as the number of divisions is quite large, two ranks in a vicinity represent nearly equal slopes.

For a slope not in $\mathbf{F}_n$, its closest fraction is computed and used subsequently. Results are shown for different values of n in Fig. 16 and Table 5 for the image `soldier`. It is evident from these that RFT-based polygonal approximation always preserves the overall shape of the object irrespective of the value chosen for n. Although the sequence of vertices for a very small value of n (e.g., $n = 10$) is not the same as that for a larger value, it becomes stable once n has a sufficiently large value (e.g., $n = 50$).

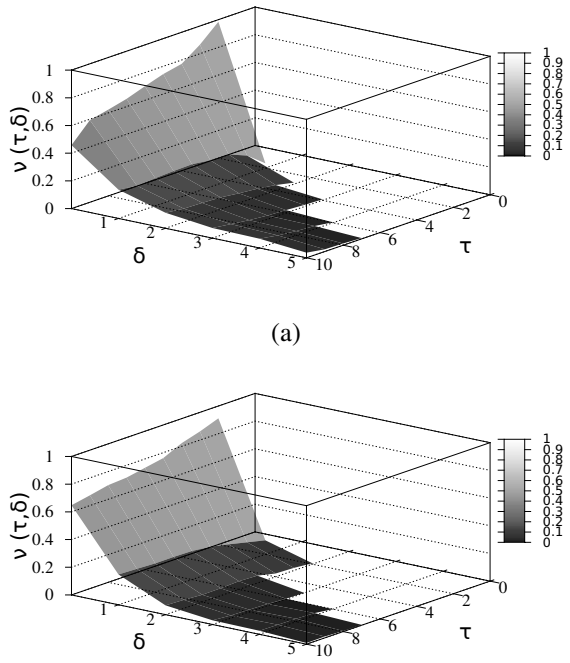
5. Application to skew correction

A skew creeps in unknowingly in a document image during its scanning due to an improper placement of the document page on the scanner bed, or due to some technical malfunction or limitations of the acquisition mechanism. Hence, efficient detection of the amount of the skew is the very first step of preprocessing for effective character and graphics recognition, document layout analysis, etc.

Since the 1990s, several works have been done on skew correction of document images with varying accuracy and variegated techniques. These are mostly based on the least-squares method, median gradient, variance of black-white transition, Hough transform, etc. (Jiang *et al.*, 1997; Li *et al.*, 2007; Pavlidis and Zhou, 1991; Yan, 1993; Yu and Jain, 1996). Later, other techniques have been proposed based on the textual/non-textual classifier, line fitting, morphological

Table 5. Effect of n on the result obtained by the algorithm for the image `soldier`. (CPU time is shown in seconds for the polygonization part.)

Method	m	$\tau = 2$		$\tau = 5$		$\tau = 8$	
		m'	CPU time	m'	CPU time	m'	CPU time
Proposed							
$n = 10$	302	160	0.000083	96	0.000058	78	0.000051
$n = 20$	302	152	0.000075	90	0.000046	75	0.000044
$n = 30$	302	145	0.000070	89	0.000046	73	0.000042
$n = 50$	302	146	0.000064	87	0.000044	72	0.000050
$n = 100$	302	144	0.000063	87	0.000047	72	0.000051
$n = 150$	302	145	0.000062	87	0.000046	72	0.000051
$n = 200$	302	143	0.000061	87	0.000045	72	0.000055
Existing algorithm (Bhowmick and Bhattacharya, 2007)	302	156	0.000094	90	0.000071	73	0.000068



(a)

(b)

Fig. 15. Error frequency distribution (ν versus τ and δ) for the image `bird-children`: algorithm of Bhowmick and Bhattacharya (2007) (a), proposed algorithm (b).

operations, convex hull, etc. (Cao *et al*., 2003; Chaudhuri and Pal, 1997; Chou *et al*., 2007; Das and Chanda, 2001; Yuan and Tan, 2007).

All the above methods mostly work in the real/Euclidean domain and are based on trigonometric procedures. The algorithm proposed here, in contrast, is based on the AFT. It detects the skew directly from the gray-scale image by extracting and processing *digitally straight* edges, without resorting to any preprocessing like binarization or thinning. These straight edges

Algorithm 4. SKEW-DETECT-FT.

Input: E, $\mathbf{F}_n^{(1)}$, $\overline{F}_n$, w
Output: Skew slope

$n_B = \lceil \frac{1}{w}|\overline{F}_n| \rceil$
for $i \leftarrow 0, 1, \ldots, n_B$ **do**
  $B[i].len \leftarrow 0$, $B[i].wrank \leftarrow 0$
  $a_i \leftarrow w \cdot i$, $b_i \leftarrow \min\{a_i + w - 1, |\overline{F}_n|\}$ ▷ $[a_i, b_i]$ = range of ranks in $B[i]$
for *each* $e_j(v = (x, y), v' = (x', y')) \in E$ **do**
  $dy = |y - y'|, dx = |x - x'|$
  $l_j \leftarrow \max\{dy, dx\}$
  $\rho_j \leftarrow \mathbf{F}_n^{(1)}[dy][dx]$
  $i \leftarrow \lceil \frac{1}{w}\rho_j \rceil$
  $B[i].len \leftarrow B[i].len + l_j$
  $B[i].wrank \leftarrow B[i].wrank + l_j \cdot \rho_j$
select the principal bin $B[s]$ ▷ having maximum len
$\rho_w \leftarrow B[s].wrank/B[s].len$ ▷ Eqn. (11)
$p_s/q_s \leftarrow \overline{F}_n[\rho_w]$
return p_s/q_s

are extracted from the line boundaries of digitized engineering drawings, using *periodic properties* of *digital straightness* (Klette and Rosenfeld, 2004) and *exponential averaging* (Pratihar and Bhowmick, 2009). Subsequently, these straight edges are analyzed to derive the *principal axes* (skewed vertical and skewed horizontal axes) of the drawing, needed to estimate the skew. The analysis is done using the ranks of fractions corresponding to the slopes of the edges. An illustration of our algorithm is shown in Fig. 17.

5.1. Binning of straight edges and skew detection. While obtaining the set of straight edges, $E := \{e_1, e_2, \ldots, e_m\}$, we also store their Farey ranks and lengths. The length of an edge $e(v, v')$, where $v = (x, y)$ and $v' = (x', y')$, is measured as the L_∞ norm (Klette

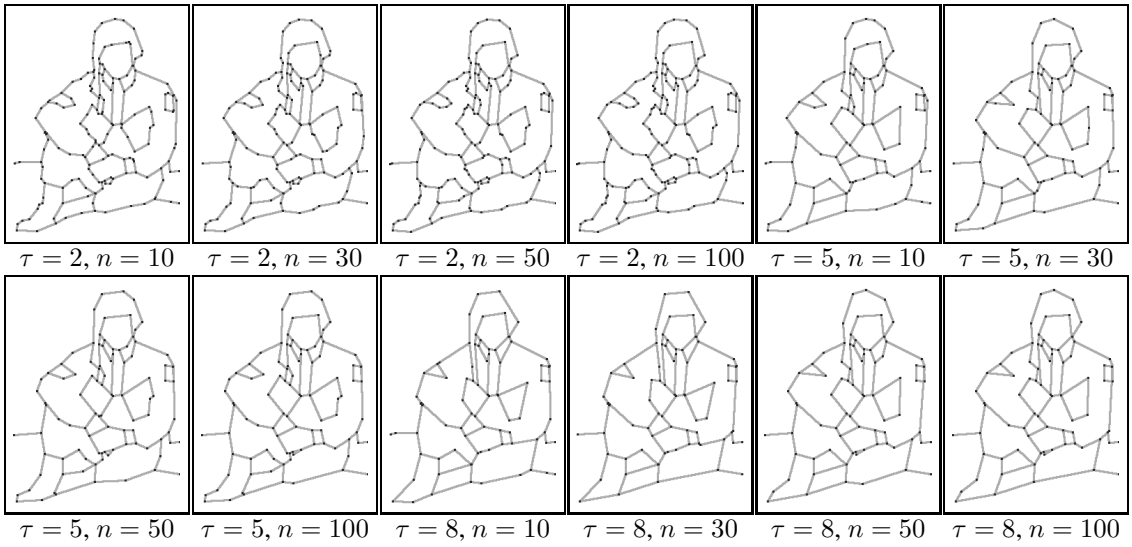


Fig. 16. Results of polygonal approximation for different values of the Farey order, n.

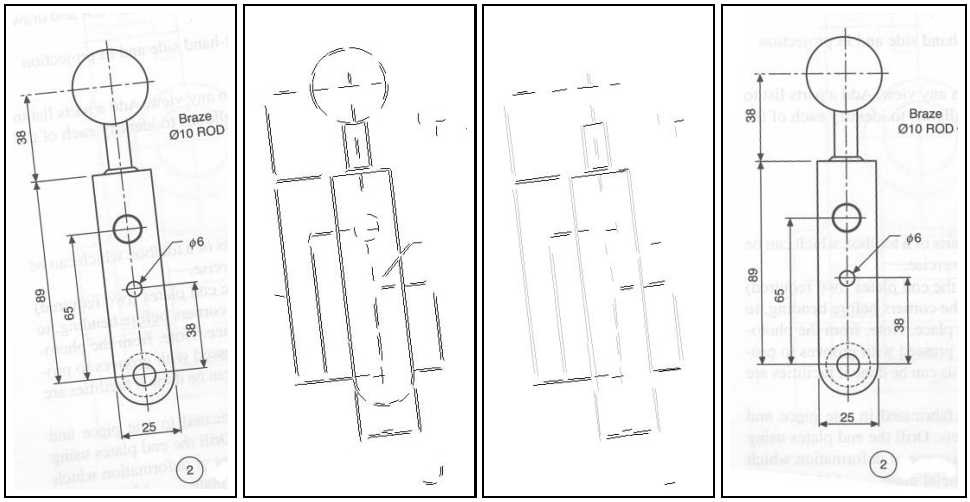


Fig. 17. Skew correction—step-wise snapshots of our algorithm: input (a), detected straight edges defining boundaries (b), principal and its orthogonal direction shown in gray and in black, respectively (c), deskewed image (d).

and Rosenfeld, 2004), and so it is given by $\|e\|_\infty = \max\{|x - x'|, |y - y'|\}$. If the length of a straight edge exceeds a threshold value, $l_{\min}$, then it is included in E. The value of $l_{\min}$ is set as twice the average font height. The rank of each edge e is obtained from $\mathbf{F}_n$ by single memory access, which saves time to a significant amount in comparison with slope-angle computation by trigonometric procedures linked to the math library.

Algorithm 4 shows the basic steps. We consider a sequence B of $n_B = \left\lceil \frac{1}{w}|\overline{F}_n| \right\rceil$ bins, each bin $B[i]$ corresponding to a range of ranks, namely, $[a_i, b_i]$, where $|\overline{F}_n|$ = is the number of fractions in $\overline{F}_n$, $a_i = w \cdot i$ and $b_i = \min\{a_i + w - 1, |\overline{F}_n|\}$, so that each bin has a fixed width, w. In our experimentation, we have taken $w = 100$ and the order of $\overline{F}_n$ as $n = 200$, whence $|\overline{F}_n| = 97856$. Accordingly, two consecutive fractions will approximately correspond to a difference of $360°/97856 = 0.003679°$, which is practically a very small value. This fact has been exploited in binning. Since there are $w = 100$ consecutive ranks in each $B[i]$, they approximately span over $0.003679 \times 100 = 0.368°$, which bounds the maximum possible binning error.

The entry of each bin $B[i]$ is initialized as 0. For each edge $e_j \in E$ having rank $\rho_j \in [a_i, b_i]$ and length l_j, the entry of $B[i]$ is increased by l_j. Hence, after processing all edges of E, each bin $B[i]$ contains the sum of lengths of all straight edges in E whose ranks lie in $[a_i, b_i]$.

Finding the principal bin. We take the bin $B[i_1]$ with the maximum sum of lengths, and check whether there exists a bin $B[i_1']$ such that $n_B - 1 \leq |i_1 - i_1'| \leq n_B + 1$ and the sum stored in $B[i_1']$ is a local maximum in B. If so, then $B[i_1]$ is the principal bin which provides the principal direction, and $B[i_1']$ provides the direction orthogonal to that of $B[i_1]$. Otherwise, we iteratively consider the bin $B[i_2]$ containing the next maximum and check the existence of $B[i_2']$ in a similar way. Within three iterations, if we get a solution. Then the skew is reported and the original image is de-skewed based on the estimated angle of the skew, as explained next; otherwise, no skew is reported.

Let $B[s]$ be the principal bin that corresponds to the skew and $B[s']$ the one with orthogonal direction, as explained above. Since a_s and $b_s (= a_s + w - 1)$ are the minimum and maximum ranks corresponding to $B[s]$, we consider the weighted mean of the ranks in $B[s]$ to estimate the resultant skew. The weight of a rank $\rho_s \in [a_s, b_s]$ is taken as the sum of all straight edges in E having rank ρ_s. Hence, if e_{sj} is the j-th edge in E having rank $\rho(e_{sj}) = \rho_s$, then the resultant skew of the image concerned is estimated as $\frac{p_s}{q_s}$, where

$$\mathbf{F}_n^{(1)}[p_s][q_s] = \left(\frac{\displaystyle\sum_{\substack{\rho(e_{sj})=\rho_s \\ a_s \leq \rho_s \leq b_s}} \|e_{sj}\|_\infty \rho_s}{\displaystyle\sum_{\substack{\rho(e_{sj})=\rho_s \\ a_s \leq \rho_s \leq b_s}} \|e_{sj}\|_\infty} \right). \qquad (11)$$

5.2. De-skewing matrix. For de-skewing, we do not have to compute the skew angle. Instead, we form the rotation matrix from p_s/q_s. The orientation of the principal direction can either be along the x-axis or along the y-axis, depending on whether $|p_s| < |q_s|$ or $|p_s| > |q_s|$. Thus, finally, two cases arise from the orientation of the principal direction. If the principal direction found is along the x-axis, then the rotation matrix for de-skewing is formed as

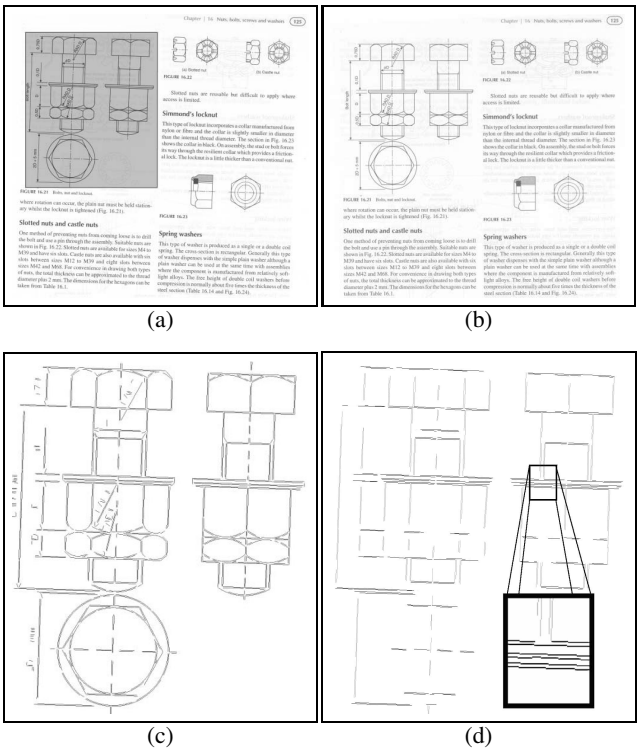
$$\begin{bmatrix} q_s/t_s & -p_s/t_s \\ p_s/t_s & q_s/t_s \end{bmatrix} = \frac{1}{t_s} \begin{bmatrix} q_s & -p_s \\ p_s & q_s \end{bmatrix},$$

and when it is along the y-axis, then it is formed as

$$\begin{bmatrix} p_s/t_s & -q_s/t_s \\ q_s/t_s & p_s/t_s \end{bmatrix} = \frac{1}{t_s} \begin{bmatrix} p_s & -q_s \\ q_s & p_s \end{bmatrix}.$$

Here, $t_s = \sqrt{{p_s}^2 + {q_s}^2}$, which is computed only once.

5.3. Results of skew correction. For the 300-dpi image shown in Fig. 18, we get 3962 straight edges. Distribution of the ranks corresponding to these edges is shown in Table 6 and in Fig. 19. As there are 97856 fractions in $\overline{F}_n$ spanning over 360°, half (48928) of them cover the range $[0, 180°)$ (see Fig. 19). As considering $\theta \geq 180°$ in an anticlockwise sense is equivalent to $360° - \theta$ in a clockwise sense, we use these 48928 ranks only. The bin width $w = 100$ makes $n_B = 490$. It is evident from the rank distribution in Table 6 and Fig. 19 that the bin $B[485]$ with the rank interval $[48500, 48599]$ has the maximum sum, 5239, contributed by 53 edges. The weighted mean of the ranks of these 53 edges, as estimated by Eqn. (11), is found to be 48549.251, which rounds off to 48549, and corresponds to the fraction $\frac{-7}{197}$, which is used for de-skewing. The principal bin $B[485]$ is verified by the existence of $B[240]$, since the bin indices differ by $485 - 240 = 245$, which is within $n_B/2 \pm 1$ (Section 5.1). Figures 17 and 18 show the intermediate steps and the final outputs after skew corrections. It is clear from these results that the proposed technique is quite efficient in detecting the skew in a digitized document image that contains engineering drawings, or a mix of text and drawings, or tabular structures bordered by straight lines. Owing to primitive operations in the integer domain and a linear-time binning procedure, the skew detection algorithm runs significantly fast with the desired level of precision. Experimental results on different datasets demonstrate its elegance, efficiency, and robustness.



(a) (b) (c) (d)

Fig. 18. Result on a document page with a mix of text and graphics: input image (a), de-skewed image (b), straight edges in E (cropped and zoomed from (a)) (c), principal direction (shown in gray) and orthogonal direction (shown in black) (d).

Table 6. Distribution of straight edges found for the image of Fig. 18, in $B[0], B[1], \ldots, B[489]$, each bin being of width $w = 100$. Bin $B[485]$ corresponds to the principal direction and Bin $B[240]$ to the orthogonal one (highlighted in gray).

rank bin	#lines	Σ len.
0– 99	1	22
300– 399	2	66
600– 699	1	37
2300–2399	1	53
2500–2599	1	57
2900–2999	1	45
3200–3299	1	41
⋮		

rank bin	#lines	Σ len.
⋮		
23700–23799	16	724
23800–23899	12	833
23900–23999	38	2768
24000–24099	50	**5026**
24100–24199	40	3398
24200–24299	8	441
24300–24399	1	75
⋮		

rank bin	#lines	Σ len.
⋮		
48100–48199	10	400
48200–48299	18	782
48300–48399	18	1098
48400–48499	30	2457
48500–48599	53	**5239**
48600–48699	37	2827
48700–48799	1	57

5.4. Comparison with the existing methods. Most of the existing methods for document skew estimation are based on the Hough transform (HT) (Amin and Fischer, 2000; Chaudhuri and Pal, 1997; Hinds *et al*., 1990; Manjunath *et al*., 2006; Le *et al*., 1994; Singh *et al*., 2008; Srihari and Govindraju, 1989; Yin, 2001; Yu and Jain, 1996). Hence, we have compared the accuracy and computation time of our AFT-based method with an HT-based one (see Chaudhuri and Pal, 1997, Algorithm A2). Although the latter is usually robust, it is computationally expensive. Recent methods, therefore, consider a reduced point set to be input to the Hough transform in order to reduce the computation time (Chaudhuri and Pal, 1997; Hinds *et al*., 1990; Manjunath *et al*., 2006; Le *et al*., 1994; Yin, 2001; Yu and Jain, 1996). To demonstrate the effectiveness of our method, we have compared it with the Hough transform working on a reduced input set comprising the edge pixels of the straight line segments from the upper envelope of the connected components (Chaudhuri and Pal, 1997). The edge points constituting the straight edges belonging to the principal bin are regarded as the reduced input set to the HT-based method under our consideration. This saves a significant amount of time while finding the peak direction.

As shown in Fig. 17, the edge points (Fig. 17(b)) comprise the input for the HT-based method. Even so, our AFT-based method shows better performance in terms of both accuracy and computation time. This is evident from the results shown in Table 7. For some manually set skew angles, the statistics (mean and standard deviation of estimated skews, with number of images $= 20$) for the proposed method and the HT-based one are presented in Table 8.

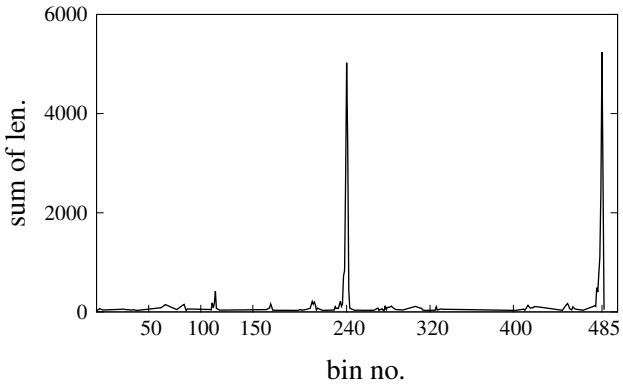


Fig. 19. Distribution of ranks of the straight edges corresponding to Fig. 18(a). The straight edges corresponding to the bins $B[240]$ and $B[485]$ are shown in black and gray in Fig. 18(d).

Table 7. CPU time (in seconds) for skew estimation by the HT-based method and the proposed one.

Image	Image Size	CPU Time	
		HT	Proposed
01	249×554	0.21	0.08
02	2256×2160	1.10	0.57
03	2546×3202	1.50	0.83
04	2240×3050	1.16	0.77
05	2260×3086	1.15	0.76
06	964×978	0.43	0.30
07	789×600	0.31	0.14
08	2380×3212	1.33	0.86
09	698×768	0.34	0.23
10	980×403	0.42	0.26

6. Concluding remarks

In this paper, we have proposed an efficient algorithm for construction of an augmented Farey table (AFT). We showed how the AFT can subsequently be used for linearity checking, slope measurement, etc., while solving various image-analytic problems like polygonal approximation of digital curves and skew correction of scanned engineering drawings.

Apart from the applications shown in this paper, an AFT can be used to speed up many other computational

Table 8. Mean and standard deviation (SD) of skew angles estimated by the HT-based method and the proposed one.

Method		Skew (Manual)				
		2°	5°	10°	30°	-5°
HT	Mean	2.14	5.17	10.21	29.84	−5.08
	SD	0.24	0.35	0.25	0.34	0.24
Proposed	Mean	1.98	5.10	10.12	30.06	−5.10
	SD	0.35	0.13	0.18	0.13	0.19

procedures. One such procedure is the de-skewing of scanned images of text documents; as shown by Pratihar *et al*. (2013), the use of AFT makes the de-skewing process significantly fast and efficient, often outweighing the performance of the existing algorithms. Another interesting application of the AFT lies in vectorization of thick digital lines (and arbitrary curves), which was reported earlier by Pratihar and Bhowmick (2010).

Based on our studies and experiments, we foresee AFT as a potential tool for various applications related to low-level processing and analysis of digital images. We end our discussion here with its possible use in the well-known problem of computing the convex hull of an object in the digital plane.

While traversing along an object boundary, 'convexity' of a vertex is decided based on the angular turn at that vertex, which is measured with respect to its preceding and succeeding vertices. The related computation can be done based on the ranks of the fractions corresponding to the incoming and the outgoing edges at the concerned vertex.

An AFT-based computation would eventually facilitate the process of computing the convex hull of the object in the digital plane. We also foresee that the AFT can be used in fractal compression (Nikiel, 2007), which is a lossy image compression technique, since fractal compression has a theoretical connection with Farey trees (Devaney, 1999).

References

Amin, A. and Fischer, S. (2000). A document skew detection method using the Hough transform, *Pattern Analysis and Applications* **3**(3): 243–253.

Attneave, F. (1954). Some informational aspects of visual perception, *Psychological Review* **61**(3): 183–193.

Bhowmick, P. and Bhattacharya, B.B. (2007). Fast polygonal approximation of digital curves using relaxed straightness properties, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **29**(9): 1590–1602.

Buzer, L. (2009). Optimal simplification of polygonal chains for subpixel-accurate rendering, *Computational Geometry* **42**(1): 45–59.

Cao, Y., Wang, S. and Li, H. (2003). Skew detection and correction in document images based on straight-line fitting, *Pattern Recognition Letters* **24**(12): 1871–1879.

Charrier, E. and Buzer, L. (2009). Approximating a real number by a rational number with a limited denominator: A geometric approach, *Discrete Applied Mathematics* **157**(16): 3473–3484.

Chaudhuri, B.B. and Pal, U. (1997). Skew angle detection of digitized Indian script documents, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(2): 182–186.

Chou, C., Chu, S. and Chang, F. (2007). Estimation of skew angles for scanned documents based on piecewise covering by parallelograms, *Pattern Recognition* **40**(2): 443–455.

Chung, K.L., Liao, P.H. and Chang, J.M. (2008). Novel efficient two-pass algorithm for closed polygonal approximation based on LISE and curvature constraint criteria, *Journal of Visual Communication and Image Representation* **19**(4): 219–230.

Cormen, T.H., Leiserson, C.E. and Rivest, R.L. (2000). *Introduction to Algorithms*, Prentice Hall of India, New-Delhi.

Das, A.K. and Chanda, B. (2001). A fast algorithm for skew detection of document images using morphology, *International Journal on Document Analysis and Recognition* **4**(2): 109–114.

Das, S., Halder, K., Pratihar, S. and Bhowmick, P. (2010). Properties of Farey sequence and their applications to digital image processing, *4th International Conference on Information Processing, Bangalore, India*, pp. 71–81.

Devaney, R.L. (1999). The Mandelbrot set, the Farey tree, and the Fibonacci sequence, *The American Mathematical Monthly* **106**(04): 289–302.

Dinesh, R. and Guru, D.S. (2009). Non-parametric adaptive approach for the detection of dominant points on boundary curves based on non-symmetric region of support, *International Journal of Image and Graphics* **9**(4): 541–557.

Graham, R., Knuth, D. and Patashnik, O. (1994). *Concrete Mathematics*, Addison-Wesley, London.

Hardy, G.H. and Wright, E.M. (1968). *An Introduction to the Theory of Numbers*, Oxford University Press, New York, NY.

Hinds, S., Fisher, J. and D'Amato, D.P. (1990). A document skew detection method using run length encoding and the Hough transform, *Proceedings of the International Conference on Pattern Recognition, Los Alamitos, CA, USA*, pp. 464–468.

Hu, H. and Yan, H. (1997). Polygonal approximation of digital curves based on the principles of perceptual organization, *Pattern Recognition* **30**(5): 701–718.

Jiang, H.-F., Han, C.-C. and Fan, K.-C. (1997). A fast approach to the detection and correction of skew documents, *Pattern Recognition Letters* **18**(7): 675–686.

Klette, R. and Rosenfeld, A. (2004). *Digital Geometry: Geometric Methods for Digital Picture Analysis*, Morgan Kaufmann, San Francisco, CA.

Knuth, D. E. (1997). *The Art of Computer Programming*, Vol. 2, 3rd Edn., Addison-Wesley, Boston, MA.

Koutroumbas, K.D. (2012). Piecewise linear curve approximation using graph theory and geometrical concepts, *IEEE Transactions on Image Processing* **21**(9): 3877–3887.

Kumar, M.P., Goyal, S., Jawahar, C.V. and Narayanan, P.J. (2002). Polygonal approximation of closed curves across multiple views, *3rd Indian Conference on Computer Vision, Graphics and Image Processing, Ahmadabad, India*, pp. 317–322.

Le, D.S., Thoma, G.R. and Wechsler, H. (1994). Automatic page orientation and skew angle detection for binary document images, *Pattern Recognition* **27**(10): 1325–1344.

Li, S., Shen, Q. and Sun, J. (2007). Skew detection using wavelet decomposition and projection profile analysis, *Pattern Recognition Letters* **28**(5): 555–562.

Liu, H., Latecki, L. and Liu, W. (2008). A unified curvature definition for regular, polygonal, and digital planar curves, *International Journal of Computer Vision* **80**(1): 104–124.

Manjunath, V.N., Kumar, G.H. and Shivakumara, P. (2006). Skew detection technique for binary document images based on Hough transform, *International Journal of Information and Communication Engineering* **3**(7): 493–499.

Masood, A. (2008). Dominant point detection by reverse polygonization of digital curves, *Image and Vision Computing* **26**(5): 702–715.

Melkman, A. and O'Rourke, J. (1988). On polygonal chain approximation, *in* G.T. Toussaint (Ed.), *Computational Morphology*, North-Holland, Amsterdam, pp. 87–95.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(10): 1615–1630.

Mokhtarian, F. and Mohanna, F. (2002). Content-based video database retrieval through robust corner tracking, *IEEE Workshop on Multimedia Signal Processing, St. Thomas, Virgin Islands, USA*, pp. 224–228.

Neumann, R. and Teisseron, G. (2002). Extraction of dominant points by estimation of the contour fluctuations, *Pattern Recognition* **35**(7): 1447–1462.

Neville, E.H. (1950). *The Farey Series of Order 1025*, Cambridge University Press, Cambridge.

Nguyen, T.P. and Debled-Rennesson, I. (2011). A discrete geometry approach for dominant point detection, *Pattern Recognition* **44**(1): 32–44.

Nikiel, S. (2007). A proposition of mobile fractal image decompression, *International Journal of Applied Mathematics and Computer Science* **17**(1): 129–136, DOI: 10.2478/v10006-007-0012-5.

O'Connell, K.J. (1997). Object-adaptive vertex based shape coding method, *IEEE Transactions on Circuits and Systems for Video Technology* **7**(1): 251–255.

Parvez, M.T. and Mahmoud, S.A. (2010). Polygonal approximation of digital planar curves through adaptive optimizations, *Pattern Recognition Letters* **31**(13): 1997–2005.

Pătraşcu, C.E. and Pătraşcu, M. (2004). Computing order statistics in the Farey sequence, *Symposium on Algorithmic Number Theory, Burlington, VT, USA*, pp. 358–366.

Pavlidis, T. and Zhou, J. (1991). Page segmentation by white streams, *International Conference on Document Analysis and Recognition, Saint-Malo, France*, pp. 945–953.

Pawlewicz, J. and Pătraşcu, M. (2009). Order statistics in the Farey sequences in sublinear time and counting primitive lattice points in polygons, *Algorithmica* **55**(2): 271–282.

Prasad, D.K., Leung, M.K.H., Quek, C. and Cho, S. (2012). A novel framework for making dominant point detection methods non-parametric, *Image and Vision Computing* **30**(11): 843–859.

Pratihar, S. and Bhowmick, P. (2009). A thinning-free algorithm for straight edge detection in a gray-scale image, *International Conference on Advances on Pattern Recognition, Kolkata, India*, pp. 341–344.

Pratihar, S. and Bhowmick, P. (2010). Vectorization of thick digital lines using Farey sequence and geometric refinement, *ICVGIP, Chennai, India*, pp. 518–525.

Pratihar, S. and Bhowmick, P. (2011). Skew correction of engineering drawings by digital-geometric analysis of Farey ranks, *International Conference on Image Information Processing (ICIIP), Shimla, India*, pp. 1–6.

Pratihar, S., Bhowmick, P., Sural, S. and Mukhopadhyay, J. (2013). Skew correction of document images by rank analysis in Farey sequence, *International Journal of Pattern Recognition and Artificial Intelligence* **27**(7): Article ID 1353004.

Ray, B.K. and Ray, K.S. (1992). An algorithm for detection of dominant points and polygonal approximation of digitized curves, *Pattern Recognition Letters* **13**(12): 849–856.

Ray, K.S. and Ray, B.K. (2013). Polygonal approximation of digital curve based on reverse engineering concept, *International Journal of Image and Graphics* **13**(4): Article ID 1350017.

Rosin, P.L. (1997). Techniques for assessing polygonal approximation of curves, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(6): 659–666.

Rosin, P.L. and West, G.A.W. (1988). Detection of circular arcs in images, *4th Alvey Vision Conference, Manchester, UK*, pp. 259–263.

Rosin, P.L. and West, G.A.W. (1995). Non-parametric segmentation of curves into various representations, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17**(12): 1140–1153.

Routledge, N. (2008). Computing Farey series, *Mathematical Gazette* **92**(523): 55–62.

Sarkar, B., Singh, L.K. and Sarkar, D. (2004). A genetic algorithm-based approach for detection of significant vertices for polygonal approximation of digital curves, *International Journal of Image and Graphics* **4**(2): 223–239.

Schroeder, M. (2006). Fractions: Continued, Egyptian and Farey, *in* M.R. Schroeder (Ed.), *Number Theory in Science and Communication,* Springer, Berlin/Heidelberg, pp. 55–86.

Singh, C., Bhatia, N. and Kaur, A. (2008). Hough transform based fast skew detection and accurate skew correction methods, *Pattern Recognition* **41**(12): 3528–3546.

Srihari, S.N. and Govindraju, V. (1989). Analysis of textual images using the Hough transform, *Machine Vision Applications* **2**(3): 141–153.

Teh, C.H. and Chin, R.T. (1989). On the detection of dominant points on digital curves, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2**(8): 859–872.

Van, T.T. and Le, T.M. (2016). Content-based image retrieval using a signature graph and a self-organizing map, *International Journal of Applied Mathematics and Computer Science* **26**(2): 423–438, DOI: 10.1515/amcs-2016-0030.

Wall, K. and Danielsson, P.-E. (1984). A fast sequential method for polygonal approximation of digitized curves, *Computer Vision Graphics and Image Processing* **28**(3): 220–227.

Wang, K., Shi, T., Liao, G. and Xia, Q. (2013). Image registration using a point-line duality based line matching method, *Journal of Visual Communication and Image Representation* **24**(5): 615–626.

Wang, L., Neumann, U. and You, S. (2009). Wide-baseline image matching using line signatures, *International Conference on Computer Vision (ICCV), Kyoto, Japan*, pp. 1311–1318.

Yan, H. (1993). Skew correction of document images using interline cross-correlation, *CVGIP: Graphical Models and Image Processing* **55**(6): 538–543.

Yin, P.-Y. (2001). Skew detection and block classification of printed documents, *Image and Vision Computing* **19**(8): 567–579.

Yin, P.Y. (2003). Ant colony search algorithms for optimal polygonal approximation of plane curves, *Pattern Recognition* **36**(8): 1783–1797.

Yin, P.Y. (2004). A discrete particle swarm algorithm for optimal polygonal approximation of digital curves, *Journal of Visual Communication and Image Representation* **15**(2): 241–260.

Yu, B. and Jain, A.K. (1996). A robust and fast skew detection algorithm for generic documents, *Pattern Recognition* **29**(10): 1599–1630.

Yuan, B. and Tan, C.L. (2007). Convex hull based skew estimation, *Pattern Recognition* **40**(2): 456–475.

Zhang, L. and Koch, R. (2013). An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency, *Journal of Visual Communication and Image Representation* **24**(7): 794–805.

Zhang, Q., Wang, Y. and Wang, L. (2015). Registration of images with affine geometric distortion based on maximally stable extremal regions and phase congruency, *Image and Vision Computing* **36**(C): 23–39.

Sanjoy Pratihar received his BTech and ME degrees from North Eastern Hill University, Shillong, India, and from Bengal Engineering and Science University, Shibpur (presently the Indian Institute of Engineering Science and Technology), India, respectively, both in computer science and engineering. He received his PhD from the Indian Institute of Technology, Kharagpur, India. Currently he is serving as an assistant professor in the Department of Computer Science and Engineering, National Institute of Technology Meghalaya, Shillong. His research interests include digital geometry, low-level image processing, document image processing, shape analysis, and intelligent human-computer interaction. He has coauthored several research papers, published in various international conference proceedings, edited volumes, and journals in these fields.

Partha Bhowmick graduated from the Indian Institute of Technology, Kharagpur, India, and received his MSc and PhD degrees from Indian Statistical Institute, Kolkata, India. He is currently an associate professor in the Department of Computer Science and Engineering, Indian Institute of Technology. His research focuses primarily on digital geometry, but he also works on algorithmic art, combinatorial image analysis, and computer graphics. He has coauthored over 100 research papers in these areas, which have been published in peer-reviewed international journals, edited volumes, and international conference proceedings. He has also co-authored one book on digital geometry, and holds four US patents.

Received: 20 October 2016
Revised: 12 March 2017
Accepted: 2 April 2017